

## Monografía

# Identificación de sistemas no lineales por medio de redes neuronales auto-regresivas

Universidad de Buenos Aires  
Facultad de Ingeniería

**Alumno:** Ignacio Ezequiel Cavicchioli  
**Padrón:** 109428  
**Email:** icavicchioli@fi.uba.ar

28 de diciembre de 2025

### Resumen

En el presente trabajo se estudia la identificación y el control de un sistema no lineal de tanques acoplados mediante el uso de modelos lineales y modelos basados en redes neuronales. A partir de un modelo físico no lineal continuo, se obtiene una representación en tiempo discreto que se utiliza como referencia para simulación y validación.

La identificación del sistema se aborda mediante el entrenamiento de redes neuronales auto-regresivas, utilizando datos generados a partir de la planta no lineal. El desempeño de estos modelos se evalúa y se compara con el de modelos lineales obtenidos tanto por linealización analítica como mediante herramientas de identificación. Los resultados muestran que, dentro de un régimen de operación acotado, las redes neuronales permiten representar con mayor precisión la dinámica del sistema, aunque a costa de un mayor esfuerzo de ajuste.

Asimismo, se analiza un enfoque híbrido que combina un modelo lineal con una red neuronal destinada a compensar el error de modelado, el cual no presenta mejoras significativas respecto de los enfoques puramente lineales o puramente no lineales. Finalmente, se estudia el diseño de controladores a partir de modelos neuronales y se evalúa la posibilidad de clonar un controlador PID mediante una red neuronal, poniendo en evidencia limitaciones en términos de robustez y rechazo de perturbaciones.

**Palabras clave:** identificación de sistemas, redes neuronales, sistemas no lineales, control PID.

# Índice

1	Introducción	3
2	Sistema elegido	3
2.1	Modelo matemático	4
2.2	Simulaciones	6
3	Identificación de sistemas	7
3.1	Suposiciones y decisiones	7
3.2	MLP de 1 capa oculta	7
3.2.1	Introducción	7
3.2.2	Entrenamiento	8
3.2.3	Resultados	9
3.2.4	Análisis de resultados	10
3.3	Sistema lineal asistido	11
3.3.1	Introducción	11
3.3.2	Entrenamiento	11
3.3.3	Resultados	11
3.3.4	Análisis de resultados	12
3.4	Discusión general de resultados	13
4	Control de sistemas	14
4.1	Diseño de controlador con modelo no lineal identificado como referencia	14
4.1.1	Teoría del control PID	14
4.1.2	Diseño del controlador y resultados	15
4.2	Clonado del controlador	17
4.2.1	Entrenamiento	17
4.2.2	Resultados	19
4.2.3	Análisis de resultados	24
5	Conclusiones	25
6	Referencias	25
7	Apéndice 1 - Resultados de la primer experiencia de identificación	26
8	Apéndice 2 - Resultados de la segunda experiencia de identificación	26

## 1. Introducción

Las redes neuronales en sus múltiples formas constituyen sistemas no lineales con un amplio alcance de aplicación en campos como la biología, neurociencia y, al que se avoca este trabajo, aprendizaje automático (*machine learning*, ML). En particular, nos interesa centrarnos en las aplicaciones de las redes neuronales en el campo de control automático. Esta doctrina se encarga del diseño sistemas para regular, guiar o estabilizar procesos de manera autónoma, mediante la realimentación y corrección continua de errores.

La denominada “caja de herramientas” de aquellos en el área de control está compuesta por ciertos artefactos matemáticos que permiten encarar estos problemas, como la linealización de un sistema, el control PID, realimentación de estados, *loop-shaping*, observadores, etc. Lo que no se ha tocado en las materias de control son las estrategias no lineales. En líneas generales, todos los sistemas reales exhiben cierto grado de no linearidad, lo que implica que las estrategias de control lineales son válidas siempre que las no linealidades sean despreciables. Análogamente, las herramientas de identificación de sistemas basadas en la linealización de un sistema fallaran en modelar las no linealidades de estos.

Este trabajo va a ahondar sobre el uso de redes neuronales en la doctrina de control automático, específicamente como identificadores de sistemas y clonadores de controladores.

## 2. Sistema elegido

El sistema elegido está compuesto por 2 tanques de agua de dimensiones diferentes, dónde el primer tanque actúa como una cisterna amortiguadora de fluctuaciones en el caudal seguido de un reservorio que ajusta el caudal de salida. Este tipo de esquemas podrían encontrarse en procesos industriales que requieren de un caudal estabilizado.

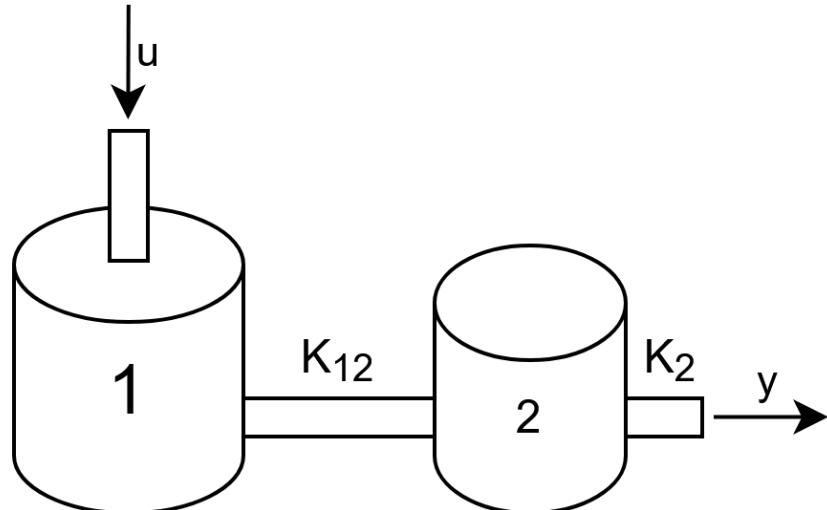


Figura 1: Sistema elegido

La figura 1 muestra el sistema recién descrito, agregando los caudales de entrada  $u$  e  $y$ . Ambos caudales son muestrados a 1 Hz, que debería ser más que suficiente para este tipo de dinámicas lentas. Tanto la tubería que une los tanques como la que sale del segundo tienen cierto coeficiente hidráulico asociado a su geometría, siendo estos  $K_{12}$  y  $K_2$  respectivamente.

Las razones por la cuales se eligió este sistema son las siguientes:

- Simplicidad: Es un sistema de vector de estados de 2 dimensiones, simple de modelar, linealizar, simular y controlar.
- No linealidad: Como se va a ver en el inciso matemático, el sistema no es lineal, que sería un requisito si se está intentando evaluar la capacidad de copiar no linealidades de las redes neuronales.
- Realismo: Se prefirió elegir un sistema que sea fácil de entender pero real, no una reducción de un sistema más complejo.

## 2.1. Modelo matemático

El modelado de este sistema hace uso de varias leyes físicas de la hidráulica. Primero, se plantea que el líquido es incompresible, por lo que el volumen solo varía si los caudales de entrada y salida no son iguales.

$$\frac{dV}{dt} = \sum q_{in} - \sum q_{out} \quad (1)$$

Además, en este caso que el área es independiente del nivel de agua, el volumen es función del área del tanque y su superficie. Podemos derivar respecto del tiempo.

$$V(t) = A \cdot h(t) \xrightarrow{d/dt} \frac{dV}{dt} = A \cdot \frac{dh(t)}{dt} \quad (2)$$

Luego se aplica la ley de caudal, que relaciona el caudal entre 2 puntos con la diferencia de nivel entre ellos mismos.

$$q = k \sqrt{2g} \sqrt{\Delta h} \quad (3)$$

El caudal entre los tanques resulta:

$$q_{12}(t) = k_{12} \sqrt{2g} \sqrt{h_1(t) - h_2(t)} \quad (4)$$

El caudal de salida del segundo tanque, que también es la salida  $y$ , es:

$$q_2(t) = y(t) = k_2 \sqrt{2g} \sqrt{h_2(t)} \quad (5)$$

Ahora, se plantea un balance en cada tanque igualando (1) y (2).

$$A_1 \cdot \frac{dh_1}{dt} = u(t) - q_{12}(t) \quad (6)$$

$$A_2 \cdot \frac{dh_2}{dt} = q_{12}(t) - q_2(t) = q_{12}(t) - y(t) \quad (7)$$

sustituyendo con (4) :

$$A_1 \cdot \frac{dh_1}{dt} = u(t) - k_{12} \sqrt{2g} \sqrt{h_1(t) - h_2(t)} \quad (8)$$

$$A_2 \cdot \frac{dh_2}{dt} = k_{12} \sqrt{2g} \sqrt{h_1(t) - h_2(t)} - k_2 \sqrt{2g} \sqrt{h_2(t)} \quad (9)$$

Finalmente, se asume que:

$$h_1 \geq h_2 \geq 0$$

Con todo esto se plantea el espacio de estados **no lineal** y continuo, con la forma de a continuación:

$$\frac{d}{dt} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} (u - k_{12} \sqrt{2g} \sqrt{h_1 - h_2}) \cdot \frac{1}{A_1} \\ (k_{12} \sqrt{2g} \sqrt{h_1 - h_2} - k_2 \sqrt{2g} \sqrt{h_2}) \cdot \frac{1}{A_2} \end{bmatrix} \quad (10)$$

$$y = k_2 \sqrt{2g} \sqrt{h_2} \quad (11)$$

Las expresiones (10) y (11) se van a linealizar en torno al punto de equilibrio  $(x_e, u_e)$  y las constantes físicas indicadas abajo.

- $g = 9,81$
- $A_1 = 0,342, A_2 = 0,126$
- $k_{12} = 5 \times 10^{-4}, k_2 = 1 \times 10^{-3}$
- $h_{1s} = 1,019, h_{2s} = 0,204$
- $u_s = 0,002$

Las variables de estado elegidas se redefinen como variaciones en torno a ese mismo estado, por lo que de ahora en más las alturas  $h_1$  y  $h_2$  no son las mismas que en la planta no lineal. El proceso arranca planteando la linealización en sí, que se ve en la ecuación (12). La expresión (13) se cumple por definición del punto  $(x_e, u_e)$ . En (14) y (15) se obtiene la matriz A del espacio de estados lineal. En (16) se obtiene la matriz B, y en (18), la C.

Las expresiones (17) y (18) constituyen el espacio de estados lineal para la dinámica de los tanques.

$$\overset{\circ}{X} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = f(x_e, u_e) + \frac{df}{dx}|_{(x_e, u_e)}(x - x_e) + \frac{df}{du}|_{(x_e, u_e)}(u - u_e) \quad (12)$$

$$f(x_e, u_e) = 0 \quad (13)$$

$$\frac{df}{dx}|_{(x_e, u_e)} = \begin{bmatrix} \frac{-k_{12}\sqrt{2g}}{A_1 \cdot 2\sqrt{h_1 - h_2}} & \frac{k_{12}\sqrt{2g}}{A_1 \cdot 2\sqrt{h_1 - h_2}} \\ \frac{k_{12}\sqrt{2g}}{A_2 \cdot 2\sqrt{h_1 - h_2}} & \frac{-k_{12}\sqrt{2g}}{A_2 \cdot 2\sqrt{h_1 - h_2}} - \frac{k_2\sqrt{2g}}{A_2 \cdot 2\sqrt{h_2}} \end{bmatrix}|_{(x_e, u_e)} \quad (14)$$

$$\frac{df}{dx}|_{(x_e, u_e)} \approx \begin{bmatrix} -0,00359 & 0,000359 \\ 0,00976 & -0,04878 \end{bmatrix} \quad (15)$$

$$\frac{df}{du}|_{(x_e, u_e)} = \begin{bmatrix} \frac{1}{A_1} \\ 0 \end{bmatrix}|_{(x_e, u_e)} = \begin{bmatrix} 2,92 \\ 0 \end{bmatrix} \quad (16)$$

$$\overset{\circ}{\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}} = \begin{bmatrix} -0,00359 & 0,000359 \\ 0,00976 & -0,04878 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + \begin{bmatrix} 2,92 \\ 0 \end{bmatrix} u \quad (17)$$

$$y = [0 \ 0,0049] \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \quad (18)$$

Además de usar este modelo matemático, se dispone de un sistema en espacio de estados de misma dimensión estimado a partir del *dataset* de 14400 muestras (explicado más adelante) usando el *toolbox* correspondiente de *MATLAB*. La idea es poder contrastar la calidad de este sistema lineal identificado con la de los modelos no lineales también identificados.

## 2.2. Simulaciones

Para esta monografía se decidió hacer uso de *scripts MATLAB* y el entorno de *Simulink* debido a la versatilidad que trae en lo que es simulación de sistemas, además que es la herramienta usada en las materias de control automático. La figura 2 muestra el sistema no lineal armado, que simula la planta en tiempo real.

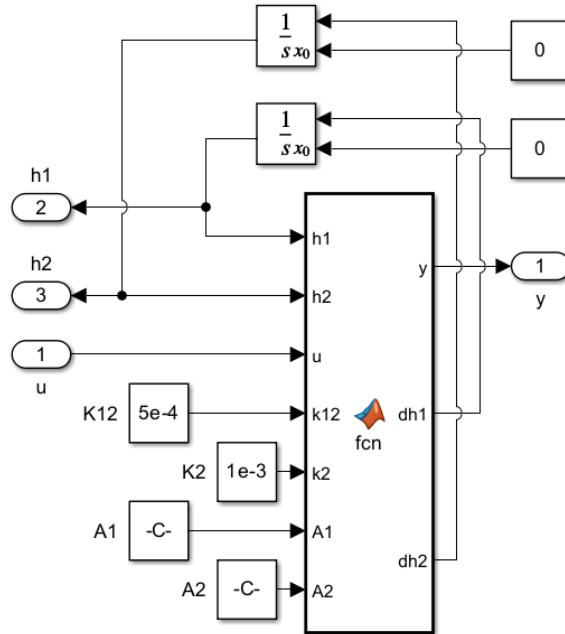


Figura 2: Planta no lineal armada en *simulink*

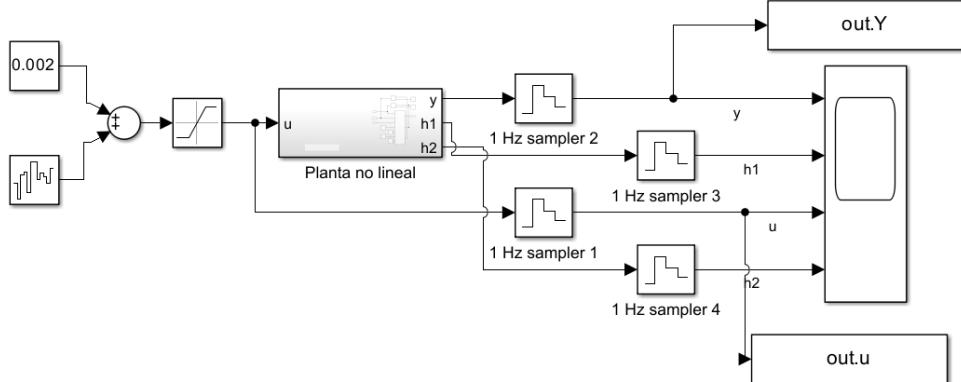


Figura 3: Muestreo de planta no lineal

La figura 3 muestra el sistema de *simulink* que realiza el muestreo de la planta. La entrada  $u$  es una secuencia de escalones de amplitudes aleatorias (ruido blanco gaussiano) alrededor del 10% del punto operativo. Esto se hizo para poder tener una buena variedad de respuestas al escalón, que se espera ayude a que la red aprenda la dinámica subyacente.

Como el muestreo se realizó a 1 Hz, se decidió generar un *dataset* de 4 horas (14400 muestras), y otro de 1 hora (3600 muestras). La intención es observar cuánto empeora la *performance* de la red con menos muestras de entrenamiento. Como ya se explicó, estos sets de muestras están contenidos en una vecindad del punto operativo, y no son representativos de todo el espacio de estados del sistema, sino que solo de la porción simulada.

### 3. Identificación de sistemas

La primer experiencia de esta monografía se centra en la temática de identificación de sistemas a partir de muestras del mismo, abordado desde el lado de redes neuronales. Particularmente, se propone entrenar una red neuronal para que aprenda la dinámica de la planta, y que pueda reproducirla fehacientemente.

#### 3.1. Suposiciones y decisiones

Con el objetivo de asegurar resultados commensurables, los experimentos se armaron y realizaron siguiendo los siguientes preceptos:

1. La única métrica de comparación elegida es el RMSE (ec. (19)). Este se calcula sobre las secuencias de salida de todas las redes entrenadas.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{x} - x)^2} \quad (19)$$

2. El RMSE se calculó sobre la secuencia de datos de salida pero sin incluir el transitorio inicial. Esto es porque el transitorio inicial no está bien representado en los datos de entrenamiento.
3. El RMSE se usó para evaluar el funcionamiento en una región acotada del espacio de estados, no la capacidad de extrapolación.
4. Todos los modelos deben poder usarse de forma auto-regresiva, es decir, realimentados con sus propias salidas. Esto conllevó que los entrenamientos se tuvieran que realizar hasta que se observase convergencia en la simulación.
5. La cantidad de *epochs* y la cantidad de intentos de entrenamiento no son fijas, sino que se ajustaron empíricamente, asegurando el cumplimiento del ítem 2.
6. Los datos se separaron en *train* y *test* en una proporción de 85 % a /15 %. La separación se realizó de forma secuencial, es decir, se tomó el primer 85 % de las muestras para entrenar y el resto para evaluar, manteniendo cierta semblanza al orden temporal. Se podría argumentar que las muestras de evaluación no necesariamente se van a distribuir idénticamente que las de entrenamiento, pero se priorizó variedad de muestras, y el hecho de que los escalones se generaron de forma aleatoria debería ayudar.
7. Se aplicó una normalización en media y varianza a los datos previo a entrenar. Se espera que esto mejore la *performance* (a entradas más espaciadas, pesos más separados) y la convergencia en el *fitting*.

Con esto dicho, se procede a los ensayos.

#### 3.2. MLP de 1 capa oculta

##### 3.2.1. Introducción

El primer *approach* pensado es usar un MLP de 1 capa oculta y entrenarlo en base a los datos de la planta en una configuración auto-regresiva como la de la figura 4. Dado lo visto en la materia, se espera que el MLP logre captar las no-linealidades de la planta con una capacidad que aumenta con la cantidad de neuronas en su capa oculta. En palabras más simples, se tiene la expectativa que los modelos más chicos tengan peor desempeño que los grandes.

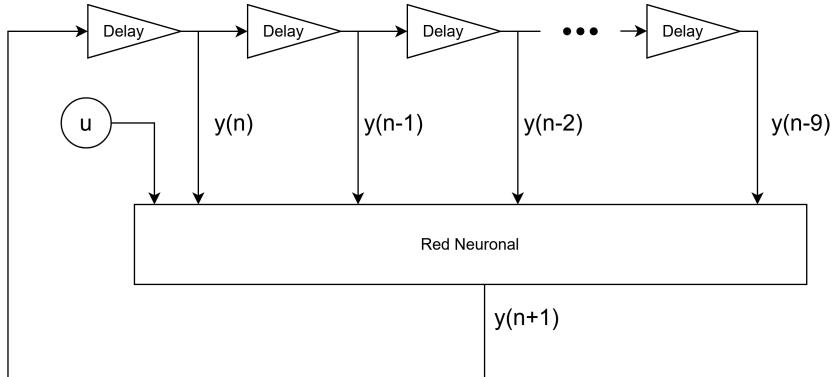


Figura 4: Modelo de red

El vector de datos de entrada  $\mathbf{x}(n)$  elegido consta de 10 muestras previas de  $y$  y la referencia actual  $u$ . La salida esperada de la red es la siguiente muestra, efectivamente entrenando un predictor del siguiente estado.

$$\mathbf{x}(n) = [y(n), y(n - 1), \dots, y(n - 9), u(n)]$$

$$\hat{y}(n + 1) = f(y(n), y(n - 1), \dots, y(n - 9), u(n))$$

Algunos lectores podrían argumentar que sería injusto comparar un modelo lineal que usa la salida actual con uno auto-regresivo de orden 10, pero la elección no es arbitraria, sino que basada en el hecho de que el sistema es lento. Si se usaran pocos retardos, escalones de entrada pequeños causarían que el vector de entrada sea un apilamiento de números casi idénticos, posiblemente dificultando la inferencia de la dinámica. En cambio, una ventana temporal mayor permite que los cambios exponenciales se observan de forma más completa y la red pueda capturar la velocidad de variación correcta.

Todo esto introduce una asimetría respecto del modelo lineal, que usa solo el estado actual. Sin embargo, como el objetivo principal es evaluar la capacidad de las redes neuronales para reproducir la dinámica del sistema, se consideró aceptable priorizar una buena representación temporal.

### 3.2.2. Entrenamiento

Como se adelantó previamente, se decidió crear con 2 *datasets* así que los experimentos van a verse duplicados. Para esta parte se decidió entrenar redes de 20, 10, 5 y 2 preceptrones en la capa oculta.

Algunos puntos interesantes a destacar del entrenamiento son:

- Se entrenó por una cantidad fija de *epochs*, permitiendo interrumpir cuando el gradiente era menor a  $1e - 9$  (que nunca sucedió).
- El proceso de entrenamiento fue manejado por la función *train*<sup>1</sup> del *Deep learning toolbox* de *MATLAB*.
- En general, los modelos pequeños requirieron de más intentos de entrenamiento para llegar a un sistema que cumpliera con los requisitos planteados. El parámetro modificado fue cantidad de *epochs*.

---

<sup>1</sup> Documentación oficial

- Hubo casos en los que los sistemas obtenidos tenían un estado estacionario oscilante, invariante respecto la acción de control. Esto fue considerado un espurio llamativo, aunque lamentablemente no se guardaron imágenes.
- El modelo lineal estimado se obtuvo usando la herramienta de identificación de sistemas propia de *MATLAB*.

### 3.2.3. Resultados

Las figuras del apéndice 7 muestran las señales de interés, incluyendo la entrada  $u$ , la salida  $y$  real, la  $y$  del sistema linealizado y la salida de cada red entrenada. Las imágenes se trasladaron a un apéndice para que no estorben en la lectura.

Mediante inspección visual, la respuesta real (fig. 22) y las respuestas de las redes neuronales aparecen casi indistinguibles, salvo por los transitorios iniciales. Sin embargo, lo mismo se observa para las respuestas obtenidas de los sistemas lineales. Por esto, se presenta un cuadro con los RMSE de todas las respuestas, calculado usando de referencia la salida del sistema no lineal original. Las secuencias fueron tomadas a partir de la muestra número 2000, evitando los transitorios, tal como se dijo que se iba a hacer.

Modelo	RMSE
MLP 20-14400	$2,36 \times 10^{-7}$
MLP 10-14400	$3,24 \times 10^{-7}$
MLP 5-14400	$1,06 \times 10^{-6}$
Modelo lineal (estimado)	$1,80 \times 10^{-6}$
MLP 5-3600	$1,95 \times 10^{-6}$
MLP 2-14400	$2,01 \times 10^{-6}$
MLP 20-3600	$6,71 \times 10^{-6}$
Modelo lineal (matemático)	$6,83 \times 10^{-6}$
MLP 2-3600	$1,88 \times 10^{-5}$
MLP 10-3600	$2,46 \times 10^{-5}$

Cuadro 1: Comparación del error cuadrático medio (RMSE) entre los distintos modelos evaluados

### 3.2.4. Análisis de resultados

A partir de los resultados expuestos en el cuadro 1, se notan tendencias en la *performance* de los varios modelos analizados. Primero, los modelos entrenados en base a más muestras (14400) tienen RMSE inferiores a los entrenados en base al set de 3600 muestras - que es consistente con las hipótesis antes mencionadas de la capacidad de captar no-linealidades.

En segundo lugar, el sistema lineal obtenido matemáticamente (apodado “matemático” en el cuadro), logró mejores ajustes que dos redes por un orden de magnitud. Esto puede deberse a que el modelo lineal es muy bueno o que las redes resultaron *sub-par*. Dicho esto, es inferior que el resto de las redes, incluyendo 2 entrenadas con el *dataset* pequeño, dando a entender que, incluso con pocas muestras se puede lograr un modelo no lineal mejor que uno simple derivado matemáticamente.

El “podio” de los resultados está compuesto de 3 redes neuronales y el modelo lineal estimado, todos entrenadas con el *dataset* grande . La mejor red supera al modelo lineal por 7,6 veces, y no sorprende que sea la red más grande (20 unidades en la capa oculta).

Se cree que el orden de auto-regresividad elegido (10 muestras previas) también influenció los resultados, pero se desconoce su alcance.

En líneas generales, los resultados apuntan a que, para la auto-regresividad elegida, la cantidad de muestras disponibles permite el entrenamiento de modelos más complejos, que a su vez llegan a precisiones superiores que sus contrapartes lineales,

Ahora bien, se debe resaltar que ninguno de los modelos obtenidos es inherentemente malo, sino que el tipo de modelo a usar queda determinado por los requerimientos de la aplicación. Los experimentos recién detallados solo indican que una forma de lograr respuestas más similares a la real es por el uso de sistemas no lineales en la identificación.

### 3.3. Sistema lineal asistido

#### 3.3.1. Introducción

El segundo enfoque ideado surge como una posible avenida de mejora del *approach* anterior: Si se puede lograr un buen desempeño con un modelo lineal, tal vez se puede corregir el error de la estimación lineal por medio de una red neuronal que aprenda el error de estimación. Se supone que, teniendo el sistema lineal como estimador del estado, la red podría dedicar toda su estructura a aprender la codificación que lleva la estimación lineal a la real, logrando una mejor estimación.

La figura 5 muestra el sistema recién descrito. El hiperparámetro  $m$  se va a ajustar manualmente para minimizar el error de entrenamiento y evaluación.

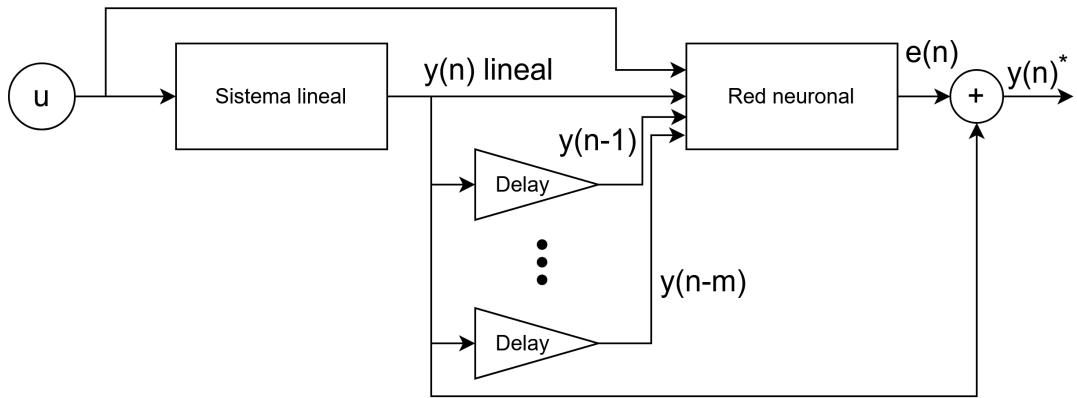


Figura 5: Modelo de red

Inicialmente, se espera que este sistema logre mejor precisión que si solo se usara el lineal, pero no se puede afirmar a priori si va a ser mejor que las otras redes.

Se va a usar el modelo lineal identificado por *MATLAB*, y no el obtenido por medio del desarrollo matemático debido a 2 razones:

- El modelo identificado dió mejores resultados.
- El uso del modelo identificado tiene más sentido en el contexto de esta monografía, que se centra en la identificación de sistemas, y no su derivación matemática.

#### 3.3.2. Entrenamiento

El proceso de *fitting* comenzó generando un conjunto de datos auxiliar, formado por el error de estimación del sistema lineal  $e = y - y_{lin}$ , la salida lineal  $y_{lin}$  y la entrada de los sistemas. A partir de este conjunto, se construyeron los vectores de entrada a las redes ya especificados (los retardos de la salida lineal y la entrada  $u$ ), mientras que como objetivo se empleó el error de estimación.

Como los resultados del entrenamiento no eran buenos, se decidió revertir la función de entrenamiento a su estado original, con los conjuntos de *train*, *test* y *validation*.

#### 3.3.3. Resultados

El cuadro 2 muestra el RMSE de cada uno de los experimentos realizados. Cada prueba intentó mejorar la anterior de alguna forma, sea reduciendo la dimensión de la entrada o la cantidad de neuronas. La columna de “Figuras” referencia las respuestas en el tiempo de cada prueba, ubicadas en el apéndice 8 para que no entorpezcan la lectura.

Prueba	Descripción de la red	RMSE	Figuras
1	MLP de 20 unidades en la capa oculta. Usó $y_{lin}$ desde $n$ hasta $n - 9$ .	$8,71 \times 10^{-6}$	Figs. 33, 34
2	MLP de 20 unidades en la capa oculta. Usó únicamente $y_{lin}(n)$ .	$9,18 \times 10^{-6}$	Figs. 35, 36
3	MLP de 10 unidades en la capa oculta. Usó $y_{lin}$ desde $n$ hasta $n - 4$ .	$6,21 \times 10^{-6}$	Figs. 37, 38
4	MLP de 5 unidades en la capa oculta. Usó $y_{lin}$ desde $n$ hasta $n - 4$ .	$5,78 \times 10^{-6}$	Figs. 39, 40
5	MLP de 5 unidades en la capa oculta. Usó $y_{lin}$ desde $n$ hasta $n - 9$ .	$5,87 \times 10^{-6}$	Figs. 41, 42

Cuadro 2: Resultados de las pruebas

### 3.3.4. Análisis de resultados

Antes que nada, nótese que, si bien los mejores sistemas obtenidos solo son marginálmente mejores que el sistema lineal desarrollado matemáticamente, todos ellos implican un empeoramiento respecto del modelo lineal a partir del cual fue construido el enfoque. En este sentido, la comparación directa pierde relevancia, ya que el deterioro introducido por la identificación sugiere la existencia de un problema con el enfoque.

Las gráficas de las salidas de todas las pruebas hechas muestran espurios generados por la red neuronal, no están en la salida lineal, que empeoran el RMSE y constituyen transiciones de estados que la planta original no podría experimentar nunca, tales como escalones a la salida. Esto claramente es un efecto indeseado introducido por las redes neuronales entrenadas.

Por último, otro punto desfavorable para estos modelos híbridos es que su obtención tomó más tiempo que la de las redes neuronales solas o incluso del modelo lineal estimado.

Todo lo observado apunta a que estos sistemas son inferiores a las alternativas vistas antes tanto en tiempo como en ajuste. No obstante, es probable que, con un cambio de enfoque y arquitectura, la idea de estimar el error  $y - y_{lin}$  de forma no lineal si sea viable.

### 3.4. Discusión general de resultados

En luz de los resultados ya expuestos en los incisos previos, se podrían hacer las siguientes afirmaciones:

- Las redes neuronales sirven para hacer identificación de sistemas, y pueden lograr mejores ajustes que los modelos lineales.
- El tamaño de las redes no es algo de menor importancia ya que parece condicionar la capacidad de ajuste final. Deberá ser elegida a conciencia y por medio de “prueba y error”.
- Se observó una relación positiva entre la cantidad de neuronas en la capa oculta y el ajuste a la salida real, para un orden de auto-regresividad dado (10 muestras previas).
- Se notó que el uso de más muestras permite obtener redes neuronales con mejor ajuste.
- Parecería preferible elegir un *approach* puramente lineal o no lineal para la identificación que intentar fusionarlos. Esto último toma más tiempo y los resultados obtenidos no lo justifican.

Es clave entender que el uso de un sistema no lineal en el área de identificación de sistemas puede o no resultar un exceso. Como se dijo antes, la complejidad del modelo requerido queda completamente determinada por la aplicación. En este caso de 2 tanques de agua, un sistema no lineal es innecesario, pero para procesos “menos lineales” podría darse que el error de la estimación lineal es inadmisible, requiriéndose usar técnicas como las usadas.

## 4. Control de sistemas

Habiendo concluido la sección de identificación de sistemas, se puede avanzar al segundo tema de esta monografía, que es el control. La primer idea es mostrar que el modelo no lineal compuesto por una red neuronal efectivamente sirve para diseñar un controlador. Luego, se propone clonar un controlador (como se hizo con la planta no lineal) para analizar su desempeño a lazo cerrado.

En esta sección se evita la mención de cotas como márgenes de fase, ganancia y estabilidad, debido a que no son relevantes para la monografía y, como indica la p.581, sección 19.11 del libro “*Control System Design*”, muchas nociones comunes para sistemas lineales no aplican para los no lineales.

### 4.1. Diseño de controlador con modelo no lineal identificado como referencia

Para esta experiencia, se tomó la red neuronal con mejor ajuste del inciso 3 y se diseñó un controlador del tipo PID, buscando una respuesta veloz pero no sub-amortiguada. Luego, se verificó su funcionamiento cambiando la red neuronal por el modelo no lineal real.

#### 4.1.1. Teoría del control PID

Citando el capítulo 6 del libro “*Control System Design*”, un controlador PID - proporcional, integral y derivativo - es una estructura de control casi universalmente usada en la industria por su simpleza. Una posible transferencia completa de este tipo de controlador es la siguiente:

$$C_{PID}(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{\gamma_d K_d s + 1}$$

- **Acción proporcional**  $K_p$ : genera una acción de control proporcional al error. Estabiliza plantas ya estables, pero no corrige el error en estado estacionario.
- **Acción integral**  $K_i$ : produce una acción de control proporcional al error acumulado. Corrige el error en estado estacionario para referencias tipo escalón pero puede causar problemas de fase debido al polo en el origen.
- **Acción derivativa**  $K_d$ : devuelve una acción de control proporcional a la derivada del error. En la práctica, la existencia de ruido en los sensores y mediciones puede introducir derivadas elevadas y, por ende, acciones de control desmedidas. Hay que usarla con cuidado y filtrando adecuadamente.

No se eligió otra estrategia de control porque el sistema es simple y estable a lazo abierto.

#### 4.1.2. Diseño del controlador y resultados

El diseño de un PID es un proceso iterativo en el que se van ajustando los coeficientes  $K_p$ ,  $K_i$  y  $K_d$  hasta obtener la respuesta deseada. En este caso particular, el proceso se llevó a cabo en *simulink*, y la figura 6 muestra el lazo final. La imagen 7 muestra las salidas a lazo abierto y cerrado, la acción de control y la referencia.

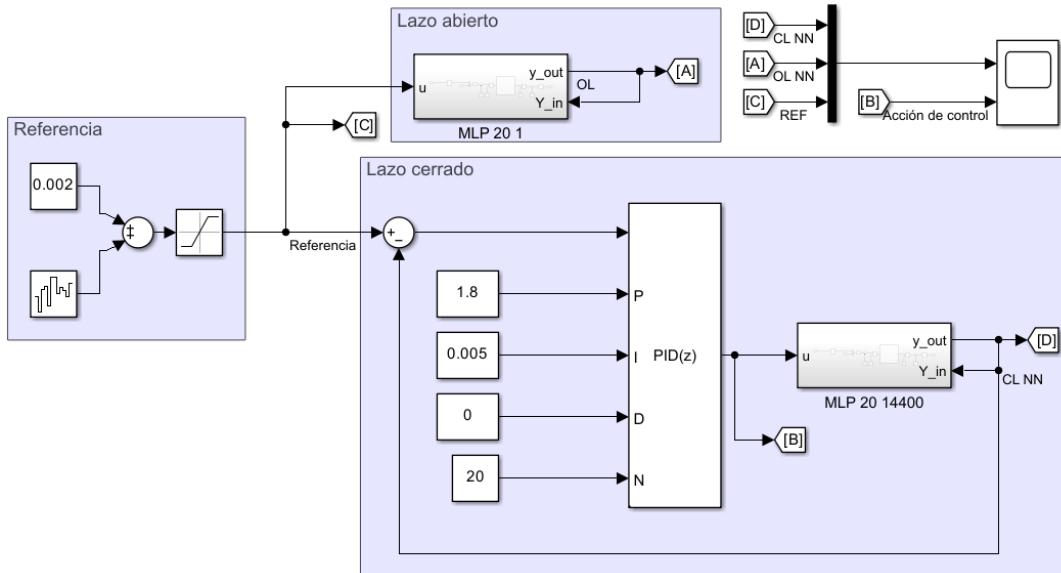


Figura 6: Esquema de control de la red neuronal por PID

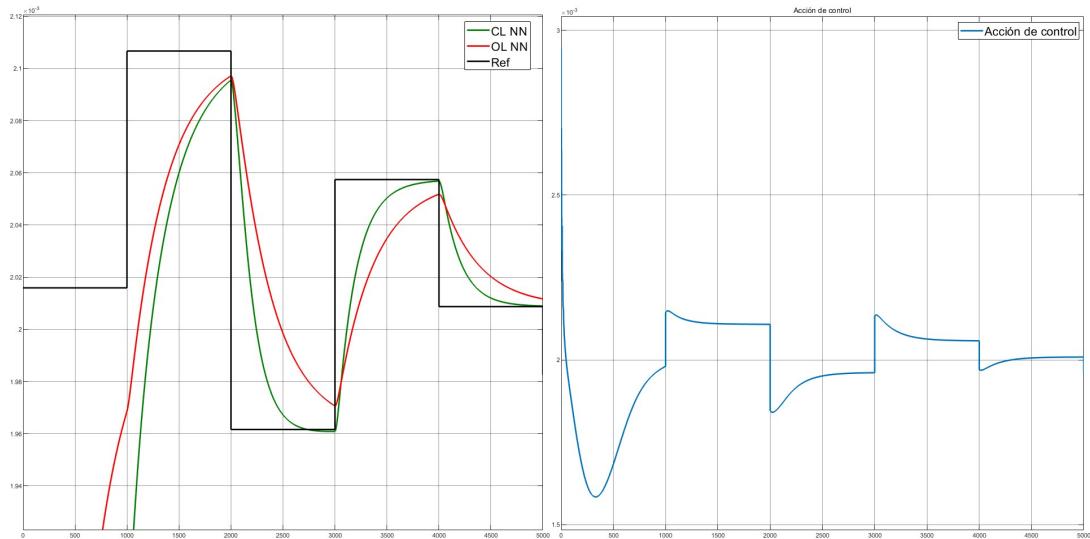


Figura 7: Simulación a lazo cerrado - planta armada con red neuronal

Nótese que la respuesta verde, correspondiente a la salida a lazo cerrado, llega antes que la roja (lazo abierto) al estado estacionario fijado por la referencia. Esto indica que el PID logró acelerar el sistema sin volverlo extremadamente agresivo u oscilante. Además, la acción de control es acotada, sin divergencias. La constante  $K_d$  resultó nula, por lo que el control es en realidad un PI pero se va a seguir referenciando como PID por el bloque de *simulink*.

Las figuras 8 y 9 muestran el sistema armado con la planta no lineal y las salidas, tal como antes. Se observa que el controlador diseñado sobre el modelo no lineal, conformado por redes neuronales, funcionó con la planta real. Esto está alineado con las observaciones de la experiencia de identificación, y corrobora que el sistema identificado por una red neuronal sirve para el diseño de controladores para un punto de operación dado.

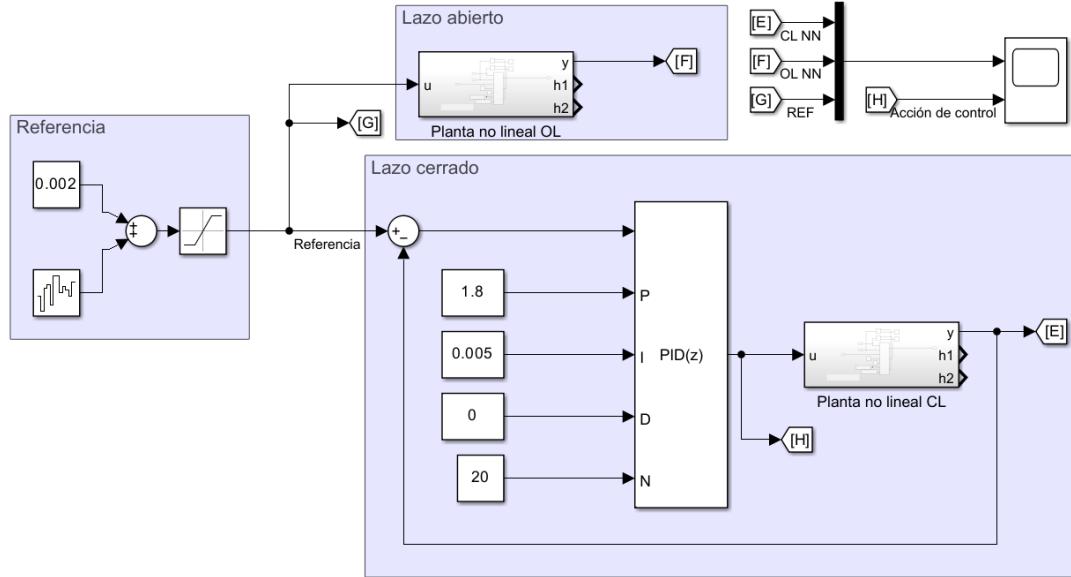


Figura 8: Esquema de control de la planta no lineal por PID

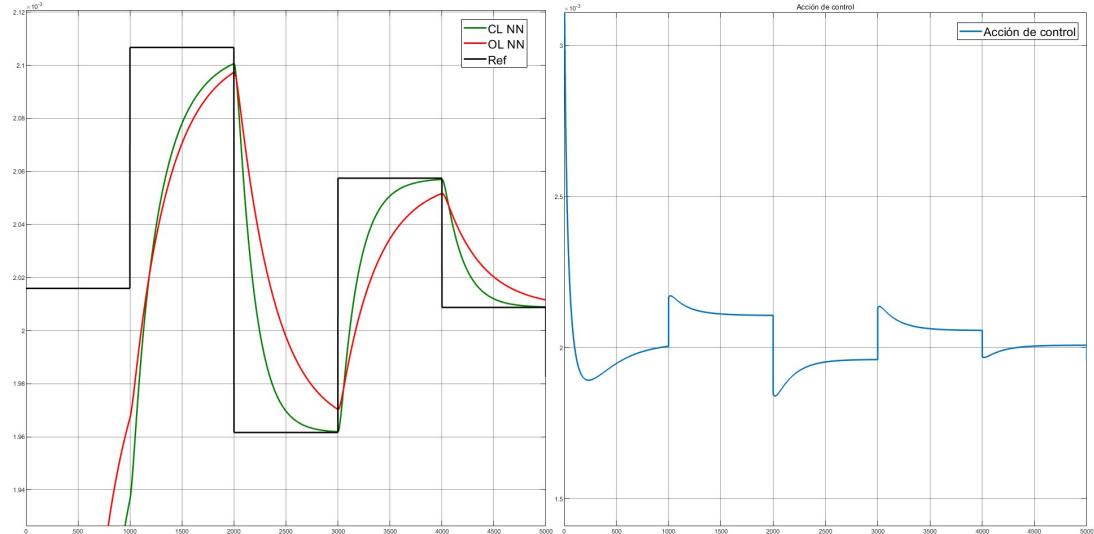


Figura 9: Simulación a lazo cerrado - planta no lineal real

## 4.2. Clonado del controlador

Teniendo un controlador funcional, se procedió a generar muestras de este y clonarlo por medio de una red neuronal. El procedimiento es idéntico al empleado en identificación de sistemas, dónde lo novedoso está en estudiar si una red neuronal puede reemplazar al controlador. Se espera que la red pueda emular al controlador como lo hizo con la planta, pero se duda que capte la esencia de lo que realmente hace (véase la transferencia de un PID previamente estipulada).

### 4.2.1. Entrenamiento

Usando el esquema de la figura 8, se armó un set de datos de 14400 muestras constituido por la referencia, el error (referencia menos salida), la salida del controlador y la salida de la planta. Se mantuvo un tamaño idéntico al que dió los mejores resultados en la identificación de sistemas.

Como arquitectura, se eligió una red neuronal auto-regresiva de 20 neuronas en su capa oculta que usa de entrada las últimas 10 salidas de la planta, aparte de la referencia y error. Se espera que la redundancia de información de usar la referencia, salida y error mejore el resultado ya que la red no necesita aprender el error a partir de las otras.

La figura 10 muestra el error de entrenamiento/testeo. Se observa que corta en un punto de relativa convergencia (derivada baja). La imagen 11 contiene unos gráficos que muestran la *performance* de la red respecto de los datos. La identificación parece haber funcionado porque el error de predicción es acotado, lo que da esperanza de que el controlador clonado funcione.

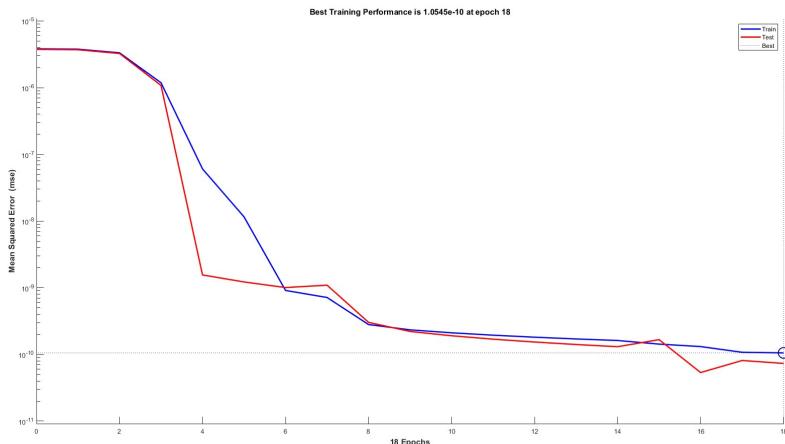


Figura 10: Error de entrenamiento y evaluación de la red

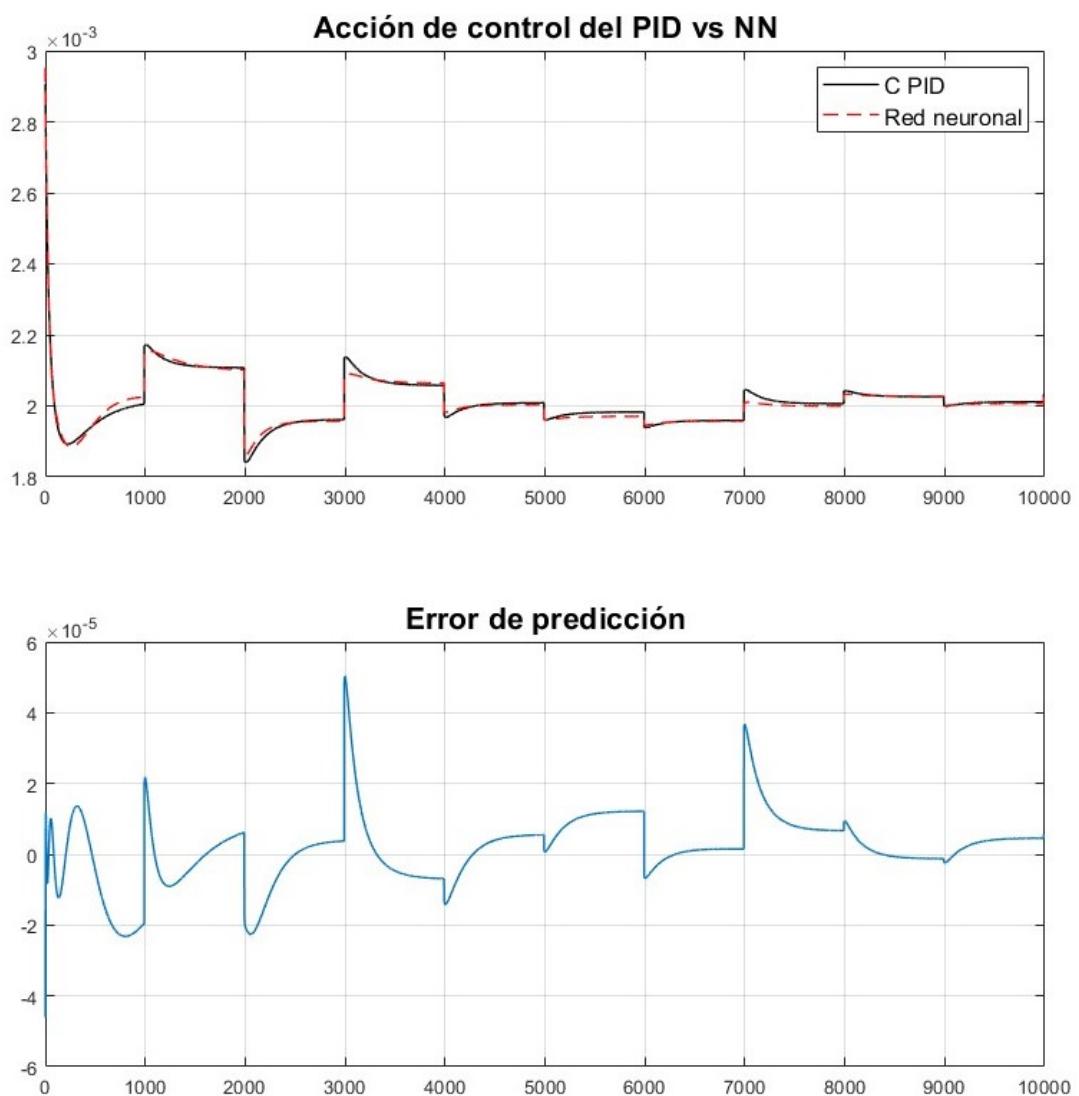


Figura 11: Evaluación de la red

#### 4.2.2. Resultados

La figura 12 muestra la respuesta al escalón del controlador real y del clonado. Son prácticamente coincidentes, lo que es un indicio de que el entrenamiento funcionó. La figura 13 muestra las acciones de control para las respuestas al escalón; no se ven espurios inadecuados.

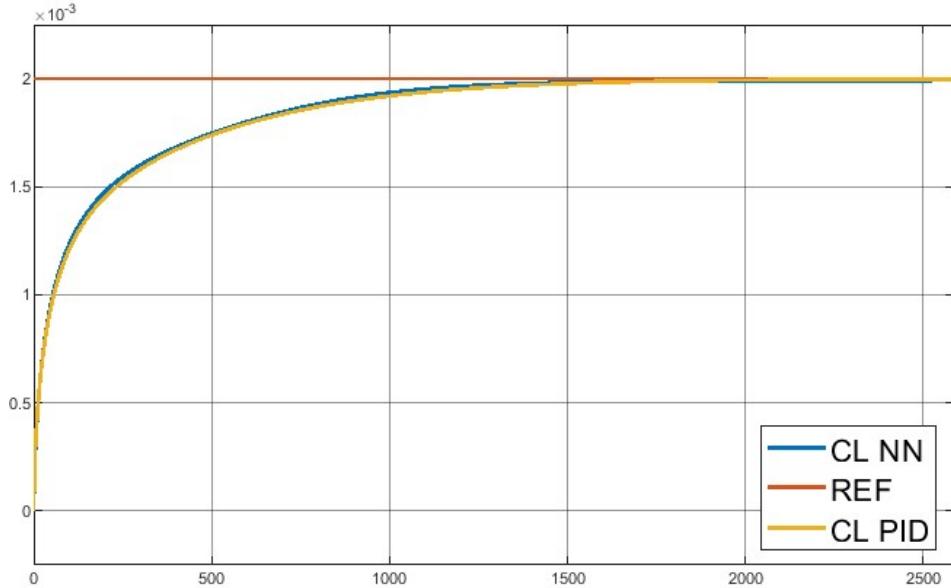


Figura 12: Respuestas al escalón

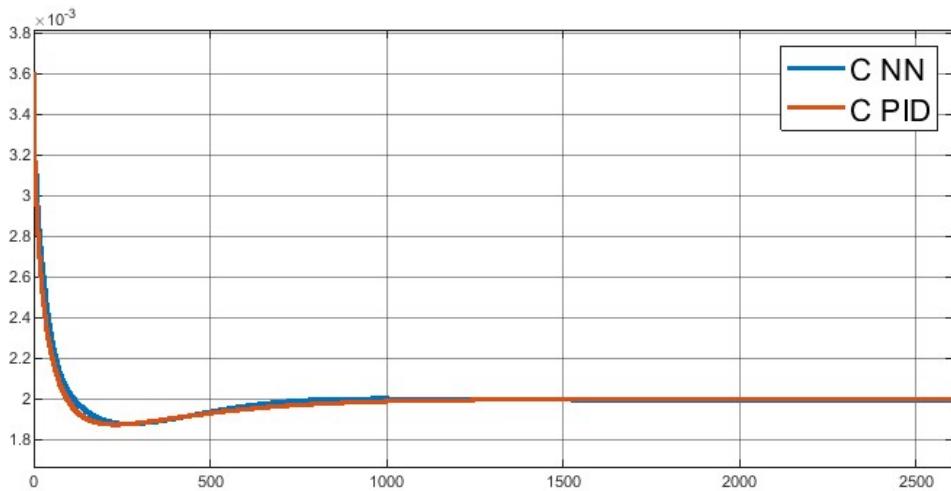


Figura 13: Respuestas al escalón - acciones de control

Ahora bien, la imagen 14 muestra las salidas a lazo cerrado cuando varía la referencia. Se empiezan a observar problemas con el controlador clonado, principalmente que su error en estado estacionario no es nulo, como el del controlador original (el PI/PID). Es más, la figura 15, simulada con el esquema 16<sup>2</sup>, indica que hay algo fundamentalmente mal, ya que el controlador clonado no comprende que el ruido introducido en el lazo se debe eliminar y, en cambio, está respondiendo a estos como si fueran cambios en la referencia.

<sup>2</sup>Las perturbaciones introducidas son de igual amplitud que los cambios en la referencia de antes ( $\pm 10\%$  del estado estacionario) pero centradas en el origen.

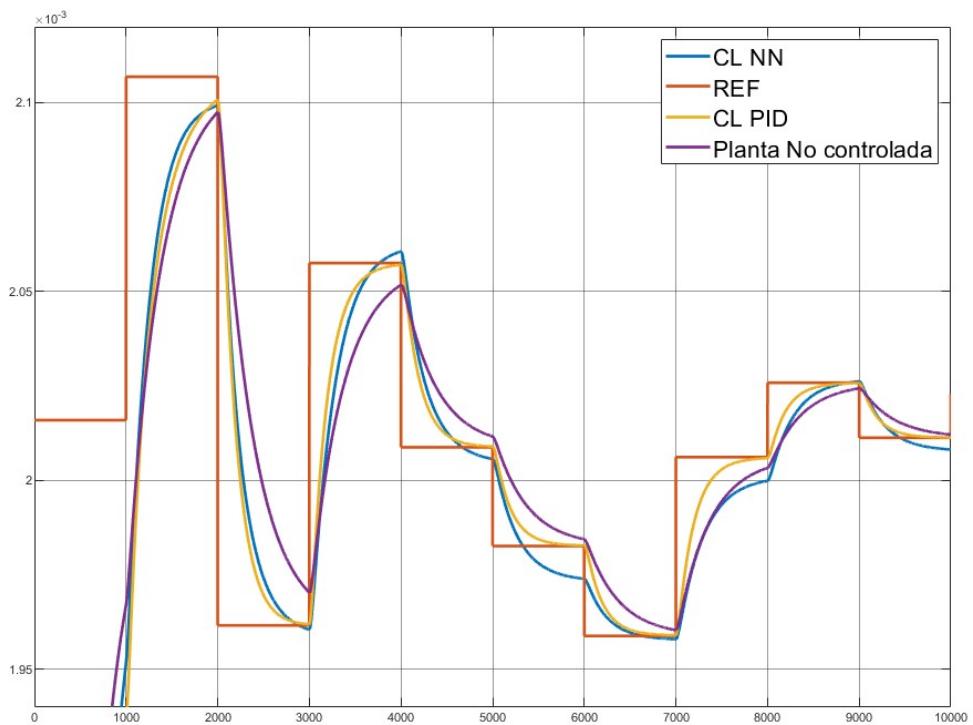


Figura 14: Respuesta a cambios en la referencia

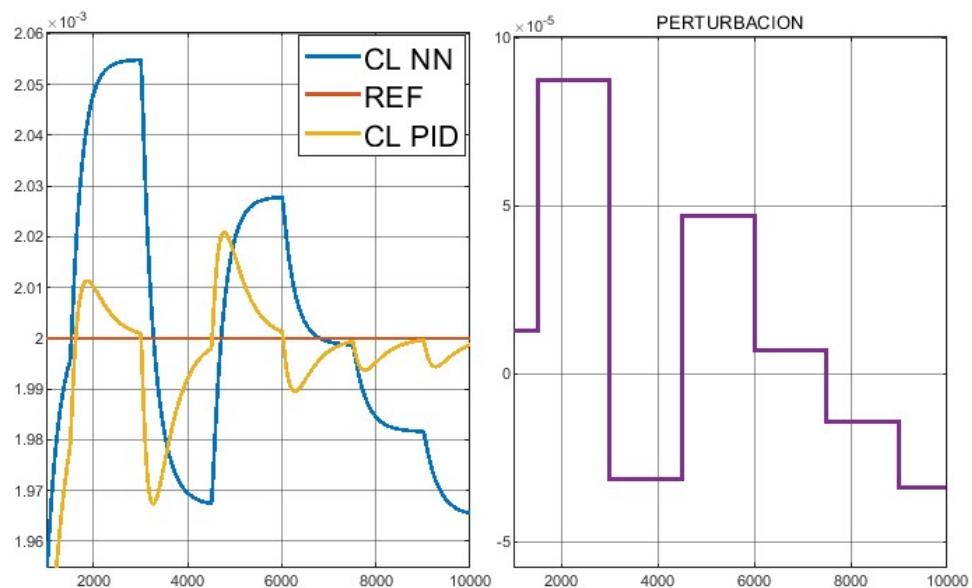


Figura 15: Respuesta a ruido en la salida del controlador

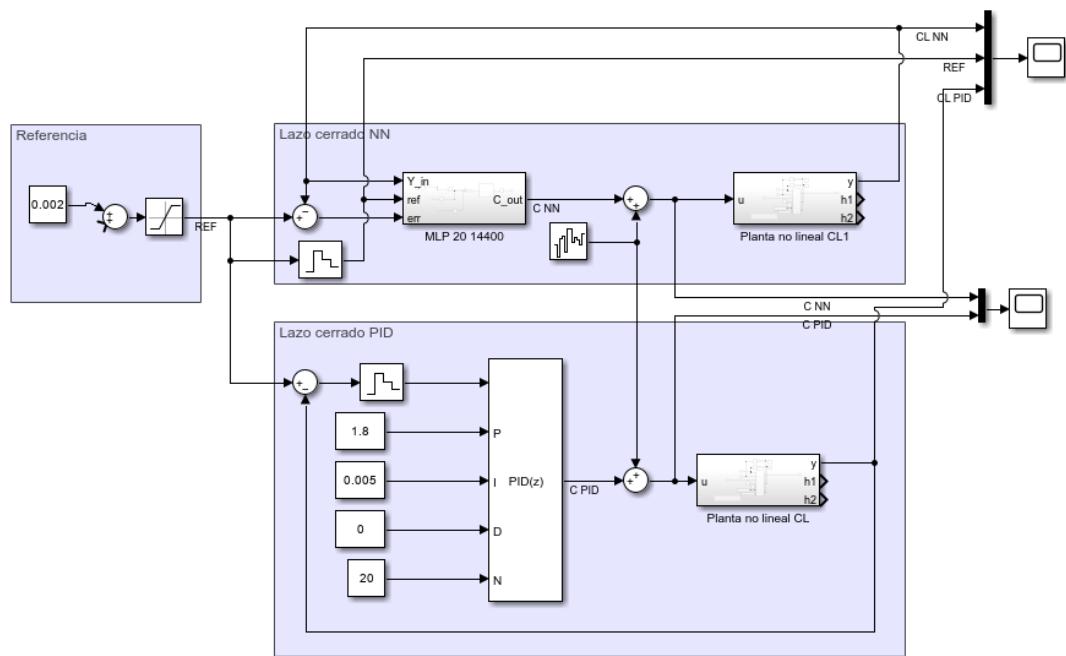


Figura 16: Esquema de simulación de ruido en la salida del controlador

En retrospectiva, el *dataset* nunca debería haber incluido la referencia, ya que es invariante con las perturbaciones en el lazo, algo que no se pensó hasta que se hicieron los experimentos. por completitud, se entrenó otra red usando un set de datos que usa solo el error del lazo y salida, e incluye muestras obtenidas perturbando a la entrada de la planta (mitad de cambios en la referencia, mitad perturbaciones).

Los resultados se pueden ver en las figuras 17, 18, 19 y 20. Parece haber “aprendido” el control desde la referencia como antes, aunque con error en estado estacionario, pero no se logró que rechace perturbaciones a la salida del controlador/entrada a la planta.

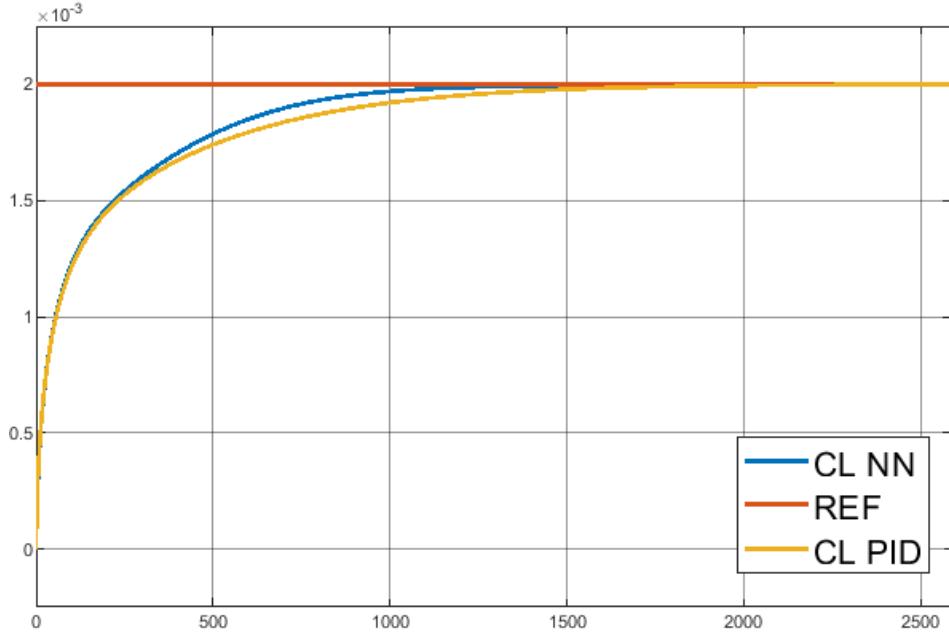


Figura 17: Respuesta al escalón

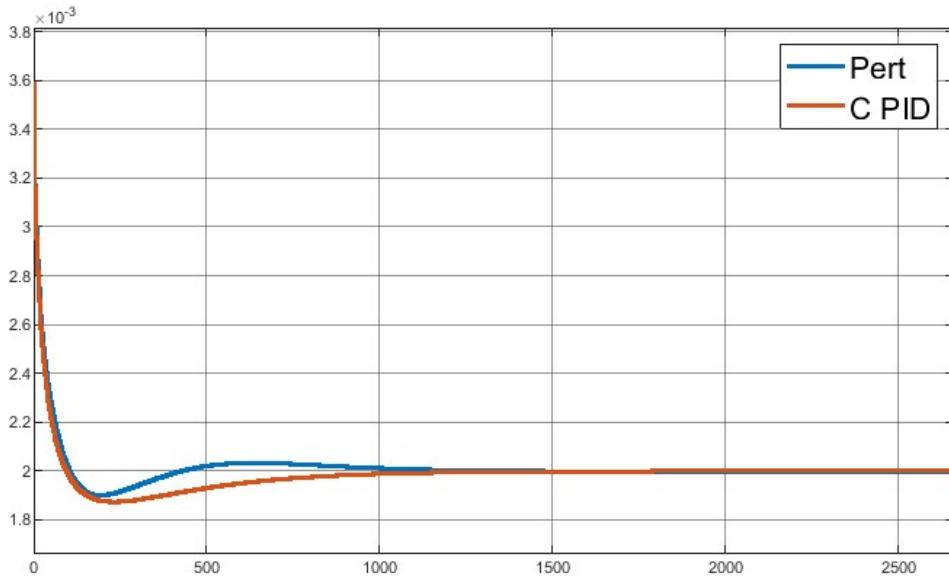


Figura 18: Respuesta al escalón

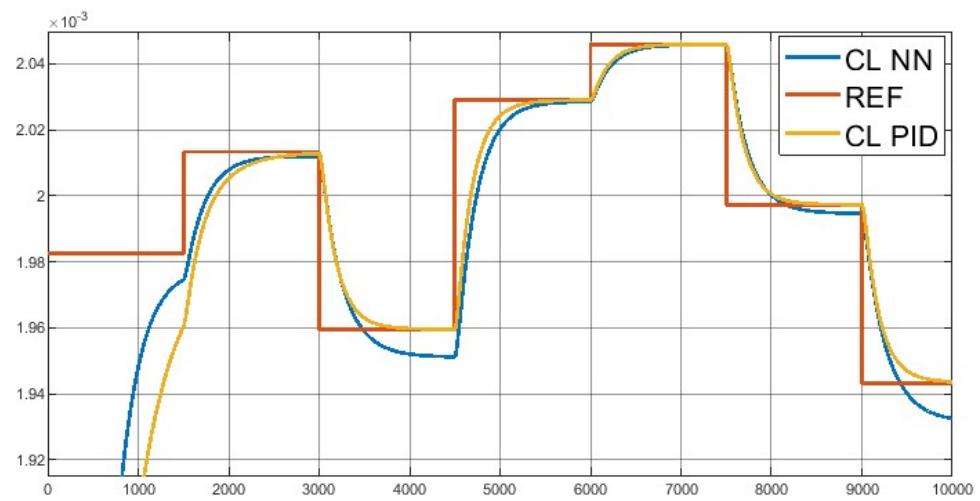


Figura 19: Respuesta a cambios en la referencia

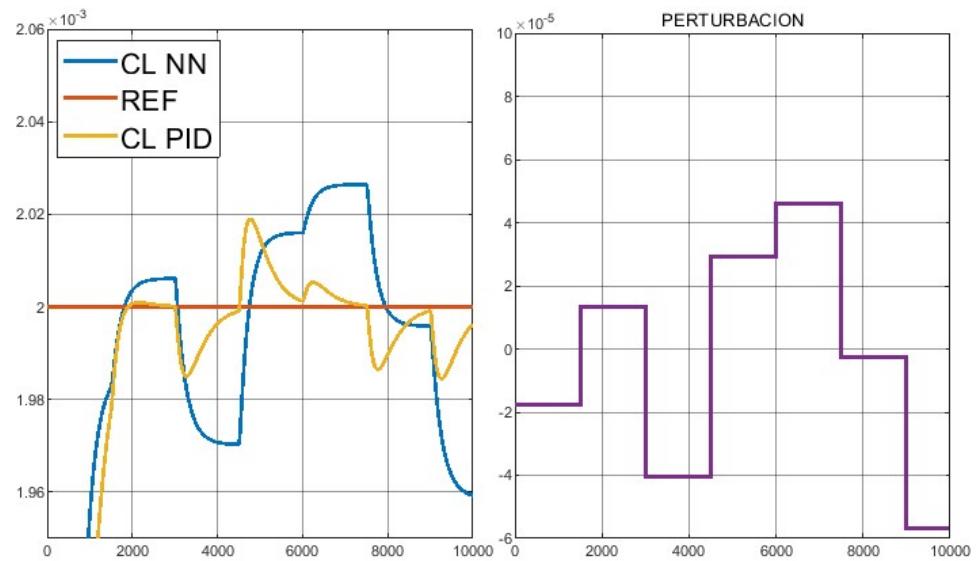


Figura 20: Respuesta a ruido en la salida del controlador (perturbaciones)

#### 4.2.3. Análisis de resultados

Los resultados obtenidos no favorecen la idea de clonar controladores con redes neuronales. Aunque se mantiene la posibilidad de controlar en base a cambios en la referencia, los controladores obtenidos también interpretaron las perturbaciones como acciones de control válidas. Esto resulta en un controlador que amplifica el ruido proveniente del exterior y, por ende, no utilizable en la práctica.

Las redes obtenidas no son más que sistemas que intentaron memorizar salidas a partir de entradas, razón por la cual el error en estado estacionario no es nulo, excepto en ciertos casos. De igual manera se esperaba que las redes no lograran captar la complejidad del PID entero, y es razonable que no pueda copiar una operación de integración (la parte “I” del PID). Es posible que cambios en la arquitectura ayuden con el clonado, siendo uno posible el agregado de un integrador como entrada de la red. De esta manera, la red podría usar la integral del error como parte del vector de entrada y aprender la acción de control que lo minimiza.

Igualmente, se destaca que la experiencia si revalidó las capacidades de identificación de sistemas de las redes neuronales. Si no fuera por estas, tampoco se podría controlar desde la referencia como se hizo.

## 5. Conclusiones

En esta monografía se abordó la identificación y el control de un sistema no lineal de tanques acoplados por medio de modelos lineales y sistemas basados en redes neuronales.

La primer experiencia se centró en el entrenamiento de modelos no lineales relativamente variados, y su posterior comparación con sistemas lineales, derivados de la planta original. Se observó que, con la arquitectura y cantidad demuestras adecuadas, las redes neuronales logran captar adecuadamente las dinámicas del sistema dentro del régimen considerado, superando a los modelos lineales. Se destaca que el uso de redes neuronales requiere de más prueba y error que la linealización, pudiendo ser desfavorable en ciertos contextos más acotados en tiempo.

Asimismo, se exploró un enfoque híbrido, que combina modelos lineales con redes neuronales encargadas de estimar el error de modelado. Los resultados mostraron que este esquema no mejora el desempeño del modelo lineal, y solo introduce dinámicas indeseadas.

En lo que respecta el diseño del control, la segunda experiencia, se demostró que un PID diseñado sobre un modelo no lineal de la planta puede ser aplicado exitosamente sobre la planta real. Esto valida el uso de los modelos no lineales como herramienta de identificación de sistemas.

Finalmente, se analizó la posibilidad de clonar el controlador PID por medio de una red neuronal. Los resultados fueron desfavorables: el sistema clonado actúa de forma adecuada ante cambios en la referencia pero se confunde las perturbaciones de lazo con cambios en la referencia, y generando acciones de control indebidas. Esto trae a luz ciertas de las limitaciones de las redes neuronales a la hora de tener que clonar estructuras complejas.

En conjunto, los resultados muestran que las redes neuronales constituyen una herramienta valiosa para la identificación de sistemas no lineales, siempre que su utilización se realice con criterio y analizando el compromiso entre complejidad y desempeño.

## 6. Referencias

- Material de la materia “Control Automático”.
- Control System Design. Autores: Graham C. Goodwin, Stefan F. Graebe, Mario E. Salgado. Año: 2000
- “[Neural network \(machine learning\)](#)”, Wikipedia
- “[Machine learning](#)”, Wikipedia
- “[Neural network](#)”, Wikipedia
- “[Introduction to neural network control systems](#)”, *MATLAB resources*

## 7. Apéndice 1 - Resultados de la primer experiencia de identificación

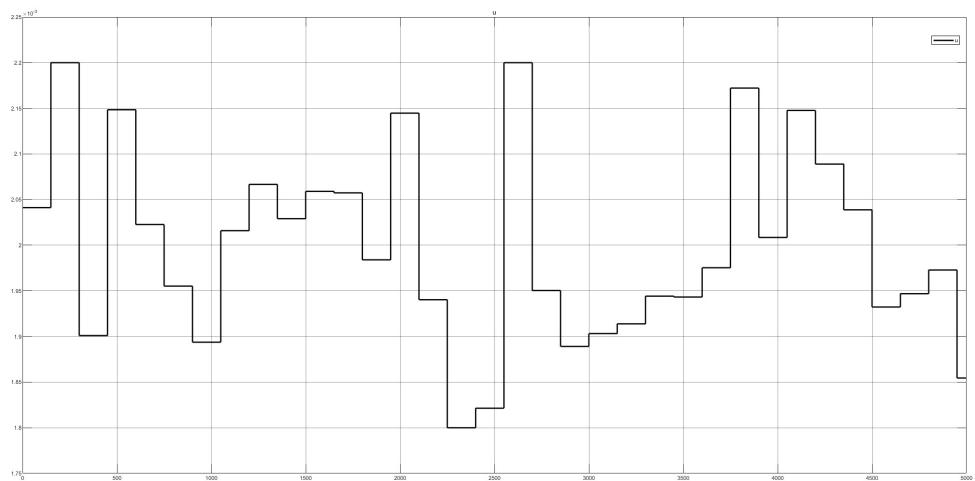


Figura 21: Entrada  $u$

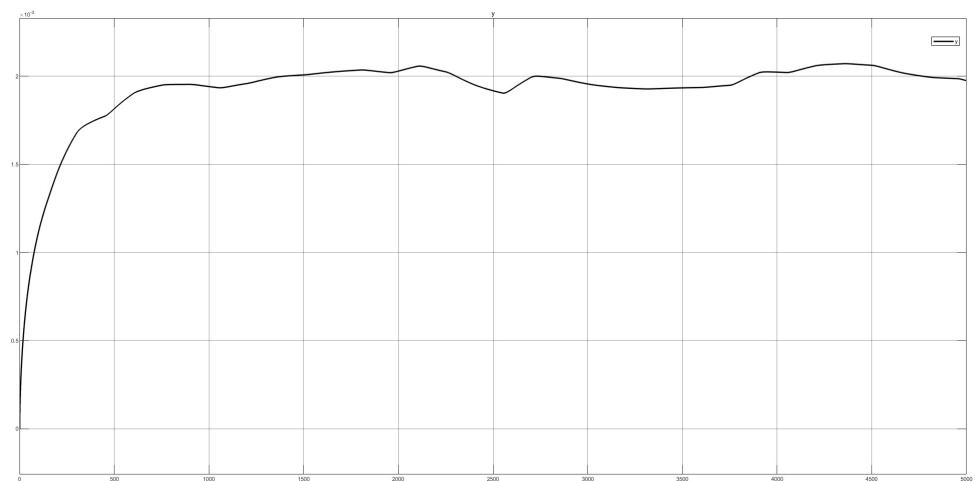


Figura 22: Salida  $y$  de la planta no lineal

## 8. Apéndice 2 - Resultados de la segunda experiencia de identificación

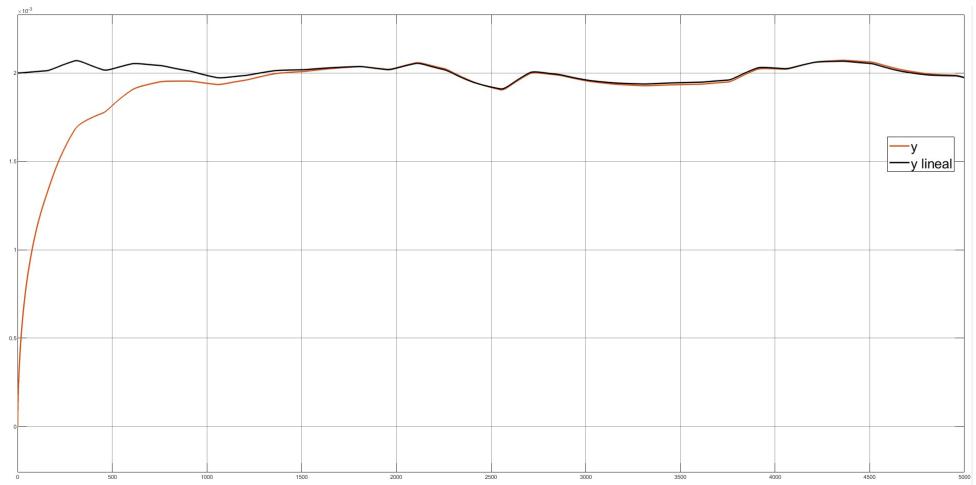


Figura 23: Comparación entre salida no lineal y lineal

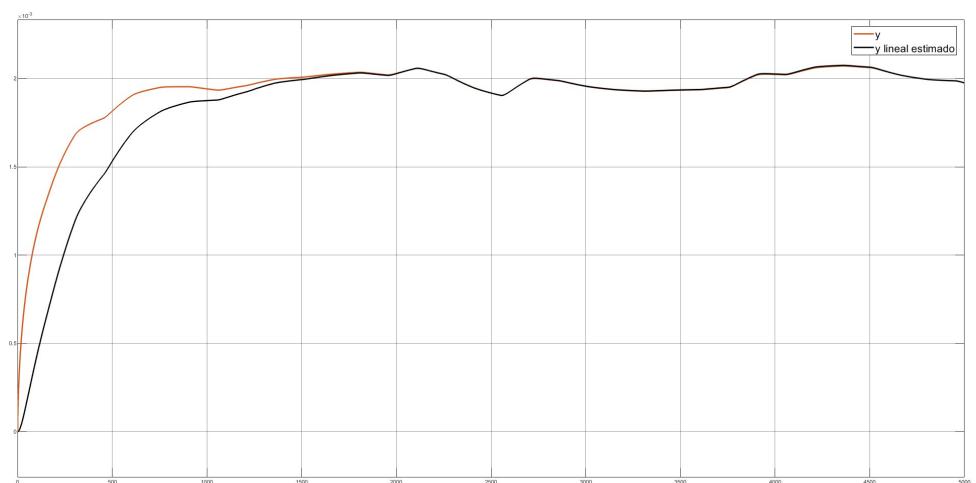


Figura 24: Comparación entre salida no lineal y lineal del modelo estimado

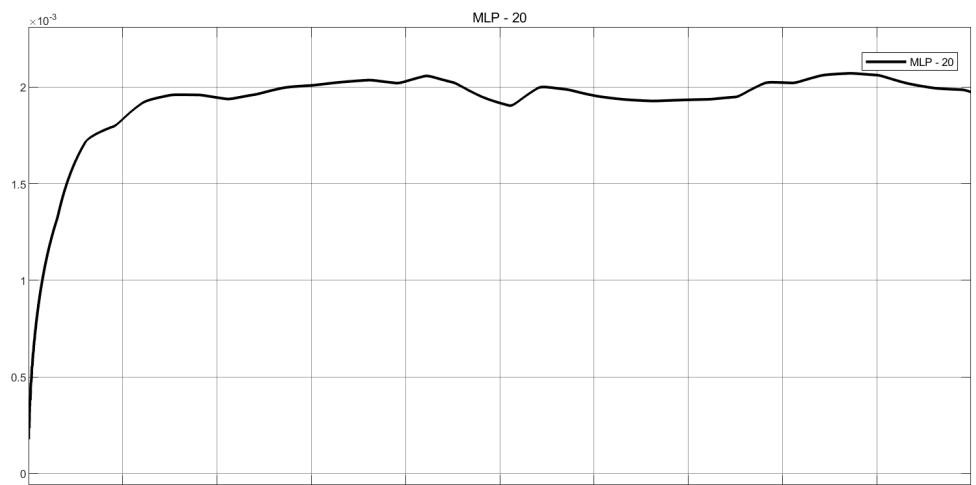


Figura 25: Salida del MLP de 20 neuronas en capa oculta - *dataset* de 14400 muestras

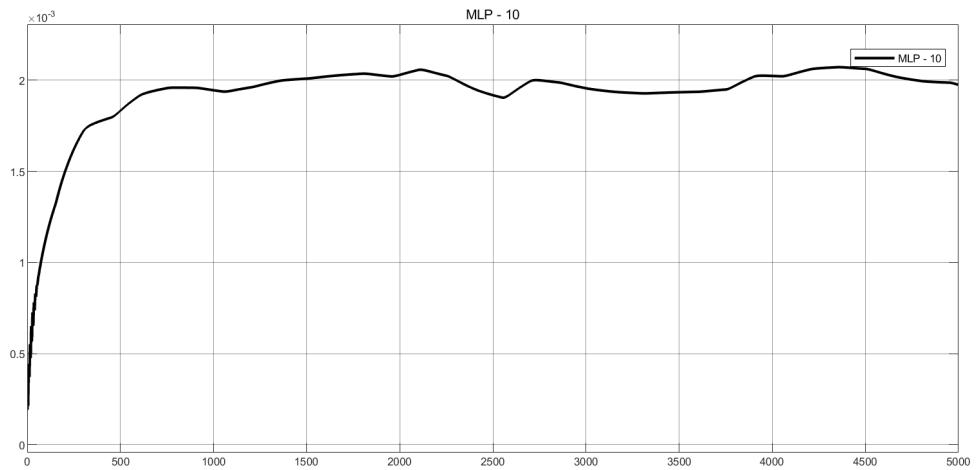


Figura 26: Salida del MLP de 10 neuronas en capa oculta - *dataset* de 14400 muestras

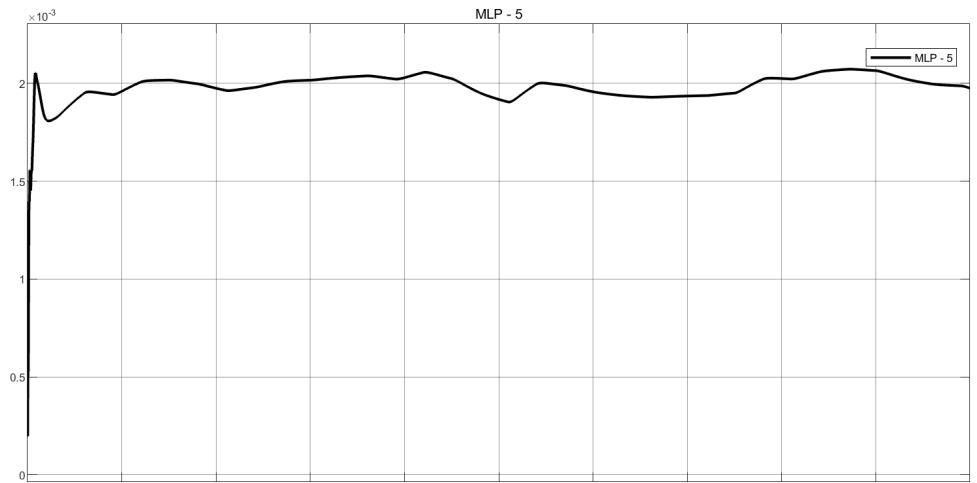


Figura 27: Salida del MLP de 5 neuronas en capa oculta - *dataset* de 14400 muestras

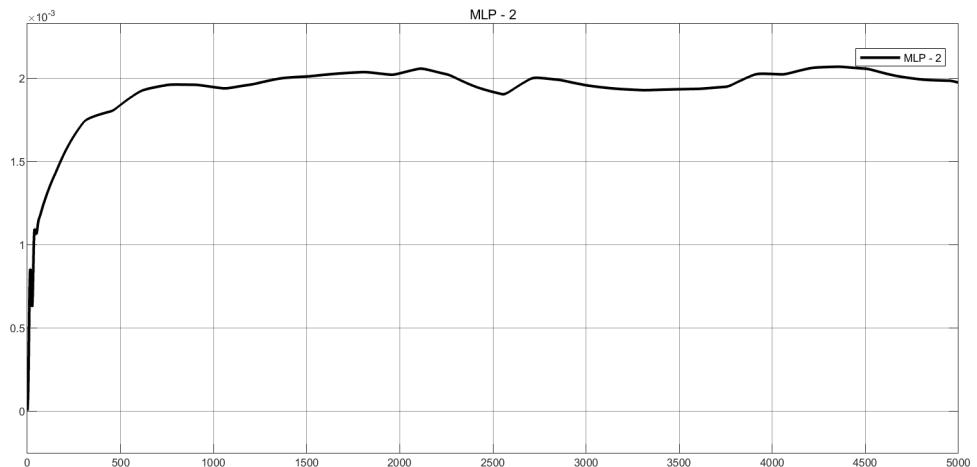


Figura 28: Salida del MLP de 2 neuronas en capa oculta - *dataset* de 14400 muestras

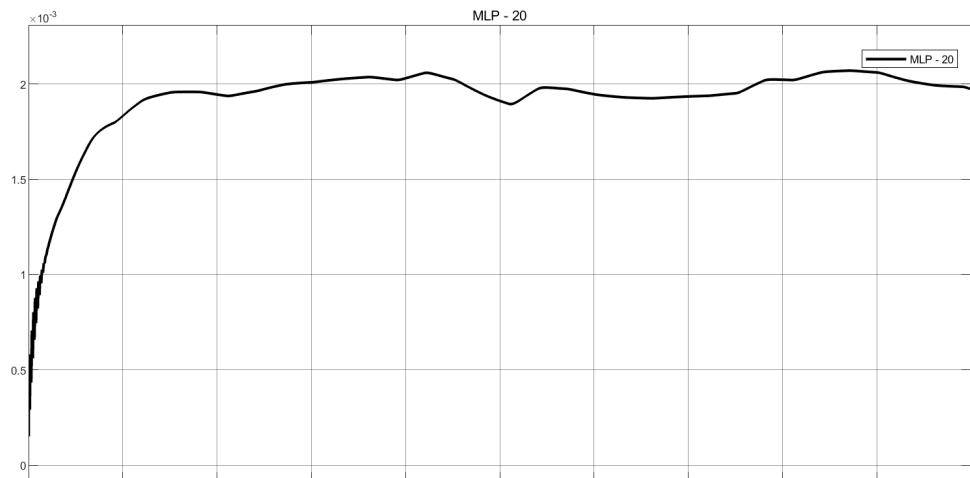


Figura 29: Salida del MLP de 20 neuronas en capa oculta - *dataset* de 3600 muestras

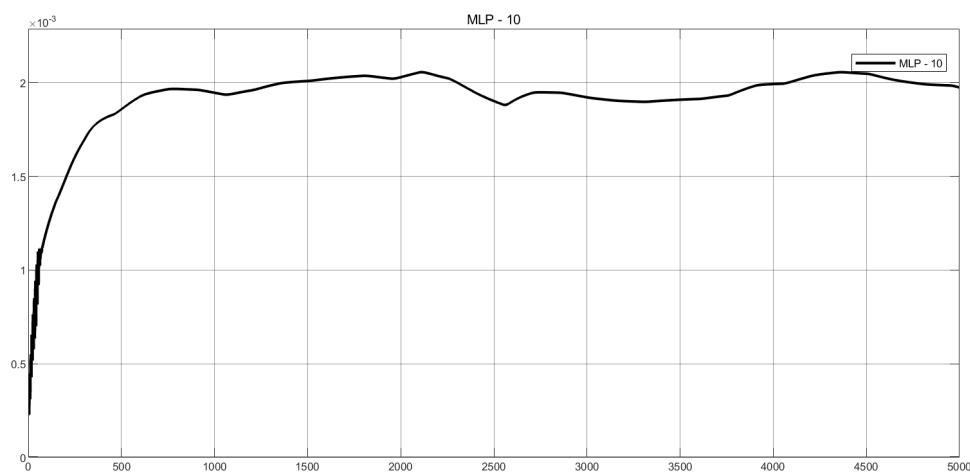


Figura 30: Salida del MLP de 10 neuronas en capa oculta - *dataset* de 3600 muestras

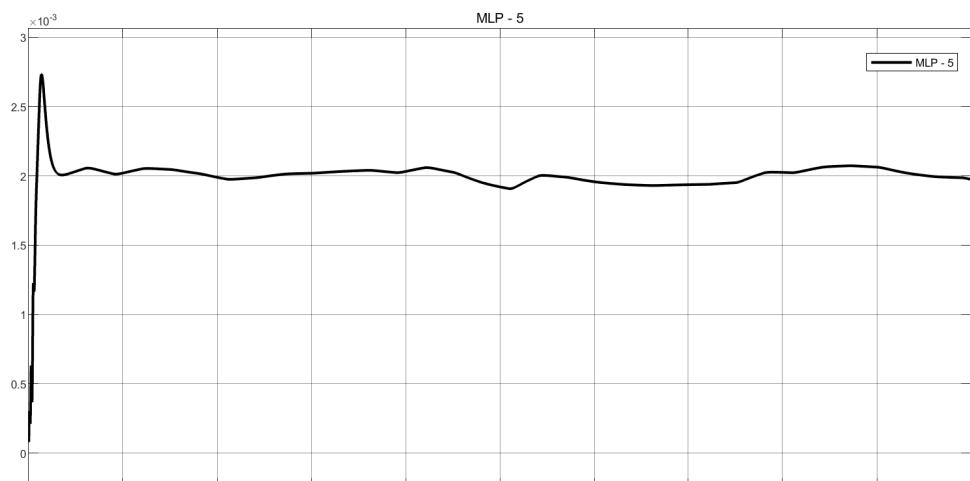


Figura 31: Salida del MLP de 5 neuronas en capa oculta - *dataset* de 3600 muestras

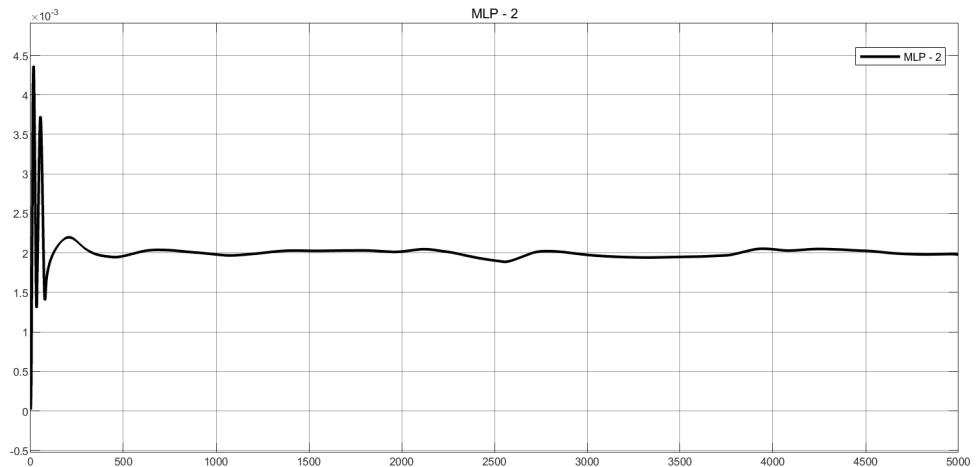


Figura 32: Salida del MLP de 2 neuronas en capa oculta - *dataset* de 3600 muestras

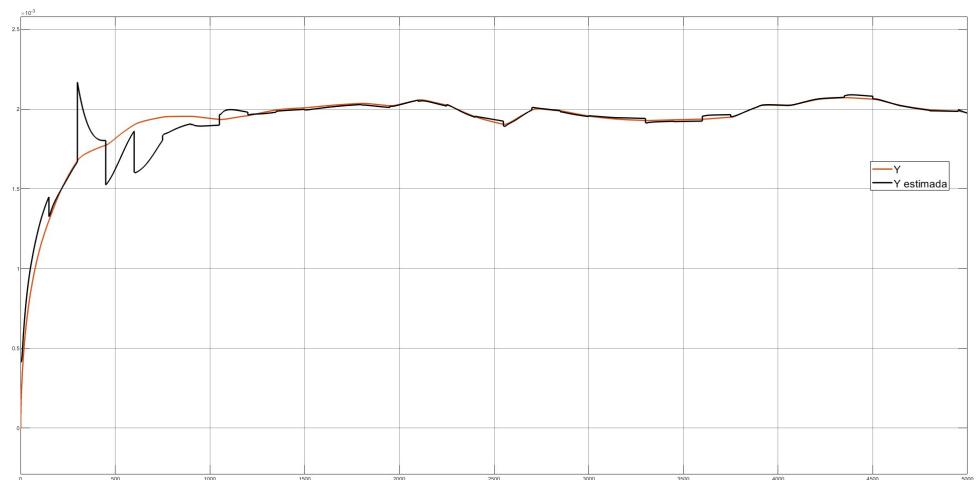


Figura 33: Comparación de salidas - prueba 1

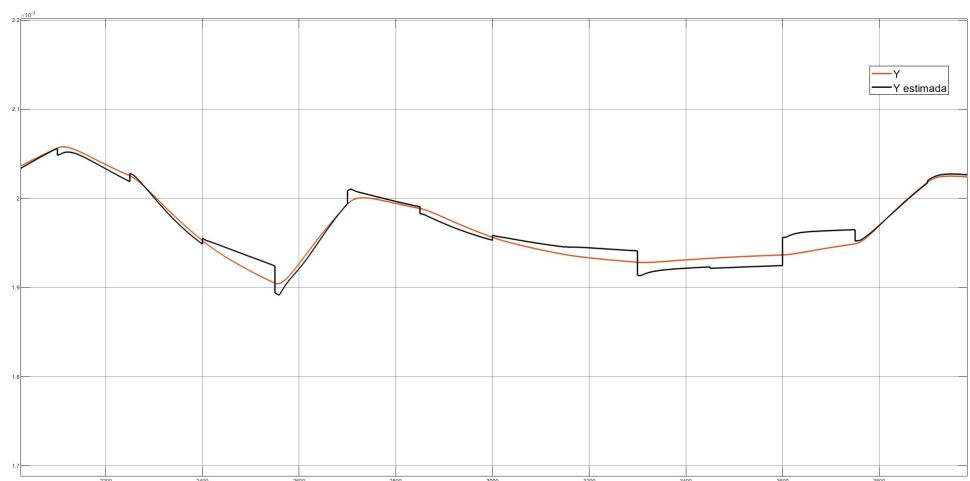


Figura 34: Comparación de salidas - prueba 1

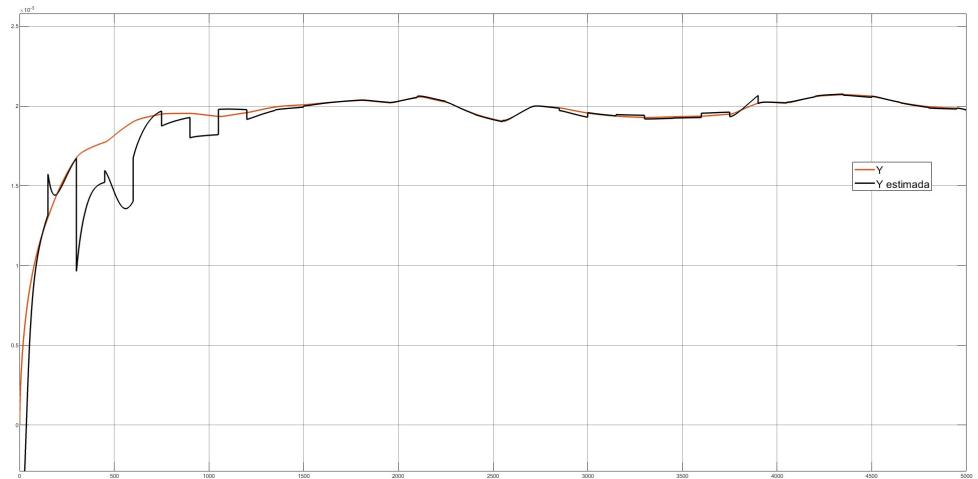


Figura 35: Comparación de salidas - prueba 2

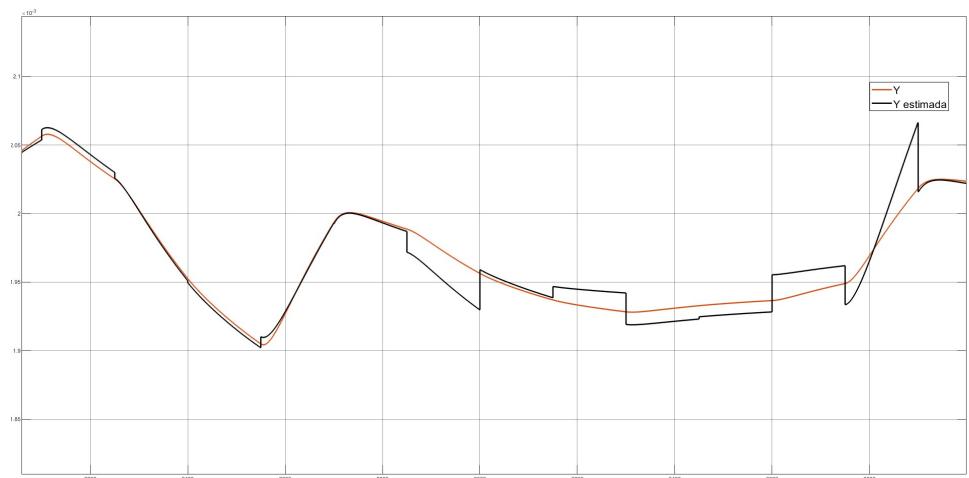


Figura 36: Comparación de salidas - prueba 2

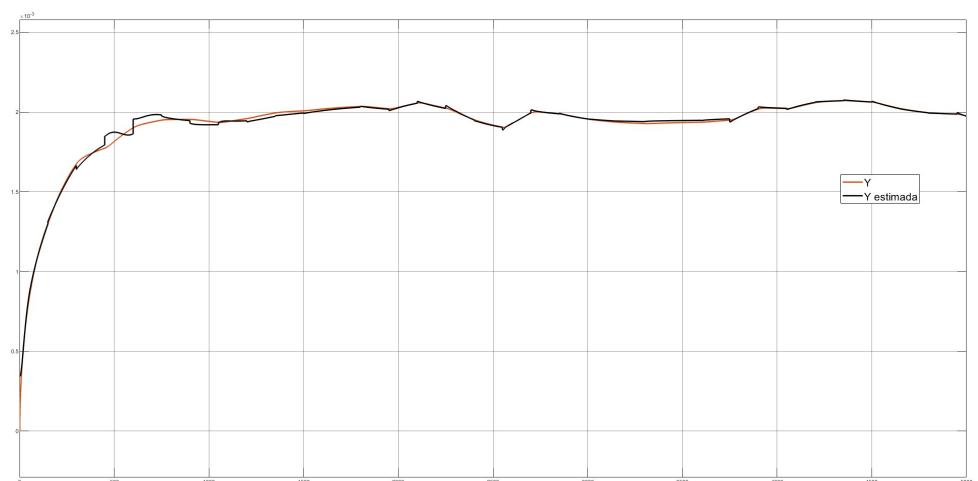


Figura 37: Comparación de salidas - prueba 3

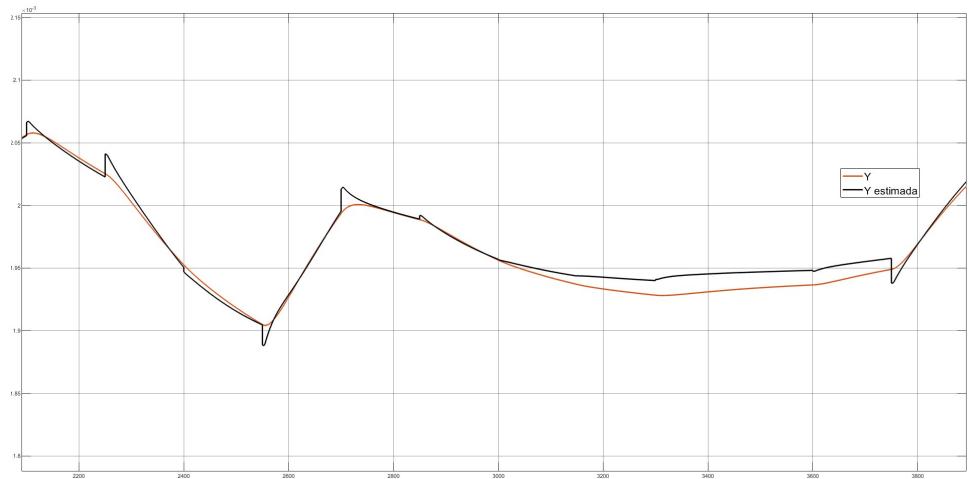


Figura 38: Comparación de salidas - prueba 3

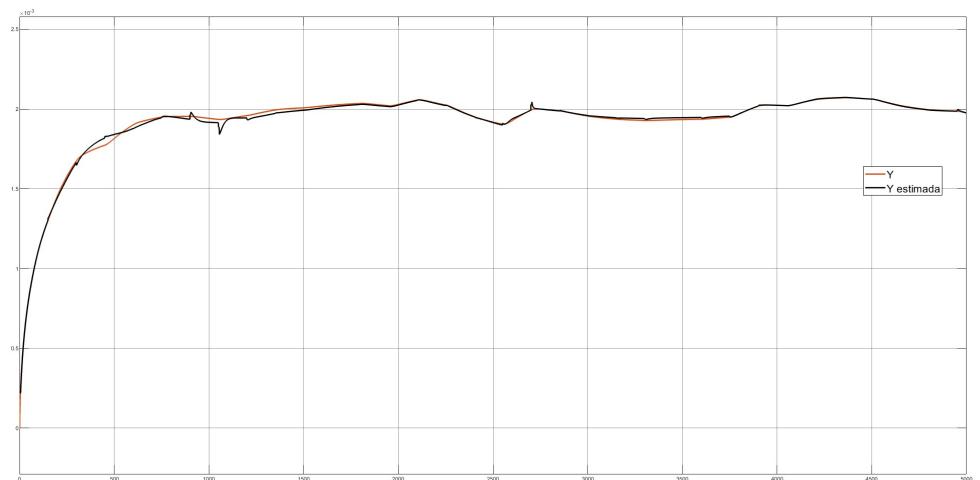


Figura 39: Comparación de salidas - prueba 4

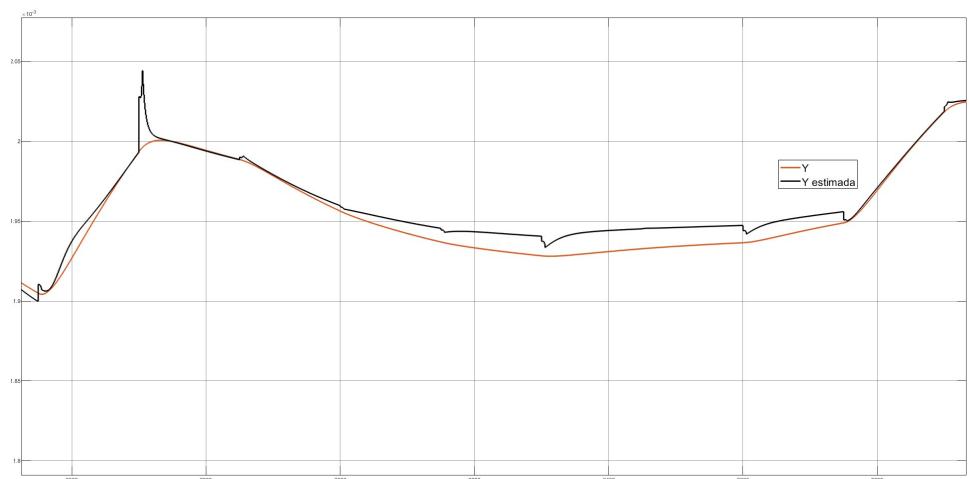


Figura 40: Comparación de salidas - prueba 4

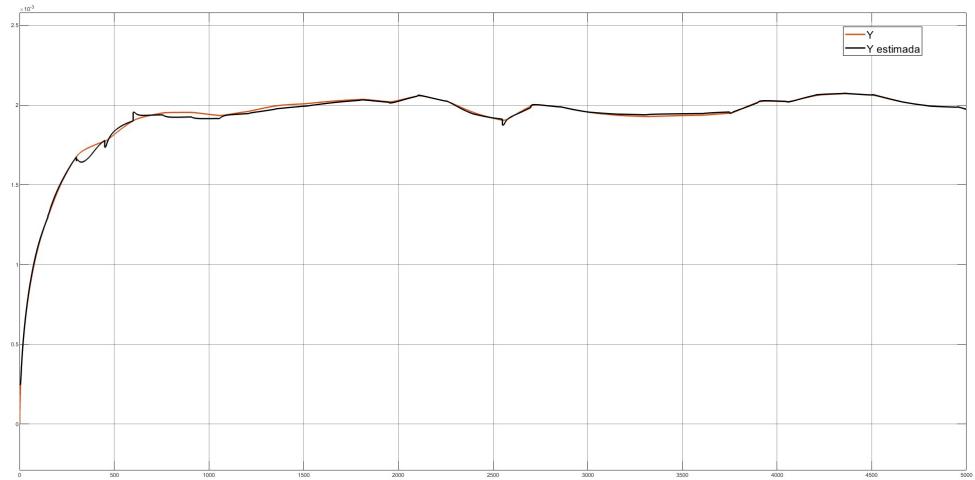


Figura 41: Comparación de salidas - prueba 5

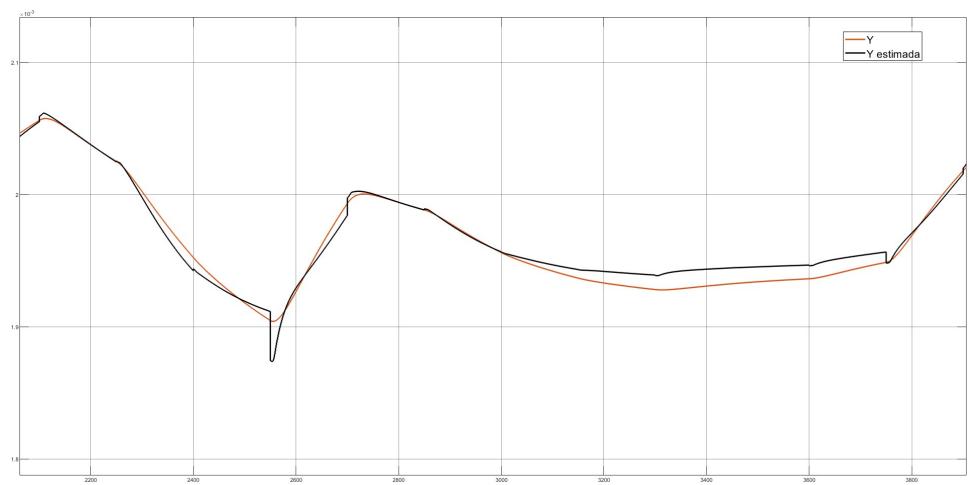


Figura 42: Comparación de salidas - prueba 5