Correcciones

Ignacio Cavicchioli

18 de enero de 2026

1. Preguntas y respuestas

1.1. Pregunta 1

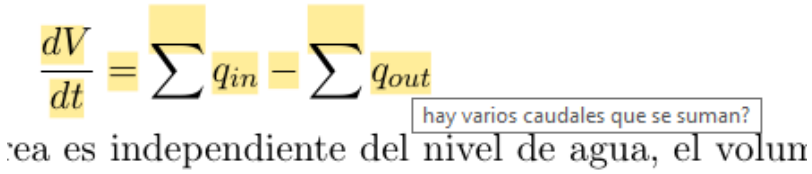


Figura 1

Esta expresión se refiere a que la diferencia entre los caudales entrantes y salientes determina la variación del volumen. En el problema de tanques simplificado que encaré solo hay una entrada y salida de caudal, así que la sumatoria sería sobre 1 solo elemento e innecesaria. Confunde porque quise hacer toda la demostración, en control nos daban las ecuaciones no lineales de la dinámica ya obtenidas sin todo el planteo, pero me pareció demasiado "galerazo". Adjunto una foto de las consignas de un ejercicio de control a modo de ejemplo.

PROBLEMA 4. TANQUES DE AGUA ACOPLADOS

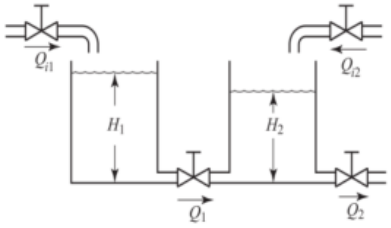


Figura 4.1: Tanques Acoplados.

Obtener un modelo en espacio de estados del siguiente sistema de control de nivel de los tanques representado esquemáticamente en la Fig. 4.1.
Tips de modelado:

- Ecuaciones no lineales de la dinámica:

$$Q_1 = \alpha_1\sqrt{H_1 - H_2} \qquad\qquad Q_2 = \alpha_2\sqrt{H_2}$$

- La variación en el volumen de líquido en cada tanque, es igual al caudal de entrada menos el caudal de salida. Para más información, buscar el desarrollo del libro de Dorf y Bishop.
- Además, por la geometría de los tanques, el volumen de líquido es proporcional al nivel, es decir:

$$V_1 = \gamma_1 H_1 \qquad\qquad V_2 = \gamma_2 H_2$$

- **Linealización en espacio de estados:** Verificar que den lo mismo con el toolbox de Symbolic de MATLAB.
- Asuma que todas las constantes son unitarias ($\alpha_1, \alpha_2, \gamma_1, \gamma_2$, iguales a uno).
- Calcular los valores de equilibrio para H_1, H_2 y Q_{i1}, tales que el caudal de salida es de 1 $\frac{m^3}{seg.}$ suponiendo que $Q_{i2} = 0$.
- Conectarle al sistema en Simulink un control proporcional/integral que realimenta el nivel del segundo tanque y tiene como señal de referencia un caudal un 10% mayor que el caudal de salida de equilibrio del segundo tanque. Simular los resultados.
- Agregar a la simulación un escalón de perturbación a la señal Q_{i2} que sea de un valor del 20% del valor de Q_{i1} en el equilibrio.

Figura 2

1.2. Pregunta 2

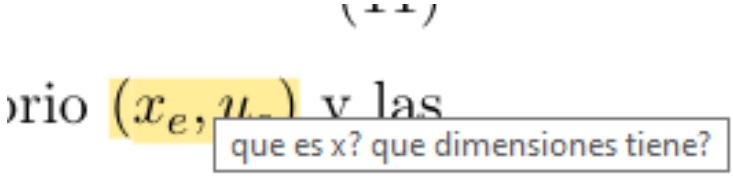


Figura 3

En esta parte me faltó aclarar que h_1 y h_2 son las variables de estado del sistema, y x_e es el vector que representa el punto de equilibrio de dichas variables. La notación es la misma que usamos en control, pero x_e es un vector de 2 dimensiones.

1.3. Pregunta 3

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*.

media y varianza a lo largo del tiempo? como se aplica eso en un escenario online?

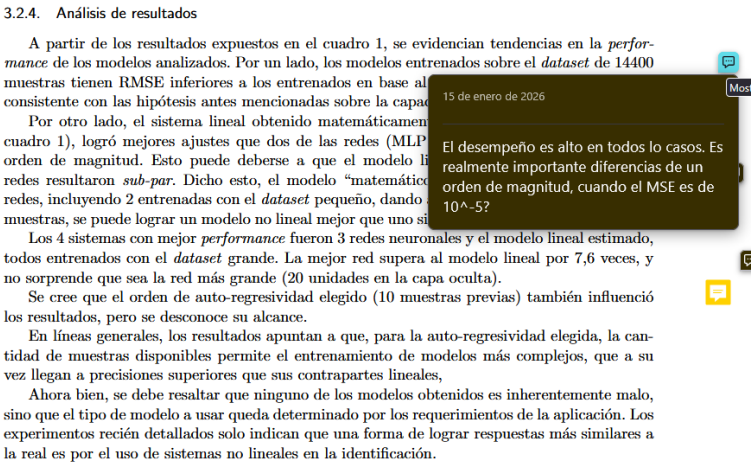
Figura 4

No me pareció que la normalización en media/varianza fuera a ir en contra del entrenamiento. En todo caso, pensé que ayudaría con el espaciado de las muestras. Aparte es lineal, lo que mantiene las relaciones de mayor y menor incluso en el tiempo - si dos instantes de tiempo cumplían que uno era mayor que el otro, lo siguen cumpliendo luego de la transformación. En otras palabras, ayudarían en la excursión de las señales y mantiene las relaciones que importan.

Si esto se quisiera usar en tiempo real, se me ocurre que se puede ir calculando la media y varianza con un filtro tipo promedio móvil con factor de olvido, que lo usamos en el taller de control para estimar un valor de un acelerometro. Sino se puede tener un stack en el que se van agregando las muestras del proceso y descartando las más viejas y se estima sobre las muestras de ese array. La segunda forma es peor porque usa mucha más memoria y es una operación matricial grande y lenta (para la varianza). La primer forma sirve para procesos no tan estacionarios (de media variante).

Con las estimaciones en tiempo real se pueden hacer las normalizaciones y des-normalizaciones.

1.4. Pregunta 4



3.2.4. Análisis de resultados

A partir de los resultados expuestos en el cuadro 1, se evidencian tendencias en la *performance* de los modelos analizados. Por un lado, los modelos entrenados sobre el *dataset* de 14400 muestras tienen RMSE inferiores a los entrenados en base al
consistente con las hipótesis antes mencionadas sobre la capa

Por otro lado, el sistema lineal obtenido matemáticamen
cuadro 1), logró mejores ajustes que dos de las redes (MLP
orden de magnitud. Esto puede deberse a que el modelo li
redes resultaron *sub-par*. Dicho esto, el modelo "matemátic
redes, incluyendo 2 entrenadas con el *dataset* pequeño, dando
muestras, se puede lograr un modelo no lineal mejor que uno si

Los 4 sistemas con mejor *performance* fueron 3 redes neuronales y el modelo lineal estimado, todos entrenados 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.

Figura 5

Si, no puedo refutar que 10^{-5} es bueno. Hice la suposición de que esa diferencia pequeña de 1/1000000 es el error de las no linealidades. Tal vez no elegí correctamente la planta, debería haber sido menos lineal.

1.5. Pregunta 5

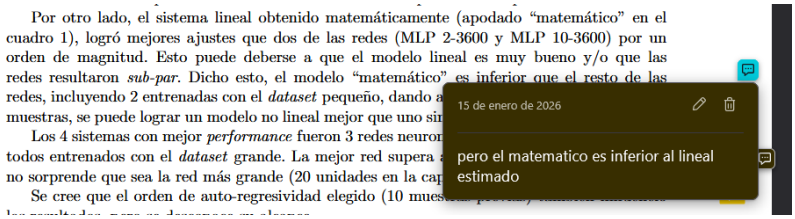


Figura 6

Acá falló mí proof-reading. En esta sección comparaba el modelo matemático lineal con las redes, y luego explico que el modelo estimado (no el derivado de forma matemática) logró un desempeño cercano pero inferior al de las mejores redes. Me faltó decir que el modelo lineal estimado superó al derivado matemáticamente.

1.6. Pregunta 6

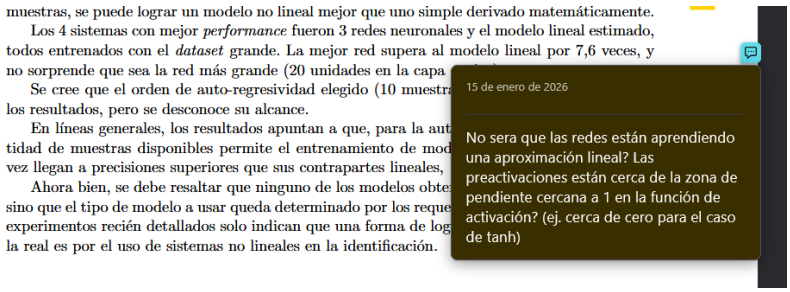


Figura 7

Entiendo que se refiere a que las redes, en vez de aprender las no linealidades propiamente dichas, están aprendiendo los parámetros de tal forma que las activaciones operan en la región más lineal de la tangente. Para todas las fotos (tomadas de MATLAB): a la izquierda están las pre-activaciones y a la derecha, las activaciones propiamente dichas.

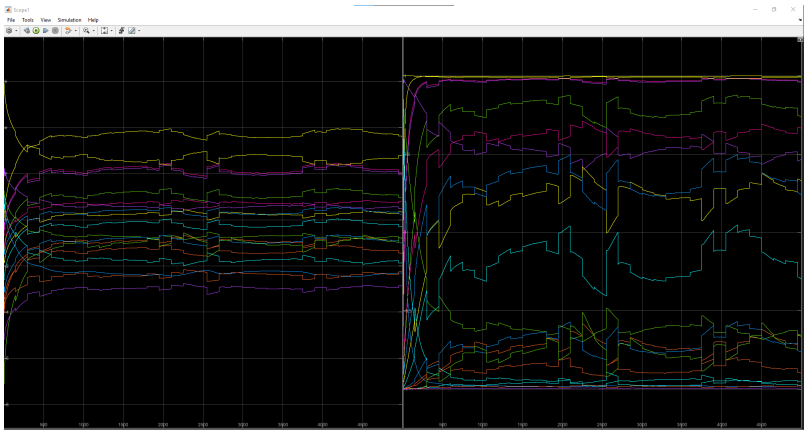


Figura 8: MLP 20 14400

En el MLP 20 14400 se observa que hay neuronas en regiones de activación lineal, no lineal y de saturación. No parecería que haya aprendido una estimación lineal.

En el MLP 2 3600 las pre-activaciones están en 1 y 0,5, siendo la de 0,5 relativamente lineal.

En el MLP 2 14400 las pre-activaciones son simétricas respecto del cero y, como $tanh(0{,}6) = 0{,}537$, es más lineal.

En el MLP 10 3600 las pre-activaciones están distribuidas más como el MLP 20 14400.

En el MLP 5 14400 las pre-activaciones están distribuidas de forma variada, con algunas más lineales que otras.

No se colocan el resto de las imágenes, pero se nota que los modelos de 2 neuronas son los más lineales, mientras que el resto muestra mayor "variedad" en la pre-activación: algunas neuronas capturaran dinámicas más lineales, otras no-lineales y algunas tienen salidas casi constantes/saturadas.

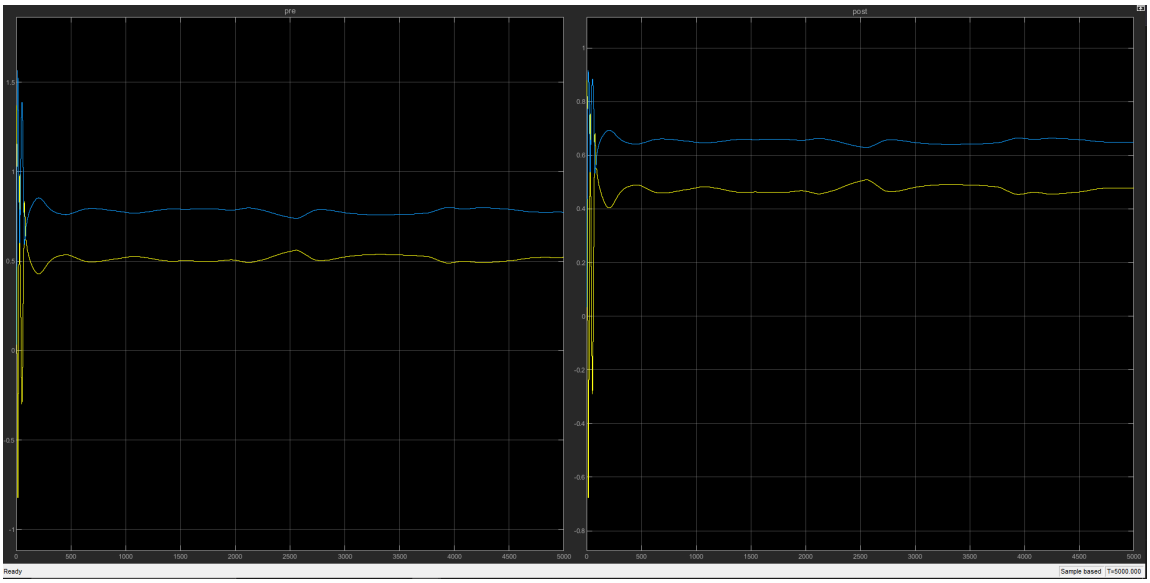


Figura 9: MLP 2 3600

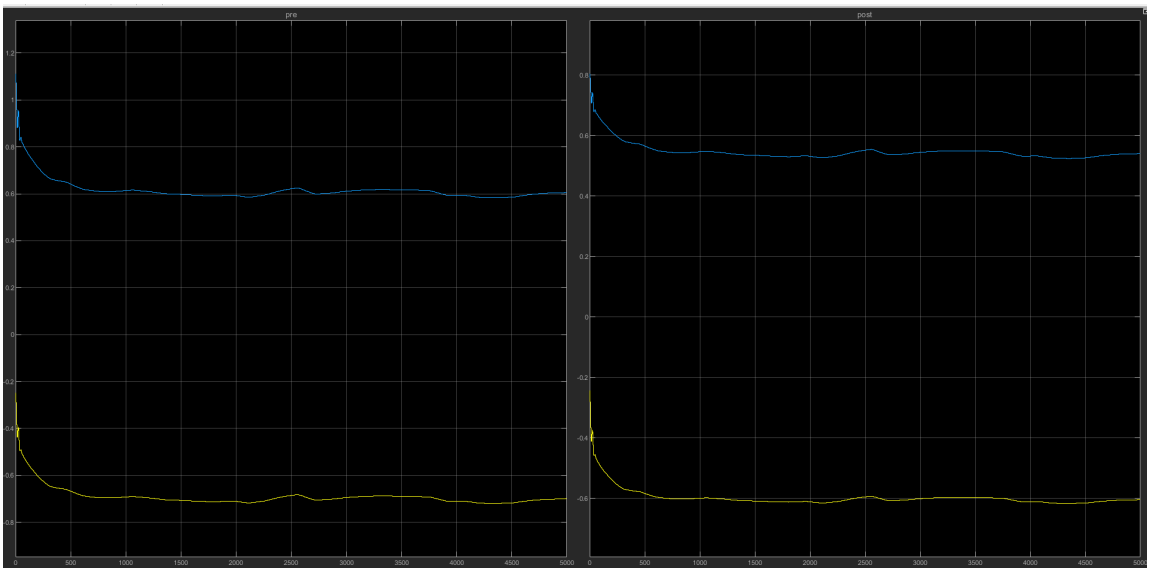


Figura 10: MLP 2 14400

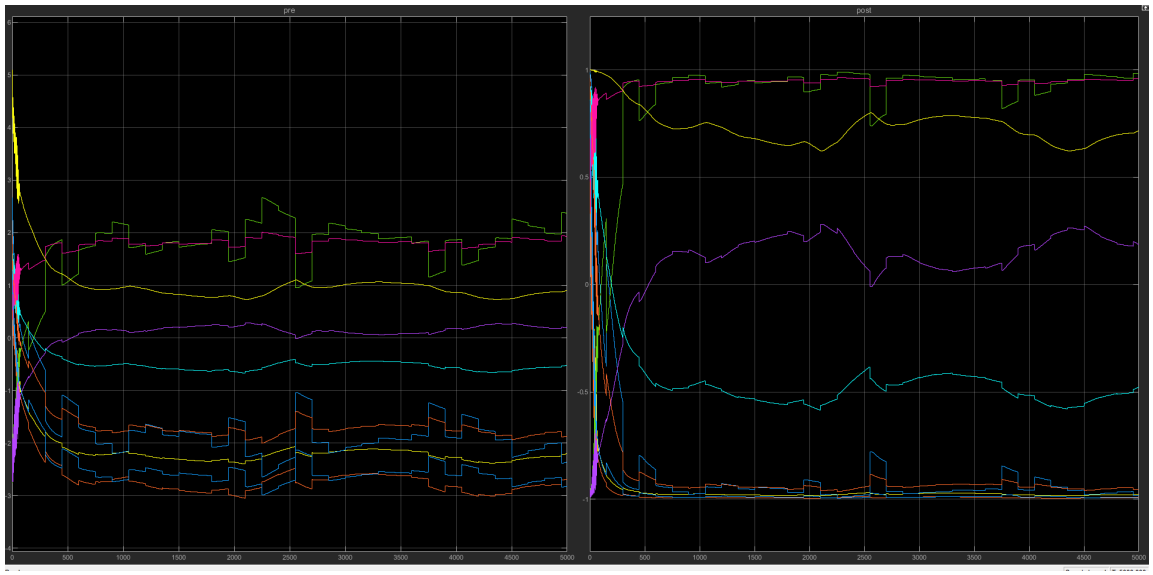


Figura 11: MLP 10 3600

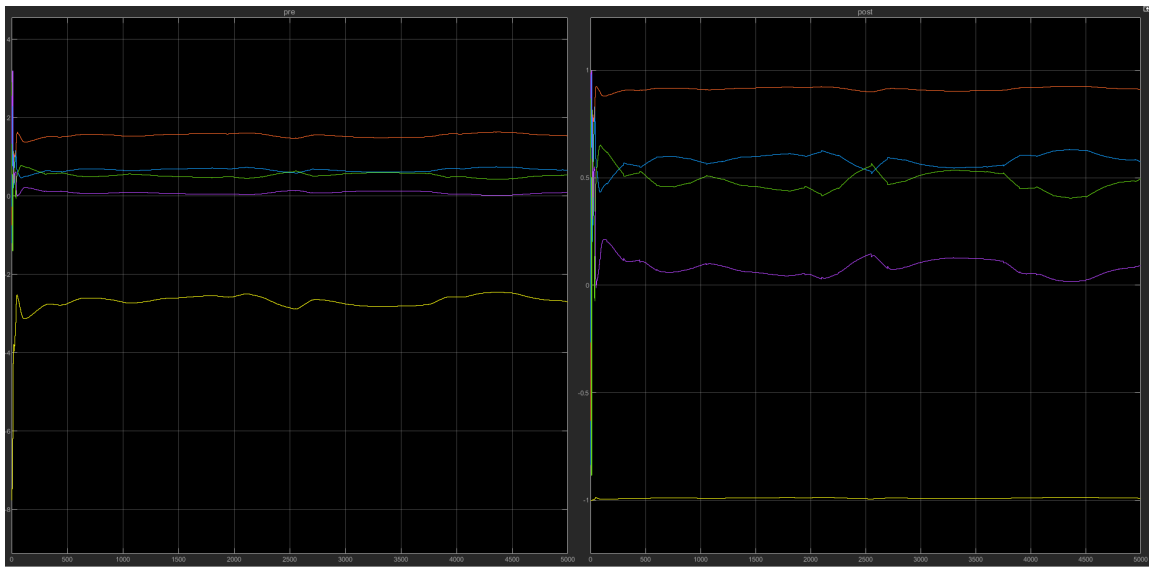


Figura 12: MLP 5 14400

1.7. Pregunta 7

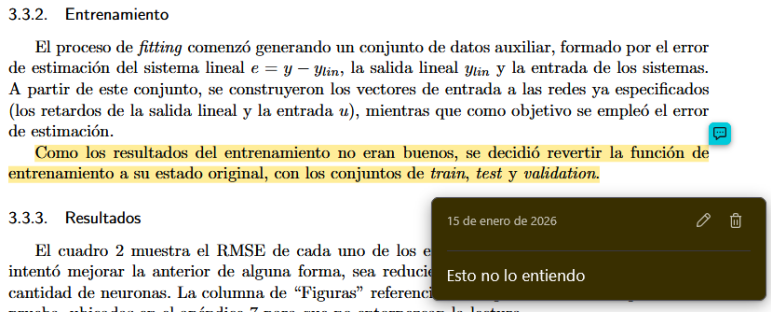


Figura 13

El primer experimento tenía una función para entrenar llamada "entrenar_red", que se tuvo que cambiar porque no daba buenos resultados para los datos de la segunda experiencia.Se ajustó en el momento hasta que funcionó (es decir, entrenó como corresponde).

```
function net = entrenar_red(Xn, Tn, numNeuronas, epochs)
% ENTRENAR_RED Entrena una red neuronal feedforward (fitnet)
% sin early stopping y con división determinista
%
% Uso:
%   net = entrenar_red(Xn, Tn, numNeuronas)

% ---------------------------------------------------------
% Crear red
% ---------------------------------------------------------
net = fitnet(numNeuronas);

% Funciones de transferencia
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';

% ---------------------------------------------------------
% División determinista SIN validación
% ---------------------------------------------------------
N = size(Xn, 1);   % muestras

nTrain = floor(0.85 * N);

idxTrain = 1 : nTrain;
idxTest  = nTrain + 1 : N;

net.divideFcn = 'divideind';
```

```
net.divideParam.trainInd = idxTrain;
net.divideParam.valInd   = [];      % <-- elimina early stopping
net.divideParam.testInd  = idxTest;

net.trainParam.epochs = epochs;
net.trainParam.min_grad = 1e-9;
net.trainParam.mu_max = 1e12;
net.trainParam.time = inf;


% ---------------------------------------------------------------
% Desactivar preprocesamiento interno
% ---------------------------------------------------------------
net.inputs{1}.processFcns  = {};
net.outputs{2}.processFcns = {};

% ---------------------------------------------------------------
% Entrenamiento
% ---------------------------------------------------------------
net = train(net, Xn.', Tn.');
end
```

```
function net = entrenar_redV2(Xn, Tn, numNeuronas, epochs)
% entrenar_redV2 Entrena una red neuronal feedforward (fitnet)
% sin early stopping y con división determinista
%
% Uso:
%   net = entrenar_redV2(Xn, Tn, numNeuronas)

% ---------------------------------------------------------------
% Crear red
% ---------------------------------------------------------------
net = fitnet(numNeuronas);

% Funciones de transferencia
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';

% ---------------------------------------------------------------
% División determinista SIN validación
% ---------------------------------------------------------------
N = size(Xn, 1);   % muestras


net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio   = 0.15;
```

```
net.divideParam.testRatio  = 0.15;


net.trainParam.epochs = epochs;



% -------------------------------------------------------------
% Entrenamiento
% -------------------------------------------------------------
net = train(net, Xn.', Tn.');
end
```

Básicamente, se regresó a una configuración más estándar de "fitnet" (la de MATLAB). En la primer parte de la monografía se usó una división fija del *dataset* sin validación, con ajuste de parámetros a mano.Como esto empezó a dar malos resultados en el otro experimento, se regresó a la división aleatoria con validación y los parámetros por defecto, que sí funcionaron correctamente.

1.8. Pregunta 8

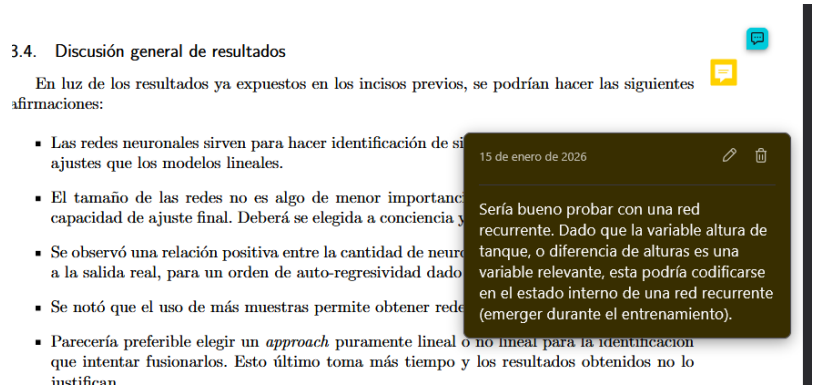


Figura 14

La configuración usada en la monografía responde a un MLP realimentado externamente con cierta memoria implícita en la realimentación (en la entrada tiene muestras previas de la salida). Mí idea fue probar esta idea de los estados internos con una red recurrente armada en MATLAB, pero no se pudo. En cambio, se armó en python (pytorch), resultando en las imágenes de más adelante. La arquitectura se fue ajustando desde 1 capa LSTM de 5 neuronas hasta obtener una red funcional, y terminó de 2 capas LSTM de 32 unidades cada una. Se logró un RMSE de $2{,}222591e-06$, que es bueno, pero lo interesante es observar los estados internos.

Se observó que el estado interno de la unidad 7 (fig. 15) de la red recurrente presenta una forma visualmente correlacionada con la altura del tanque. El tema es que la salida del sistema es proporcional a la altura del segundo tanque, por lo que de cierta manera la red disponía de la información de esa variable interna. No se puede afirmar que la red haya "descubierto" la altura.

Sin embargo, resulta relevante notar que una dinámica fuertemente correlacionada con la altura/salida aparece en el estado oculto de una unidad interna, previo a la capa de salida. Esto sugiere que la red construye una representación interna, y no que solo opera sobre su entrada para generar una salida.

Este resultado indica que, aun cuando la salida es función directa de una de las variables de estado, estas redes recurrentes pueden organizar la información en forma de una representación interna, abriendo la posibilidad de aplicación a sistemas donde la salida es una combinación más compleja de variables. Otra posibilidad es usar esta representación para forzar una reducción en dimensión - tal vez con una elección de arquitectura consciente, se podría llegar a tener una capa LSTM mínima en la que exista una representación del problema en menor dimensión. Es como los autoencoders en el sentido conceptual de que se hace una reducción dimensional, no arquitectónico.

Al correo se adjuntan 2 imágenes demasiado grandes para una hoja de documento en la que están todos los estados ocultos de todas las neuronas de las 2 capas.

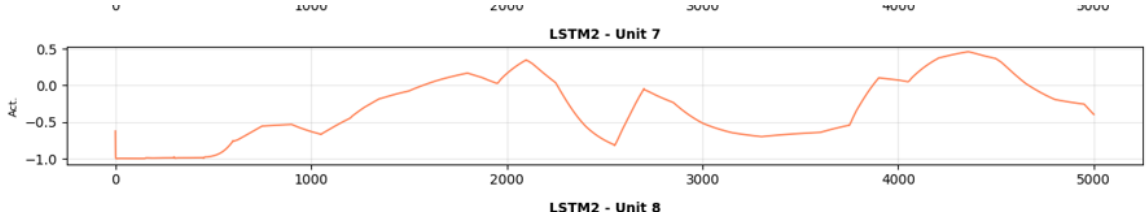


Figura 15: Estado oculto de la unidad 7 de la segunda capa

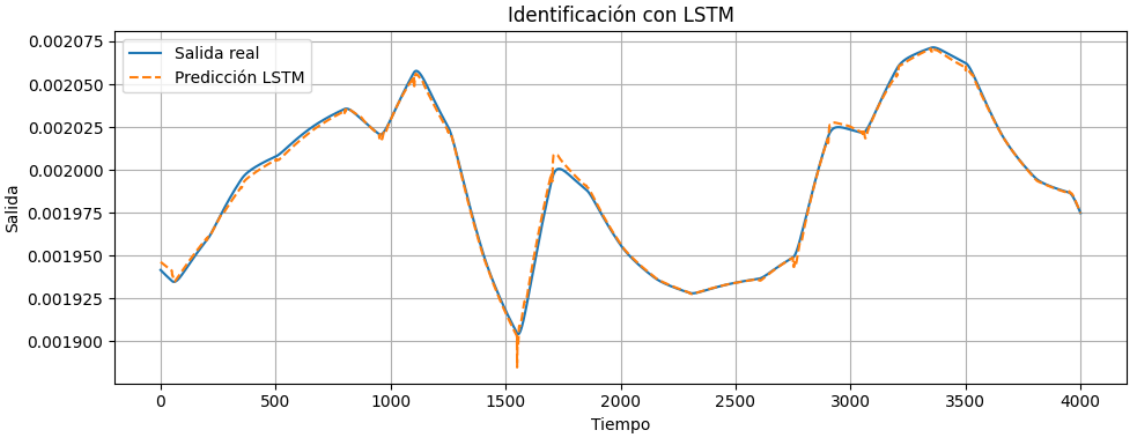


Figura 16: Salida real y predicción con LSTM RNN

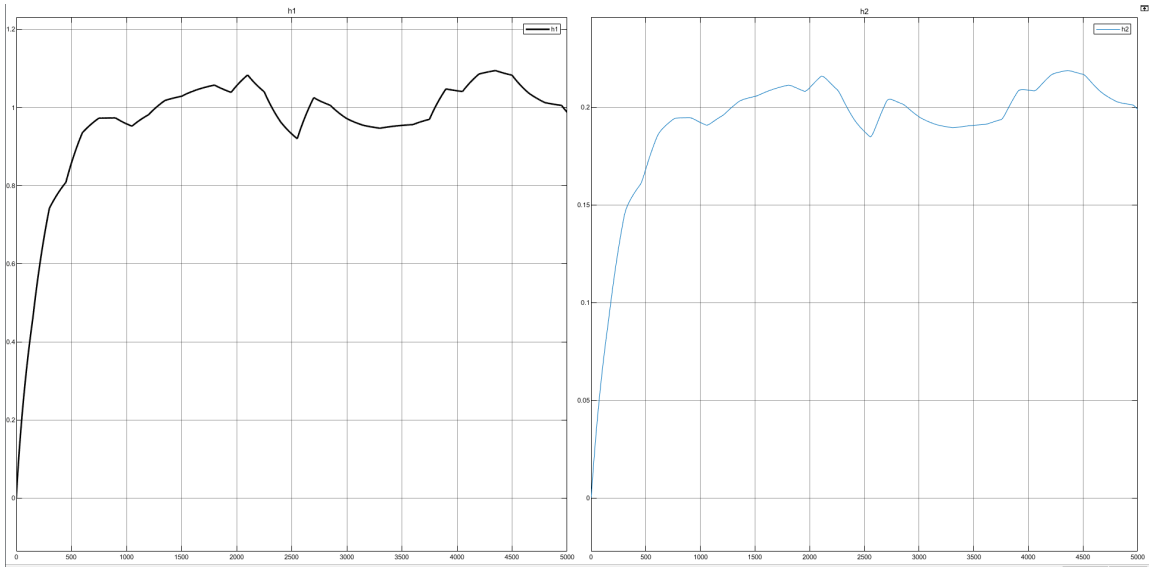
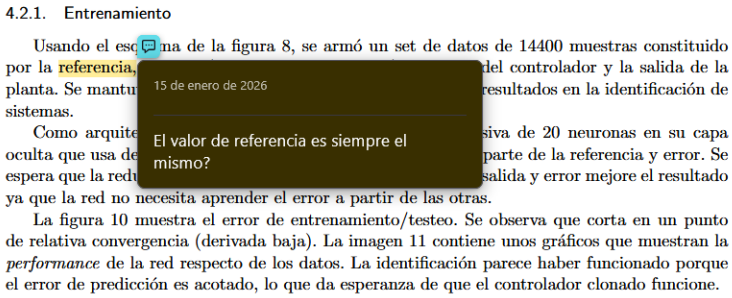


Figura 17: Alturas reales

1.9. Pregunta 9



4.2.1. Entrenamiento

Usando el esquema de la figura 8, se armó un set de datos de 14400 muestras constituido por la referencia, del controlador y la salida de la planta. Se mantuvo resultados en la identificación de sistemas.

15 de enero de 2026

El valor de referencia es siempre el mismo?

Como arquitectura siva de 20 neuronas en su capa oculta que usa de parte de la referencia y error. Se espera que la red 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 18

No, la referencia cambia. Adjunto el gráfico de la serie temporal de la referencia del *dataset* usado. La razón por la que luego se quita del *dataset* es porque queda codificada en el error. Esto último fue una decisión que, si vemos un esquema típico de control, es razonable ya que el controlador no recibe la referencia, sino que el error, y lo busca minimizar. Sin embargo, una red neuronal no es un controlador, y no tendría sentido obligarla a usar las mismas entradas, por lo que siento que no habría estado mal dejarle la referencia incluso si no cambiaba con las perturbaciones en el lazo.

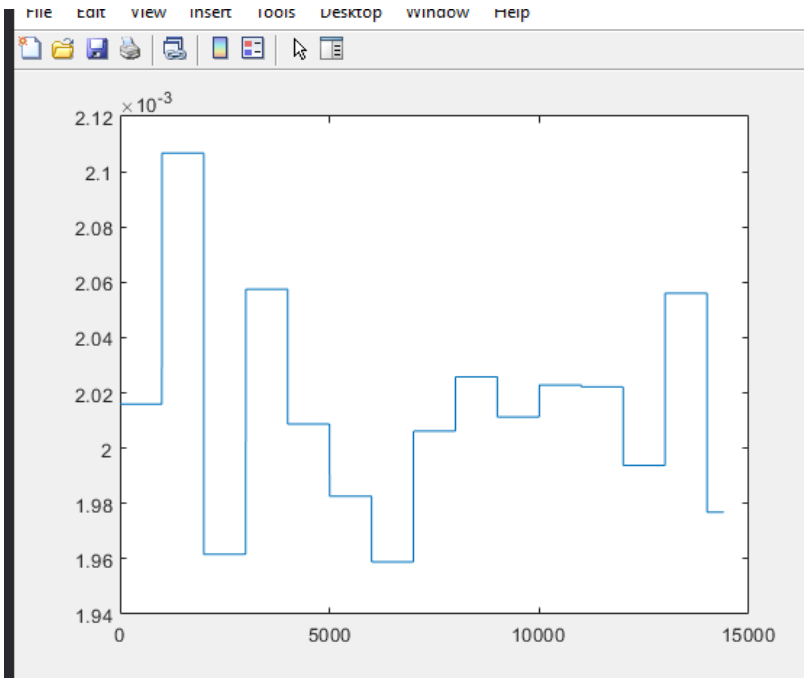
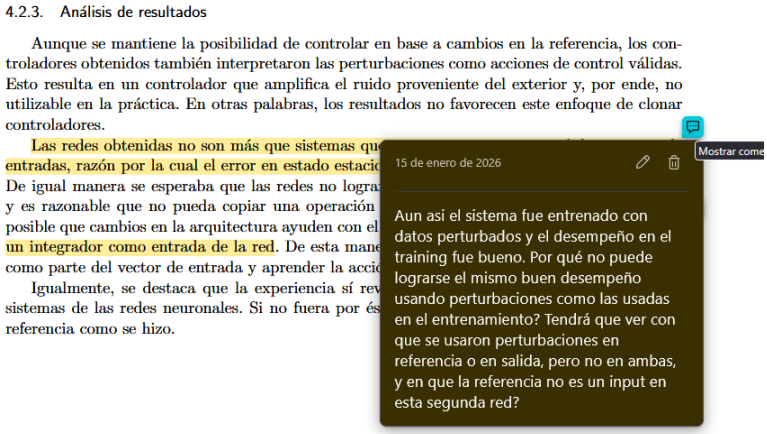


Figura 19

1.10. Pregunta 10


4.2.3. Análisis de resultados

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. En otras palabras, los resultados no favorecen este enfoque de clonar controladores.

Las redes obtenidas no son más que sistemas que
entradas, razón por la cual el error en estado estacio
De igual manera se esperaba que las redes no logra
y es razonable que no pueda copiar una operación
posible que cambios en la arquitectura ayuden con el
un integrador como entrada de la red. De esta mane
como parte del vector de entrada y aprender la acci

Igualmente, se destaca que la experiencia sí rev
sistemas de las redes neuronales. Si no fuera por és
referencia como se hizo.

Mostrar come

15 de enero de 2026

Aun asi el sistema fue entrenado con datos perturbados y el desempeño en el training fue bueno. Por qué no puede lograrse el mismo buen desempeño usando perturbaciones como las usadas en el entrenamiento? Tendrá que ver con que se usaron perturbaciones en referencia o en salida, pero no en ambas, y en que la referencia no es un input en esta segunda red?

Figura 20

Existe la posibilidad de que, disponiendo de la referencia, la segunda red (entrenada con el

dataset con ambos tipos de perturbaciones) aprenda a rechazar las perturbaciones a la salida del controlador. Sin embargo, dados los comentarios en la monografía y el mismo desarrollo hecho en este documento, diría que es más útil un cambio de red que intentar hacer que un MLP emule un efecto de acción integral en un controlador. De cierta manera, esto delimita el campo de aplicación natural del MLP y da espacio a otros tipos de redes más adaptadas a este tipo de problemas.

1.11. Pregunta 11

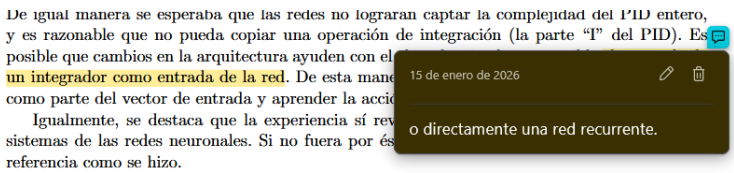


Figura 21

Si, notando los beneficios vistos antes de poder capturar variables de estado en la misma red en los estados ocultos, no tiene sentido usar MPLs en configuraciones "extrañas".