ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Desarrollo y evaluación de desempeño de sistemas de control inteligente basados en herramientas de simulación “in-the-loop”

Játiva Cervantes, Jean Carlo

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Gordillo Orquera, Rodolfo Xavier, Ph. D.

1 de agosto del 2022

Jativa_TrabajoTitulación.pdf

Scanned on: 15:25 July 31, 2022 UTC

Identical Words	407
Words with Minor Changes	32
Paraphrased Words	409
Omitted Words	0

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Certificación

Certifico que el trabajo de titulación: "**Desarrollo y evaluación de desempeño de sistemas de control inteligente basados en herramientas de simulación "in-the-loop"**" fue realizado por el señor **Játiva Cervantes Jean Carlo**; el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 1 de agosto del 2022

Firma:

Firmado electrónicamente por:
RODOLFO XAVIER
GORDILLO ORQUERA

.....

Ing. Gordillo Orquera, Rodolfo Xavier, Ph.D.
C. C: 1001510203

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Autoría

Yo, **Játiva Cervantes Jean Carlo**, con cédula de ciudadanía N° 1716562697, declaro que el contenido, ideas y criterios del trabajo de titulación: **Desarrollo y evaluación de desempeño de sistemas de control inteligente basados en herramientas de simulación "in-the-loop"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 1 de agosto del 2022

Firma:

.....

Játiva Cervantes, Jean Carlo

C. C: 1716562697

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Autorización de Publicación

Yo, **Játiva Cervantes Jean Carlo** con cédula de ciudadanía N° 1716562697 autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Desarrollo y evaluación de desempeño de sistemas de control inteligente basados en herramientas de simulación "in-the-loop"**", en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 1 de agosto del 2022

Firma:

.....

Játiva Cervantes, Jean Carlo

C. C: 1716562697

Dedicatoria

A mi mis padres, hermano y abuelos que siempre han sido mi norte, mi fuerza y sin su apoyo estaría incompleto. A mis amigos con quienes compartimos muchas alegrías y por nuestra capacidad de ver una solución cuando adelante había un problema. A mí mismo, por nunca darme por vencido y por recordarme que mi determinación es mi mayor virtud.

Játiva Cervantes Jean Carlo.

Agradecimiento

A mi familia por su amor y apoyo incondicional en toda mi vida, por su confianza y fe en mi, y por sus anhelos de verme triunfar. A mis profesores, en especial a mi tutor de tesis, por ser parte del cumplimiento de este objetivo, por sus comentarios y retroalimentación a lo largo del desarrollo de este proyecto.

Játiva Cervantes Jean Carlo.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

El presente proyecto tiene como objetivo impulsar el desarrollo de estrategias de control inteligente aplicables en entornos industriales actuales. Existen limitaciones en la implementación de controladores inteligentes en procesos de automatización debido a que gran parte de las plataformas de hardware no disponen de las librerías necesarias, o la realización del algoritmo de control se da de forma manual o no automatizada. Sin embargo, nuevas herramientas como la generación de código y el proceso de simulación in-the-Loop permiten aterrizar controladores inteligentes en sistemas embebidos. Esta propuesta contempla el diseño de las siguientes estrategias de control: un controlador difuso para un sistema que emplea un actuador no lineal, un controlador por ajuste de ganancias que rija sobre el reactor de tanque agitado (CSTR), y una red neuronal por modelo inverso que actúe sobre un modelo entrada salida. Posteriormente los controladores son convertidos a texto estructurado, generado a partir de la herramienta PLC Code de Simulink. Por último, se presentan los resultados de los controladores tanto en la simulación efectuada en Matlab/Simulink, como en la puesta en marcha virtual donde el algoritmo de control es ejecutado en el PLC virtual de TIA Portal y la planta aún reside en Simulink. La comparación de los resultados muestra que existe una buena relación de lo experimental con la implementación, sin embargo, algunos controladores no tuvieron el éxito esperado.

Palabras clave: controlador inteligente, generación de código, in-the-loop, puesta en marcha virtual.

Abstract

This project aims to increase the development of intelligent control strategies applicable in current industrial environments. There are constraints in the deployment of intelligent controllers in automation processes because most hardware platforms do not have the required libraries, or the implementation of the control algorithm is manual or non-automated. Nevertheless, new tools such as code generation and in-the-Loop simulation process make it feasible to introduce intelligent controllers in embedded systems. This proposal provides the design of the following control strategies: a fuzzy controller for a system that employs a nonlinear actuator, a gain scheduling controller that operates on the stirred tank reactor (CSTR), and a neural network inverse model acting on an input-output model. The controllers are then turned into structured text, which is generated from the Simulink PLC Code tool. Finally, the results of the controllers are presented both in the simulation performed in Matlab/Simulink and in the virtual commissioning where the control algorithm is executed in the virtual PLC of TIA Portal and the plant still runs in Simulink. The comparison of the results shows that there is a good relationship between the experimental and the implementation, however, some controllers were not as successful as expected.

Keywords: intelligent controller, code generation, in-the-loop, virtual commissioning.

Capítulo I

Introducción

Antecedentes

Actualmente la industria está atravesando su cuarta revolución, esta se encuentra estrechamente ligada con la industria 4.0 cuyo eje central es la interconectividad entre múltiples sistemas, donde el almacenamiento, interpretación y análisis de grandes volúmenes de información juega un papel fundamental. La integración de la información a lo largo de la cadena de producción permite que las empresas disminuyan la cantidad de recursos utilizados y optimicen procesos industriales, entre otros beneficios. Las nuevas tecnologías disruptivas (Big Data, Machine Learning, IA, Cloud Computing, etc.) que van surgiendo deben ser aprovechadas para la creación y/o mejoramiento de productos/servicios. Otra ventaja que trae consigo la industria 4.0 es la introducción de manufactura inteligente, cuyo objetivo es ser el enlace entre los entornos físicos y digitales, de esta manera se consigue optimizar el diseño, despliegue y operación de los procesos de manufactura. (Jbair, y otros, 2019)

Dentro de la manufactura inteligente se manejan las herramientas de modelamiento, estas son ampliamente usadas en la industria ya que ofrecen capacidades y características a lo largo del ciclo tanto de producto como de sistema. Las herramientas de modelamiento también llamada ingeniería virtual permiten conocer el comportamiento del sistema, mediante la interacción de sus componentes a partir de la simulación realizada.

Sin embargo, a pesar de los significativos avances en múltiples áreas de la automatización no se han extendido completamente las estrategias de control inteligente en entornos industriales debido a que las plataformas de hardware no cuentan con las prestaciones como

librerías para la implementación de estos algoritmos, o en el caso de que sí las tengan no son ampliamente conocidas y la información que existe es precaria.

Una manera de solucionar esto y al mismo tiempo acortar la brecha entre sistemas físicos y digitales es la generación de código, se trata de un proceso de transformación del comportamiento del sistema de control que ha sido definido en un nivel alto de abstracción durante la fase de diseño, en código que ya puede ser directamente implementado en el proceso de automatización.

Uno de los procedimientos usados para que la generación de código sea de calidad es mediante el proceso de simulación “in-the-loop”, conformada por MIL (Model-in-the-Loop), SIL (Software-in-the-Loop), PIL (Processor-in-the-Loop) y HIL (Hardware-in-the-Loop); estas etapas forman parte de la verificación del diseño basado en modelos. Antes de que el modelo sea implementado en el hardware para producción se deben realizar unos pasos de verificación, y es aquí donde el proceso de simulación in-the-loop entra en acción.

Este proceso de simulación ya ha sido utilizado en varios proyectos, como en el trabajo (Aluisa, 2014) donde se desarrolló un sistema de identificación de procesos industriales en línea, aquí se ensayaron varios algoritmos de identificación para un conjunto de plantas en tiempo real. Otro trabajo reciente (Sánchez, 2020) plantea un escenario parecido donde se realizó la identificación y control neuronal de sistemas dinámicos, pero esto no fue realizado en tiempo real sino a partir de herramientas de simulación.

El presente trabajo está organizado de la siguiente manera: en el capítulo dos se realiza una revisión general de la teoría y herramientas involucradas en el desarrollo de controladores, seguido por un resumen del estado del arte acerca de la implementación de controladores inteligentes. El capítulo tres comprende el diseño de las estrategias de control para cada uno de

los sistemas no lineales, además, se presentan los resultados de la simulación únicamente efectuada en Matlab/Simulink, mientras que en el capítulo cuatro las mismas estrategias de control son exportadas como texto estructurado, y ejecutadas sobre el controlador virtual en el entorno de desarrollo integrado de Siemens, TIA Portal. En el capítulo cinco se compara el desempeño individual de cada etapa a partir de las métricas de la señal de salida. Por último, en el capítulo seis se presentan las conclusiones acerca de la implementación de estrategias de control inteligente sobre entornos industriales reales.

Justificación e importancia

La calidad que posee el software embebido es uno de los grandes problemas actuales a nivel industrial, existen múltiples errores sobre las unidades de control eléctrico (ECU) cuando ya se integra el software embebido, esto sin contar con los errores previos que son desvelados en las fases de testeo. Debido a que los tiempos de desarrollo de productos se acortan, además de que el tiempo de lanzamiento al mercado se vuelve cada vez más importante, el testeo de software se ha convertido en el mayor cuello de botella del ciclo de desarrollo. Si se aumentan el número de pruebas ejecutadas en la fase de testeo para mejorar la calidad del software entra en conflicto con el hecho de que por si el testeo ya ocupa la mayor cantidad de recursos y de tiempo disponible en un proyecto. La fase de testeo forma parte del 75% del costo del desarrollo de software y esta métrica continúa subiendo, si se pretende eliminar los errores mediante testeo variante se influye sobre la plataforma completa de software, introduciendo complejidad al testeo. Por ejemplo, si un proceso tiene una decena de variantes esto puede llevar a cientos o incluso miles de pruebas para una sola plataforma de software, volviendo de esta manera a la fase de testeo en un problema abrumador. (Reyes, 2014)

Cuando se conecta la unidad de control eléctrico con la planta en lazo cerrado, se pretende que el controlador actúe sobre diferentes elementos de la planta y al mismo tiempo mida su respuesta por medio de sensores. En este enfoque la fase de testeo solo puede ser ejecutada una vez que se disponga de un prototipo de la planta que por lo general suele ser caro, además de que cuando se conectan la unidad de control y la planta por primera vez existe un gran riesgo de que la planta sufra daños si el software de la unidad de control encuentra un problema. (Jbair, y otros, 2019)

Además, también se debe tomar en cuenta que las herramientas de modelamiento mencionadas anteriormente no se han extendido completamente a las fases de despliegue y operación. Razón por la cual la brecha entre los entornos digitales y físicos aún es considerable, a esto se debe añadir que la generación de código para su uso industrial suele ser limitado, impráctico o no proveen la solución completa, con esta pobre generación de código múltiples algoritmos de control inteligente tales como redes neuronales, algoritmos genéticos y lógica difusa quedan solamente plasmados en la teoría y no son aterrizados a un uso real.

Entre las ventajas del uso de las herramientas in-the-loop se destacan la realización de ensayos sobre los modelos sin comprometer el proceso físico de ninguna manera, estas pruebas otorgan una noción de cómo se comporta el sistema bajo una situación de estrés o atípica. Otro beneficio es la mejora continua de la calidad del código generado a medida que se pasa del proceso de simulación a la realización física, obteniendo así una evaluación previa que sigue un proceso de depuración con las herramientas in-the-loop. Mediante el uso de las técnicas de control clásicas el proceso de generación de código se da de manera directa a partir del uso de funciones o librerías, en cambio, para las estrategias de control inteligente la generación de código es un procedimiento más escabroso, cuya adaptación presenta cierta complejidad ya que no se disponen de las herramientas que sí hay para los controladores clásicos como un PID.

Este proyecto busca promover el uso de controladores inteligentes más allá de los fines pedagógicos, la evaluación de estos controladores sobre sistemas complejos no lineales mediante el uso de herramientas para la generación de código.

Alcance del Proyecto

Como primer etapa del proceso de simulación in-the-loop se debe crear un modelo de la planta y controlador, se lo puede considerar como el lazo de control cerrado convencional, este procedimiento lleva el nombre de MIL, aquí se evalúa el controlador sobre el modelo simulado de la planta. Cuando el modelo sea verificado la siguiente etapa es SIL, donde se genera código únicamente para el modelo del controlador y se remplaza el bloque del controlador con este código. Posteriormente se ejecuta la simulación con el bloque del controlador y la planta, mediante esto se obtendrá una idea clara de si el modelo del controlador puede ser convertido a código y si su hardware es implementable. (MathWorks Support Team, 2021)

La siguiente etapa del proceso es PIL, en esta etapa el modelo del controlador es embebido en un procesador y se ejecuta una simulación en lazo cerrado con la planta simulada. Es decir, se reemplaza el subsistema del controlador por un bloque PIL que contenga el código del controlador que se ejecuta sobre un procesador.

Una vez que el modelo de la planta ha sido verificado usando PIL, el software operacional y el modelo simulado de la planta están listos para el testeo HIL, aquí se verifica la funcionalidad integral y operacional del desempeño. Además, se identifican problemas relacionados con la comunicación de los canales, como atenuación y delay.

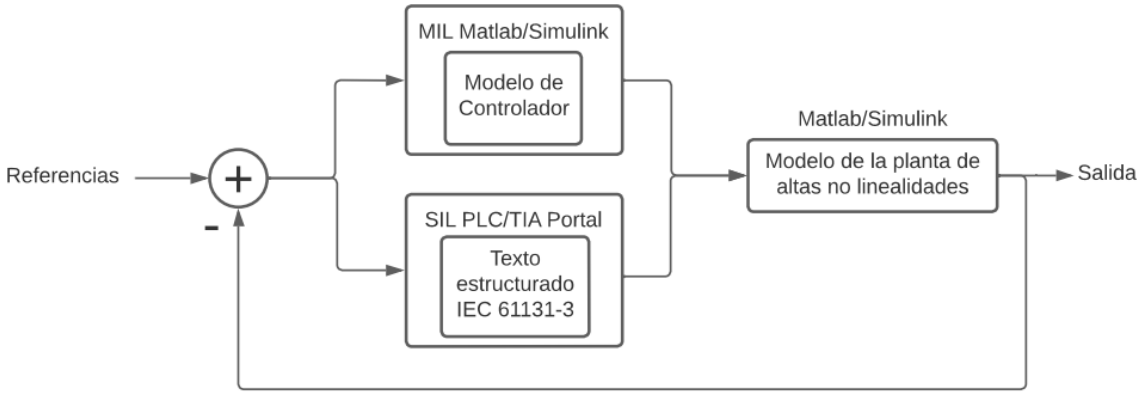
En el presente proyecto se plantea dos etapas del proceso de simulación in-the-Loop, estas son las de MIL y SIL orientadas hacia la generación de código. Para ambas etapas se tiene considerado tres casos de estudio representativos, estos son tres plantas que presentan altas no

linealidades como lo son: un sistema que contempla un actuador no lineal para controlar el proceso, el reactor de tanque agitado el cual es característico de procesos industriales (De La Concha-Gómez, Ramírez-Muñoz, Márquez-Baños, Haro, & Alonso-Gómez, 2019) y el tercer sistema es el modelo de entrada y salida propuesto por (Narendra & Parthasarathy, 1990) que presenta altas no linealidades.

La etapa MIL se desarrollará completamente sobre Matlab/Simulink para el diseño de los controladores y la evaluación de desempeño de los sistemas de control en lazo cerrado. Para la segunda etapa SIL los controladores que en la anterior etapa eran un bloque en el sistema de control, ahora por medio de la herramienta de generación de código PLC Code de Simulink, serán ejecutados sobre el PLC virtual en el entorno de desarrollo integrado de Siemens, TIA Portal, mientras que las plantas no lineales se mantendrán sobre Matlab/Simulink. Este procedimiento descrito anteriormente se resume en la Figura 1, donde se observa el sistema retroalimentado con el modelo del controlador y con el código generado. Finalmente, la validación de resultados de la fase SIL corresponde a que el código que se haya generado efectivamente controle los modelos de los casos de estudio planteados, manteniendo las características que presentaban los resultados en la simulación (MIL).

Figura 1

Esquema de realización de las herramientas in-the-loop



Para el caso de estudio del sistema de actuador no lineal se propone la cancelación de las no linealidades a través de la realización de un controlador difuso, en este escenario primeramente se generan las características estáticas del actuador, luego para el controlador se realiza la fusificación las de variables de entrada, la creación de funciones de membresía, defusificación y creación de la curva de control. El diseño del controlador difuso se lo realizará mediante aproximación de Mamdani y Takagi-Sugeno, analizando el desempeño de las dos metodologías y escogiendo la que mejor reproduzca las características inversas y directas del actuador.

En el escenario del reactor de tanque agitado se plantea una estrategia de control por ajuste de ganancias. Para este controlador en particular se requiere como primer paso la linealización en los puntos de operación específicos, la sintonización de los controladores PID para cada uno de los puntos establecidos y el ajuste de las ganancias proporcional, integral y derivativa del controlador para su implementación.

Por último, para el modelo de entrada y salida que presenta altas no linealidades se propone un control por modelo inverso. El diseño del controlador por modelo inverso empieza

desde la generación de los patrones de entrenamiento para los modelos, luego la identificación del modelo directo para evaluación y del modelo inverso para control y selección óptima del modelo (número de neuronas en cada capa, capas ocultas, etc.).

Hasta este punto se habrá completado la fase MIL del proceso in-the-loop, continuando con la fase SIL donde la conversión a código se hace mediante el uso de la herramienta PLC Code. Este trabajo propone automatizar la generación de código otorgando un valor agregado a los procedimientos convencionales, buscando así establecer una metodología en entornos virtuales a partir del proceso de simulación in-the-loop, donde se asegura que la fase de testeo ocurra mucho más antes y permita mejorar la calidad del software generado.

Objetivos

Objetivo general

Diseñar y evaluar el desempeño de estrategias de control inteligente mediante el uso de herramientas de simulación in-the-loop para controlar sistemas no lineales complejos.

Objetivos Específicos

Investigar la teoría acerca del proceso de simulación in-the-loop, de las herramientas para la generación de código para la implementación industrial de controladores.

Diseñar las diferentes estrategias de control para los procesos de actuador no lineal, reactor de tanque agitado y modelo de entrada y salida de altas no linealidades.

Evaluar las estrategias de control inteligente para la primera etapa de la simulación in-the-loop, MIL, mediante el uso de Matlab/Simulink.

Evaluar las estrategias de control inteligente para la segunda etapa de la simulación in-the-loop, SIL, mediante el uso de Matlab/Simulink y TIA Portal.

Comparar el desempeño de ambas etapas del proceso de simulación in-the-loop a partir de los resultados individuales de cada etapa.

Capítulo II

Marco Teórico

Introducción

A continuación, se exponen los conceptos necesarios para el diseño de los controladores y su posterior generación de código. Un marco teórico concreto permite establecer los fundamentos sobre los cuales se trabajará y da lugar a la compresión de las herramientas que se usarán para la generación de código.

La distribución explica las estrategias de control por utilizar, comenzando por la lógica difusa, controladores autoajustables, redes neuronales, identificación de sistemas no lineales y los controladores por modelo de referencia y modelo inverso. Por último, se presenta un resumen de las herramientas orientadas hacia la generación de código y sus usos.

Controladores basados en lógica difusa

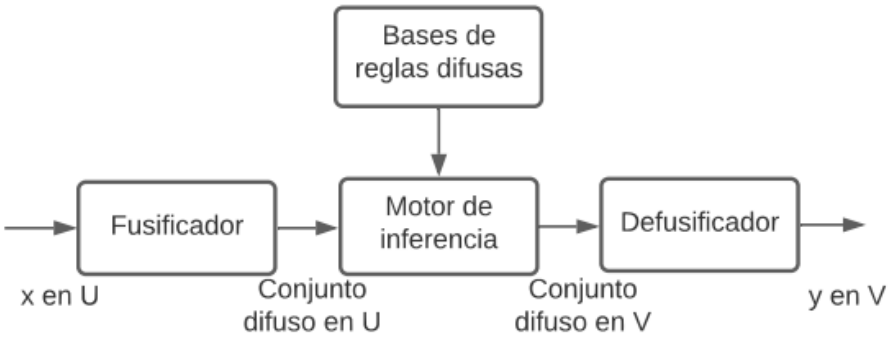
La lógica difusa tiene un gran campo de aplicación, desde de la teoría de control hasta aplicaciones en inteligencia artificial, a diferencia de la lógica booleana, donde las variables solo toman valores enteros de 0 o 1, la lógica difusa emplea valores de verdad parciales donde el valor final depende de un rango entre completamente cierto o completamente falso.

(Salvador, 2007)

Los sistemas difusos son usados para contrarrestar incertidumbres que no siempre son randómicos, sino que provienen de perturbaciones externas o son inherentes a la propia planta. La lógica difusa está basada en un sistema cognitivo, este no es más que una máquina de inferencia o un conjunto de reglas, el esquema de estos sistemas se presenta en la Figura 2.

Figura 2

Sistema difuso con fusificador y defusificador



Nota: Gráfico elaborado a partir de la información encontrada en (Salvador, 2007)

Fusificación

El proceso de fusificación consiste en mapear las entradas que se tienen en un conjunto de valores de membresía, en esta etapa se deben considerar el número de entradas, el tamaño del universo de discusión y el número de conjuntos difusos. Para fusificar una entrada como lo sería el error en un sistema de control e, se pueden usar tres formas estándares de conjuntos:

- {NB, NS, ZO, PS, PB}
- {NB, NM, NS, ZO, PS, PM, PB}
- {NB, NM, NS, NZ, PZ, PS, PM, PB}

Siendo N negativo, Z cero, P positivo, B grande, M mediano y S pequeño.

Conjunto de reglas

De manera general, el conjunto de reglas difusas se conforma de:

$$SI\ antecedente\ 1\ y\ antecendente\ 2\ ENTONCES\ consecuente$$

La parte *SI* es llamada premisa y se conforma por la o las condiciones correspondientes a las entradas del sistema, mientras que la parte *ENTONCES* llamada consecuente es la acción que se realiza para determinada condición.

El modelo difuso tradicional es el de Mamdani, otro ampliamente conocido es el de Takagi-Sugeno, la diferencia de estos modelos radica en que Takagi-Sugeno propone funciones lineales o constantes como salida, obteniendo mejor precisión en la curva de control; en cambio Mamdani mantiene los conjuntos difusos como salidas. Ambos tipos de controladores presentan las siguientes características:

- El conjunto de reglas sintetiza el conocimiento humano por el cual se pretende controlar el sistema.
- La parte cognitiva decide que regla es la prioridad en determinado tiempo, decidiendo el valor de entrada hacia la planta.
- El bloque fusificador modifica las entradas para que sean interpretadas y comparadas con el conjunto de reglas.
- El bloque defusificador transforma las salidas de la máquina de inferencia en las entradas al proceso.

Inferencia difusa

Del conjunto de reglas se obtiene la matriz de relación difusa *R*. Si se tiene nueva información de las premisas *A* y *B,* se tiene como resultado:

$$C = (A \times B) \cdot R$$

Siendo la matriz *C* un vector difuso, que debe ser defusificado para conseguir un valor práctico. (Gordillo, 2020)

Defusificación

La defusificación consiste en mapear a partir de un conjunto de señales de control difusas contenidas en una ventana de la salida difusa en una señal de control no difusa. El método para realizarlo es mediante el centroide. Para sistemas discretos se expresa como:

$$u(kT) = \frac{\sum_{i=1}^{n} x_i \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}$$

El controlador difuso opera en un sistema en lazo cerrado, comparando la salida de la planta con la entrada o referencia y a partir de esto decide que salida del controlador o entrada del proceso es la más óptima. Los sistemas difusos tienen como objetivo incluir la parte cognitiva o conocimiento humano a los sistemas de ingeniería.

Controlador por ajuste de ganancias

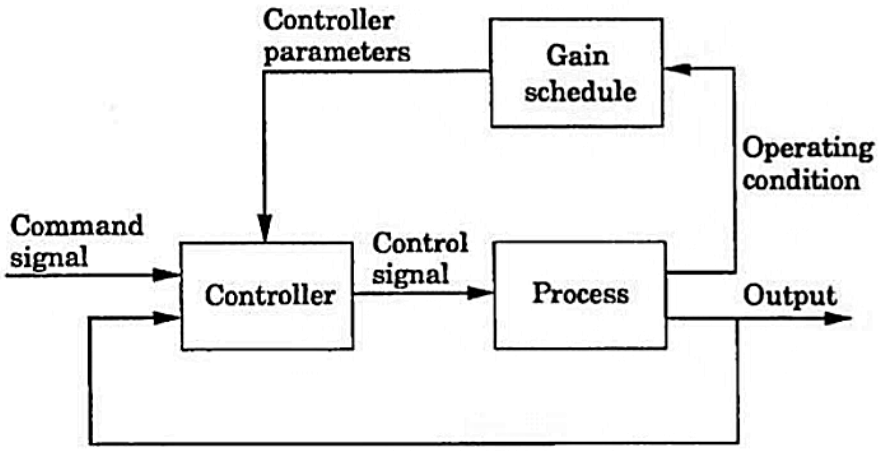
El ajuste de ganancias es una técnica empleada para controlar sistemas no lineales cuya dinámica cambia de una condición de operación a otra. Se basa en la medición de estas condiciones de operación propias del proceso para compensar las variaciones de los parámetros del proceso o las no linealidades. Al mantener un monitoreo de las condiciones de operación del proceso es posible el ajuste de los parámetros del controlador.

El principal problema de estos sistemas de control es encontrar las variables apropiadas para el ajuste, esto se suele hacer en función del conocimiento de la física que rige el proceso. Una vez que se hayan encontrado estas variables que se relacionan con la dinámica del proceso, es posible reducir los efectos de las no linealidades mediante el cambio de los parámetros del controlador como funciones de las variables ya determinadas, esto se propone en la Figura 3 que ejemplifica el esquema de este sistema.

El controlador es sintonizado para cada una de las condiciones de operación, tanto la estabilidad como el desempeño son evaluadas durante la simulación del sistema. Un punto importante a considerar es la transición entre diferentes condiciones de operación, ya que se busca que el cambio sea suave y no abrupto. (Åström & Björn, 2008)

Figura 3

Esquema del controlador por ajuste de ganancias



Nota: Gráfico extraído de (Åström & Björn, 2008)

Controladores basados en redes neuronales

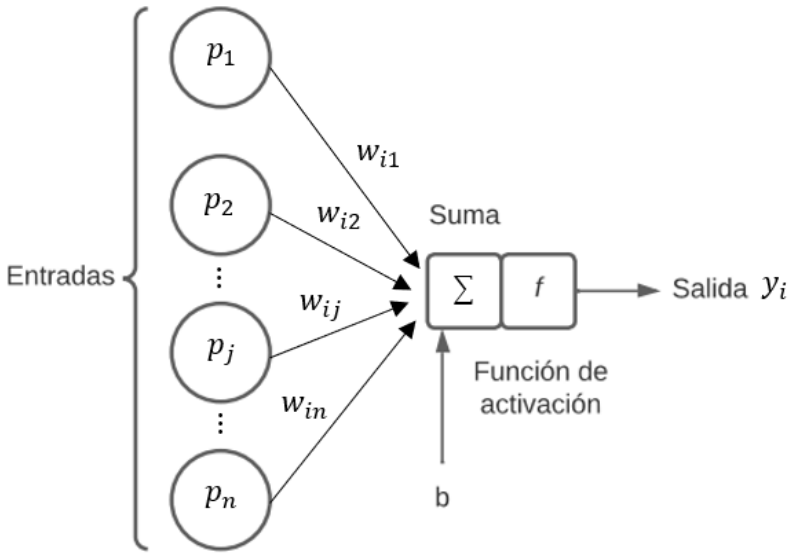
Las redes neuronales son un conjunto de células nerviosas, también llamadas neuronas, alojadas en el cerebro. El cerebro humano cuenta con billones de neuronas individuales y hasta trillones de interconexiones entre ellas. La función de estas neuronas es la de continuamente procesar y transmitir información.

Una red neuronal artificial asemeja la estructura de la neurona biológica, copiando la estructura y el comportamiento, se encarga de recibir señales diferentes p_n que son procesadas de una manera anteriormente definida (entrenamiento). El resultado del procesamiento dicta si

la neurona activa su salida o no. Las partes de las cuales se compone una neurona se visualiza en la Figura 4. (Pham & Xing, 2018)

Figura 4

Modelo de una neurona artificial



Nota: Gráfico elaborado a partir de la información encontrada en (Nacelle, 2009)

Los elementos que forman parte de la neurona artificial son:

- Las entradas ($p_1, \dots, p_n$) se encargan de recibir datos de otras neuronas, estas son características como color, peso, mediciones de sensores, etc.
- Los pesos sinápticos ($w_{i1}, \dots, w_{in}$) son modificados de acuerdo al entrenamiento de la red realizado, añadiendo el factor de importancia. Estos miden la intensidad de interacción entre dos neuronas que están conectadas por un enlace.
    - La suma ponderada de las entradas y los pesos sinápticos genera la entrada ponderada total o potencial postsináptico.

- La función de activación se aplica al final a la suma del potencial postsináptico y se suma el bias, obteniendo la salida de la neurona.

Siendo la ecuación $y_i = f\left(\sum_{j=1}^{n} w_{ij} \cdot p_j + bias\right)$ el modelo matemático más elemental de las redes neuronales. En la función de activación de la forma $y = f(x)$, x es el potencial postsináptico, y es el estado de activación. Este modelo de red neuronal es conocido como perceptrón. En la Tabla 1, se resumen algunas funciones de activación más comunes.

Tabla 1

Funciones de activación comunes

Función	***Ecuación***	***Rango***
Identidad	$y = x$	$[-\infty, +\infty]$
Escalón	$y = sign(x)$	$\{-1, +1\}$
	$y = sign(x)$	$\{0, +1\}$
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -1 \\ x, & si + 1 \leq x \leq 1 \\ +1, & si\ x > +1 \end{cases}$	$[-1, +1]$
Sigmoide	$y = \frac{1}{1 + e^{-x}}$	$[0, +1]$
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$
Sinusoidal	$y = Asen(\omega x + \varphi)$	$[-1, +1]$

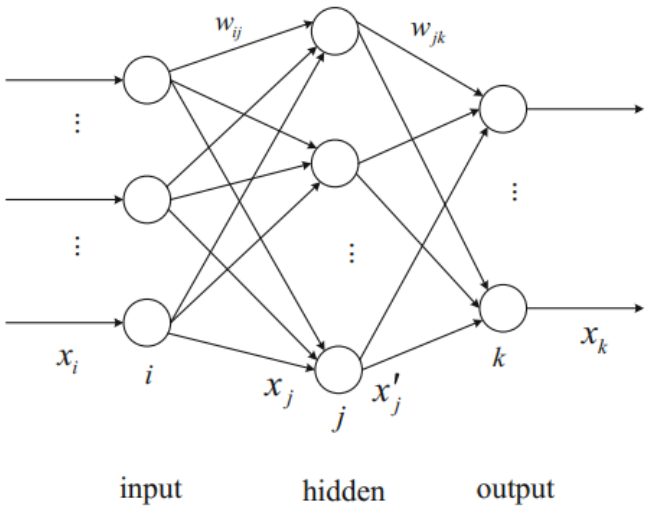
Redes neuronales de aprendizaje profundo

Consisten en un aprendizaje con varias capas ocultas donde se configuran algunos parámetros como épocas, tipo de red feedforward o backforward para no producir un ajuste excesivo. Se trata de un algoritmo en el que el número de capas ocultas y el número de

neuronas son mayores que las de redes neuronales convencionales. La estructura de este tipo de red neuronal se muestra en la Figura 5.

Figura 5

Estructura de una red neuronal multicapa



Nota: Gráfico extraído de (Pham & Xing, 2018)

Este tipo de red neuronal está conformado por:

- Capa de entrada: es la capa que se encuentra más hacia la izquierda donde se suministra los datos de entrada.
- Capa oculta: se trata de un conjunto de neuronas ubicadas entre las capas de entrada y salida, en esta capa todas las unidades de procesamiento son interconectadas a las capas que le suceden y anteceden.
- Capa de salida: es la capa que se encuentra más hacia la derecha, puede conformarse por una sola neurona, obteniendo una salida de forma binaria. Puede también estar formada por más de una neurona consiguiendo salidas con datos enteros o flotantes.

Entrenamiento

El entrenamiento o aprendizaje es el procedimiento de ajuste de parámetros de la red neuronal. Este proceso se basa en determinar un conjunto de pesos sinápticos que le permita a la red efectuar el procesamiento ansiado.

Para la construcción de una red neuronal se debe escoger el modelo, la arquitectura de red, establecer los pesos sinápticos iniciales como nulos o aleatorios. Para que la red opere correctamente es necesario entrenarla, el método más común es modificar los pesos sinápticos siguiendo una regla de aprendizaje, esta se basa en la optimización de una función de error o coste, la cual mide el desempeño de la operación de la red. Este proceso ha terminado, es decir, que la red ha aprendido cuando los valores de los pesos permanecen estables. La ecuación que dicta las modificaciones que sufren los pesos en la etapa de aprendizaje es la siguiente:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k)$$

Donde k es la etapa de aprendizaje, $w_{ij}(k+1)$ es el nuevo peso, $w_{ij}(k)$ es el peso viejo y $\Delta w_{ij}(k)$ es el cambio de peso. (Matich, 2001)

Los algoritmos de aprendizaje pretenden reducir la función de costo, con el uso de métodos numéricos iterativos. La convergencia del algoritmo comprueba si cierta arquitectura, junto con la regla de aprendizaje, es capaz de dar solución al problema. Entre los tipos de entrenamiento se tiene:

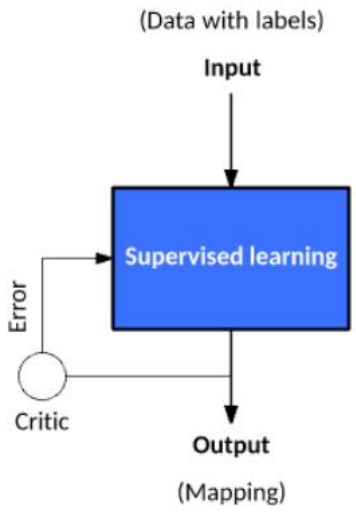
Entrenamiento supervisado

Es realizado a partir de un conjunto de datos que incluye las salidas deseadas de tal manera que una función calcule el error para una predicción dada. Se basa en la creación de una función que puede ser entrenada a partir de un conjunto de datos, para posteriormente ser

aplicada la misma función a otro conjunto de datos que no sean los del entrenamiento, así obteniendo el desempeño predictivo. (Jones, 2017)

Figura 6

Esquema de entrenamiento supervisado



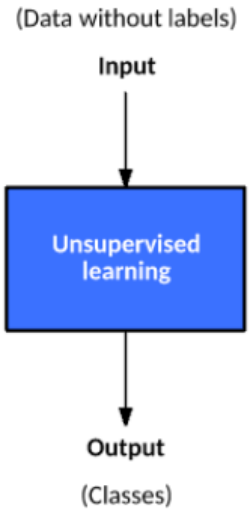
Nota: Gráfico extraído de (Jones, 2017)

Entrenamiento no supervisado

Es realizado a partir de un conjunto de datos que no incluye las salidas deseadas, por lo que existe manera de supervisar la función. En este caso la función se encarga de segmentar los datos en clases o grupos para que cada uno de estos contenga una pequeña porción del conjunto de datos original con características comunes. (Jones, 2017)

Figura 7

Esquema de entrenamiento no supervisado



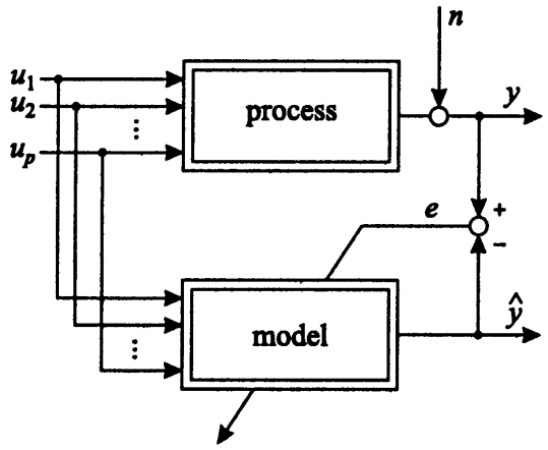
Nota: Gráfico extraído de (Jones, 2017)

Identificación de sistemas no lineales

La tarea de identificar sistemas no lineales puede ser escabrosa y de alta complejidad debido a que los procesos no lineales son únicos en el sentido de que no comparten muchas propiedades. Por lo que se busca un esquema general para la identificación y modelamiento de sistemas no lineales, es decir, que describan una variedad de sistemas estructuralmente diferentes. En la Figura 8 se representa el esquema para la tarea de identificación, el modelo que se consigue debe representar el comportamiento del proceso lo más fiel posible. Para valorar la calidad del modelo se mide en función del error entre la salida del proceso y la salida del modelo. Siendo utilizado el error para ajustar los parámetros del modelo. (Nelles, 2001)

Figura 8

Esquema de identificación de sistemas



Nota: Gráfico extraído de (Nelles, 2001)

Para la correcta identificación de los sistemas no lineales se aconseja seguir los siguientes pasos:

- Escoger las entradas del modelo

Usualmente se trata de una etapa de prueba y error más conocimiento previo. Los procedimientos que se suelen seguir es usar todas las entradas disponibles, la combinación de diferentes entradas, selección de entradas de manera supervisada y no supervisada.

- Escoger las señales de excitación

Deben ser escogidas de acuerdo al tipo de proceso y el propósito que se pretende con el modelo. Las mediciones que se realizan en el modelamiento son la principal fuente de información que se tiene, por lo que este paso dicta la calidad del modelo que se consigue.

- Escoger la arquitectura del modelo

Se consideran varios criterios como el tipo de problema, es decir, si es clasificación, aproximación de sistemas estáticos o identificación de sistemas dinámicos, el uso pretendido

(simulación, optimización, control o detección de errores), dimensión del problema (número de entradas y salidas relevantes), tiempos de entrenamiento y evaluación. (Nelles, 2001)

- Validación del modelo

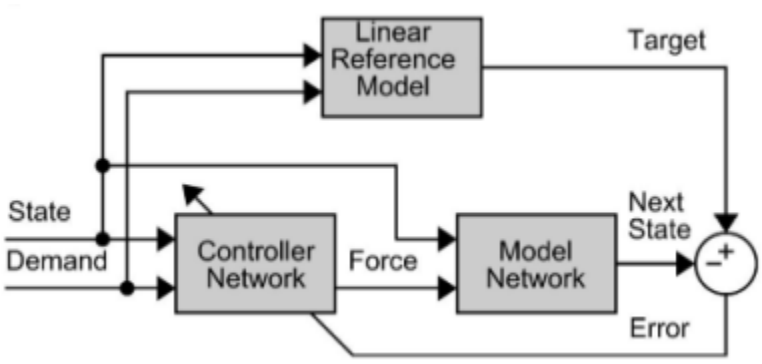
Evalúa si los pasos anteriormente realizados fueron ejecutados con éxito o no. En la mayoría de casos la obtención del modelo no es el paso final, sino que sirve como base para futuros pasos como el diseño de controladores o un sistema de detección de errores.

Control neuronal por modelo de referencia

Este tipo de control hace uso de dos redes neuronales, una red es destinada para el controlador y la otra red para el modelo de la planta. Una vez identificada la red de la planta, se realiza el entrenamiento del controlador para que la salida de la planta siga la salida del modelo de referencia. Para el entrenamiento de la red no se deben cambiar los pesos de la red ya identificada, los que se deben cambiar so los pesos de la red del controlador mediante la retro propagación. El esquema común de este controlador se encuentra en la Figura 9.

Figura 9

Esquema del control por modelo de referencia



Control neuronal inverso

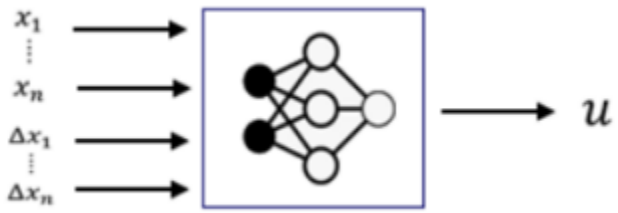
Este controlador pretende cancelar la dinámica de la planta al colocar un elemento en cascada, este elemento es la red neuronal que en este caso será la aproximación matemática al inverso de la planta. Así se consigue que la salida sea lo más parecida a la referencia.

(Toro, Garzón, & López, 2009)

Los insumos de esta red son los estados y sus variaciones, mientras que la señal de control es la salida de este tipo de red, el esquema de esta red se presenta en la Figura 10.

Figura 10

Esquema del control por modelo inverso



Proceso de simulación in-the-loop

El testeo in-the-loop es principalmente usado en el desarrollo del software del controlador para sistemas embebidos, entendiéndose como sistemas embebidos a un procesador involucrados en contexto técnico. Estos sistemas cubren un gran número de aplicaciones de todas las áreas desde vida cotidiana, tales como máquinas de lavar, teléfonos inteligentes, sistemas de control, entre otros.

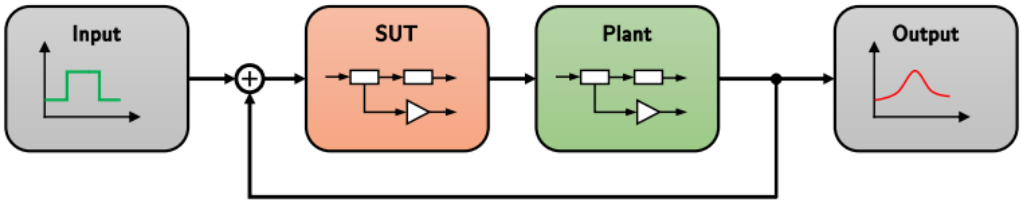
Para las pruebas que se realizan sobre un sistema embebido, se puede seguir el procedimiento más simple, donde se colocan estímulos como entradas del sistema aislado y las

salidas son monitoreadas a fin de ser evaluadas, esta estrategia también es llamada como simulación de un sentido.

Por lo contrario, en una simulación retroalimentada o también llamada simulación de lazo cerrado, el sistema bajo prueba (SUT), también conocido como el controlador, se encuentra acoplado a la planta. Después de un desarrollo inicial del sistema embebido y del modelo de la planta, este procedimiento es el siguiente paso para verificar el diseño realizado. Esta estrategia se puede ejecutar si la complejidad del controlador y sus características puedan ser limitadas sin comprometer su precisión requerida. En la Figura 11 se muestra la configuración de la simulación en lazo cerrado. (Wittmann, 2020)

Figura 11

Esquema de la simulación en lazo cerrado

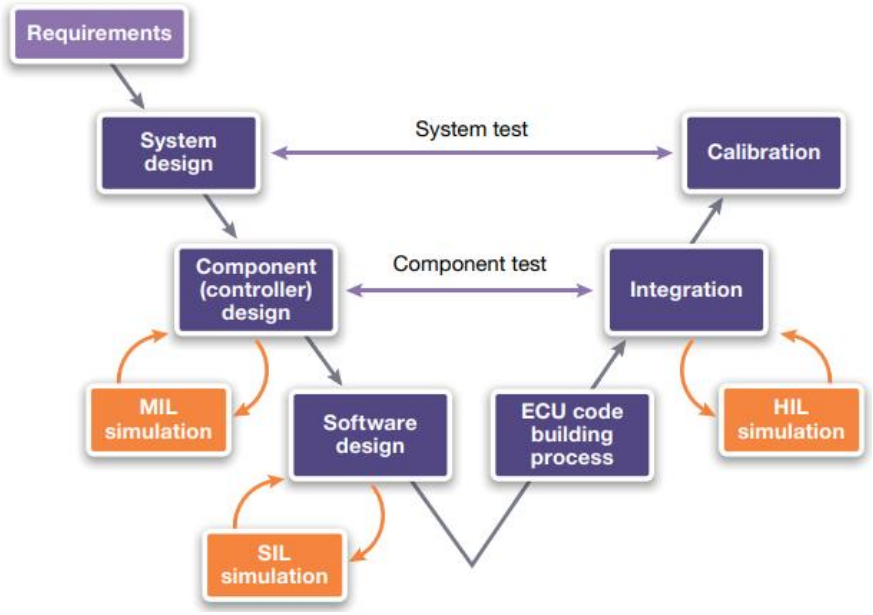


Nota: Gráfico extraído de (Wittmann, 2020)

El proceso de simulación in-the-loop completo está compuesto de varios pasos que se ejecutan de manera cronológica como se muestra en la Figura 12. El primer paso es el de Model in-the-Loop (MIL), seguido por Software-in-the-Loop (SIL), posteriormente como parte del ciclo se realiza la etapa de Hardware-in-the-Loop (HIL). Ocasionalmente se realiza la etapa de Processor-in-the-Loop (PIL) después de SIL, pero no suele ser común. (Reyes, 2014)

Figura 12

Esquema del proceso de simulación in-the-loop



Nota: Gráfico extraído de (Reyes, 2014)

Model-in-the-Loop

Las primeras etapas en la caracterización de procesos es determinar las leyes de modelos físicos y matemáticos de todos los componentes involucrados en el proceso. Los prerrequisitos para esta etapa son contar con modelos para el controlador y la planta, adicionalmente se debe tener un entorno de simulación en el que puedan ser ejecutados. El objetivo de esta etapa será testear la arquitectura del modelo, obtener señales de referencia y verificar el desempeño de los algoritmos desarrollados.

Software-in-the-Loop

Esta etapa es ejecutada para validar la implementación de la ley de control. El controlador que en la anterior fase era un bloque ahora pasa a ser código (generalmente en C), este código se deriva del modelo y puede ser hecho a mano o generado de manera

automatizada. Tal como en la etapa MIL, existe un modelo virtual de la planta que puede ser ejecutado en la misma máquina que el software del controlador. El software es generado para una arquitectura determinada, el escalamiento necesario forma parte del software.

Processor-in-the-Loop

Es ejecutado para superar limitaciones de SIL. Esta etapa se centra en la implementación del algoritmo de control, pero en esta ocasión el código en C es compilado para una arquitectura de procesador determinada. La ejecución del código del controlador se da en sincronización con la simulación de la planta física. Aquí es donde se considera las localidades de memoria y el tiempo de ejecución.

Hardware-in-the-Loop

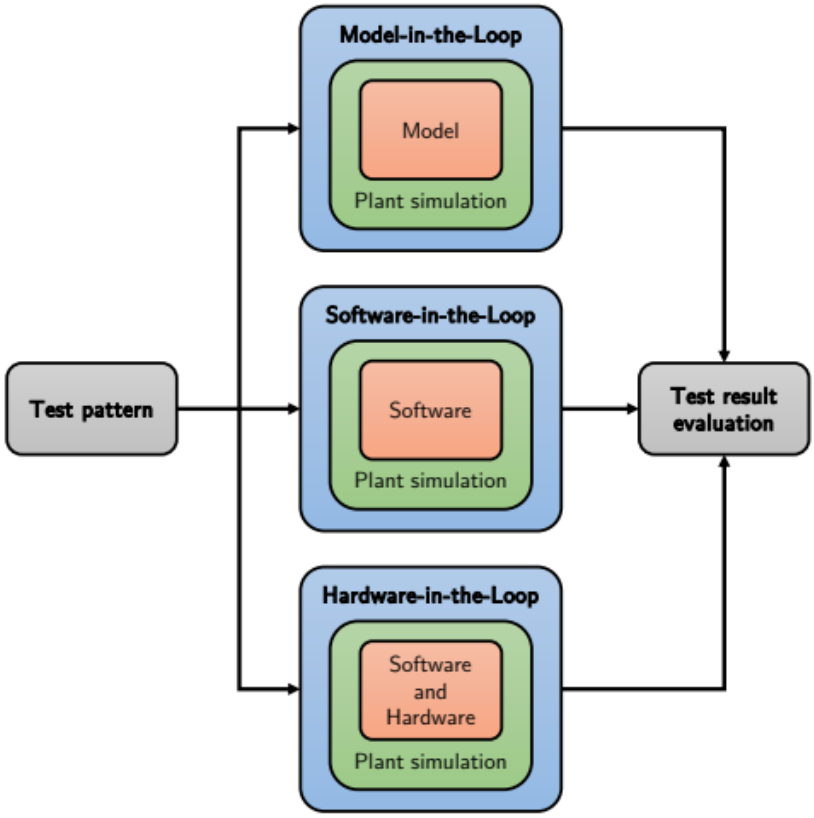
Las últimas pruebas son ejecutadas sobre el software del controlador en la plataforma de hardware determinada. La diferencia primordial con anteriores etapas es que los modelos deben tener un comportamiento en tiempo real, significando que la respuesta de los modelos sea garantizada en intervalos de tiempo especificados. De esta manera se asegura pruebas realistas para que la comunicación con el controlador se de en un entorno real. Esta etapa tiene como objetivo la correcta operación del software embebido en la plataforma de hardware y sus componentes.

MIL y SIL son generalmente ejecutadas en las tempranas etapas del desarrollo, se centran en buscar errores funcionales y optimización del diseño. En cambio, para encontrar problemas relacionados con el hardware real que será usado, se ejecutan las pruebas de HIL. En estas pruebas se determinan problemas con las interfaces del hardware externo y errores que involucren a los sensores y actuadores. (Wittmann, 2020)

Parte del proceso de simulación es que se diseñen pruebas/testeos que puedan ser reusados en tantas etapas posibles. Esta estrategia permite usar las mismas señales de entrada para todas las fases. Adicionalmente, los resultados de los testeos en las etapas individuales pueden ser comparados con cada una y determinar desviaciones y eliminar posibles errores. Esta técnica se contempla en la Figura 13, se debe notar que la representación física de las señales de cada etapa es diferente.

Figura 13

Estrategia de simulación integrada de las etapas MIL, SIL y HIL



Nota: Gráfico extraído de (Wittmann, 2020)

Estado del arte

Actualmente no se han extendido las herramientas como funciones, librerías o bloques de programación que permitan la implementación de estrategias de control inteligente en las principales plataformas de hardware. La metodología que se usa para implementar un controlador difuso suele ser empezar por su diseño en Matlab, y posteriormente desarrollar desde el inicio en el entorno de desarrollo integrado de determinado PLC, un bloque de función con el uso de lenguaje estructurado de control (SCL). Esta propuesta por (Portillo & Ordoñez, 2020) obtuvo como resultados la satisfactoria implementación de un controlador difuso en un PLC Siemens S7-1200 que controla la frecuencia de un motor, sin embargo, se debe señalar que el procedimiento es redundante y se realiza el diseño dos veces en diferentes softwares.

En cuanto a redes neuronales se refiere, recientemente Siemens ha introducido el módulo SIMATIC S7-1500 TM NPU (unidad de procesamiento neuronal) que permite la integración de algoritmos de inteligencia artificial y la lógica convencional de los PLCs. Es necesario tomar en cuenta que solo se puede utilizar en la línea de PLCs S7-1500 y se trata de una adquisición separada del sistema de control. (SIEMENS, s.f.)

Otro enfoque propuesto por (Ponce & Ramírez, 2010) establece la introducción de los algoritmos inteligentes en la plataforma de software LabVIEW de National Instruments. Esta referencia provee una guía para el diseño de controladores difusos, redes neuronales e incluso algoritmos genéticos desarrollados en su totalidad en LabVIEW, sin embargo, la programación gráfica que es necesaria realizar para obtener un controlador inteligente es demasiado extensible y un proceso que podría ser considerado engorroso por ser propenso a errores.

De acuerdo a lo señalado se presenta la Tabla 2, donde se compara las prestaciones y otras características de los softwares: TIA Portal de Siemens, LabVIEW de National Instruments y Matlab/Simulink.

Tabla 2

Cuadro comparativo de las herramientas de TIA Portal, LabView y Matlab/Simulink

	TIA Portal	***LabVIEW***	***Matlab/Simulink***
Controlador Difuso	Bloque de funciones creadas por el usuario	PID and Fuzzy Logic Toolkit	Fuzzy Logic Designer Neuro-Fuzzy Designer
Redes Neuronales	Por medio del módulo S7-1500 TM NPU	Neural Network Toolkit Deep Learning Toolkit	Neural Net Time Series Neural Net Fitting Neural Net Clustering
Tiempo de desarrollo del algoritmo	Lento	Lento	Rápido
Licencias	Ilimitada Anual Prueba de 21 días	Tres tipos: Base, Completo y Profesional Prueba de 45 días	Perpetua Anual Prueba de 30 días

Si bien el desarrollo de las estrategias de control es posible en dos de las plataformas de software más populares como lo son TIA Portal y LabVIEW, es un procedimiento largo en el cual el desarrollador no cuenta con las facilidades que si se presentan en Matlab. A continuación, se

presentan las herramientas propias de Matlab/Simulink para extender los controladores a plataformas de hardware.

Herramientas orientadas hacia la generación de código

Matlab/Simulink cuenta con varias herramientas para la generación de código, entre estas se tiene: PLC Code, C/C++ Code, Hardware Description Language (HDL) Code. Dependiendo de la plataforma de hardware seleccionada se puede generar código en C/C++, HDL y código bajo el estándar IEC 61131-3 (texto estructurado o diagrama Ladder). Tener este soporte de implementación en diferentes lenguajes y la selección de los entornos de desarrollo integrados de los PLCs industriales más comunes, como se observa en la Tabla 3, permite a los ingenieros diseñar y testear el software independientemente de la plataforma de hardware. Otros de los beneficios de este enfoque son que se reduce el tiempo y esfuerzo en la creación de código de manera manual, además, se reducen errores por medio de la generación automática. (MathWorks, 2021)

Tabla 3

Soporte de generación de código según la plataforma de PLC provisto por MathWorks

Compañía	***IDE***	***IEC 61131-3***	***C/C++***
ABB	Automation Studio™	✓	✓
Omron	Sysmac® Studio	✓	
Rockwell Automation	RSLogix™/Studio 5000	✓	
Schneider Electric	Unity Pro	✓	
Siemens	TIA Portal/STEP® 7	✓	✓

Nota: Extraído de (MathWorks, 2021)

Capítulo III

Diseño de estrategias de control y evaluación en etapa MIL

A continuación, se presenta el diseño de los controladores inteligentes que actúan sobre los procesos no lineales representativos, se especifica las condiciones que deben cumplir los controladores y se muestran los resultados de la simulación realizada en Matlab/Simulink.

Caso de estudio actuador no lineal

Una gran fuente de variaciones proviene de los actuadores tales como válvulas que poseen una característica no lineal. El caso de estudio presenta un simple lazo cerrado con un controlador PI, un actuador no lineal y la planta cuya función de transferencia es de tercer orden como se presenta en (Åström & Björn, 2008).

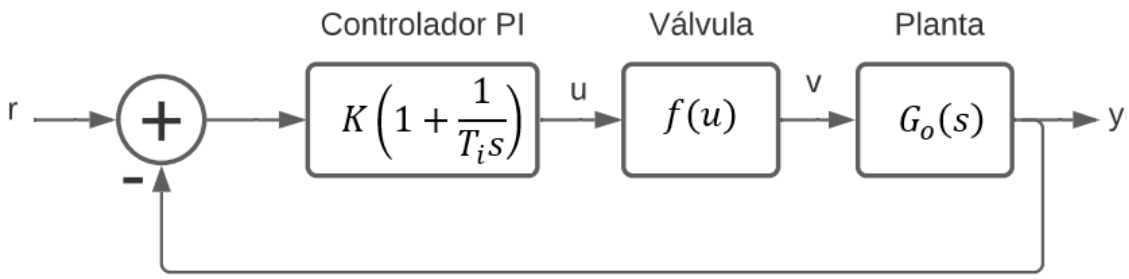
La función característica del actuador es la siguiente:

$$v = f(u) = u^4 \text{ para } u \geq 0$$

De este sistema ya se conocen los valores de sintonización del controlador PI, los cuales son $K = 0.15$, $T_i = 1$ para una función de transferencia $G_o(s) = \frac{1}{(s+1)^3}$. El lazo de control completo del caso de estudio se presenta en la Figura 14.

Figura 14

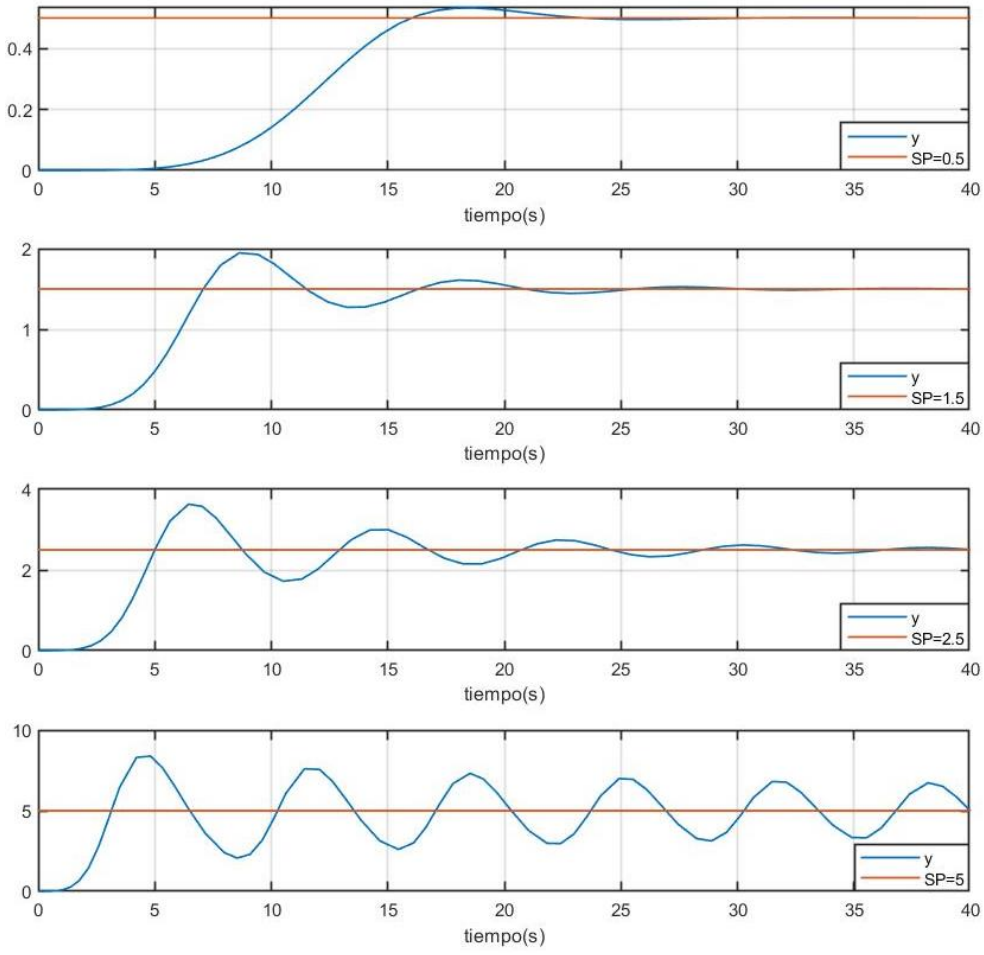
Lazo de control cerrado del sistema compuesto por el actuador no lineal



La linealización del sistema alrededor de un punto de operación demuestra la ganancia incremental de la válvula es su derivada $f'(u)$, por ende, la ganancia de lazo es proporcional a la derivada. El sistema demuestra comportarse correctamente en determinado punto de operación y de manera pobre en otro. Las respuestas a diferentes puntos de operación se muestran en la Figura 15.

Figura 15

Respuesta del sistema del actuador no lineal

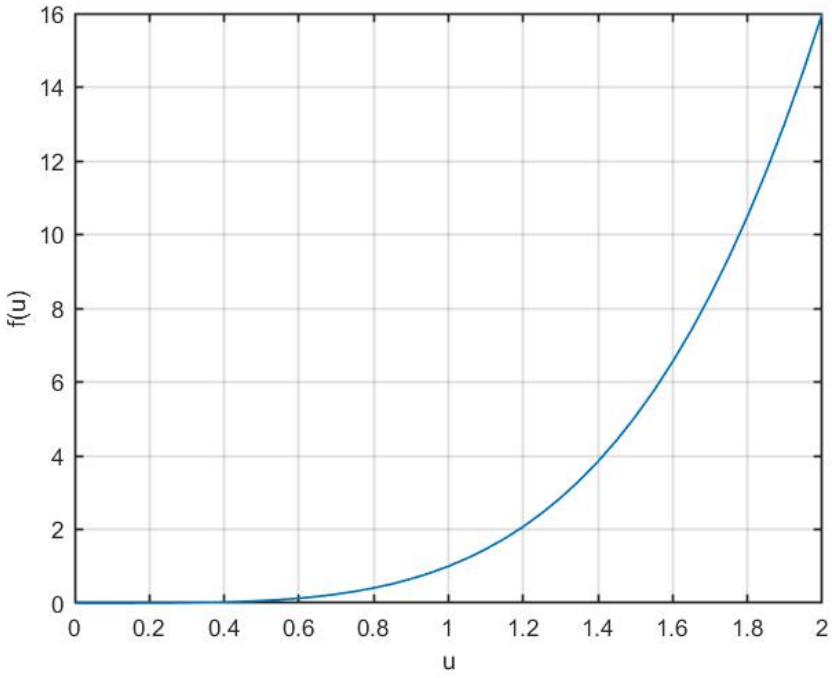


Esto demuestra que el controlador PI ha sido sintonizado para operar bien a valores de setpoint menores. Para valores mayores la respuesta en lazo cerrado se vuelve inestable. Una

manera de solucionar este problema es mediante una aproximación a la función inversa de la no linealidad del actuador. En la Figura 16 se muestra la curva de la función no lineal del actuador.

Figura 16

Característica no lineal del actuador

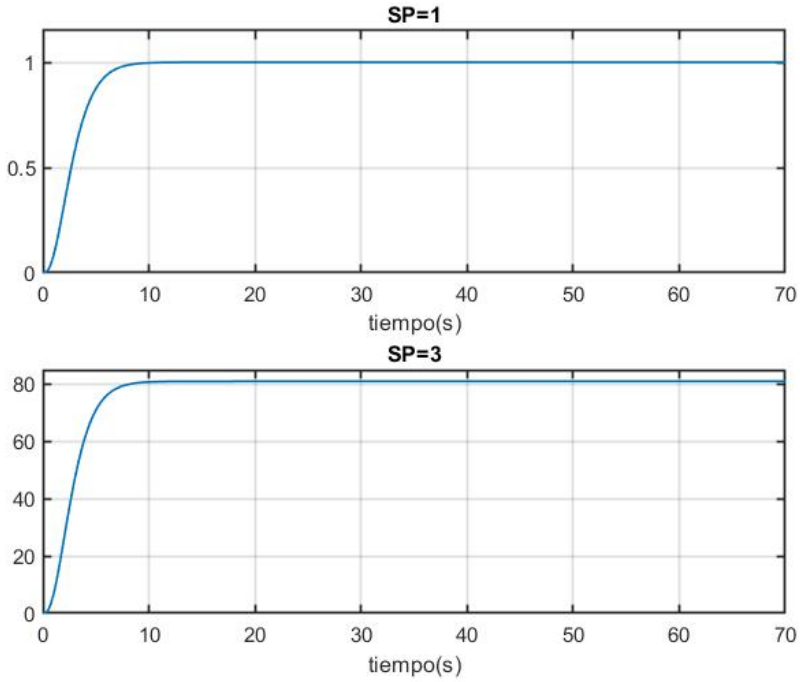


Antes del diseño de los controladores se debe caracterizar a la planta no lineal en lazo abierto, esto proporciona la información del tiempo de establecimiento aproximado que se podría esperar en lazo cerrado con la aplicación de los controladores difusos.

De acuerdo a la Figura 17, la respuesta en lazo abierto presenta un tiempo de establecimiento alrededor de los 10 a 15 segundos, como particularidad se debe destacar que para el SP=3 el actuador, es decir, la válvula se satura rápidamente y converge al valor de 81 (3^4).

Figura 17

Respuesta en lazo abierto de la planta no lineal

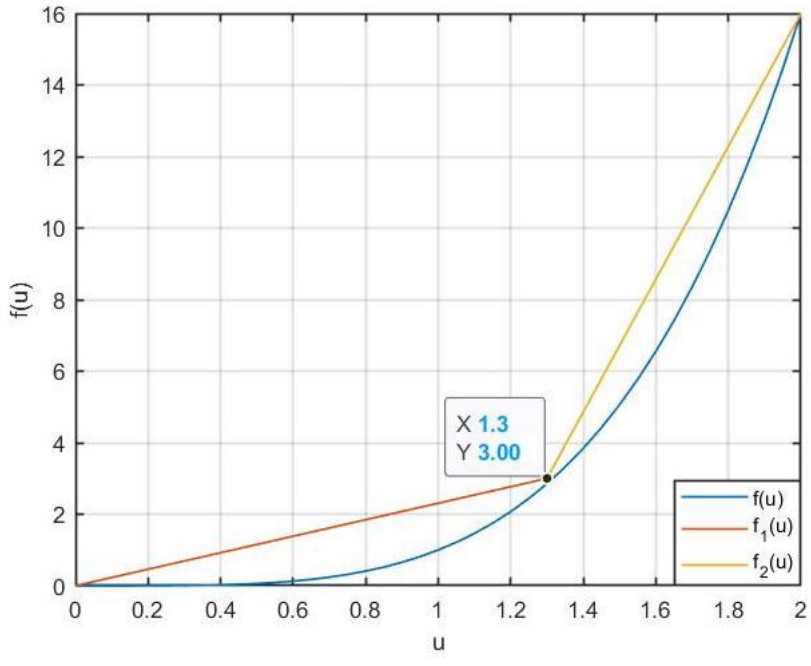


Controlador Difuso Takagi-Sugeno

Primero se debe aproximar la curva no lineal del actuador mediante dos rectas como se muestra en la Figura 18.

Figura 18

Característica no lineal del actuador y su aproximación mediante rectas



Las funciones de estas rectas son las siguientes:

$$f_1(u) = 2.31u$$

$$f_2(u) = 18.57u - 21.14$$

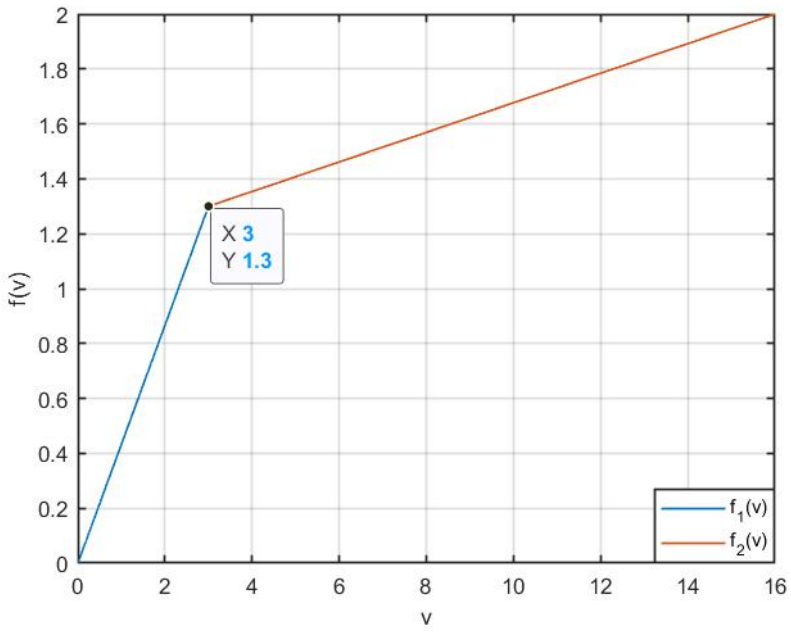
Las funciones inversas de las rectas anteriores son, en la Figura 19 se muestra ambas rectas de las funciones inversas:

$$f_1^{-1}(u) = \frac{1}{2.31}v$$

$$f_2^{-1}(u) = \frac{v+21.14}{18.57}$$

Figura 19

Funciones inversas



De acuerdo a la Figura 19 ya se puede establecer los límites de las funciones de membresía de la entrada, las funciones lineales de salida y las reglas que forman parte del controlador difuso.

Funciones de membresía de entrada

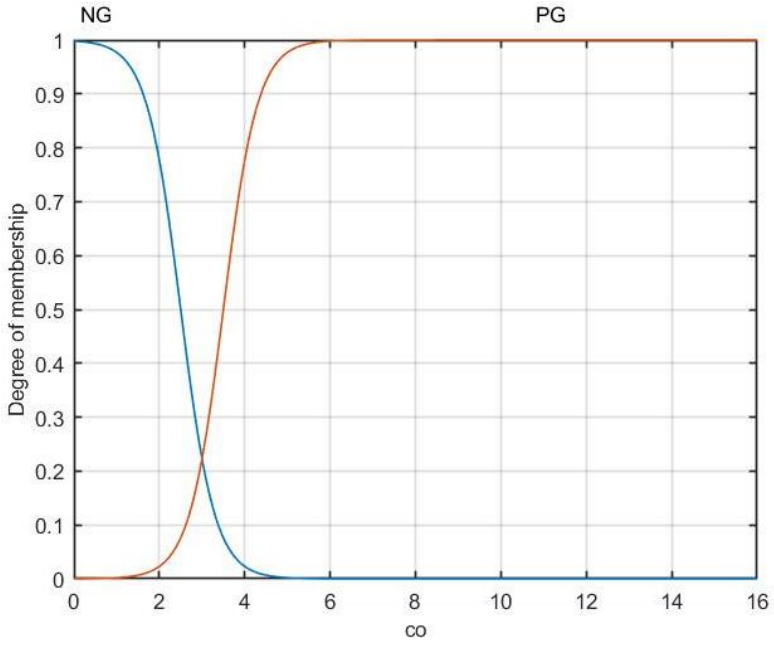
Las funciones de membresía de entrada trabajan en un intervalo de [0 16], puesto que el dominio de ambas funciones inversas es este. Las funciones de membresía son del tipo sigmoidal y de acuerdo a las funciones inversas se determina que sus parámetros son:

$$M(NG) = sigmf(co; -2.5, 2.5)$$

$$M(PG) = sigmf(co; 2.5, 3.5)$$

Figura 20

Funciones de membresía de la entrada



Funciones lineales de salida

Las funciones lineales de salida son las mismas funciones inversas que ya fueron determinadas, su intervalo es de [0 2] puesto que ese el domino de las aproximaciones originales.

Reglas de control

Tan solo se tienen dos reglas de control ya que se tienen dos rectas, estas reglas son:

$$SI\ co\ es\ NG\ ENTONCES\ u\ es\ f_1^{-1}(co)$$

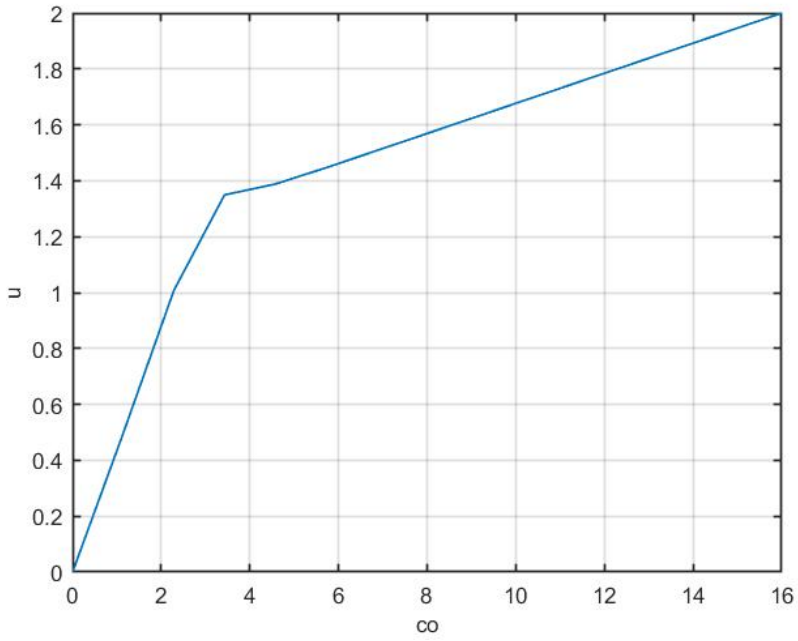
$$SI\ co\ es\ PG\ ENTONCES\ u\ es\ f_2^{-1}(co)$$

Curva de control

De acuerdo a estas especificaciones la curva de control generada se muestra en la Figura 21.

Figura 21

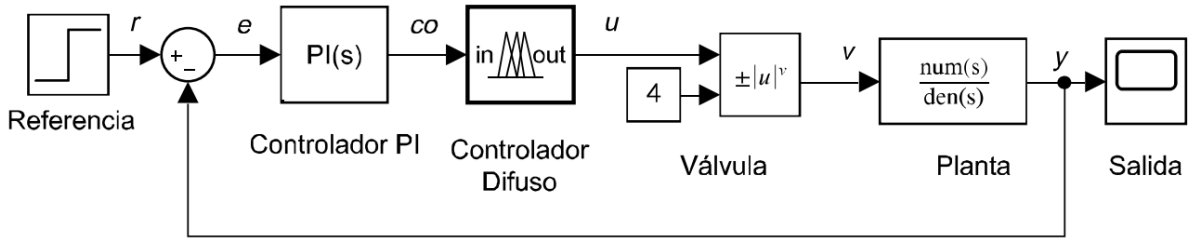
Curva de control por Takagi-Sugeno



Una vez diseñado el controlador se procede a su implementación en la etapa MIL como se muestra en la Figura 22 su esquema de simulación.

Figura 22

Lazo de control cerrado con compensación del actuador no lineal



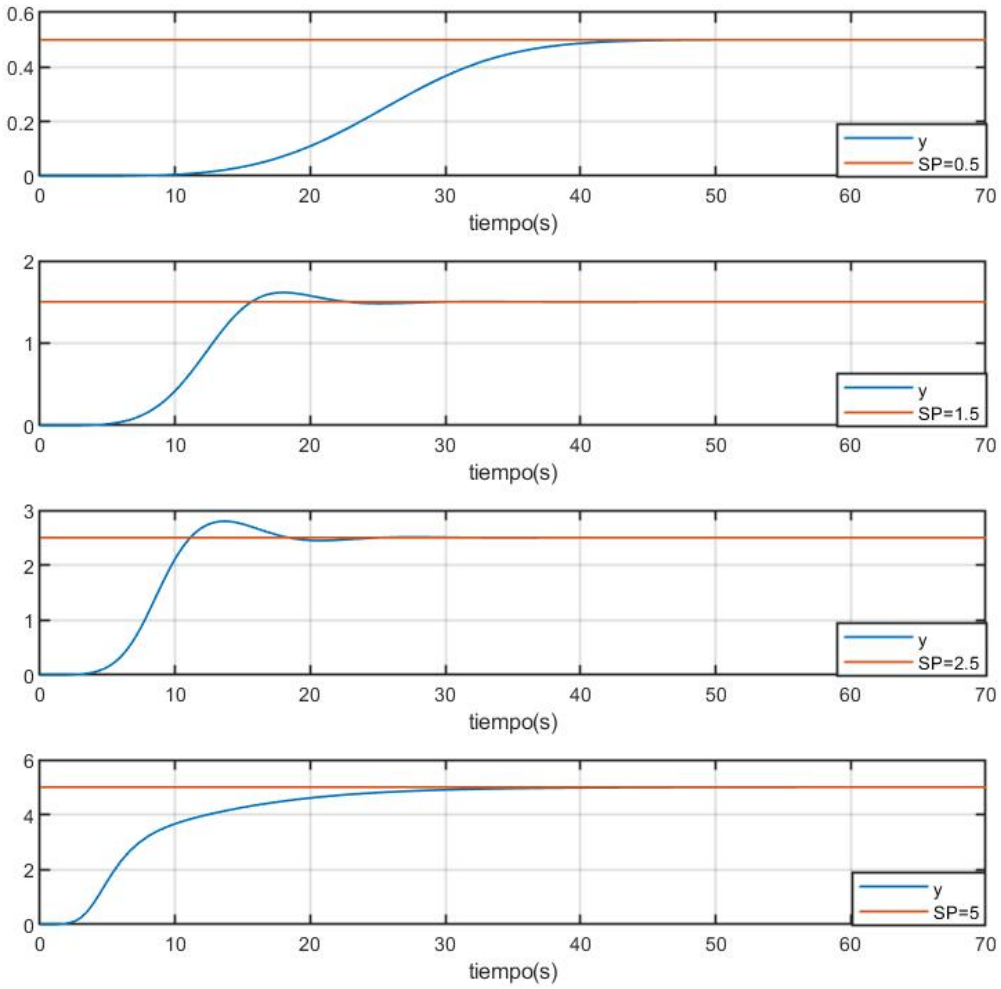
La compensación de la no linealidad del actuador se da mediante la aproximación a la función inversa de la válvula, la salida del controlador PI (co) es filtrada a través del controlador difuso antes de ser aplicada el actuador, esto da la relación matemática:

$$v = f(u) = f\left(\hat{f}^{-1}(co)\right)$$

La función $f\left(\hat{f}^{-1}(co)\right)$ proporciona menor variación en la ganancia que la función original del actuador. De darse el caso que la aproximación de la inversa ($\hat{f}^{-1}$) sea la inversa exacta, entonces $v = co$.

Figura 23

Respuesta del sistema del actuador no lineal con corrección de la no linealidad mediante Takagi-Sugeno



La Figura 23 muestra las respuestas del sistema a cuatro puntos de referencia, implementando la aproximación a la inversa de la función característica del actuador. Si se compara esta respuesta con las que no incluía la corrección de la función del actuador, se nota una mejora considerable en el desempeño del sistema a lazo cerrado.

Controlador Mamdani

Funciones de membresía de entrada

Las funciones de membresía de entrada trabajan en el mismo intervalo de [0 16]. En este caso se usan varias funciones de membresía, en total cinco, de esta manera se puede aproximar mejor las pendientes de las funciones inversas.

$$M(NG) = trapmf(co; -4.133, -4.133, 0.1867, 1.067)$$

$$M(NP) = trimf(co; -0.0165, 1.07, 2.11)$$

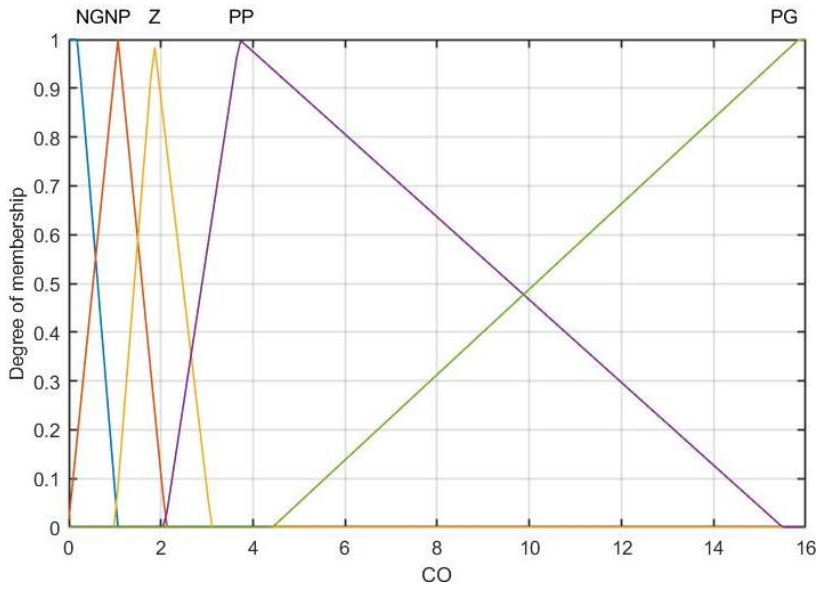
$$M(Z) = trimf(co; 0.9964, 1.846, 3.096)$$

$$M(PP) = trimf(co; 2.08, 3.704, 15.5)$$

$$M(PG) = trapmf(co; 4.42, 15.85, 16.3, 16.6)$$

Figura 24

Funciones de membresía de la entrada del controlador Mamdani



Funciones de membresía de salida

Las funciones de membresía de salida trabajan en el intervalo de [0 2]. Igualmente se usan cinco funciones de membresía para aproximar las pendientes de las funciones inversas.

$$M(NG) = trapmf(u; -0.493, -0.493, 0.056, 0.331)$$

$$M(NP) = trimf(u; 0.342, 0.653, 0.942)$$

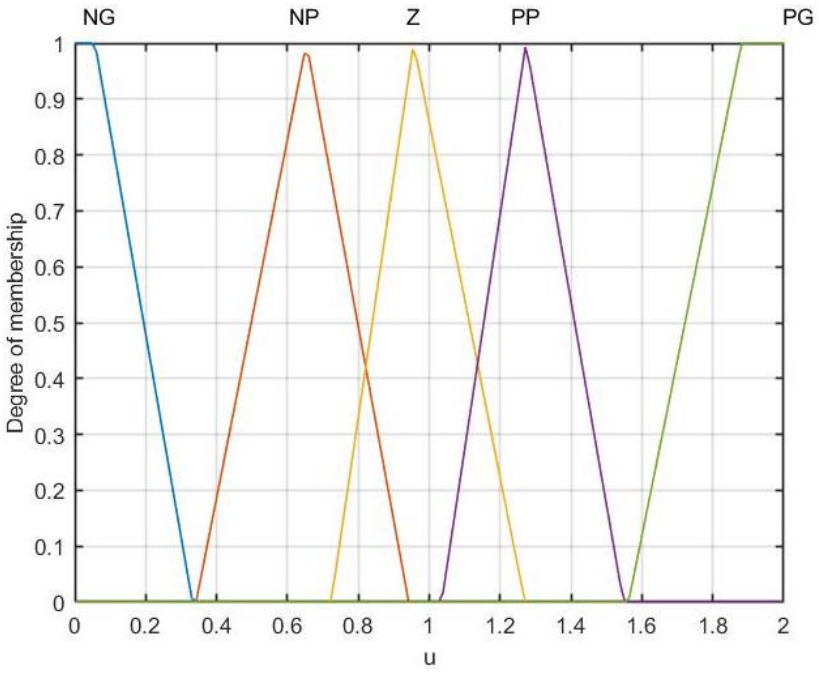
$$M(Z) = trimf(u; 0.723, 0.955, 1.27)$$

$$M(PP) = trimf(u; 1.035, 1.273, 1.548)$$

$$M(PG) = trapmf(u; 1.562, 1.881, 2.2, 2.2)$$

Figura 25

Funciones de membresía de la salida del controlador Mamdani



Reglas de control

En este caso se tienen cinco reglas de control, puesto que cada función de membresía de entrada esta correspondida su función de membresía de salida.

$$SI\ co\ es\ NG\ ENTONCES\ u\ es\ NG$$

$$SI\ co\ es\ NP\ ENTONCES\ u\ es\ NP$$

$$SI\ co\ es\ Z\ ENTONCES\ u\ es\ Z$$

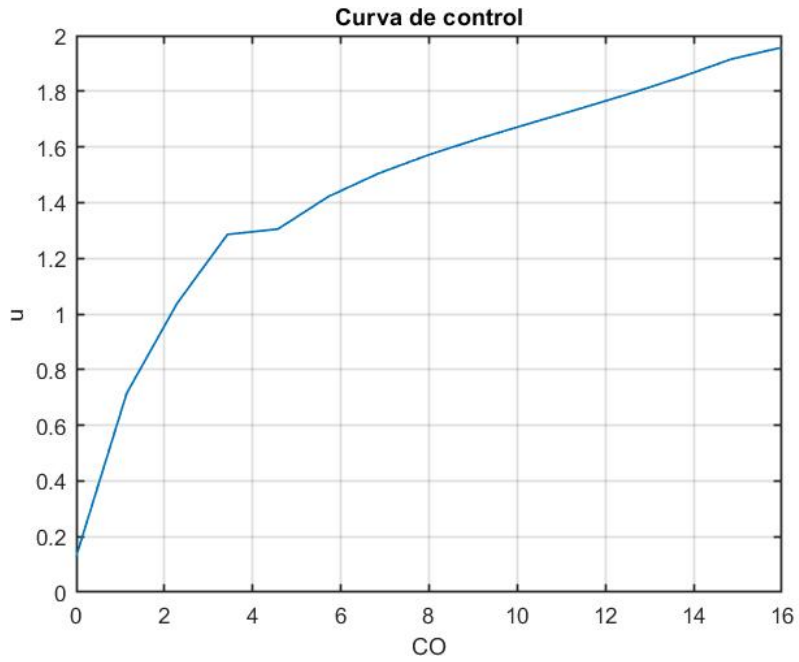
$$SI\ co\ es\ PP\ ENTONCES\ u\ es\ PP$$

$$SI\ co\ es\ PG\ ENTONCES\ u\ es\ PG$$

De acuerdo al proceso de defusificación, la curva de control generada se muestra en la Figura 26.

Figura 26

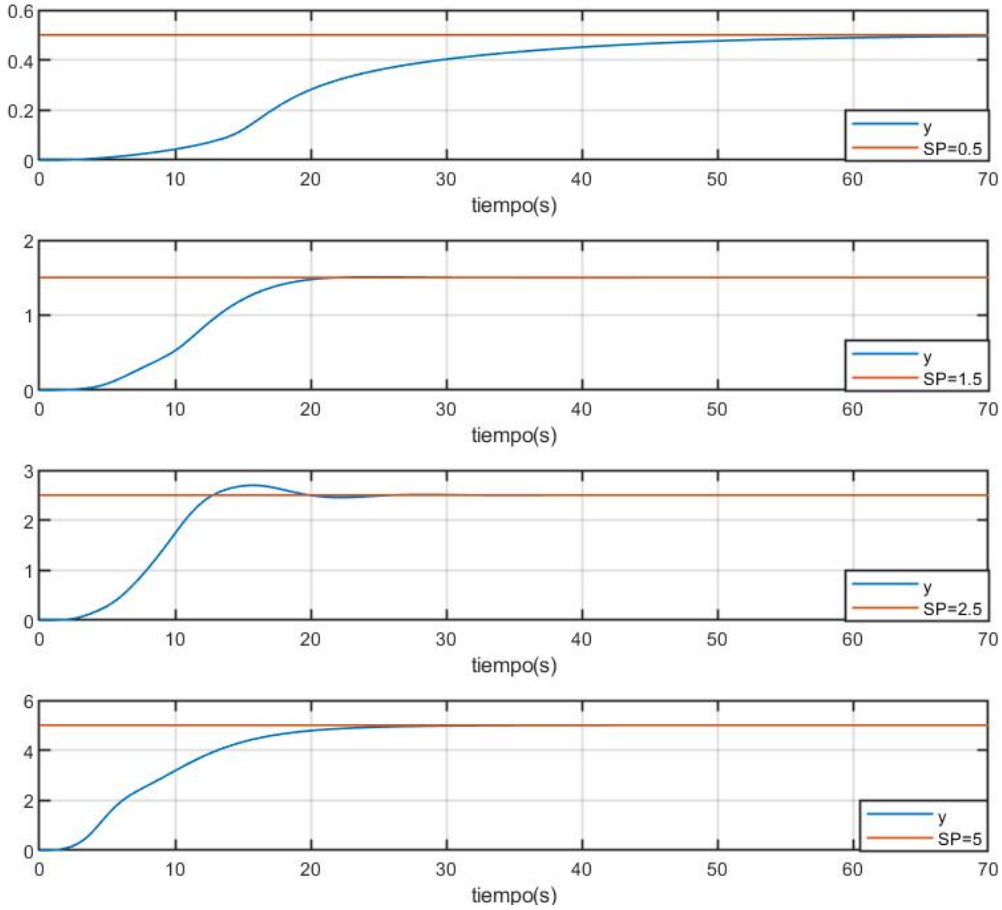
Curva de control por Mamdani



Diseñado el controlador se procede a su implementación en la etapa MIL, con el mismo esquema de simulación que se propuso en el controlador Sugeno.

Figura 27

Respuesta del sistema del actuador no lineal con corrección de la no linealidad mediante Mamdani



En la Figura 27 se puede apreciar un desempeño similar al conseguido con el controlador Takagi-Sugeno, efectivamente se cancelan las no linealidades y se comprueba que para setpoints mayores a 2 la respuesta del sistema es estable. Como puntualidades se debe señalar que para un setpoint de 0.5 se tarda aproximadamente 60 segundos en alcanzar la referencia, esto se debe a que la aproximación a la función inversa no tiene la misma pendiente que si la tiene en el controlador Sugeno. Para el resto de setpoints el desempeño es análogo al del anterior controlador, sin embargo, puesto que la aproximación a las funciones inversas se da con mucha más facilidad con el controlador Sugeno, se ve favorecido este último.

En cuanto a la implementación en código de este controlador se debe considerar que no solo es el controlador difuso, sino también el controlador PI que ya estaba definido previamente. Por lo tanto, se procede con la implementación en código del controlador PI.

Controlador PI digital

Primero se obtiene la función de transferencia continua del controlador, esta es:

$$G_c(s) = 0.15\left(1 + \frac{1}{s}\right)$$

$$G_c(s) = \frac{0.15s + 0.15}{s}$$

Dado que el tiempo de constante de la planta $G_o(s) = \frac{1}{(s+1)^3}$, es de $\tau = 1\ [s]$, el periodo de muestreo debe ser al menos la décima parte, es decir, $T_s = 0.1\ [s]$. Esta planta es discretizada mediante el retenedor de orden cero y con el periodo de muestreo descrito:

$$G_c(z) = \frac{CO(z)}{E(z)} = \frac{0.15 - 0.135z^{-1}}{1 - z^{-1}}$$

De esta función de transferencia se obtiene la ecuación de diferencias que es implementada como código del controlador PI discreto:

$$CO(z) = 0.15E(z) - 0.135E(z-1) + CO(z-1)$$

Controlador difuso tipo Takagi-Sugeno

La salida del controlador PI discreto pasa a ser la entrada del controlador difuso, el código de este se realiza con los principios de fusificación (funciones de membresía) y defusificación (funciones lineales de salida y combinación lineal ponderada) del controlador Takagi-Sugeno, ambos controladores se unen en uno solo que contrarreste las no linealidades de la planta, el reto es que funcionen en armonía juntos y que no se desajusten al valor de referencia de los setpoints, el código conjunto es el siguiente:

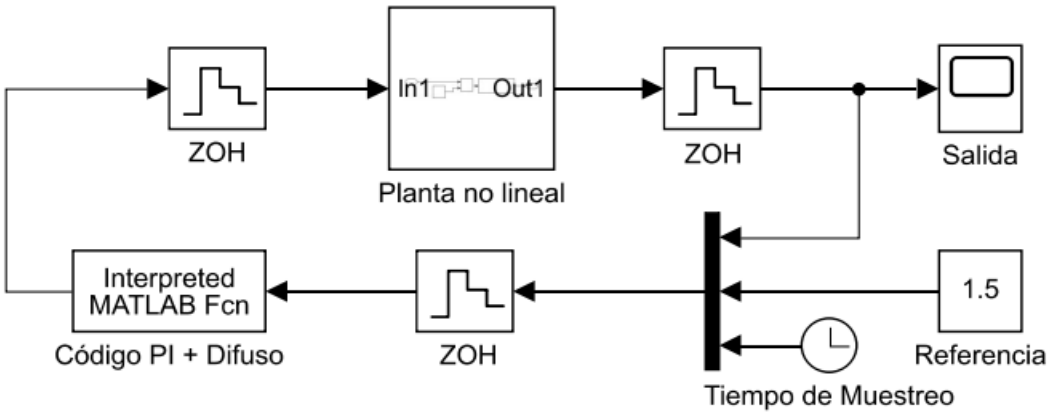
```
function v=CasoEstudio1(Z);
global ek1 uk1
% Entradas Función
sp=Z(1);
yk=Z(2);
t=Z(3);
% Controlador PI digital
s0=0.15; s1=-0.135;
% Señal de error retroalimentada
ek=sp-yk;
% Valores que toma la salida del controlador PI
CO=0:0.1:16
a=10;
% Funciones de membresía de entrada
NG=sigmf(CO,[-2.5*a 2.5]);
PG=sigmf(CO,[2.5*a 3.5]);
N=length(CO);

if t==0
    u=s0*ek;
    v=u;
    ek1=ek;
    uk1=u;
end
if t>0
    u=uk1+s0*ek+s1*ek1
    uk1=u;
    ek1=ek;
    % Inferencia de Sugeno - curva de control
    uaux=abs(u);
        % Obtención del índice según el valor de la
salida del controlador
```

```
        % PI
        uaux=round(uaux,1)
        i=uaux*10+1;
        % Valores que toman las funciones de
membresía
        W=[NG(i) PG(i)];
        % Evaluación de la 1ra función según la
salida
        % del controlador PI
        f1=1/2.31*abs(u);
        % Evaluación de la 2da función según la
salida
        % del controlador PI
        f2=1/18.57*abs(u)+21.14/18.57;
        % Proceso de defusificación mediante
combinación lineal ponderada
        v=(W(1)*f1+W(2)*f2)/sum(W);
end
```

El esquema de simulación en el cual se testeo este código se muestra en la Figura 28:

Figura 28

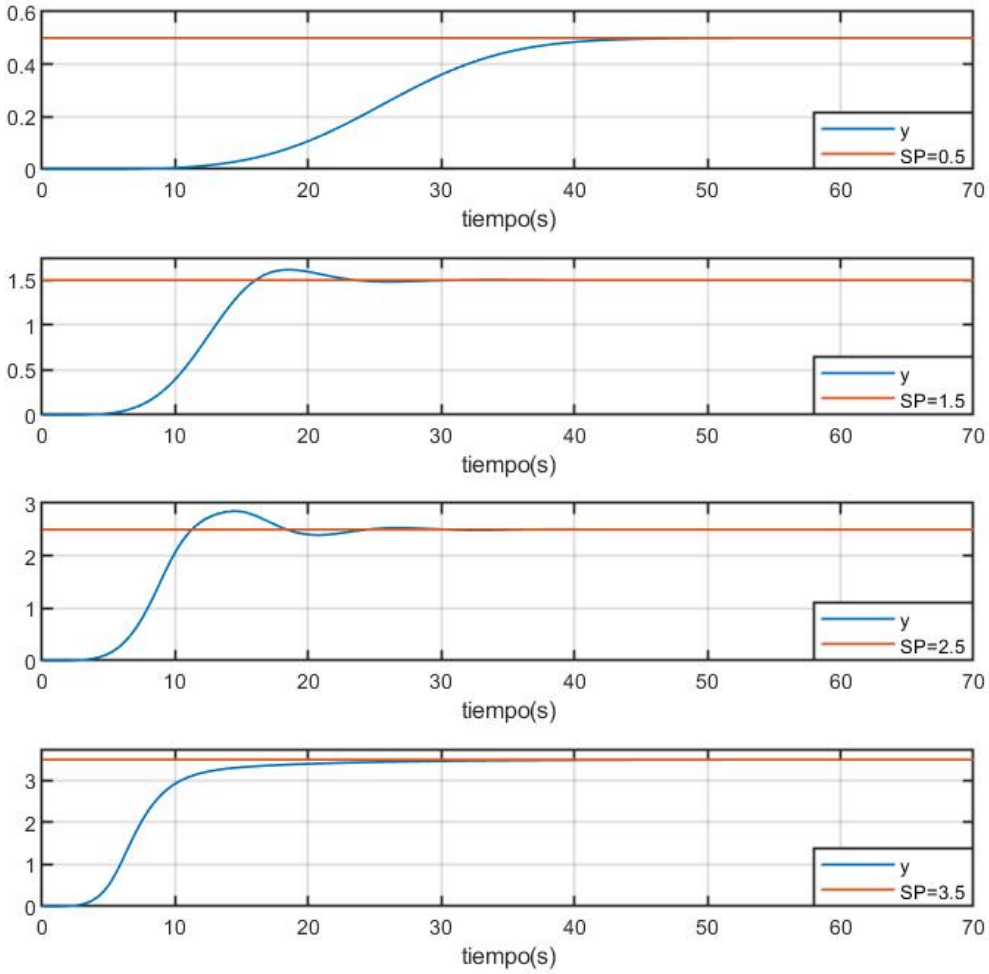
Esquema de simulación del primer caso de estudio, controlador por código



La respuesta del controlador realizado por código a los diferentes setpoints se visualiza en la Figura 29, el código no presenta problemas para ajustarse al rango de entrada de la referencia y realiza el seguimiento a la misma.

Figura 29

Respuesta del sistema del actuador no lineal con corrección de la no linealidad mediante Takagi-Sugeno, controlador por código



Caso de estudio reactor de tanque agitado

El nombre completo de este modelo es el de reactor químico tipo tanque con agitación continua (CSTR), los reactores químicos son el lugar físico donde ocurren las reacciones químicas, las reacciones químicas se tratan de la distribución de átomos de determinada molécula (reactante) para formar otras (productos). El uso de los reactores se da para garantizar el flujo de los reactantes en su interior, conseguir la extensión deseada de la reacción o tiempo de mezcla, y permiten condiciones de presión, temperatura y composición de tal manera que la reacción se produzca en el grado y velocidad deseada.

Este modelo es ampliamente usado en la ingeniería química, ya que muestra ventajas respecto a la uniformidad de presión, composición y temperatura. El modelo CSTR presenta características de no linealidad, tiempo elevado de retardo y la manera que interactúan entradas y salidas hacen complejo el diseño del sistema de control.

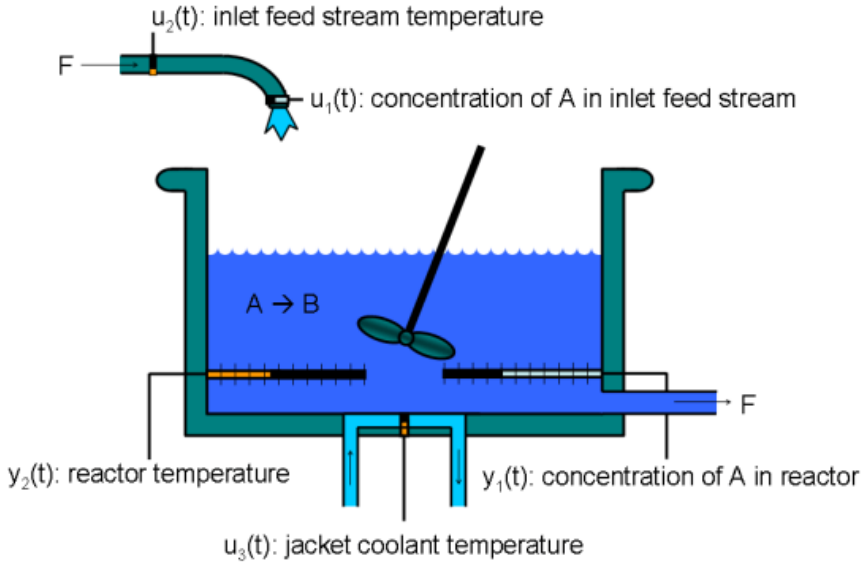
(Peña, Pérez, Miranda, & Sánchez, 2008)

Descripción del proceso

Este modelo se rige por dos etapas, la primera es dedicada a la formación del producto y la segunda de retiro del calor del reactor mediante una chaqueta. Al tratarse de un proceso continuo siempre habrá entrada de reactante y salida del producto del sistema a una velocidad volumétrica constante, la densidad del fluido también permanece constante. CSTR parte de la consideración que el sistema se encuentra en operación, por lo que las condiciones de inicio y parada no forman parte del análisis. Otra consideración es que se da una mezcla perfecta, mediante una reacción exotérmica e irreversible, $A \rightarrow B$, A, es el reactante, que es convertido en B, el producto. En la Figura 30 se presenta el diagrama del modelo CSTR. (MathWorks, s.f.)

Figura 30

Representación del modelo CSTR



Nota: Gráfico extraído de (MathWorks, s.f.)

Las entradas del modelo CSTR son las siguientes:

- $u_1 \rightarrow C_{Af}$ es la concentración del reactante A en el suministro de flujo $[kmol/m^3]$
- $u_2 \rightarrow T_f$ es la temperatura del suministro de flujo $[°K]$
- $u_3 \rightarrow T_c$ es la temperatura del refrigerante $[°K]$

Las entradas C_{Af} y T_f se asumen como perturbaciones no medibles constantes, mientras que la entrada T_c es la variable de control. A partir de estas consideraciones, se asume que la mezcla en el reactor ya alcanzó un determinado nivel de temperatura para el cual la reacción genera calor, a esto se conoce como reacción exotérmica. Cuando se empieza a liberar calor, se procede a retirarlo por la apertura de una válvula de agua fría de la chaqueta, así se mantiene la temperatura del reactor entre los valores de operación fijados para el proceso.

Mientras que las salidas del modelo son:

- $y_1 \rightarrow C_A$ es la concentración del reactante A en el reactor $[kmol/m^3]$
- $y_2 \rightarrow T$ es la temperatura en el reactor $[°K]$

El modelo CSTR se consigue aplicando los principios de balance de masa y energía. La tasa de cambio de la concentración del reactante A en el reactor por unidad de tiempo se expresa en la siguiente ecuación diferencial:

$$\frac{dC_A}{dt} = \frac{F}{V}\big(C_{Af}(t) - C_A(t)\big) - r(t)$$

Donde:

- V es el volumen del reactor $[m^3]$
- F es el flujo volumétrico $[m^3/h]$
- $r(t)$ es la taza de reacción por unidad de volumen, de acuerdo a la ecuación de Arrehenius:

$$r(t) = k_0 e^{-\frac{E}{RT(t)}} C_A(t)$$

- E es la energía de activación
- R es la constante de gases ideales de Boltzmann
- T es la temperatura del reactor
- k_0 es el factor pre-exponencial de Arrehenius $[1/h]$

De manera similar aplicando los principios de balance de masa y energía, y asumiendo el volumen del reactor como constante, se obtiene la ecuación diferencial que expresa la taza de cambio de la temperatura por unidad de tiempo.

$$\frac{dT(t)}{dt} = \frac{F}{V}\big(T_f(t) - T(t)\big) - \frac{\Delta H}{\rho C_p} r(t) - \frac{UA}{\rho C_p V}\big(T(t) - T_c(t)\big)$$

El primer y tercer término de la ecuación diferencial describen los cambios producidos por la temperatura del flujo de entrada T_f y de la temperatura del refrigerante T_c, respectivamente. El segundo término representa la influencia sobre la temperatura del reactor producida por la reacción química del recipiente.

De la ecuación anterior:

- ΔH es el calor producido por la reacción $[kcal/kmol]$
- C_p es el coeficiente de capacidad calorífica
- ρ es la densidad del material en el modelo CSTR $[kg/m^3]$
- U es el coeficiente de transferencia calorífica
- A es el área de sección transversal $[m^2]$ del reactor

En la Tabla 4 se presentan los valores que toman las constantes y variables que forman parte del modelo CSTR con el que se trabajará.

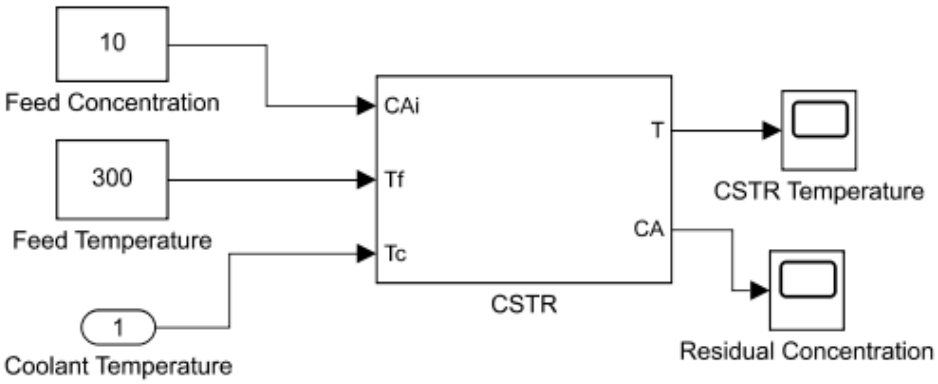
Tabla 4

Parámetros del modelo no lineal CSTR

Parámetro	***Valor***	***Unidad***	***Descripción***
$\boldsymbol{F}$	1	m^3/h	Flujo volumétrico
$\boldsymbol{V}$	1	m^3	Volumen del reactor
$\boldsymbol{R}$	1.985875	$kcal/kmol$	Constante de gases ideales de Boltzmann
$\boldsymbol{\Delta H}$	-5.960	$kcal/kmol$	Calor de la reacción por mol
$\boldsymbol{E}$	11.843	$kcal/kmol$	Energía de activación por mol
$\boldsymbol{k_0}$	34930.800	$1/h$	Factor pre-exponencial de Arrehenius
$\boldsymbol{\rho C_p}$	500	$kcal/m^3 \cdot °K$	Densidad multiplicada por la capacidad calorífica
$\boldsymbol{UA}$	150	$kcal/°K \cdot h$	Coeficiente de transferencia calorífica multiplicado por el área del reactor

Figura 31

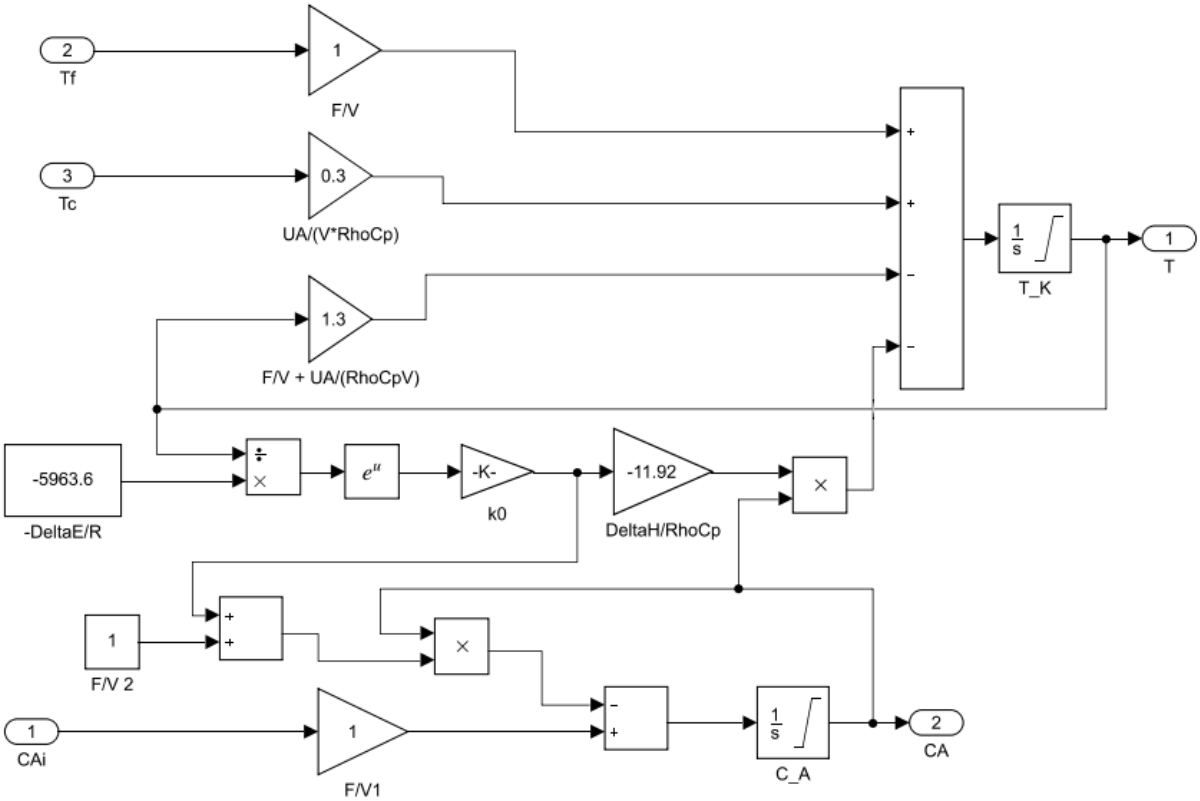
Diagrama de bloques del modelo CSTR



Nota: Gráfico extraído de (MathWorks, s.f.)

Figura 32

Modelo matemático del reactor de tanque agitado



Nota: Gráfico extraído de (MathWorks, s.f.)

Para el modelo el valor inicial de C_A es de $8.5698\ [kmol/m^3]$ y el valor inicial de la temperatura T es de $311.2639\ [°K]$. Este punto de operación se encuentra en equilibrio cuando la concentración del flujo de entrada C_{Af} es $10[kmol/m^3]$, la temperatura del flujo de entrada T_f es $300\ [°K]$, y la temperatura del refrigerante T_c es $292\ [°K]$. Estas entradas se muestran en la Figura 31, donde se presenta al modelo CSTR como un subsistema Simulink, mientras que en la Figura 32, se muestra el detalle del sistema no lineal con las ecuaciones diferenciales que rigen el modelo.

Para estos valores de las entradas la concentración del reactante A en el reactor, C_A, actúa en un rango de 0 a 10 $[kmol/m^3]$, en este mismo rango el controlador debe de operar.

Controlador PID por ajuste de ganancias

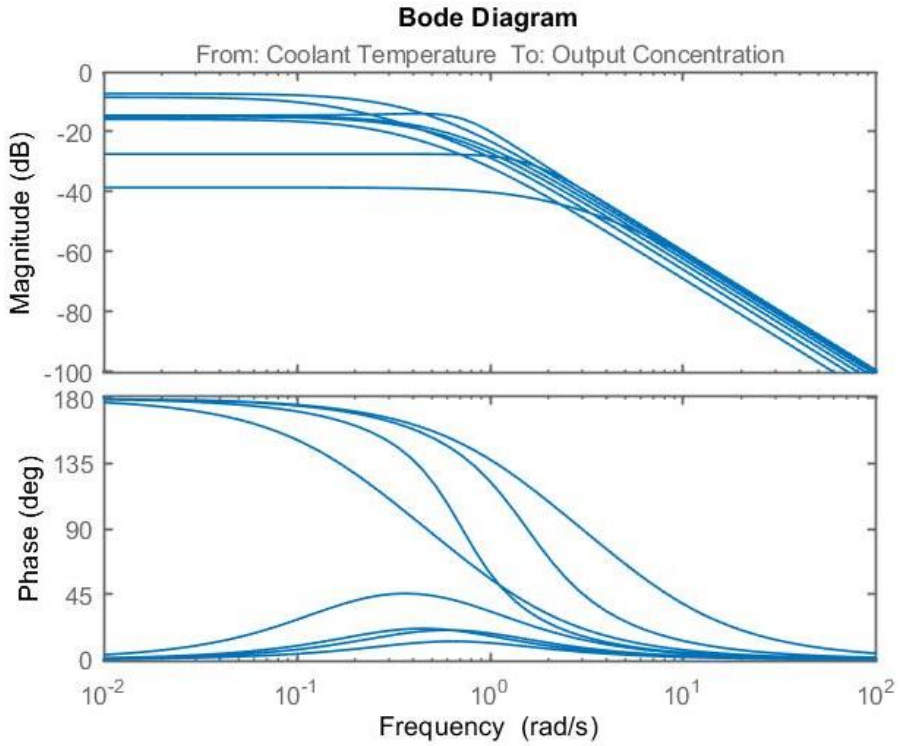
Este controlador consiste en la variación de las ganancias K_p, K_i y K_d del controlador PID, tomando en consideración los puntos de operación de la planta, estos son para concentración del reactante A en el reactor, C_A, de 1 hasta 8 $[kmol/m^3]$. Este controlador es propuesto por (MathWorks, s.f.).

Linealización de puntos de operación

Como primer paso se debe realizar la linealización de la planta para los puntos de operación. Para esto se hace uso del modelo del reactor de tanque agitado de Simulink, se específica las regiones de operación con los cuales trabaja el controlador; posteriormente se configura un conjunto de ocho plantas lineales que tengan como punto de equilibrio los valores de concentración del reactante A en el reactor (de 1 hasta 8 $[kmol/m^3]$).

Figura 33

Diagrama de bode de las plantas lineales para los puntos de operación



Los diagramas de bode para cada una de las plantas se observa en la Figura 33, tanto la magnitud como la fase cambian de manera significativa de un punto de operación a otro, por esta razón el controlador debe ser un PID por ajuste de ganancias y no uno individual ya que su desempeño solo sería óptimo en su punto de operación correspondiente.

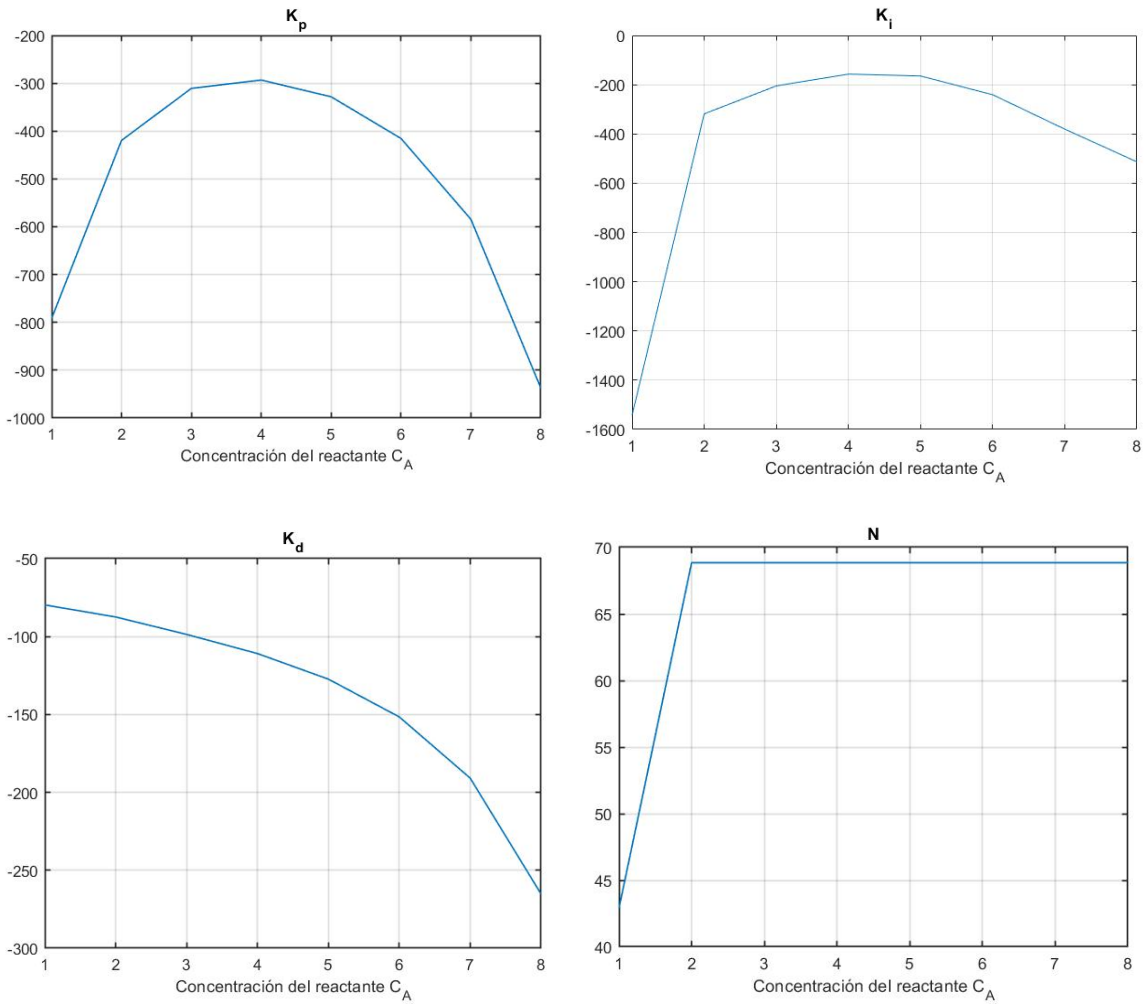
Se procede a sintonizar los controladores para las plantas conseguidas mediante el comando *pidtune* de Matlab, el controlador que se sintoniza para cada una de las plantas es un PID con filtro de la acción derivativa. Ya que el controlador PID por ajuste de ganancias será exportado como código en la etapa SIL, es necesario que los controladores individuales de cada planta sean discretos, por lo que se establece el periodo de muestreo de $T_s = 0.01\ [s]$, además, la frecuencia de cruce de ganancia es de $10\ [rad/s]$.

Ajuste de ganancias

Se extraen los valores de K_p, K_i, K_d y el orden del filtro de la acción derivativa N, de cada uno de los controladores para formar el ajuste de cada ganancia, en la Figura 34 se observa el ajuste realizado.

Figura 34

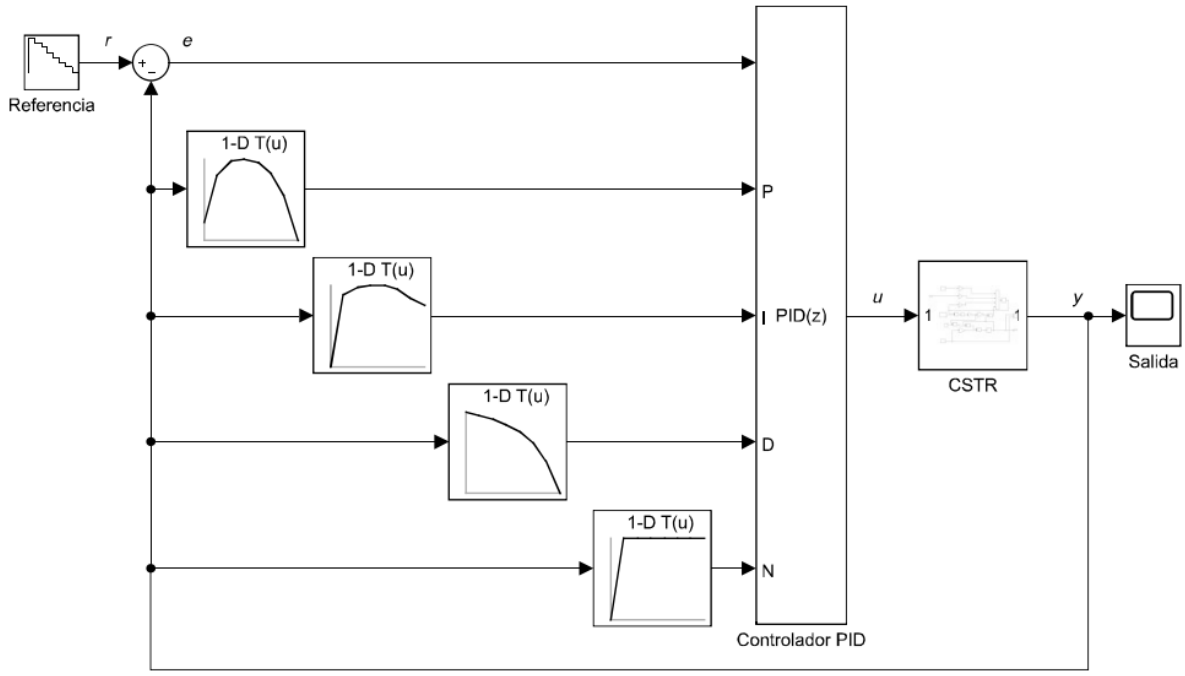
Ajuste de ganancias K, K_i, K_d y N



A partir de estas curvas de control se obtiene el ajuste de ganancias para el controlador PID, el esquema de simulación es el de la Figura 35.

Figura 35

Esquema de simulación para el controlador PID por ajuste de ganancias

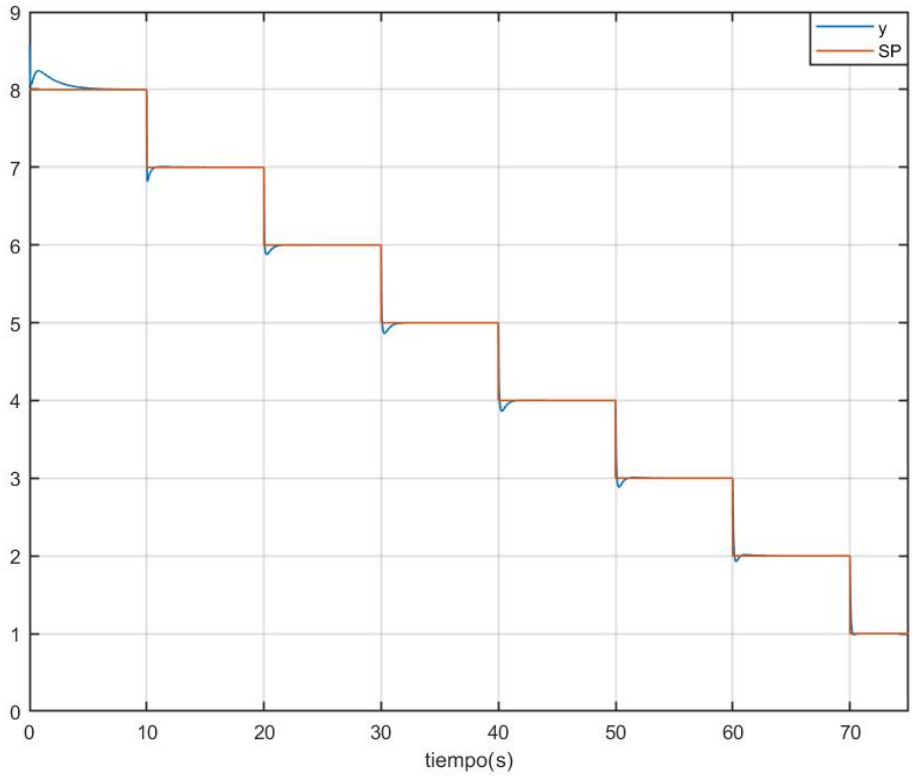


En este esquema el objetivo no es la cancelación de las no linealidades de manera directa con una aproximación a la inversa, lo que se realiza es un barrido de valores para los parámetros del controlador (K, K_i, K_d, N) mediante los cuales el conjunto de controlador y ajuste compensa la característica no lineal del reactor de tanque agitado.

De acuerdo a la Figura 36, el controlador PID por ajuste de ganancias se desempeña correctamente en cada uno de los puntos de operación que se habían definido, al contar con acción integral no presenta error en estado estacionario, además, para cada uno de los puntos de operación tiene un tiempo de establecimiento de aproximadamente 3 segundos. La transición entre los puntos de operación es suave y no se produce de manera abrupta, presenta un leve sobre impulso que es corregido por el controlador.

Figura 36

Respuesta a diferentes setpoints del CSTR mediante controlador por ajuste de ganancias



Caso de estudio modelo de entrada/salida

Los sistemas no lineales representados por modelos de entrada salida se clasifican en grupos de acuerdo a como ocurren las no linealidades en el modelo, son cuatro grupos de plantas discretas cuyas ecuaciones de diferencias son:

Modelo I: $y_p(k+1) = \sum_{i=0}^{n-1} \alpha_i y_p(k-i) + g[u(k), u(k-1), \cdots, u(k-m+1)]$

Modelo II: $y_p(k+1) = f\left[y_p(k), y_p(k-1), \cdots, y_p(k-n+1)\right] + \sum_{i=0}^{m-1} \beta_i u(k-i)$

Modelo III: $y_p(k+1) = f\left[y_p(k), y_p(k-1), \cdots, y_p(k-n+1)\right]$

$$+g[u(k), u(k-1), \cdots, u(k-m+1)]$$

Modelo IV: $y_p(k+1) = f\begin{bmatrix} y_p(k), y_p(k-1), \cdots, y_p(k-n+1); \\ u(k), u(k-1), \cdots, u(k-m+1) \end{bmatrix}$

Donde $[u(k), y_p(k)]$ representan la pareja de datos de entrada salida de la planta SISO para el instante k, y $m \leq n$. Para los cuatro modelos la salida en el instante $k+1$ depende tanto de los n valores anteriores $y_p(k-i)(i = 0, 1, \cdots, n-1)$, como de los m valores anteriores $u(k-j)(j = 0, 1, \cdots, m-1)$

La planta que se tratará en este caso de estudio pertenece al Modelo III, donde la dependencia no lineal de $y_p(k+1)$ entre $y_p(k-1)$ y $u(k-j)$ se asume como separable. (Narendra & Parthasarathy, 1990)

La ecuación que rige el modelo es la siguiente:

$$y_p(k+1) = \frac{y_p(k)}{1 + y_p^2(k)} + u^3(k)$$

Este modelo tiene la forma de $f[y_p(k)] = y_p(k)/(1 + y_p^2(k))$ y $g[u(k)] = u^3(k)$ de acuerdo a la ecuación del Modelo III.

Adquisición de datos

Como primer paso para la adquisición de datos del modelo se usará la entrada senoidal:

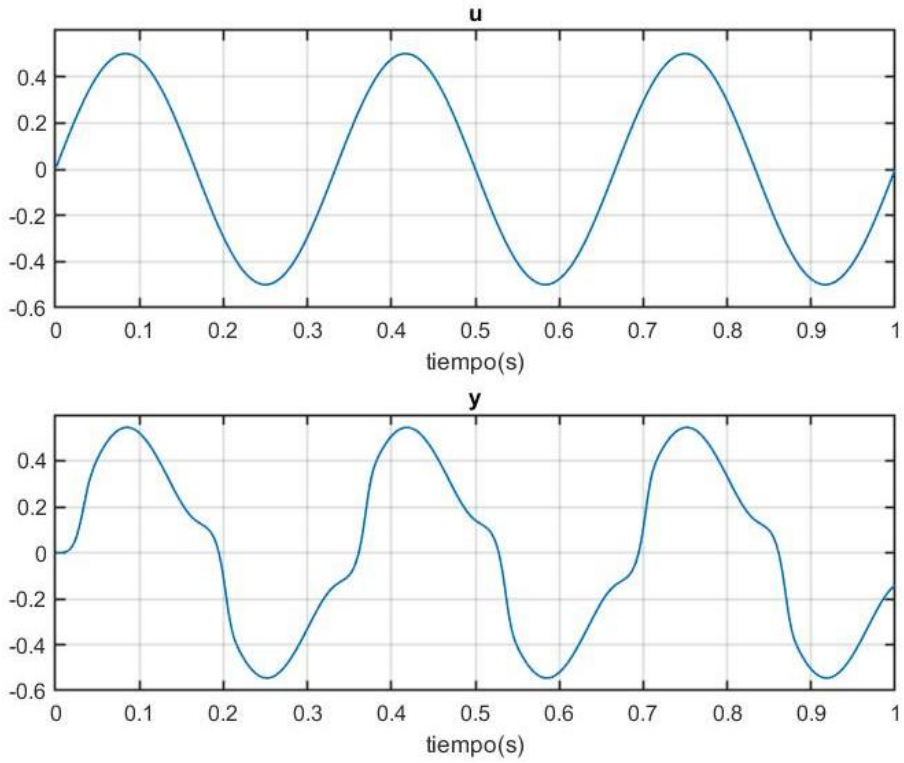
$$u(k \cdot Ts) = 0.5 \sin(6\pi k \cdot T_s)$$

Siendo el periodo de muestreo, $T_s = 1[ms]$

En la Figura 37, se visualiza la señal usada como entrada y la respuesta en lazo abierto del sistema no lineal propuesto por Narendra.

Figura 37

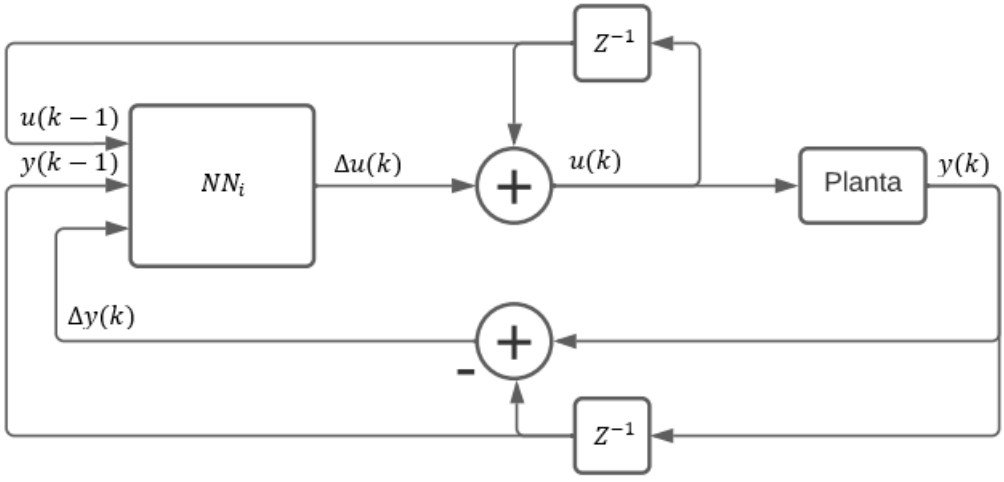
Señal de entrada senoidal y respuesta en lazo abierto del sistema no lineal



Arquitectura de red

Figura 38

Topología de la red neuronal inversa



De acuerdo a la Figura 38, se muestra el flujo de datos para la red neuronal inversa, nótese que los insumos de la red son la pareja de datos entrada salida retrasados $[u(k-1), y(k-1)]$ y la variación de la salida $\Delta y(k) = y(k) - y(k-1)$; mientras que la salida de la red es la variación de la entrada $\Delta u(k) = u(k) - u(k-1)$, razón por la cual se la denomina red neuronal inversa.

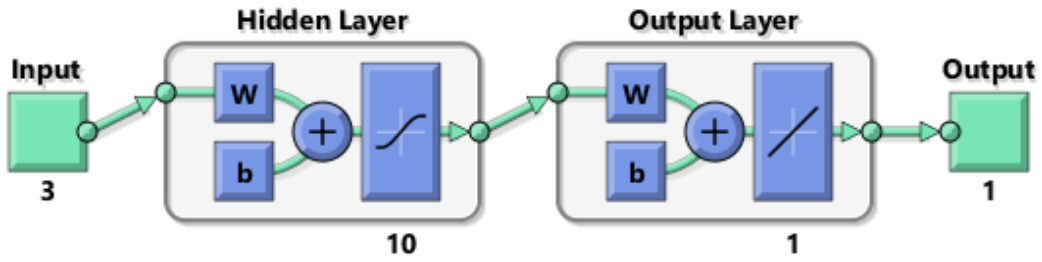
En síntesis, la red neuronal inversa se conforma de los siguientes entradas y targets:

$$P_m = \begin{bmatrix} u(k-1) \\ y(k-1) \\ \Delta y(k) \end{bmatrix} \quad T_m = [\Delta u(k)]$$

En la Figura 39, se muestra la arquitectura de la red neuronal inversa, esta se compone por las tres entradas y única salida explicadas anteriormente. El mejor desempeño se consiguió con 10 neuronas en la capa oculta utilizando una función de activación tangente hiperbólica sigmoidal, mientras que la capa de salida emplea una función de activación lineal.

Figura 39

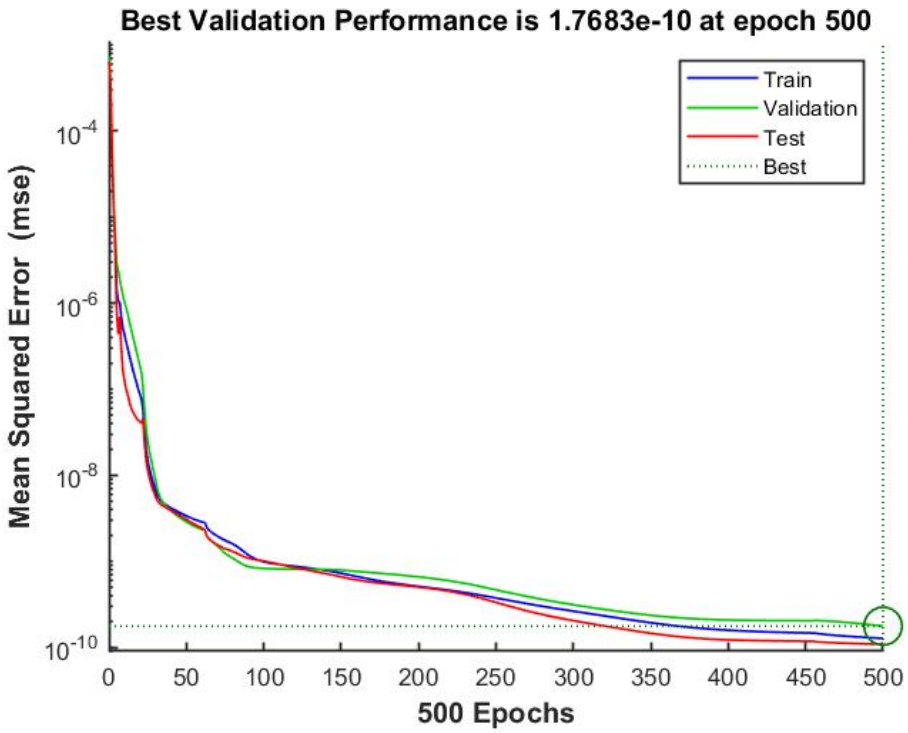
Arquitectura de red neuronal inversa



Para el aprendizaje se configuraron 500 épocas, gradiente mínimo de 1×10^{-10} y tasa de aprendizaje de 0.3. El desempeño de la red neuronal se muestra en la Figura 40, donde la validación converge a 1.7683×10^{-10} en la época 500.

Figura 40

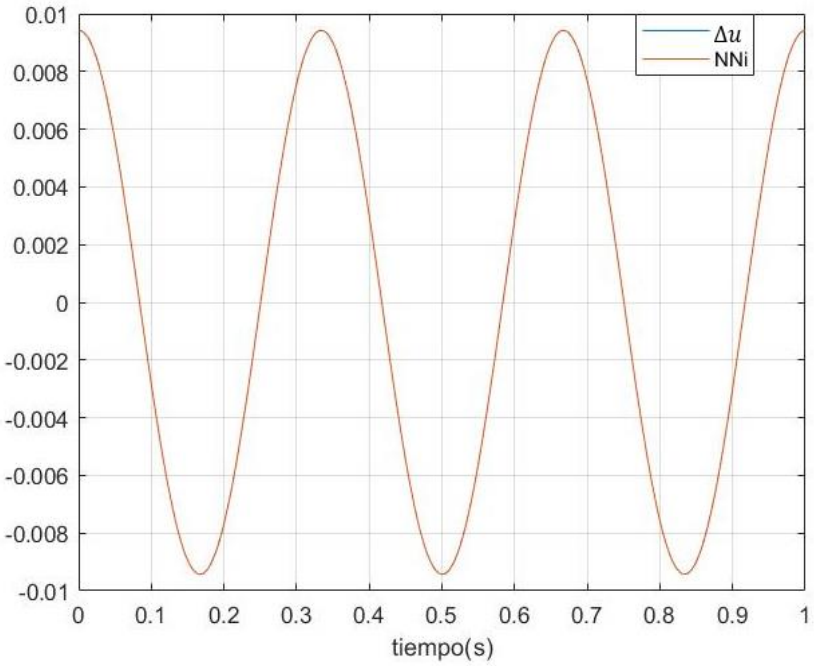
Error cuadrático medio en función de épocas



Realizado el aprendizaje se comprueba que la salida de la red neuronal inversa, Δu, sean iguales tanto la señal conseguida por la adquisición de datos como la del aprendizaje de la red neuronal. Esto se visualiza en la Figura 41, donde ambas señales se encuentran solapadas.

Figura 41

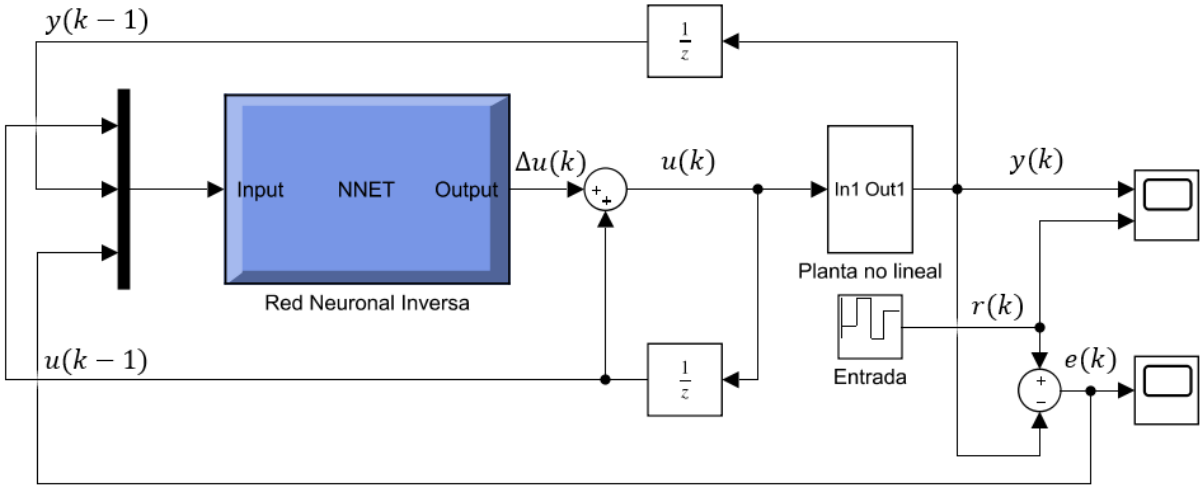
Validación de la red neuronal inversa para el modelo de Narendra



Después del aprendizaje de la red neuronal se exporta la misma hacia Simulink para realizar el control en conjunto con la planta del modelo de Narendra, el esquema de simulación se ejemplifica en la Figura 42.

Figura 42

Esquema de simulación de la red neuronal inversa en conjunto con la planta no lineal

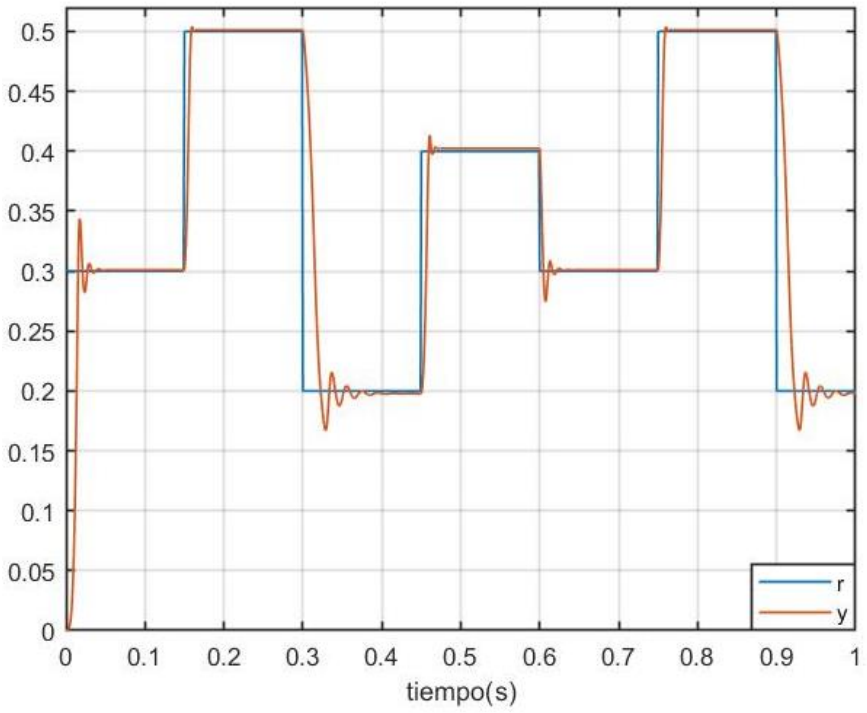


Nótese en la Figura 42 que el error $e(k)$ ha sido reemplazado por la variación de la salida $\Delta y(k)$, esto se realiza con el objetivo de controlar la planta no lineal.

En la Figura 43 se muestra la respuesta del controlador generado por red neuronal inversa, este presenta un desempeño correcto, logrando el control de la planta no lineal propuesta por Narendra. Cabe señalar que no se ajusta completamente al valor de referencia, puesto que no cuenta con algún tipo de acción integral que permita que el error en tiempo estacionario sea igual a cero.

Figura 43

Respuesta en lazo cerrado del conjunto red neuronal inversa y planta no lineal



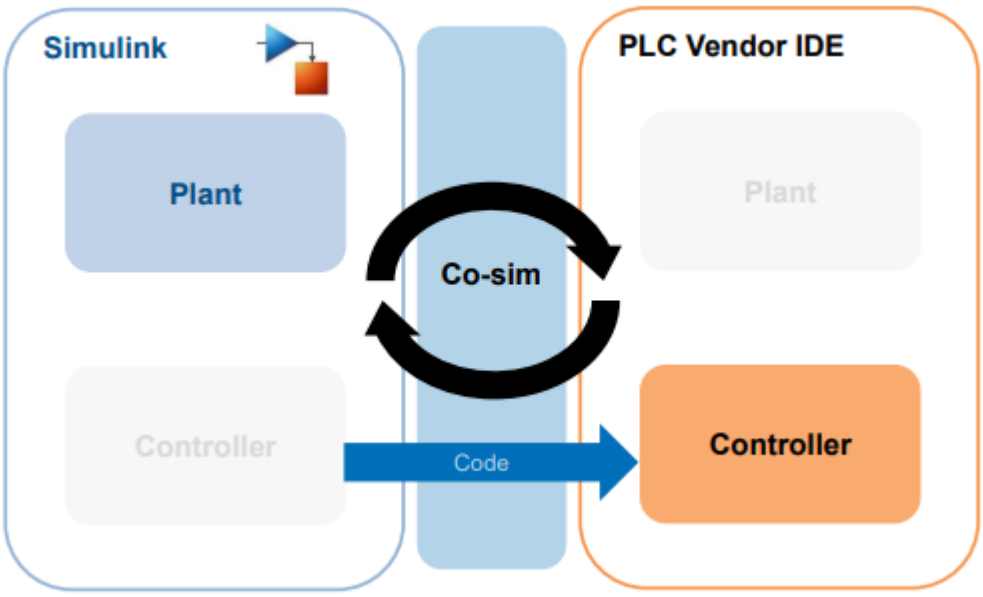
Capítulo IV

Realización e implementación en etapa SIL

El siguiente paso es integrar el modelo con un sistema real, la primera acción será generar el código de los controladores, para esto se hizo uso de la herramienta PLC Code de Simulink/Matlab, por medio de esta herramienta se obtiene código de los modelos ya validados y probados anteriormente, además, el código generado es texto estructurado bajo el estándar IEC 61131-3 que presenta la característica de ser rastreable con el diseño del modelo. El código generado está orientado hacia el entorno de desarrollo integrado TIA Portal de Siemens.

Figura 44

Diagrama de puesta en marcha virtual ("virtual commissioning")



Nota: Gráfico extraído de (Dijkstra & Marchant, 2020)

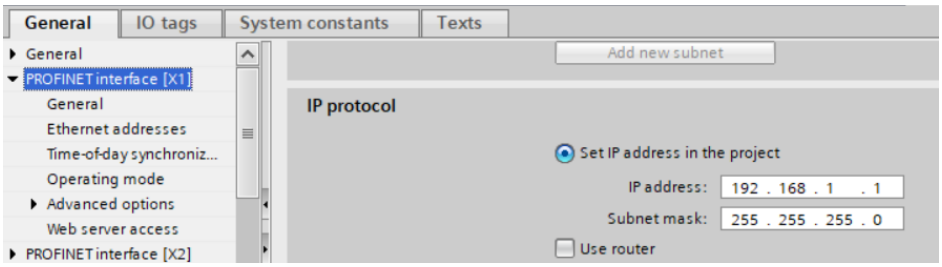
La Figura 44 ejemplifica como ocurre la puesta en marcha virtual propia de la etapa SIL, se comprende que el modelo de la planta reside en Simulink mientras que el controlador mediante la generación de código ha sido exportado hacia el PLC de Siemens en TIA Portal, la

“co-simulación” es posible gracias a la comunicación cliente-servidor por medio del protocolo de comunicación OPC UA.

El PLC donde se ejecuta el algoritmo de control es el S7-1500 CPU 1518-4 PN/DP MPF. Antes de exportar el código hacia el controlador es necesario configurar la interfaz PROFINET y el protocolo de comunicación OPC UA del servidor, es decir, del PLC. Para esto desde TIA Portal se selecciona el apartado de “Device Configuration” y posteriormente la opción de “PROFINET interface [X1]”, desde aquí se configura la dirección IP del controlador la cual es 198.168.1.1 como se muestra en la Figura 45.

Figura 45

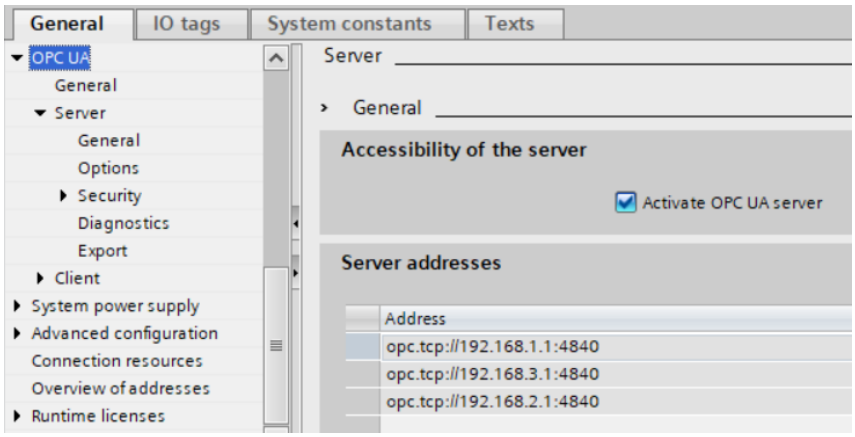
Dirección IP configurada en el controlador



Ahora desde la opción de “OPC UA” se debe habilitar el acceso hacia el servidor OPC UA del controlador, es necesario constatar la dirección IP que previamente fue configurada como se muestra en la Figura 46.

Figura 46

Habilitación del servidor OPC UA y sus direcciones

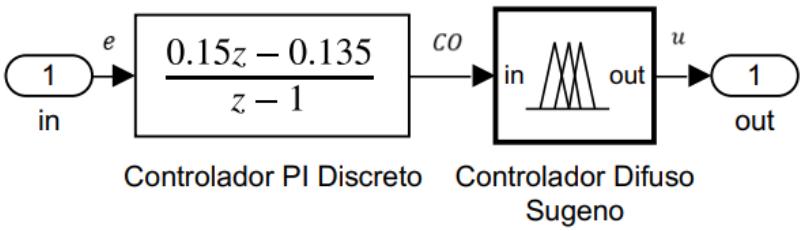


Caso de estudio actuador no lineal

Como ya se analizó en la etapa MIL, el controlador de este caso de estudio corresponde al conjunto PI discreto – Difuso tipo Sugeno, por lo cual se realiza un subsistema de estos dos bloques como se visualiza en la Figura 47.

Figura 47

Modelo subsistema controlador PI discreto – Difuso tipo Sugeno

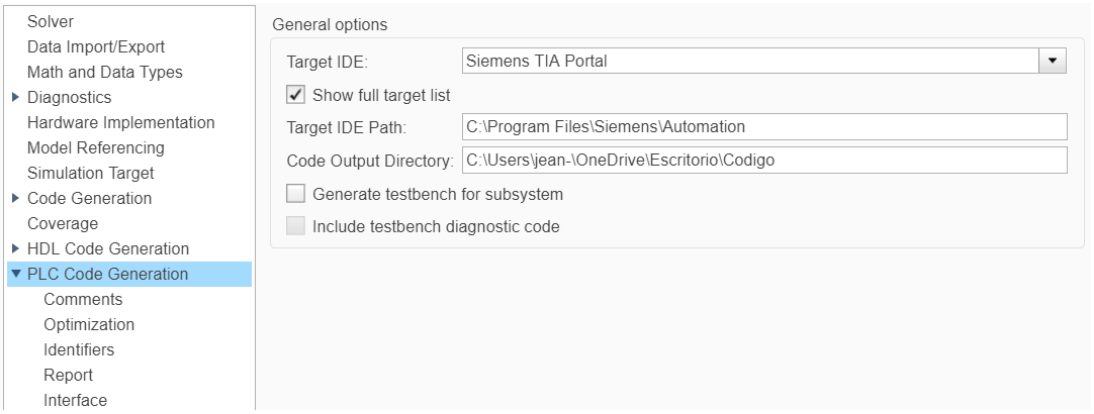


Antes de generar el código se selecciona que Simulink trate al subsistema como una unidad atómica, además, es necesario configurar el periodo de muestreo, esto es de gran importancia y debe ser el mismo que se preestableció en la etapa MIL ($T_s = 0.1\ [s]$), si el periodo de muestreo es diferente el algoritmo no converge. Por último, en las opciones de

generación de código se debe escoger como IDE “Siemens TIA Portal” como se aprecia en la Figura 48.

Figura 48

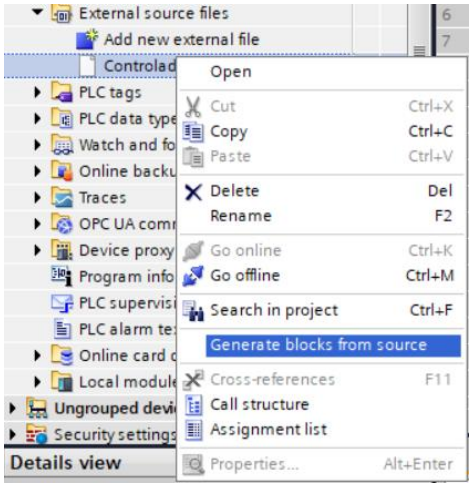
Opciones de configuración PLC Code



El archivo generado tiene la extensión SCL, este es el archivo que debe ser importado desde TIA Portal en el apartado de “External source files”, una vez realizado esto se generan los bloques de programación propios del código generado como se muestra en la Figura 49.

Figura 49

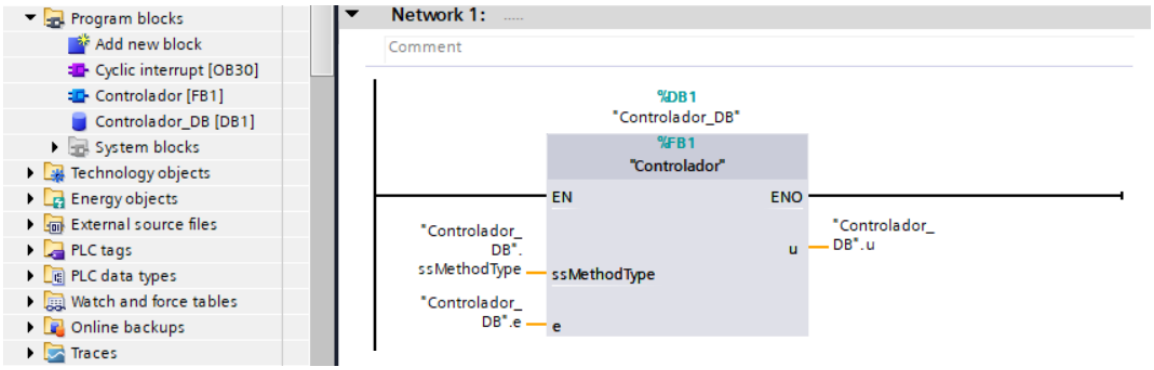
Generación de bloques a partir del código generado



Cuando se hayan generado los bloques correspondientes del código exportado, se debe crear una interrupción cíclica cuyo periodo sea igual al periodo de muestreo ya establecido anteriormente ($T_s = 0.1\ [s]$). En el programa de la interrupción cíclica se arrastra el bloque generado, por defecto se creará un bloque de datos y se configuran las entradas/salidas del programa de la interrupción, la vista de proyecto hasta este momento se muestra en la Figura 50.

Figura 50

Vista de proyecto en TIA Portal con el controlador configurado



En la Tabla 5 se aprecian el conjunto de datos entrada/salida del bloque de datos de TIA Portal y su representación en el lazo de control.

Tabla 5

Conjunto de datos entrada/salida del controlador PI digital – Difuso Sugeno

Tipo	***Tag TIA Portal***	***Lazo de control***
Entrada	Controlador_DB.e	Señal de error, e
Salida	Controlador_DB.u	Salida del controlador difuso Sugeno, u

Ahora se procede con la creación de una función definida por el usuario desde Matlab con el nombre de “Read_OPC_Func”, esta permite el intercambio de datos con el servidor OPC desde Matlab.

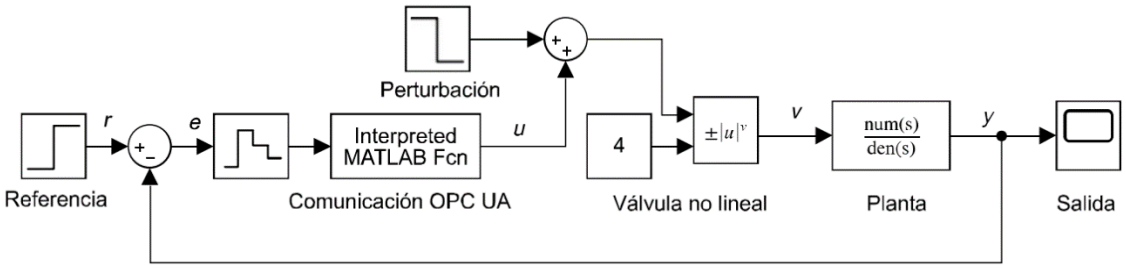
```
function [x] = Read_OPC_Func(y)
% variables
persistent init_Server;
persistent init_Nodes;
persistent uaClient;
persistent Var_Node_In;
persistent Var_Node_Out;
persistent testVal;
% inicialización de variables
if (isempty(init_Server))
     testVal = 0;
     init_Server = 0;
     init_Nodes = 0;
end
% Dirección del servidor OPC UA (PLC)
% y conexión del cliente (Simulink)con el servidor
if init_Server == 0
     init_Server = 1;
     uaClient = opcua('192.168.1.1',4840);
     connect(uaClient);
end
% Definición de los nodos de variable del servidor
if uaClient.isConnected == 1 && init_Nodes == 0
     init_Nodes = 1;
     % Lectura de variables del servidor OPC UA
     Var_Node_Out =
opcuanode(3,'"Controlador_DB"."e"',uaClient);
     Var_Node_In =
opcuanode(3,'"Controlador_DB"."u"',uaClient);
end
% Lectura y escritura de las variables del servidor
if uaClient.isConnected == 1 && init_Nodes == 1
     % Lectura del valor de la salida del controlador
Difuso
     % y almacenamiento en "val"
     [val, ~, ~] = readValue(uaClient, Var_Node_In);
```

```
     % Asignar a la entrada de la función y el valor
del error
     % del lazo de control
     writeValue(uaClient, Var_Node_Out, y);
     % Asignar el valor de "val" a la variable
"testVal"
     testVal = val;
end
% Asignar el valor de la salida del controlador
difuso
% a la salida x de la función
x = double(testVal);
end
```

Esta función es integrada en el modelo de Simulink por medio del bloque “Interpreted Matlab Function”, el bloque reemplaza al controlador PI digital – Difuso Sugeno de la etapa MIL, el esquema de puesta en marcha virtual es el que se muestra en la Figura 51.

Figura 51

Esquema de puesta en marcha virtual, caso de estudio actuador no lineal

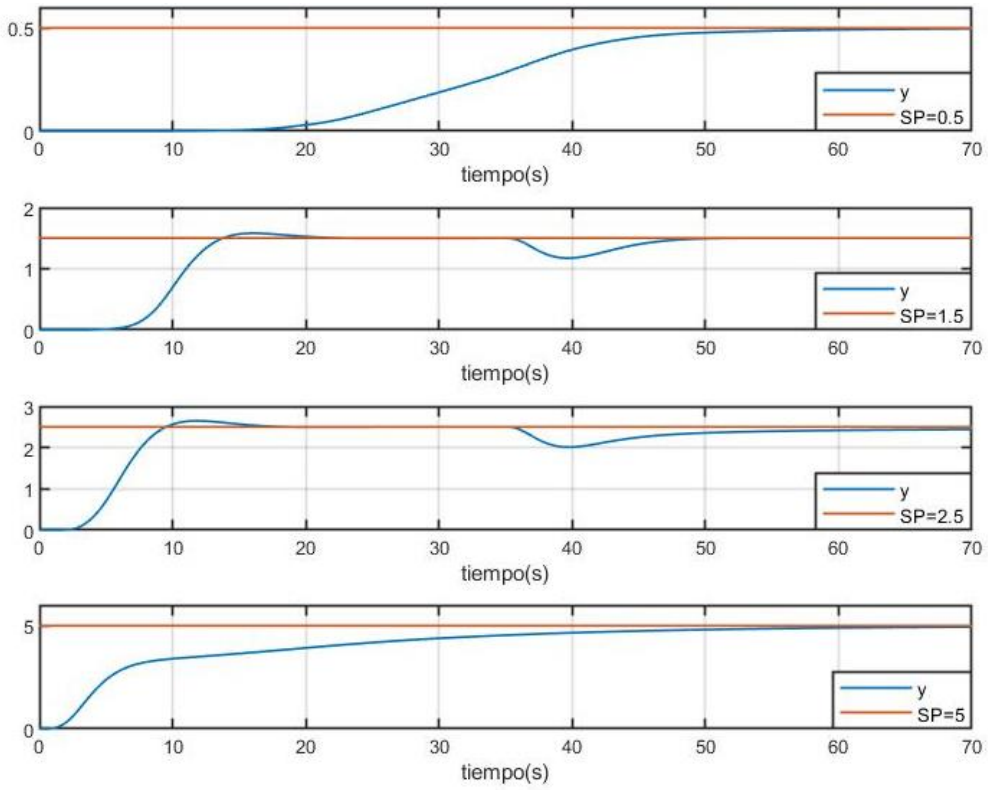


Para arrancar la co-simulación es necesario crear una instancia en el programa “PLCSIM Advanced V3.0” con la dirección IP del PLC (192.168.1.1), una vez creada la instancia se debe cargar el programa al PLC simulado, durante el proceso de compilación y carga del programa no han de existir errores. Se configura al PLC en línea para que el controlador ejecute el algoritmo de control y se actualicen los registros de entrada/salida, por último, se ejecuta la simulación

desde Simulink. Una característica de la co-simulación es que la gráfica desde Simulink se dibuja en tiempo real mientras se ejecuta el algoritmo de control.

Figura 52

Respuesta del primer caso de estudio, implementación SIL



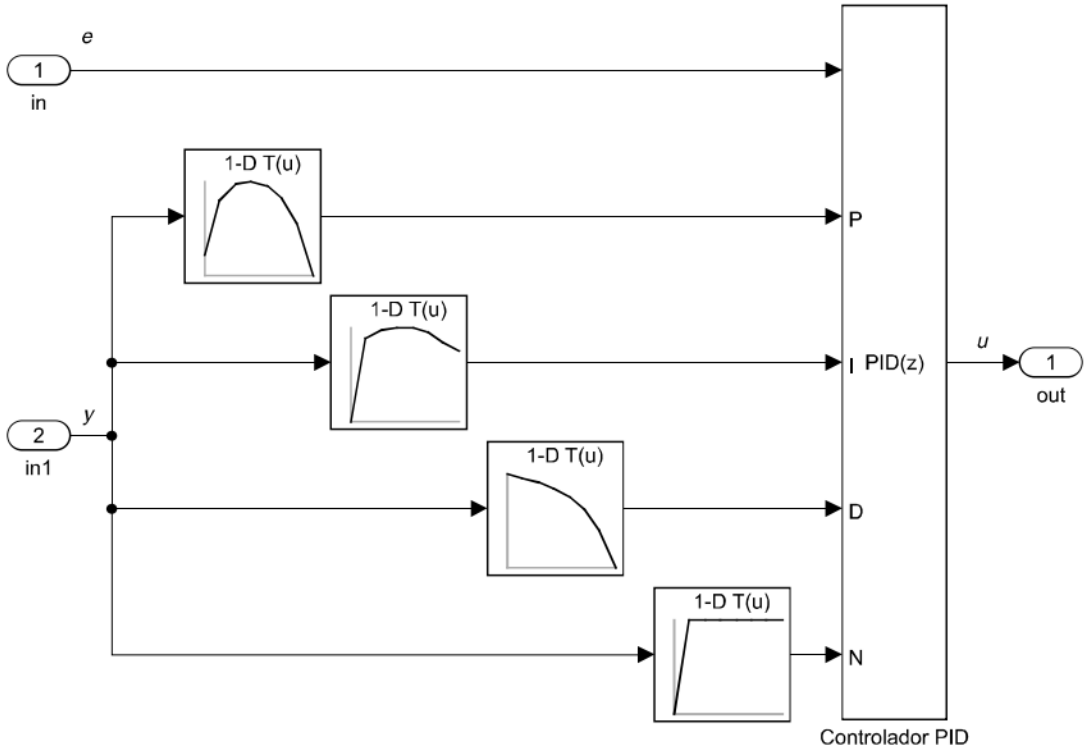
En la Figura 52 se aprecian los resultados de la co-simulación para el sistema del actuador no lineal, se observa que no se ha perdido desempeño para ninguno de los setpoints en el rango establecido, para los setpoints de 1.5 y 2.5 se provocó una perturbación a los 35 segundos de amplitud -0.1, en ambos casos el valor de la salida se ajusta nuevamente a la referencia, esto se debe a la acción integral proporcionada por el controlador PI discreto.

Caso de estudio reactor de tanque agitado

Para este caso de estudio el controlador está conformado por los ajustes individuales de las ganancias K_p, K_i y K_d en forma de *Lookup Table* y del controlador PID, este conjunto de bloques se establece como un subsistema como se observa en la Figura 53.

Figura 53

Modelo subsistema del controlador por ajuste de ganancias



A este subsistema se lo debe tratar como una unidad atómica como se realizó en el anterior caso de estudio, al exportar el código se establece el periodo de muestreo igual que en la etapa MIL ($T_s = 0.01\ [s]$). Desde TIA Portal se importa el código generado por medio del archivo con la extensión SCL como ya se explicó en el primer caso de estudio, posteriormente se generan los bloques de este archivo importado y se crea una interrupción cíclica que tenga configurado el mismo tiempo de muestreo $T_s = 0.01\ [s]$ que ejecute de manera constante el

controlador. Por último, el bloque que se generó a partir del código es arrastrado hacia la interrupción cíclica y se crea por defecto el bloque de datos donde se establecen las entradas/salidas, este conjunto de datos se presenta en la Tabla 6.

Tabla 6

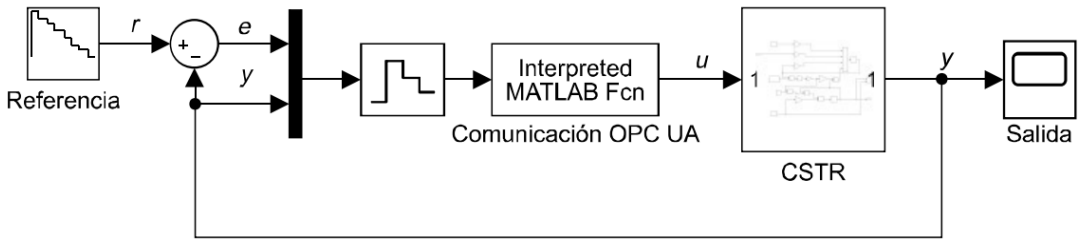
Conjunto de datos entrada/salida del controlador por ajuste de ganancias

Tipo	***Tag TIA Portal***	***Lazo de control***
Entrada	Controlador_DB.In	Señal de error, e
Entrada	Controlador_DB.In1	Señal de salida, y
Salida	Controlador_DB.out	Salida del controlador PID, u

La función que permite el intercambio de datos con el servidor OPC desde Matlab es similar a la del caso de estudio anterior, con la diferencia que ahora presenta dos entradas. Esta función es integrada en el modelo de Simulink reemplazando el controlador por ajuste de ganancias de la etapa MIL, el esquema de puesta en marcha virtual es el que se muestra en la Figura 54.

Figura 54

Esquema de puesta en marcha virtual, caso de estudio rector de tanque agitado

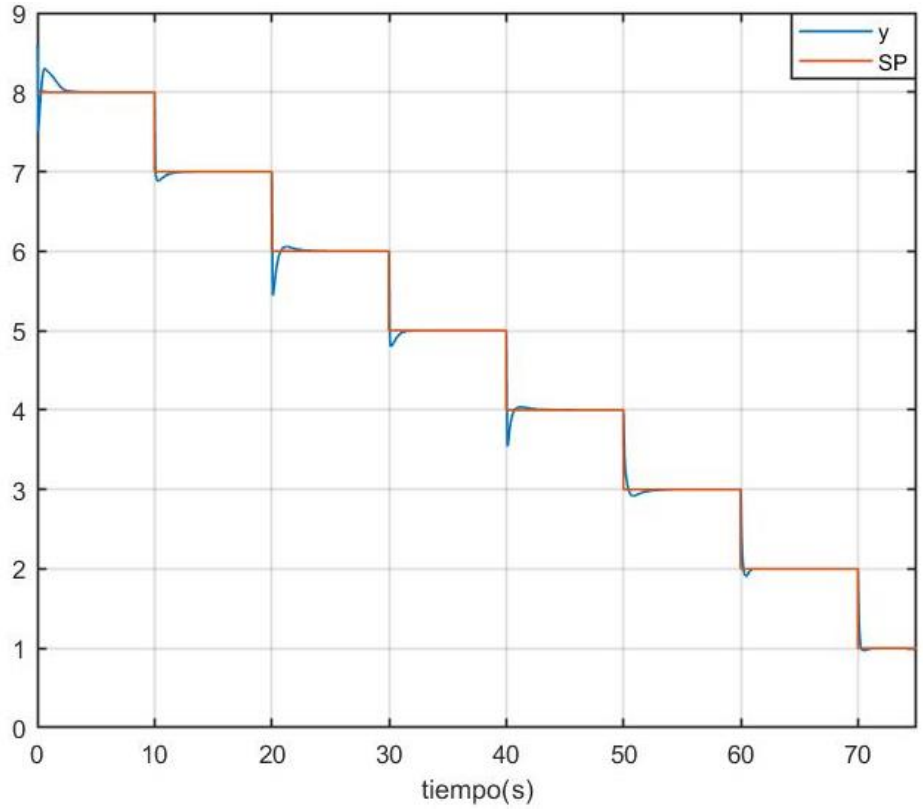


En la Figura 55 se aprecia el resultado de la puesta en marcha virtual para el caso de estudio del reactor de tanque agitado, para todos los setpoints de concentración el controlador

se ajusta a cada uno de los valores gracias a la acción integral del control PID, como observaciones se presenta un mayor sobre impulso para las concentraciones de $6\ [kmol/m^3]$ y $4\ [kmol/m^3]$, y un tiempo de establecimiento menor a 2 segundos.

Figura 55

Respuesta del segundo caso de estudio, implementación SIL

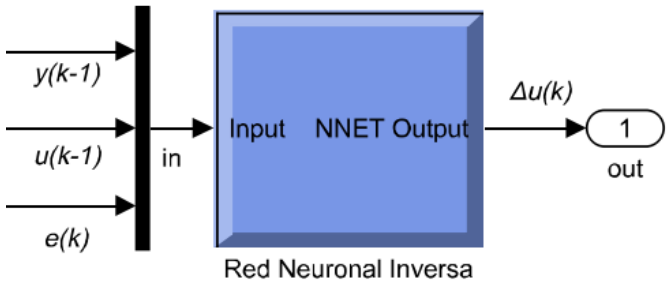


Caso de estudio modelo de entrada/salida

El controlador que debe ser exportado en este caso de estudio es la red neuronal, por lo que este bloque desde el esquema de Simulink debe ser tratado como una unidad atómica, configurar el periodo de muestreo en $T_s = 1\ [ms]$ como ya se estableció en la etapa de simulación MIL y ser exportado hacia el IDE de TIA Portal. Nótese de acuerdo a la Figura 56 se envía el bloque de datos de entrada en un solo vector $[u(k-1), y(k-1), e(k)]$.

Figura 56

Modelo subsistema del controlador por red neuronal inversa



El archivo con extensión SCL es importado desde TIA Portal y se generan los bloques correspondientes del código, se establece una interrupción cíclica con el mismo periodo de muestreo de $T_s = 1\ [ms]$. Para finalizar con la configuración en TIA Portal se arrastra el bloque del controlador hacia la interrupción cíclica y se crea el bloque de datos por defecto.

Estos pasos que involucran la exportación del código desde Simulink y la importación desde TIA Portal se deben seguir igual como se ha explicado en los tres casos de estudio, ya que se trata de la metodología para la generación de código desde Matlab hacia el PLC.

El conjunto de entradas/salida de este controlador, su tag desde TIA Portal y su representación en el lazo de control se muestra en la Tabla 7.

Tabla 7

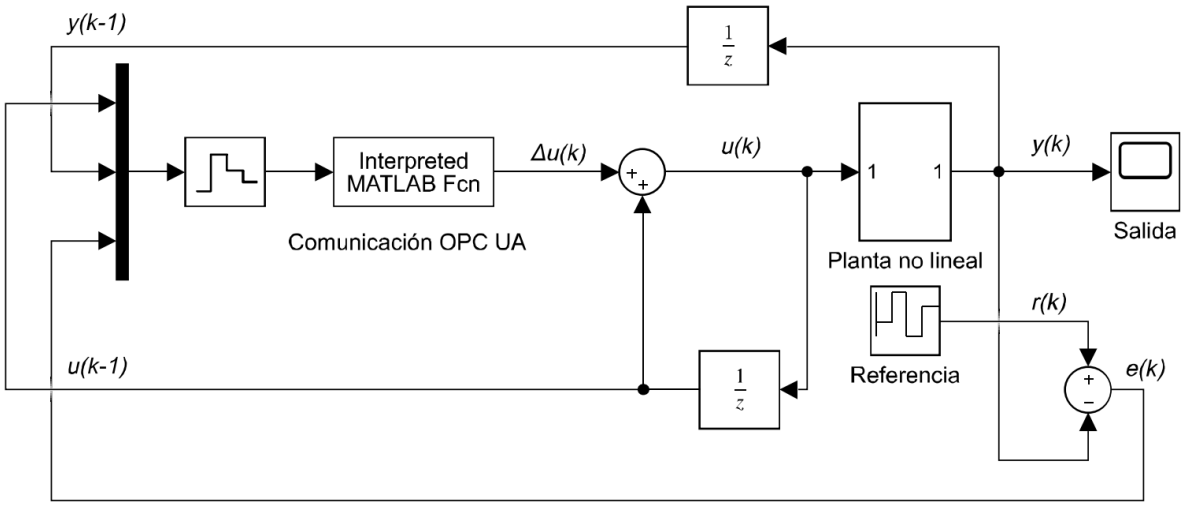
Conjunto de datos entrada/salida del controlador por red neuronal inversa

Tipo	***Tag TIA Portal***	***Lazo de control***
Entrada	Controlador_DB.In[0]	Señal de entrada retrasada, $u(k-1)$
	Controlador_DB.In[1]	Señal de salida retrasada, $y(k-1)$
	Controlador_DB.In[2]	Señal de error, $e(k)$
Salida	Controlador_DB.out	Variación de la señal de entrada, $\Delta u(k)$

Desde Matlab se crea la función que permita la comunicación entre el cliente (Simulink) y el servidor OPC (PLC/TIA Portal), para este caso de estudio la función tiene una sola entrada que son el vector de regresores, y una sola salida como se explica en la Tabla 5. Esta función reemplaza la red neuronal inversa del esquema de simulación de la etapa MIL, dando paso al nuevo esquema de co-simulación que se aprecia en la Figura 57.

Figura 57

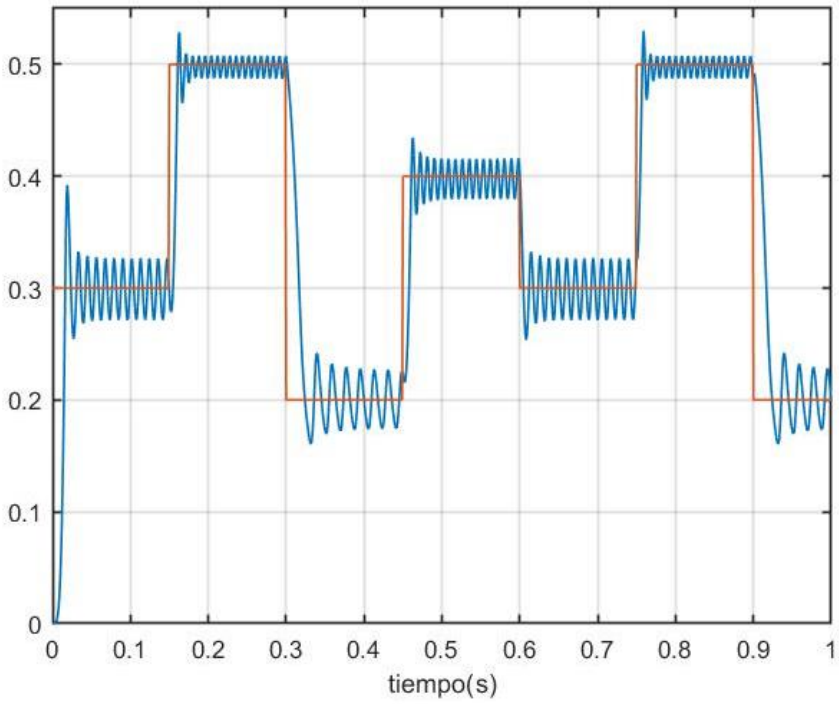
Esquema de puesta en marcha virtual, caso de estudio modelo entrada/salida



En la Figura 58 se presenta la respuesta de la puesta en marcha virtual para el caso de estudio del modelo entrada/salida no lineal, se observa que el sistema de control realiza el seguimiento a cada uno de los valores referenciales, sin embargo, muestra el efecto ringing que ya se notaba en la etapa MIL pero en la implementación SIL es más agresivo, esto se debe a que el controlador de la red neuronal inversa aproxima de manera inexacta la cancelación dinámica del proceso.

Figura 58

Respuesta del tercer caso de estudio, implementación SIL



Una alternativa para solucionar el efecto ringing es implementar un controlador Neuro-Fuzzy que utilice como entradas el vector de regresores, $[u(k-1), y(k-1), e(k)]$, y como salida la variación de la señal de entrada, $\Delta u(k)$, este conjunto de datos es adquirido directamente del esquema de simulación de la etapa MIL.

Controlador Neuro-Fuzzy

Funciones de membresía de entrada

Se manejan tres entradas en el controlador Neuro-Fuzzy, $[u(k-1), y(k-1), e(k)]$, cada una de estas tiene sus propias funciones de membresía, el intervalo en que trabajan las funciones de membresía de cada entrada depende de la adquisición de las señales de la red neuronal. Se utilizaron tres funciones de membresía del tipo triangular, estas son N, Z y P, sus

parámetros se resumen en la Tabla 8. Las funciones de membresía de cada entrada se presentan en la Figura 59.

Figura 59

Funciones de membresía de las entradas del controlador Neuro-Fuzzy

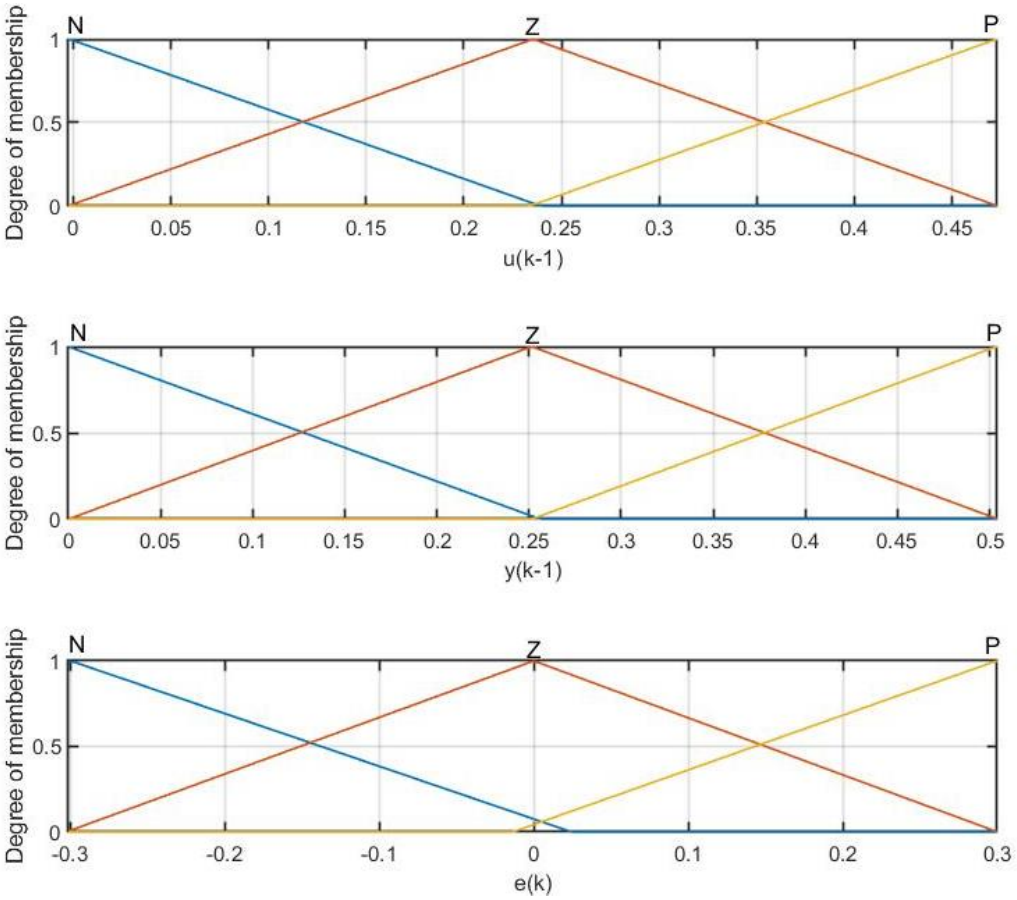


Tabla 8

Parámetros de las funciones de membresía de entrada

Entrada	***Parámetros***
$\boldsymbol{u(k-1)}$	$M(N) = trimf(u; -0.02, 0, 0.235)$ $M(Z) = trimf(u; -0.02, 0.235, 0.47)$ $M(P) = trimf(u; 0.235, 0.47, 0.50)$

$\boldsymbol{y(k-1)}$	$M(N) = trimf(y; -0.02, 0, 0.257)$
	$M(Z) = trimf(y; 0, 0.257, 0.50)$
	$M(P) = trimf(y; 0.257, 0.50, 0.505)$
$\boldsymbol{e(k)}$	$M(N) = trimf(e; -0.305, -0.30, 0.02)$
	$M(Z) = trimf(e; -0.30, 0, 0.30)$
	$M(P) = trimf(e; -0.01, 0.30, 0.305)$

Funciones lineales de salida

El controlador Neuro-Fuzzy es del tipo Sugeno, las funciones lineales que conforman la salida son constantes que fueron determinadas por la misma aplicación, estas se resumen en la Tabla 9.

Tabla 9

Parámetros de las funciones lineales de salida

Nombre	***Parámetro***
$\boldsymbol{MF_1}$	-1.223
$\boldsymbol{MF_2}$	0.274
$\boldsymbol{MF_3}$	0.028
$\boldsymbol{MF_4}$	0.069
$\boldsymbol{MF_5}$	-0.050
$\boldsymbol{MF_6}$	0.020
$\boldsymbol{MF_7}$	0.877
$\boldsymbol{MF_8}$	-0.362
$\boldsymbol{MF_9}$	12.774
$\boldsymbol{MF_{10}}$	-0.614
$\boldsymbol{MF_{11}}$	0.023
$\boldsymbol{MF_{12}}$	0.0234
$\boldsymbol{MF_{13}}$	-0.274
$\boldsymbol{MF_{14}}$	0.020

Nombre	***Parámetro***
MF_{15}	0.034
MF_{16}	0.241
MF_{17}	-0.035
MF_{18}	0.151
MF_{19}	0.838
MF_{20}	-0.040
MF_{21}	0.119
MF_{22}	-0.318
MF_{23}	0.032
MF_{24}	3.491×10^{-5}
MF_{25}	-0.020
MF_{26}	-5.054×10^{-4}
MF_{27}	0.059

Reglas de control

Al contar con 27 constantes de las funciones lineales de salida, se tiene 27 reglas que forman el controlador difuso, estas son:

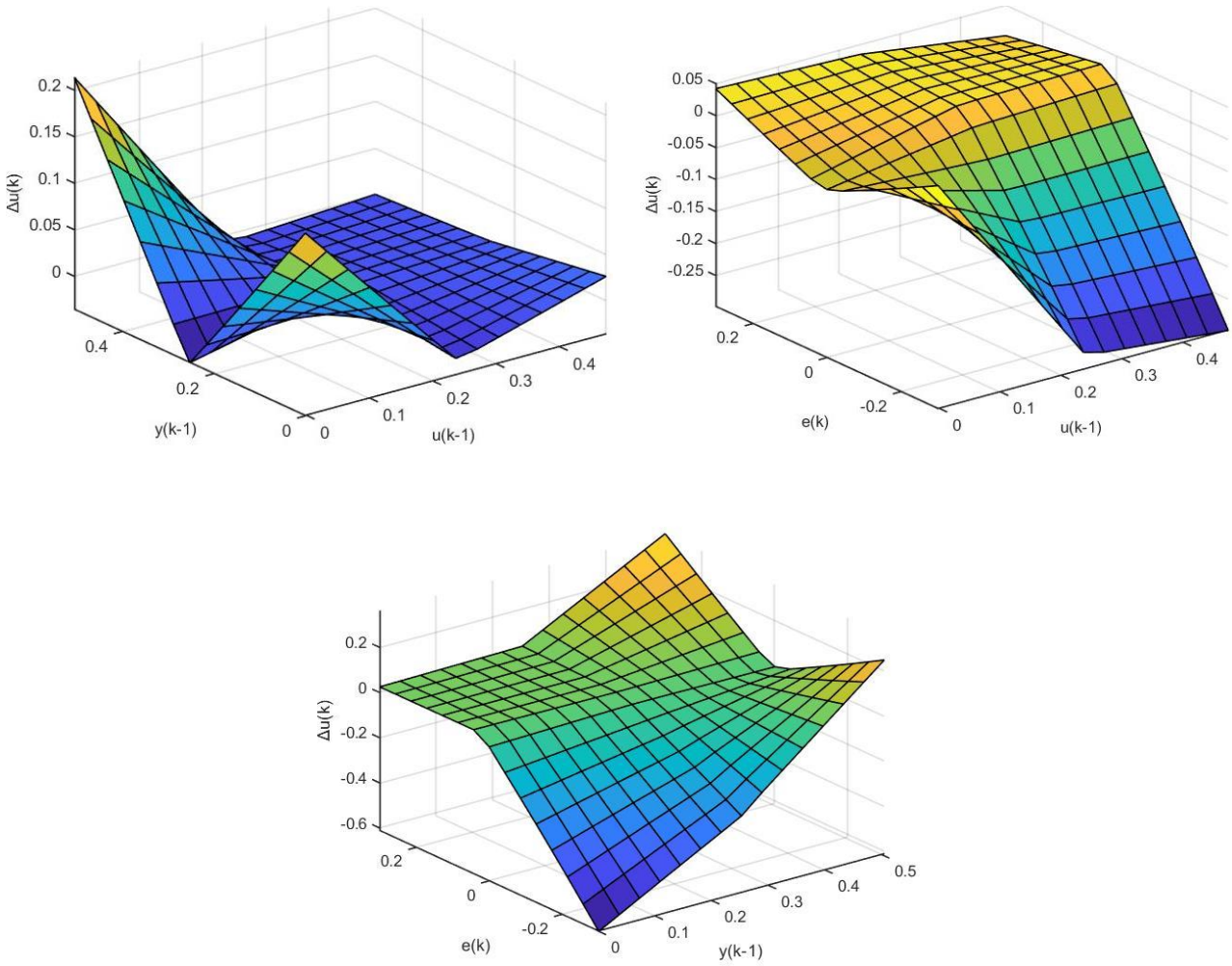
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_1$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_2$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_3$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_4$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_5$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_6$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_7$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_8$$
$$SI\ u(k-1)\ es\ N\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_9$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_{10}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_{11}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_{12}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_{13}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_{14}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_{15}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_{16}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_{17}$$
$$SI\ u(k-1)\ es\ Z\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_{18}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_{19}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_{20}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ N\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_{21}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_{22}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_{23}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ Z\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_{24}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ N\ entonces\ \Delta u(k)\ es\ MF_{25}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ P\ entonces\ \Delta u(k)\ es\ MF_{26}$$
$$SI\ u(k-1)\ es\ P\ \&\ y(k-1)\ es\ P\ \&\ e(k)\ es\ Z\ entonces\ \Delta u(k)\ es\ MF_{27}$$

Superficies de control

Al contar con tres entradas y una salida se generan tres superficies de control, estas se presentan en la Figura 60.

Figura 60

Superficies de control del controlador Neuro-Fuzzy



Este controlador diseñado es exportado hacia TIA Portal bajo la metodología ya conocida, para la puesta en marcha virtual se usa el mismo esquema de la Figura 57. La función que comunica Simulink con el servidor OPC UA cambia los tags de los datos entrada/salida según la Tabla 10.

Tabla 10

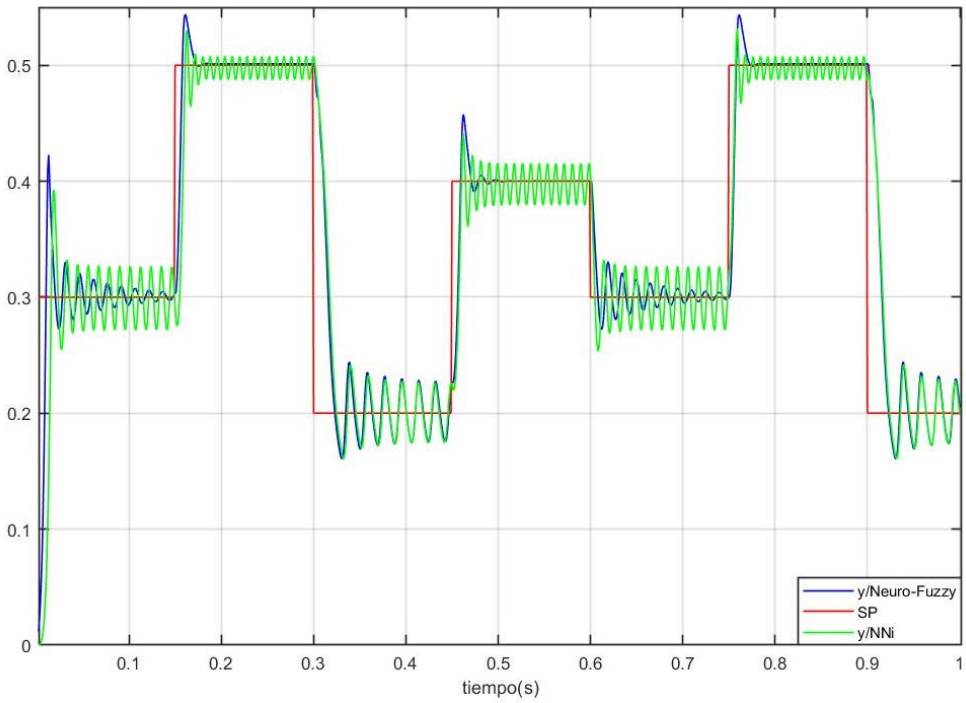
Conjunto de datos entrada/salida del controlador Neuro-Fuzzy

Tipo	***Tag TIA Portal***	***Lazo de control***
Entrada	Fuzzy_DB.In[0]	Señal de entrada retrasada, $u(k-1)$
	Fuzzy_DB.In[1]	Señal de salida retrasada, $y(k-1)$
	Fuzzy_DB.In[2]	Señal de error, $e(k)$
Salida	Fuzzy_DB.out	Variación de la señal de entrada, $\Delta u(k)$

La respuesta del sistema de control utilizando el controlador Neuro-Fuzzy para el modelo de entrada/salida no lineal se presenta en la Figura 61, como se puede observar el efecto rizo fue atenuado satisfactoriamente para los valores de referencia mayores a 0.3 e incluso no existe para el setpoint de 0.5, sin embargo, para la referencia de 0.2 aún presenta efecto ringing. El controlador difuso omite los coeficientes de los pesos que se usan para la red neuronal, razón por la cual su implementación durante la puesta en marcha virtual se ve favorecido en desempeño y atenúa el efecto ringing.

Figura 61

Respuesta a diferentes setpoints del tercer caso de estudio, implementación SIL del controlador Neuro-Fuzzy



Capítulo V

Validación de resultados

A continuación, se compara el desempeño de los resultados individuales de las etapas MIL y SIL para cada uno de los casos de estudio. Esto se realiza con el objetivo de verificar que no se haya perdido rendimiento en la etapa de generación de código y las prestaciones del controlador en la etapa SIL sean las mismas que en la etapa MIL.

Caso de estudio actuador no lineal

Para este sistema en particular se compara el resultado para cada uno de los setpoints en los que debe operar el actuador no lineal, estos son $[0.5\ 1.5\ 2.5\ 5]$.

Figura 62

Comparación de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=0.5

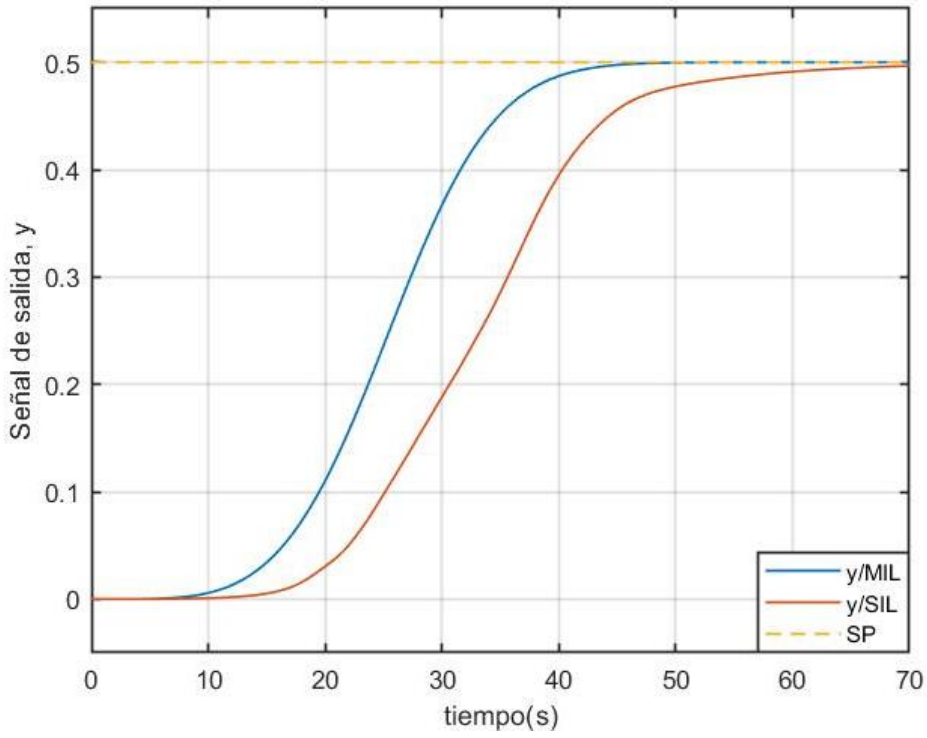


Tabla 11

Comparación de desempeño de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=0.5

Etapa	***Overshoot (%)***	***Tiempo de establecimiento (s)***	***Error en estado estacionario***
MIL	N/A	54	7.31×10^{-7}
SIL	N/A	70	3.4×10^{-3}

De acuerdo a la Figura 62 y según la Tabla 11, el desempeño de la etapa MIL es mejor que el de la etapa SIL, se ajusta al valor de referencia más pronto, mientras que la puesta en marcha virtual toma más tiempo. En la simulación de modelos (MIL) la señal de error se ajusta casi en su totalidad a cero, y en la etapa SIL el error aún presenta milésimas de valor. Para este setpoint el resultado es mejor en la etapa MIL, esto se da porque los registros de entrada/salida del controlador están en cero y aún no cuentan con valores hasta cuando se ejecute la simulación en Simulink.

Figura 63

Comparación de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=1.5

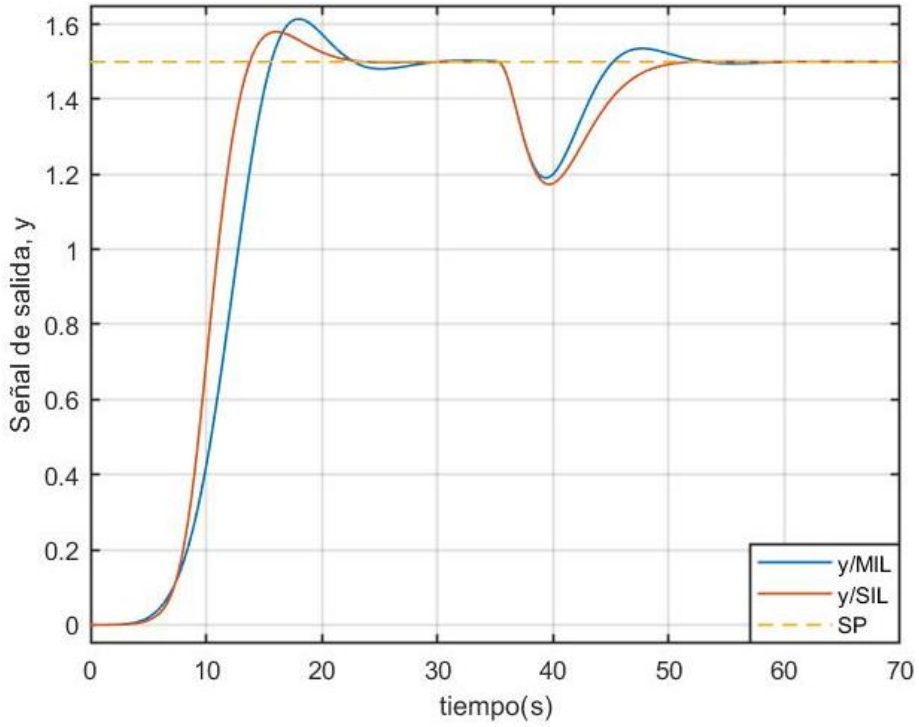


Tabla 12

Comparación de desempeño de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=1.5

Etapa	***Overshoot (%)***	***Tiempo de establecimiento (s)***	***Error en estado estacionario***
MIL	7.57	35	5.25×10^{-5}
SIL	5.33	33	4.36×10^{-6}

Para el setpoint de 1.5 mostrado en la Figura 63, el desempeño en la etapa SIL es superior al de la etapa MIL como se evidencia en la Tabla 12, la respuesta de la puesta en marcha virtual se ajusta al valor de referencia más pronto que el de la simulación de modelos.

Se debe destacar que en la etapa SIL en respuesta a la perturbación que ocurre a los 35 segundos se ajusta más rápido y presenta menor sobre impulso.

Figura 64

Comparación de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=2.5

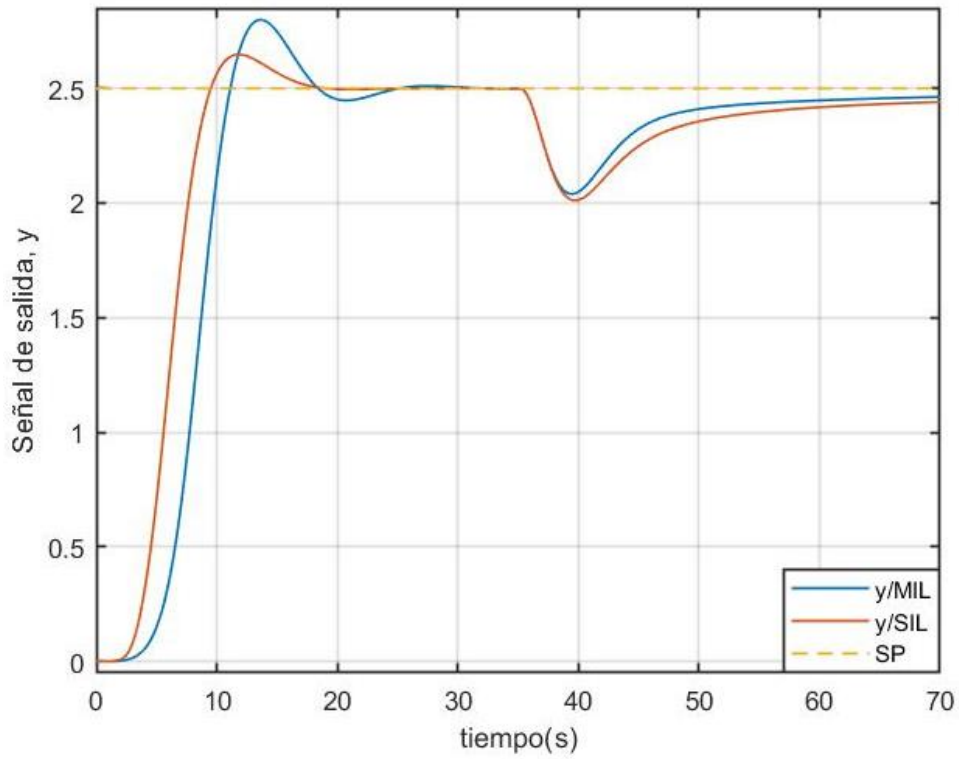


Tabla 13

Comparación de desempeño de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=2.5

Etapa	***Overshoot (%)***	***Tiempo de establecimiento (s)***	***Error en estado estacionario***
MIL	11.95	32	1.59×10^{-5}
SIL	5.90	27	4.72×10^{-5}

Similar a lo ocurrido en el anterior resultado, para el valor de referencia de 2.5 presentado en la Figura 64, el desempeño de la etapa SIL es mejor que el de la etapa MIL como se constata en la Tabla 13, nuevamente siendo más rápido en ajustarse al valor del setpoint. Los valores de error en estado estacionario son antes de que ocurra la perturbación. En ninguna de las respuestas de las dos etapas se ajusta el valor del error completamente a cero después de la perturbación.

Figura 65

Comparación de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=5

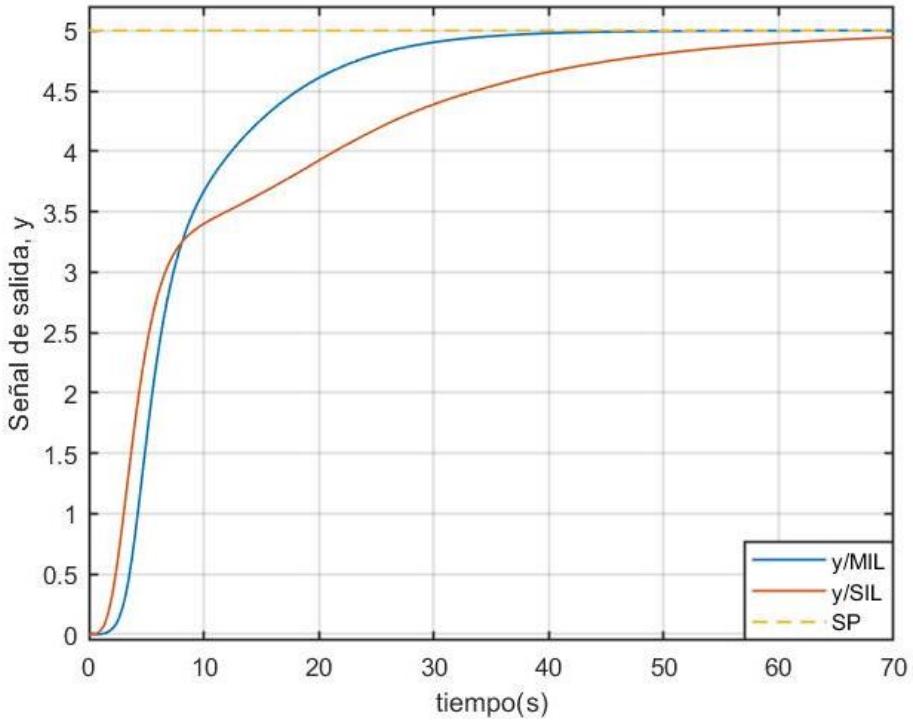


Tabla 14

Comparación de desempeño de las etapas MIL y SIL del primer caso de estudio para el valor de referencia SP=5

Etapa	***Overshoot (%)***	***Tiempo de establecimiento (s)***	***Error en estado estacionario***
MIL	N/A	50	2.60×10^{-4}
SIL	N/A	70	566×10^{-3}

Para el último valor de referencia la salida se satura muy pronto en ambas etapas como se presenta en la Figura 65, en este caso la respuesta de la etapa MIL es más deseable que la de la etapa SIL ya que no presenta la transición tan lenta al valor de referencia. Además, presenta un error en estado estacionario menor en la etapa MIL como muestra la Tabla 14.

En resumen, el controlador en su forma de código ejecutado sobre el PLC correspondiente a la etapa SIL no presenta ningún cambio drástico en cuanto a desempeño comparado con la simulación de modelos de la etapa MIL, e incluso mejora su rendimiento para los setpoints de $[1.5\ 2.5]$.

Caso de estudio reactor de tanque agitado

Para el modelo CSTR se debe comprobar que, para los puntos de operación de concentración del reactante A en el reactor, C_A, de 1 hasta 8 $[kmol/m^3]$, los controladores de las dos etapas se ajusten al valor de referencia.

Figura 66

Comparación de resultados de las etapas MIL y SIL del segundo caso de estudio

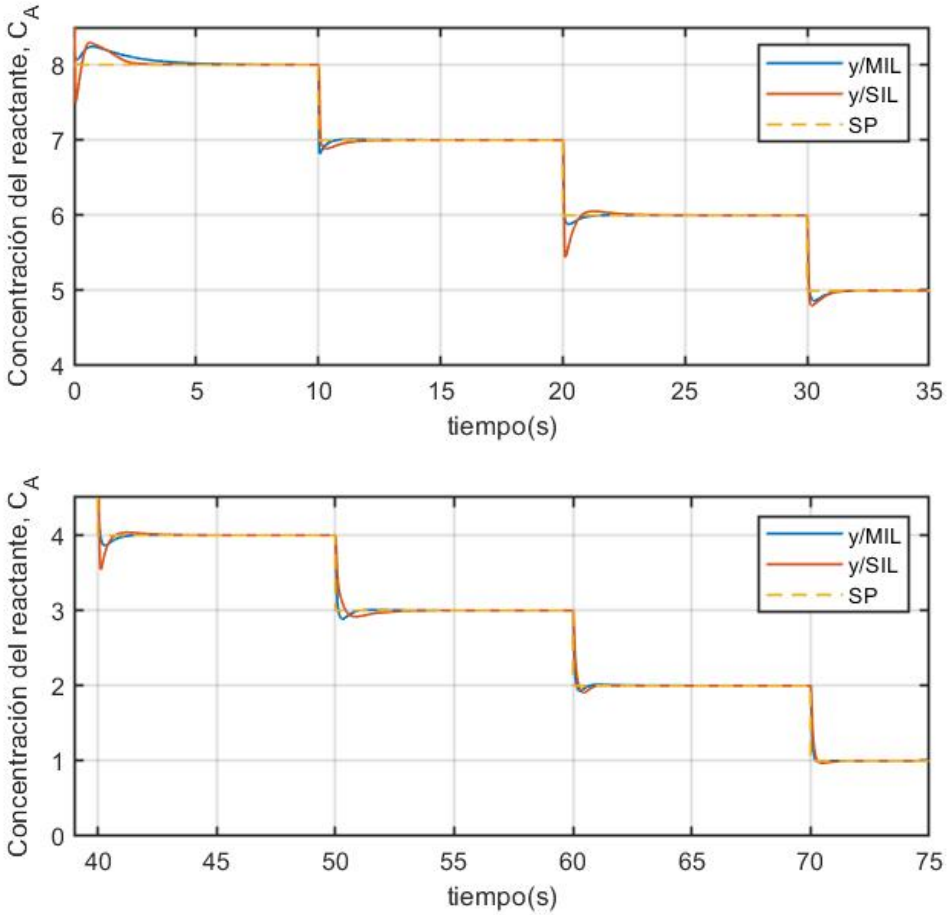


Tabla 15

Comparación de desempeño de las etapas MIL y SIL del segundo caso de estudio

$C_A[kmol/m^3]$	*Etapa*	*Overshoot (%)*	*Tiempo de establecimiento (s)*
8	*MIL*	3	6
	SIL	3.62	4
7	*MIL*	2.71	2.5
	SIL	1.71	3

6	*MIL*	2	3
	SIL	9.5	3.5
5	*MIL*	2.8	2.5
	SIL	4.2	2.5
4	*MIL*	3.5	2
	SIL	11.5	3
3	*MIL*	4	2
	SIL	3	3.5
2	*MIL*	4	2
	SIL	5	2
1	*MIL*	2	1
	SIL	3	2

De acuerdo a la Figura 66, tanto el resultado de la etapa MIL como el de la etapa SIL se ajusta a los valores de referencia, por lo que no fue necesario medir el error en estado estacionario, la transición entre setpoints es suave gracias al ajuste de las ganancias del controlador. Sin embargo, de acuerdo a la Tabla 15, para los setpoints de $6\ [kmol/m^3]$ y $4\ [kmol/m^3]$ la respuesta de la puesta en marcha virtual presenta mayor sobre impulso que la simulación de modelos.

El controlador de la etapa SIL no presenta una caída en desempeño notable en comparación con él de la etapa MIL, la respuesta en la puesta en marcha virtual se ajusta a cada uno de los puntos de operación manteniendo la transición suave del controlador por ajuste de ganancias y presenta dos leves sobre impulsos que no vuelven al sistema inestable.

Caso de estudio modelo de entrada/salida

Del modelo de (Narendra & Parthasarathy, 1990) se compara el rendimiento del controlador neuronal inverso de la etapa MIL con el controlador Neuro-Fuzzy implementado en la etapa SIL que atenuaba el efecto rizo sobre la señal de salida.

Figura 67

Comparación de resultados de las etapas MIL y SIL del tercer caso de estudio

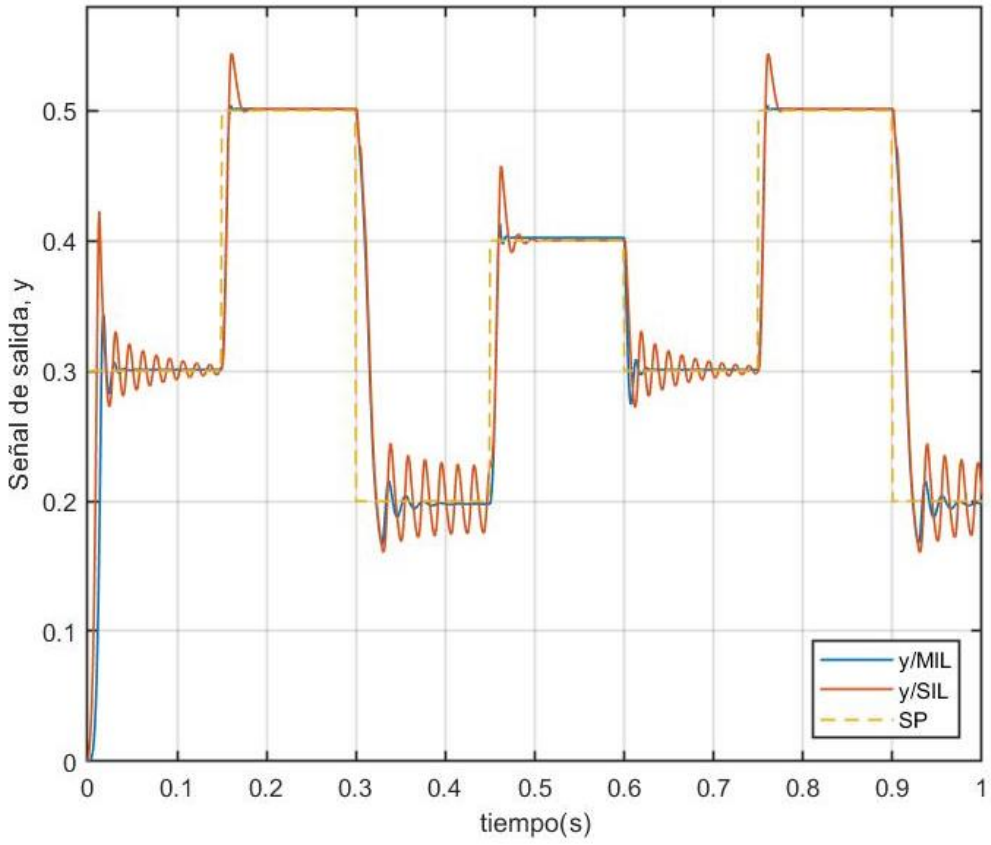


Tabla 16

Comparación de desempeño de las etapas MIL y SIL del tercer caso de estudio

Setpoint	*Etapa*	*Overshoot (%)*	*Tiempo de establecimiento (s)*	*Error en estado estacionario*	*¿Presenta Rizo?*
0.5	*MIL*	0.8	0.03	1.3×10^{-3}	No
	SIL	8	0.05	1.2×10^{-3}	No
0.4	*MIL*	2.49	0.035	2.4×10^{-3}	No
	SIL	12.49	0.062	1.87×10^{-4}	No
0.3	*MIL*	8.66	0.05	8.95×10^{-4}	No
	SIL	9.66	0.13	2.80×10^{-4}	Sí
0.2	*MIL*	17	0.125	2.4×10^{-3}	No
	SIL	20	-	-	Sí

De acuerdo a la Figura 68, tanto en la etapa SIL como MIL existe seguimiento al valor de referencia, sin embargo, es más notable la presencia del efecto rizo en la respuesta de la puesta en marcha virtual para los setpoints menores a 0.3 en comparación con la simulación de modelos. En ambas etapas se presenta el efecto rizo debido a las características propias del controlador que aproxima de manera no exacta a la función inversa del modelo entrada/salida. A pesar de que el sobre impulso y el tiempo de establecimiento es mejor en la etapa MIL, el error en estado estacionario para determinados setpoints es menor en la etapa SIL como se presenta en la Tabla 16. Aún con el efecto ringing que presenta para un valor de 0.2 como referencia, la respuesta de la etapa SIL no se vuelve inestable en ningún momento, y mantiene el seguimiento a múltiples setpoints como el controlador por redes neuronales.

Capítulo VI

Conclusiones, Recomendaciones y Trabajos Futuros

Conclusiones

- La metodología propuesta para el desarrollo de estrategias de control inteligente por medio de las herramientas de simulación in-the-loop, ratifica que estos algoritmos de control pueden ser implementados en entornos industriales.
- La generación de código de los tres controladores consiguió resultados positivos, puesto que el desempeño de la puesta en marcha virtual, SIL, reproduce las características obtenidas durante la simulación de modelos, MIL. Sin embargo, existieron algunas diferencias en el transitorio de las respuestas que no ocasionaron ninguna inestabilidad en los sistemas de control.
- El controlador difuso que actúa sobre el sistema del actuador no lineal muestra el mejor desempeño de los controladores, debido a que los resultados de la etapa SIL superan en rendimiento a los de la etapa MIL.
- El controlador por ajuste de ganancias que controla el modelo CSTR presenta el rendimiento más similar en ambas etapas de simulación, ya que las métricas de desempeño son la que menor variación tienen de los tres controladores.
- La estrategia de redes neuronales que controla el modelo entrada salida no lineal presenta un resultado adverso como el efecto ringing que se nota en ambas etapas de simulación, el cual es producido debido a la cancelación no exacta de los polos.

Recomendaciones

- Es recomendable optar por otra estrategia de control diferente al control neuronal inverso para el modelo entrada salida no lineal ya que presenta oscilaciones en su respuesta para determinados setpoints.
- Se recomienda que en la fase de diseño del controlador sean consideradas las necesidades del sistema de control, si bien los cambios en el diseño son fáciles de implementar gracias al proceso de simulación in-the-loop, es mejor que en el diseño inicial se tomen en cuenta todas las particularidades.
- Es recomendable tener cierto grado de familiaridad con el IDE hacia donde se exporta el código generado, puesto que desde este software se configura el controlador, protocolos de comunicación, sus registros de entradas/salidas, etc.
- Se recomienda antes de la ejecución de la puesta en marcha virtual se haya verificado que el periodo de muestreo sea el mismo tanto en la exportación de código en Simulink como en la interrupción cíclica en TIA Portal, que la dirección IP del controlador sea la misma en la función definida por el usuario, que el nombre de los tags de los registros entrada/salida del PLC sean iguales en la función que permite el intercambio de datos.

Trabajos Futuros

Otras estrategias de control como, por ejemplo, control por modelo de referencia o control adaptativo son propuestas que pueden ser implementadas en el modelo CSTR. Para el sistema del actuador no lineal es posible reproducir de mejor manera la función inversa de la válvula a partir de un mayor número de funciones lineales.

En este proyecto se ha establecido una metodología para la implementación de estrategias de control inteligente mediante las dos primeras etapas del proceso de simulación

in-the-loop, sin embargo, si se busca completar el proceso de simulación se debe integrar la etapa HIL donde el controlador es ejecutado en una plataforma de hardware física. Cumpliendo con esta etapa se tendrá una noción clara de cómo está operando el software embebido en la plataforma de hardware, tomando en consideración que los tiempos de respuesta se den en intervalos determinados consiguiendo así comportamiento en tiempo real.

Las simulaciones realizadas en las etapas MIL y SIL dan una noción de cuánto tiempo debería tomar la obtención de respuestas en la etapa HIL. El desafío en la siguiente etapa de simulación será solventar los problemas que puedan ocurrir sobre el hardware real como módulos de entradas/salidas, o interfaces que se usen para la comunicación entre el controlador físico y la planta que aún reside en Simulink.

Fuentes Bibliográficas

Aluisa, P. (2014). *DESARROLLO DE CONTROLADORES CON REDES NEURONALES DE APRENDIZAJE PROFUNDO APLICANDO TENSORFLOW.* Sangolquí: ESPE.

Åström, K., & Björn, W. (2008). *Adaptive Control.* New York: Dover Publications.

De La Concha-Gómez, A., Ramírez-Muñoz, J., Márquez-Baños, V., Haro, C., & Alonso-Gómez, A. (2019). EFFECT OF THE ROTATING REFERENCE FRAME SIZE FOR SIMULATING A MIXING STRAIGHT-BLADE IMPELLER IN A BAFFLED STIRRED TANK. *Revista Mexicana De Ingeniería Química*, 1143-1160.

Dijkstra, B., & Marchant, R. (2020). *Virtual Commissioning with Simulink.* MathWorks.

Gordillo, R. (2020). *Control Inteligente.*

Jbair, M., Ahmad, B., Mus'ab, A., Vera, D., Harrison, R., & Ridler, T. (2019). *Automatic PLC Code Generation Based on Virtual Engineering Model.* University of Warwick.

Jones, T. (5 de diciembre de 2017). *Models for machine learning.* Obtenido de IBM Developer: https://developer.ibm.com/articles/cc-models-machine-learning/

MathWorks. (2021). *Virtual Commissioning with Model-Based Design.*

MathWorks. (s.f.). *CSTR Model*. Obtenido de https://www.mathworks.com/help/mpc/gs/cstr-model.html

MathWorks Support Team. (2 de enero de 2021). *What is MIL, SIL, PIL, HIL and how do they integrate in Model Based Design approach?* (MathWorks) Recuperado el 21 de junio de

2021, de https://www.mathworks.com/matlabcentral/answers/440277-what-is-mil-sil-pil-hil-and-how-do-they-integrate-in-model-based-design-approach

Matich, D. (2001). *Redes Neuronales: Conceptos Básicos y Aplicaciones.* Rosario: Universidad Tecnológica Nacional – Facultad Regional Rosario .

Nacelle, A. (2009). *Redes neuronales artificiales.* Montevideo: Universidad de la Republica. Obtenido de Universidad de la República: http://www.nib.fmed.edu.uy/Seminario%202009/Monografias%20seminario%202009/Nacell-Redes%20NeuronalesImplementacion.pdf

Narendra, K., & Parthasarathy, K. (1990). *Identification and Control of Dynamical Systems Using Neural Networks.* IEEE TRANSACTlONS ON NEURAL NETWORKS. VOL. I. NO. I.

Nelles, O. (2001). *Introduction. In: Nonlinear System Identification.* Obtenido de https://doi.org/10.1007/978-3-662-04323-3_1

Peña, E., Pérez, A., Miranda, A., & Sánchez, J. (2008). *Modelado de un reactor químico tipo CSTR y evaluación del control predictivo aplicando Matlab-Simulink .* Valencia: INGENIERÍA UC.

Petkov, P., Slavov, T., & Kralev, J. (2018). *Design of Embedded Robust Control Systems Using Matlab/Simulink.* Londres: The Institution of Engineering and Technology.

Pham, D. T., & Xing, L. (2018). *Neural Networks for Identification, Prediction and Control.* Cardiff: Springer .

Ponce, P., & Ramírez, F. (2010). *Intelligent Control Systems with LabVIEW.* Ciudad de México: Springer.

Portillo, E., & Ordoñez, J. (2020). *Design and Implementation of a Fuzzy Controller for Frequency Control of a Motor in an S7-1200 PLC.* 18th LACCEI International Multi-Conference for Engineering, Education, and Technology.

Reyes, V. (2014). *Virtual Hardware "In-the-Loop": Earlier Testing for Automotive Applications.* Mountain view: Synopsys.

Salvador, C. (2007). *Diseno e implementaci on de un controlador difuso con ganancia auto-ajustable para un intercambiador de calor.* Monterrey: Instituto Tecnologico y de Estudios Superiores de Monterrey.

Sánchez, P. (2020). *DESARROLLO DE CONTROLADORES CON REDES NEURONALES DE APRENDIZAJE PROFUNDO APLICANDO TENSORFLOW.* Sangolquí: ESPE.

SIEMENS. (s.f.). *Industrial Automation Systems SIMATIC* . Obtenido de Artificial intelligence in SIMATIC: https://new.siemens.com/us/en/products/automation/systems/industrial/io-systems/artificial-intelligence.html

Toro, V., Garzón, J., & López, J. (2009). *Control Neuronal por Modelo Inverso de un Servosistema Usando Algoritmos de Aprendizaje Levenberg-Marquardt y Bayesiano.* Cartagena: VIII Congreso de la Asociación Colombiana de Automática.

Wittmann, C. (2020). *Comparison of the Deviations between MiL, SiL and HiL Testing.* Technische Hochschule Ingolstad.

Apéndices