Trabajo Fin de Máster
Máster en Ingeniería Industrial

Identificación y control no lineal del TCLab

Autor: Alberto Sánchez-Cid Bueno

Tutor: Daniel Limón Marruedo

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

Sevilla, 2023

Trabajo Fin de Máster
Máster en Ingeniería Industrial

Identificación y control no lineal del TCLab

Autor:
Alberto Sánchez-Cid Bueno

Tutor:
Daniel Limón Marruedo
Catedrático

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

Sevilla, 2023

Trabajo Fin de Máster: Identificación y control no lineal del TCLab

Autor: Alberto Sánchez-Cid Bueno
Tutor: Daniel Limón Marruedo

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Este trabajo no habría sido posible sin la ayuda de mi tutor Daniel Limón, quien me ha ayudado y guiado a lo largo de él, siempre atento y paciente conmigo. También quiero agradecer a Teodoro Álamo, catedrático del mismo departamento, que me haya concedido una tutoría para explicarme el algoritmo ADMM que se expone en este trabajo, del que él es autor.

Luego todo el apoyo que mis seres queridos me han brindado, en especial mi familia. Me han animado a acabar el trabajo cada vez que bajaba el ritmo y me distraía. Les agradezco mucho que siempre hayan estado ahí.

Alberto Sánchez-Cid Bueno
Sevilla, 2023

Resumen

Esta memoria describe el proceso de diseño de un controlador basado en un modelo no lineal, identificado de medidas reales de TCLab, un módulo de Arduino también conocido como un laboratorio de bolsillo creado con fines académicos, y un sistema termodinámico MIMO, de naturaleza no lineal.

El trabajo parte del enfoque lineal. Primero se identifica el modelo con una estructura de regresores independientes que después se convierte a espacio de estados, la expresión que utiliza un controlador MPC. Este paso del trabajo sirve, por un lado, para comprobar cómo se defiende un controlador basado en un modelo de estas características en el sistema en cuestión, aunque su verdadera finalidad es establecer la base sobre la que se desarrolla el controlador no lineal.

La no linealidad se logra cuando se introduce la lógica borrosa. La combinación de múltiples modelos lineales que identifican diferentes regiones de las salidas dan lugar a un modelo instantáneo, nuevo en cada muestra, que proporciona estimaciones que dejan de ser lineales. Este grado de complejidad adicional también afecta al diseño del controlador. Cuando antes solo existía un algoritmo que resuelve el problema de optimización a través de un proceso iterativo en cada muestra, ahora es también necesario incluir un bucle externo al algoritmo que le ayude a converger el modelo instantáneo.

El controlador MPC no lineal diseñado consigue una mayor precisión respecto del lineal, a cambio de un mayor coste computacional. Los resultados de su ensayo han sido esperanzadores, pudiéndose valorar la posibilidad de probarlo en otros sistemas no lineales de dinámica diferente.

Abstract

This memoir describes the process of designing a nonlinear-based controller, identified from real measured data from TCLab, an Arduino module also known as a pocket-size laboratory, created for academic purposes, and a MIMO thermodynamic system, a system of nonlinear nature.

The project starts from the linear approach. First, the model is identified with a structure of independent regressors, which is then converted to state space, the expression used by an MPC controller. This step of the work serves, on the one hand, to check how a controller based on such a model is defended in the system in question, although its real purpose is to establish the basis on which the nonlinear controller is developed.

Nonlinearity is achieved when fuzzy logic is introduced. The combination of multiple linear models identifying different regions of the outputs results in an instantaneous model, new in each sample, which provides estimates that are no longer linear. This additional degree of complexity also affects the controller design. Where previously there was only one algorithm that solves the optimization problem through an iterative process at each sample, it is now also necessary to include an external loop to the algorithm to help it converge the instantaneous model.

The nonlinear MPC controller designed achieves higher accuracy with respect to the linear one, in exchange for a higher computational cost. The results of the test have been encouraging, and the possibility of testing it on other nonlinear systems with different dynamics can be considered.

Índice

Resumen III
Abstract V

1 Descripción del equipo TCLab 1

2 Modelo lineal 3
2.1 Primeros ensayos 3
2.1.1 Salto en escalón 3
2.1.2 Control por relé 4
2.2 Obtención de datos 5
2.2.1 PRBS 5
2.2.2 Chirp 7
2.3 Identificación 8
2.3.1 Entrenamiento 8
2.3.2 Resultados 10

3 Control lineal 13
3.1 Espacio de estados 13
3.2 MPC lineal 14
3.3 Simulación 17
3.4 Ensayo en TCLab 19

4 Modelo no lineal 23
4.1 Lógica borrosa 23
4.2 Obtención de datos de entrenamiento 25
4.3 Identificación no lineal 25
4.4 Resultados 27

5 Control no lineal 33
5.1 Simulación 33
5.2 Ensayo en TCLab 34

6 Conclusiones 37

Apéndice A Programas 39
A.1 Modelo lineal 39

A.2 Control lineal 51
A.3 Modelo no lineal 62
A.4 Control no lineal 72

Índice de Figuras 83
Bibliografía 85

1 Descripción del equipo TCLab

En este trabajo fin de máster se desarrolla todo el proceso de diseño de un controlador no lineal, creado con el propósito de ser implementado en un sistema de esta misma naturaleza. Para ello nos servimos de TCLab (Temperature Control Lab), un equipo de experimentación de tamaño reducido y bajo coste que permite ensayar diferentes estrategias de control cómodamente en cualquier lugar con toma eléctrica, fabricado para el aprendizaje de identificación de sistemas y de diseño de controladores. Consta de dos transistores calentadores aleteados, cuyas entradas manipulables son sus potencias eléctricas, de máximas diferentes entre sí. Atornillado a cada una de ellas están los sensores de temperatura, las dos salidas controlables (en la Figura 1.1, sensor 1 a la derecha, sensor 2 a la izquierda). Además incluye una luz LED para indicar visualmente la temperatura.

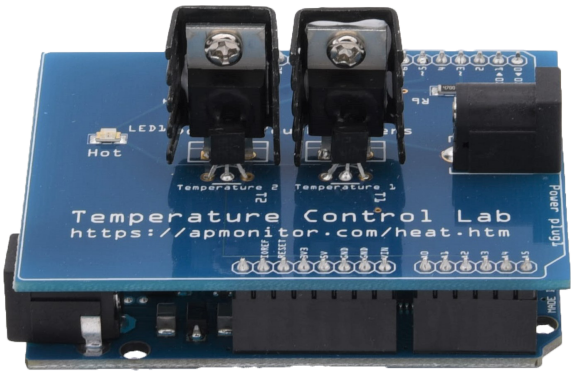

Figura 1.1 TCLab.

La proximidad entre los dos calentadores causa la correlación que complica el control individual. Las temperaturas medidas por ambos sensores se ven afectadas por las potencias disipadas, aunque en mayor grado por la del calentador parejo, tanto por cercanía como porque existe conducción.

El ejemplar utilizado para este TFM es de la Figura 1.2, un modelo Leonardo propiedad del departamento. Como se puede observar, la particularidad que tiene es que está confinado dentro de una caja protectora impresa en 3D de paredes altas, con el propósito principal de eliminar el ruido que puedan producir ráfagas de aire repentinas, aunque se quita la tapa durante los ensayos. Las conexiones requeridas son USB con un ordenador y de alimentación, que aconsejamos que sea externa, pues se genera ruido indeseado en las medidas de los sensores si ésta también se conecta al ordenador.

De la página web del fabricante [1] , APMonitor, se pueden descargar gratuitamente un archivo comprimido con programas plantilla que permiten realizar ensayos básicos como encender la luz LED o las resistencias por unos segundos. Se pueden encontrar escritos en los lenguajes de Arduino,

Figura 1.2 Ejemplar de TCLab empleado.

Python y Matlab. Para los programas escritos en los dos primeros no se requiere nada especial, se ejecutan sin más problema, mientras que para Matlab es necesario instalar un Add-on, un módulo al que accede desde la aplicación que incluye con las funciones necesarias.

De estas tres opciones se escoge Matlab por la mayor familiaridad. Partiendo de esta decisión, pasamos a presentar las funciones necesarias para interactuar con TCLab:

- tclab: Es el programa mediante el que se establece una conexión entre TCLab y Matlab; conectarlo al ordenador no es suficiente. Al igual que en las plantillas, siempre aparece al principio.
- x=t1(): Almacena la última medida de temperatura del sensor 1 en la variable x. Para el sensor 2, x=t2().
- h1(x): El calentador 1 empieza a funcionar al porcentaje de potencia x. Para el otro calentador, h2(x).

En el apéndice se puede encontrar los programas utilizados para este trabajo. Solo aquellos que comienzan por "ENSAYO" emplean estas funciones y siempre las tres.

Para terminar con la descripción, aunque por lo general se va a ignorar, se deja constancia de que existe un ligero error en las medidas puesto que a temperatura ambiente el sensor 1 siempre da una medida de en torno a 0,1 ºC superior a la del sensor 2.

2 Modelo lineal

Tomando como referencia las plantillas, en Matlab se preparan los ensayos que nos servirán para conocer mejor el comportamiento del sistema recomienda. Para estos ya es necesario tomar decisiones, siendo la más determinante el punto de operación. De forma arbitraria, tras varias pruebas preliminares para tener un primer contacto con TCLab, se determina que ambos calentadores funcionen al 50 %. Es bien sabido que la no linealidad de un sistema termodinámico radica en que un incremento de energía en él se traduce en un aumento de temperatura cada vez menor según esta sea mayor. La dominancia clara del polo lento sobre el otro lo convierte en uno de segundo orden sobreamortiguado. Para el modelo lineal, lo que nos ocupa en este capítulo, que pretende con una expresión relativamente sencilla replicar el comportamiento de TCLab, se sitúa en un punto intermedio para minimizar el error, como se detallará más adelante.

Más información que se ha extraído de la primera toma de contacto es que alcanzar el régimen permanente requiere de mucho tiempo, si los calentadores se mantienen a unas potencias constantes hasta entonces. A través de la experiencia se estima adecuado esperar una hora para considerar que ha llegado a él, aunque con veinte minutos se acerca mucho al equilibrio. También, con el uso más prolongado de TCLab a lo largo del TFM, se constata su fuerte relación con la temperatura ambiental, como cabría esperar de un sistema termodinámico excitado con calentadores de baja potencia. Con esto se quiere aclarar que un modelo lineal difícilmente será útil para todos los escenarios, no difiriendo en la respuesta temporal pero si en la magnitud: un modelo entrenado con datos de invierno no replicará bien el funcionamiento del sistema en verano.

2.1 Primeros ensayos

2.1.1 Salto en escalón

Partiendo del sistema en equilibrio, se incrementa el valor de consigna de un calentador que se mantiene en el tiempo, registrando todo el transitorio hasta alcanzar un nuevo equilibrio. Se debe realizar en torno al punto de operación aunque en este caso la diferencia en dinámica es mínima entre distintos puntos. La Figura 2.1 muestra superpuestas las medidas de dar escalones al calentador asociado de 25 a 50 % tras otro de 50 a 75 mientras el otro calentador se mantiene al 50 %.

Mediante este ensayo se obtiene la constante de tiempo (τ), una aproximación simplificada pero bastante acertada del comportamiento de un sistema de segundo orden sobreamortiguado. Hay que realizarlo tantas veces como entradas haya, Más allá de darnos un mejor conocimiento de TCLab, este dato es útil para el ensayo Chirp, cuyo fin es el de validar el modelo obtenido.

Cabe explicar que TCLab, como cualquier sistema termodinámico, no responde igual a un escalón positivo que a uno negativo. En el segundo caso la cesión de calor al ambiente está más presente, lo que causa que la respuesta sea más lenta, pero lejos de ser determinante. Para ser más fiel a la

realidad habría que crear un modelo compuesto por dos para cada sensor; sin embargo, como esta mayor complejidad apenas supone una mejoría, el modelo se reduce a una respuesta dinámicamente idéntica para los dos tipos de incrementos.

El Programa A.1 utilizado es el primero que se encuentra en el apéndice. Tiene un bucle anidado que realiza tantas iteraciones como segundos queden para el siguiente escalón dentro de otro que recorre los escalones programados en la inicialización. Los ensayos siempre incluyen una sección al final donde se actualizan las gráficas con datos del sistema en una figura. En la superior se representan las temperaturas medidas durante el ensayo y en la inferior las consignas de los actuadores.

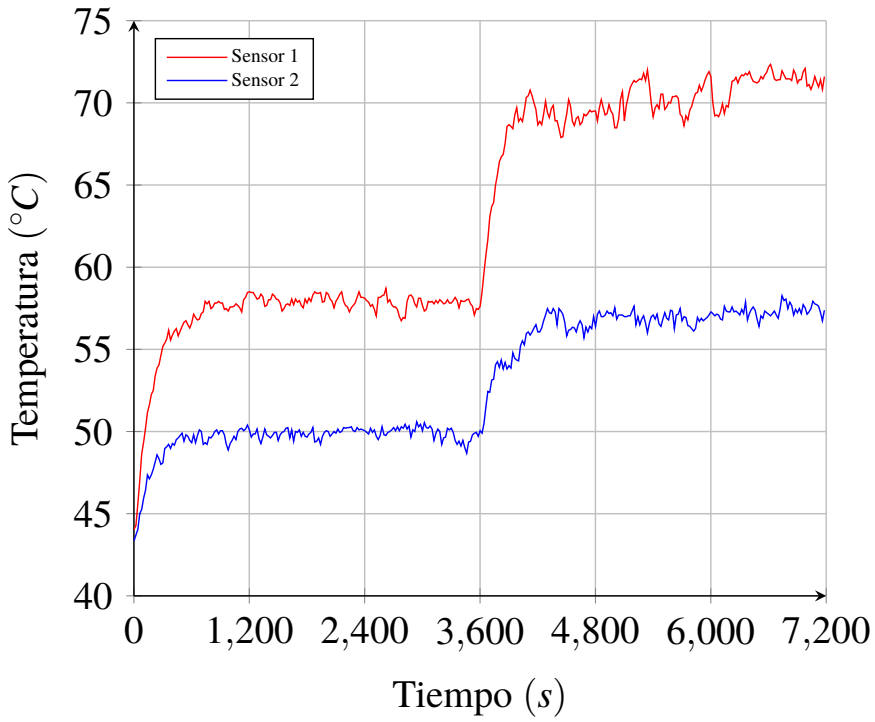

Figura 2.1 Resultados del ensayo de salto en escalón.

2.1.2 Control por relé

Alcanzado el equilibrio en el punto de operación, se calcula la media de las medidas de los últimos treinta segundos de un sensor. Ésta la toma como referencia un controlador que manipula con valores binarios la entrada pareja, es decir, cuando la temperatura medida es inferior a la referencia pone el calentador a máxima potencia (100 %); cuando es superior, lo apaga. Este tipo de control provoca una oscilación en las temperaturas en torno a la referencia, más notorio en las medidas del sensor junto al calentador manipulado, de la que se extrae el periodo crítico (T_c). Una forma sencilla de determinarlo es medir los intervalos entre picos y sacar su media.

Para mejorar los resultados, en el Programa A.2 se añade una banda de histéresis de 0,4 (0,2 a cada lado) en torno a la referencia, un valor mínimo con el reducir el impacto de lecturas ruidosas. En la Figura 2.2 aparecen los resultados, satisfactorios para el sensor 1, aunque no tanto para el sensor 2 debido al ruido, motivo por el cual se decide finalmente mostrar, para ilustrar este fenómeno. Aún con este inconveniente, ambos ensayos muestran un T_c aproximado de 65 s, así que se toma como un único valor.

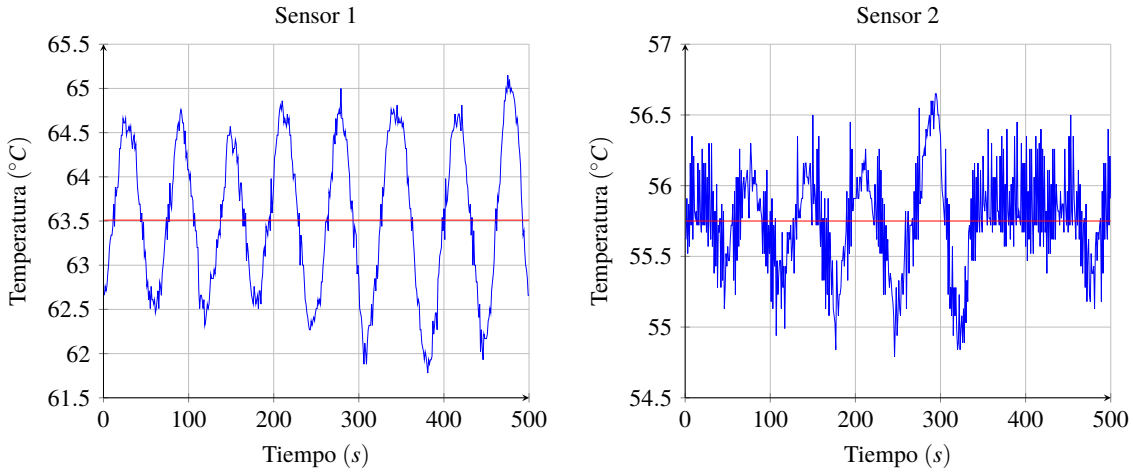

Figura 2.2 Resultados del ensayo de control en relé.

2.2 Obtención de datos

Para mayor claridad, se repasan los datos numéricos obtenidos hasta ahora:

$$\tau = 180\,s,\ T_c = 65\,s.$$

Siguiendo la teoría de Niquist, la frecuencia de muestreo debe ser al menos el doble de la máxima, o dicho de otro modo, el tiempo de muestreo debe ser inferior a la mitad del tiempo de oscilación. No obstante, por recomendación general, incluido APMonitor:

$$T_m = T_c/5 = 13\,s$$

Así queda fijado el tiempo que transcurrirá entre cambios en las entradas y podemos pasar a la obtención de datos para el entrenamiento del modelo y su validación posterior. Las medidas de los sensores se seguirán tomando en cada segundo para después filtrar los resultados.

2.2.1 PRBS

El ensayo PRBS (PseudoRandom Binary Sequence), el Programa A.3, consiste en excitar una de las entradas con señales binarias mientras la otra se mantiene en el valor que se le asignó para el punto de operación, por lo que se parte de él. Se realiza un único ensayo que recoja la respuesta de ambos calentadores para que no haya una divergencia en las temperaturas en el punto de operación.

La señal no varía en una secuencia binaria entre 0 o 100, sino entre 20 o 80, elegido arbitrariamente con el objetivo de que el modelo no quede muy influenciado por la no linealidad del sistema. Con el tiempo de conmutación mínimo T_{sw} y el número de bits n se crea la señal de unos y ceros que luego se adapta en amplitud a la antes descrita.

$$T_{sw} = T_c,\ 2^n \geq \frac{\omega_{max}}{\omega_{min}} \rightarrow n = 8.$$

$$\omega_{min} = \frac{0{,}1}{\tau} = 5{,}5510^{-4}\,rad/s,\ \omega_{max} = \frac{2\pi}{T_{sw}} = 0{,}0967\,rad/s.$$

Los resultados del ensayo se ilustran en la Figura 2.3 junto con las consignas en la Figura 2.4.

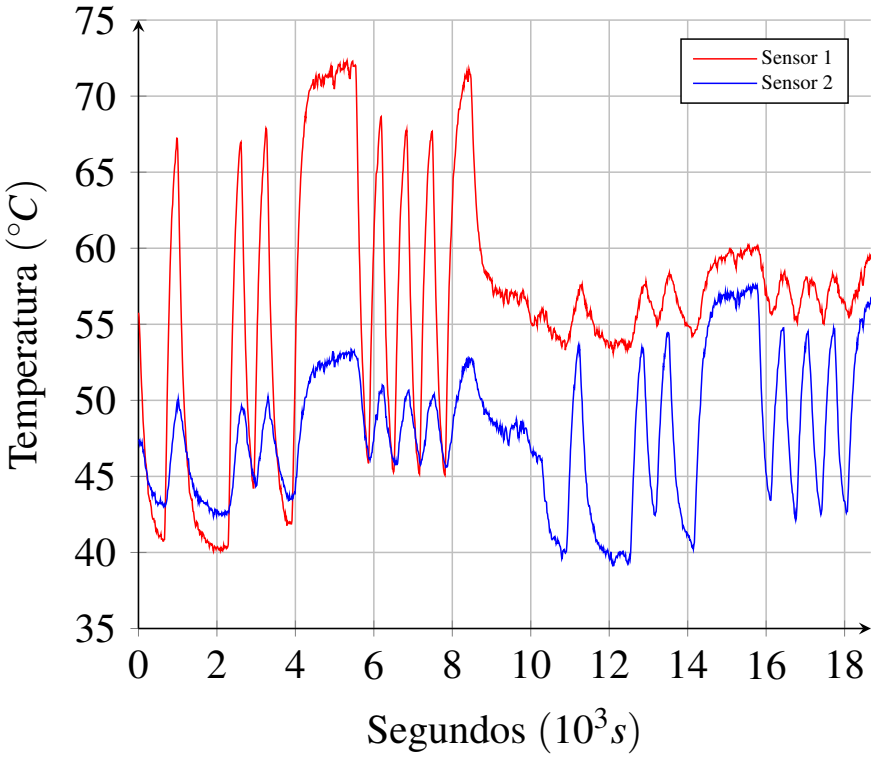

Figura 2.3 Salidas del ensayo PRBS.

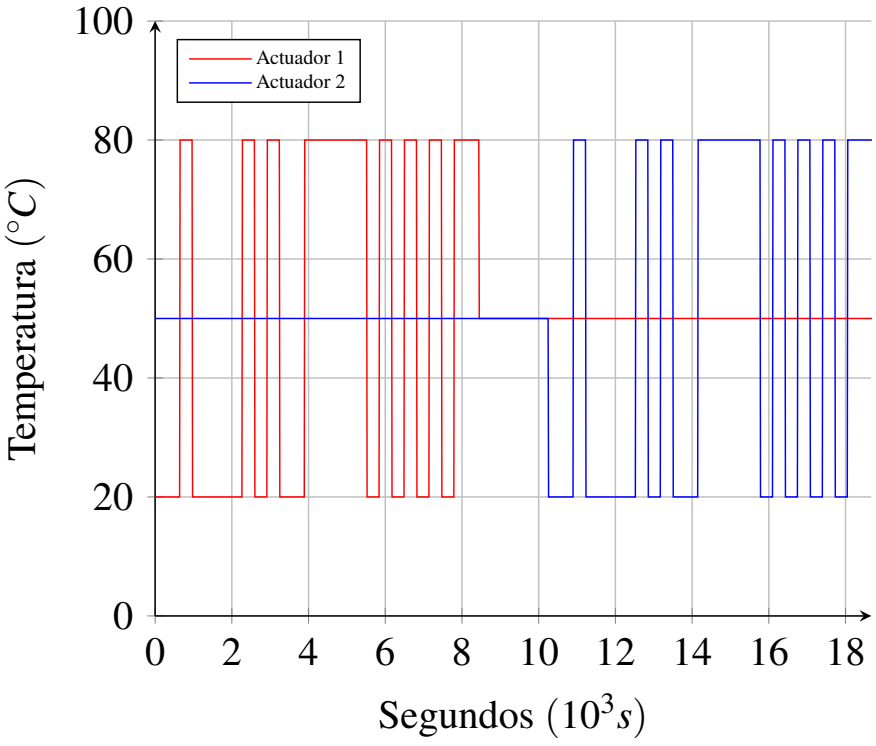

Figura 2.4 Entradas del ensayo PRBS.

2.2.2 Chirp

La entrada es manipulada por una senoide cuya frecuencia aumenta a una razón determinada con el tiempo, en este caso lineal, desde la frecuencia mínima hasta la máxima durante un tiempo T_{chirp}. Su expresión matemática es esta:

$$\omega(t) = \frac{\omega_{max} - \omega_{min}}{T_{chirp}} * t + \omega_{min},\ t \in [0, T_{chirp}]$$

Donde T_{chirp}, la duración del ensayo por actuador.

$$T_{chirp} = 20\pi\tau = 11310\,s$$

En el Programa A.4 la amplitud de la senoide es de 30, tal que se mueve entre el 20 y el 80 %, al igual que el ensayo PRBS. Las consignas se introducen cada segundo en lugar de cada tiempo de muestreo debido a la deformación que sufre la onda a frecuencias altas.

Este ensayo resalta uno de los mayores obstáculos a la hora de controlar TCLab: la deriva. Se nota sobre todo en las frecuencias más altas, donde también los sensores parece que detecta mucho ruido cuando se ensaya el calentador lejano a este. La Figura 2.5 muestra un comportamiento complicado que puede llevar a pensar que este ensayo no sea el más indicado para realizar la validación de un modelo de este sistema. No obstante, se decide seguir optando por este método de validación. A frecuencias bajas los datos sí son aceptables y es interesante también comprobar cuál es el comportamiento del modelo a frecuencias altas.

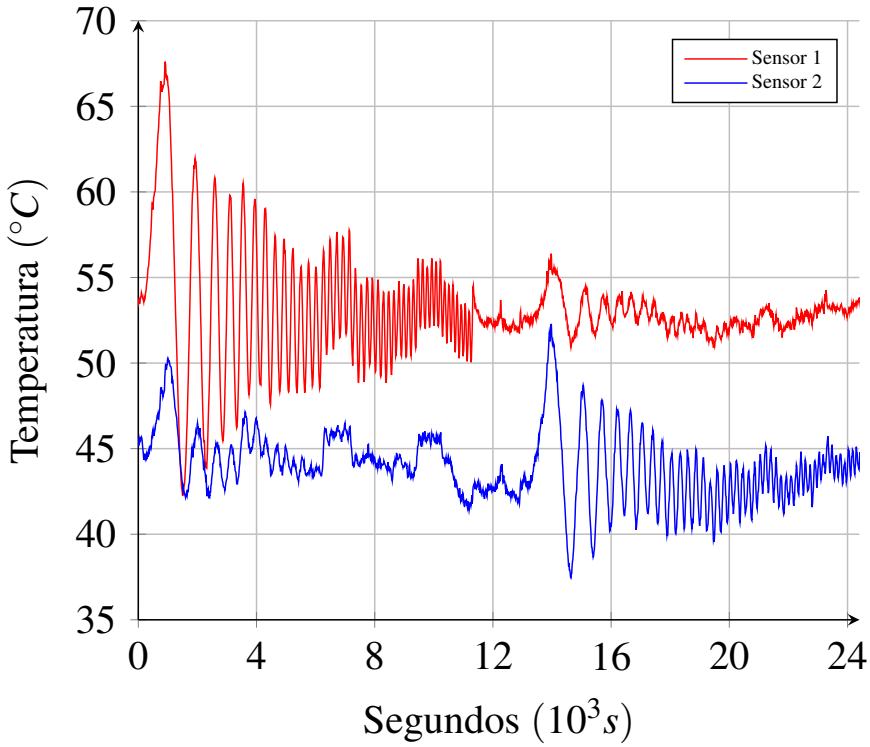

Figura 2.5 Salidas del ensayo Chirp.

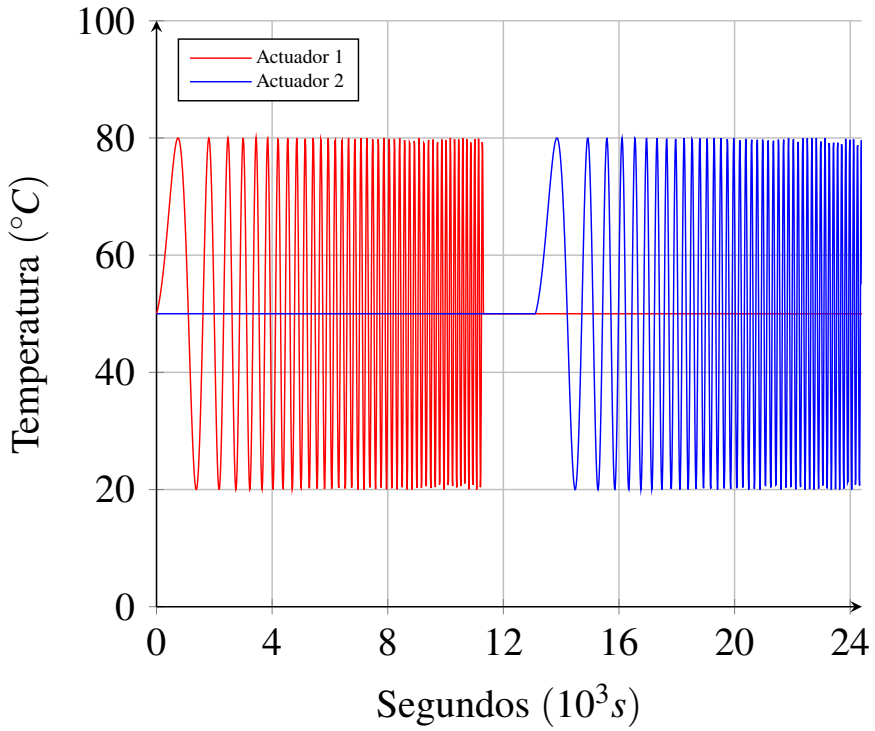

Figura 2.6 Entradas del ensayo Chirp.

2.3 Identificación

Para la creación del modelo hay que tomar nuevas decisiones como el método de identificación y sus parámetros, ajustados en función de cómo queremos que el algoritmo de identificación enfoque el proceso: penalización del error, ponderación de los parámetros del modelo, etc. Una vez decidido se preparan los datos de entrenamiento y la validación al formato de este.

2.3.1 Entrenamiento

El Programa A.5 lleva a cabo todo el proceso de identificación y validación, empezando por el tratamiento de los datos brutos del ensayo PRBS. El tratamiento comienza con un filtrado de los datos, sin incluir el escalón inicial para alcanzar el equilibrio. Como el ensayo se han registrado las medidas cada segundo, de él resultado se toman las muestras cuyo tiempo sea múltiplo de T_m y se les resta los valores del punto de operación. Así desplazamos las medidas a un nuevo eje de coordenadas local ubicado a la altura de éste, lo que también significa que se crea un eje para cada sensor.

Se decide crear dos regresores independientes para las salidas, que en conjunto modelan TCLab. Se exploran múltiples configuraciones. Los regresores son vectores cuya longitud queda definida por los valores enteros de los parámetros ny para la salida en cuestión, $nu1$ para el primer calentador y $nu2$ para el segundo. Determinan el número máximo de valores pasados de cada uno, empezando desde el actual, que se necesitan para la regresión. En este caso no es necesario incorporar a los regresores parámetros ligados a otras salidas pues el estado de una no afecta a la otra y viceversa. Los tres grupos de coeficientes se ordenan primero por grupo (en el orden mencionado) y luego por novedad, siendo el primero el coeficiente para el dato más reciente y el último el más antiguo que se contempla.

El programa empieza por crear el regresor más básico, con un solo coeficiente por grupo hasta el más complejo establecido, de una longitud de $ny + nu1 + nu2$ elementos, pasando por todas las

combinaciones intermedias, creando $ny * nu1 * nu2$ regresores por salida. Para la creación de cada configuración se recurre a un triple bucle anidado con índices i, j, k que indican el número de coeficientes por grupo para el regresor de la iteración. A continuación se ilustra el aspecto genérico del regresor.

$$\hat{y}_t = r_t\theta = \begin{bmatrix} y_{t-1}, & \cdots, & y_{t-i}, & u^1_{t-1}, & \cdots, & u^1_{t-j}, & u^2_{t-1}, & \cdots, & u^2_{t-k} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_i \\ b_1 \\ \vdots \\ b_j \\ c_1 \\ \vdots \\ c_k \end{bmatrix} \rightarrow \hat{Y} = R\theta$$

Dentro del tercer bucle primero se crea la matriz de datos R con la que se entrena el regresor. En la expresión anterior, r_t es la fila de R que se utiliza para la estimación del instante t.

Luego se anida un cuarto bucle para las distintas configuraciones la matriz P. Aparece en el primer término, de la Ecuación (4.3), que corresponde a la formulación del Ridge Regression, expresado en formato matricial.

$$\theta_N = arg \min_{\theta \in \mathbb{R}^{n_\theta}} \frac{1}{2}\theta^T P\theta + \|Y - R\theta\|^2_D$$

Se trata de un problema explícito, de igual número de ecuaciones que de incógnitas. Se deriva respecto de la variable, el regresor θ. El resultado se iguala a cero y se reorganizan los términos para determinarla.

La matriz diagonal D pondera los errores cometidos por el regresor individualmente, en función de algún interés. Aquí, como no hay motivo de discriminación entre errores, la diagonal se rellena con unos. Con el modelo no lineal esto cambia, como se verá más adelante.

Retomando el hilo del programa principal, en el cuarto bucle anidado se construye la matriz diagonal P a partir de vectores enteros de longitudes i, j y k creados por la Función A.7. Luego se juntan en uno solo y se diagonaliza, es decir, se crea una matriz cuadrada de ceros con el arreglo en la diagonal. Al final, de todos los regresores que se identifican dentro de este bucle con diferentes P solo nos quedaremos con uno. Se les proporciona la misma excitación en las entradas del ensayo PRBS y se calcula su secuencia de estimaciones. El que cometa el menor error cuadrático medio con respecto las medidas es el que se guarda en la celda $Thetareg$, celda de registro de todos los regresores de la salida, en la coordenada (i,j,k), y su error en $Errcuad$ en esta misma coordenada.

Una vez obtenidos todos los regresores, se les somete a validación: se realizan con ellos nuevas estimaciones con las consignas en la entrada del ensayo Chirp y se calculan los errores cuadráticos medios aquí cometidos. De los datos del ensayo solo empleamos la respuesta de baja frecuencia, es decir, con la primera mitad del ensayo de cada calentador. Las medidas de alta frecuencia muestran un comportamiento tan pobre y difícil de replicar por un modelo que en cualquier caso no habría resultados muy positivos, por lo que se recortan. Los nuevos errores calculados se almacenan en la matriz $Errval$.

El último paso que queda de la identificación es la selección de un regresor para cada salida según algún criterio. En este trabajo se toman los errores cuadráticos medios de tanto el entrenamiento como la validación, se suman, haciendo así uno valga tanto como el otro, y la suma se pondera en función del número de coeficientes, de la longitud del regresor, o dicho de otra manera, de su complejidad. Se opta por multiplicar todos los errores por la raíz décima de esta longitud, una

forma de destacar regresores más sencillos y con estimaciones bastante acertadas sobre otros más complicados y que cometen un error ligeramente menor.

2.3.2 Resultados

Explicado el proceso de identificación, definimos los valores de los parámetros que se han presentado a lo largo de esta sección.

ny	$nu1$	$nu2$
4	3	3

Bajo el criterio de selección descrito, se exponen en tabla por sensor, de mejor a peor, de izquierda a derecha, las características y resultados de los cinco regresores que menor error cuadrático medio ponderado cometen. En este contexto, se entiende como ny, $nu1$ y $nu2$ el número de coeficientes que tiene el regresor dedicado a la salida del sensor y a los calentadores.

Sensor 1:

	1º	2º	3º	4º	5º
ny	2	2	2	3	1
$nu1$	1	2	1	1	3
$nu2$	3	3	2	3	3
E.c.m.p. Ent.	0.5958	0.5448	0.6795	0.5126	0.5749
E.c.m.p. Val.	1.3330	1.4398	1.3286	1.5275	1.4837
E.c.m.p. Total	1.9289	1.9846	2.0081	2.0401	2.0587

Sensor 2:

	1º	2º	3º	4º	5º
ny	3	4	2	1	1
$nu1$	3	3	3	3	3
$nu2$	1	1	1	1	2
E.c.m.p. Ent.	0.8986	0.9016	0.9911	1.0805	1.0890
E.c.m.p. Val.	1.7245	1.7756	1.7016	1.7006	1.7668
E.c.m.p. Total.	2.6231	2.6772	2.6926	2.7811	2.8559

De ambas tablas se sacan las mismas conclusiones: para el calentador parejo al sensor es suficiente un coeficiente mientras que para el otro se requiere un mayor número para captar bien su efecto. Añadir un retraso en las consignas del calentador lejano, para el tiempo de muestreo considerado, podría ayudar a conseguir mejores regresores. También se comprueba que el error en la segunda tabla es mayor, tanto en el entrenamiento como en la validación, debido seguramente al ruido en este sensor.

A continuación se muestra el contenido de los mejores regresores para cada sensor, el modelo de TCLab. Para mayor claridad, se separa con líneas verticales los grupos de coeficientes de ny, $nu1$ y $nu2$.

* Sensor 1:

$$\theta_1 = \begin{bmatrix} 1{,}4310 & -0{,}4651 | & 0{,}0185 | & -0{,}0001 & -0{,}0012 & 0{,}0037 \end{bmatrix}^T$$

* Sensor 2:

$$\theta_2 = \begin{bmatrix} 1{,}0918 & 0{,}0911 & -0{,}2190 | & 0{,}0003 & -0{,}0017 & 0{,}0063 | & 0{,}0112 \end{bmatrix}^T$$

En las Figuras 2.7 y 2.8 se ilustran junto a las medidas de los ensayos de entrenamiento. Es fácil comprobar que los regresores adaptan bien cambios mientras y validación las estimaciones de los regresores en ambos escenarios.

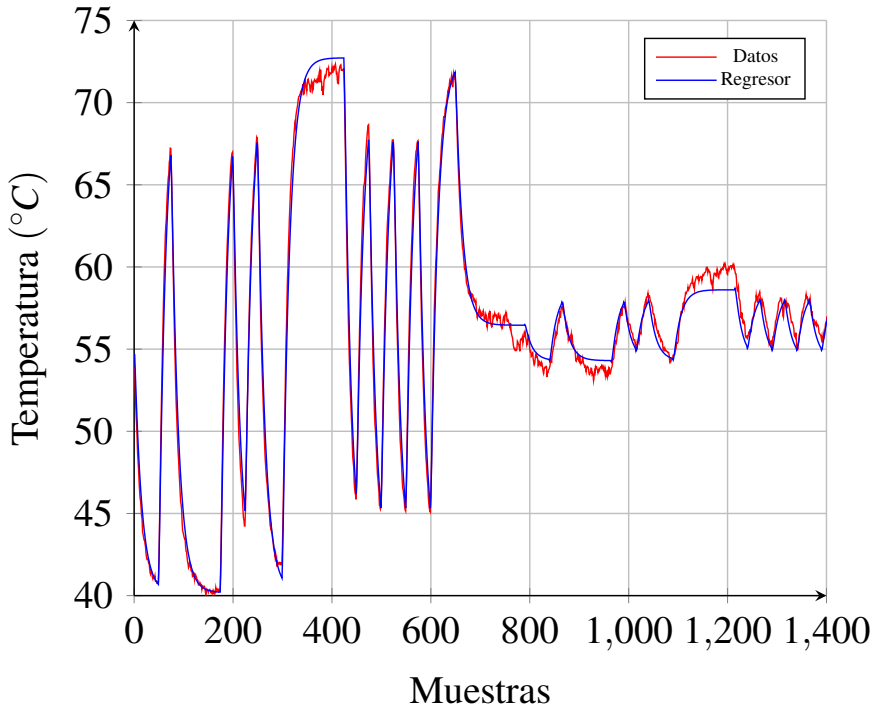

Figura 2.7 Datos de entrenamiento para el sensor 1 y estimación del modelo.

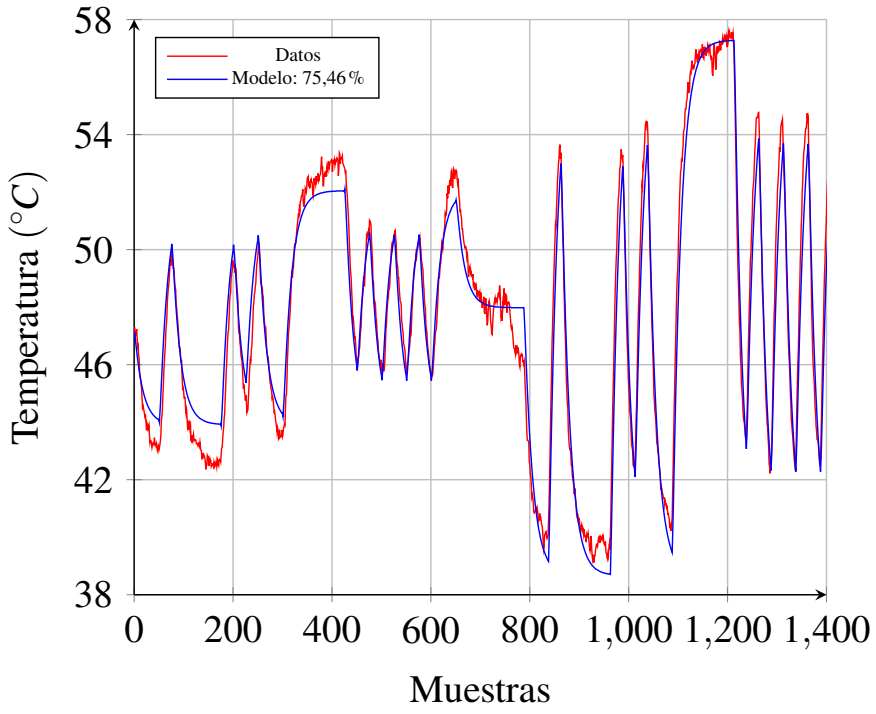

Figura 2.8 Datos de entrenamiento para el sensor 2 y estimación del modelo.

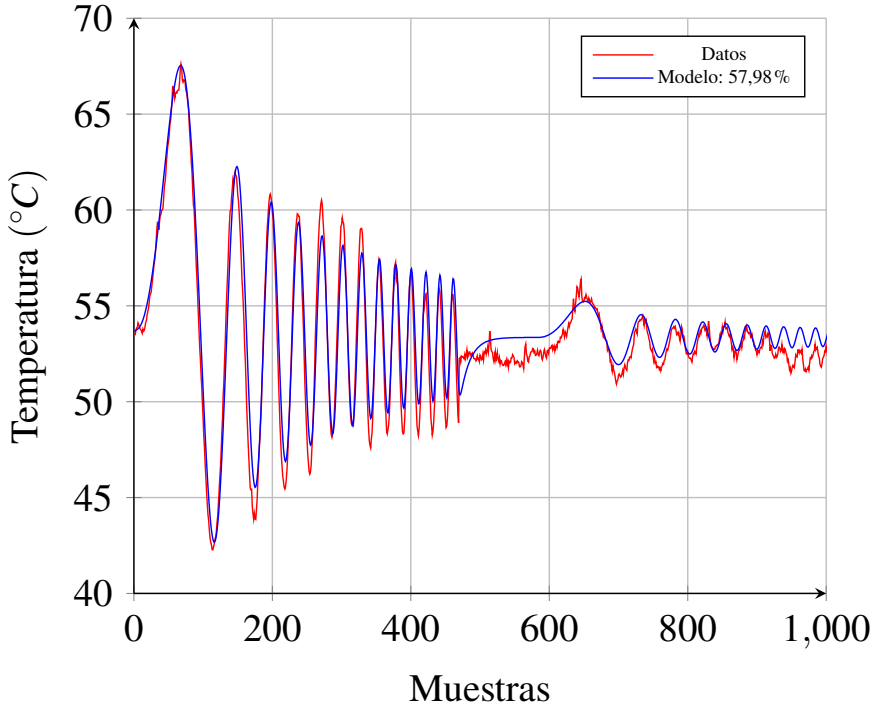

Figura 2.9 Datos de validación para el sensor 1 y estimación del modelo.

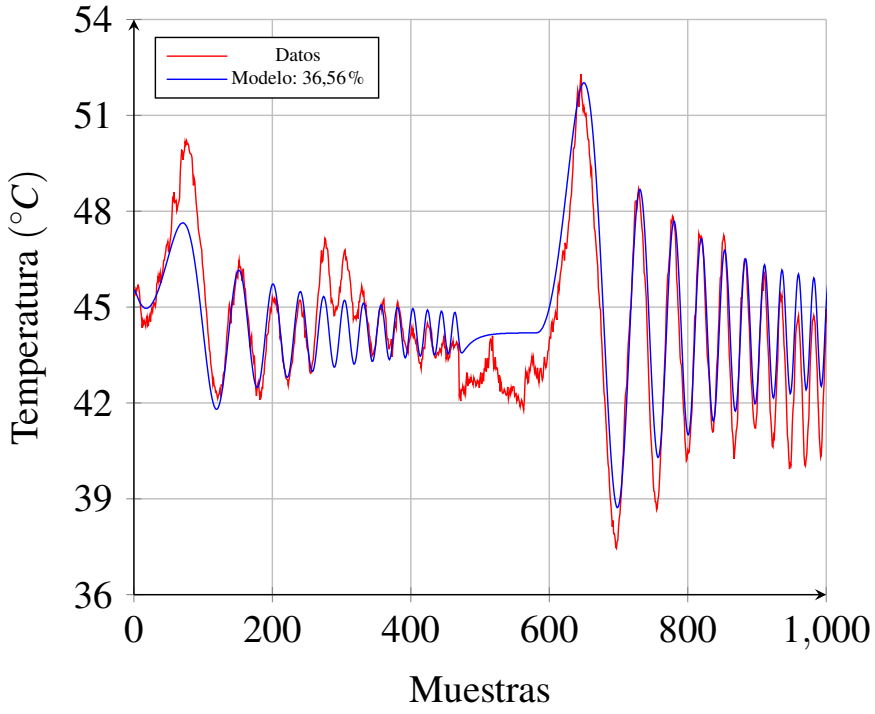

Figura 2.10 Datos de validación para el sensor 2 y estimación del modelo.

3 Control lineal

Los regresores seleccionados se unifican en un modelo de espacio de estados, la expresión habitual que se emplea para un controlador MPC (Model Predictive Control).

3.1 Espacio de estados

Se presenta el modelo conformado por los regresores en formulación espacio de estados. El vector de estados x es un registro dinámico que proyecta la evolución del sistema, con los estados actuales x_t coincidiendo con las últimas medidas (o estimaciones). Está primero ordenados por antigüedad, luego por número de sensor.

$$\begin{aligned} x_{t+1|t} &= A_{nx*nx}x_t + B_{nx*ns}u_t \\ y_t &= C_{ns*nx}x_t + D_{ns*ns}u_t \end{aligned}$$

$$x_t = \left[x^1_{t-n+1},\ \ x^2_{t-n+1},\ \ \cdots,\ \ x^1_{t-1},\ \ x^2_{t-1},\ \ x^1_t,\ \ x^2_t\right]^T,\ u_t = \left[u^1_t,\ \ u^2_t\right]^T.$$

Los nuevos datos *nx* y *ns* son las longitudes de los vectores de estado y entrada (y salida, problema 2x2), respectivamente. Otro dato nuevo es *n*, el máximo de *ny*, *nu*1 y *nu*2 de todos los regresores, en este caso 3. Este sirve para que tengan la misma longitud todos los grupos de coeficientes. Aquellos grupos de coeficientes que no lleguen a tener esta longitud se rellenen con ceros hasta que sea así.

$$\begin{aligned} \theta_1 &= \left[1{,}4310 \quad -0{,}4651 \quad 0 \right| \quad 0{,}0185 \quad 0 \quad 0 \left| \quad -0{,}0001 \quad -0{,}0012 \quad 0{,}0037\right]^T \\ \theta_2 &= \left[1{,}0918 \quad 0{,}0911 \quad -0{,}2190 \right| \quad 0{,}0003 \quad -0{,}0017 \quad 0{,}0063 \left| \quad 0{,}0112 \quad 0 \quad 0\right]^T \end{aligned}$$

Otro aspecto importante a destacar en la conversión del modelo es la ausencia de consignas pasadas en el vector u. Esto se logra con formulación canónica observable del modelo de espacio de estados, en la que el vector de estado se “observa" de forma directa de las últimas salidas y entradas. Se disponen los coeficientes de los regresores a lo largo de las dos últimas columnas A y de toda B atendiendo al orden interno de estos y de los vectores. Como no hay términos cruzados en los regresores, todos estos elementos de A son ceros. Estas son las matrices resultantes.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.2190 \\ 1 & 0 & 0 & 0 & -0.4651 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0.0911 \\ 0 & 0 & 1 & 0 & 1.4310 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1.0918 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0.0037 \\ 0.0063 & 0 \\ 0 & -0.0012 \\ -0.0017 & 0 \\ 0.0185 & -0.0001 \\ 0.0003 & 0.0112 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

3.2 MPC lineal

Primero presentamos la función objetivo, aquella cuyo resultado se busca minimizar con las secuencias de salidas a predecir y de consignas futuras a obtener. Persigue en cada muestra que el sistema evolucione en el tiempo hacia la situación marcada por las referencias, x_r y u_r.

$$\begin{aligned} &\min_{x,u} V_{NP}(y,u) = \sum_{i=0}^{NP} \|x(i) - x_r\|_Q^2 + \|u(i) - u_r\|_R^2 + \|x(NP) - x_r\|_P^2 \\ &s.a.\ x(0) = x \\ &\quad x(j+1|j) = Ax(j) + Bu(j) + d(j), j \in \mathbf{I}_0^{NP-1} \\ &\quad u(NC+j-1) = u(NC-1), j \in \mathbf{I}_0^{NP-NC}, NC \in \mathbf{I}_1^{NP} \\ &\quad A_y y \leq b_y \\ &\quad A_t x(NP) \leq b_t \\ &\quad A_u u \leq b_u \end{aligned}$$

En obtener un resultado lo más próximo al óptimo tiene una importancia crítica una configuración adecuada de las matrices que ponderan el error en la referencia del estado (Q), en la entrada (R) y del último estado predicho (P). La última salida predicha se define con el horizonte de predicción (NP), otro parámetro que a cambio de mayor coste computacional simula la respuesta del sistema real y otorga al algoritmo de resolución una mejor visión de las consecuencias de sus previsiones, y por tanto soluciones más acertadas, hasta cierto punto. Las restricciones a las que está sujeta la función objetivo se describen a continuación para su fácil compresión:

1. Definición del estado inicial, el estado en el que se encuentra en ese instante el sistema.
2. Predicciones basadas en el modelo hasta NP muestras futuras, incluyendo perturbaciones medibles d si conociesen. Como no es así, se trata de un vector de elementos nulos.
3. Las consignas más allá del horizonte de control (NC) toman el valor calculado para esta muestra, que debe ser mayor o igual a 1 y nunca mayor que NP.
4. Restricciones de caja en el estado. Limitaciones físicas del sistema.
5. Restricción terminal del estado. Impone que la última predicción realizada se encuentre dentro de un espacio del estado acotado que lo acerque a la referencia.
6. Restricciones de caja en las entradas. Limitaciones físicas de los actuadores.

Podemos reescribir la función objetivo en lo que se conoce como la aproximación simultánea. Comienza por la definición de los vectores z y z_r.

$$\begin{aligned} z &= \begin{bmatrix} x(0), & u(0), & x(1), & u(1), & \cdots, & x(NP-1), & u(NP-1), & x(NP) \end{bmatrix}^T, \\ z_r &= \begin{bmatrix} x_r, & u_r, & x_r, & u_r, & \cdots, & x_r, & u_r, & x_r \end{bmatrix}^T. \end{aligned}$$

Por tanto, la nueva formulación del MPC da lugar a una expresión más compacta dependiente únicamente de z y con una sola restricción de igualdad y otra de caja.

$$\begin{aligned} &\min_z V_{NP}(z) = z^T H z + h^T(z_r) z + cte \\ &s.a.\ Fz = f(x,d) \\ &\quad Gz \leq g(z_r) \end{aligned}$$

La supermatriz diagonal H alterna entre Q y R hasta NP veces y termina con P. El vector fila h es el producto . Mientras la primera es constante, la segunda cambiará de la referencia.

$$H = 2 * diag\left(\left[Q,R,Q,R,\cdots,Q,R,P\right]\right),$$
$$h = -2 * z_r^T H = \left[x_r^T Q, u_r^T R, \cdots, x_r^T Q, u_r^T R, x_r^T P\right],$$

La supermatriz F se compone de las tres primeras restricciones, a las que se le dedica un modulo a cada una, ya que así es más fácil crearla para cada configuración que se explore.

$$F = \begin{bmatrix} Finic \\ Fmod \\ Fnc \end{bmatrix}, \quad f = \begin{bmatrix} x(0) \\ d \\ 0 \end{bmatrix}.$$

El primer modulo ($Finic$) es simplemente una matriz identidad de tamaño nx seguida de una matriz de ceros. La suma del número de columnas de ambas es igual a la longitud de z.

$$Finic = \begin{bmatrix} I_{nx} & 0 & \cdots & 0 \end{bmatrix}$$

El segundo módulo, $Fmod$, contiene el modelo. Lo repite en cada fila de la supermatriz desplazado una muestra en el futuro cada vez.

$$Fmod = \begin{bmatrix} A & B & -I_{nx} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & A & B & -I_{nx} & \cdots & 0 & 0 & 0 \\ & & \vdots & & & \ddots & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \cdots & A & B & -I_{nx} \end{bmatrix}$$

Por último, el módulo Fnc se compone una matriz parecida a la C del modelo, solo que con nx columnas en lugar de ns, a la que se nombra NCp, que en lugar de multiplicar 1 por el estado actual es por la consigna para este mismo instante.

$$Fnc = \begin{bmatrix} 0 & \cdots & -NCp & NCp & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & -NCp & 0 & NCp & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & & \vdots & & \ddots & & \vdots & \\ 0 & \cdots & -NCp & 0 & 0 & \cdots & 0 & NCp & 0 \end{bmatrix}, \quad NCp = \begin{bmatrix} 0 & \cdots & 0 & 0 & I_{ns} \end{bmatrix}.$$

Sin incluir este último módulo, la diferencia entre la longitud de z (incógnitas) y filas de F (ecuaciones) nos deja $NP * ny$ variables libres. Incluyéndolo desciende a $NC * ny$.

G se define como una matriz identidad. Se divide g en $gmin$, con las restricciones inferiores y $gmax$ con las superiores.

$$gmin \leq Gz \leq gmax \rightarrow max(gmin, min(z, gmax))$$

Las restricciones de los actuadores son fáciles de establecer; son conocidas y solo precisan de restarles las consignas para el punto de operación. En cambio las limitaciones de los estados no están definidas, dependen fuertemente de la temperatura ambiente, por lo que se establecen unos que no debería nunca alcanzar en condiciones normales máximos, que evite la consideración de predicciones absurdas.

Por otro lado quedan las restricciones terminales, definidas como un conjunto invariante local en torno al punto de equilibrio. Se toma la envolvente de las posibles trayectorias del estado y se confina la predicción NP en un espacio en el que el sistema debe encontrarse en NP muestras. Este espacio lo conforman diferentes intervalos para cada salida, determinados en torno a las referencias como las distancias de las últimas medidas a las mismas multiplicadas por un factor λ, un vector

que refleja la dinámica del sistema. Se calcula como la suma de los coeficientes de los regresores asociados a cada salida elevado a NP, que en ausencia de cambios en las entradas, caracterizan la rapidez de la respuesta, y por tanto el tamaño de los intervalos, de la capacidad de contracción del sistema.

$$x_r - \lambda \leq x(NP) \leq x_r + \lambda$$
$$\lambda = \left[(\sum_{i=1}^{nx1} \theta_1(i))^{NP}, \quad (\sum_{i=1}^{nx2} \theta_2(i))^{NP}\right] |(x_r - x(0))|$$

Finalmente pasamos a plantear el algoritmo que se va a utilizar para la resolución de la función objetivo del MPC en cada tiempo de muestreo: ADMM (Alternating Direction Method of Multipliers). Se emplea en problemas en los que es fácil dar con una solución, en nuestro caso de z, que cumpla con la restricción de igualdad o la de caja pero no tanto una que satisfaga ambas.

Es sencillo de programar, como se puede comprobar al echar un vistazo el Programa A.9, y de adaptar a la función objetivo definida. Consiste en llegar al valor óptimo de la función objetivo cuando las variables z y v, su dual, cumplan la nueva restricción de igualdad. Para poder aplicar el algoritmo, la función V_{NP} se separa en dos, cada una dependiente de una variable diferente, $\psi(z)$, convexa separable, y $Q(v)$, cuadrática convexa.

$$\begin{aligned} \min_{z \in \mathbf{Z}, v \in \mathbf{R}} \ & \psi(z) + Q(v) \\ s.a. \ & Fv = f \\ & Gz \leq g \\ & v = z \end{aligned}$$

La nueva restricción se dualiza, es decir, se introduce en la función objetivo como una diferencia de las dos partes de la igualdad multiplicada por una nueva variable dual λ, con lo que z y v se desacoplan. No obstante, con el fin de reducir en número de iteraciones del ADMM, se introduce la norma cuadrática de la misma diferencia junto a otra variable dual ρ, introduciendo términos que multiplican z y v que impiden el desacople.

$$\begin{aligned} \min_{z \in \mathbf{Z}, v \in \mathbf{R}} \ & \psi(z) + Q(v) + \lambda^T (z - v) + \frac{\rho}{2} \|z - v\|_2^2 \\ s.a. \ & Fv = f \\ & Gz \leq g \end{aligned}$$

En cada iteración del ADMM, se calcula una de las variables dejando fijo el valor de la otra, a consecuencia de no separar la función objetivo en dos problemas diferentes. Con esta idea presente, se desarrolla el ciclo que debe repetir el algoritmo hasta que satisfacer la condición de finalización.

1. $v_{k+1} = arg \min_{v \in \mathbf{V}} \ Q(v_k) - \lambda_k^T v_k + \frac{\rho}{2} \|z_k - v_k\|_2^2$
2. $z_{k+1} = arg \min_{z \in \mathbf{Z}} \ \psi(z_k) + \lambda_k^T z_k + \frac{\rho}{2} \|z_k - v_{k+1}\|_2^2$
3. $\lambda_{k+1} = \lambda_k + \rho (z_{k+1} - v_{k+1})$
4. Condición de finalización: máx $(|z_{k+1} - z_k|, |z_{k+1} - v_{k+1}|) \leq tol, \ k <= MaxIter$

Queda definir las funciones $Q(v)$ y $\psi(z)$. La primera se convierte, en su integridad, en $V_N P$. La primera línea de la iteración, dualizando la restricción, expandiendo los términos y desechando

aquellos no dependan de v, tendría este aspecto.

$$v_{k+1} = arg\min_{v\in\mathbf{V}} \frac{1}{2} v_k^T (H+\rho I) v_k + v_k^T (h^T - \lambda_k - \rho z_k)$$
$$s.a.\ Fv = f$$

Creamos un sistema de dos ecuaciones al aplicar las condiciones de KKT (Karush-Kuhn-Tucker) sobre con esta expresión.

$$\begin{bmatrix} H+\rho I & F^T \\ F & 0 \end{bmatrix} \begin{bmatrix} v_{k+1} \\ w \end{bmatrix} = \begin{bmatrix} \lambda_k + \rho z_k - h^T \\ f \end{bmatrix}$$

La función $\psi(z)$, como se puede intuir por la definición de $Q(v)$, es igual a cero, la expresión más sencilla. El cálculo de z es naturalmente más fácil., cuya minimización se reduce a derivarla.

$$z_{k+1} = arg\min_{v\in\mathbf{V}} \frac{1}{2}\rho z_k^T z_k + z_k^T(\lambda_k - \rho v_{k+1})$$
$$s.a.\ gmin \leq z \leq gmax$$

Se minimiza obteniendo su derivada, que es inmediata, igualar a cero y despejar. Luego solo hay que calcular para los elementos de z, desacoplables debido a que la variable a minimizar es un vector y el hessiano diagonal (una matriz identidad), el óptimo sin restricciones saturado.

$$z_{k+1}(i) = gmin(i) \leq v_{k+1}(i) - \lambda_{k+1}(i)/\rho \leq gmax(i), i \in \mathbf{I}_1^{NP(nx+nu)+nx}$$

Se actualiza el valor de λ como se muestra en la explicación del algoritmo, cuya misión es favorecer la aproximación entre z y v. Al final de la iteración se comprueba si se cumple la condición de finalización: si z y v han convergido a un valor, y si z no ha cambiado desde la iteración anterior, ambas dentro de la tolerancia tol. Se incrementa en uno el valor de k en caso negativo y se repite el proceso entero; en caso afirmativo, o de que se haya alcanzado un número máximo de iteraciones ($MaxIter$), obtenemos una z que contendrá, además de los estados predichos, las consignas para presente y el futuro, aunque lo más apropiado y extendido es solo emplear las primeras y volver a calcular las demás consignas en los próximos tiempos de muestreo.

3.3 Simulación

En el Programa A.8 preparamos el entorno de simulación. Primero carga los datos del modelo, los regresores y la matriz *dn* con información de estos. Luego hay una sección en la que el usuario define los valores de los parámetros presentados a lo largo de la presentación del MPC.

NPlim	*rho*	*tol*	*MaxIter*
5	1	1e-3	20

El nuevo parámetro, *NPlim*, define cuántos horizontes de predicción se van a probar en la simulación, desde uno al valor de este, con todos los horizontes de control posibles. Con *NPlim* igual a 1 solo se realiza la simulación para *NP* igual a 1 y *NC* a 1 también. Si es 3, se prueba a demás las dos combinaciones de *NP* igual a 2 (*NC* a 1 y a 2) y las tres de *NP* a 3, con un total de seis combinaciones a simular.

El principal objetivo en este punto del TFM es diseñar el MPC más adecuado para el control de TCLab. Junto con las posibles parejas de *NP* y *NC*, se prueban combinaciones de Q, R y P, matrices

diagonales de valor único. Se consideran los siguientes, para las tres matrices:

$$\begin{bmatrix}0{,}001 & 0{,}01 & 0{,}1 & 1 & 10 & 100 & 1000\end{bmatrix}$$

Después se definen las referencias en términos absolutos para los dos sensores, además de la duración de cada pareja en muestras. Multiplicado por el Tm, se obtiene la duración en segundos.

$$Ref = \begin{bmatrix}60 & 45 & 45 & 60 & 70\\ 50 & 50 & 40 & 60 & 50\end{bmatrix}, \quad t = \begin{bmatrix}50 & 50 & 50 & 50 & 50\end{bmatrix}.$$

La duración por pareja sería de 650 segundos, casi 11 minutos, y de 55 minutos en total.

El último parámetro que queda por establecer son los límites del estado y de entrada en coordenadas locales del punto de operación. Como ya se mencionó durante la explicación de *gmin* y *gmax*, para TCLab no se pueden definir claramente los de estado, así que se ponen unos inalcanzables, que no entorpezcan al algoritmo pero que le indique si una estimación es absurda o no.

$$Xmin = \begin{bmatrix}-50\\ -50\end{bmatrix}, \quad Xmax = \begin{bmatrix}50\\ 50\end{bmatrix}, \quad Umin = \begin{bmatrix}-50\\ -50\end{bmatrix}, \quad Umax = \begin{bmatrix}50\\ 50\end{bmatrix}..$$

Después de dar valor numérico a los parámetros el programa continúa con la creación del modelo (matrices A, B y C) con la Función A.9 y el guardado de sus dimensiones en constantes. Las referencias se trasladan a las coordenadas locales del punto de operación con las medias de los datos de entrenamiento. Se crean matrices de registro de seis dimensiones (nº de salidas, nº de iteraciones,$NP+NC$, nº de posibles de Q, nº de posibles de R, nº de posibles de P) , para los estados (*xsim*), las salidas (*ysim*) y las entradas (*usim*).

Las referencias se adaptan a las coordenadas locales con las medias de los datos de entrenamiento en y_r. No obstante hace falta traducirlas para que pueda manejarlas el modelo, para lo que se crea la matriz N. Se trata de las matrices del modelo metidas en una supermatriz en la que el estado próximo de la primera ecuación del modelo en estado de espacios, alcanzada la referencia, debe ser igual a la actual, por lo que queda un único vector de estados y se puede sacar factor común.

$$y_r = Nz_r \rightarrow \quad \begin{bmatrix}x_r\\ u_r\end{bmatrix} = \begin{bmatrix}A - I_{nx} & B\\ C & 0\end{bmatrix}^{-1} \begin{bmatrix}0\\ y_r\end{bmatrix}$$

Se crea las bases para la creación de F en todos los escenarios que se contemplan. *Finic* es constante, se puede definir en este punto; del resto de modulos se crean los componentes.

El proceso de simulaciones de siete bucles comienza con el de horizonte de predicción, de 1 a *NPLim*, con índice *NP*. Dentro del primero de muchos se añade a F el módulo *Fmod* y se le da la longitud necesaria a *gmin* y a *gmax*. El segundo es el del horizonte de control, en el que se completa F con *Fnc* si fuese necesario y se crea un vector columna de ceros para f. Luego le siguen tres bucles anidados con una utilidad semejante: dar a Q, R y P, en este orden. Dentro del tercero de ellos se configura H y se inicia la simulación para el escenario establecido con doble bucle temporal, recorriendo t e iterando el número de muestras indicado en cada elemento.

Cada muestra de una simulación sigue este ciclo: actualiza z_r, f, h, *gmin* y *gmax*, utiliza la función A.10 para calcular z y almacena los resultados en los registros *xsim*, *usim* y *ysim*. También, justo antes del último paso, si una de las consignas para la muestra actual es igual a alguno de los límites, se recalcula la parte que corresponde con $x(1)$, el que será $x(0)$ en la siguiente muestra, con el modelo debido a que se puede haber producido un desajuste entre estado y entrada en el cálculo de z dentro de ADMM, pues en estos casos normalmente se supera el número máximo de iteraciones.

A todas las simulaciones realizadas se le calcula el error cuadrático medio respecto a las referencias. De todas ellas nos quedamos con las cinco que el menor error.

	1º	2º	3º	4º	5º
NP	4	4	4	4	4
NC	4	4	4	3	4
Q	1000	1000	1000	1000	1000
R	0,1	0,01	0,001	0,01	0,1
P	100	100	100	100	10

Predominan las configuraciones en las que NP y NC son iguales, como era de esperar. Sin embargo, ninguna de ellas tiene NP igual a 5. Nos quedamos con la primera opción para ensayarla en TCLab. No obstante, aquí el error cuadrático medio es más orientativo que determinante. Como bien confirma, para nosotros tiene mayor importancia que TCLab alcance las referencias establecidas las consignas se encuentren próximas a sus propias referencias. Un sistema termodinámico es lento y su, por así decirlo, inercia no se altera de forma inmediata, mientras los calentadores, unos elementos electrónicos, no nos cuesta nada que funcionen a una potencia u otra. La cuestión que se nos plantea entonces es si esta configuración es la más apropiada para controlar el sistema real, y lo que más llama la atención es la enorme ponderación del estado sobre las entradas, que puede resultar excesiva. A modo de comparación, se prueba otra configuración: $Q = 10$, $R = 0{,}1$ y $P = 1$, con $NC = 2$, para comprobar si se puede conseguir resultados parecidos con menor esfuerzo para el algoritmo.

La Figura 3.1 muestra las respuestas del modelo con las dos configuraciones; la Figura 3.2 las consignas que el controlador proporciona a los calefactores a lo largo de la simulación.

En la simulación el MPC con $NC = 4$ es notablemente más rápido. Alcanza siempre las referencias, con una muy leve sobreoscilación, mientras el otro a veces no lo hace en el número de muestras propuesto. En cuanto a las entradas, las consignas más suaves de la configuración alternativa dibujan picos menos puntiagudos, apenas saturando los actuadores simulados, alejado del comportamiento más agresivo del MPC "más" ajustado.

3.4 Ensayo en TCLab

Reescribimos el programa de simulación y lo adaptamos al funcionamiento de TCLab en el Programa A.11. Se añaden las funciones para la comunicación con TCLab, con lectura de las medidas de los sensores cada segundo y paso de consigna a las entradas en cada tiempo de muestreo. El estado inicial lo aporta las medidas y no la predicción en z. Como parámetros ahora también aparecen NP, NC, Q, R y P.

Como ocurría con los ensayos previos, es necesario esperar una hora a que TCLab alcance el punto de operación, con temperaturas en el equilibrio diferentes a las medias empleadas para el modelo, que desplazarán las referencias hacia el nuevo origen de coordenadas local justo antes de empezar el ensayo como tal.

En las Figuras 3.3 y 3.4 se muestra los resultados del sistema utilizando los dos controladores. En la segunda se deja de añadir las referencias a las entradas ya que dos ensayos con puntos de operación diferentes hacen que también lo sean estas. Hay que considerar que los puntos de operación de los que parte el ensayo son más de 10 ºC superiores a los que se obtuvieron de los datos de entrenamiento, lo que puede afectar a la precisión del modelo.

Se podría considerar al controlador escogido por el error cuadrático medio como el mejor. Consigue que el estado de TCLab evolucione hacia las referencias sin error permanente. En su detrimento se le achaca la gran inestabilidad que presentan las entradas. En el otro controlador también es perceptible pero es mucho más asumible.

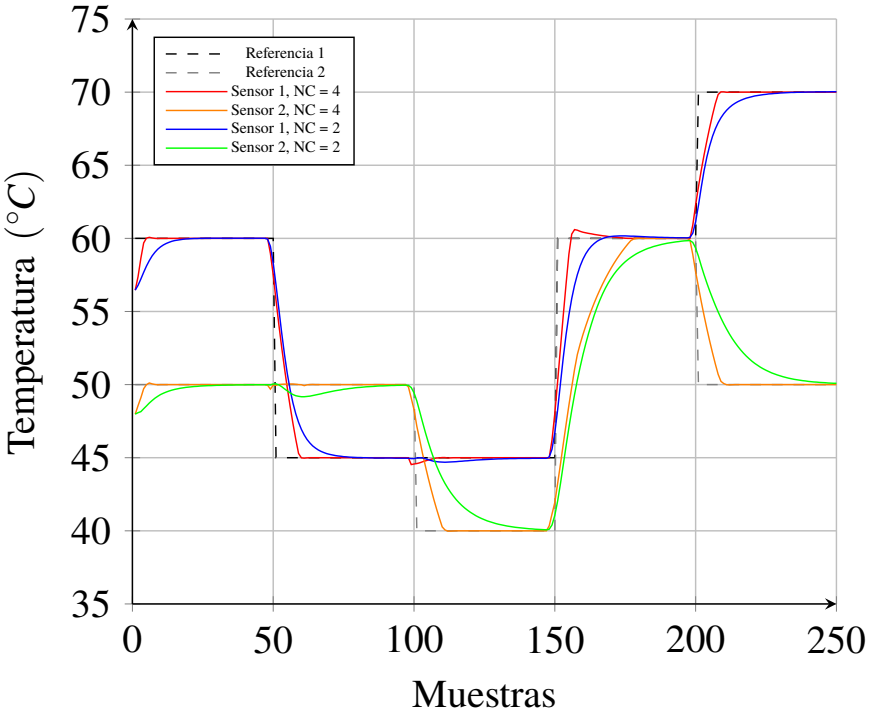

Figura 3.1 Salidas de la simulación del modelo lineal.

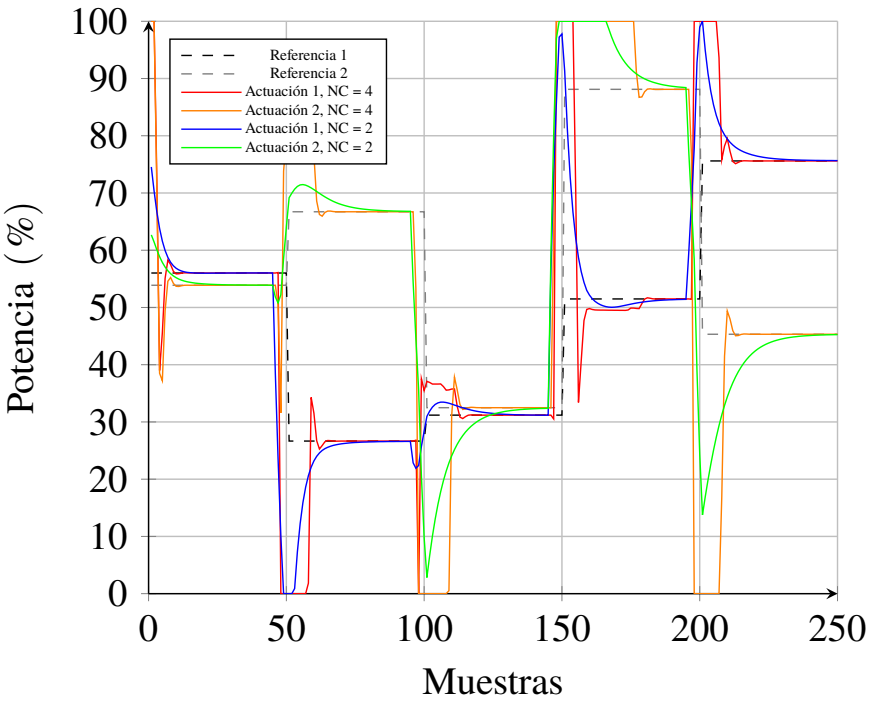

Figura 3.2 Entradas de la simulación del modelo lineal.

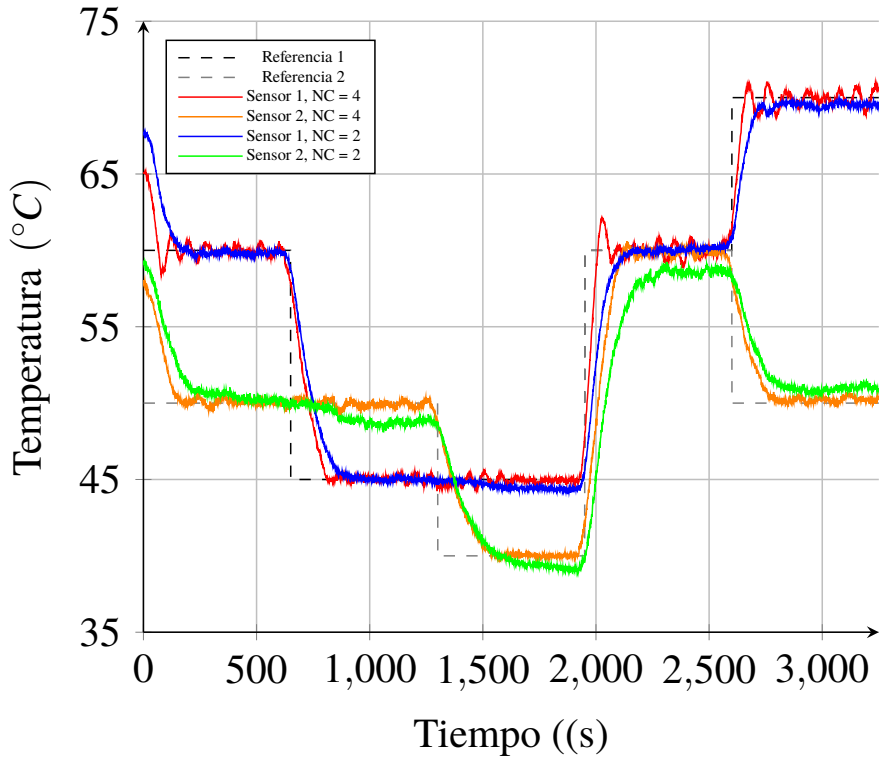

Figura 3.3 Comparativa de MPC ensayado y simulado: medidas de los sensores.

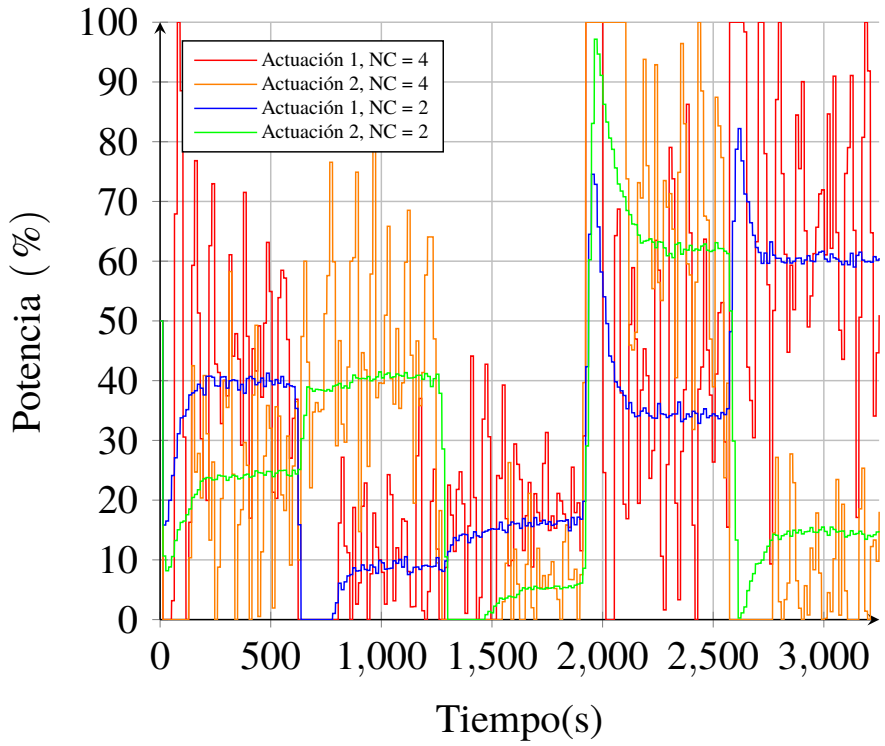

Figura 3.4 Comparativa de MPC ensayado y simulado: actuación.

4 Modelo no lineal

El salto a la no linealidad está motivado por superar una serie de inconvenientes que la linealización es incapaz de evitar. El motivo de este enfoque se fundamenta en la falta de precisión del modelo lineal en estados alejados del punto de operación. Un modelo lineal es una solución aceptable e incluso recomendable en aquellos sistemas en los que se desea que el sistema se mantenga en torno a un punto de operación establecido y se separe de él lo mínimo, como pasa en buena parte de los procesos industriales. No obstante, en aquellos sistemas en que se prevee que su estado cambie mucho dentro de su espacio, esta simplificación puede resultar excesiva.

Para crear un modelo no lineal nos apoyamos en lo ya desarrollado hasta ahora. En realidad son varios modelos lineales diferentes que en conjunto dan una estimación no lineal. Esto es posible gracias a la incorporación de la conocida como lógica borrosa, o "fuzzy logic" en inglés.

4.1 Lógica borrosa

En este contexto la lógica borrosa se utiliza para realizar una estimación de la salida a partir de de la combinación de las estimaciones dadas por múltiples regresores. Cada sensor cuenta con su grupo de regresores propio. Al igual que con el modelo lineal, los grupos son independientes entre sí, aunque no solo en tamaño, sino también en el número de regresores que componen el conjunto. Las últimas medidas de los sensores, indicadores del estado del sistema, indican cuáles de ellas se usan y cómo se ponderan mediante el grado de membresía de las medidas a un conjunto borroso.

Se divide el espectro de temperaturas que puede medir un sensor (con el origen desplazado al punto de operación) en tantos conjuntos borrosos como regresores vaya a tener un sensor y a

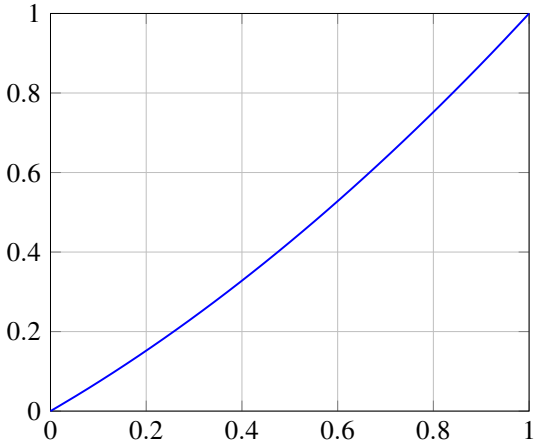

Figura 4.1 Función de reubicación.

cada uno se le asigna una función de membresía. Para la definición de los conjuntos borrosos nos servimos de la nombrada como función de reubicación que se dibuja en la Figura 4.1, que nos es más que una función cuadrática que redistribuye puntos equiespaciados. Sus coeficientes son elegidos arbitrariamente, cumpliendo que $f(0) = 0$ y $f(1) = 1$.

Sobre ella volcamos el conocimiento que tenemos sobre el sistema real. TCLab, como sistema termodinámico que es, la zona "menos lineal" se encuentra en el rango de temperaturas más bajas. Según aumenta la temperatura la no linealidad se va suavizando. Es por ello que con función de reubicación se desplaza los puntos que definen los conjuntos borrosos hacia el origen, estrechando aquellos más cercanos a él y ensanchando los más lejanos. Para adaptarlo a los dos sensores, se multiplican los puntos por el espectro que se estima que tenga cada uno: 70 y 55 ºC, respectivamente.

Los tipos de funciones que se utilizan en este trabajo son triangulares y trapezoidales. Aparecen respectivamente en las Figuras 4.2 y 4.3, en ambos casos en dos escenarios: cuando se divide el espectro en dos conjuntos borrosos y en cuatro.

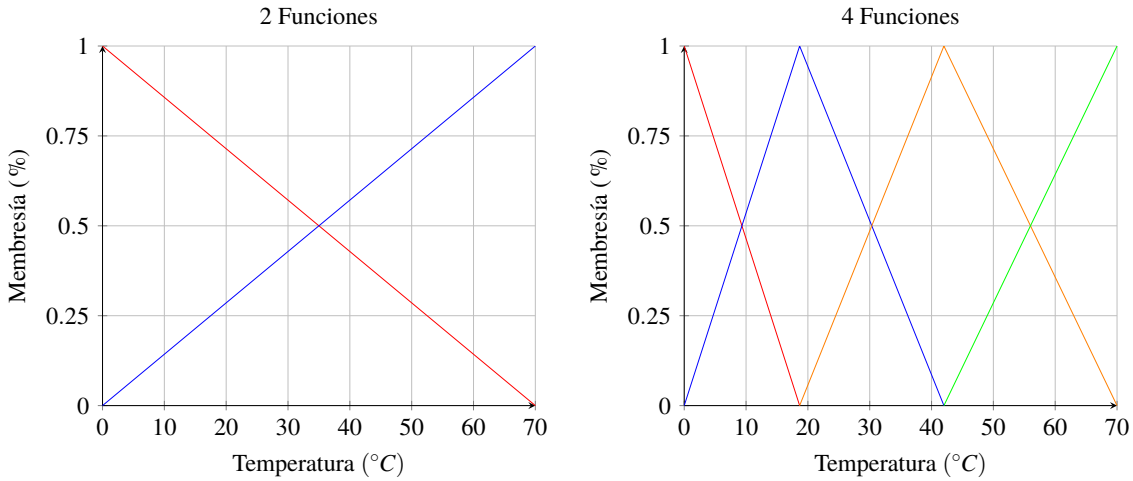

Figura 4.2 Funciones triangulares.

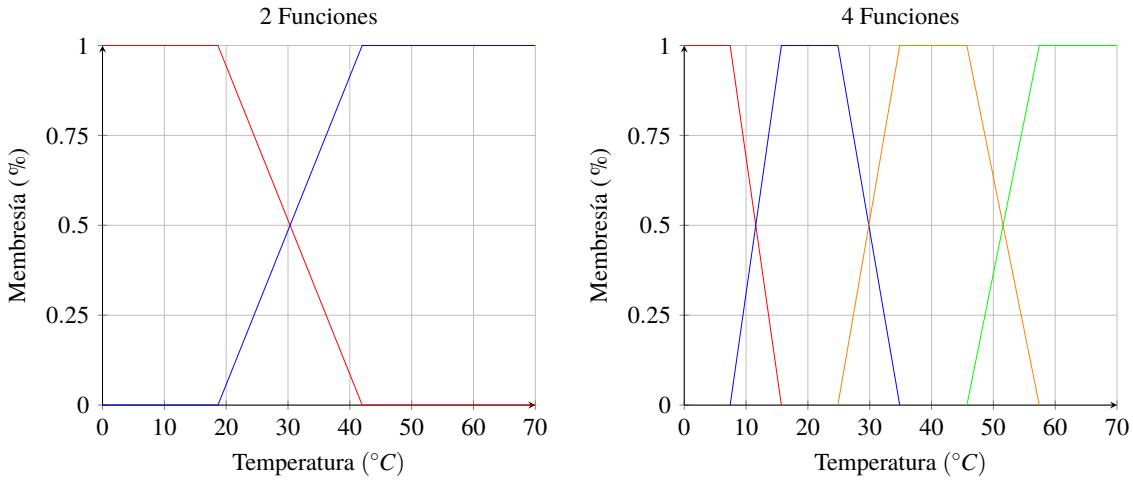

Figura 4.3 Funciones trapezoidales.

Los dos son muy parecidos, siendo la diferencia en que las primeras solo consigue la membresía plena en un valor muy concreto, prácticamente inalcanzable, y el segundo tiene un tramo. Aquí, además, se impone que solo puedan solaparse dos conjuntos a la vez y que la suma de las membresías a distintos conjuntos sea siempre igual a uno. Esto se traduce en que con funciones triangulares la estimación siempre la aportan dos regresores, siempre no lineal, mientras que con los trapezoidales hay estimaciones lineales en los tramos de membresía plena.

Este tipo de sistema borroso en el que el resultado es numérico se conocen como de Takagi-Sugeno. Para el sistema en cuestión, se decide que hay tantas reglas como conjuntos, cada una con una condición, que es pertenecer a él. De cumplirse, se calcula la estimación con el regresor que tiene asignado. Si hay varias, se ponderan las estimaciones en función de los grados de membresía. Como se ve en la Ecuación (4.1), una ventaja que tiene la lógica borrosa para los modelos identificados con regresores es que la estimación final se puede calcular como una ponderación de los regresores multiplicado por el estado actual. Resulta una forma muy sencilla de crear modelos en espacio de estados.

$$\hat{y}(t) = \frac{\sum_{i=1}^{n} w_i \hat{y}_i(t)}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i \theta_i^T r(t)}{\sum_{i=1}^{n}} = \frac{\sum_{i=1}^{n} w_i \theta_i^T}{\sum_{i=1}^{n} w_i} r(t)$$

4.2 Obtención de datos de entrenamiento

Datos como la constante de tiempo y el tiempo mínimo de conmutación expresan el comportamiento del sistema, no hace falta volver a sacarlos. Los datos de validación siguen cumpliendo con su propósito, en torno a un punto de operación diferente pero que el modelo no lineal debe ser capaz de replicar con gran semejanza. En cambio, los datos de entrenamiento no aportan la riqueza que requiere un modelo de varios regresores distribuidos a lo largo del espacio de la salida. Todo lo que se explica en este apartado está volcado en el Programa A.12.

El ensayo PRBS no aporta suficiente información de todo el espectro de temperaturas de TCLab y su comportamiento a lo largo de él. En su lugar se recurre a otro ensayo, muy similar. Los calentadores pueden funcionar a una de estas tres potencias: 0, 50 y 100 %. El número de combinaciones asciende a nueve. A diferencia del PRBS, en el que solo cambia la potencia de una resistencia mientras la otra se mantenía en el punto de operación, estas combinaciones se van turnando. Del PRBS se aprovecha la duración de las consignas de entrada. Cada combinación tiene un contador que indica los tiempos de muestreo en los que ha estado presente durante el ensayo, de forma que el siguiente sea el que tenga el contador más bajo, y en caso de empate se elige uno aleatoriamente.

La duración del ensayo es de siete horas, sin contar la hora para alcanzar el equilibrio en el punto de operación. Esta elección tan arbitraria hace que se deje cumplir la propiedad del PRBS de media nula, pero al estar el punto operación situado a la menor temperatura posible, esta propiedad carece de sentido. Las leyendas incluye la similitud entre el modelo y las medidas filtradas.

4.3 Identificación no lineal

El Programa A.13 es una versión del usado para la identificación lineal. Sigue la misma línea: encontrar la combinación de *ny*, *nu*1 y *nu*2 que minimiza el error cuadrático medio, pero esto se aplica ahora a todo el grupo de regresores. Un nuevo parámetro, *MaxDivis* establece hasta que número de conjuntos se divide el espectro de temperaturas, explorando las opciones intermedias a partir de dos.

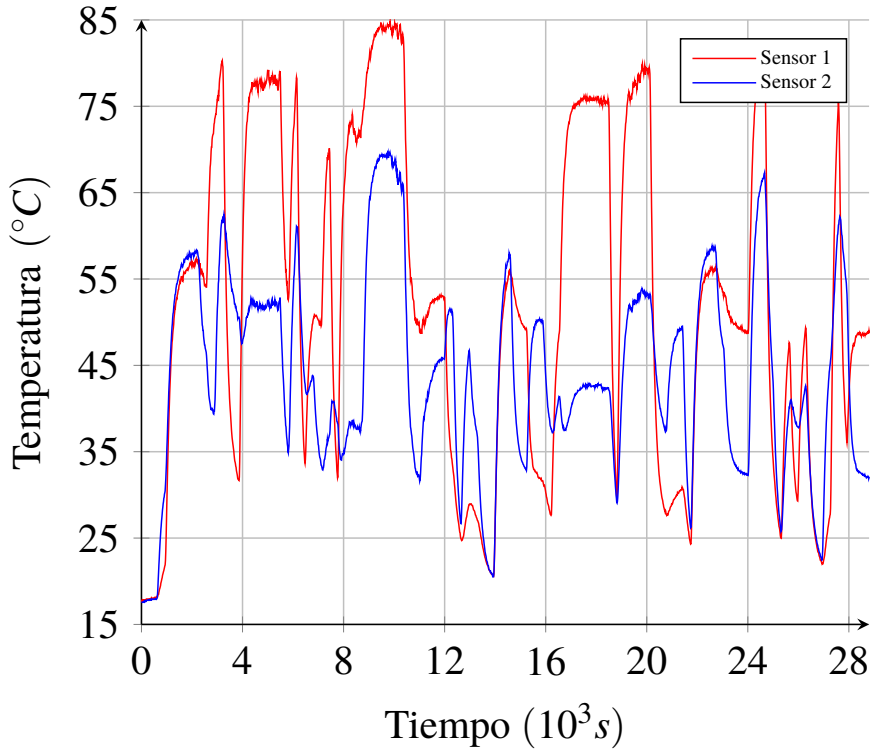

Figura 4.4 Salidas del ensayo de obtención de datos para el modelo no lineal.

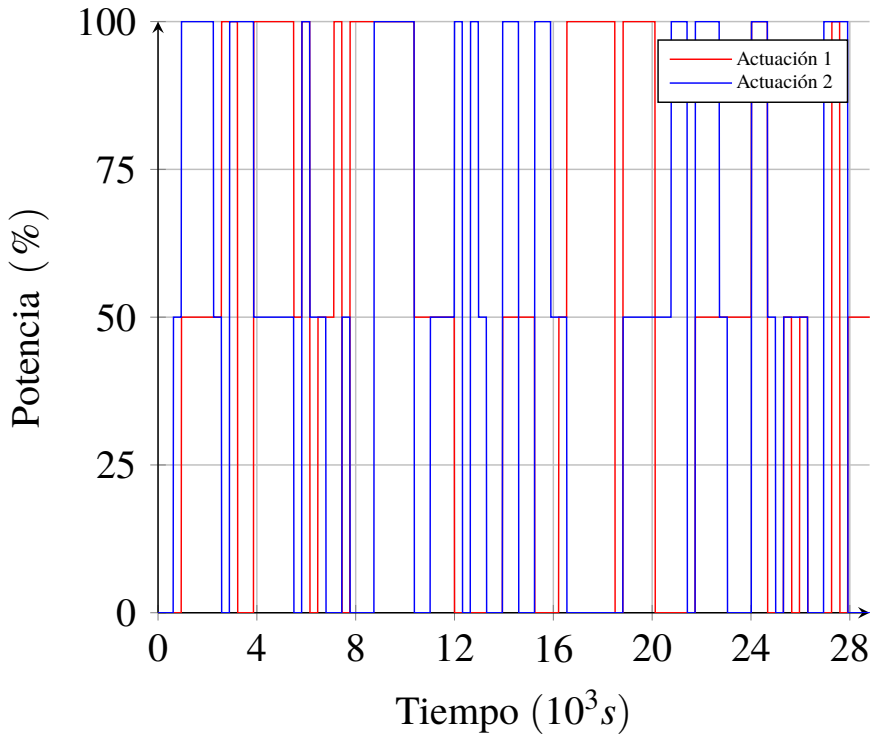

Figura 4.5 Entradas del ensayo de obtención de datos para el modelo no lineal.

Revisitamos la formulación de Ridge Regression.

$$\theta_N = arg \min_{\theta \in \mathbb{R}^{n_\theta}} \frac{1}{2}\theta^T P \theta + \|Y - R\theta\|_2^D$$

En el caso lineal se probaba con numerosas posibilidades de P y se dejaba D como matriz identidad; ahora cambian los roles. Debido al mayor coste computacional, se decide reducir P a una matriz identidad. D, por otro lado, ahora cumple una función: discriminar los errores individualmente por el grado de membresía a un conjunto. Pondera el error con la pertenencia al conjunto de la última medida, que se obtiene al emplear la Función A.14. Además, con el fin de aumentar la presencia en la identificación por conjunto de aquellas medidas de membresía baja, y por consiguiente de su error, en lugar del grado tal cual, la diagonal de D contiene la raíz cuadrada de ellos.

De cada combinación de i, j y k, los índices de los bucles de ny, $nu1$ y $nu2$, se realiza la identificación de los conjuntos en un cuarto bucle anidado. Los regresores resultantes se agrupan en una matriz que se almacena en las celdas $Thetareg1$ y $Thetareg2$ en la coordenada $[i,j,k,l]$, donde l representa el número de conjuntos en que se divide el espectro de temperaturas.

Ahora toca combinar. En el programa de identificación lineal se calcula el error medio cuadrático justo después de obtener el regresor debido a las posibilidades que ofrece múltiples configuraciones de P. Aquí se hace después de obtener todos los regresores, cuando se agrupan entre ellos de manera que no tienen por qué coincidir en el número de coeficientes para el estado y las entradas, pero sí deben coincidir en que deben de pertenecer a un conjunto borroso con el mismo tipo de misma función de membresía. También estos conjuntos deben de cubrir el espectro de temperaturas de forma que la suma de las membresías siempre sea uno, o dicho de otra manera, los regresores escogidos deben estar asociados a conjuntos diferentes, que se han repartido todo el rango de temperaturas.

Se termina la identificación con la validación, que se recuerda que sigue siendo los datos recogidos del ensayo Chirp. Realizando las mismas combinaciones de regresores, se calculan los errores que comete sus estimaciones con respecto a las medidas. Las ponderaciones de estos funcionan de forma parecida: la raíz décima de la media de coeficientes, la suma de todos los de los regresores dividida entre el número de regresores. Se escoge, de entre todas las combinaciones, aquella cuya suma de los errores de entrenamiento y validación ponderados sea menor.

4.4 Resultados

Para la identificación se han escogido estos valores para los parámetros.

ny	$nu1$	$nu2$	$MaxDivis$
3	3	3	4

Se reduce ny a 3, ya que apenas influye que valga 4 y reduce significativamente el número de errores, que se calcula $\sum_{x=2}^{MaxDivis-1}(ny*nu1*nu2)^(x)$. Con ny igual a 3 habría que realizar la respuesta de 1.727.568 modelos con los datos de entrenamiento mientras que igual 3 habría 551.853.

De los resultados, se muestra en cuatro tablas. Se ordenan primero por orden de numeración de los regresores y luego en triangular y trapezoidal. Tienen tantas filas como divisiones se hayan probado. Constan de dos columnas: la de la izquierda contiene la configuración de ny, $nu1$ y $nu2$ de los regresores que conforman los grupos que menor error cuadrático medio ponderado total han cometido, de izquierda a derecha; la de la derecha contiene una tabla con los errores ponderados cometidos por los grupos de regresores, en el mismo orden, con respecto los datos de entrenamiento y de validación, junto a la suma de ambos en la última fila.

θ_1 - Triangular

Nº funciones : 2											
ny	1 3	1 3	1 3	1 3	2 2	*Ent.*	3,51	3,55	3,56	3,56	3,55
*nx*1	2 2	2 3	2 2	2 2	2 1	*Val.*	0,74	0,74	0,75	0,75	0,78
*nx*2	3 3	3 3	3 2	3 1	3 3	*Total*	4,25	4,29	4,31	4,31	4,33
Nº funciones : 3											
ny	1 2 3	1 2 3	1 2 3	1 2 3	1 2 2	*Ent.*	1,75	1,76	1,77	1,78	1,79
*nx*1	3 1 2	2 1 2	3 1 3	2 1 3	3 1 3	*Val.*	0.62	0,62	0,63	0,63	0,63
*nx*2	3 3 3	3 3 3	3 3 3	3 3 3	3 3 3	*Total*	2,37	2,38	2,40	2,41	2,42
Nº funciones : 4											
ny	1 3 2 1	1 3 2 1	1 3 2 1	1 2 2 1	1 2 2 1	*Ent.*	1,22	1,23	1,23	1,22	1,23
*nx*1	2 1 1 3	2 1 1 3	2 1 1 3	2 1 1 3	2 1 1 3	*Val.*	0,66	0,65	0,66	0,66	0,66
*nx*2	3 3 3 3	3 3 3 1	3 3 3 2	3 3 3 3	3 3 3 1	*Total*	1,88	1,88	1,88	1,88	1,88

θ_1 - Trapezoidal

Nº funciones : 2											
ny	2 3	2 3	2 3	3 3	2 3	*Ent.*	2,13	2,15	2,14	2,16	2,16
*nx*1	2 1	3 1	2 1	2 1	3 1	*Val.*	0,68	0,68	0,68	0,69	0,69
*nx*2	3 3	3 3	3 2	3 3	3 2	*Total*	2,81	2,82	2,82	2,85	2,85
Nº funciones : 3											
ny	1 2 1	1 2 3	1 2 1	1 2 2	1 2 1	*Ent.*	1,10	1,06	1,12	1,10	1,11
*nx*1	3 2 3	3 1 2	3 1 3	3 1 3	3 2 3	*Val.*	1.23	1,27	1,21	1,24	1,22
*nx*2	3 3 3	3 3 3	3 3 3	3 3 3	3 3 2	*Total*	2,33	2,33	2,33	2,33	2,33
Nº funciones : 4											
ny	1 3 2 1	1 2 2 1	1 3 2 1	1 2 2 1	1 3 2 2	*Ent.*	0,99	0,99	1,00	1,00	0,99
*nx*1	2 1 1 3	2 1 1 3	2 1 1 3	2 1 1 3	2 1 1 3	*Val.*	0,63	0,63	0,63	0,63	0,64
*nx*2	3 3 3 3	3 3 3 3	3 3 3 3	3 3 3 3	3 3 3 3	*Total*	1,62	1,62	1,63	1,63	1,63

Para el sensor 1, la mejor opción por el criterio elegido es un grupo de regresores compuesto por cuatro con función de membresía trapezoidal, por una escueta diferencia con la opción triangular. Sorprende que el error cometido con respecto los datos de validación sea menor que con los respecto los de entrenamiento en la mayoría de escenarios. En ambos se vislumbran derivas en las medidas que complican sacar buenos resultados.

Un análisis en más profundidad nos permite detectar patrones que se repiten en la estructura de los regresores, no sin motivo aparente. Casi todos los regresores tienen *nu*2 igual a 3, pero el valor *nu*1 disminuye según el regresor esté más cerca del centro, al contrario que *ny*.

θ_2 - Triangular

Nº funciones : 2											
ny	2 3	2 3	2 3	2 3	1 3	*Ent.*	5,33	5,49	5,52	5,72	5,84
*nx*1	3 3	3 3	3 3	2 3	2 3	*Val.*	5,02	4,94	4,97	4,86	4,76
*nx*2	1 1	1 2	1 3	1 1	1 1	*Total*	10,4	10,4	10,5	10,6	10,60
Nº funciones : 3											
ny	1 3 1	1 3 1	1 3 1	1 3 1	1 3 3	*Ent.*	3,96	3,98	3,99	4,00	4,02
*nx*1	3 3 3	3 3 3	3 3 3	3 3 3	3 3 2	*Val.*	5,01	5,01	5,03	5,06	5,06
*nx*2	1 1 2	1 1 3	1 1 1	1 3 2	1 3 3	*Total*	8,97	8,99	9,02	9,06	9,08
Nº funciones : 4											
ny	1 2 1 3	1 2 1 3	1 2 1 3	1 2 3 3	1 2 3 3	*Ent.*	4,35	4,32	4,34	3,51	3,56
*nx*1	3 3 3 1	2 3 3 3	3 3 3 2	3 1 3 3	3 3 3 1	*Val.*	4,83	4,87	4,85	5,69	5,64
*nx*2	1 1 3 1	1 1 3 1	1 1 3 1	1 3 2 1	1 1 2 1	*Total*	9,18	9,19	9,19	9,20	9,20

θ_2 - Trapezoidal

Nº funciones : 2											
ny	2 3	2 3	2 3	2 3	2 3	*Ent.*	3,89	3,72	3,93	4,06	4,24
*nx*1	3 3	3 3	3 3	2 3	2 3	*Val.*	5,06	5,24	5,08	5,09	4,92
*nx*2	1 2	1 1	1 3	1 1	1 2	*Total*	8,95	8,96	9,01	9,15	9,16
Nº funciones : 3											
ny	1 3 3	1 3 1	1 3 1	1 3 2	1 3 3	*Ent.*	3,38	3,85	3,82	3,60	3,39
*nx*1	3 3 3	3 3 3	3 3 3	3 3 3	3 3 3	*Val.*	9,33	8,94	8,99	9,20	9,42
*nx*2	1 1 1	1 1 1	1 1 2	1 1 1	1 3 1	*Total*	12,7	12,8	12,8	12,8	12,8
Nº funciones : 4											
ny	1 3 3 3	1 3 3 3	1 3 3 3	1 3 3 3	1 3 3 3	*Ent.*	3,58	3,61	3,63	3,67	3,64
*nx*1	3 3 3 3	2 3 3 3	3 3 3 3	2 3 3 3	3 3 3 3	*Val.*	3,82	3,81	3,79	3,77	3,80
*nx*2	1 1 1 1	1 1 2 1	1 1 2 1	1 1 2 1	1 1 3 1	*Total*	7,40	7,42	7,42	7,44	7,44

Los resultados del sensor 2, como ocurría el caso lineal, son peores, pero aún más que en este caso. Aquí también la mejor opción se obtiene de un grupo de cuatro regresores con funciones de membresía trapezoidales. En cuanto a los patrones, *nu*2, el parámetro de la entrada más próxima al sensor en cuestión, suele tener un valor igual a 1, mientras *nu*1 se tiende a fijar en 3. El último valor de *ny*, a diferencia del otro sensor, es alto, dejando así de dibujar un resalto a lo largo de los regresores.

A continuación se muestra la composición de los regresores escogidos. Podrían tener funciones y número de regresores entre sí, ya que los modelos de cada sensor siempre han sido independientes entre sí, pero lo lógico es que dos sensores iguales se parezcan mucho.

$$\theta_1 = \begin{bmatrix} 0{,}9685 & 1{,}6449 & 1{,}4435 & 0{,}9374 \\ 0 & -0{,}6945 & -0{,}4776 & 0 \\ 0 & 0{,}0289 & 0 & 0 \\ 0{,}0040 & 0{,}0134 & 0{,}0201 & 0{,}0053 \\ 0{,}0230 & 0 & 0 & 0{,}0155 \\ 0 & 0 & 0 & 0{,}0150 \\ 0{,}0016 & 0{,}0015 & 0{,}0019 & 0{,}0033 \\ -0{,}0028 & -0{,}0014 & -0{,}0019 & -0{,}0005 \\ 0{,}0057 & 0{,}0022 & 0{,}0031 & 0{,}0025 \end{bmatrix}, \quad \theta_2 = \begin{bmatrix} 0{,}9699 & 1{,}3168 & 1{,}2538 & 1{,}1524 \\ 0 & -0{,}0467 & -0{,}0128 & 0{,}0397 \\ 0 & -0{,}2871 & -0{,}2650 & -0{,}2203 \\ -0{,}0117 & 0{,}0020 & 0{,}0031 & 0{,}0002 \\ 0{,}0069 & -0{,}0021 & -0{,}0033 & 0{,}0003 \\ 0{,}0137 & 0{,}0041 & 0{,}0049 & 0{,}0040 \\ 0{,}0130 & 0{,}0066 & 0{,}0079 & 0{,}0097 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Las estimaciones de los mejores modelos, tanto triangulares como trapezoidales, con los datos de entradas del ensayo de entrenamiento son dibujados en las Figuras 4.6 y 4.7. Las estimaciones a partir de los datos de validación en las Figuras 4.8 y 4.9.

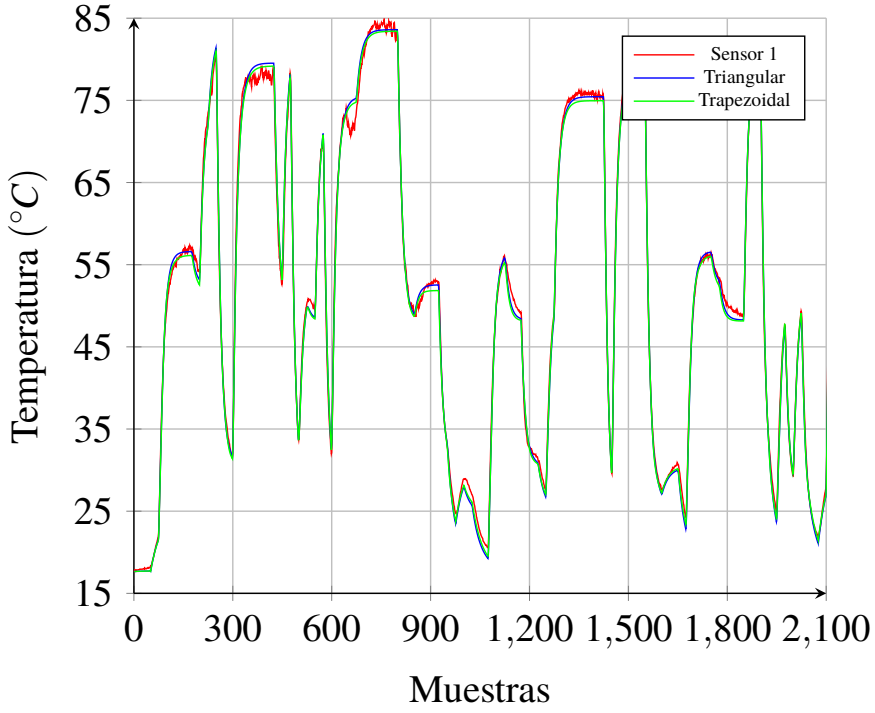

Figura 4.6 Datos de entrenamiento para el modelo del sensor 1 y su respuesta.

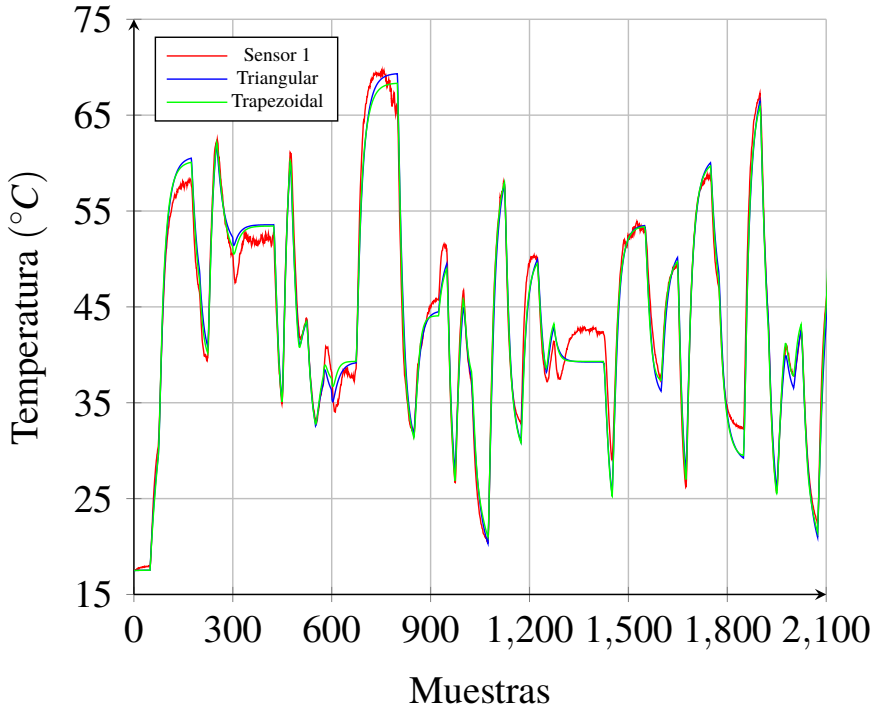

Figura 4.7 Datos de entrenamiento para el modelo del sensor 2 y su respuesta.

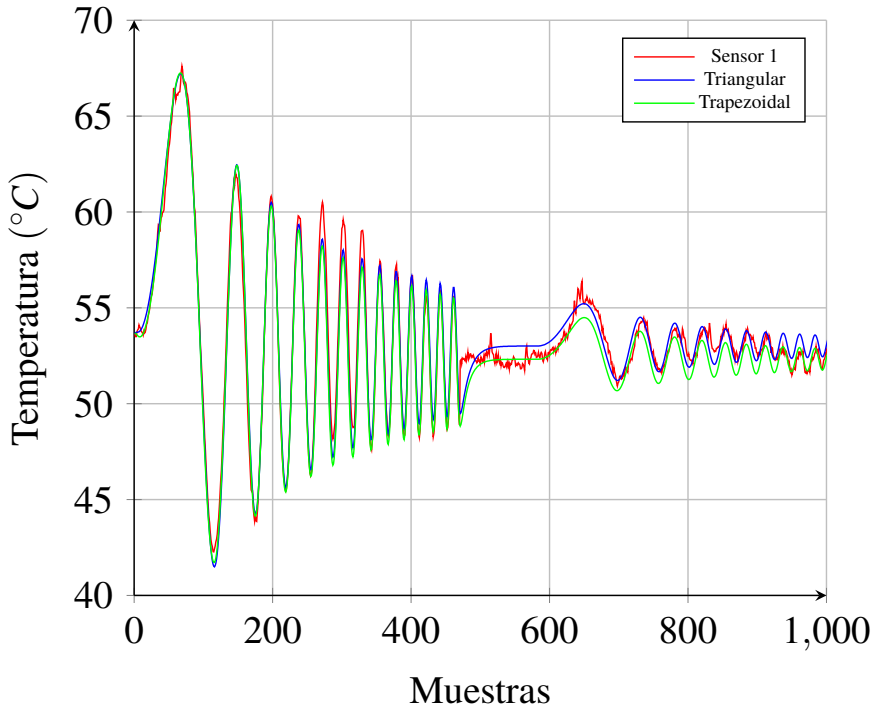

Figura 4.8 Datos de validación para el modelo del sensor 1 y su respuesta.

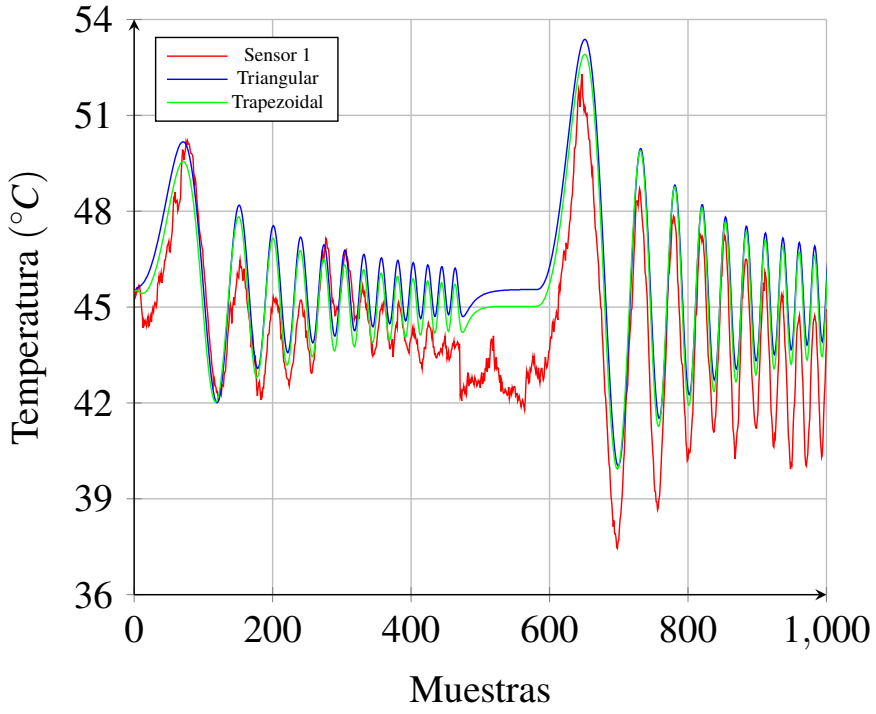

Figura 4.9 Datos de validación para el modelo del sensor 2 y su respuesta.

5 Control no lineal

La expresión de un modelo en espacio de estados es lineal. Cuando se quiere expresar de este modo un modelo no lineal, es necesario linealizarlo. Esto implica que las matrices que conforman el modelo en espacio de estados ya no son constantes, sino variantes en el tiempo, reescribiéndose en cada tiempo de muestreo. A esta variante se la conoce como LPV (Linear Parameter Varying).

$$x(t+1) = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t)$$

La lógica borrosa facilita mucho la linealización, ya que las matrices A y B se formulan a partir de los regresores, ponderados en función del grado de membresía de la última medida de los sensores. También resulta una ventaja a la hora de realizar un control predictivo, pues las matrices que se definen para el futuro se hacen a partir de las predicciones. Eso sí, el algoritmo de resolución de la función objetivo tardará más en converger a una solución, algo en lo que se indaga más adelante. Por como se ha definido C, esta permanece invariante en el tiempo.

5.1 Simulación

El programa A.20 repite la tónica del simulador del modelo lineal. Para el modelo que se carga, junto con las funciones de membresía que necesite, se busca la combinación de Q, R y P que mejores resultados dé, la cual se mantiene constante. Estos son los puntos afectados:

1. F no es constante. El uso de un modelo no lineal solo afecta al módulo $Fmod$, el único que se constituye de las matrices del modelo. Las matrices A(t) y B(t) se obtienen a partir de la membresía del estado x(t) a los diferentes conjuntos borrosos utilizando la Función A.9. Se llama para realizar las NP predicciones, usando x(t+j|t) para construir las matrices que permiten predecir x(t+j+1|t). Esto no impide preparar F a falta de la adición de estas matrices.
2. Las referencias de los sensores, además de ser adaptadas a las coordenadas locales del punto de operación, con su membresía a los diferentes determinan las matrices A y B, necesarias para su conversión. Dicho de otro modo, hay que calcular una matriz N diferente para cada referencia.
3. La ponderación de la restricción terminal, como las medidas y a su vez sus membresías cambian con el tiempo, se complica. En esta ocasión se simplifica calculando los factores para las coeficientes de estado en la muestra actual.
4. La z final es producto de que el algoritmo ADMM ha convergido a una solución que a su vez ha hecho converger a las matrices de modelo utilizadas para las predicciones.

Este último punto requiere de una explicación más distendida. Cuando se trata de modelos no lineales, el algoritmo ADMM proporciona una solución para el problema que se le propone, y que entre los elementos que lo define está F. Cada z que devuelve el algoritmo la reescribe. Por tanto, es necesario un proceso iterativo externo a ADMM que incluya la llamada de su función, y en la nueva Función A.21, que devuelve a la solución la z final para la muestra.

Se muestran los valores dados a los parámetros, que no cambian respecto del controlador lineal.

NPlim	*rho*	*tol*	*MaxIter*
5	1	1e-3	20

Y entonces las configuraciones NP, NC, Q, R y P que menor error cuadrático medio han cometido con ellos.

	1º	2º	3º	4º	5º
NP	5	5	5	5	5
NC	5	5	5	5	5
Q	1000	1000	1000	100	100
R	0,1	0,001	0,01	0,01	0,001
P	1000	1000	1000	100	100

Se recuerda que esta tabla se elabora a título orientativo. Con la misma sospecha de una ponderación excesiva del estado, se propone otra configuración, con $NC = 2$, $Q = 10$ y $P = 10$.

En la Figura 5.1 se presenta las simulaciones del modelo no lineal trapezoidal con las dos configuraciones comentadas, con la Figura 5.2 ilustrando las consignas dadas a lo largo de ellas.

La respuesta es un poco más lenta, resultando evidente en la cuarta referencia. Salvo por esto, el MPC no lineal funciona adecuadamente en este entorno, a pesar de partir de unas temperaturas muy alejadas de las referencias.

5.2 Ensayo en TCLab

En cuanto a código el cambio entre el ensayo no lineal y el lineal es mínimo, además de lo descrito en el apartado anterior, como demuestra el programa A.22. El gran cambio es la espera para alcanzar el punto de operación: se reduce a tomar unas pocas medidas a temperatura ambiente para calcular las medias. Este ensayo es el más corto de todos, durando menos de una hora. Las Figuras 5.3 y 5.4 muestran la evolución del estado y las consignas de las entradas de los MPC con las dos configuraciones.

Tiene lugar un escenario parecido, pero no igual, al del controlador lineal. El de horizonte de control igual al de predicción alcanza las referencias sin apenas sobreoscilación aunque con oscilaciones en régimen permanente debido a las consignas agresivas. El controlador alternativo es mucho más comedido en este aspecto. Le cuesta alcanzar las referencias, no haciendolo las dos primeras veces pero sí las siguientes, lo que significar que en esta ocasión haya actuado mejor el controlador alternativo.

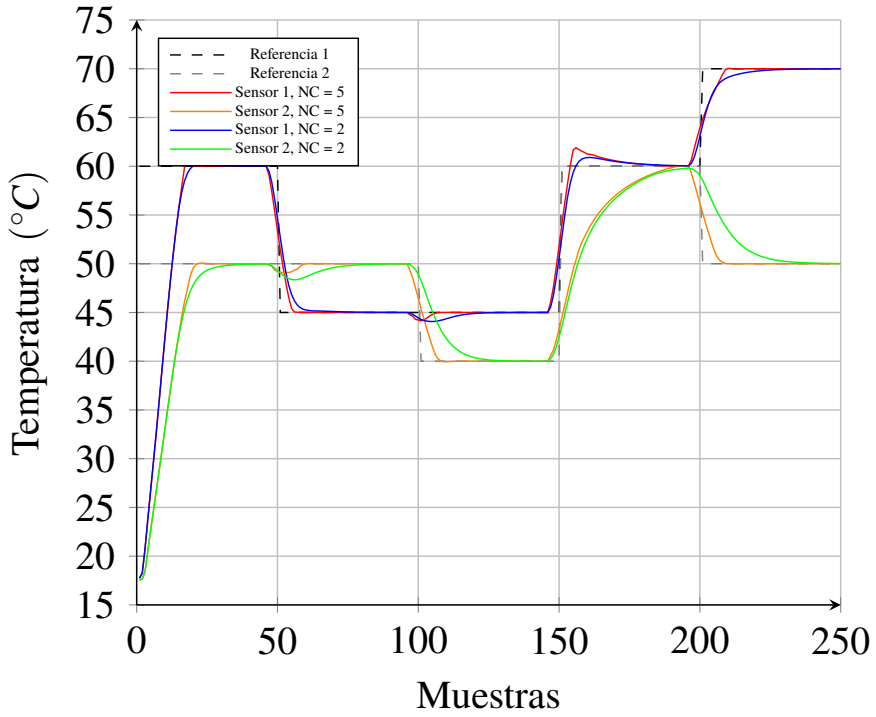

Figura 5.1 Simulaciones de los MPC triangular y trapezoidal: sensores.

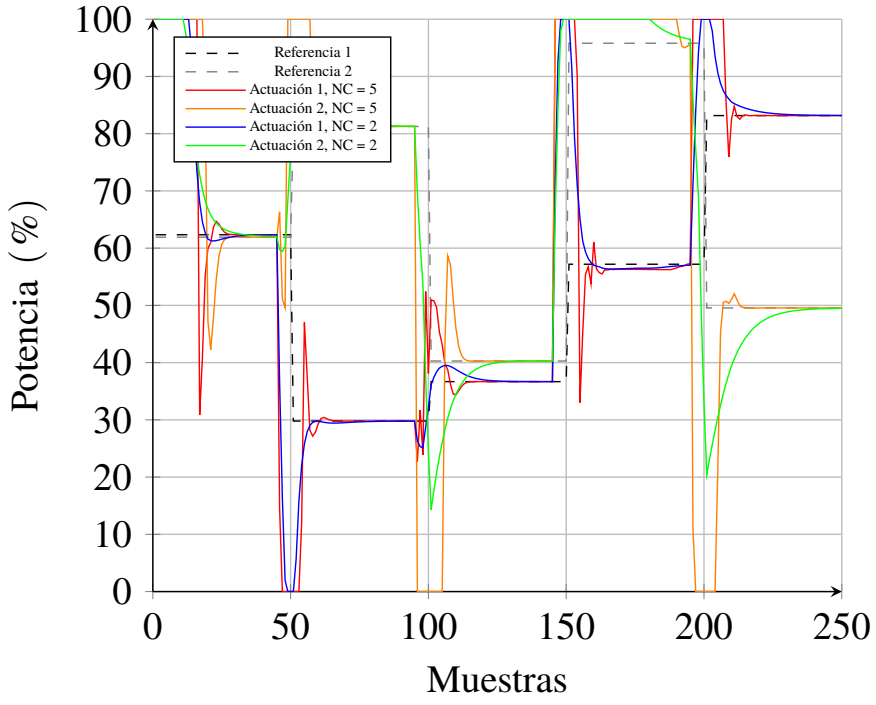

Figura 5.2 Simulaciones de los MPC triangular y trapezoidal: actuadores.

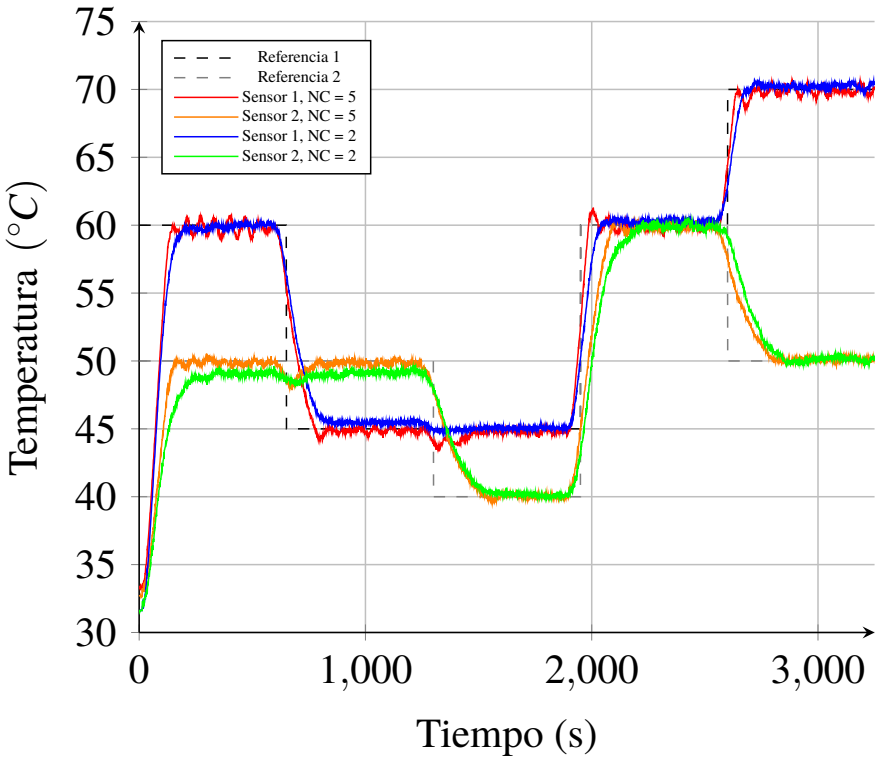

Figura 5.3 Ensayos de los MPC triangular y trapezoidal: sensores.

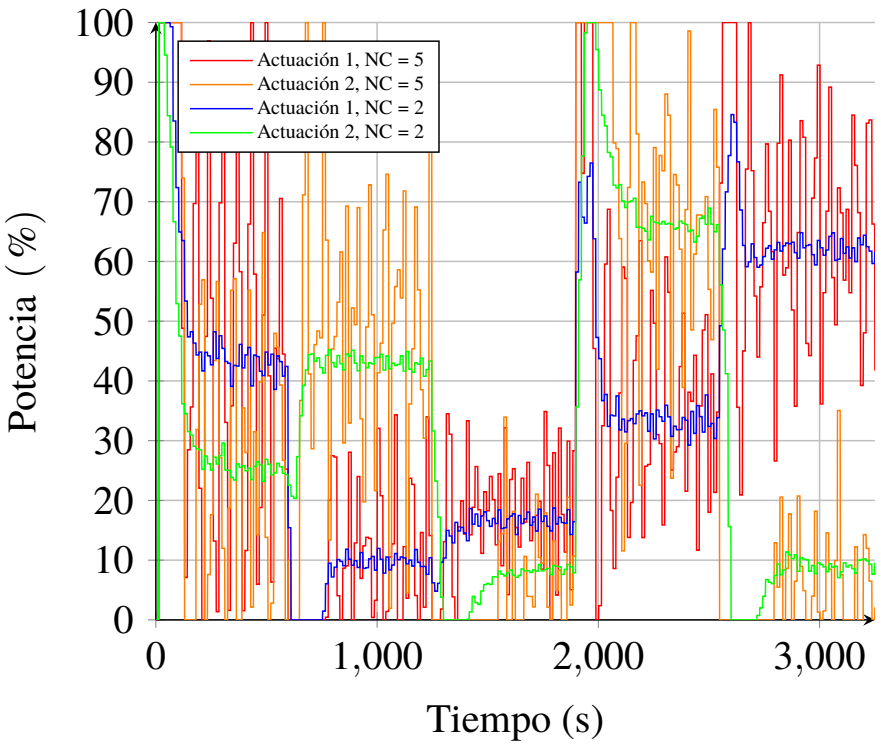

Figura 5.4 Ensayos de los MPC triangular y trapezoidal: actuadores.

6 Conclusiones

Se ha conseguido diseñar un MPC no lineal basado en un modelo identificado también no lineal con el que es sencillo obtener uno linealizado en cada muestra, con resultados satisfactorios. Siempre es posible profundizar más en todos los conceptos que se ha propuesto, como otras ideas para conseguir la no linealidad con modelos lineales o la modificación a tiempo real de las matrices Q, R y P en función del estado del sistema, en aquellos que puedan requerirlo.

Sería interesante probar esta estrategia tanto de identificación como de control en sistemas reales de diferente índole, complejidad y de número de entradas y salidas. Como ideas se propone ampliar lo desarrollado en este trabajo probando otros métodos de regresión y diferentes tipos de funciones de membresía.

Apéndice A
Programas

Aquí se incluye el código escrito en el lenguaje de Matlab. Cada sección corresponde a un capítulo y los programas se suceden en orden de aparición en la memoria.

A.1 Modelo lineal

Listing A.1 ENSAYO_ESCALON.m.

```
%% Inicializacion
close all; clear; clc

tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2

h1esc = [0 0:25:100]; % Escalones para la entrada 1
h2esc = [0 50*ones(1,length(h1esc)-1)]; % Escalones para la entrada 2
dur = [200 60*60*ones(1,length(h1esc)-1)]; % Tiempo entre escalones
escals = min(length(h1esc),length(h2esc)); % Numero de escalones

%% Ensayo de saltos en escalón

n = 0;
disp('Comienzo ensayo escalones')
for iter = 1:escals
   ht1 = h1esc(iter);
   ht2 = h2esc(iter);
   h1(ht1);
   h2(ht2);

   for i = 1:dur(iter)
```

```
        tic;

        t1 = T1C();
        t2 = T2C();
        h1s = [h1s,ht1];
        h2s = [h2s,ht2];
        t1s = [t1s,t1];
        t2s = [t2s,t2];

        %Representación gráfica
        n = n+1;
        time = linspace(0,n+1,n);
        clf
        subplot(2,1,1)
        plot(time,t1s,'r.');
        hold on
        plot(time,t2s,'b.');
        ylabel('Temperature (degC)')
        legend('Temperature 1','Temperature 2','Location','NorthWest')
        subplot(2,1,2)
        plot(time,h1s,'r-','LineWidth',2);
        hold on
        plot(time,h2s,'b--','LineWidth',2);
        ylabel('Heater (0-5.5 V)')
        xlabel('Time (sec)')
        legend('Heater 1','Heater 2','Location','NorthWest')
        drawnow;
        t = toc;
        pause(max(0.01,1.0-t))
    end
end

h1(0); h2(0); % Apaga calentadores

disp('Fin ensayo escalones')
```

Listing A.2 ENSAYO_RELE.m.

```
%% Inicializacion
close all; clear; clc

tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2
```

```
PO = [50;50]; % Entradas para el P.O.

T = 1; % 1: Ensayo en sensor 1, 2: Ensayo en sensor 2
hist = 0.2; % Banda de histéress = 2 * hist
dur = 30*60; % Duración del ensayo en segundos
inic = 60*60; % Duración de la espera en segundos

%% Espera hasta alcanzar P.O.

h1(PO(1));
h2(PO(2));

for i=1:inic
    tic;
    t1 = T1C();
    t2 = T2C();
    h1s = [h1s,PO(1)];
    h2s = [h2s,PO(2)];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    clf
    plot(t1s,'r.')
    hold on
    plot(t2s,'b.')
    t = toc;
    pause(max(0.01,1.0-t))
end

if T==1
    disp('Test relé Heater 1')
    t1ref=mean(t1s(end-29:end));
    h1(100);
else
    disp('Test relé Heater 2')
    t2ref=mean(t2s(end-29:end));
    h2(100);
end

%% Ensayo de control en relé

n=0;
disp('Comienzo ensayo relé')
for i = 1:dur
    tic;

    if T == 1
        if t1 > t1ref+hist
            ht1 = 0;
        elseif t1 < t1ref-hist
            ht1 = 100;
```

```
        end
        h1(ht1);
    else
        if t2 > t2ref+hist
            ht2 = 0;
        elseif t2 < t2ref-hist
            ht2 = 100;
        end
        h2(ht2);
    end

    t1 = T1C();
    t2 = T2C();

    h1s = [h1s,ht1];
    h2s = [h2s,ht2];
    t1s = [t1s,t1];
    t2s = [t2s,t2];

    % Representación gráfica
    n = n+1;
    time = linspace(0,n+1,n);
    clf
    subplot(2,1,1)
    plot(time,t1s(inic+1:end),'r.');
    hold on
    plot(time,t2s(inic+1:end),'b.');
    ylabel('Temperature (degC)')
    legend('Temperature 1','Temperature 2','Location','NorthWest')
    subplot(2,1,2)
    plot(time,h1s(inic+1:end),'r-','LineWidth',2);
    hold on
    plot(time,h2s(inic+1:end),'b--','LineWidth',2);
    ylabel('Heater (0-5.5 V)')
    xlabel('Time (sec)')
    legend('Heater 1','Heater 2','Location','NorthWest')
    drawnow;
    t = toc;
    pause(max(0.01,1.0-t))
end

h1(0); h2(0); % Apaga calentadores

disp('Fin ensayo relé')
```

Listing A.3 ENSAYO_PRBS.m.

```
%% Inicialización
close all; clear; clc
```

```
tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2

PO = [50;50]; % Entradas para el P.O.

Tm = 65; % Tiempo de muestreo (aumentado)
amp = 60; % Amplitud
inic = 60*60; % Duracion espera para P.O.
rest = 30*60; % Duracion reposo entre ensayos

[u,t] = PRBS(8,65,13,1); % Generación de señal PRBS

cont = 0; suma = 0; mini = 20;
while cont<mini || suma~=cont/2 % Recorte de la señal
    cont=cont+1;
    suma=suma+u(cont);
end
u = amp*u(1:cont)+(100-amp)/2; % Entrada con forma de señal PRBS

dur = Tm*length(u); % Duracion de cada ensayo
tot = [dur rest dur];

%% Espera para alcanzar el P.O.

h1(PO(1));
h2(PO(2));

for i = 1:inic
    tic;
    t1 = T1C();
    t2 = T2C();
    h1s = [h1s,PO(1)];
    h2s = [h2s,PO(2)];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    clf
    plot(t1s,'r.');
    hold on
    plot(t2s,'b.');
    t = toc;
    pause(max(0.01,1.0-t))
end

%% Ensayo PRBS
```

```
n=0;

disp('Comienzo ensayo PRBS')
for k = 1:length(tot)
    if k == 2 % Reposo
        ht1 = PO(1);
        h1(ht1);
    end
    iter = 0;

    for i = 0:tot(k)-1
        tic;

        if mod(i,Tm) == 0 % Cada tiempo de muestreo
            iter = iter+1;
            if k == 1 % Ensayo del actuador 1
                ht1 = u(iter);
                h1(ht1);
            elseif k == 3 % Ensayo del actuador 2
                ht2 = u(iter);
                h2(ht2);
            end
        end

        t1 = T1C();
        t2 = T2C();
        h1s = [h1s,ht1];
        h2s = [h2s,ht2];
        t1s = [t1s,t1];
        t2s = [t2s,t2];

        % Representación gráfica
        n = n+1;
        time = linspace(0,n+1,n);
        clf
        subplot(2,1,1)
        plot(time,t1s(inic+1:end),'r.','MarkerSize',10);
        hold on
        plot(time,t2s(inic+1:end),'b.','MarkerSize',10);
        ylabel('Temperature (degC)')
        legend('Temperature 1','Temperature 2','Location','NorthWest')
        subplot(2,1,2)
        plot(time,h1s(inic+1:end),'r-','LineWidth',2);
        hold on
        plot(time,h2s(inic+1:end),'b--','LineWidth',2);
        ylabel('Heater (0-5.5 V)')
        xlabel('Time (sec)')
        legend('Heater 1','Heater 2','Location','NorthWest')
        drawnow;
```

```
        t = toc;
        pause(max(0.01,1-t))
    end
end

h1(0); h2(0); % Apaga calentadores

disp('Fin ensayo PRBS')
```

Listing A.4 ENSAYO_CHIRP.m.

```
%% Inicialización
close all; clear; clc

tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2

PO = [50;50]; % Entradas para punto de operación

inic = 60*60; % Duración de la espera para P.O.
rest = 30*60; % Duración de reposo entre ensayos
Tm = 13.0; % Tiempo de muestreo
T180 = 65; % Tm=T180/5
amp = 30; % Amplitud de la señal/2

Tdom = 180; % Constante de tiempo
wmin = 0.1/Tdom; % Frecuencia mínima
wmax = 2*pi/T180/4; % Frecuencia máxima

Tchirp = floor(2*pi*10*Tdom); % Duración de cada ensayo Chirp
w = wmin+(wmax-wmin)*[0:Tchirp]/Tchirp; % Señal Chirp

u=PO(1)+amp*sin(w.*[0:Tchirp]); % Entrada con forma de señal Chirp
tot=[Tchirp rest Tchirp]; % Duración total

%% Espera para alcanzar el P.O.

h1(PO(1));
h2(PO(2));

for i = 1:inic
    tic;
    t1 = T1C();
    t2 = T2C();
```

```
    h1s = [h1s,PO(1)];
    h2s = [h2s,PO(2)];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    clf
    plot(t1s,'r.');
    hold on
    plot(t2s,'b.');
    t = toc;
    pause(max(0.01,1.0-t))
end

%% Ensayo Chirp

n = 0;

disp('Comienzo test Chirp')
for k = 1:length(tot)
    if k == 2 % Reposo
        ht1 = PO(1);
        h1(ht1);
    end

    for i = 1: tot(k)
        tic;

        if k == 1 % Ensayo del actuador 1
            ht1 = u(i);
            h1(ht1);
        elseif k == 3 % Ensayo del actuador 2
            ht2 = u(i);
            h2(ht2);
        end

        t1 = T1C();
        t2 = T2C();
        h1s = [h1s,ht1];
        h2s = [h2s,ht2];
        t1s = [t1s,t1];
        t2s = [t2s,t2];

        % Representación gráfica
        n = n+1;
        time = linspace(0,n+1,n);
        clf
        subplot(2,1,1)
        plot(time,t1s(inic+1:end),'r.');
        hold on
        plot(time,t2s(inic+1:end),'b.');
        ylabel('Temperature (degC)')
```

```
        legend('Temperature 1','Temperature 2','Location','NorthWest')
        subplot(2,1,2)
        plot(time,h1s(inic+1:end),'r-','LineWidth',2);
        hold on
        plot(time,h2s(inic+1:end),'b--','LineWidth',2);
        ylabel('Heater (0-5.5 V)')
        xlabel('Time (sec)')
        legend('Heater 1','Heater 2','Location','NorthWest')
        drawnow;
        t = toc;
        pause(max(0.01,1.0-t))
    end
end

h1(0); h2(0); % Apaga calentadores

disp('Fin test Chirp')
```

Listing A.5 IDENT_LIN.m.

```
%% Base de datos
close all; clear; clc

load IDENT_LIN_DATOS.mat % Datos de entrenamiento brutos
[u,y] = filtro(h1s,h2s,t1s,t2s,Tm); % Filtrado y muestreado
PO = [50 50];

medias=mean(y,1);
ym = y-repmat(medias,length(y),1); % Salidas menos P.O.
um = u-repmat(PO,length(u),1); % Entradas menos entradas para P.O.

%% Parámetros

ny = 4; % Parámetros máximos
nyt = 0; % Valores pasados de las sondas
nu1 = 3; % Valores pasados del calentador 1
nu2 = 3; % Valores pasados del calentador 2
nut1 = 0; % Valores pasados de las entradas
nut2 = 0;

Regul = 4;

%% Regitros

Modres1 = cell(ny,nu1,nu2);
Modres2 = Modres1;
Errcuad1 = 1e6*ones(ny,nu1,nu2);
Errcuad2 = Errcuad1;
Preg1 = cell(ny,nu1,nu2);
```

```
Preg2 = Preg1;
Thetareg1 = cell(ny,nu1,nu2);
Thetareg2 = Thetareg1;

%% Identificación

for i = 1:ny
    Pny = p_theta([],Regul,i);
    for j = 1:nu1
        Pnu1 = p_theta([],Regul,j);
        for k = 1:nu2
            Pnu2 = p_theta([],Regul,k);
            tope = max([i+nyt,j+nut1,k+nut2]);
            E1 = zeros(length(ym)-tope,i+j+k);
            E2 = E1;
            for l = tope+1:length(ym)
                E1(l-tope,:) = [ym(l-1:-1:l-i,1)',um(l-1-nut1:-1:l-j-nut1
                    ,1)',um(l-1-nut2:-1:l-k-nut2,2)'];
                E2(l-tope,:) = [ym(l-1:-1:l-i,2)',um(l-1-nut1:-1:l-j-nut1
                    ,1)',um(l-1-nut2:-1:l-k-nut2,2)'];
            end
            tau = ones(length(E1),1);

            S1 = ym(tope+1:end,1); S2 = ym(tope+1:end,2); % Salidas para
                regresión
            Em1 = E1; Em2 = E2;

            for l = 0:size(Pny,1)*size(Pnu1,1)*size(Pnu2,1)-1
                P = diag([Pny(floor(l/size(Pnu1,1)/size(Pnu2,1))+1,:),...
                    Pnu1(mod(floor(l/size(Pnu2,1)),size(Pnu1,1))+1,:),...
                    Pnu2(mod(l,size(Pnu2,1))+1,:)]);

                theta1 = ridge(E1,S1,tau,P);
                theta2 = ridge(E2,S2,tau,P);

                for m = 2:length(E1)
                    Em1(m,1) = Em1(m-1,:)*theta1; % yhat_i
                    Em2(m,1) = Em2(m-1,:)*theta2;
                    Em1(m,2:i) = Em1(m-1,1:i-1); % Actualizar valores
                        pasados de sonda i
                    Em2(m,2:i) = Em2(m-1,1:i-1);
                end

                Err1 = sum((ym(tope:end-1,1)-Em1(:,1)).^2)/size(Em1,1);
                if Errcuad1(i,j,k) > Err1
                    Modres1{i,j,k} = Em1(:,1);
                    Errcuad1(i,j,k) = Err1;
                    Thetareg1{i,j,k}= theta1;
                    Preg1{i,j,k} = P;
                end
```

```
                Err2 = sum((ym(tope:end-1,2)-Em2(:,1)).^2)/size(Em2,1);
                if Errcuad2(i,j,k) > Err2
                    Modres2{i,j,k} = Em2(:,1);
                    Errcuad2(i,j,k) = Err2;
                    Thetareg2{i,j,k} = theta2;
                    Preg2{i,j,k} = P;
                end
            end
        end
    end
end

%% Validacion

load CHIRP_DATOS.mat
[uv,yv] = filtro(h1s,h2s,t1s,t2s,Tm); % Filtrado y muestreado
yv = [yv(1:472,:);yv(900:1485,:)];
uv = [uv(1:472,:);uv(900:1485,:)];
mediasv = mean(yv,1);
yvm = yv-repmat(mediasv,length(yv),1);
uvm = uv-repmat(PO,length(uv),1);
Valres1 = cell(ny,nu1,nu2);
Valres2 = Valres1;
Errval1 = 1e6*ones(ny,nu1,nu2);
Errval2 = Errval1;
for i = 1:ny
    for j = 1:nu1
        for k = 1:nu2
            tope = max([i+nyt,j+nut1,k+nut2]);
            Ev1 = zeros(length(yvm)-tope,i+j+k);
            for l = 1:j
                Ev1(:,i+l) = uvm((tope-l+1:end-l)-nut1,1);
            end
            for l =1:k
                Ev1(:,i+j+l) = uvm((tope-l+1:end-l)-nut2,2);
            end
            Ev2 = Ev1;

            Ev1(1,1:i) = yvm((tope:-1:tope+1-i)-nyt,1)';
            Ev2(1,1:i) = yvm((tope:-1:tope+1-i)-nyt,2)';

            theta1 = Thetareg1{i,j,k};
            theta2 = Thetareg2{i,j,k};

            for l = 2:length(Ev1)
                Ev1(l,1) = Ev1(l-1,:)*theta1; % yhat_i
                Ev2(l,1) = Ev2(l-1,:)*theta2;
                Ev1(l,2:i) = Ev1(l-1,1:i-1); % Actualizar valores pasados
                    de sonda i
                Ev2(l,2:i) = Ev2(l-1,1:i-1);
```

```
            end
            Valres1{i,j,k} = Ev1(:,1);
            Valres2{i,j,k} = Ev2(:,1);
            Errval1(i,j,k) = sum((yvm(tope+1:end,1)-Ev1(:,1)).^2)/size(
                Ev1,1);
            Errval2(i,j,k) = sum((yvm(tope+1:end,2)-Ev2(:,1)).^2)/size(
                Ev2,1);
        end
    end
end

%% Selección de modelo

Par = zeros(ny,nu1,nu2);
for i = 1:ny
    for j = 1:nu1
        for k = 1:nu2
            Par(i,j,k) = i+j+k;
        end
    end
end
Par = nthroot(Par,10);

Parms1 = Errcuad1.*Par;
Parms2 = Errcuad2.*Par;
Parvs1 = Errval1.*Par;
Parvs2 = Errval2.*Par;

lista1 = 1e6*ones(6,5);
lista2 = lista1;
aux1 = Parms1+Parvs1;
aux2 = Parms2+Parvs2;
for i = 1:ny
    for j = 1:nu1
        for k = 1:nu2
            flag1 = true; flag2 = true;
            for l = 1:size(lista1,2)
                if lista1(end,l)>aux1(i,j,k) && flag1
                    lista1(:,l+1:end) = lista1(:,l:end-1);
                    lista1(:,l) = [i;j;k;Parms1(i,j,k);Parvs1(i,j,k);aux1(
                        i,j,k)];
                    flag1 = false;
                end
                if lista2(end,l)>aux2(i,j,k) && flag2
                    lista2(:,l+1:end) = lista2(:,l:end-1);
                    lista2(:,l) = [i;j;k;Parms2(i,j,k);Parvs2(i,j,k);aux2(
                        i,j,k)];
                    flag2 = false;
                end
            end
```

```
        end
    end
end
```

Listing A.6 ridge.m.

```
function theta = ridge(R,Y,tau,P)

D = diag(tau);
DR = D*R;
H = P + (DR')*DR;
theta = H\(DR'*Y);

end
```

Listing A.7 p_theta.m.

```
function Pres = p_theta(P,Top,Len,Ind)

if isempty(P)
    Pres=ones(1,Len);
    Pres=p_theta(Pres,Top,Len,1);
else
    if Ind==1, Cont=Top; else, Cont=P(end,Ind-1); end
    Pres = P;
    Paux = P(end,:);
    for i = 2 : Cont
        Paux(Ind) = Paux(Ind)+1;
        Pres = [Pres;Paux];
        if Ind ~= Len
            Pres=p_theta(Pres,Top,Len,Ind+1);
        end
    end
end

end
```

A.2 Control lineal

Listing A.8 TEST_MPC.m.

```
%% Base de datos
close all;clear;clc;

Kalman = false; % Si se desea probar el filtro de Kalman al final del
    programa
```

```
load MOD_LIN.mat % theta1, theta2, dn, Tm, medias

%% Parámetros

NPlim = 5;
rho = 1;
tol = 1e-3;
MaxIter = 20;

Qel = [0.001, 0.01, 0.1, 1, 10, 100, 1000];
Rel = [0.001, 0.01, 0.1, 1, 10, 100, 1000];
Pel = [0.001, 0.01, 0.1, 1, 10, 100, 1000];

Yref = [60 45 45 60 70; % Referencia para sensor 1
        50 50 40 60 50]; % Referencia para sensor 2
t = 50*ones(1,size(Yref,2)); % Muestras dedicadas a cada pareja de
    referencias

Xsup = [50;50]; % Límites superiores de los estados
Xinf = [-50;-50]; % Límites superiores de los estados

PO = [50;50]; % Valores de consigna para el P.O.
Usup = [100-PO(1);100-PO(2)]; % Límites superiores de las entradas
Uinf = [-PO(1);-PO(2)]; % Límites inferiores de las entradas

%% Espacio de estados

% Dimensiones de los vectores de estado y entrada
ns = size(dn,2);
nx1 = dn(1,1);
nx2 = dn(1,2);
nu1 = max(dn(2,:));
nu2 = max(dn(3,:));
n = max([nx1,nx2,nu1,nu2]);
nx = ns*n;

% Ajuste de regresores
auxth1 = zeros(n*size(dn,1),size(theta1,2));
auxth1(n-nx1+1:n,:) = theta1(nx1:-1:1,:);
auxth1(n+(n-dn(2,1)+1:n),:) = theta1(nx1+(dn(2,1):-1:1),:);
auxth1(2*n+(n-dn(3,1)+1:n),:) = theta1(nx1+dn(2,1)+(dn(3,1):-1:1),:);
theta1=auxth1;

auxth2 = zeros(n*size(dn,1),1);
auxth2(n-nx2+1:n,:) = theta2(nx2:-1:1,:);
auxth2(n+(n-dn(2,2)+1:n),:) = theta2(nx2+(dn(2,2):-1:1),:);
auxth2(2*n+(n-dn(3,2)+1:n),:) = theta2(nx2+dn(2,2)+(dn(3,2):-1:1),:);
theta2=auxth2;
```

```
% Matrices del modelo
theta = zeros(n*size(dn,1),1,2);
theta(:,:,1) = theta1;
theta(:,:,2) = theta2;
[A,B,C] = crea_matrices(theta,n);

%% Límites

Xinf = repmat(Xinf,n,1);
Xsup = repmat(Xsup,n,1);

Ulim = [-Uinf;Usup];

Rminp = [Xinf;Uinf]; % Para MPC
Rmaxp = [Xsup;Usup];

%% Referencias

N = [A-eye(nx),B;C,zeros(ns)];

t(1) = t(1)+n-1;
t(end) = t(end)+NPlim;
Ref = zeros(ns,sum(t)); % Referencias para representación gráfica
aux = 0;
for i = 1:length(t)
    Ref(:,aux+1:aux+t(i)) = Yref(:,i).*ones(2,t(i));
    aux = aux+t(i);
end
Zref = N\[zeros(nx,size(Ref,2));Ref-medias'.*ones(2,size(Ref,2))]; %
    Referencias menos medias

t(end) = t(end)-NPlim;

%% Registros

xsim = zeros(nx,sum(t)+1,sum(1:NPlim),length(Qel),length(Rel),length(Pel
    ));
usim = zeros(ns,sum(t),sum(1:NPlim),length(Qel),length(Rel),length(Pel))
    ;
ysim = zeros(ns,sum(t),sum(1:NPlim),length(Qel),length(Rel),length(Pel))
    ;

t(1) = t(1)-n+1;

%% Creación de F, gmin y gmax

Finic = eye(nx);
Fncp = [zeros(ns,nx),eye(ns)];

% Coeficientes de restriccion terminal de g
```

```
gcoef = [sum(theta1(:,1));sum(theta2(:,1))];

%% Simulación

NPcont=0;
for NP = 1 : NPlim

    gmin = [repmat(Rminp,NP,1);Xinf];
    gmax = [repmat(Rmaxp,NP,1);Xsup];
    lambda = gcoef.^NP;

    AB = repmat({[A,B]},1,NP);
    zeroI = repmat({[zeros(nx,ns),-eye(nx)]},1,NP);
    Fmod = [blkdiag(AB{:}),zeros(nx*NP,nx)] + [zeros(nx*NP,nx),blkdiag(
        zeroI{:})];
    Fbase = [Finic, zeros(nx,size(Fmod,2)-size(Finic,2)); Fmod];
    for NC = 1 : NP
        if NP > NC
            Fnc = repmat({Fncp},1,NP-NC);
            F = [Fbase;[zeros(ns*(NP-NC),(NC-1)*(nx+ns)),-repmat(Fncp,NP-
                NC,1),blkdiag(Fnc{:}),zeros(ns*(NP-NC),nx)]];
        else
            F = Fbase;
        end
        f = zeros(size(F,1),1);

        NPcont = NPcont+1;
        for Qi = 1 : length(Qel)
        Q = Qel(Qi)*eye(nx);
        for Ri = 1 : length(Rel)
        R = Rel(Ri)*eye(ns);
        QR = repmat({blkdiag(Q,R)},1,NP);
        for Pi = 1 : length(Pel)
        P = Pel(Pi)*eye(nx);
        H = 2*blkdiag(QR{:},P);

        z = zeros(NP*(nx+ns)+nx,1);
        iter = n;
        for k = 1:length(t) % Por pareja de referencias
            for i = 1:t(k) % Por muestras asignadas a cada pareja
                zr = [reshape(Zref(:,iter:iter+NP-1),1,[]), Zref(1:nx,
                    iter+NP)'];
                h = (-zr*H)';
                f(1:nx) = z(1:nx);
                gmin(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),
                    iter+NP)...
                    -abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP))
                        .*lambda;
                gmax(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),
                    iter+NP)...
```

```
                    +abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP))
                        .*lambda;
                z = admm(H,h,F,f,gmin,gmax,z,rho,tol,MaxIter);
                usim(:,iter,NPcont,Qi,Ri,Pi) = z(nx+(1:ns));
                aux = [-usim(:,iter,NPcont,Qi,Ri,Pi);usim(:,iter,NPcont,
                    Qi,Ri,Pi)];
                if any(aux >= Ulim)
                    z(nx+ns+(1:nx)) = [A,B]*z(1:nx+ns);
                end
                xsim(:,iter+1,NPcont,Qi,Ri,Pi) = z(nx+ns+(1:nx));
                ysim(:,iter,NPcont,Qi,Ri,Pi) = C*xsim(:,iter,NPcont,Qi,Ri,
                    Pi);
                z = [z(nx+ns+1:end);zeros(nx+ns,1)];
                iter=iter+1;
            end
        end
        end
        end
        end
    end
end

%% Resultados

errormin = 1e6*ones(1,NPcont*length(Qel)*length(Rel)*length(Pel));
QRP = zeros(4,NPcont*length(Qel)*length(Rel)*length(Pel));
for p = 1:NPcont
for i = 1:length(Qel)
    for j = 1:length(Rel)
        for k = 1:length(Pel)
            aux = sum(sum(abs(Ref(:,1:sum(t))-(ysim(:,n:end,p,i,j,k)+
                medias'))));
            flag = true;
            for l = 1:length(QRP)
                if errormin(l)>aux && flag
                    errormin(l+1:end) = errormin(l:end-1);
                    errormin(l) = aux;
                    QRP(:,l+1:end) = QRP(:,l:end-1);
                    QRP(:,l) = [p;i;j;k];
                    flag = false;
                end
            end
        end
    end
end
end

for i = 1:10
    plot((ysim(:,n:end,QRP(1,i),QRP(2,i),QRP(3,i),QRP(4,i))+medias')')
    hold on
```

```
end
plot(Ref(:,1:sum(t))')
```

Listing A.9 crea_matrices.m.

```
function [A,B,C] = crea_matrices(theta,n,F_1,F_2,Y)

[~,d2,d3] = size(theta);

if nargin > 2
    tau_1 = fuzzyficacion(Y(1),F_1);
    tau_2 = fuzzyficacion(Y(2),F_2);
    tau = [tau_1;tau_2];
else
    tau = ones(d3,d2);
end

Abase = [zeros(d3,(n-1)*d3);eye((n-1)*d3)];
Ath = zeros(n*d3,d3);
B = Ath;
C = [zeros(d3,(n-1)*d3),eye(d3)];

for i = 1:n
    for j = 1:d3
        Ath((i-1)*d3+j,j) = tau(j,:)*theta(i,:,j)'/sum(tau(j,:));
        B((i-1)*d3+j,:) = tau(j,:)*[theta(i+n,:,j)',theta(i+2*n,:,j)']/
            sum(tau(j,:));
    end
end

A = [Abase Ath];

end
```

Listing A.10 admm.m.

```
function z = admm(H,h,F,f,gmin,gmax,z0,rho,tol,MaxIter)

lambda = zeros(length(z0),1);
K = [H+rho*eye(size(H)), F'; F, zeros(size(F,1))]^(-1);
z = z0; zk = z0;

Pin = true; k = 1;

while Pin && k <= MaxIter
    v = K*[lambda+rho*z-h;f];
    v = v(1:length(z0));
    z = min(gmax,max(v-lambda/rho,gmin));
    lambda = lambda+rho*(z-v);
```

```
    Pin = max(max(abs(z-v)),max(abs(z-zk)))>=tol;
    zk = z;
    k = k+1;
end

end
```

Listing A.11 ENSAYO_MPC.m.

```
%% Inicialización
close all; clear; clc

tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2

load MOD_LIN.mat % theta1, theta2, dn, Tm, medias

%% Parámetros

NP = 4;
NC = 2;
Q = 10;
R = 0.1;
P = 1;

rho = 1;
tol = 1e-3;
MaxIter = 20;

Yref = [60 45 45 60 70; % Referencias para sensor 1
        50 50 40 60 50]; % Referencias para sensor 2

t = 50*ones(1,size(Yref,2)); % Duración de cada pareja de referencias
inic = 60*60; % Duración de espera para P.O.

Xsup = [50;50]; % Límites superiores de los estados
Xinf = [-50;-50]; % Límites superiores de los estados

PO = [50;50]; % Entradas para el punto de operación
Usup = [100-PO(1);100-PO(2)]; % Límites superiores de las entradas
Uinf = [-PO(1);-PO(2)]; % Límites inferiores de las entradas

%% Espacio de estados
```

```
% Dimensiones de los vectores de estado y entrada
ns = size(dn,2);
nx1 = dn(1,1);
nx2 = dn(1,2);
nu1 = max(dn(2,:));
nu2 = max(dn(3,:));
n = max([nx1,nx2,nu1,nu2]);
nx = ns*n;

% Ajuste de regresores
auxth1 = zeros(n*size(dn,1),size(theta1,2));
auxth1(n-nx1+1:n,:) = theta1(nx1:-1:1,:);
auxth1(n+(n-dn(2,1)+1:n),:) = theta1(nx1+(dn(2,1):-1:1),:);
auxth1(2*n+(n-dn(3,1)+1:n),:) = theta1(nx1+dn(2,1)+(dn(3,1):-1:1),:);
theta1=auxth1;

auxth2 = zeros(n*size(dn,1),1);
auxth2(n-nx2+1:n,:) = theta2(nx2:-1:1,:);
auxth2(n+(n-dn(2,2)+1:n),:) = theta2(nx2+(dn(2,2):-1:1),:);
auxth2(2*n+(n-dn(3,2)+1:n),:) = theta2(nx2+dn(2,2)+(dn(3,2):-1:1),:);
theta2=auxth2;

% Matrices del modelo
theta = zeros(n,3,2);
theta(:,:,1) = theta1;
theta(:,:,2) = theta2;
[A,B,C] = crea_matrices(theta);

%% Limites

Xinf = repmat(Xinf,n,1);
Xsup = repmat(Xsup,n,1);

Ulim = [-Uinf;Usup];

Rminp = [Xinf;Uinf]; % Para MPC
Rmaxp = [Xsup;Usup];

%% Matrices MPC

% Creación de H
Q = Q*eye(nx);
R = R*eye(ns);
P = P*eye(nx);
QR = repmat({blkdiag(Q,R)},1,NP);
H = blkdiag(QR{:},P);

% Creación de F

Finic = eye(nx);
```

```
Fncp = [zeros(ns,nx),eye(ns)];

AB = repmat({[A,B]},1,NP);
zeroI = repmat({[zeros(nx,ns),-eye(nx)]},1,NP);
Fmod = [blkdiag(AB{:}),zeros(nx*NP,nx)] + [zeros(nx*NP,nx),blkdiag(zeroI
    {:})];
F = [Finic, zeros(nx,size(Fmod,2)-size(Finic,2)); Fmod];
if NP > NC
    FNC = repmat({Fncp},1,NP-NC);
    F = [F;[zeros(ns*(NP-NC),(NC-1)*(nx+ns)),-repmat(Fncp,NP-NC,1),
        blkdiag(FNC{:}),zeros(ns*(NP-NC),nx)]];
end

f = zeros(size(F,1),1);

% Coeficientes de restriccion terminal de g
gmin = [repmat(Rminp,NP,1);Xinf];
gmax = [repmat(Rmaxp,NP,1);Xsup];
gcoef = [sum(theta1(:,1));sum(theta2(:,1))];
lambda = gcoef.^NP;

%% Espera para alcanzar el P.O.

h1(PO(1));
h2(PO(2));

for i = 1:inic
    tic;
    t1 = T1C();
    t2 = T2C();
    h1s = [h1s,PO(1)];
    h2s = [h2s,PO(2)];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    clf
    plot(t1s,'r.')
    hold on
    plot(t2s,'b.')
    treg = toc;
    pause(max(0.01,1.0-treg))
end

%% Referencias

medias = [mean(t1s(end-30:end));mean(t2s(end-30:end))]; % P.O.

N = [A-eye(nx),B;C,zeros(ns)];

t(1) = t(1)+n-1;
t(end) = t(end)+NPlim;
```

```
Ref = zeros(ns,sum(t)); % Referencias para representación gráfica
aux = 0;
for i = 1:length(t)
    Ref(:,aux+1:aux+t(i)) = Yref(:,i).*ones(2,t(i));
    aux = aux+t(i);
end
Zref = N\[zeros(nx,size(Ref,2));Ref-medias'.*ones(2,size(Ref,2))]; %
    Referencias menos medias

t(end) = t(end)-NPlim;

%% Registros

x = zeros(nx,sum(t)+1);
y = zeros(ns,sum(t));
u = zeros(ns,sum(t));

t(1) = t(1)-n+1;

%% Ensayo de MPC

iter = n; % Índice del vector de entrada u
cont = 0; % Índice de t de temperatura tomadas
reg = []; % Registro del tiempo de cálculo

x((n-1)*ns+(1:ns),iter) = [t1;t2]-medias;
y(:,iter) = [t1;t2];
z = zeros(NP*(nx+ns)+nx,1);
z(2:n) = t1s(end-(nx1-1)*Tm:Tm:end-Tm)'-medias(1);
z(n+(2:n)) = t2s(end-(n-1)*Tm:Tm:end-Tm)'-medias(2);

disp('Comienzo ensayo MPC')
for j = 1:length(t)
    for i = 1:Tm*t(j)
        tic;
        t1 = T1C();
        t2 = T2C();
        t1s = [t1s,t1];
        t2s = [t2s,t2];

        if mod(i,Tm) == 0
            z((n-1)*ns+(1:ns)) = [t1;t2]-medias;
            zr = [reshape(Zref(:,iter:iter+NP-1),1,[]), Zref(1:nx,iter+NP
                )'];
            h = (-zr*H)';
            f(1:nx) = z(1:nx);
            gmin(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),iter+
                NP)...
                -abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP)).*
                    lambda;
```

```
            gmax(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),iter+
                NP)...
                +abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP)).*
                    lambda;
            z = admm(H,h,F,f,gmin,gmax,z,rho,tol,MaxIter);
            u(:,iter) = [z(nx+totbc(2:end))];
            h1(u(1,iter)+PO(1));
            h2(u(2,iter)+PO(2));
            aux = [-u(:,iter);u(:,iter)];
            if any(aux >= Ulim)
                z(nx+ns+(1:nx)) = [A,B]*z(1:nx+ns);
            end
            x(:,iter+1) = z(nx+ns+(1:nx));
            z = [z(nx+ns+1:end);zeros(nx+ns,1)];
            iter=iter+1;
        end

        h1s = [h1s,u(1,iter-1)+PO(1)];
        h2s = [h2s,u(2,iter-1)+PO(2)];

        % Representación gráfica
        cont = cont+1;
        time = linspace(0,cont+1,cont);
        clf
        subplot(2,1,1)
        plot(time,t1s(inic+1:end),'r.')
        hold on
        plot(time,t2s(inic+1:end),'b.')
        plot(time,Ref(1,1:cont),'g--')
        plot(time,Ref(2,1:cont),'k--')
        ylabel('Temperature (degC)')
        legend('Temperature 1','Temperature 2','Location','NorthWest')
        subplot(2,1,2)
        plot(time,h1s(inic+1:end),'r')
        hold on
        plot(time,h2s(inic+1:end),'b')
        ylabel('Heater (0-5.5 V)')
        xlabel('Time (sec)')
        legend('Heater 1','Heater 2','Location','NorthWest')
        drawnow;
        treg = toc;
        reg = [reg treg];
        pause(max(0.01,1.0-treg))
    end
end

h1(0); h2(0); % Apaga calentadores

disp('Fin ensayo MPC')
```

A.3 Modelo no lineal

Listing A.12 ENSAYO_PSEUDOPRBS.m.

```
%% Inicialización
close all; clear; clc

tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2

PO=[0;0]; % Entradas para el P.O.

Tm=13*5; % Tiempo de muestro
dur = ceil(8*3600/Tm); % Duracion del ensayo
inic=60*60; % Duración espera para P.O.

[u,t]=PRBS(8,Tm,13,1); % Generacion de la señal PRBS
u = u(1:dur);
u1 = u; u2 = u;
estados = 3^2; % Núumero de combinaciones de consignas para las
    entradas
estreg = zeros(estados,1); % Registro de presencia de los estados
cambio = u(1);
j = 1;
for i = 1:dur
    if u(i) ~= cambio % Se rige por los flancos de una señal PRBS
        cambio = u(i); % Vale 0 o 1
        j = find(estreg==min(estreg)); % La siguiente consigna es la que
             ha tenido menor presencia
        ind = randi(length(j)); % En caso de empate, se escoge uno de
            ellos aleatoriamente
        j = j(ind);
    end
    estreg(j) = estreg(j)+1;
    u1(i) = 50*(ceil(j/3)-1);
    u2(i) = 50*mod(j-1,3);
end

%% Espera para llegar al P.O.

h1(PO(1));
h2(PO(2));

for i = 1:inic
```

```
    tic;
    t1 = T1C();
    t2 = T2C();
    h1s = [h1s,PO(1)];
    h2s = [h2s,PO(2)];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    clf
    plot(t1s,'r.');
    hold on
    plot(t2s,'b.');
    t = toc;
    pause(max(0.01,1.0-t))
end

%% Ensayo PseudoPRBS

n=0;

disp('Comienzo ensayo PRBS')
for i = 0:dur*Tm-1
    tic;
    if mod(i,Tm) == 0 % Cada tiempo de muestreo
        ht1 = u1(i/Tm+1);
        h1(ht1);
        ht2 = u2(i/Tm+1));
        h2(u2(i/Tm+1));
    end

    t1 = T1C();
    t2 = T2C();
    h1s = [h1s,ht1];
    h2s = [h2s,ht2];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    n = n+1;
    time = linspace(0,n+1,n);
    clf
    subplot(2,1,1)
    plot(time,t1s(inic+1:end),'r.');
    hold on
    plot(time,t2s(inic+1:end),'b.');
    ylabel('Temperature (degC)')
    legend('Temperature 1','Temperature 2','Location','NorthWest')
    subplot(2,1,2)
    plot(time,h1s(inic+1:end),'r-','LineWidth',2);
    hold on
    plot(time,h2s(inic+1:end),'b--','LineWidth',2);
    ylabel('Heater (0-5.5 V)')
    xlabel('Time (sec)')
```

```
    legend('Heater 1','Heater 2','Location','NorthWest')
    drawnow;
    t = toc;
    pause(max(0.01,1-t))
end

h1(0); h2(0); % Apaga calentadores

disp('Fin ensayo PRBS')
```

Listing A.13 IDENT_NL.m.

```
%% Base de datos
close all; clear; clc

Valid = true; % Si se desea una validación posterior del modelo

load IDENT_NL_DATOS.mat % Datos de entrenamiento brutos
[u,y] = filtrofuzzy(h1s,h2s,t1s,t2s,Tm,time); % Filtrado y muestreado

medias = mean(y(1:10,:),1);
PO = [0 0];

ym = y-repmat(medias,length(y),1); % Salidas menos P.O.
um = u-repmat(PO,length(u),1); % Entradas menos entradas para P.O.

%% Parametros

Tritrap =true;
% False: Triangular
% True: Trapezoidal

ny = 3; % Parámetros máximos
nyt = 0; % Valores pasados de la sonda
nu1 = 3; % Valores pasados del calentador a la se le que acopla la
    sonda i
nu2 = 3;
nut1 = 0; % Retraso en entradas
nut2 = 0;

MaxDivis = 4;
MaxTemp1 = 70;
MaxTemp2 = 55;

%% Funciones de membresia

a = 0.3; b = 0.7;
ubic = @(x) a*x.^2 + b*x;
```

```
Divis1 = cell(MaxDivis-1,1);
Divis2 = Divis1;

if Tritrap
    for i = 2:MaxDivis
        Faux = zeros(i,4,2);
        esp = linspace(0,1,i*2);
        esp1 = MaxTemp1*ubic(esp); % Punto de referencia
        esp2 = MaxTemp2*ubic(esp); % Punto de referencia
        for j = 1:i
            if j == 1
                Faux(1,2:4,1) = esp1(1:3);
                Faux(1,2:4,2) = esp2(1:3);
            elseif j == i
                Faux(end,:,1) = [esp1(end-2:end), MaxTemp1];
                Faux(end,:,2) = [esp2(end-2:end), MaxTemp2];
            else
                Faux(j,:,1) = esp1((j-2)*2+1+(1:4));
                Faux(j,:,2) = esp2((j-2)*2+1+(1:4));
            end
        end
        Divis1{i-1} = Faux(:,:,1);
        Divis2{i-1} = Faux(:,:,2);
    end
else
    for i = 2:MaxDivis
        Faux = zeros(i,3,2);
        esp = linspace(0,1,i);
        esp1 = MaxTemp1*ubic(esp); % Punto de referencia
        esp2 = MaxTemp2*ubic(esp); % Punto de referencia
        for j = 1:i
            if j == 1
                Faux(1,2:3,1) = esp1(1:2);
                Faux(1,2:3,2) = esp2(1:2);
            elseif j == i
                Faux(end,:,1) = [esp1(end-1:end), MaxTemp1];
                Faux(end,:,2) = [esp2(end-1:end), MaxTemp2];
            else
                Faux(j,:,1) = esp1(j-2+(1:3));
                Faux(j,:,2) = esp2(j-2+(1:3));
            end
        end
        Faux = [Faux(:,1:2,:),Faux(:,2,:),Faux(:,3,:)];
        Divis1{i-1} = Faux(:,:,1);
        Divis2{i-1} = Faux(:,:,2);
    end
end

%% Identificación
```

```
Thetareg1 = cell(ny,nu1,nu2,MaxDivis-1);
Thetareg2 = Thetareg1;

for i = 1:ny
    for j = 1:nu1
        for k = 1:nu2

            tope = max([i+nyt,j+nut1,k+nut2]);
            E1 = zeros(length(ym)-tope,i+j+k);
            E2 = E1;
            for l = tope+1:length(ym)
                E1(l-tope,:) = [ym(l-1:-1:l-i,1)',um(l-1-nut1:-1:l-j-nut1
                    ,1)',um(l-1-nut2:-1:l-k-nut2,2)'];
                E2(l-tope,:) = [ym(l-1:-1:l-i,2)',um(l-1-nut1:-1:l-j-nut1
                    ,1)',um(l-1-nut2:-1:l-k-nut2,2)'];
            end

            S1 = ym(tope+1:end,1); S2 = ym(tope+1:end,2); % Salidas para
                regresión
            Em1 = E1; Em2 = E2;

            P = eye(i+j+k);
            for l = 1:MaxDivis-1
                tau_1 = fuzzyficacion(E1(:,1),Divis1{l});
                tau_2 = fuzzyficacion(E2(:,1),Divis2{l});
                theta1 = zeros(i+j+k,l+1);
                theta2 = theta1;

                I1 = cell(l+1,1); I2 = I1;
                for m = 1:l+1
                    I1{m} = find(tau_1(:,m)>0);
                    I2{m} = find(tau_2(:,m)>0);
                end
                for n = 1:l+1
                    theta1(:,n) = ridge(E1(I1{n},:),S1(I1{n}),tau_1(I1{n},
                        n),P);
                    theta2(:,n) = ridge(E2(I2{n},:),S2(I2{n}),tau_2(I2{n},
                        n),P);
                end
                Thetareg1{i,j,k,l} = theta1;
                Thetareg2{i,j,k,l} = theta2;
            end
        end
    end
end

%% Cálculo del error

Errmod1 = cell(MaxDivis-1,1);
Errmod2 = Errmod1;
```

```
tope = max([ny+nyt,nu1+nut1,nu2+nut2]);

Em1 = zeros(length(ym)-tope,ny+nu1+nu2);
for l = 1:nu1
    Em1(:,ny+l) = um((tope-l+1:end-l)-nut1,1);
end
for l =1:nu2
    Em1(:,ny+nu1+l) = um((tope-l+1:end-l)-nut2,2);
end
Em2 = Em1;
Em1(1,1:ny) = ym((tope:-1:tope+1-ny)-nyt,1)';
Em2(1,1:ny) = ym((tope:-1:tope+1-ny)-nyt,2)';

for l = 1:MaxDivis-1
    Errmod1{l} = error_nl(zeros((ny*nu1*nu2)*ones(1,l+1)),Thetareg1
        (:,:,:,l),...
        1,zeros(l+1,3),ym(tope:end-1,1),Em1,Divis1{l},[ny,nu1,nu2]);
    Errmod2{l} = error_nl(zeros((ny*nu1*nu2)*ones(1,l+1)),Thetareg2
        (:,:,:,l),...
        1,zeros(l+1,3),ym(tope:end-1,2),Em2,Divis2{l},[ny,nu1,nu2]);
end

%% Validacion

load CHIRP_DATOS.mat
[uv,yv] = filtro(h1s,h2s,t1s,t2s,Tm); % Filtrado y muestreado
yv = [yv(1:472,:);yv(900:1485,:)];
uv = [uv(1:472,:);uv(900:1485,:)];
mediasv = [t1s(1), t2s(1)];
yvm = yv-repmat(mediasv,length(yv),1);
uvm = uv-repmat(PO,length(uv),1);

Errval1 = cell(MaxDivis-1,1);
Errval2 = Errval1;

Ev1 = zeros(length(yvm)-tope,ny+nu1+nu2);
for l = 1:nu1
    Ev1(:,ny+l) = uvm((tope-l+1:end-l)-nut1,1);
end
for l =1:nu2
    Ev1(:,ny+nu1+l) = uvm((tope-l+1:end-l)-nut2,2);
end
Ev2 = Ev1;
Ev1(1,1:ny) = yvm((tope:-1:tope+1-ny)-nyt,1)';
Ev2(1,1:ny) = yvm((tope:-1:tope+1-ny)-nyt,2)';

for l = 1:MaxDivis-1
    Errval1{l} = ErrorNL(zeros((ny*nu1*nu2)*ones(1,l+1)),Thetareg1(:,:,:,
        l),...
        1,zeros(l+1,3),yvm(tope:end-1,1),Ev1,Divis1{l},[ny,nu1,nu2]);
```

```
    Errval2{l} = ErrorNL(zeros((ny*nu1*nu2)*ones(1,l+1)),Thetareg2(:,:,:,
        l),...
        1,zeros(l+1,3),yvm(tope:end-1,2),Ev2,Divis2{l},[ny,nu1,nu2]);
end

%% Selección de modelo

top5 = zeros(length(Errmod1),5);
Coord = cell(length(Errmod1),1);
for i = 1:length(Errmod1)
    Par = creaPar([ny,nu1,nu2,i+1]);
    Parm = Errmod2{i}.*Par;
    Parv = Errval2{i}.*Par;
    [top5(i,:),Coord{i}] = buscaminimo(Parm+Parv);
end
```

Listing A.14 fuzzyfication.m.

```
function tau = fuzzyficacion(Y,F)

numest = size(Y,2); % Número de salidas
numfunc = size(F,1); % Número de funciones de membresía
numreg = numfunc^numest; % Número de reglas
tau = zeros(size(Y,1),numreg);

for i = 1:size(Y,1)
    yk = Y(i,:);
    pk = zeros(numest,numfunc);
    for j = 1:numest
        for k = 1:numfunc
            pk(j,k) = membresia_trapezoide(yk(j),F,k,numfunc);
        end
    end
    tau(i,:) = crea_tau(pk,numest,numfunc,0,0);
end
```

Listing A.15 crea_tau.m.

```
function tau=crea_tau(pk,M,N,j,k)
    if k == 0
        tau = zeros(1,N^M);
        for i = 1:N^M
            tau(i) = crea_tau(pk,M,N,i,1);
        end
    elseif k > M
        tau = 1;
    else
        res = crea_tau(pk,M,N,mod(j-1,N^(M-k))+1,k+1);
        tau = res*pk(k,ceil(j/N^(M-k)));
```

```
    end
end
```

Listing A.16 membresia_trapezoide.m.

```
function mu = membresia_trapezoide(y,F,i,N)

if i == 1 % Primera función
    pend1 = (F(i,4)-y)/(F(i,4)-F(i,3));
    mu = (y<F(i,3))+pend1*(y>=F(i,3) && y<=F(i,4));
elseif i == N % Última función
    pend1 = (y-F(i,1))/(F(i,2)-F(i,1));
    mu = (y>F(i,2))+pend1*(y>=F(i,1) && y<=F(i,2));
else % Resto de funciones
    pend1 = (y-F(i,1))/(F(i,2)-F(i,1));
    pend2 = (F(i,4)-y)/(F(i,4)-F(i,3));
    mu = pend1*(y>=F(i,1) && y<=F(i,2))+(y>=F(i,2) && y<=F(i,3))+pend2*(
        y>=F(i,3) && y<=F(i,4));
end
```

Listing A.17 error_nl.m.

```
function [Mat,Ind] = error_nl(Mat0,Theta,Nivel,Ind0,y,E,Func,dn)

Ind = Ind0;
Mat = Mat0;

if Nivel == size(Func,1)
    % Prepara theta
    dN = [0 dn];
    aux = [0, ones(1,length(dn)-1)];
    mult = [1; dn(1); dn(1)*dn(2)];
    reg = zeros(sum(dn),size(Func,1));
    for m = 1:size(Func,1)-1
        th = Theta{(Ind(m,:)-aux)*mult};
        dInd = [0 Ind(m,:)];
        for n = 1:length(dn)
            reg(sum(dN(1:n))+(1:Ind(m,n)),m) = th(sum(dInd(1:n))+(1:Ind(m
                ,n)),m);
        end
    end

    % Respuesta y error
    auxe = [0, ones(1,size(Func,1)-1)];
    multe = ones(size(Func,1),1);
    for m = 2:size(Func,1)
        multe(m) = multe(m-1)*prod(dn);
    end
    for i = 1:dn(1)
```

```
        Ind(Nivel,1) = i;
        for j = 1:dn(2)
            Ind(Nivel,2) = j;
            for k = 1:dn(3)
                Ind(Nivel,3) = k;
                reg(:,end)=zeros(sum(dn),1);
                th = Theta{(Ind(end,:)-aux)*mult};
                dInd = [0 Ind(end,:)];
                for n = 1:length(dn)
                    reg(sum(dN(1:n))+(1:Ind(end,n)),end) = th(sum(dInd(1:n
                        ))+(1:Ind(end,n)),end);
                end
                for n = 2:length(E)
                    tau = fuzzyficacion(E(n-1,1),Func)';
                    E(n,1) = E(n-1,:)*reg*tau/sum(tau); % yhat_i
                    E(n,2:dn(1)) = E(n-1,1:dn(1)-1); % Actualizar valores
                         pasados de sonda i
                end
                Mat((((Ind-aux)*mult)'-auxe)*multe) = (sum((y-E(:,1)).^2)
                    )/size(E,1);
            end
        end
    end
else
    for i = 1:dn(1)
        Ind(Nivel,1) = i;
        for j = 1:dn(2)
            Ind(Nivel,2) = j;
            for k = 1:dn(3)
                Ind(Nivel,3) = k;
                [Mat,Ind] = ErrorNL(Mat,Theta,Nivel+1,Ind,y,E,Func,dn);
            end
        end
    end
end


end
```

Listing A.18 crea_Par.m.

```
function Par = crea_Par(dn,Par0,Nivel,Ind0)

dim=dn(4);

if nargin == 1
    Par = zeros(prod(dn(1:3))*ones(1,dn(4)));
    Ind = zeros(1,dim);
    Par = creaPar(dn,Par,1,Ind);
```

```
    Par = nthroot(Par/dim,10);
elseif Nivel < dim
    Ind = Ind0;
    Par = Par0;
    for i = 1:prod(dn(1:3))
        Ind(Nivel) = Ind(Nivel)+1;
        Par = creaPar(dn,Par,Nivel+1,Ind);
    end
else
    Ind = Ind0;
    Par = Par0;
    aux = [0 ones(1,dim-1)];
    mult = ones(dim,1);
    for m = 2:dim
        mult(m) = mult(m-1)*prod(dn(1:3));
    end
    for i = 1:prod(dn(1:3))
        Ind(Nivel) = Ind(Nivel)+1;
        trash = Ind;
        for j = 1:Nivel
            trash(j) = mod(Ind(j)-1,dn(1))+1+mod(floor((Ind(j)-1)/dn(1)),
               dn(2))+1+floor((Ind(j)-1)/dn(1)/dn(2))+1;
        end
        Par((Ind-aux)*mult)= sum(trash);
    end
end
```

Listing A.19 busca_minimo.m.

```
function [Res,Coord] = busca_minimo(Mat,Nivel,Coord0,Ind0,Reg)

dim = ndims(Mat);

if nargin == 1
    num = 5;
    dim = ndims(Mat);
    Coord = zeros(dim,num);
    Ind = zeros(dim,1);
    Reg = 1e3*ones(1,num);
    [Res,Coord] = buscaminimo(Mat,1,Coord,Ind,Reg);
elseif Nivel < dim
    Ind = Ind0;
    Coord = Coord0;
    Res = Reg;
    for i = 1:size(Mat,Nivel)
        Ind(Nivel) = Ind(Nivel)+1;
        [Res,Coord] = buscaminimo(Mat,Nivel+1,Coord,Ind,Res);
    end
else
```

```
    Ind = Ind0;
    Coord = Coord0;
    Res = Reg;
    aux = [0 ones(1,dim-1)];
    mult = ones(dim,1);
    for m = 2:dim
        mult(m) = mult(m-1)*size(Mat,1);
    end
    for i = 1:size(Mat,Nivel)
        Ind(Nivel) = Ind(Nivel)+1;
        flag = true;
        for j = 1:length(Res)
            if Res(j) > Mat((Ind'-aux)*mult) && flag
                Res(j+1:end) = Res(j:end-1);
                Res(j) = Mat((Ind'-aux)*mult);
                Coord(:,j+1:end) = Coord(:,j:end-1);
                Coord(:,j) = Ind;
                flag = false;
            end
        end
    end
end

end
```

A.4 Control no lineal

Listing A.20 TEST_MPC_NL.m.

```
%% Base de datos
close all; clear; clc;

% load MPC_DATOS_TRI.mat
load MPC_DATOS_TRAP.mat

%% Parámetros

NPlim = 1;
rho = 1;
tol = 1e-3;
MaxIter = 20;

Yref = [60 45 45 60 70; % Referencia para sensor 1
        50 50 40 60 50]; % Referencia para sensor 2
t = 50*ones(1,size(Yref,2)); % Muestras dedicadas a cada pareja de
    referencias

Xsup = [100;100]; % Límites superiores de los estados
```

```
Xinf = [-5;-5]; % Límites superiores de los estados

PO = [0;0];
Usup = [100-PO(1);100-PO(2)]; % Límites superiores de las entradas
Uinf = [-PO(1);-PO(2)]; % Límites inferiores de las entradas

%% Datos

%Dimensiones de los vectores de estado, salida ysim entrada
ns = size(dn,2);
nx1 = dn(1,1);
nx2 = dn(1,2);
nu1 = max(dn(2,:));
nu2 = max(dn(3,:));
n = max([nx1,nx2,nu1,nu2]);
nx = ns*n;

% Ajuste de regresores
auxth1 = zeros(n*size(dn,1),size(theta1,2));
auxth1(n-nx1+1:n,:) = theta1(nx1:-1:1,:);
auxth1(n+(n-dn(2,1)+1:n),:) = theta1(nx1+(dn(2,1):-1:1),:);
auxth1(2*n+(n-dn(3,1)+1:n),:) = theta1(nx1+dn(2,1)+(dn(3,1):-1:1),:);
theta1=auxth1;

auxth2 = zeros(n*size(dn,1),size(theta2,2));
auxth2(n-nx2+1:n,:) = theta2(nx2:-1:1,:);
auxth2(n+(n-dn(2,2)+1:n),:) = theta2(nx2+(dn(2,2):-1:1),:);
auxth2(2*n+(n-dn(3,2)+1:n),:) = theta2(nx2+dn(2,2)+(dn(3,2):-1:1),:);
theta2=auxth2;

theta = zeros(n*size(dn,1),size(Func1,1),ns);
theta(:,:,1) = theta1;
theta(:,:,2) = theta2;

%% Límites

Xinf = repmat(Xinf,n,1);
Xsup = repmat(Xsup,n,1);

Ulim = [-Uinf;Usup];

Rminp = [Xinf;Uinf]; % Para MPC
Rmaxp = [Xsup;Usup];

%% Referencias

t(1) = t(1)+n-1;
t(end) = t(end)+NPlim;

Ref = zeros(ns,sum(t)); % Referencias para representación gráfica
```

```
Zref = zeros(nx+ns,sum(t));
aux = 0;
for i = 1:length(t)
    Ref(:,aux+1:aux+t(i)) = Yref(:,i).*ones(ns,t(i));
    [A,B,C] = crea_matrices(theta,n,Func1,Func2,Yref(:,i)-medias');
    N = [A-eye(nx), B; C, zeros(ns)];
    Zref(:,aux+1:aux+t(i)) = N\[zeros(nx,t(i));Ref(:,aux+1:aux+t(i))-
        medias'];
    aux = aux+t(i);
end

t(end) = t(end)-NPlim;

%% Matrices de regulación

Qel = [0.001, 0.01, 0.1, 1, 10, 100, 1000];
Rel = [0.001, 0.01, 0.1, 1, 10, 100, 1000];
Pel = [0.001, 0.01, 0.1, 1, 10, 100, 1000];

%% Registros de simulación

xsim = zeros(nx,sum(t)+1,sum(1:NPlim),length(Qel),length(Rel),length(Pel
    ));
usim = zeros(ns,sum(t),sum(1:NPlim),length(Qel),length(Rel),length(Pel))
    ;
ysim = zeros(ns,sum(t),sum(1:NPlim),length(Qel),length(Rel),length(Pel))
    ;

t(1) = t(1)-n+1;

%% Módulos de F

Finic = eye(nx);
Fncp = [zeros(ns,nx),eye(ns)];

STR = struct('theta',theta,'Func1',Func1,'Func2',Func2,'n',n,'ns',ns,'nx
    ',nx,'C',C);

%% Simulacion

NPcont=0;
for NP = 1 : NPlim

    gmin = [repmat(Rminp,NP,1);Xinf];
    gmax = [repmat(Rmaxp,NP,1);Xsup];

    zeroI = repmat({[zeros(nx,ns),-eye(nx)]},1,NP);
    Fmod = [zeros(nx*NP,nx),blkdiag(zeroI{:})];
    Fbase = [Finic, zeros(nx,size(Fmod,2)-size(Finic,2)); Fmod];
```

```
for NC = 1 : NP
    if NP > NC
        Fnc = repmat({Fncp},1,NP-NC);
        F = [Fbase;[zeros(ns*(NP-NC),(NC-1)*(nx+ns)),-repmat(Fncp,NP-
            NC,1),blkdiag(Fnc{:}),zeros(ns*(NP-NC),nx)]];
    else
        F = Fbase;
    end
    f = zeros(size(F,1),1);

    NPcont = NPcont+1;
    for Qi = 1 : length(Qel)
    Q = Qel(Qi)*eye(nx);
    for Ri = 1 : length(Rel)
    R = Rel(Ri)*eye(ns);
    QR = repmat({blkdiag(Q,R)},1,NP);
    for Pi = 1 : length(Pel)
    P = Pel(Pi)*eye(nx);
    H = blkdiag(QR{:},P);

    z = zeros(NP*(nx+ns)+nx,1);
    iter = n;
    for k = 1:length(t) % Por pareja de referencias
        for i = 1:t(k) % Por muestras asignadas a cada pareja
            [A,B] = crea_matrices(theta,n,Func1,Func2,C*z(1:nx));
            F(nx+(1:nx),1:nx+ns) = [A,B];
            zr = [reshape(Zref(:,iter+(0:NP-1)),1,[]), Zref(1:nx,iter+
                NP)'];
            h = (-zr*H)';
            f(1:nx) = z(1:nx);
            gcoef = [sum(A(:,(n-1)*ns+1));sum(A(:,n*ns))];
            lambda = gcoef.^NP;
            gmin(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),
                iter+NP)...
                -abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP))
                    .*lambda;
            gmax(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),
                iter+NP)...
                +abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP))
                    .*lambda;
            z = admm_nl(H,h,F,f,STR,z,NP,gmin,gmax,rho,tol,MaxIter);
            usim(:,iter,NPcont,Qi,Ri,Pi) = z(nx+(1:ns));
            aux = [-usim(:,iter,NPcont,Qi,Ri,Pi);usim(:,iter,NPcont,
                Qi,Ri,Pi)];
            if any(aux >= Ulim)
                z(nx+ns+(1:nx)) = [A,B]*z(1:nx+ns);
            end
            xsim(:,iter+1,NPcont,Qi,Ri,Pi) = z(nx+ns+(1:nx));
            ysim(:,iter,NPcont,Qi,Ri,Pi) = C*z(1:nx);
            z = [z(nx+ns+1:end);zeros(nx+ns,1)];
```

```
                iter=iter+1;
            end
        end
        end
        end
        end
    end
end

%% Error

errormin = 1e6*ones(1,NPcont*length(Qel)*length(Rel)*length(Pel));
QRP = zeros(4,NPcont*length(Qel)*length(Rel)*length(Pel));
for p = 1:NPcont
for i = 1:length(Qel)
    for j = 1:length(Rel)
        for k = 1:length(Pel)
            aux = sum(sum(abs(Ref(:,n+(0:sum(t)-1))-(ysim(:,n:end,p,i,j,k
                )+medias'))));
            flag = true;
            for l = 1:length(QRP)
                if errormin(l)>aux && flag
                    errormin(l+1:end) = errormin(l:end-1);
                    errormin(l) = aux;
                    QRP(:,l+1:end) = QRP(:,l:end-1);
                    QRP(:,l) = [p;i;j;k];
                    flag = false;
                end
            end
        end
    end
end
end

for i = 1:10
    plot((ysim(:,n:end,QRP(1,i),QRP(2,i),QRP(3,i),QRP(4,i))+medias')')
    hold on
end
plot(Ref(:,1:sum(t))')
```

Listing A.21 admm_nl_.m.

```
function z = admm_nl(H,h,F0,f,S,z0,NP,gmin,gmax,rho,tol,MaxIter)

Pin = true;
k = 1;

z = z0; zk = z;
F = F0; C = S.C;
```

```
while Pin && k <= MaxIter
    for i = 1:NP-1
        [A,B,C] = crea_matrices(S.theta,S.n,S.Func1,S.Func2,S.C*z((S.nx+
            S.ns)*i+(1:S.nx)));
        F(S.nx*(i+1)+(1:S.nx),(S.nx+S.ns)*i+(1:S.nx+S.ns)) = [A,B];
    end
    z = admm(H,h,F,f,gmin,gmax,z,rho,tol,MaxIter);
    Pin = max(abs(z-zk))>=0.1;
    zk = z;
    k = k+1;
end

end
```

Listing A.22 ENSAYO_MPC_NL.m.

```
%% Inicialización
close all; clear; clc

tclab;

figure(1)
t1s = []; % Registro de las medidas del sensor 1
t2s = []; % Registro de las medidas del sensor 2
h1s = []; % Registro de las consignas de la entrada 1
h2s = []; % Registro de las consignas de la entrada 2

load MOD_LIN.mat % theta1, theta2, dn, Tm, medias

%% Parámetros

NP = 4;
NC = 2;
Q = 10;
R = 0.1;
P = 1;

rho = 1;
tol = 1e-3;
MaxIter = 20;

Yref = [60 45 45 60 70; % Referencias para sensor 1
        50 50 40 60 50]; % Referencias para sensor 2

t = 50*ones(1,size(Yref,2)); % Duración de cada pareja de referencias
inic = 60*60; % Duración de espera para P.O.

Xsup = [50;50]; % Límites superiores de los estados
Xinf = [-50;-50]; % Límites superiores de los estados
```

```
PO = [50;50]; % Entradas para el punto de operación
Usup = [100-PO(1);100-PO(2)]; % Límites superiores de las entradas
Uinf = [-PO(1);-PO(2)]; % Límites inferiores de las entradas

%% Espacio de estados

% Dimensiones de los vectores de estado y entrada
ns = size(dn,2);
nx1 = dn(1,1);
nx2 = dn(1,2);
nu1 = max(dn(2,:));
nu2 = max(dn(3,:));
n = max([nx1,nx2,nu1,nu2]);
nx = ns*n;

% Ajuste de regresores
auxth1 = zeros(n*size(dn,1),size(theta1,2));
auxth1(n-nx1+1:n,:) = theta1(nx1:-1:1,:);
auxth1(n+(n-dn(2,1)+1:n),:) = theta1(nx1+(dn(2,1):-1:1),:);
auxth1(2*n+(n-dn(3,1)+1:n),:) = theta1(nx1+dn(2,1)+(dn(3,1):-1:1),:);
theta1=auxth1;

auxth2 = zeros(n*size(dn,1),1);
auxth2(n-nx2+1:n,:) = theta2(nx2:-1:1,:);
auxth2(n+(n-dn(2,2)+1:n),:) = theta2(nx2+(dn(2,2):-1:1),:);
auxth2(2*n+(n-dn(3,2)+1:n),:) = theta2(nx2+dn(2,2)+(dn(3,2):-1:1),:);
theta2=auxth2;

% Matrices del modelo
theta = zeros(n,3,2);
theta(:,:,1) = theta1;
theta(:,:,2) = theta2;
[A,B,C] = crea_matrices(theta);

%% Limites

Xinf = repmat(Xinf,n,1);
Xsup = repmat(Xsup,n,1);

Ulim = [-Uinf;Usup];

Rminp = [Xinf;Uinf]; % Para MPC
Rmaxp = [Xsup;Usup];

%% Matrices MPC

% Creación de H
Q = Q*eye(nx);
R = R*eye(ns);
```

```
P = P*eye(nx);
QR = repmat({blkdiag(Q,R)},1,NP);
H = blkdiag(QR{:},P);

% Creación de F

Finic = eye(nx);
Fncp = [zeros(ns,nx),eye(ns)];

AB = repmat({[A,B]},1,NP);
zeroI = repmat({[zeros(nx,ns),-eye(nx)]},1,NP);
Fmod = [blkdiag(AB{:}),zeros(nx*NP,nx)] + [zeros(nx*NP,nx),blkdiag(zeroI
    {:})];
F = [Finic, zeros(nx,size(Fmod,2)-size(Finic,2)); Fmod];
if NP > NC
    FNC = repmat({Fncp},1,NP-NC);
    F = [F;[zeros(ns*(NP-NC),(NC-1)*(nx+ns)),-repmat(Fncp,NP-NC,1),
        blkdiag(FNC{:}),zeros(ns*(NP-NC),nx)]];
end

f = zeros(size(F,1),1);

% Coeficientes de restriccion terminal de g
gmin = [repmat(Rminp,NP,1);Xinf];
gmax = [repmat(Rmaxp,NP,1);Xsup];
gcoef = [sum(theta1(:,1));sum(theta2(:,1))];
lambda = gcoef.^NP;

%% Espera para alcanzar el P.O.

h1(PO(1));
h2(PO(2));

for i = 1:inic
    tic;
    t1 = T1C();
    t2 = T2C();
    h1s = [h1s,PO(1)];
    h2s = [h2s,PO(2)];
    t1s = [t1s,t1];
    t2s = [t2s,t2];
    clf
    plot(t1s,'r.')
    hold on
    plot(t2s,'b.')
    treg = toc;
    pause(max(0.01,1.0-treg))
end

%% Referencias
```

```
medias = [mean(t1s(end-30:end));mean(t2s(end-30:end))]; % P.O.

N = [A-eye(nx),B;C,zeros(ns)];

t(1) = t(1)+n-1;
t(end) = t(end)+NPlim;
Ref = zeros(ns,sum(t)); % Referencias para representación gráfica
aux = 0;
for i = 1:length(t)
    Ref(:,aux+1:aux+t(i)) = Yref(:,i).*ones(2,t(i));
    aux = aux+t(i);
end
Zref = N\[zeros(nx,size(Ref,2));Ref-medias'.*ones(2,size(Ref,2))]; %
    Referencias menos medias

t(end) = t(end)-NPlim;

%% Registros

x = zeros(nx,sum(t)+1);
y = zeros(ns,sum(t));
u = zeros(ns,sum(t));

t(1) = t(1)-n+1;

%% Ensayo de MPC

iter = n; % Índice del vector de entrada u
cont = 0; % Índice de t de temperatura tomadas
reg = []; % Registro del tiempo de cálculo

x((n-1)*ns+(1:ns),iter) = [t1;t2]-medias;
y(:,iter) = [t1;t2];
z = zeros(NP*(nx+ns)+nx,1);
z(2:n) = t1s(end-(nx1-1)*Tm:Tm:end-Tm)'-medias(1);
z(n+(2:n)) = t2s(end-(n-1)*Tm:Tm:end-Tm)'-medias(2);

disp('Comienzo ensayo MPC')
for j = 1:length(t)
    for i = 1:Tm*t(j)
        tic;
        t1 = T1C();
        t2 = T2C();
        t1s = [t1s,t1];
        t2s = [t2s,t2];

        if mod(i,Tm) == 0
            z((n-1)*ns+(1:ns)) = [t1;t2]-medias;
```

```
        zr = [reshape(Zref(:,iter:iter+NP-1),1,[]), Zref(1:nx,iter+NP
            )'];
        h = (-zr*H)';
        f(1:nx) = z(1:nx);
        gmin(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),iter+
            NP)...
            -abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP)).*
                lambda;
        gmax(NP*(nx+ns)+(n-1)*ns+(1:ns)) = Zref((n-1)*ns+(1:ns),iter+
            NP)...
            +abs(z((n-1)*ns+(1:ns))-Zref((n-1)*ns+(1:ns),iter+NP)).*
                lambda;
        z = admm(H,h,F,f,gmin,gmax,z,rho,tol,MaxIter);
        u(:,iter) = [z(nx+totbc(2:end))];
        h1(u(1,iter)+PO(1));
        h2(u(2,iter)+PO(2));
        aux = [-u(:,iter);u(:,iter)];
        if any(aux >= Ulim)
            z(nx+ns+(1:nx)) = [A,B]*z(1:nx+ns);
        end
        x(:,iter+1) = z(nx+ns+(1:nx));
        z = [z(nx+ns+1:end);zeros(nx+ns,1)];
        iter=iter+1;
    end

    h1s = [h1s,u(1,iter-1)+PO(1)];
    h2s = [h2s,u(2,iter-1)+PO(2)];

    % Representación gráfica
    cont = cont+1;
    time = linspace(0,cont+1,cont);
    clf
    subplot(2,1,1)
    plot(time,t1s(inic+1:end),'r.')
    hold on
    plot(time,t2s(inic+1:end),'b.')
    plot(time,Ref(1,1:cont),'g--')
    plot(time,Ref(2,1:cont),'k--')
    ylabel('Temperature (degC)')
    legend('Temperature 1','Temperature 2','Location','NorthWest')
    subplot(2,1,2)
    plot(time,h1s(inic+1:end),'r')
    hold on
    plot(time,h2s(inic+1:end),'b')
    ylabel('Heater (0-5.5 V)')
    xlabel('Time (sec)')
    legend('Heater 1','Heater 2','Location','NorthWest')
    drawnow;
    treg = toc;
    reg = [reg treg];
```

```
        pause(max(0.01,1.0-treg))
    end
end

h1(0); h2(0); % Apaga calentadores

disp('Fin ensayo MPC')
```

Índice de Figuras

1.1	TCLab	1
1.2	Ejemplar de TCLab empleado	2
2.1	Resultados del ensayo de salto en escalón	4
2.2	Resultados del ensayo de control en relé	5
2.3	Salidas del ensayo PRBS	6
2.4	Entradas del ensayo PRBS	6
2.5	Salidas del ensayo Chirp	7
2.6	Entradas del ensayo Chirp	8
2.7	Datos de entrenamiento para el sensor 1 y estimación del modelo	11
2.8	Datos de entrenamiento para el sensor 2 y estimación del modelo	11
2.9	Datos de validación para el sensor 1 y estimación del modelo	12
2.10	Datos de validación para el sensor 2 y estimación del modelo	12
3.1	Salidas de la simulación del modelo lineal	20
3.2	Entradas de la simulación del modelo lineal	20
3.3	Comparativa de MPC ensayado y simulado: medidas de los sensores	21
3.4	Comparativa de MPC ensayado y simulado: actuación	21
4.1	Función de reubicación	23
4.2	Funciones triangulares	24
4.3	Funciones trapezoidales	24
4.4	Salidas del ensayo de obtención de datos para el modelo no lineal	26
4.5	Entradas del ensayo de obtención de datos para el modelo no lineal	26
4.6	Datos de entrenamiento para el modelo del sensor 1 y su respuesta	30
4.7	Datos de entrenamiento para el modelo del sensor 2 y su respuesta	30
4.8	Datos de validación para el modelo del sensor 1 y su respuesta	31
4.9	Datos de validación para el modelo del sensor 2 y su respuesta	31
5.1	Simulaciones de los MPC triangular y trapezoidal: sensores	35
5.2	Simulaciones de los MPC triangular y trapezoidal: actuadores	35
5.3	Ensayos de los MPC triangular y trapezoidal: sensores	36
5.4	Ensayos de los MPC triangular y trapezoidal: actuadores	36

Bibliografía

[1] APMonitor, *Página web del fabricante de Temperature Control Lab*.

[2] P. K. S. Tam F. H. F. Leung, L. K. Wong, *Design of Stable TS Fuzzy Model Based Systems*, Stability Issues in Fuzzy Control (Francisco Aracil, Javier y Gordillo, ed.), Physica-Verlag, Heidelberg, 2000, pp. 227–240.

[3] Mario Fernández Rangel, *Control del sistema TCLab con técnicas de control predictivo*, Universidad de Sevilla, 2021.