Grado en Ingeniería Electrónica Industrial y Automática

Curso 2019/2020

CONTROL INTELIGENTE

Reporte 3

Control de un Motor Diesel con una Red Neuronal

Emanuel David Nuñez Sardinha

13 de Enero de 2020

Introduccion

El objetivo de este proyecto es el diseño de un controlador basado en redes neuronales para un motor diesel simulado. Nos concentramos en el aspecto práctico, el proceso de diseño del controlador, y corrección de errores y optimización final.

Se realiza una presentación del problema, donde se explica el modelo, consideraciones generales, y modificaciones hechas a la planta anterior al proceso de optimización.

En el estudio preliminar, se discuten problemas relevantes a este reporte, particularmente referentes a consistencia de señal y tiempos de ejecución. Por consideraciones de espacio, problemas que ya hayan sido discutidos en reportes previos han sido omitidos.

En optimización discutimos el proceso de desarrollo:la generación correcta de datos de referencia, el proceso de entrenamiento de la red, y un análisis breve de el efecto de los parámetros en el resultado final.

Presentación del problema

Igualmente que en los proyectos anteriores, se ha escogido el modelo de un motor diesel incluido en los ejemplos de Matlab. El modelo corresponde a un sistema cerrado no lineal complejo, que computa la velocidad de un motor frente a una señal respecto al tiempo (Figura 1). Podemos acceder a este desde la pestaña de comandos tecleando *SIdemo_enginewc*.

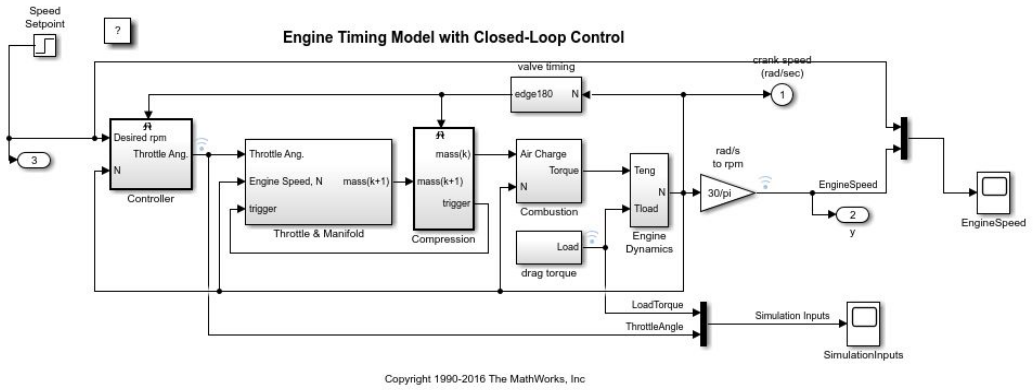


Figura 1: Modelo original, sin cambios

El controlador incluido por defecto es un controlador discreto PI(z), que toma como entrada las revoluciones por minuto deseadas, calcula la diferencia respecto a la velocidad actual y genera la señal de control adecuada. El controlador contiene un bloque de conversión de unidades internamente (rad/s a rpm) y un bloque *Timing,* que sincroniza el número de ejecuciones del motor con las rotaciones, causando que a mayor velocidad de giro se actualice más rápido el controlador.

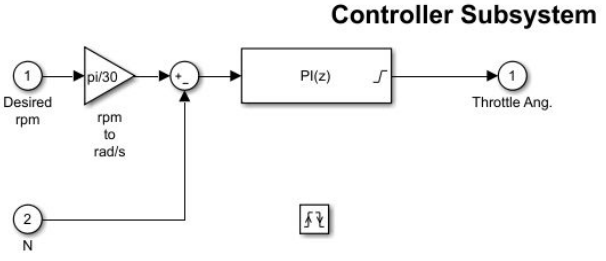


Figura 2: Bloque controlador

Para implementar un controlador NARMA-L2, es necesario aislar la planta a un bloque independiente (Figura 3). Se ha removido el bloque correspondiente del controlador y reemplazado por una entrada directamente al ángulo de apertura de combustible (0 a 90 grados). No es necesario el cálculo de diferencia de velocidad, ya que el controlador recibe la salida final de la planta. La salida se ha simplificado a solo el número de revoluciones resultantes. El bloque donde previamente estaba el controlador solo cuenta con una entrada

e internamente solo conecta entrada con salida, pero se ha mantenido para asegurar mantener el bloque *Trigger*, que controla el ritmo de ejecución.

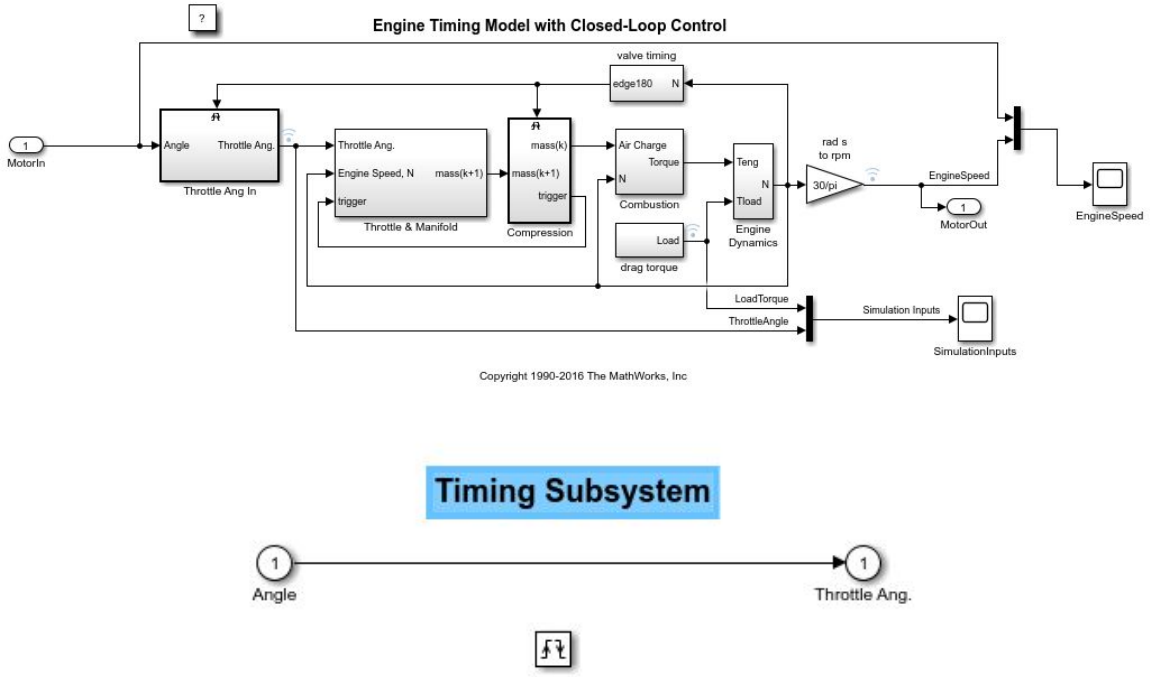


Figura 3: Modelo de planta aislada e interior del controlador vacío.

Se ha implementado el esquema estándar de control de bloques NARMA-L2. La red alimenta directamente a la planta en un sub-bloque, y se realimenta con la salida. Se ha añadido un bloque extra de entrada para mayor control en la señal inicial.

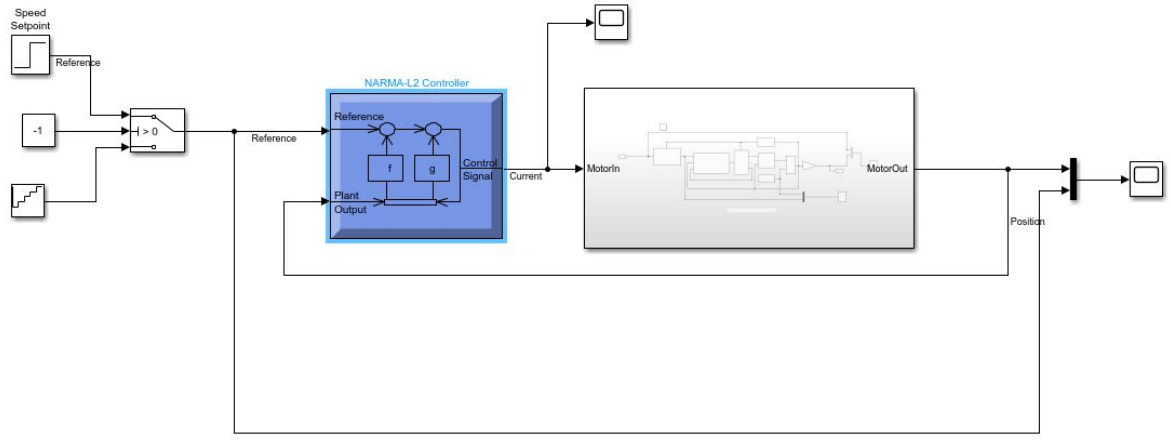


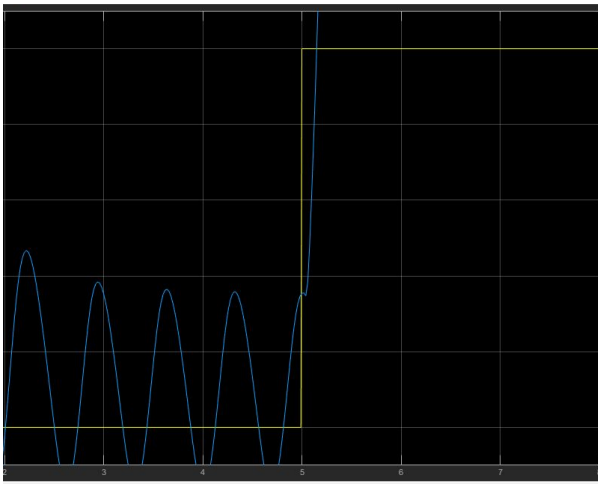
Figura 4: Sistema final

Estudio Preliminar

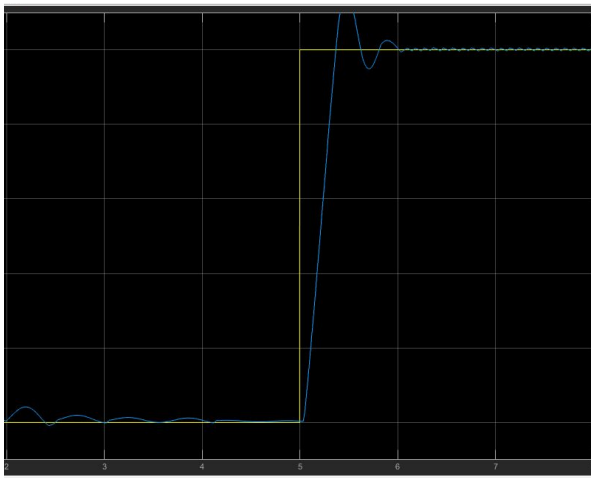
Resolución de ejecución

Resaltamos una característica notable del sistema estudiado, su gran dependencia del intervalo de tiempo de muestreo. Utilizar la opción “Auto” en el intervalo de tiempo no garantiza una respuesta consistente, esencial para obtener datos de entrenamiento. En la Figura 5 se muestra la respuesta del sistema con el bloque PI predeterminado, variando únicamente la resolución temporal. Notamos que para valores pequeños de resolución de tiempo, la respuesta es alterada muy drásticamente, sin embargo, para t = 0.001s tenemos un buen compromiso entre calidad de respuesta y tiempo de ejecución.

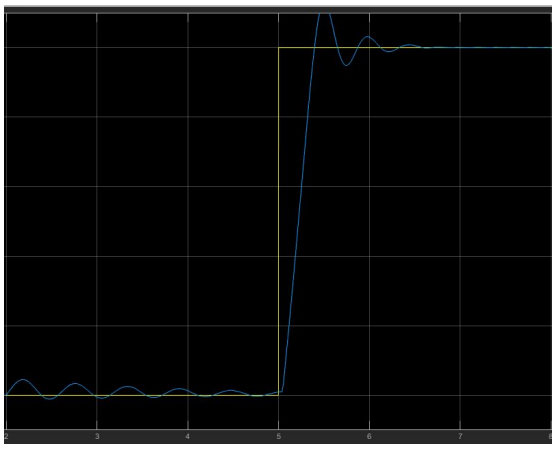
Esta consideración parece pequeña inicialmente, pero dado que luego tenemos que generar cantidades masivas de datos, nos permite ejecutar el sistema *10 veces más rápido* con resultados casi idénticos.



t = 0.01s	t = 0.005s



t = 0.001s	t = 0.0005s



t = 0.0001s (Valor predeterminado en el demo)

Figura 5: Efecto de resolución temporal en respuesta predeterminada

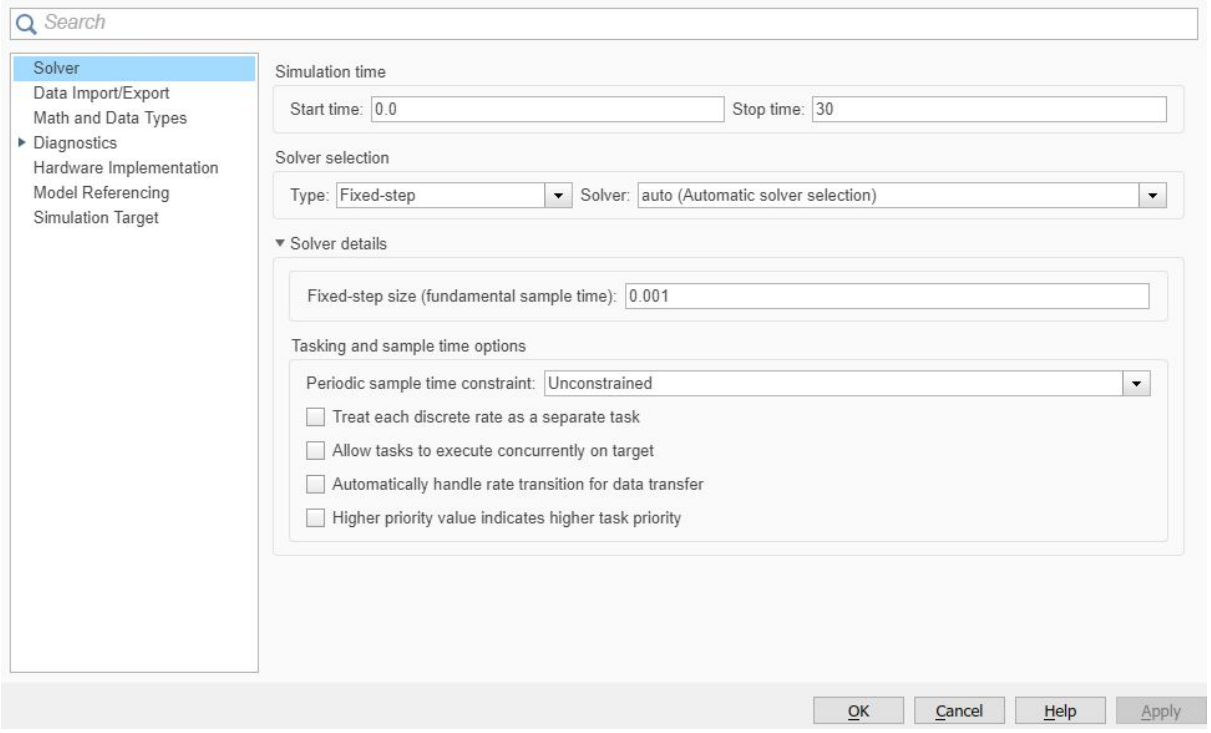


Figura 6: Ubicación de configuración de tiempo de muestreo en Simulink

Distorsiones de señal

Previamente a la toma de datos, queremos minimizar las distorsiones presentes en el sistema. Para obtener resultados similares a la realidad, la planta del motor diesel cuenta con fuentes internas de ruido. Aunque no podemos remover el ruido periódico sin afectar drásticamente el sistema (Bloque "*valve timing*"), podemos modificar el bloque "*Drag torque*", que simula cambios arbitrarios en el torque/carga que ejerce el motor.

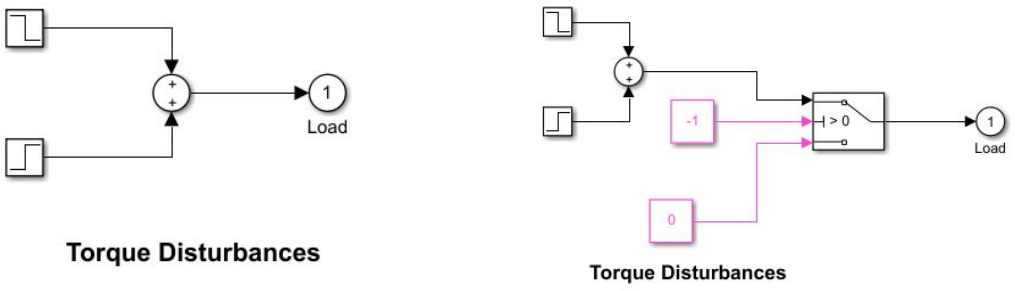
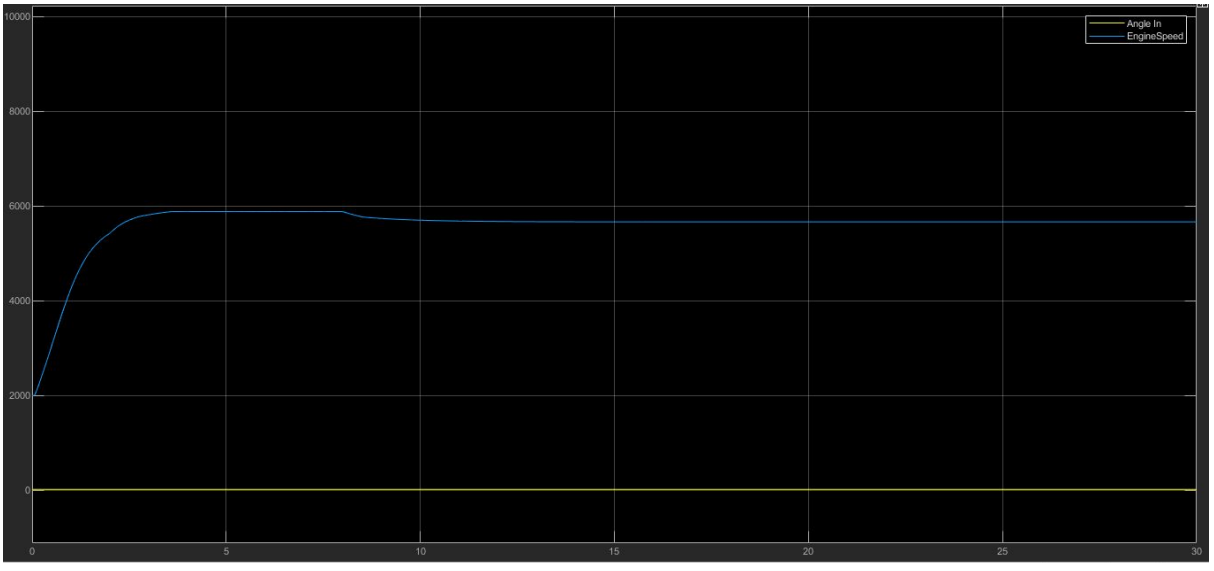


Figura 7: Bloque "Drag Torque" (a) Original (b) Modificado

Se ha realizado una modificación simple para poder activar o desactivar el efecto de esta distorsión. El modelo será entrenado con el sistema sin distorsiones. Podemos ver el efecto en la Figura 8, donde la nueva respuesta permanece constante.



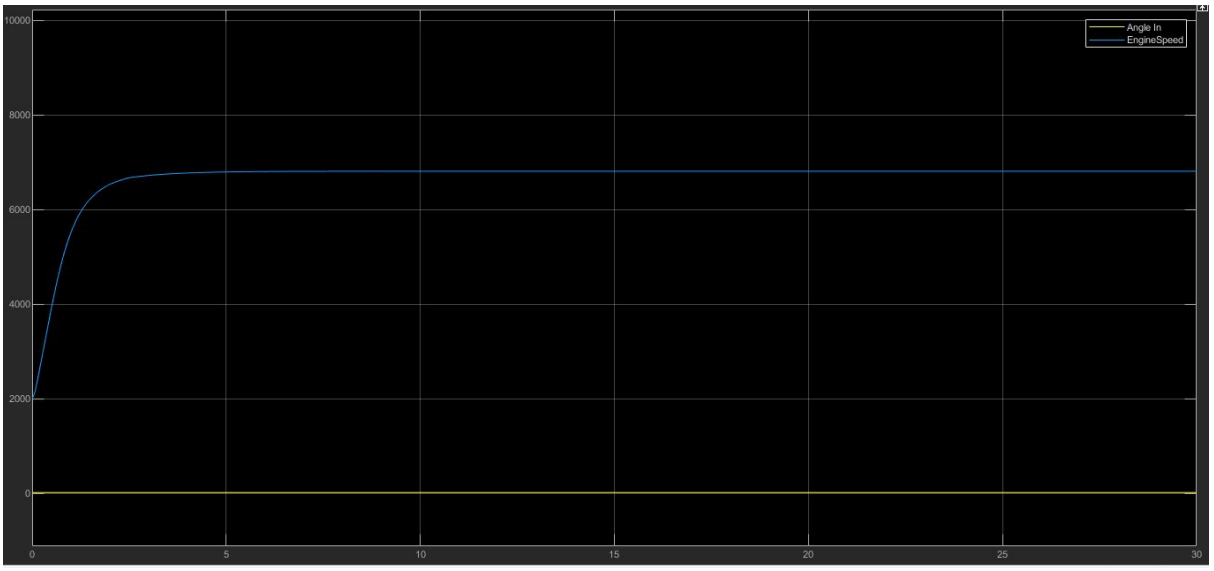


Figura 8: (a) Distorsión por carga activada (b) Distorsión por carga desactivada

Optimización

Dado la naturaleza de el sistema, los resultados y eficiencia de la optimización estará casi exclusivamente determinada por la configuración de nuestro bloque NARMA-L2 (Figura 9).

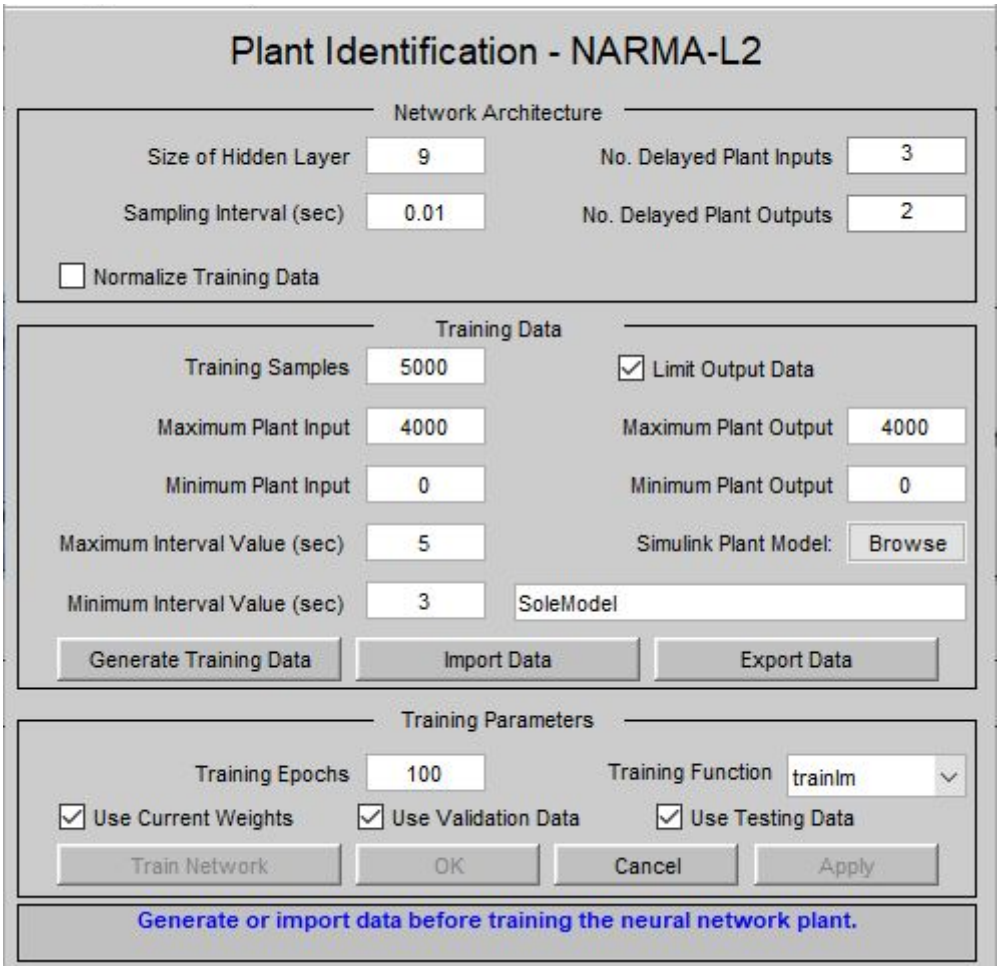
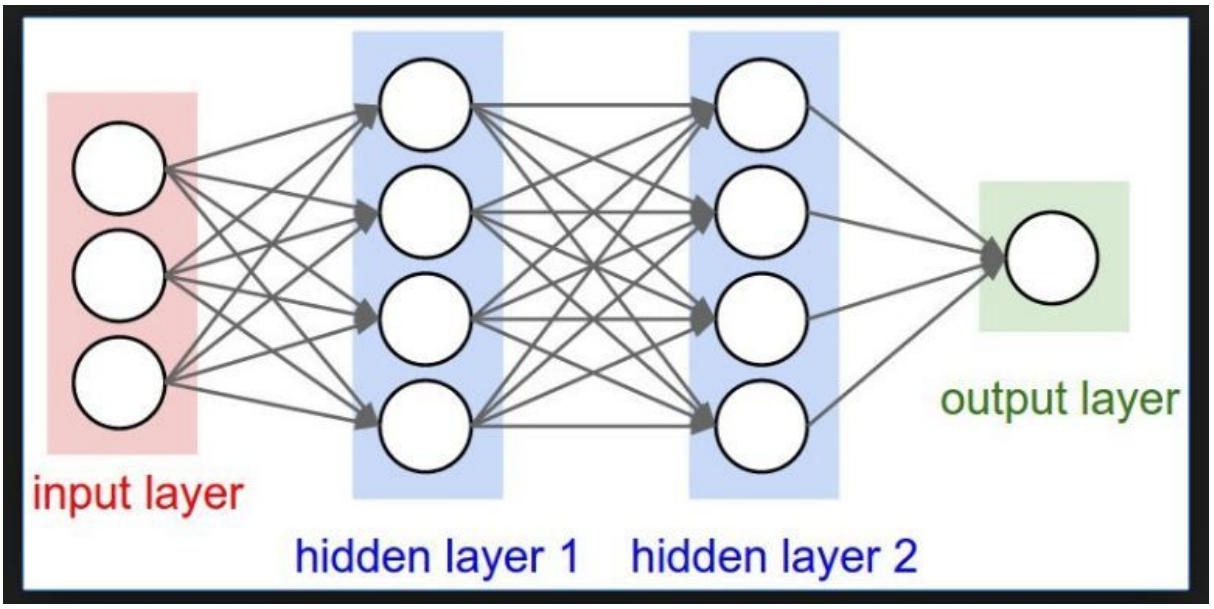


Figura 9: Bloque NARMA-L2

Propiedades

Arquitectura de la Red

- **Size of Hidden Layer:** Tamano de capa oculta. Numero de capas internas (Figura 10.a)
- **Delayed plant inputs & outputs:** Número de neuronas de entrada/salida que no dan una respuesta inmediata, sino guardan/utilizan datos de otro momento temporal.
- **Intervalo de muestreo:** Tiempo entre muestras sucesivas en el sistema. Puede considerarse como análogo a la resolución temporal de las muestras. En la Figura 10.b mostramos el efecto en los datos obtenidos para 1 sec y 3 sec. Escogemos un valor inicial de 0.1 sec.
- **Normalize Training data:** Normalizar datos de entrenamiento.



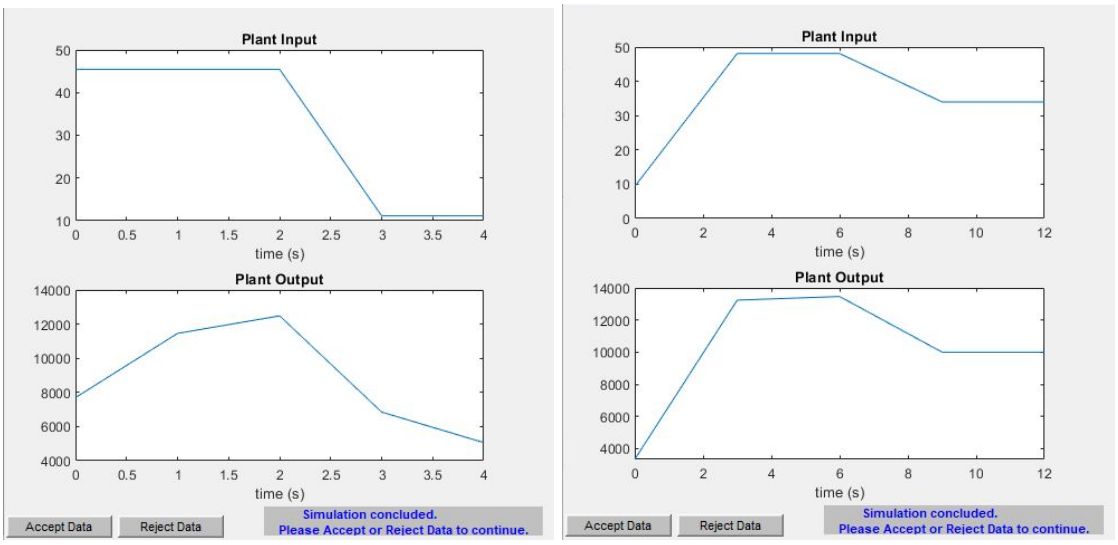


Figura 10: (a) Representación de red neuronal (https://www.mdpi.com/2227-7390/7/10/959)
(b) Datos para intervalos de muestreo 1 y 3

Datos de entrenamiento

- **Training Samples:** Número de muestras tomadas del modelo. La duración temporal final de la muestra está dada por (Sampling Interval) * (Training Samples) en segundos. En general, más muestras resulta en una mejor respuesta.
- **Maximum & Minimum Plant Input:** Entradas maxima y minima que puede recibir la planta. El controlador debe ser capaz de recibir un ángulo entre [0-90], pero el modelo original tiene un controlador PI configurado con una salida saturada entre [0-70] (Figura 11). Inicialmente, utilizamos estos valores como entrada de planta para nuestro modelo.

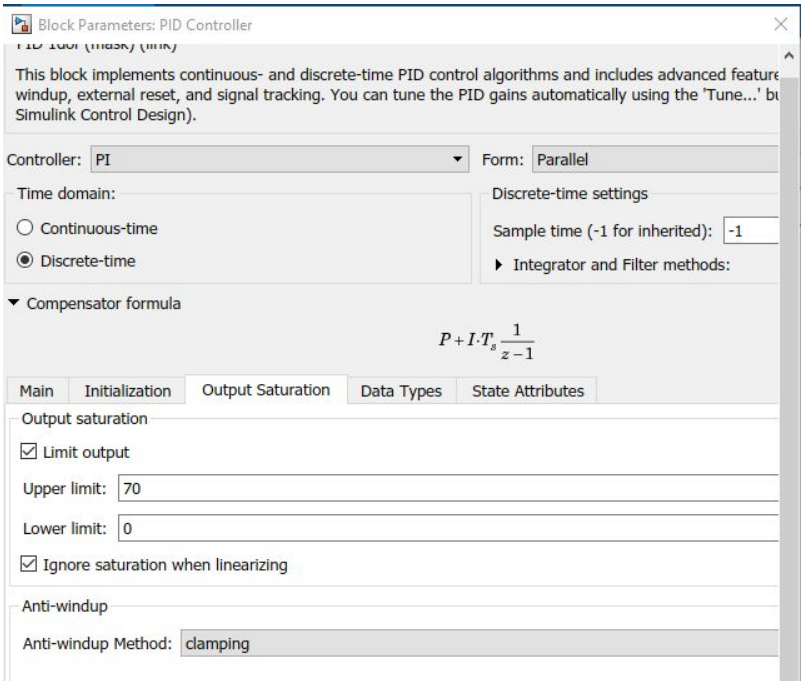


Figura 11: Salida del controlador de la planta original.

- **Maximum & Minimum Interval Value:** Corresponde al tiempo que la señal permanece constante en los datos de prueba. Como se ve en la respuesta predeterminada del sistema con su controlador inicial (12), la señal toma aproximadamente 1 segundo en acercarse, y puede tomar más de 5 en estabilizarse. Asignamos un rango de [3-5] sec.

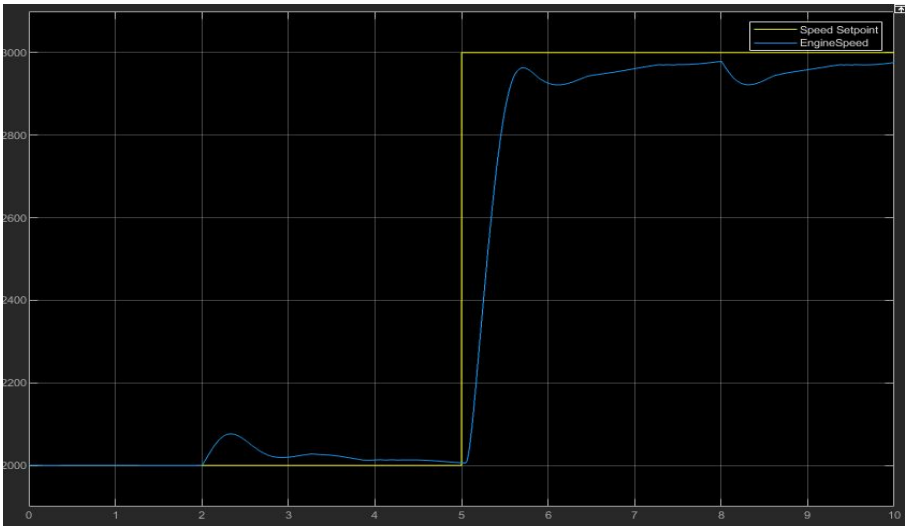


Figura 12. Respuesta con valores predeterminados

- **Maximum & Minimum Plant Output:** Rango de la señal de respuesta de la planta. Nuestra planta en particular opera para valores bastante elevados, con una señal de referencia de [2000-3000] rpm (Figura 13). Iniciamos con un valor conservador de [0-5000] rpm.

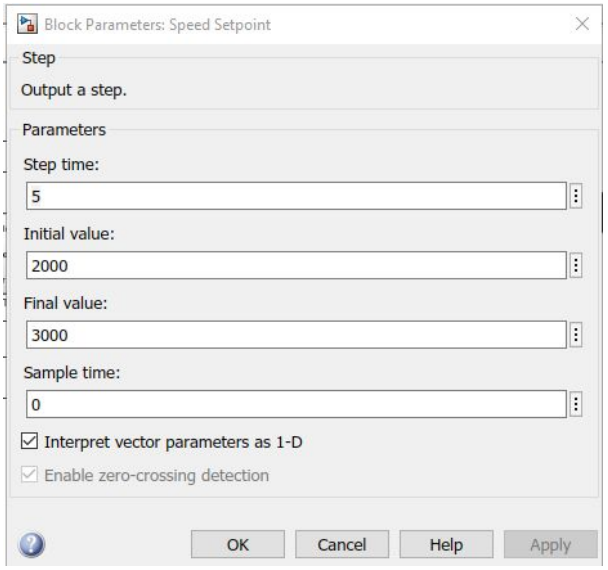


Figura 13. Senal de entrada con valores predeterminados

- **Training Epochs:** Análogo a número de iteraciones para la configuración de la red. Un número mayor resulta en mejor respuesta.

Generación de datos de entrenamiento

Con las propiedades definidas previamente, generamos una muestra inicial de datos. Mostramos adicionalmente una red entrenada con estos datos (Figura 14).

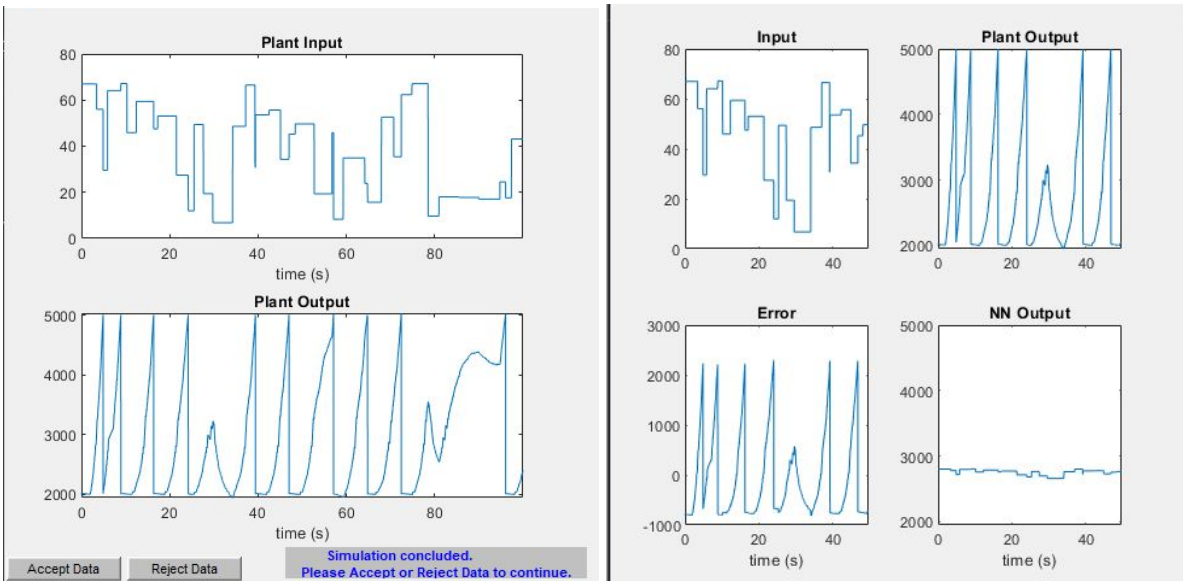


Figura 14: (a) Primera muestra de datos.
(b) Comparación de respuesta real con respuesta de la Red después de entrenar para estos datos.

Notamos un comportamiento errático de la planta durante el entrenamiento y por consecuencia, una respuesta incorrecta de la red entrenada. Para generar el modelo correctamente, necesitamos una muestra de datos representativa del sistema. En base a la muestra previa, notamos unas restricciones adicionales:

- El controlador PI que utilizamos como referencia envía una respuesta entre 0-70 grados, para mantener la planta entre 2000-3000 rpms. Sin embargo, como se muestra en la Figura 15, mantener un ángulo inicial de 15 grados por unos segundos resulta en una respuesta de más de 10000 rpms. Removemos las restricciones de salida superior de la planta (Max plant output = +inf).

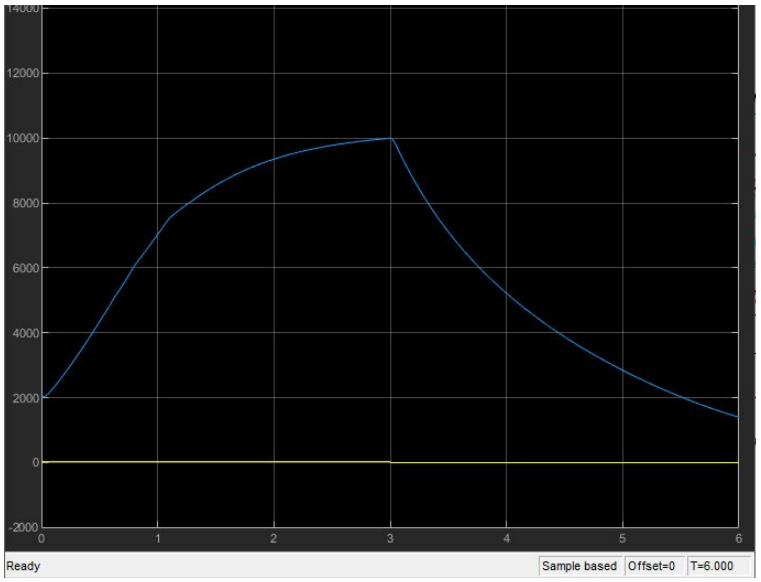


Figura 15: Respuesta a escalón de 15 a 1 grados.

- Partiendo de esto, la planta es muy sensible al ángulo de entrada. Esto afecta directamente los datos generados para la red, ya que de potencialmente el rango completo [0-90] grados, una colección pequeña de valores contendrá la información necesaria para enseñar al sistema como comportarse en valores pequeños. Esto puede solucionarse parcialmente reduciendo nuestro rango de entrada, o aumentando el número de muestras. (Max plant input = 40).
- Mantener un ángulo de entrada cercano a 0 durante el tiempo suficiente causa que la planta simule un “atasco” del motor, parando su funcionamiento. Para evitar esto, cambiamos el valor mínimo de la entrada a 1 grado. (Min plant input = 1).

Con esto en cuenta, generamos un nuevo conjunto de datos de prueba (Figura 16). Notamos que los datos son representativos del comportamiento de la planta. Exportamos los datos como *“dot05time10000samples.mat”.*

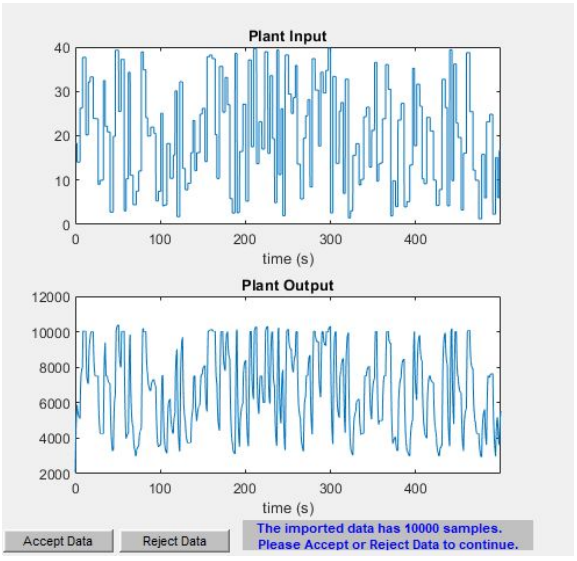


Figura 16: Muestras utilizadas

Entrenamiento del modelo

Con los datos generados en el apartado anterior entrenamos el modelo obtenemos la respuesta mostrada en la Figura 17.

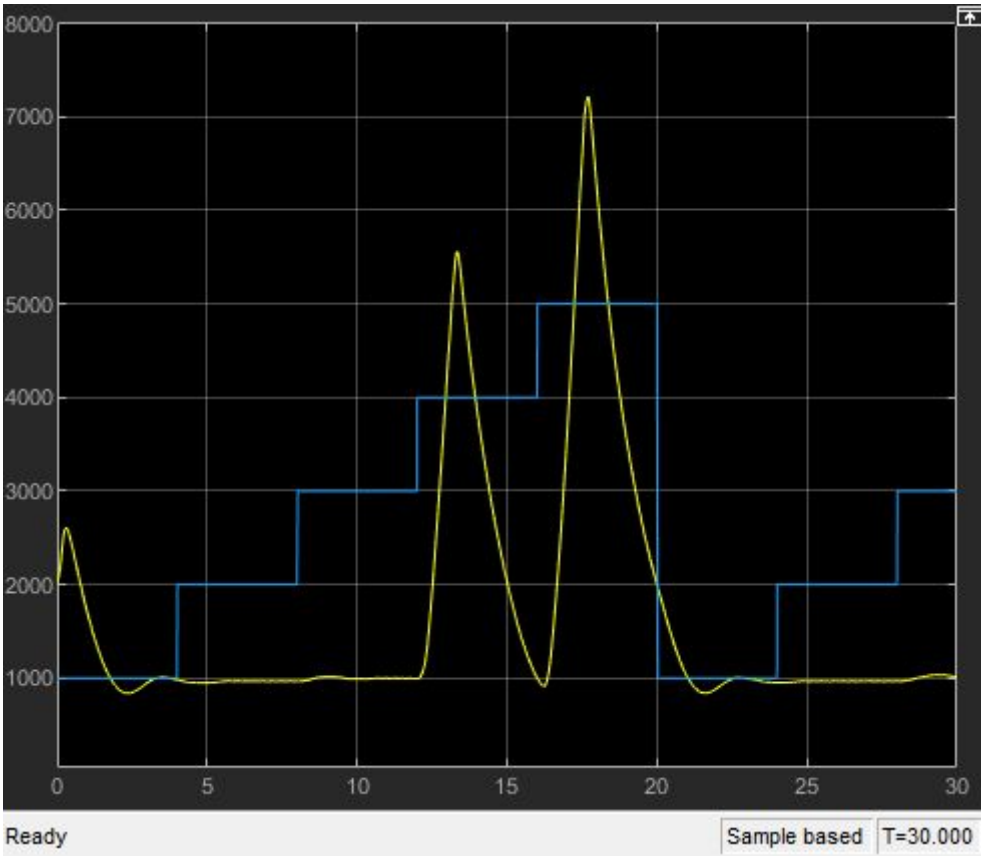


Figura 17: Respuestas iniciales. La red no es capaz de replicar el comportamiento de la planta hasta reducir el el rango de entrada de la planta

Después de limitar el número de entradas a la planta al rango [1,20], obtenemos la respuesta mostrada en la Figura 18.a. Notablemente, tenemos un controlador que sigue la señal, pero con oscilaciones fuertes, similar a nuestros primeros intentos de optimización en reportes anteriores. Observando detenidamente la señal de salida del bloque NARMA/entrada a la planta (Figura 18.b). Notamos que la señal alcanza los valores límites frecuentemente, y que la resolución de respuesta es pequeña.

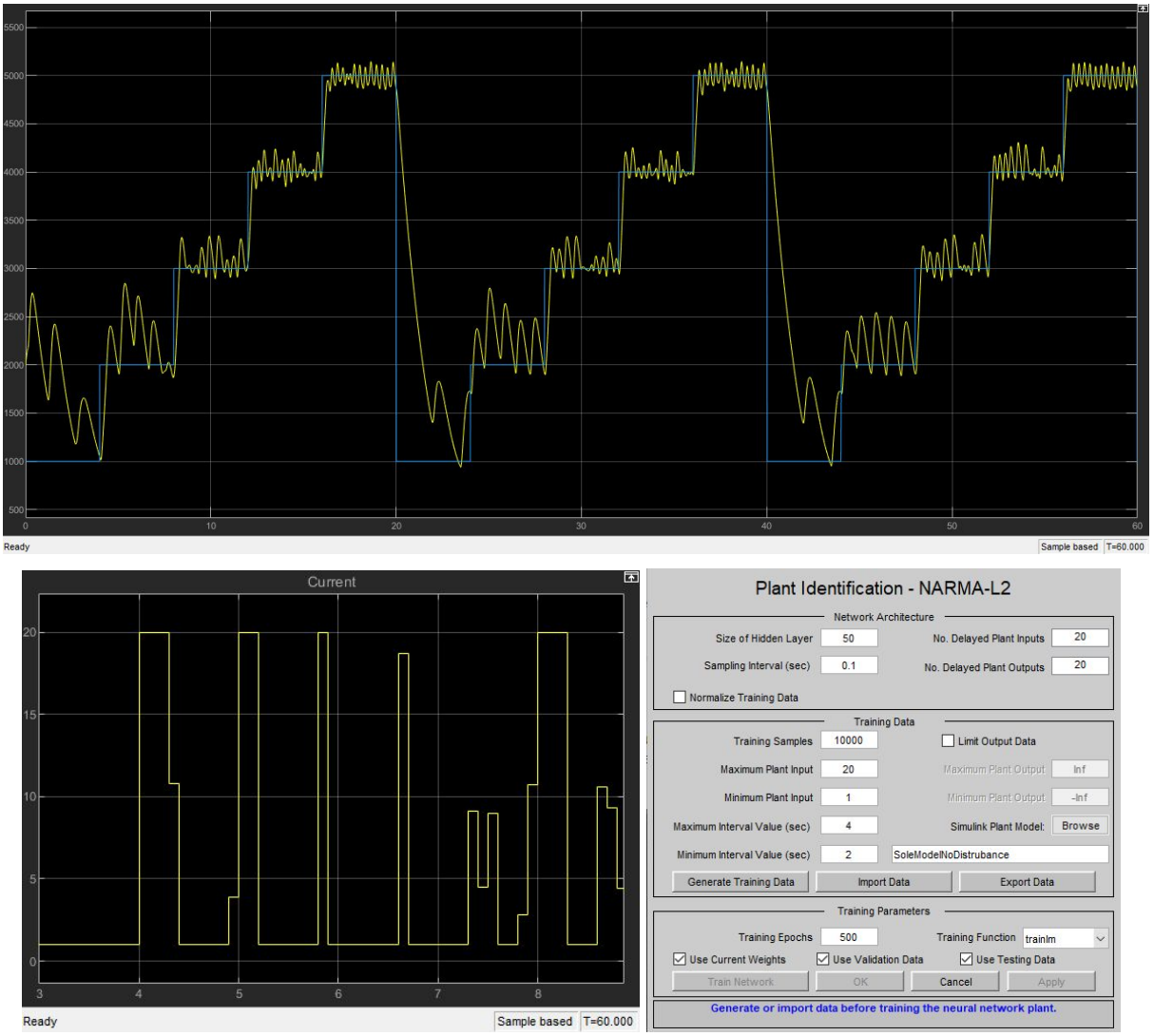
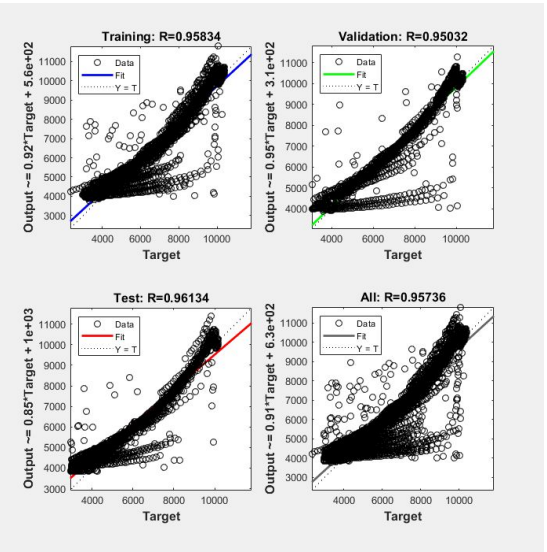
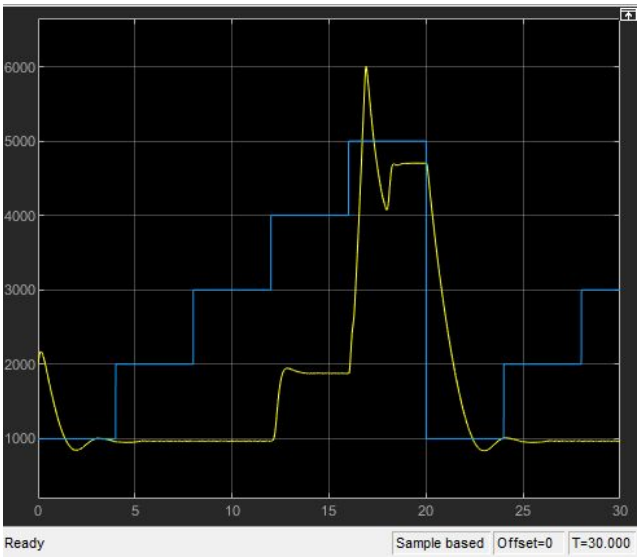


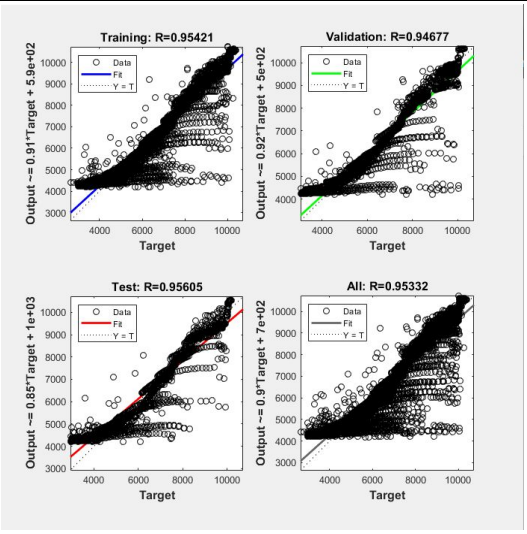
Figura 18: Respuesta para primera iteración

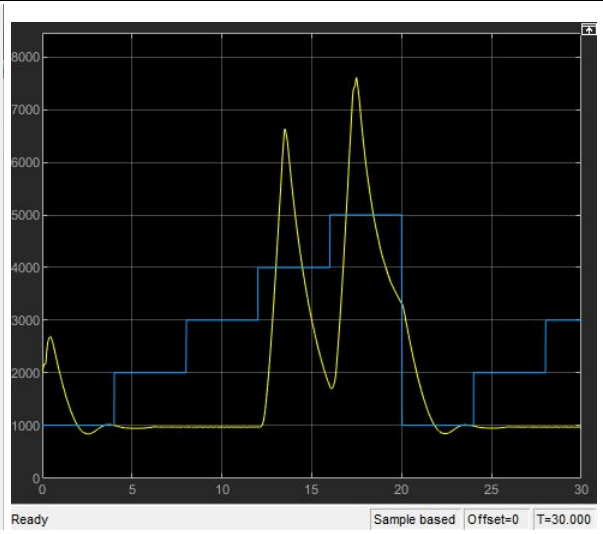
Repetimos para un rango más elevado de entrada [1-40], y duplicamos la resolución temporal. Después de generar datos, experimentamos con diferentes parámetros y sus efectos (Figura 18). Podemos apreciar que una correlación más alta suele implicar una red neuronal efectiva como controlador. Resaltamos que los valores escogidos para la configuración de la red resultan en un tiempo muy elevado de ejecución, por lo que en futuras iteraciones se seleccionan valores más conservadores.



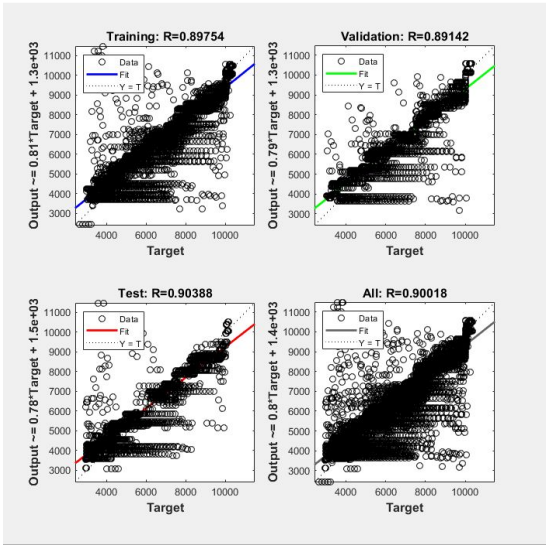


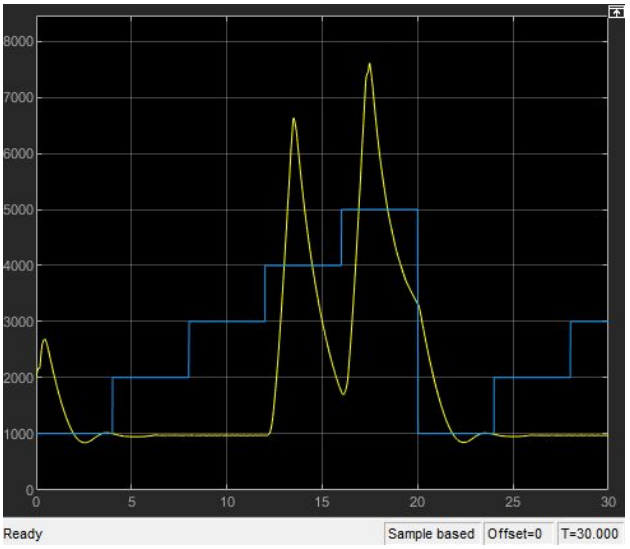
Tamano capa 10	Delayed Inputs 3	Delayed Outputs 2	R^2~.957



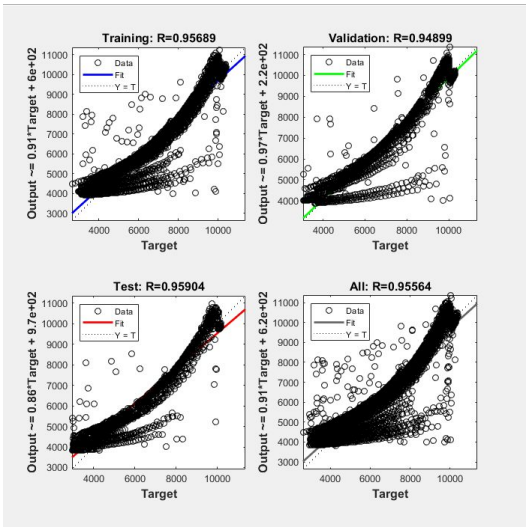


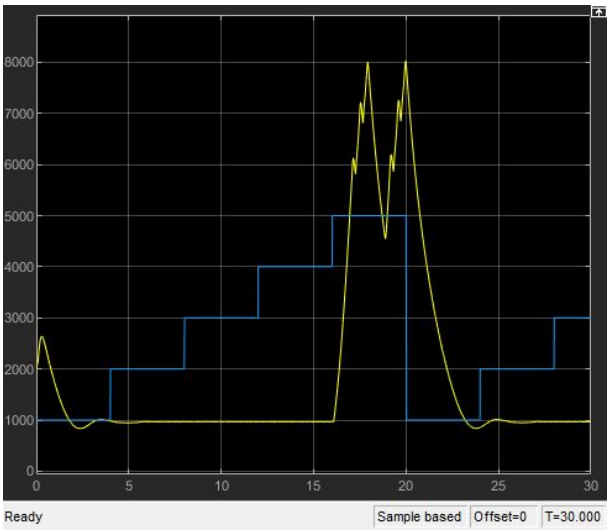
Tamano capa 5	Delayed Inputs 5	Delayed Outputs 5	R^2~0.953





Tamano capa 10	Delayed Inputs 5	Delayed Outputs 5	R^2~0.906

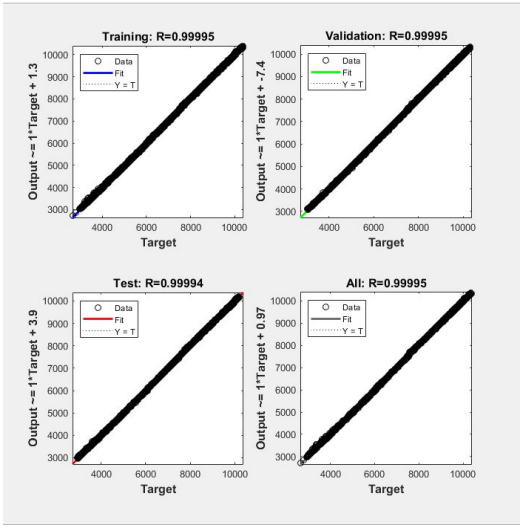


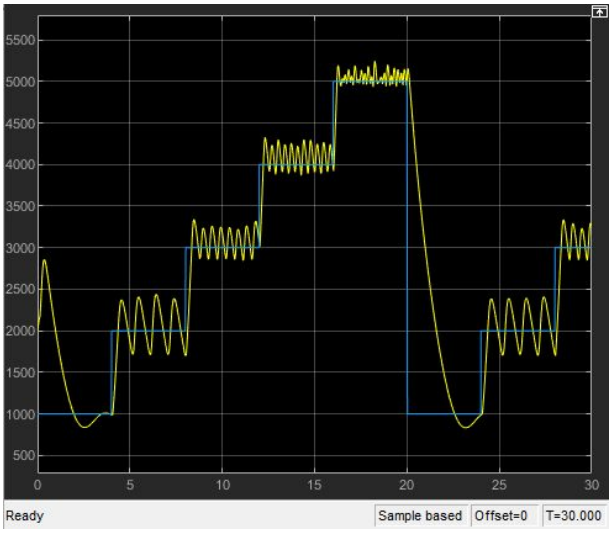


Tamano capa 20	Delayed Inputs 5	Delayed Outputs 5	R^2~0.955

Figura 18: Resultados y respuestas para varias combinaciones de parámetros

Durante este proceso de experimentación, notamos un factor clave: Los resultados no son consistentemente repetibles, los resultados del entrenamiento pueden cambiar entre ejecuciones del algoritmo.





Tamano capa 5	Delayed Inputs 5	Delayed Outputs 5	**R^2~0.99995**

Figura 19: Diferente respuesta para combinación de parámetros analizada previamente. "*NarmaMotorS555.sxl*"

Para estudiar el efecto de las variables en la respuesta final, realizamos un breve experimento: Fijando dos variables y cambiando la una tercera, tomando el mejor valor de

correlación para 3 ejecuciones, obtenemos los datos presentados en la tabla a continuación (Figura 20).

Tamano capa	Delayed Inputs	Delayed Outputs	R^2
5	***5***	***5***	***0.99995***
5	5	3	0.956
5	5	7	0.95
5	3	5	0.9865
5	7	5	0.99957
3	5	5	0.85
7	5	5	0.82
7	***7***	***7***	***0.99995***
6	6	6	0.87767
4	***4***	***4***	***0.9999***
3	3	3	0.8744

Figura 20: Combinaciones de valores probados. R^2 mejor de 3 intentos. Respuestas que resultan en controladores exitosos resaltadas.

Obtuvimos dos configuraciones que resultan en respuestas correctas. Notamos que R^2 debe estar considerablemente cerca de 1 para funcionar apropiadamente. Curiosamente, grupos de parámetros con los mismos valores generaron respuestas exitosas. Todas las respuestas exitosas obtenidas hasta ahora son muy similares, con oscilaciones violentas.

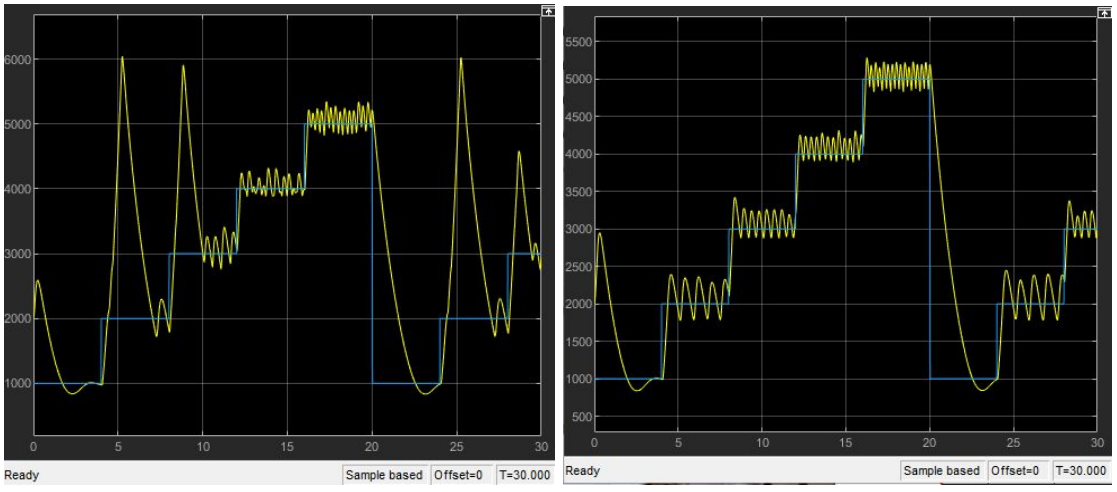


Figura 21: Respuestas para (a) Valores =7 "*NarmaMotorS777.sxl*",
(b) Valores = 4 "*NarmaMotorS444.sxl*"

Conclusion

En el transcurso de este reporte, conseguimos desarrollar exitosamente un controlador basado en una red neuronal para un sistema muy complejo. Interpretamos la información obtenida en diferentes etapas para tomar decisiones informadas y realizamos modificaciones a la planta para generar datos más relevantes a nuestras necesidades particulares. Sin embargo, obtuvimos una respuesta poco estable, con un gran número de oscilaciones.

Utilizar redes neuronales como controlador involucra una desventaja importante: no es posible tener control directo sobre la respuesta, solo sobre los factores que la generan. Durante el entrenamiento del modelo, el diseñador dispone de control de parámetros sin equivalente real, que pueden resultar un poco arbitrarios. Sin embargo, es posible diseñar muestras de datos particulares que mejores resultados particulares, utilizando información del sistema para aplicar correcciones y concentrarse en zonas de trabajo problemáticas. En nuestro caso alteramos los rangos de entrada de la planta para concentrarse en valores pequeños y su respuesta.

Debido a los tiempos elevados de generación de datos y entrenamiento, experimentar con diferentes configuraciones resulta particularmente difícil. Optimizar el sistema previamente resulta extremadamente importante. La consideración inicial con tiempos de ejecución permite disminuir nuestro tiempo de ejecución de potencialmente varias horas a fracciones de esta.

En una red neuronal, la distribución de pesos inicial se realiza al azar, y como vimos durante la sesión de entrenamiento, puede causar resultados muy diferentes para las mismas condiciones iniciales. Esta falta de repetibilidad es un problema muy grande al obtener conclusiones confiables sobre qué factores influencian al sistema sin realizar experimentos a gran escala, fuera del foco de trabajo de este proyecto.

El uso de redes neuronales puede presentar una ventaja marcada respecto a otros sistemas. Si se dispone de suficientes recursos de hardware y tiempo, es posible replicar el comportamiento de la planta. Para mejorar el controlador obtenido en este reporte, la mejor opción es simplemente aumentar el número de muestras de entrada y ciclos de ejecución.