CONTROL DE UN LABORATORIO DE CONTROL DE TEMPERATURA MEDIANTE REDES NEURONALES RECURRENTES

Cristian Blanco Fernández[1], Jesús Enrique Sierra-García[2], Matilde Santos[3]
[1]ETSI Informática, UNED, 28040-Madrid, España, cblanco177@alumno.uned.es
[2]Departamento de Ingeniería Electromecánica, Universidad de Burgos, 09006-Burgos
[3]Instituto de Ingeniería del Conocimiento, Universidad Complutense de Madrid, 28040-Madrid

Resumen

El control predictivo (MPC – Model Predictive Control) de procesos es una estrategia extendida, que se basa en la resolución de un problema de optimización en tiempo real, lo que puede ser computacionalmente muy costoso en función de la naturaleza del problema en cuestión. Para superar esta limitación, se ha investigado la posibilidad de utilizar redes neuronales entrenadas para sustituir a este tipo de controladores. La idea subyacente es que para problemas que muestran un comportamiento predecible, se puede entrenar una red a partir de un controlador optimizado para que pueda sustituirlo. De esta forma los costes computacionales se trasladan al entrenamiento de la red, permitiendo el control en tiempo real sin necesidad de realizar operaciones computacionalmente complejas. Este trabajo explora esta idea a partir de un laboratorio de control de temperatura. Se entrenan dos tipos de redes neuronales recurrentes, y se compara su funcionamiento con el de un controlador tradicional.

Palabras clave: redes neuronales recurrentes, RNN, LSTM, laboratorio de temperatura, eficiencia computacional.

1 INTRODUCCIÓN

Las redes neuronales artificiales, ANN por sus siglas en inglés, han experimentado un desarrollo muy importante en diversos campos debido a su potencial para representar todo tipo de sistemas complejos. En el campo del control existen muchos trabajos relativos al uso de redes neuronales, como la propuesta realizada por Kumar [1] para el control de una estación espacial, o las estrategias para el control de la turbina de un aerogenerador expuestas en [2] y [3]. Si bien es cierto que este tipo de controles tienen problemas a la hora de enfrentarse a cambios en el sistema, en [4] se presenta un controlador que muestra robustez a la hora de guiar un cuatri-rotor con cambios en su masa y las perturbaciones de viento sufridas.

Existen otros trabajos relacionados, como el que se puede encontrar en [5], donde se propone el modelado de la respuesta de un laboratorio de temperatura mediante redes LSTM. En [6] se presenta un neuro-controlador para la temperatura de un criostato de helio líquido mediante aprendizaje *online*. La motivación de estos trabajos es la de diseñar formas de control cuyo coste computacional sea bajo durante la operación. En [7] se propone el uso de redes neuronales como sustituto de controladores basados en la optimización de modelos (MPC), ya que el coste computacional de los MPC hace que para determinadas aplicaciones no sean efectivos. Sin embargo, el control MPC para la temperatura ha sido utilizado con éxito y comparado ventajosamente a otras implementaciones del control.

Por ello, en este trabajo se propone utilizar redes neuronales, capaces de emular el comportamiento de un MPC, para así trasladar el mayor coste computacional al entrenamiento de la red *offline*, y que no se produzca durante la operación del sistema. Se han aplicado dos tipos de redes neuronales, una red neuronal recurrente RNN (*Recurrent Neural Network*), y una modificación de ésta con memoria a largo plazo, LSTM (*Long Short-Term Memory*).

La estructura del artículo es la siguiente. En primer lugar se resume el funcionamiento de las redes recurrentes utilizadas en el control. Posteriormente se presenta el modelo no lineal utilizado para diseñar el controlador MPC. A continuación se explica el entrenamiento de las redes neuronales a partir de los datos generados por el MPC. Por último, se presentan los resultados obtenidos y la discusión de los mismos.

2 REDES NEURONALES

El fundamento de las redes neuronales para el modelado de procesos dinámicos es la adaptación de cada neurona individual para predecir la salida a

https://doi.org/10.17979/spudc.9788497498418.0193

partir de la combinación lineal de la información de entrada, y la aplicación de una transformación no lineal (función de activación). Algunas funciones de activación comunes son la sigmoide, la función ReLu (*rectified linear unit*), o la tangente hiperbólica. La Figura 1 muestra un esquema de una neurona aislada.

La estructura de una red consiste en varias neuronas de este tipo, donde la información se transmite a lo largo de varias capas. El factor peso w_i de las conexiones de la red se optimiza mediante el proceso de aprendizaje, minimizando una función del error. Esta función se propaga hacia atrás en la red para realizar el ajuste de los parámetros.

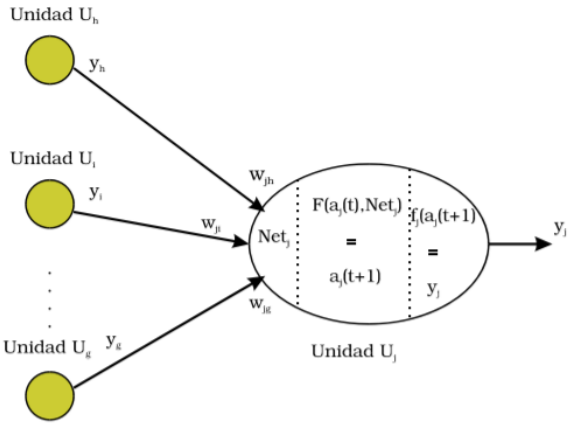

Figura 1 – Esquema de perceptrón [8]

Siendo y el conjunto de entradas de cada unidad conectada a la neurona, w el peso que multiplica a cada entrada, Net_j la entrada total en la neurona j, $a_j(t)$ y $a_j(t+1)$ los estados de activación en distintos instantes de tiempo, F la función de activación, y f_j la función de salida.

2.1 Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) extienden las redes convencionales en la dimensión temporal, por lo que son idóneas para problemas en los que hay una relación secuencial entre las entradas y las salidas. En un laboratorio de temperatura, dada la correlación temporal entre el calor aportado y la temperatura medida, son muy adecuadas.

La particularidad de este tipo de redes es que la salida de una neurona (o una capa, dependiendo de la arquitectura) en un instante de tiempo sirve como entrada a esa misma neurona en el instante siguiente. Esto se implementa típicamente mediante un elemento h que se actualiza cada vez que la neurona se activa, y del cual depende también el resultado final que predice la misma. Este elemento sirve como un resumen de las entradas de la neurona. Una arquitectura sencilla de este tipo de redes se presenta en la Figura 2.

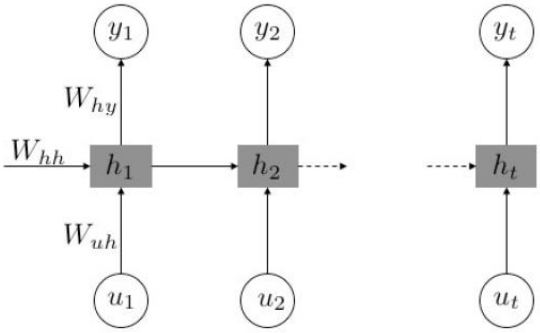

Figura 2 – Esquema de una neurona RNN [7]

Donde W_{hh}, W_{hy}, y W_{uh} son las matrices de pesos, u_t la entrada, y_t la salida, y h_t el estado escondido. La dinámica de la RNN se describe como sigue:

$$h_t = f_h(h_{t-1}, u_t), \qquad y_t = f_y(h_t) \tag{1}$$

Donde f_h es una función de transición que describe la dinámica de h_t, y f_y es una función que relaciona la salida y_t con el elemento h. La función de transición de la Figura 2 se define como una composición elemento a elemento con una transformación afín de u_t y h_{t-1} de la forma:

$$h_t = f_h(W_{hh}h_{t-1} + W_{uh}u_t) \tag{2}$$

Donde W_{hh} y W_{uh} son las matrices de pesos que relacionan los elementos h intermedios con las entradas u. De forma análoga, f_y se define como una transformación afín de h_t, con una matriz de pesos W_{hy}.

$$y_t = f_y(W_{hy}) \tag{3}$$

Una forma típica de generar estas redes es utilizar la tangente hiperbólica como función f_h y la identidad como f_y. Las matrices de pesos se suelen actualizar mediante el algoritmo *back-propagation*.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{uh}u_t) \tag{4}$$
$$y_t = W_{hy}h_t \tag{5}$$

Este tipo de redes se pueden utilizar para generar una secuencia de datos a partir de una secuencia de entrada (*many to many*), predecir el siguiente valor de la secuencia de entrada (*many to one*), o predecir una secuencia a partir de un valor de entrada (*one to many*).

Un problema del entrenamiento de estas redes neuronales es el de los gradientes evanescentes (*vanishing gradients*). A la hora de hacer la propagación hacia atrás se calcula la derivada de las

https://doi.org/10.17979/spudc.9788497498418.0193

funciones de activación y, para las funciones típicas, estas varían muy poco cerca de los valores límite de la función. Esto puede producir que los valores de los pesos de entrada no se actualicen, o se actualicen de forma muy lenta. Además, este tipo de redes olvidan la información de los pasos iniciales a medida que se recorre la secuencia de datos, lo que unido al problema de los gradientes puede hacer complicado para estar redes capturar dependencias a largo plazo. Estas desventajas han hecho que se hayan desarrollado distintos tipos de redes recurrentes que se comporten mejor en estos aspectos.

2.2 Memoria Larga a Corto Plazo

Las redes LSTM son un tipo de redes neuronales recurrentes desarrolladas para superar las carencias de las RNN a la hora de recordar información a largo plazo, se propusieron por primera vez en [9]. La Figura 3 muestra el esquema de una unidad de una red LSTM.

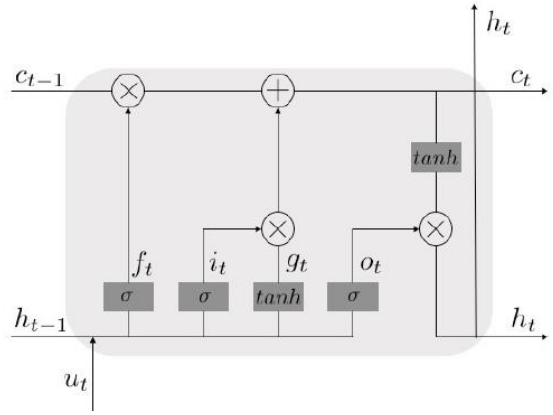

Figura 3 – Esquema de una neurona LSTM [7]

La estructura de la Figura 3 está compuesta principalmente por una puerta de entrada (*input*) i_t, puerta de memoria o salida c_t, una puerta de olvido (*forget*) f_t, y una puerta de salida (*output*) o_t. La puerta g_t es un complemento a la de entrada.

$$f_t = \sigma(W_{uf}u_t + W_{hf}h_{t-1}) \quad (6)$$
$$i_t = \sigma(W_{ui}u_t + W_{hi}h_{t-1}) \quad (7)$$
$$o_t = \sigma(W_{uo}u_t + W_{ho}h_{t-1}) \quad (8)$$
$$g_t = \tanh(W_{ug}u_t + W_{hg}h_{t-1}) \quad (9)$$

Las matrices W en este caso son análogas a las de las RNN. La nueva salida c_t se determina controlando el flujo de información en la neurona.

En primer lugar, la puerta f_t determina la información de h_{t-1} que se mantiene sin modificar, y para cada elemento c_{t-1} produce un valor entre 0 (olvidado por completo) y 1 (recordado por completo). Las puertas i_t y g_t determinan la nueva información que se quiere pasar a c_t combinando sus salidas; la primera puerta decide qué valores se van a actualizar, y la segunda produce una selección de posibles valores nuevos. Por último, se actualiza el valor de h_t con la información que se desea pasar a la siguiente capa. La puerta o_t decide la cantidad de información del estado oculto que puede pasar a la siguiente capa. Esto además se combina con un filtro con el valor de la nueva salida c_t para producir el valor de h_t.

$$c_t = f_t c_{t-1} + i_t g_t \quad (10)$$
$$h_t = o_t \tanh(c_t) \quad (11)$$

De esta forma se modifica el estado de salida y el estado oculto, c_t y h_t, sin eliminar de forma sistemática la información más antigua. Esto permite mejorar el funcionamiento de las redes recurrentes para series de datos en las que existen dependencias a largo plazo.

3 LABORATORIO DE CONTROL DE TEMPERATURA

El laboratorio de control de temperatura [10], también llamado TCLab, es una aplicación de control retroalimentado que consta de una placa de Arduino, un LED, dos calentadores, y dos sensores de temperatura. La Figura 4 muestra un esquema del laboratorio de temperatura. Pero además se puede simular el comportamiento de este laboratorio sin necesidad de adquirir el hardware [5].

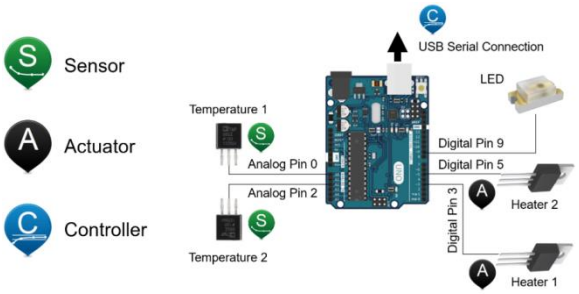

Figura 4 – Esquema de un TCLab

El sistema funciona de forma sencilla: la potencia de los calentadores (*heater*) 1 y 2 se puede controlar mediante una entrada que ajusta la entrega de potencia entre el 0 y el 100%. Los sensores de temperatura 1 y 2 devuelven las medidas de la temperatura en los mismos. La Figura 5 muestra la arquitectura de conexión de un TCLab ya que los algoritmos de control propuestos en este trabajo se implementarán en simulación.

En el TCLab simulado se puede dar un input a los calentadores y obtener una respuesta, para definir un sistema no lineal de la transferencia de calor a los sensores similar al presentado en [11]. Las

ecuaciones analíticas del modelo se presentan a continuación.

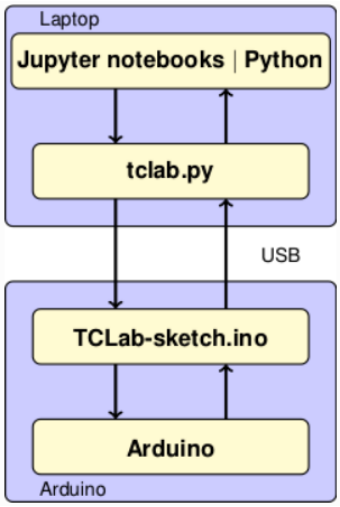

Figura 5 – Arquitectura TCLab [5]

$$mc_p \frac{dT_{h1}}{dt} = UA(T_a - T_{h1}) + \epsilon\sigma A(T_a^4 - T_{h1}^4) + Q_{c12} + Q_{R12} + \alpha_1 Q_1 \quad (12)$$

$$mc_p \frac{dT_{h2}}{dt} = UA(T_a - T_{h2}) + \epsilon\sigma A(T_a^4 - T_{h2}^4) + Q_{c12} - Q_{R12} + \alpha_1 Q_1 \quad (13)$$

Donde m es la masa del sistema (kg), T_{hi} la temperatura en los calentadores (K), U el coeficiente de transferencia de calor (W/m^2K), A la superficie que no está entre los calentadores (m^2), T_a la temperatura ambiente (K), ϵ el coeficiente de emisividad, σ la constante de Boltzmann (W/m^2K^4), α_i el factor de calentamiento ($W/\%\ heater$), y Q_i el calor aportado por los calentadores ($\%$). La primera parte de la derecha de las ecuaciones es el termino convectivo y la segunda el término radiante. El parámetro Q_{c12} es el calor transmitido por convección entre los calentadores (W), y el Q_{r12} es el calor transferido por radiación (W).

$$Q_{c12} = U_s A_s (T_{h2} - T_{h1}) \quad (14)$$

$$Q_{r12} = \epsilon\sigma A(T_{h2}^4 - T_{h1}^4) \quad (15)$$

Siendo U_s (W/m^2K) y A_s (m^2) el coeficiente de transferencia de calor entre los calentadores y la superficie entre calentadores, respectivamente.

Para complementar este modelo analítico se añaden dos ecuaciones empíricas que sirven para representar el retardo que hay entre la temperatura de los calentadores y la medida de los sensores.

$$\tau \frac{dT_{c1}}{dt} = -T_{c1} + T_{h1} \quad (16)$$

$$\tau \frac{dT_{c2}}{dt} = -T_{c2} + T_{h2} \quad (17)$$

Donde τ es el tiempo de respuesta del sistema (s) y T_{ci} la temperatura medida en los sensores (K).

Los parámetros U, U_s, τ y α_i son dependientes de cada sistema, y es necesario identificarlos. Para ello en este trabajo se ha utilizado la librería *Gekko* [12] con el algoritmo *Ipopt* de optimización. Se ajustan los parámetros minimizando la diferencia entre las temperaturas que predice el modelo y las que se miden del TCLab. La entrada son los valores de los calentadores, es decir Q_1 y Q_2.

Los valores de los parámetros optimizados se muestran en la Tabla 1. La Figura 6 y la Figura 7 muestran la comparativa de la respuesta en temperatura en el sensor 1 (T_1) y en el sensor 2 (T_2) del modelo planteado con los parámetros optimizados. Se puede observar cómo para ambas variables la identificación da buenos resultados.

Tabla 1 – Parámetros optimizados

Parámetro	Unidades	Valor
U	$[W/m^2K]$	8.132
U_s	$[W/m^2K]$	16.675
τ	$[s]$	25.0
α_1	$[W/\%]$	0.0102
α_2	$[W/\%]$	0.0045

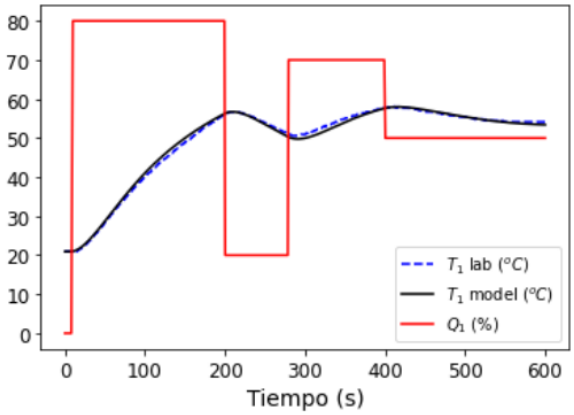

Figura 6 – Modelo no lineal T_1-Q_1

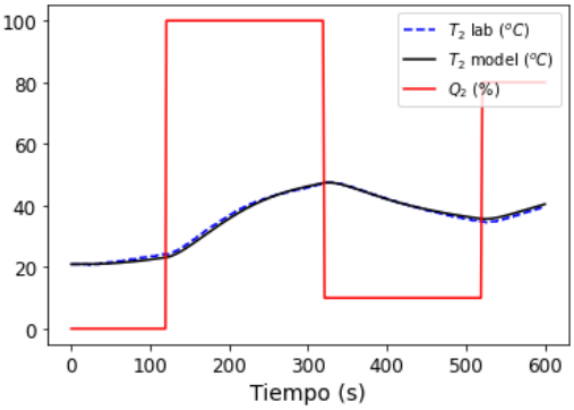

Figura 7 – Modelo no lineal T_2-Q_2

https://doi.org/10.17979/spudc.9788497498418.0193

El controlador basado en este modelo se diseña para controlar la entrada del calentador 1 en función de la medida obtenida del sensor de temperatura 1 del TCLab. De nuevo se utiliza la librería *Gekko* [12] para optimizar la señal del MPC, en modo de control simultáneo y el algoritmo *Ipopt*. La Figura 8 muestra la arquitectura de control.

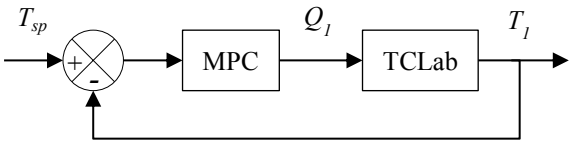

Figura 8 – Arquitectura controlador MPC

En este caso la señal que se desea obtener es la entrada al calentador 1, Q_1, como porcentaje de potencia, para minimizar la diferencia de temperatura T_1 del TCLab con respecto a una temperatura de referencia T_{sp}.

El resultado del seguimiento con el MPC se puede ver en la Figura 9, y la señal de control en la Figura 10. Estas serán las entradas que se utilizarán en las redes neuronales.

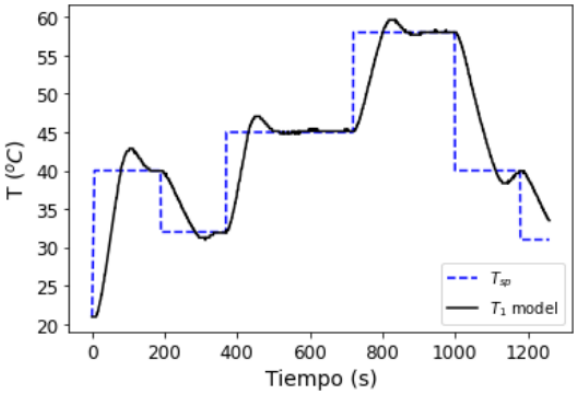

Figura 9 – Resutlado controlador MPC

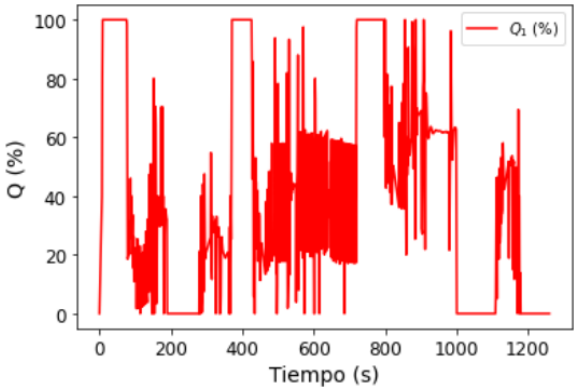

Figura 10 – Señal de control MPC

4 CONTROL MEDIANTE REDES NEURONALES

El objetivo de este trabajo es comparar el rendimiento de dos tipos de redes neuronales recurrentes para realizar el control de un TCLab, utilizando como referencia el controlador MPC presentado en el apartado anterior. La Figura 11 muestra un esquema con del control con las redes neuronales.

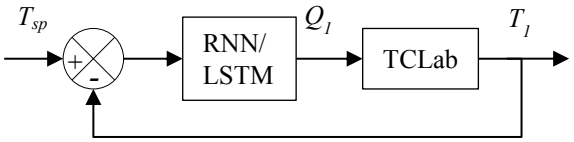

Figura 11 – Arquitectura controlador con redes neuronales

Las redes neuronales se han programado con Tensorflow [13]. Ambas redes tienen la configuración basada en la propuesta de [14]. Constan de tres capas, las dos primeras con 100 neuronas, y la tercera es una capa de salida de una neurona, con función de activación ReLu. La Aunque no se han optimizado de forma automática, se han testeado diversos valores y se han seleccionado los que daban unos resultados aceptables.

Tabla 2 resume los híper-parámetros de ambas redes. Aunque no se han optimizado de forma automática, se han testeado diversos valores y se han seleccionado los que daban unos resultados aceptables.

Tabla 2 – Híper-parámetros de las redes

Parámetro	Descripción	Valor
Capa 1		
Unidades	Número de neuronas	100
Tipo	Tipo de neuronas	RNN/LSTM
Input	Vector de entrada	15x100
Drop-out	Factor de neuronas a desactivar	0.1
Output	Vector de salida	15x100
Capa 2		
Unidades	Número de neuronas	100
Tipo	Tipo de neuronas	RNN/LSTM
Input	Vector de entrada	15x100
Drop-out	Factor de neuronas a desactivar	0.1
Output	Vector de salida	100
Capa 3		
Unidades	Número de neuronas	1

Parámetro	Descripción	Valor
Tipo	Tipo de neuronas	ReLu
Input	Vector de entrada	100
Drop-out	Factor de neuronas a desactivar	0.0
Output	Vector de salida	1

La información de entrada en la primera capa es una matriz que contiene los 15 últimos valores del error medidos, que alimentan a las 100 unidades que componen la capa. Se utiliza un factor de *drop-out*, es decir, de neuronas que se pueden desactivar durante el entrenamiento, de 0.1. La capa 2 tiene una estructura similar a la capa 1, pero en la salida de ésta hacia la capa 3 se elimina la dimensión de la matriz que contiene la información temporal, es decir, la serie de los 15 últimos valores del error. Por último, la capa 3 consta de una sola unidad, conectada a todas las neuronas de la capa anterior. En esta capa evidentemente se prescinde del factor de *drop-out*, y genera la salida Q_1, valor que sirve como entrada al TCLab.

En esencia, las dos primeras capas tratan de capturar la relación temporal entre el error de temperatura y la salida del MPC. Para esto las redes toman como entrada la diferencia entre la temperatura deseada T_{sp} y la temperatura del sensor 1, T_1, y produce como salida el control del calentador 1, Q_1. Es importante remarcar que sólo se ha utilizado el calentador y el sensor 1. Por último, para el aprendizaje se utiliza un optimizador *adam* con función objetivo el error cuadrático medio.

4.1 Entrenamiento de las Redes

El entrenamiento se realiza a partir de datos obtenidos por el controlador MPC. Para ello se genera un perfil de temperatura a lo largo de 21 minutos, como el mostrado en la Figura 9, y se almacena la entrada y salida del controlador MPC.

Estos datos se dividen en tres conjuntos, el 60% se utilizan para el aprendizaje, un 20% para la validación, y el 20% restante para ensayar los resultados. Dado que se trata de una serie temporal esta división se realiza en orden cronológico. La Figura 12 muestra el resultado del entrenamiento de la RNN para el intervalo de tiempo del conjunto de ensayo. La Figura 13 contiene la información análoga para la LSTM.

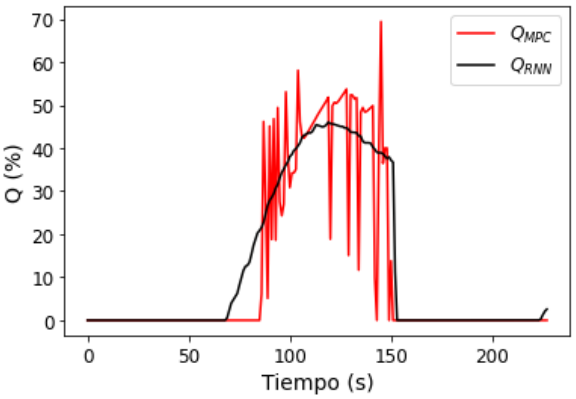

Figura 12 – Ensayo RNN

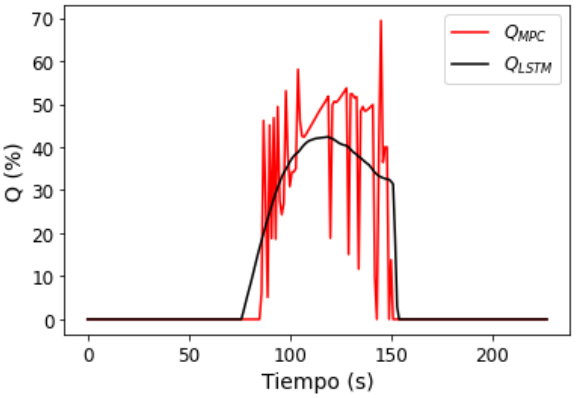

Figura 13 – Ensayo LSTM

5 RESULTADOS

En esta sección se aplican estas estrategias de control para que la temperatura del TCLab se ajuste a un nuevo perfil, distinto del de entrenamiento, durante un intervalo de 30 minutos, y se comparan los resultados obtenidos. La Figura 14 muestra las temperaturas generadas por el TCLab con las entradas proporcionadas por cada controlador, junto con el perfil de temperatura de referencia. La Figura 15 contiene la comparativa de la entrada del control de calor.

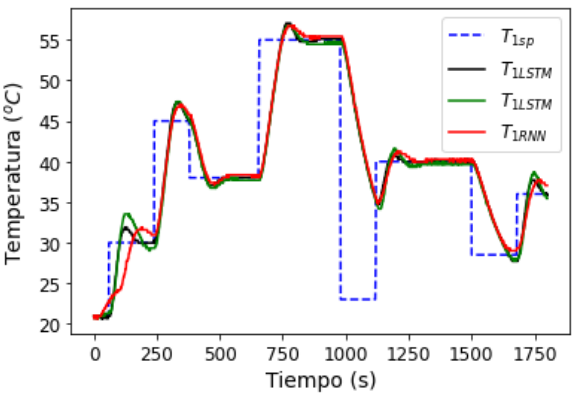

Figura 14 – Comparativa de temperaturas

https://doi.org/10.17979/spudc.9788497498418.0193

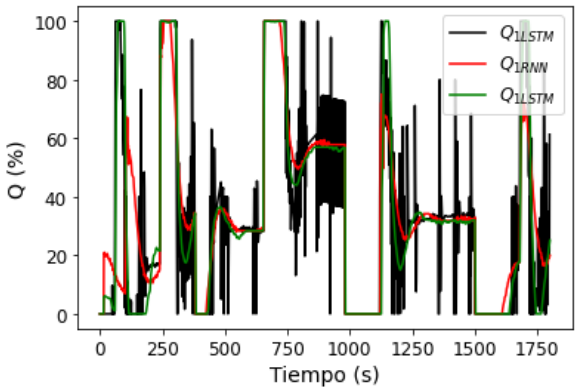

Figura 15 – Comparativa de control de calor

Ambas redes consiguen controlar el TCLab para ajustar la temperatura T_1 a la referencia deseada T_{sp}. La Tabla 3 muestra el promedio del error calculado como la diferencia, en valor absoluto, de T_{sp} y T_1 en el intervalo de tiempo de 30 minutos. Se observa que la desviación media de todos los controladores es del mismo orden.

Tabla 3 – Error medio

Control	Error [º]
MPC	1.49
RNN	1.41
LSTM	1.24

Se comprueba también el coste computacional de cada controlador, para estudiar la hipótesis discutida en la introducción, es decir, si las redes neuronales generan un control computacionalmente más eficiente. Para esto se mide el tiempo que tarda cada método en producir una señal de control en cada paso. Dado que todo el algoritmo de control se implementa desde el ordenador, el tiempo que tarda en responder cada controlador se mide de forma sencilla implementando un reloj en el propio algoritmo, que registra la diferencia de tiempo antes y después de ejecutar la optimización con el MPC, o la predicción con las redes neuronales, cada vez que se ejecuta dicho algoritmo. Todos los controladores se ejecutan usando la CPU de un ordenador portátil con un procesador Intel® Core™ i5-8250 de 1.60 GHz con cuatro núcleos, con 8 GB de RAM, y sistema operativo Windows 10 de 64 bits.

Para eliminar la influencia que pueda tener la carga de trabajo de la CPU en un momento puntual en la comparación de tiempos, se han hecho 50 ensayos de 5 minutos. Cada controlador se emplea para mantener un perfil de temperatura aleatorio durante el tiempo estipulado, y se registra el tiempo medio de ejecución de cada acción de control.

La Tabla 4 muestra la comparativa del tiempo medio, en segundos, que necesita cada controlador para generar una respuesta. Se puede observar que las redes neuronales tienen un tiempo de respuesta medio de un orden de magnitud menor que el MPC.

Tabla 4 – Coste computacional

Control	Tiempo [s]
MPC	0.676
RNN	0.045
LSTM	0.046

6 CONCLUSIONES Y TRABAJOS FUTUROS

En este estudio se comprueba la hipótesis de que los controladores basados en redes neuronales recurrentes son candidatos adecuados para sustituir a controladores más complejos computacionalmente. Mediante la definición de un modelo no lineal de un sistema de control de temperatura TCLab, se han entrenado dos redes neuronales, una RNN sencilla y una LSTM, y se han implementado controladores basados en ellas que son capaces de mostrar un rendimiento similar al controlador MPC, con una fracción del coste computacional. De esta forma, se traslada este coste a la fase de entrenamiento, en lugar de tenerlo en la fase de operación.

Como propuesta para trabajos futuros, la más evidente es la optimización de los hiper-parámetros y la macro-estructura de las redes, que en este estudio se han mantenido constantes. Otra línea de trabajo sería extender los controladores presentados en este trabajo a un sistema con múltiples entradas y salidas. El TCLab ofrece la oportunidad de hacerlo dado que dispone de dos entradas y dos sensores. Además, sería interesante estudiar cómo afecta la optimización de las redes a los datos de entrenamiento, validación y ensayo. Como otra línea de trabajo futura se pretende aplicar esta estrategia en aerogeneradores.

English summary

RECURRENT NEURAL NETWORKS BASED CONTROL OF A TEMPERATURE CONTROL LABORATORY

Abstract

Model Predictive Control (MPC) is an extended control strategy based on the resolution of an optimization problem in real time, which can be a computationally expensive process depending on the nature of the problem. To overcome this limitation, the use of neural networks already trained as a

substitute for this type of controllers has been investigated. The underlying concept is that, for a sufficiently predictable system, a neural network can be trained, using data from an optimized controller, which can replace the MPC. With this approach the higher computational cost lies in the training of the network instead of the online operation of the system. This idea is explored using a temperature control lab. Two types of recurring neural networks are trained, and the performance and computational cost are compared with a conventional controller.

Keywords: recurrent neural networks, RNN, LSTM, temperature laboratory, computational efficiency.

Referencias

[1] K. Krishna Kumar, «Adaptive Neuro-Control for Spacecraft Attitude Control,» 1994.

[2] J. E. Sierra-García y M. Santos, «Lookup Table and Neural Network Hybrid Strategy for Wind Turbine Pitch Control,» *Sustainability,* vol. 13, nº 6, p. 3235, 2021.

[3] J. E. Sierra-García y M. Santos, «Redes neuronales y aprendizaje por refuerzo en el control de turbinas eólicas,» *Revista Iberoamericana de Automática e Informática Industrial,* vol. 18, nº 4, pp. 327-335, 2021.

[4] J. E. Sierra-García y M. Santos, «Switched learning adaptive neuro-control strategy,» *Neurocomputing,* vol. 452, pp. 450-464, 2021.

[5] J. Kantor y C. Sandrock, «TCLab: Temperature Control Laboratory,» 2018. [En línea]. Available: https://tclab.readthedocs.io/en/latest/index.html. [Último acceso: 25 01 2022].

[6] M. Santos y A. L. Dexter, «Temperature control in liquid helium cryostat using self-learning neurofuzzy controller,» *IEE Proceedings-Control Theory and Applications,* vol. 148, nº 3, pp. 233-238, 2001.

[7] S. Spielberg, P. Kumar, A. Tulsyan, B. Gopaluni y P. Loewen, «A Deep Learning Architecture for Predictive Control,» *IFAC PapersOnLine,* vol. 51, nº 18, pp. 512-517, 2018.

[8] J. R. Hilera y V. Martinez Hernando, Redes neuronales artificiales: fundamentos, modelos y aplicaciones, 1995.

[9] S. Hochreiter y J. Schmidhuber, «Long short-term memort,» *Neural Computation,* vol. 9, nº 8, pp. 1735-1780, 1997.

[10] J. D. Hedengren, «Dynamics and Control,» 05 01 2022. [En línea]. Available: http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl. [Último acceso: 25 01 2022].

[11] M. Fernández Rangel, «Control del sistema TCLab con técnicas de control predictivo,» Universidad de Sevilla, Sevilla, 2021.

[12] L. D. R. Beal, D. Hill, R. A. Martin y J. D. Hedengren, «GEKKO Optimization Suite,» *Processes,* vol. 6, nº 8, p. 106, 2018.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, Z. Davis, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard y et al., «Tensorflow: A system for large-scale machine learning,» *OSDI,* vol. 16, pp. 265-283, 2016.

[14] N. Lewis, «Emulating a PID Controller with Long Short-term Memory: Part 2,» 16 2020. [En línea]. Available: https://towardsdatascience.com/emulating-a-pid-controller-with-long-short-term-memory-part-2-4a37d32e5b47. [Último acceso: 10 01 2022].

[15] S. Kumar, B. Gopaluni y P. Loewen, «Deep reinforcement learning approaches for process control,» de *2017 6th International Symposium on Advanced Control of Industrial Processes (Ad-CONIP),*, 2017.

 https://doi.org/10.17979/spudc.9788497498418.0193