

UNIVERSIDAD RICARDO PALMA
FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL DE INGENIERÍA
ELECTRÓNICA



DISEÑO DE UN CONTROLADOR PID CON
AUTOSINTONÍA BASADO EN UN MODELO DE REDES
NEURONALES DINÁMICAS Y CONTROL ADAPTATIVO

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE
INGENIERO ELECTRÓNICO

PRESENTADA POR:

BACHILLER: CARDOZO GALVEZ, ERICK OCTAVIO

ASESOR: ING. CHONG RODRÍGUEZ, HUMBERTO

LIMA-PERÚ

2018

DEDICATORIA:

Para Epita, Kimi y Ana Paula; el antes y los después de mi bello presente.

AGRADECIMIENTO:

A mis docentes universitarios, por incentivar el interés en la investigación y estudio de esta ciencia.

ÍNDICE GENERAL

RESUMEN	xiii
ABSTRACT.....	xiv
INTRODUCCIÓN.....	1
CAPÍTULO I: PLANTEAMIENTO DEL ESTUDIO.....	4
1.1. Marco Situacional.....	4
1.2. Problematización	6
1.2.1. Problema general:.....	7
1.2.2. Problemas específicos:	8
1.3. Objetivos.....	8
1.3.1. Objetivo General:	8
1.3.2. Objetivos Específicos:	8
1.4. Importancia y justificación.	9
1.5. Alcances y limitaciones.	10
CAPÍTULO II: MARCO TEÓRICO.....	11
2.1. Antecedentes de la investigación.....	11
2.2. Óptica de la investigación.	12
2.3. Selección de variables.	12
2.3.1. Variables independientes:.....	12
2.3.2. Variables dependientes:.....	13
2.4. Fundamentos de las redes neuronales artificiales.....	13
2.4.1. ¿Qué es una red neuronal?.....	13
2.4.2. Neurona biológica:	13
2.4.3. Neurona Artificial:	15

2.4.4. Arquitectura de una red neuronal artificial.....	17
2.4.5. Red Neuronal Estática:	20
2.4.6. Red Neuronal Dinámica:	30
2.4.7. Entrenamiento de la red neuronal dinámica.	32
CAPÍTULO III. DISEÑO DEL CONTROLADOR CON AUTOSINTONÍA PID	
BASADO EN MODELO Y CONTROL ADAPTATIVO	49
3.1. Identificación de sistemas con redes neuronales dinámicas.....	49
3.1.1. Descripción del proceso de la planta con ganancia variable usada para la identificación del sistema.	49
3.1.2. Obtención del modelo del proceso aplicando redes neuronales dinámicas e identificación de sistemas.....	56
3.1.3. Validación del modelo.....	60
3.2. Diseño e implementación del controlador con autosintonía PID basado en modelo.....	62
3.2.1. Control adaptativo basado en modelo de referencia (MRAC).....	62
3.2.2. Autosintonía PID usando redes neuronales y control adaptativo basado en modelo de referencia.	64
3.2.3. Arquitectura del sistema.	64
3.2.4. El “ Jacobiano”	71
3.2.5. Implementación del controlador PID autosintonizado usando redes neuronales y control adaptativo en Simulink – Matlab.....	77
3.2.6. Resultados.	82
3.2.7. Efectos de los valores iniciales de los pesos sinápticos.	84
3.2.8. Efecto del modelo de referencia.	84
CAPÍTULO IV: ANÁLISIS DE ROBUSTEZ Y PROPUESTA DE	
IMPLEMENTACION DEL CONTROLADOR DISEÑADO	86

4.1. Controlador PID autosintonizado con redes neuronales y controla adaptativo frente a una perturbación.	86
4.2. Comparación del controlador PID autosintonizado con redes neuronales y control adaptativo versus un controlador PID clásico.	86
4.3. Propuesta de implementación práctica del controlador diseñado.	89
4.4. Diagrama P&ID para el control del sistema.	91
4.5. Sistema de control y arquitectura de control.	91
CONCLUSIONES	93
RECOMENDACIONES.	96
REFERENCIAS BIBLIOGRÁFICAS	98
ANEXOS	101
Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas (continúa).	101
Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas (continúa).	102
Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas (continúa).	103
Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas (continúa).	104
Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas.	105
Anexo 2: Script de Matlab que desarrolla la validación del sistema (continúa).	106
Anexo 2: Script de Matlab que desarrolla la validación del sistema (continúa).	107
Anexo 2: Script de Matlab que desarrolla la validación del sistema.	108
Anexo 3: Script Matlab que desarrolla la identificación del sistema con redes neuronales estáticas (continúa).	109

Anexo 3: Script Matlab que desarrolla la identificación del sistema con redes neuronales estáticas (continúa).....	110
Anexo 3: Script Matlab que desarrolla la identificación del sistema con redes neuronales estáticas.	111
Anexo 4: Desarrollo del Controlador PID autosintonizado con redes neuronales dinámicas y control adaptativo.....	112
Anexo 5: Planta desarrollada en Simulink para la identificación del sistema.	113

ÍNDICE DE TABLAS

Tabla 1: Indicadores de proximidad del modelo neuronal hallado.....	60
Tabla 2: Indicadores de proximidad del modelo neuronal hallado después de la validación.....	62
Tabla 3: Indicadores ITAE e IAE de los sistemas con un PID autosintonizado con red neuronal y un PID convencional.	89
Tabla 4: Listado de los componentes del sistema de control sugerido.....	91

ÍNDICE DE FIGURAS

Figura 1: Comparativa entre un controlador con redes neuronales y controlador PID clásico.	5
Figura 2: Comparativa de un control PID vs un control robusto basado en redes neuronales.	7
Figura 3: Planta utilizada por Pirabakaran and V. M. Becerra.	11
Figura 4: Neurona biológica.	14
Figura 5: Sinapsis de una neurona biológica.	14
Figura 6: Neurona Artificial.	15
Figura 7: Funciones de activación de una neurona artificial.	16
Figura 8: Estructura de una red neuronal.	17
Figura 9: Arquitectura de red neuronal estática.	17
Figura 10: Función sigmoidea tipo 2.	19
Figura 11: Representación geométrica del algoritmo descenso del gradiente.	22
Figura 12: Retropropagación del error – red neuronal.	24
Figura 13: Arquitectura de la red neuronal dinámica	31
Figura 14: Arquitectura red neuronal dinámica expandida	32
Figura 15: Datos de entrada salida red neuronal dinámica	32
Figura 16: Derivada total de $dx1 * dw$	33
Figura 17: Derivada total de $dx2 * dw$	34
Figura 18: Derivada total de $dx3 * dw$	35

Figura 19: Estructura red neuronal dinámica con dos salidas.	36
Figura 20: Retropropagación de las salidas hacia las entradas pasadas, para hallar la matriz jacobiana.	42
Figura 21: Respuesta de la identificación con una red neuronal estática, se puede observar que identifica hasta el ruido presente en el sistema.	47
Figura 22: Respuesta de la identificación con una red neuronal dinámica, se puede observar que elimina al ruido durante el modelamiento.	47
Figura 23: Reactor usado para la neutralización de pH.	50
Figura 24: Respuesta de la planta a una señal escalón del caudal de NaOH.	50
Figura 25: Señales tipo escalón inyectadas a la entrada del sistema.	51
Figura 26: Respuesta del sistema a las entradas escalón. observándose una ganancia estática variable.	51
Figura 27: Diagrama de bloques del modelo completo de la planta pH.	52
Figura 28: Planta a modelar y controlar en el presente trabajo.	53
Figura 29: Entradas tipo escalón al ingreso de la planta.	54
Figura 30: Respuesta pH de la planta, se observa la ganancia estática variable.	54
Figura 31: Señal tipo escalón presentado al sistema.	55
Figura 32: Respuesta de la planta a la señal tipo escalón.	55
Figura 33: Arquitectura de la red neuronal para la identificación del sistema	57
Figura 34: Señales de entrada y salida para la identificación del sistema. Se observa la generación de ruido a la salida.	58
Figura 35: Señal PRBS sin escalar empleada para generar la salida de pH para la identificación del sistema.	58

Figura 36: Salida de la planta sin escalar. Valor comprendido de pH entre 2.5 y 4.5....	59
Figura 37: Señal PRS empleada para generar la salida de pH para la validación del sistema.	61
Figura 38: Comparación salida pH de la planta VS salida pH del modelo neuronal	62
Figura 39: Controlador adaptativo básico.....	63
Figura 40: Diagrama de bloques de con control adaptivo basado en modelo de referencia.	64
Figura 41: Arquitectura autosintonía PID con redes neuronales y control adaptativo. ...	65
Figura 42: Red neuronal para la autosintonía PID (Sintonizador neuronal).....	65
Figura 43: Sintonizador neuronal. Red neuronal estática para hallar parámetros PID...	68
Figura 44: Forma adoptada para la retropropagación en la red neuronal sintonizadora.	69
Figura 45: NN emulador. Retropropagación de la salida y $\hat{y}(n+1)$ estimada hasta la entrada $U(n)$, para el cálculo del “jacobiano”.....	73
Figura 46: Arquitectura del sistema con controlador PID autosintonizado.....	77
Figura 47: Bloque NN emulador	78
Figura 48: Bloque Planta, que representa al sistema a controlar.	78
Figura 49: Bloque PID, K_p , K_i y K_d son calculados por el sintonizador neuronal.....	79
Figura 50: Bloque modelo de referencia.....	80
Figura 51: Bloque “jacobiano”.	80
Figura 52: Bloque du/dq	81
Figura 53: Desarrollo sintonizador neuronal.	81
Figura 54: Salida de la planta pH vs. Set point.....	82

Figura 55: Salida de la planta pH vs. Set point periódico.....	83
Figura 56: Respuesta del sistema a distintas señales tipo escalón.....	83
Figura 57: Efecto de elección de valores iniciales de los pesos sinápticos	84
Figura 58: Respuesta ante el modelo de referencia con tiempo de establecimiento de 500 seg.	85
Figura 59: Respuesta ante el modelo de referencia con tiempo de establecimiento de 4000 seg.	85
Figura 60: Respuesta del sistema ante una perturbación.	86
Figura 61: Sistema pH controlado con PID convencional.....	88
Figura 62: Respuesta del sistema a consignas variables y presencia de perturbación....	88
Figura 63: Instrumentación sugerida para el control del sistema.	90
Figura 64: Diagrama P&ID para el control del sistema.....	91
Figura 65: Arquitectura para el control de la plana de neutralización de pH	92

RESUMEN

La presente tesis explicó el desarrollo y la implementación de un controlador PID con autosintonía basado en un modelo de redes neuronales dinámicas y control adaptativo para el control de una planta con ganancia estática variable. Se tomó el concepto de redes neuronales como principio para el aprendizaje y desarrollo de las redes neuronales dinámicas, las cuales, por la arquitectura y método que usa, y como se comprobó, ofrecieron mayor inmunidad al ruido que las redes estáticas. La red neuronal dinámica sirvió para determinar el modelo de la planta a controlar, a su vez, entregó el cálculo del “jacobiano” a la arquitectura neuronal (esta será de tipo estática), a la cual se denominó “sintonizador neuronal”, que fue la encargada de desarrollar los parámetros óptimos K_p , K_i y K_d que luego fueron entregados al controlador PID de tipo velocidad. El aprendizaje del sistema en su conjunto, dependió de la señal del error generada por la diferencia entre una señal de comando (set point) y la salida de un modelo de referencia de segundo orden, el cual, aplicando conceptos de control adaptativo, ha guiado el comportamiento de la señal controlada del sistema.

La planta controlada se fundamentó en la dinámica de una planta de neutralización de pH, la cual presentó una ganancia estática variable en su rango de operación, lo que marcó la no linealidad existente en la planta. El modelo de la planta como el controlador PID autosintonizado fueron simulados en su conjunto por computadora haciendo uso de Matlab, para su evaluación y muestra de resultados de control.

Se comprobó en el desarrollo de esta tesis que el controlador PID autosintonizado desarrollado ofrece mayor eficacia que un controlador PID convencional.

Palabras clave: Redes neuronales estáticas, redes neuronales dinámicas, control adaptativo, jacobiano, controlador PID.

ABSTRACT

This thesis explained the development and implementation of a PID controller with auto-tuning based on a dynamic neural network model and adaptive control for the control of a plant with variable static gain. The concept of neural networks was taken as a principle for the learning and development of dynamic neural networks, which, by the architecture and method used, and as it was proved, offered greater immunity to noise than static networks. The dynamic neuronal network served to determine the model of the plant to control, in turn, gave the calculation of the "Jacobian" to the neuronal architecture (this will be of static type), which was called "neuronal tuner", which was the one in charge to develop the optimal parameters K_p , K_i and K_d that soon were given to the controller PID of type speed. The learning of the system as a whole, depended on the signal of the error generated by the difference between a command signal (set point) and the output of a reference model of second order, which, applying adaptive control concepts, has guided the behavior of the controlled signal of the system.

The controlled plant was based on the dynamics of a pH neutralization plant, which presented a variable static gain in its operating range, which marked the non-linearity of the plant. The model of the plant as the self-tuned PID controller were simulated as a whole by computer using Matlab, for its evaluation and sample of control results. It was found in the development of this thesis that the self-tuned PID controller developed offers greater efficiency than a conventional PID controller.

Keywords: Static neural networks, dynamic neural networks, adaptive control, Jacobian, PID controller.

INTRODUCCIÓN

La automatización y control de los procesos industriales, como es sabido, está teniendo mayor relevancia dentro de la industria, ya que esta ciencia ha comprobado mejorar la optimización de dichos procesos, viéndose reflejado en el incremento de la producción, ahorro de energía, en la mejora de las capacidades de los empleados (requieren gente con una buena capacitación técnica y profesional), etc., y con todo esto una ganancia económica para la industria que la emplea.

Como se menciona, la automatización y control de procesos está tomando mayor participación para las mejoras de la producción, pero dentro de ella, también se está viendo una evolución, tanto en las mejoras de la tecnología que emplea, como en los algoritmos de programación que está utilizando. Dentro de este campo (programación) hoy en día no sorprende escuchar y ver el uso de redes neuronales, lógica difusa, algoritmos genéticos, etc., para poder alcanzar y mejorar los objetivos de control. Estas nuevas técnicas ya se encuentran siendo aplicadas en el campo de identificación de sistemas, predicción de procesos o la teoría de control lineal y no lineal; haciendo que las técnicas de control clásicas usadas, como es el caso del control PID, sean poco a poco relegadas por la poca capacidad de optimización que tienen en procesos no lineales por ejemplo, o en la demora que se tiene para encontrar los parámetros óptimos de control o por las posibles pérdidas de insumos durante la sintonía de sus parámetros.

Si bien el control PID clásico se encuentra alrededor del 70% en uso de todos los procesos industriales a nivel mundial, los nuevos algoritmos (redes neuronales, lógica difusa, etc.) están ganando mayor participación principalmente por tres motivos: la teoría es universalmente conocida y compartida; el hardware que se está fabricando ya puede ejecutar tareas complicadas para llevar a cabo estos algoritmos; se comprueba que utilizando estos métodos, los objetivos de control son más óptimos que siendo realizado por un control clásico.

Ahora bien, cabe indicar que estas estrategias, métodos y algoritmos, no es la solución definitiva para el control de todo tipo de proceso, ya que para ello el profesional encargado tiene que tener la capacidad de analizar y entender el comportamiento

dinámico del proceso, pues un control PID clásico puede ser suficiente (como el control de nivel – sistema lineal) para alcanzar los objetivos de control.

De este análisis, se tiene que determinar (por métodos de identificación de sistemas) el tipo de proceso que se tiene, y con esto, diseñar e implementar el controlador adecuado.

Dentro de este contexto, la Universidad Ricardo Palma con la participación de su Escuela de Ingeniería Electrónica, no se encuentra ajena a estos desarrollos, por tal motivo, el tema de esta tesis, se centra en el diseño e implementación de un controlador con auto sintonía de sus parámetros haciendo uso de la teoría de identificación de sistemas, redes neuronales y control adaptativo. Dicho controlador será utilizado y evaluado en una planta de segundo orden con discontinuidades en su ganancia, planta en la cual un control PID clásico no es muy eficiente.

En el primer capítulo, denominado planteamiento del estudio, se enfocará el análisis situacional del tema planteado y se justificará la elaboración del proyecto; también se mostrará los objetivos, alcances y limitaciones que abarca la presente tesis.

En el segundo capítulo se aborda los conceptos teóricos de las redes neuronales artificiales, donde se explica las formas de entrenamiento de las redes neuronales estáticas y dinámicas y las diferencias que marcan el comportamiento de cada una de ellas.

En el tercer capítulo, se desarrolla el diseño del controlador con autosintonía PID basado en modelo y control adaptativo, para ello se describirá el comportamiento dinámico de la planta, el procedimiento de identificación de la planta con redes neuronales dinámicas, la validación del modelo identificado y finalmente se mostrará el proceso del diseño del controlador autosintonizado PID aplicando control adaptativo y el concepto de redes dinámicas estáticas.

En el cuarto capítulo, se analizará la robustez del controlador diseñado, para lo cual se hará una comparativa del controlador diseñado con un controlador PID convencional aplicando conceptos de la integral del error absoluto (IAE) y de la integral del tiempo por el error absoluto (ITAE), se verán las ventajas del controlador PID autosintonizado;

finalmente se presenta una propuesta de implementación práctica del controlador diseñado para su uso en sistemas reales.

CAPÍTULO I: PLANTEAMIENTO DEL ESTUDIO

1.1. Marco Situacional

Uno de los objetivos principales de las empresas industriales a nivel mundial es la optimización de sus procesos. Esto, con el propósito de obtener una mayor producción y con ello, un mayor beneficio para toda su institución (económico, social, etc.); para alcanzar este objetivo, dirigen sus esfuerzos en mejorar los tiempos de ejecución de sus tareas, ahorros en la mano de obra, ahorro en los costos y gastos de insumos, entre otros.

Para poder alcanzar dichos objetivos, una herramienta primordial es la automatización y control de sus procesos fundamentales y/o críticos, por lo que las empresas ya contemplan una inversión y planificación económica para dicha implementación.

Hoy en día, a nivel industrial existen una diversidad de estrategias de control tales como el control PID, control en cascada, control predictivo, control de razón, etc.; siendo el control PID el más ampliamente usado por su sencillez e intuición de uso. Se considera que el uso del PID en la industria, actualmente abarca cerca del 90% para lazos de control cerrado a nivel mundial.

Pero, lamentablemente, el control PID no es la solución universal para el control a todo proceso, porque hay variables a considerar para su uso, tales como la linealidad del proceso, tiempos de retardo, perturbaciones, cantidad de entradas y salidas del proceso (sistemas de entradas y salidas múltiples), etc. Dentro de este contexto un controlador PID da buenos resultados en sistemas lineales con una entrada y salida en su dinámica y con poco retardo de tiempo.

Considerando estas variables, es que se desarrollaron nuevos métodos y estrategias de control, tales como el control con redes neuronales, control difuso, control neuro-difuso, algoritmos genéticos, etc., para procesos o etapas dentro de ella que contienen no linealidades, grandes retardos, perturbaciones, o múltiples entradas y salidas.

Las nuevas y continuas investigaciones realizadas en estos campos, ha facilitado la difusión de estas para ser utilizadas en la teoría de control. El uso de estos nuevos

métodos en el control de procesos, ha permitido dar solución con alta eficiencia en plantas no lineales, que presentan grandes retardos, que tengan perturbaciones o sean multivariantes, las cuales llamaremos plantas complejas. Lamentablemente, el costo computacional por la matemática involucrada en estas estrategias no ha permitido su despegue definitivo, pues emplea algoritmos complejos que necesitan un soporte hardware todavía muy costosos hoy en día, además que requieren de operadores más capacitados (conocedores de la planta y con nociones de programación). En la figura 1 se muestra una comparativa de un sistema de control con redes neuronales y un PID clásico

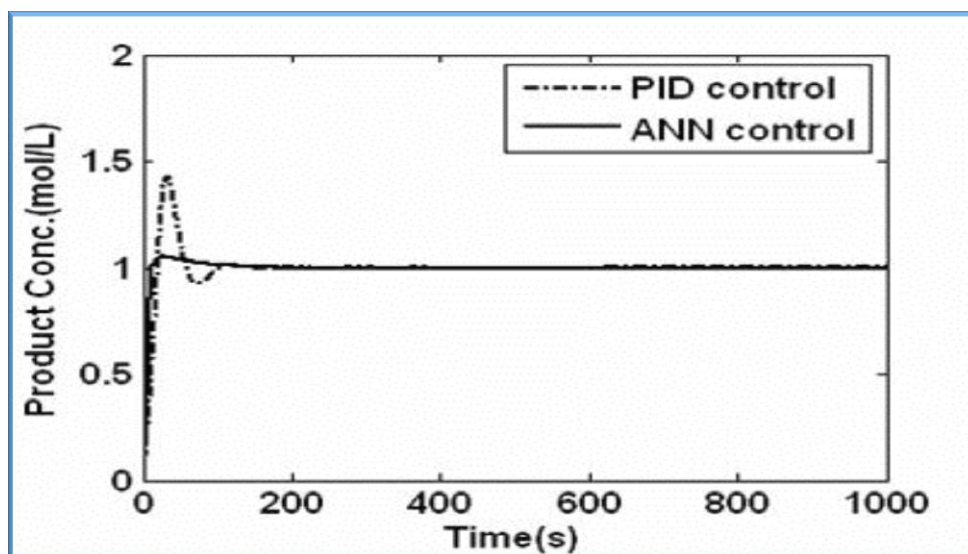


Figura 1: Comparativa entre un controlador con redes neuronales y controlador con PID clásico.
Fuente: Neural Network Control of CSTR for Reversible Reaction Using Reverence Model Approach. D. Araromi, T. Afolabi, D. Aloko.

Ahora bien, los procesos industriales reales, siempre presentarán no linealidades, perturbaciones que pueden perjudicar o impedir el control adecuado del mismo.

En tal sentido, un profesional en el tema tiene que ser capaz de identificar bien el tipo de proceso a controlar para luego diseñar y aplicar el método y estrategia de control adecuado.

Dentro de este marco situacional, se puede sintetizar que:

Toda industrial hoy en día, destina un presupuesto para automatizar y mejorar sus procesos.

A nivel mundial el uso del PID abarca casi el 90% de lazos cerrados de control, aun en plantas complejas, debido a su facilidad de uso y el poco conocimiento de la dinámica de la planta que se debe tener. Todo esto pese a ser un método de control no óptimo.

Las investigaciones realizadas en redes neuronales, neuro-difusas, algoritmos genéticos, etc., están permitiendo que estas puedan ser incorporadas dentro de estrategias avanzadas de control, como por ejemplo en control predictivo o adaptativo basado en modelos. Teniendo la desventaja que una implementación práctica-profesional de estos sistemas resultan todavía costosos.

En nuestra formación universitaria, tuvimos la orientación adecuada para poder tener un buen criterio en la identificación y metodología a utilizar en procesos con alta dificultad a controlar, ya sea por la no linealidad que presenta o por ser un sistema con múltiples entradas y salidas, sistemas en la cual un control PID clásico no sería eficiente.

El presente trabajo, investiga, diseña y evalúa un controlador autosintonizado PID basado en redes neuronales dinámicas y control adaptativo para una planta de segundo orden con discontinuidades en su ganancia; la auto sintonía para encontrar los parámetros K_p , K_i y K_d indica que será des forma automática. Se utilizará la teoría de redes neuronales dinámicas (algoritmo backpropagation) para la identificación y obtención del modelo del sistema y de control adaptativo (basado en modelo) para el control base del sistema, haciendo uso del modelo obtenido con las redes neuronales.

1.2. Problematización

Actualmente, en los procesos industriales, los métodos y estrategias de control que se utilizan, no contemplan al detalle las variables de la planta o proceso a controlar, tales como son las dinámicas lineales y no lineales del proceso, las variables de estado, tipos y cantidades de entradas y salidas del sistema, etc.

Dentro este contexto, la estrategia mayormente utilizada es el control PID, utilizada en más de un 90% en las plantas industriales, tanto por su sencillez como por su intuición en el uso. Pero en sistemas que presentan variaciones o no linealidades en sus parámetros de operación, el control PID sufre un duro revés, ya que este opera únicamente dentro de un rango lineal. Ante este problema, con controlador PID, es

necesario realizar nuevamente una nueva sintonía de los parámetros K_p , K_i y K_d del trayendo consecuencias negativas como atrasos en la producción para encontrar los parámetros óptimos, pérdida de insumos durante la sintonía PID, o la dependencia de un operador conocedor del proceso, que como se sabe no siempre se encuentra presente en la planta.

La figura 2, muestra la performance entre el control realizado usando controladores avanzados versus control PID convencional.

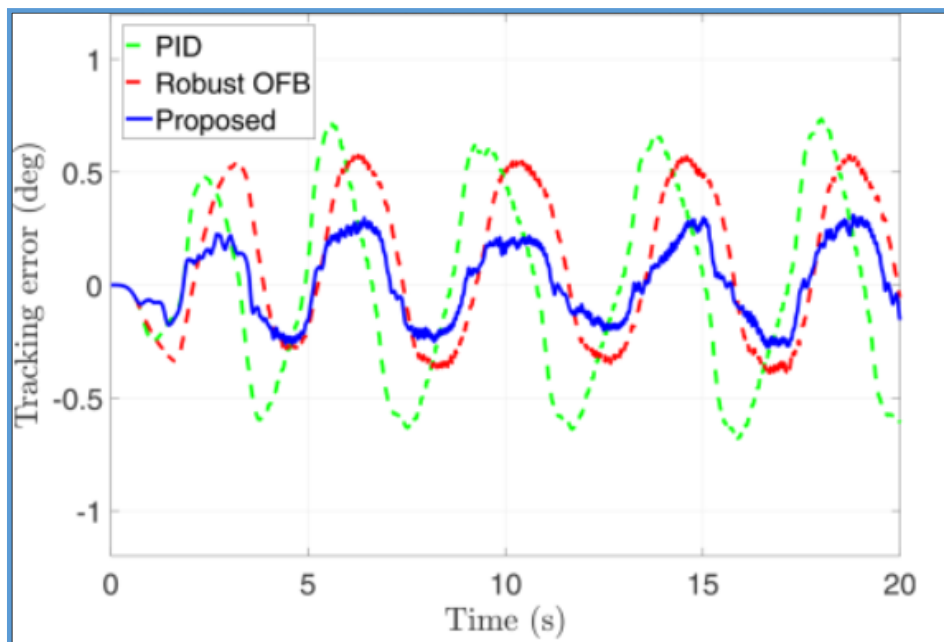


Figura 2: Comparativa de un control PID vs un control robusto basado en redes neuronales.

Fuente: Dynamic Neural Network-Based Output Feedback Tracking Control for Uncertain Nonlinear Systems. H. Dinh, S. Bhasin, R. Kamalapurkar and W. Dixon.

De lo expuesto, podemos sintetizar los siguientes problemas para el control de sistemas que presentan algún grado de no linealidad o cambios en sus parámetros de operación.

1.2.1. Problema general:

- En plantas industriales que presentan una dinámica compleja, se considera todavía el uso del controlador convencional PID, el cual no tiene una buena eficiencia para el control de este tipo de procesos, ya que a cambios de los parámetros de operación, se tiene que realizar nuevas sintonías. Es decir, opera dentro de pequeños rangos de operación (rango lineal), fuera de este, se tiene que hallar nuevos parámetros para el control de la planta (K_p , K_i , K_d).

1.2.2. Problemas específicos:

- En la gran mayoría de plantas industriales, no se conoce el comportamiento dinámico del proceso, haciendo uso netamente de la intuición del operador, esto ocasiona cambios de los parámetros de control de forma constante resultando que las variables controladas no alcancen sus objetivos (set point) fijados.
- Frente al desarrollo tecnológico que se tiene hoy en día, otro problema es que no se aprovecha de esta para implementar estrategias de control avanzadas que permitan un control óptimo de procesos complejos. Pues está comprobado que el uso de estos controladores avanzado optimizan todos los recursos empleados dentro de un proceso.
- Otro problema presentado es que actualmente un 95% de plantas industriales sintonizan sus parámetros de control basados en ensayos de prueba y error (tal como un PID convencional), este problema ocasiona pérdidas de recursos durante ese tipo de sintonización. Teniendo un modelo del proceso y un controlador avanzado adecuado, se puede conseguir parámetros de control óptimos por medio de simulaciones en computadora, de esta manera se optimiza el control del proceso real y con ello el ahorro de costos que pueden resultar de una mala sintonización de parámetros de control.

1.3. Objetivos.

1.3.1. Objetivo General:

- Diseñar un controlador autosintonizado PID, basado en redes neuronales dinámicas y control adaptativo, para la sintonía óptima de los parámetros de control en un proceso.

1.3.2. Objetivos Específicos:

- Identificar el modelo de una planta de segundo orden con discontinuidades en su ganancia; haciendo uso de redes neuronales dinámicas para conocer el comportamiento dinámico del sistema.
- Desarrollar un controlador avanzado, aplicando la teoría de control adaptativo basado en modelo para realizar la adaptación automática de los parámetros K_p , K_i y

Kd del controlador PID, para optimizar los recursos involucrados dentro de un control de procesos.

- Aplicar los algoritmos desarrollados en una planta simulada de segundo orden con discontinuidades en su ganancia y observar las mejoras y/o ventajas obtenidas con el algoritmo en comparación con un PID clásico.

1.4. Importancia y justificación.

Saber que se controla es un punto determinante y muy importante para tener éxito al momento de diseñar o implementar un sistema de control. Pues nos ayuda a conocer la dinámica del sistema, así como conseguir objetivos óptimos en tiempos adecuados.

Para lograr este objetivo, es de importante conocer la planta haciendo uso de métodos de identificación de sistemas, en este proyecto se utilizará el concepto de redes neuronales para la identificación de la planta.

Una vez conocida la dinámica del proceso e identificado el mismo, el siguiente objetivo se base en tener un buen diseño del controlador, el cual debe ser capaz de soportar cambios de los parámetros del sistema así como adecuarse de forma óptima a los mismos.

La importancia de este proyecto radica en desarrollar un controlador que ajuste sus parámetros de forma automática y óptima (parámetros Kp, Ki y Kd), incorporando redes neuronales dinámicas para la identificación del comportamiento dinámico de la planta, esto para poder anticiparnos a respuestas inesperadas y mejorar la respuesta durante su ejecución.

El desarrollo de este controlador, podremos ver la mejora que se tiene en comparación a un controlador clásico para la optimización del proceso, justificándose en el ahorro de tiempo para la sintonía del controlador, así como el ahorro económico en el gasto de insumos que se evitará durante la sintonización del mismo.

1.5. Alcances y limitaciones.

Los alcances de la presente tesis es el uso de redes neuronales artificiales dinámicas para la sintonización automática del controlador PID, usando los conceptos de control adaptativo basado en modelo (MRAC por sus siglas en inglés). Dicho alcance se basa primero en construir una planta emuladora, utilizando una red perceptrón multicapa (MLP) entrenada dinámicamente, esta planta emuladora se usará junto con otra red neuronal entrenada en línea (backpropagation) que irá ajustando los parámetros PID, tal que el error entre la señal de salida del modelo de referencia y la salida del proceso sea reducida. La red neuronal que irá sintonizando los parámetros tomará las salidas pasadas de la planta, las salidas pasadas de la señal de control y las señales pasadas del set point, las cuales serán procesadas para luego entregar los parámetros óptimos del control PID (K_p , K_i y K_d).

Dicha construcción se realizará utilizando funciones y scripts del programa Matlab y Simulink.

La limitación que presenta la presente tesis, es que no será aplicada a una planta altamente no lineal, pues la interpretación del modelo y del comportamiento de este tipo de plantas no será tratada. Otra limitación es que será únicamente simulada, presentando si todos los resultados obtenidos, pues una implementación práctica conlleva a una inversión económica considerable.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes de la investigación

Pirabakaran y Becerra (2002), centran su investigación en la aplicación de redes neuronales para la sintonía automática de controladores PID usando el concepto de control adaptativo basado en modelo de referencia, dicha investigación se aplica al control de nivel de dos tanques en serie conectados por una tubería corta. El flujo de agua que fluye por los tanques es regulado por una válvula de control, tal como se muestra en la figura 3.

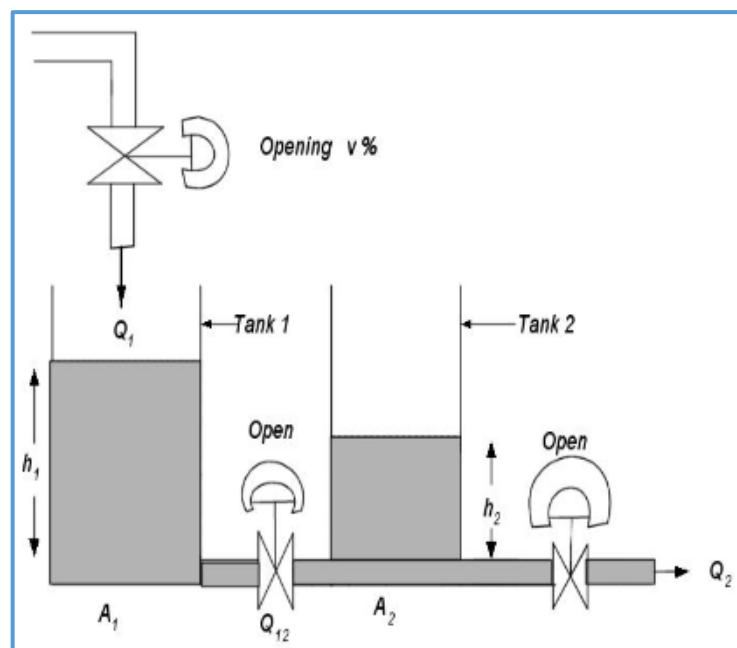


Figura 3: Planta utilizada por Pirabakaran and V. M. Becerra.

Fuente: K. Pirabakaran and V. M. Becerra. PID Autotuning Using Neural Networks and Model Reference Adaptive Control

La presente tesis se basa en los conceptos de este paper, con la diferencia que la planta a tratar es identificada por redes neuronales dinámicas y se aplica a una planta con discontinuidad en la ganancia.

Omatu, Yos y Kosaka (2009), desarrollan un neuro controlador para ser aplicado al control de un vehículo eléctrico. El neuro controlador usado se basa en el controlador PID, el cual es adaptado para resolver procesos de control o problemas de control inteligente. Esta investigación utiliza también redes neuronales estáticas para el

modelamiento, el cual se aplica al control de torque y velocidad de un vehículo eléctrico desarrollado en Japón denominado PIVOT.

Wang, Zhang, Jing, y An (2007), desarrollan el control de un sistema de aire acondicionado de frecuencia variable. Al igual que las investigaciones anteriores, usa redes neuronales estáticas, a diferencia de esta tesis, no usa control adaptativo, pero si el tipo de control que le denominan control híbrido, el cual consiste en aplicar el control neuronal y el PID clásico en forma paralela, del cual sugiere un tema interesante a seguir.

2.2. Óptica de la investigación.

La presente investigación pretende demostrar la funcionalidad de las redes neuronales en el campo de los sistemas de control ya que dentro del punto de vista de investigación las redes neuronales pueden jugar un papel muy importante en el control de procesos de alta complejidad, pues como es ya sabido, son la base de la inteligencia artificial, el cual trata temas de solución de muy altos criterios.

Pero dicha funcionalidad debe trabajar en conjunto con la teoría de control, para lo cual también se ve involucrado en la teoría de optimización empleando funciones de costo.

2.3. Selección de variables.

Para el desarrollo de este proyecto de tesis, se usarán las siguientes variables:

2.3.1. Variables independientes:

- Pesos sinápticos de la red neuronal (emulador).
- Set Point del proceso.
- Variable controlada del proceso.
- Salida del sistema del modelo de referencia.

2.3.2. Variables dependientes:

- Parámetros Kp (ganancia proporcional), Ki (ganancia integral) y Kd (ganancia derivativa), que dependen del ajuste de la función de costo en relación directa con el set point.
- Variable manipulada del proceso, depende directamente del set point.
- Pesos sinápticos de la red neuronal sintonizadora de los parámetros Kp, Ki y Kd (Neural Network Tuner). dependen de la salida del sintonizador neuronal.

2.4. Fundamentos de las redes neuronales artificiales.

2.4.1. ¿Qué es una red neuronal?

Sabemos que el cerebro humano es un sistema altamente complejo y difícil de comprender en su funcionamiento pero, gracias a las investigaciones y desarrollos que se han venido dando, de las cuales podemos mencionar a Haykin (1998), se ha podido conocer y entender “parte” del funcionamiento del cerebro, tal como la interconexión de las neuronas (sinapsis) para el aprendizaje y memorización de eventos que se presentan día a día.

Basados en estos estudios biológicos, surgió la idea de poder desarrollar un sistema capaz de “imitar” el funcionamiento que las neuronas ejecutan para aprender y/o memorizar sucesos cotidianos, dando inicio de esta manera, a lo que posteriormente se le denominaría “Inteligencia Artificial”, tal como se señala en Acosta y Zuloaga (2000).

2.4.2. Neurona biológica:

Para poder entender el desarrollo matemático que emula el comportamiento de las neuronas, se presenta en forma breve la forma del como la neurona biológica funciona.

Se sabe que la neurona es la unidad fundamental del sistema nervioso y sobre todo del cerebro, como se indica en la figura 4. Cada neurona realiza el procesamiento de las señales provenientes de otras neuronas combinándolas a través de estructuras llamadas

dendritas. Si las señales combinadas resultan superar un determinado potencial (umbral), ocasiona lo que se denomina el “disparo” de la neurona, produciéndose una señal a la salida de dicha neurona, la señal de salida se conduce sobre una estructura llamada axón. Este proceso tiene una naturaleza química pero con efectos eléctricos que pueden ser medidos, como se indica en Brío y Molina (2001).

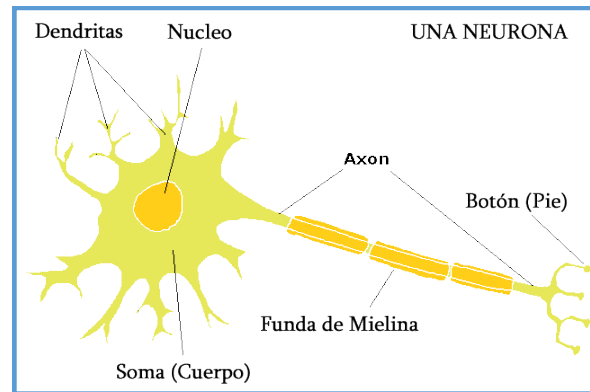


Figura 4: Neurona biológica.

Fuente: Dr. George Boeree. Universidad de Shippensburg

El axón (estructura de salida de una neurona) se divide y se conecta hacia las dendritas (entrada de una neurona) de otras neuronas, esta conexión entre el axón y las dendritas se le denomina sinapsis, tal como se puede observar en la figura 5.

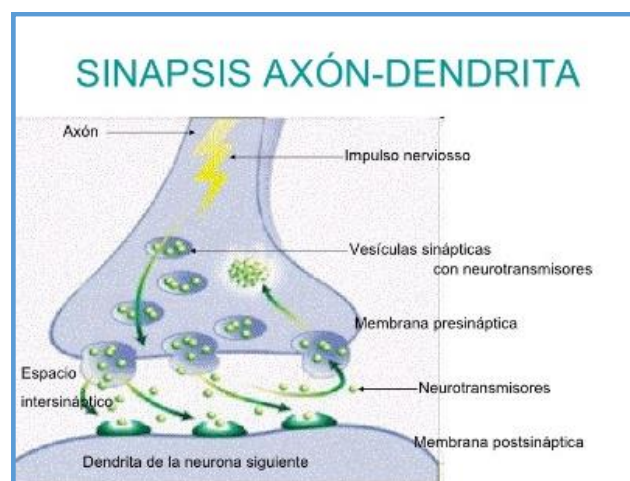


Figura 5: Sinapsis de una neurona biológica.

Fuente: <https://es.slideshare.net/guestd4aefc/funcin-de-relacion>

Se ha estimado que el cerebro contiene alrededor de 10 billones de neuronas en su córtex densamente conectadas y 60 trillones de sinapsis o conexiones, lo que convierte al cerebro en una estructura altamente eficiente a pesar de ser 6 veces más lento que un transistor de silicio (el tiempo de procesamiento del transistor de silicio se encuentra en el orden de los nanosegundos 10^{-9} s mientras que las neuronas procesan la información en el orden de los milisegundos 10^{-3} s), una manera de ver la eficiencia

del cerebro es con la eficiencia energética que tiene, pues la energía consumida por el cerebro es aproximadamente de 10^{-12} joules (J) mientras que la del transistor de silicio es cercana a 10^{-6} joules (J) por operación realizada por segundo. (Haykin, 1998).

Es así, que los estudios matemáticos se centraron en desarrollar algoritmos que emulen la recepción (entrada), el disparo (cuando se supera el umbral) y la salida de una neurona para el procesamiento de los datos y con esto obtener el aprendizaje o memorización de la estructura o sistema presentada.

2.4.3. Neurona Artificial:

Una neurona artificial, tomando una analogía con una neurona biológica, es una unidad de “procesamiento” con diversas entradas (dendritas), estas entradas son combinadas, generalmente con operaciones de multiplicación y suma, para poder obtener un nivel de actividad dentro de la neurona. Esta combinación de entradas, luego, son modificadas por una función llamada “función de activación” que determinará el nivel de umbral que permitirá el “disparo” de la neurona. Esta función de activación puede ser lineal o no lineal, dependiendo del sistema externo en la que se desarrolla el sistema. El valor de la salida de la función de activación, generalmente se interconecta con otras neuronas.

De lo indicado en el párrafo anterior, se puede determinar o modelar una neurona estándar, teniendo como referencia la figura 6.

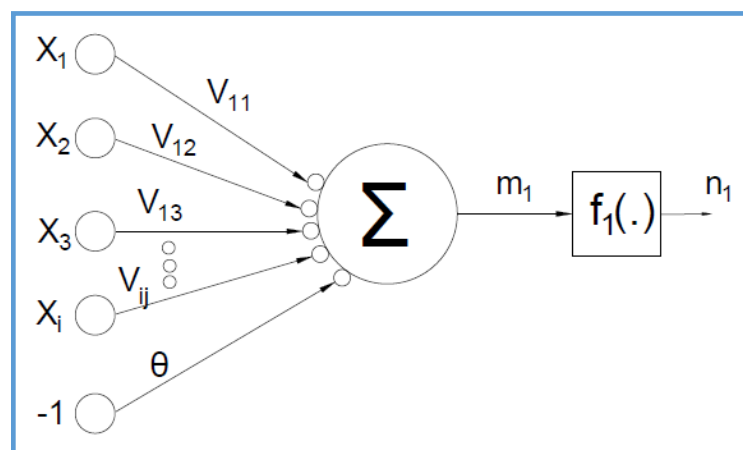


Figura 6: Neurona Artificial.
Fuente: Elaboración propia.

Donde:

- El conjunto de entradas y pesos sinápticos están denominados como $x_i; v_{ij}$.
- La regla de propagación está dada como $m_1 = \rho(x_i, v_{ij})$, siendo la más común: $\sum v_{ij} \cdot x_i$
- La función de activación, que calcula de forma simultánea la salida de la neurona y su estado de activación está dada por $n_1 = f_1(m_1)$
- Frecuentemente, se añade al conjunto de pesos de la neurona un parámetro adicional (θ_i), lo que representa añadir un grado de libertad adicional a la neurona, tal como se señala en Moran (2012).
- El modelo de una neurona estándar quedaría como:

$$n_1 = f_1(\sum v_{ij} - \theta_i)$$

Ya teniendo definida la neurona estándar, quedaría pendiente elegir una función de activación f_1 , de una de las que se muestra en la figura 7:


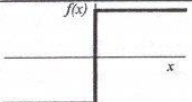
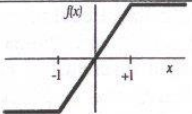
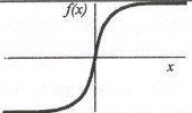
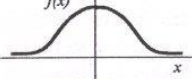
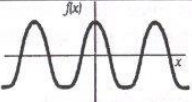
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & \text{si } x < -l \\ x, & \text{si } -l \leq x \leq +l \\ +1, & \text{si } x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \text{sen}(\omega x + \varphi)$	$[-1, +1]$	

Figura 7: Funciones de activación de una neurona artificial.

Fuente: Bonifacio Martín del Brío. Alfredo Sanz Molina (2007). Redes neuronales y sistemas borrosos. Méjico: Alfaomega.

Habiendo ya modelado la neurona, pasamos a definir el lugar que esta ocupa o desempeña dentro de una red neuronal.

Denominaremos una red neuronal, al conjunto de capas de neuronas, y cada capa neuronal estará formada por un conjunto de neuronas, tal como se aprecia en figura 8:

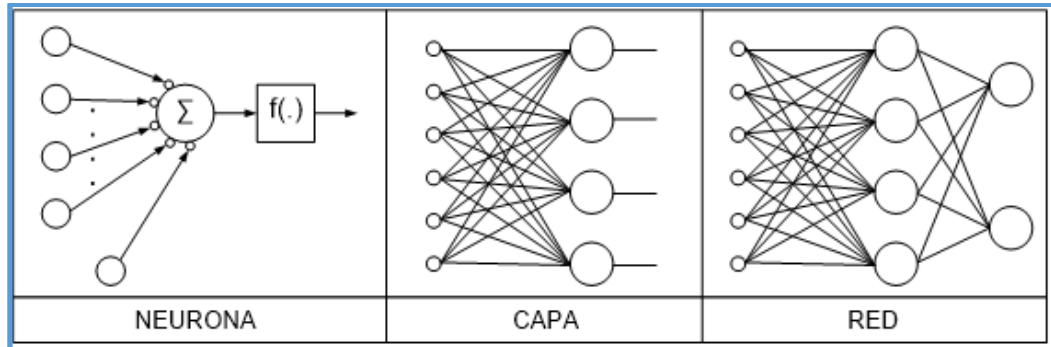


Figura 8: Estructura de una red neuronal.

Fuente: Elaboración propia.

2.4.4. Arquitectura de una red neuronal artificial.

En el presente trabajo de investigación, se trabajará con la red neuronal presentada en la figura 9, la cual tendrá una capa de entrada, una capa oculta y una capa de salida. Presentamos su arquitectura en la figura 9, así como los cálculos en matrices a ser usados para hallar los resultados denominados “hacia adelante”.

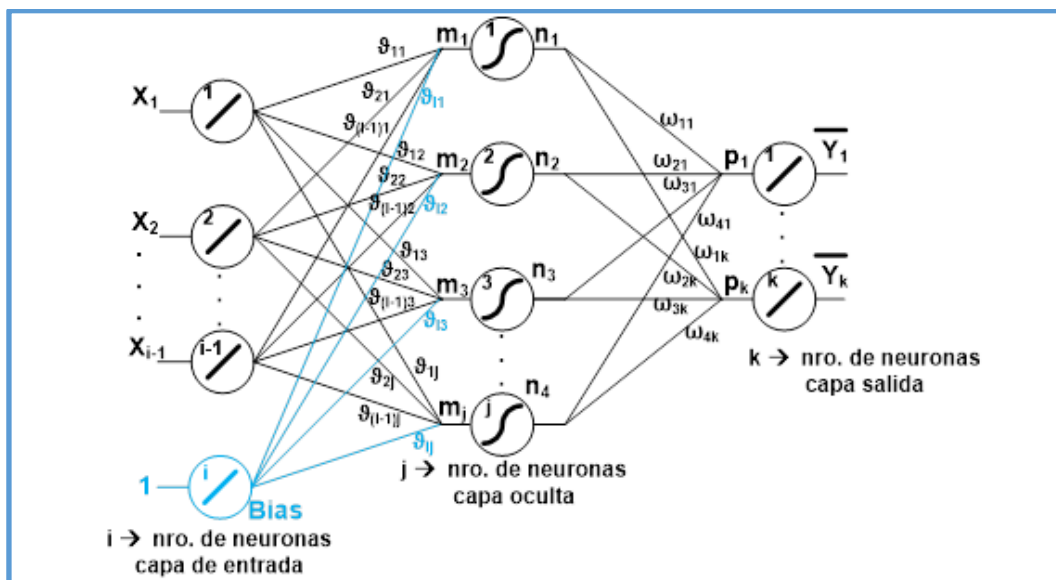


Figura 9: Arquitectura de red neuronal estática.

Fuente: Elaboración propia

Siendo:

La matriz de entrada x (con i entradas):

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{i-1} \\ 1 \end{bmatrix} \dots\dots\dots (1)$$

Dimensión: (nro. de entradas) x (1)

La matriz de los pesos de la capa oculta v_{ij} . Con i, j como número de neuronas en la capa de entrada y número de neuronas en la capa de salida respectivamente:

$$V = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1j} \\ v_{21} & v_{22} & \dots & v_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ v_{i1} & v_{i2} & \dots & v_{ij} \end{bmatrix} \dots\dots\dots (2)$$

Dimensión: (nro. de entradas) x (nro. de neuronas capa oculta).

La matriz resultante de la regla de propagación m en la capa oculta:

$$M = \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ m_j \end{bmatrix} \dots\dots\dots (3)$$

Dimensión: (nro. de neuronas capa oculta) x (1).

Teniendo la regla de propagación como: $\sum v_{ij} \cdot x_i$, resulta tener la matriz m de la siguiente manera (multiplicación matricial):

$$M = V^T \cdot X \dots\dots\dots (4)$$

Matriz n , resultante de aplicar la función de activación en la capa oculta:

.....

$$N = \begin{bmatrix} f_1(m_1) \\ f_2(m_2) \\ \dots \\ f_j(m_j) \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \dots \\ n_j \end{bmatrix} \quad (5)$$

Siendo la función de activación a utilizar en la capa oculta, la de tipo sigmoidea tipo 2, tal como se indica en la figura 10:

$$f_j = \frac{2}{1 + e^{-m_j}} - 1 \quad \dots\dots\dots (6)$$

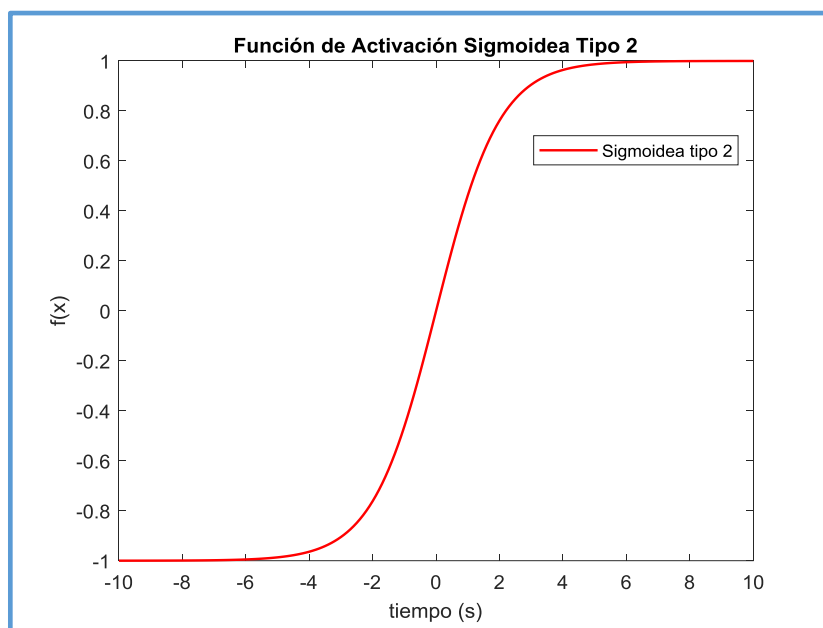


Figura 10: Función sigmoidea tipo 2.
Fuente: Elaboración propia

La matriz de los pesos en la capa de salida W :

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1k} \\ \dots & \dots & \dots & \dots \\ w_{j1} & w_{j2} & \dots & w_{jk} \end{bmatrix} \quad \dots\dots\dots (7)$$

Dimensión: (nro. de neuronas capa oculta) x (nro. de neuronas capa de salida).

Matriz \bar{Y} , matriz resultante de la capa de salida.

.....

$$\bar{Y} = \begin{bmatrix} \bar{y}_1 \\ \dots \\ \bar{y}_k \end{bmatrix} \quad (8)$$

Dimensión: (nro. de neuronas capa de salida) x (1)

Cabe indicar que tanto en la capa de entrada, como en la capa de salida, se usarán las funciones de activación lineales ($f(x) = x$), como se señala en Castillo, Castellano, Merelo y Prieto (2007), lo que en el gráfico 9 vendría a ser $f(p_k) = y_k$. Por lo tanto a nivel matricial, el resultado de la matriz \bar{Y} de salida sería:

$$\bar{Y} = W^T \cdot N \quad \dots\dots\dots (9)$$

Hasta este punto, el desarrollo mostrado de una red neuronal artificial es común entre una red neuronal estática y dinámica, de las cuales, la diferencia está en la etapa de entrenamiento, la cual se describirá a continuación.

2.4.5. Red Neuronal Estática:

Dentro de la arquitectura de una red neuronal, el objetivo principal es disminuir la diferencia entre la señal de salida de la red neuronal aproximada (\bar{Y}), y las señales de salidas de las funciones o sistemas que se quieren aproximar o modelar (Y), a la cual denominaremos error y será representado por:

$$e_k = \bar{Y}_k - Y_k ,$$

De forma matricial, con más de una neurona de salida:

$$e = \begin{bmatrix} \bar{Y}_1 - Y_1 \\ \bar{Y}_2 - Y_2 \\ \vdots \\ \bar{Y}_k - Y_k \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix} \quad \dots\dots\dots (10)$$

Para disminuir dicho *error* se emplean algoritmos y métodos matemáticos de optimización para alcanzar el objetivo. Estos algoritmos y métodos buscan adaptar los pesos sinápticos de cada capa de la red neuronal, ecuaciones (2) y (7), dentro de un proceso iterativo, para lograr minimizar así el *error* presentado.

Ahora bien, para alcanzar el objetivo primario que es la de disminuir el error, se tiene que partir de una denominada función de costo, tal como lo señala Morán (2012), el cual, para el caso de redes neuronales, está dada por:

$$J = \frac{1}{2}(e_1)^2 + \dots + \frac{1}{2}(error_k)^2$$

$$J = \frac{1}{2} \sum_{p=1}^k (e_p)^2 \dots\dots\dots (11)$$

Donde:

k , Es el número de neuronas de la capa de salida.

Habiéndose definido la función de costo, ahora se tiene que definir el algoritmo y método a usar para la adaptación de los pesos sinápticos de la red V, W de la arquitectura presentada en la figura 9.

En cuanto al algoritmo, existen diversos, pudiéndose nombrar al descenso del gradiente, método de Newton, gradiente conjugado, cuasi Newton y el algoritmo Levenberg-Marquardt, entre otros, como se indica en Acosta et al. (2000).

El algoritmo usado en esta etapa del trabajo será el del descenso del gradiente, utilizado conjuntamente con el método backpropagation, como se indica en Cardenas, Razuri, Sundgren, y Rahmani (2013).

El algoritmo de descenso del gradiente es el más simple y el más conocido, ya que solo hace uso del vector gradiente, y es por eso que se le reconoce como de primer orden.

Este algoritmo busca encontrar el punto w_{i+1} a partir de w_i , donde el punto w_{i+1} debe trasladarse en una dirección en la cual su pendiente alcance valores mínimos o cercanos a cero, a esta dirección se le denomina de entrenamiento, como se indica en la figura 11, y se la representa como:

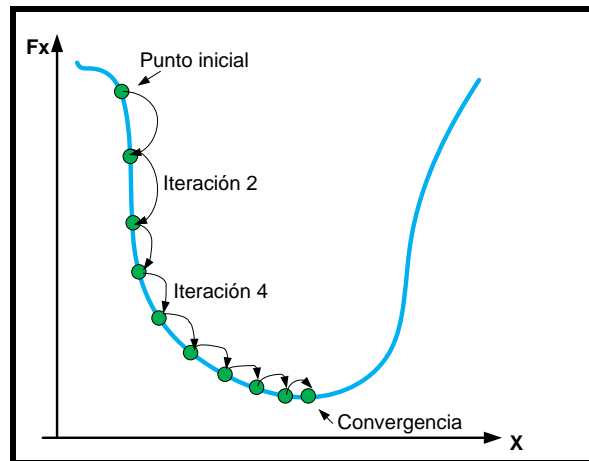


Figura 11: Representación geométrica del algoritmo descenso del gradiente.
Fuente: Elaboración propia.

$$w_{i+1} = w_i - \alpha_i \nabla f_i \quad \dots \quad (12)$$

Donde el parámetro α_i se le llama tasa de entrenamiento, que puede fijarse a priori o calcularse mediante un proceso de optimización o procesos heurísticos, para este caso se usará un valor fijo y pequeño.

Definiéndose ∇f_i como la gradiente de la función que se quiere minimizar respecto a la variable que se quiere adaptar, en este caso w_i .

Ahora, retornando a la arquitectura de la red neuronal de la figura 9, el algoritmo de descenso del gradiente se aplica a la retropropagación del error, es decir partimos del error y lo vamos retropropagando hasta llegar al parámetro que se quiere adaptar, esto se logra realizando las derivadas parciales, desde la función de costo hasta el parámetro adaptable como aparece en Cardenas, Razuri, Sundgren y Rahmani (2013).

Así se tiene que para actualizar un parámetro adaptable, se tiene que usar la ecuación (13) para actualizar los pesos sinápticos de la capa de oculta (V), y la ecuación (14) para actualizar los pesos sinápticos de la capa de salida (W).

Para la capa oculta:

$$V = V - \eta \frac{dJ}{dV} \dots\dots\dots (13)$$

Donde:

V , Es el peso sináptico de la capa oculta,

$\frac{dJ}{dV}$, es la derivada de la función de costo respecto a la variable adaptable, en este caso es el peso sináptico de la capa oculta.

η , Es el ratio de aprendizaje.

Para la capa de salida:

$$W = W - \eta \frac{dJ}{dW} \dots\dots\dots (14)$$

Donde:

W , Es el peso sináptico de la capa oculta,

$\frac{dJ}{dW}$, es la derivada de la función de costo respecto a la variable adaptable, en este caso es el peso sináptico de la capa oculta.

η , es el ratio de aprendizaje, que al inicio del entrenamiento puede ser asignado un valor aleatorio pequeño, por ejemplo entre [0.01, 0.1].

Como regla práctica, para hallar $\frac{dJ}{dV}$ o $\frac{dJ}{dW}$, aplicaremos:

$$\left(\begin{array}{c} \frac{dJ}{dV} \\ \frac{dJ}{dW} \end{array} \right) \text{ (salida de neuronas de la capa anterior). (error retropropagado)}$$

Interpretándose el error retropropagado como se indica en la figura 12, con dos neuronas en la salida (sistema 3-4-2), considerándose un bias a la entrada:

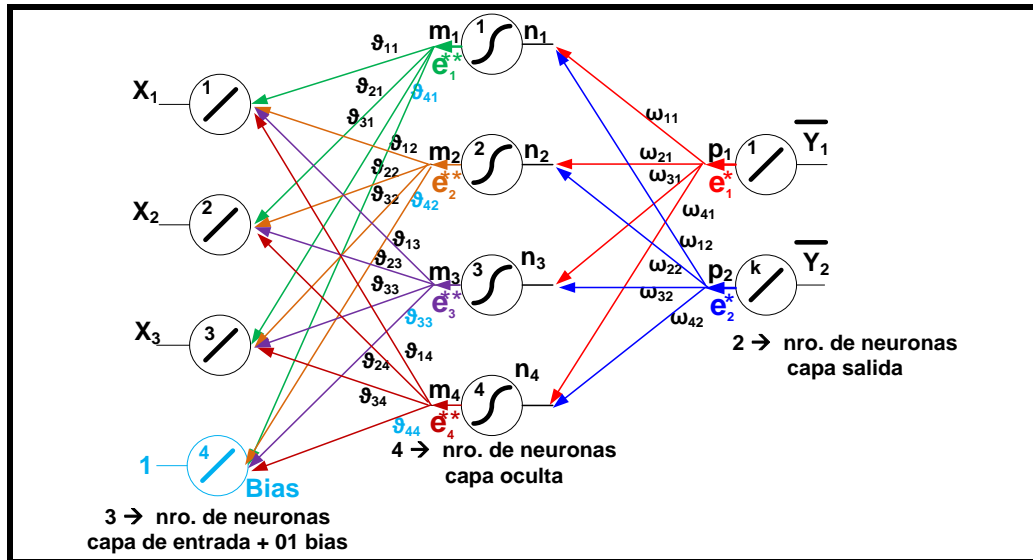


Figura 12: Retropropagación del error – red neuronal.

Fuente: Elaboración propia.

Donde:

e_1^* y e_2^* , son los errores retropropagados de la capa de salida hacia la capa oculta.

e_1^{**} y e_2^{**} , son los errores retropropagados de la capa oculta hacia la capa de entrada.

Cálculo de e_1^* y e_2^* que son los errores retropropagados de la capa de salida hacia la capa oculta, definiéndose e_1^* , como:

$$e_1^* = \frac{dJ_1}{de_1} \cdot \frac{de_1}{d\bar{y}_1} \cdot \frac{d\bar{y}_1}{dp_1} ;$$

De la ecuación (11), con $k = 1$, podemos calcular:

$$\frac{dJ_1}{de_1} = e_1,$$

Y de la ecuación (10) podemos calcular:

$$\frac{de_1}{d\bar{y}_1} = +1,$$

Sabiendo que la función de activación de la capa de salida es lineal (en otras aplicaciones se puede presentar otras funciones de activación que no son lineales), entonces:

$$\frac{d\bar{y}_1}{dp_1} = 1,$$

Quedando:

$$e_1^* = e_1;$$

Haciendo igualmente para e_2^* , queda:

$$e_2^* = e_2,$$

Expresándolo de forma matricial:

$$e^* = [e_1^* \ e_2^*] = [e_1 \ e_2] = e^T,$$

Ahora de acuerdo a la expresión:

$$\frac{dJ}{dW} \text{ (salida de neuronas de la capa anterior). (error retropropagado)}$$

También expresándolo de forma matricial, queda:

$$\frac{dJ}{dW} = N \cdot (e^*) = N \cdot (e)^T \dots\dots\dots (15)$$

Quedando finalmente la actualización de los pesos sinápticos de la capa de salida como como se indica en la ecuación (14), es decir:

$$W = W - \eta \frac{dJ}{dW}$$

Cálculo de e_1^{**} y e_2^{**} que son los errores retropropagados de la capa oculta hacia la capa de entrada, definiéndose e_1^{**} , como:

$$e_1^{**} = (e_1^* \cdot w_{11} + e_2^* \cdot w_{12}) \cdot \frac{dn_1}{dm_1},$$

Que contiene la suma de los aportes de los errores retropropagados de la capa inmediata.

Como se mencionó, siendo $n_1 = f(m_1)$, y, $f(m_1) = \frac{2}{1+e^{-m_1}} - 1$, función sigmoidea tipo 2, queda entonces:

$$\frac{dn_1}{dm_1} = \frac{d(fm_1)}{dm_1} = \frac{(1 - n_1) \cdot (1 + n_1)}{2} \dots\dots\dots (16)$$

Lo mismo para:

$$e_2^{**} = (e_1^* \cdot w_{21} + e_2^* \cdot w_{22}) \cdot \frac{dn_2}{dm_2},$$

$$e_3^{**} = (e_1^* \cdot w_{31} + e_2^* \cdot w_{32}) \cdot \frac{dn_3}{dm_3},$$

$$e_4^{**} = (e_1^* \cdot w_{41} + e_2^* \cdot w_{42}) \cdot \frac{dn_4}{dm_4},$$

Expresándolo de forma matricial:

$$e^{**} = \frac{dN}{dM} \cdot (W * (e^*)^T) = \frac{dN}{dM} \cdot (W * e) \dots\dots\dots (17)$$

Obsérvese la multiplicación punto a punto (.*) en la ecuación (17)

Siendo:

$$\frac{dN}{dM} = \begin{bmatrix} \frac{dn_1}{dm_1} \\ \frac{dn_2}{dm_2} \\ \frac{dn_3}{dm_3} \\ \frac{dn_4}{dm_4} \end{bmatrix} \dots\dots\dots (18)$$

Tomando en cuenta la expresión:

$$\left. \frac{dj}{dv} \right) \text{ (salida de neuronas de la capa anterior). (error retropropagado)}$$

Y tomándolo de forma matricial, quedaría:

$$\frac{dJ}{dV} = X \cdot (e^{**})^T \dots\dots\dots (19)$$

Quedando finalmente la actualización de los pesos sinápticos de la capa oculta como se indica en la ecuación (13), es decir:

$$V = V - \eta \frac{dJ}{dV} .$$

Para concluir, se indicará que es necesario el escalamiento de las señales de entrada – salida (llamada patrón de entrenamiento) a la red neuronal, para el trabajo completo de la misma, pues las funciones de activación sigmoidea tipo 2 trabajan en un rango de [-1, 1].

Para el presente trabajo se utilizó el escalamiento indicado en (20):

$$matriz_{escalado} = 1 + \frac{2}{[\max(matriz) - \min(matriz)]} \cdot [*] [matriz - \max(matriz)] \dots (20)$$

Nuevamente obsérvese la multiplicación (.*) punto a punto.

Ya teniendo definido la forma de actualizar los pesos sinápticos de las capas de salida y oculta, a continuación se presenta el procedimiento del método backpropagation para redes neuronales estáticas:

1. Generar las señales de entrada y salida de la función a aproximar o del sistema que se quiere modelar.
2. Escalar las entradas y salidas definidas en el paso 1, dentro del rango de operación de las funciones de activación de la red neuronal [0 1], [-1 1], etc. Una vez escaladas las entradas y salidas, ya se tiene definido los denominados patrones de entrenamiento del sistema.
3. Diseñar la red neuronal, en la cual se tiene que asignar el número de neuronas en la capa de entrada (definir si se considera bias o no), en la capa oculta y en la capa de

salida. En el caso de modelamiento de sistemas de control, como es un sistema dinámico y causal, las entradas deben considerar además las salidas anteriores al sistema, $(\bar{y}_{k-1}, \bar{y}_{k-2}, etc.)$. Así como definir la funciones de activación a utilizar en cada una de las capas.

4. Asignar valores iniciales aleatorios a los pesos sinápticos V, W pequeños, que pueden variar entre $[0.1, 0.3]$ por ejemplo.
5. Asignar un valor inicial al ratio de aprendizaje, que puede variar entre $[0.01, 0.1]$, considerando que un valor muy alto puede provocar la no convergencia del problema (saltos muy grandes) y un valor muy pequeño puede provocar mucha demora en la convergencia del mismo.
6. Definir el número de iteraciones a cumplir por el proceso de entrenamiento o un criterio de error mínimo a alcanzar para finalizar el proceso (por ejemplo error de mínimos cuadrados 'lse'). Iniciando con $k = 1$ como indicador de cada iteración.
7. Iniciar el proceso iterativo, considerando $\frac{dJ}{dV} = 0, \frac{dJ}{dW} = 0$.
8. Colocar a la entrada de la red cada patrón de entrenamiento. Iniciando por el primer patrón.
9. Realizar los cálculos para la capa de entrada, para la capa oculta y la capa de salida, considerando las funciones de activación ya definidas en el punto 1.
10. El valor obtenido en la capa de salida, hay que compararlo con el valor de salida que se desea obtener para ese patrón (salida escalada del sistema, punto 2). Es decir, se tiene que hallar el error mediante la ecuación (10).
11. Retropropagar el error, hallando las derivadas de cada función de activación de capa, emplear para ello las ecuaciones desde (15) al (19).
12. Hallar la derivada de la función de costo respecto a cada parámetro adaptable, empleando las ecuaciones (14) y (13).

13. Acumular cada derivada de la función de costo respecto a cada parámetro adaptable, es decir:

$$\frac{dJ_k}{dW_k} = \frac{dJ_{k-1}}{dW_{k-1}} + N_k \cdot (e_k^*); \text{ para los pesos sinápticos de la capa de salida.}$$

$$\frac{dJ_k}{dV_k} = \frac{dJ_{k-1}}{dV_{k-1}} + X_k \cdot (e_k^{**})^T; \text{ para los pesos sinápticos de la capa oculta.}$$

14. Si se termina de presentar todos los patrones de entrenamiento, ir al paso 15 de lo contrario presentar el próximo patrón de entrenamiento y volver al paso 9.

15. Dividir los valores acumulados hallados en el paso 13 entre el número total de patrones de entrenamiento presentados, es decir:

$$\frac{\frac{dJ_k}{dW_k}}{n_{patrones}} \quad \text{Y} \quad \frac{\frac{dJ_k}{dV_k}}{n_{patrones}}. \text{ Que resulta ser el promedio de todas las derivadas halladas en el paso 13.}$$

16. Hallar los nuevos pesos sinápticos mediante:

$$W = W - \eta \cdot \frac{\frac{dJ_k}{dW_k}}{n_{patrones}},$$

$$V = V - \eta \cdot \frac{\frac{dJ_k}{dV_k}}{n_{patrones}},$$

17. Si se alcanza el número de iteraciones definidas o un criterio de error mínimo, se culmina con el entrenamiento, de lo contrario aumentar $k = k + 1$ y retornar al paso 7.

A este tipo de procedimiento backpropagation, se le denomina tipo batch (lotes), como se indica en Gupta, Rao y Wood (n.d.), pues la actualización de los pesos, se realiza luego de presentar todos los patrones de entrenamiento, usando el promedio de las derivadas, tal como se presenta en el paso 15.

Existe otro tipo de procedimiento, llamado tipo patrón, donde los pesos sinápticos se actualizan por cada patrón presentado, no se enuncia en el presente trabajo, ya que el proceso tipo batch es el que va a ser usado para la presente tesis.

Con todo lo descrito, ya tenemos el soporte teórico para poder comprender, implementar y ejecutar una red neuronal estática, a continuación describiremos la forma de trabajo y operación de una red neuronal dinámica, para más adelante poder compararla con una red neuronal estática.

2.4.6. Red Neuronal Dinámica:

Para empezar con la descripción de una red neuronal dinámica, empezaremos explicando que es una derivada simple y derivada total, empleando para ello un ejemplo.

Sean:

$$z = 3y + 2x \quad \dots\dots\dots (21)$$

$$y = 4x + 5r,$$

$$r = 2x + 6s.$$

Ahora, si deseamos saber $\frac{dz}{dx}$, la derivada simple de (21) resultaría:

$$\frac{dz}{dx} = 2,$$

A diferencia de la derivada total (denotada con una ‘barra’ encima de la expresión de la derivada), resulta:

$$\overline{\frac{dz}{dx}} = \frac{dz}{dx} + \frac{dz}{dy} \cdot \frac{dy}{dx} + \frac{dz}{dy} \cdot \frac{dy}{dr} \cdot \frac{dr}{dx} \quad \dots\dots\dots (22)$$

Con esto podemos definir la derivada total, como la suma de todas las derivadas parciales de las funciones que involucran a la variable de referencia (‘respecto a:’).

La segunda parte de la expresión (22) que viene a ser $\frac{dz}{dy} \cdot \frac{dy}{dx}$, puede ser el resultado previo de la derivada de una función que va a ser aplicada a la derivada de otra función en un instante posterior.

Este tipo de derivada total es aplicada a las redes neuronales dinámicas, ya que como se explicará, para la actualización de los parámetros adaptables, se necesitará de las derivadas de los parámetros anteriores, tal como se explica en Morán (2012).

En la figura 13 se muestra la arquitectura de la red neuronal dinámica a emplear en esta tesis (dos entradas, una de ellas es la realimentación y una salida), en la cual se muestra que la realimentación de la salida formará parte del proceso de entrenamiento a diferencia de la red estática que no considera realimentación durante el entrenamiento. Esta realimentación suplirá de derivadas pasadas para el cálculo de las derivadas totales, como se indicó en el párrafo anterior. Además que partimos de la premisa que las funciones de activación y las derivadas de cada una de ellas son conocidas. Siendo V la matriz de los pesos sinápticos de la capa de entrada y W la matriz de los pesos sinápticos de la capa de salida, expresiones (2) y (7) respectivamente.

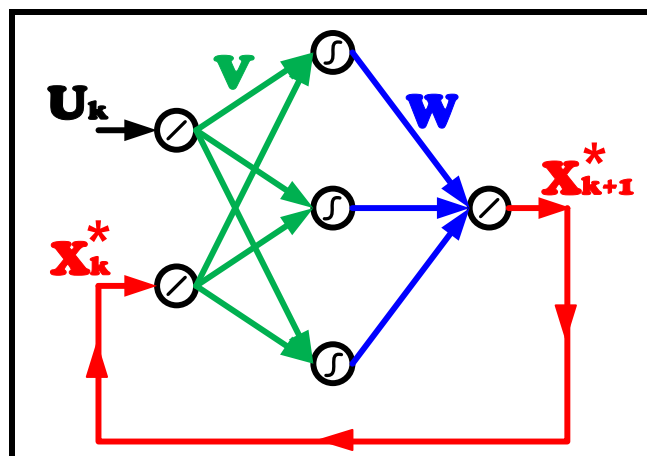


Figura 13: Arquitectura de la red neuronal dinámica
Fuente: Elaboración propia

De la figura 13 se puede observar que la salida de red neuronal anterior es la entrada a la siguiente etapa, y así se presentará hasta culminar con todos los patrones de entrenamiento, durante este proceso, no se actualizará ningún peso sináptico (los cuales se inician con valores aleatorios), esto quiere decir que la entrada a la siguiente etapa, ingresa un dato totalmente erróneo, el cual se irá corrigiendo durante el entrenamiento, proceso que realiza la actualización de V y W por retropropagación y derivadas totales, luego de presentar todos los patrones.

La figura 13 puede ser expandida mostrándose como la figura 14, para poder aclarar este punto.

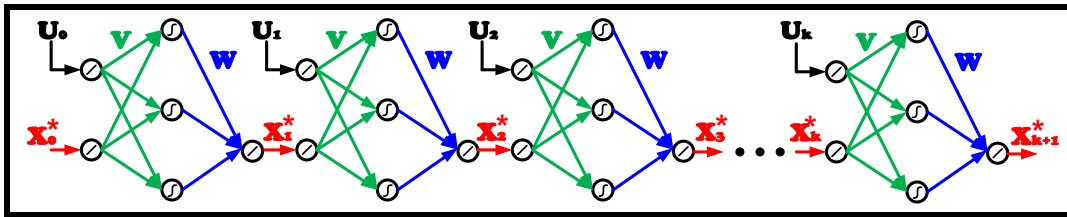


Figura 14: Arquitectura red neuronal dinámica expandida
Fuente: Elaboración propia

De la figura 14 y para el modelamiento de una función o sistema, podemos afirmar que: se conocen los datos de entrada y salida las cuales servirán para el modelamiento, al igual que la red neuronal estática, estos datos tienen que ser escalados, utilizando la ecuación (20).

Así resulta la figura 15:

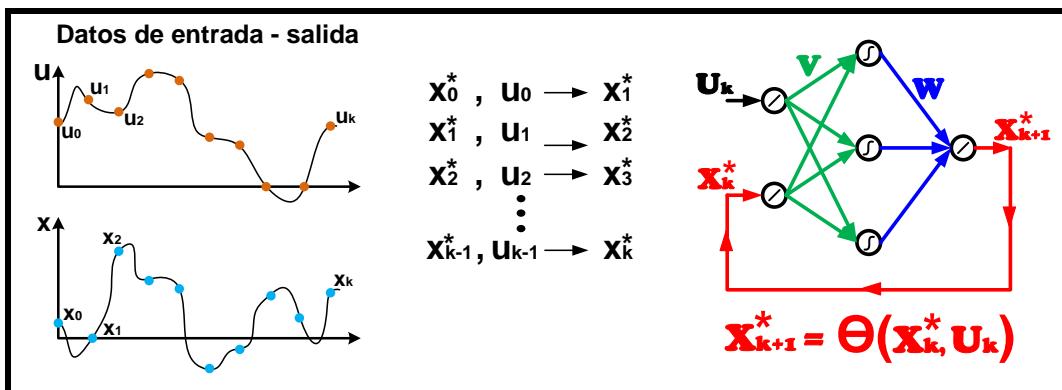


Figura 15: Datos de entrada salida red neuronal dinámica
Fuente: Elaboración propia

2.4.7. Entrenamiento de la red neuronal dinámica.

La función de costo se considera la misma que la red neuronal estática, expresión (11), la cual puede ser expresada como:

$$J = \frac{1}{2}(x_1^* - x_1) + \frac{1}{2}(x_2^* - x_2) + \dots + \frac{1}{2}(x_n^* - x_n) \dots \dots \dots (23)$$

Considerándose el error como la diferencia entre la salida aproximada de la red neuronal dinámica y la salida de la función o sistema que se quiere aproximar o modelar (salida deseada).

$$e = x^* - x \quad \dots\dots\dots (24)$$

Partiendo de la función de costo, y considerando el concepto de derivadas totales y la retropropagación de errores, se tiene la derivada ‘total’ de la función de costo respecto a la variable adaptable según las ecuaciones (27) y (28), las cuales son resultantes de las ecuaciones (25) y (26):

$$\frac{\overline{dJ}}{dv} = \frac{\overline{dJ}}{de} \cdot \frac{\overline{de}}{dx^*} \cdot \frac{\overline{dx^*}}{dv} = (e) \cdot (+1) \cdot \frac{\overline{dx^*}}{dv} \quad \dots\dots\dots (25)$$

$$\frac{\overline{dJ}}{dw} = \frac{\overline{dJ}}{de} \cdot \frac{\overline{de}}{dx^*} \cdot \frac{\overline{dx^*}}{dw} = (e) \cdot (+1) \cdot \frac{\overline{dx^*}}{dw} \quad \dots\dots\dots (26)$$

$$\frac{\overline{dJ}}{dv} = (x_1^* - x_1) \frac{\overline{dx_1^*}}{dv} + (x_2^* - x_2) \frac{\overline{dx_2^*}}{dv} + \dots + (x_n^* - x_n) \frac{\overline{dx_n^*}}{dv} \quad \dots\dots\dots (27)$$

$$\frac{\overline{dJ}}{dw} = (x_1^* - x_1) \frac{\overline{dx_1^*}}{dw} + (x_2^* - x_2) \frac{\overline{dx_2^*}}{dw} + \dots + (x_n^* - x_n) \frac{\overline{dx_n^*}}{dw} \quad \dots\dots\dots (28)$$

Ahora, de la ecuación (29) vamos a expandir el desarrollo de $\frac{\overline{dJ}}{dw}$, la cual se desarrolla de la siguiente manera: considerando la salida x_1^* de la figura 16, retroprogándolo hacia atrás en función de w , queda:

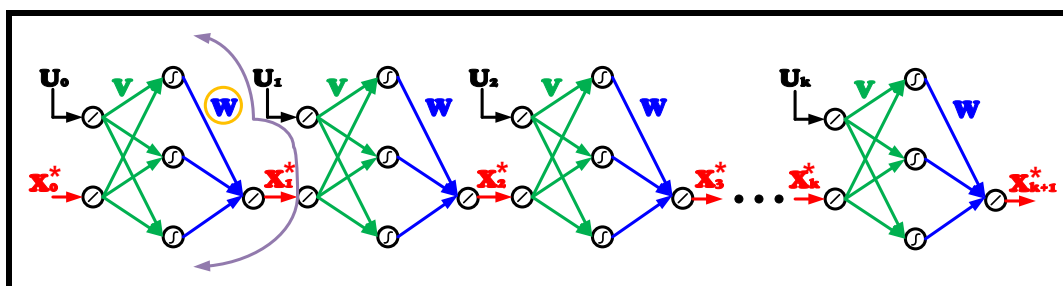


Figura 16: Derivada total de $\frac{\overline{dx_1^*}}{dw}$
Fuente: Elaboración propia

$$\frac{\overline{dx_1^*}}{dw} = \frac{dx_1^*}{dw} \quad \dots\dots\dots (29)$$

Ya que x_1^* depende únicamente del primer peso sináptico w , la derivada total será igual a la derivada simple.

Haciendo lo mismo para hallar $\frac{dx_2^*}{dw}$, en este caso, según la figura 17, retropropagando x_2^* , vemos que depende de los pesos w de la segunda y primera etapa; hasta aquí, vale recalcar que no se actualiza ningún peso sináptico, es decir w de la segunda y primera etapa son los mismos; siendo una derivada simple en la segunda etapa a la cual se suma las derivadas parciales que llegan hasta la primera etapa, quedando la derivada total como indica la expresión (30).

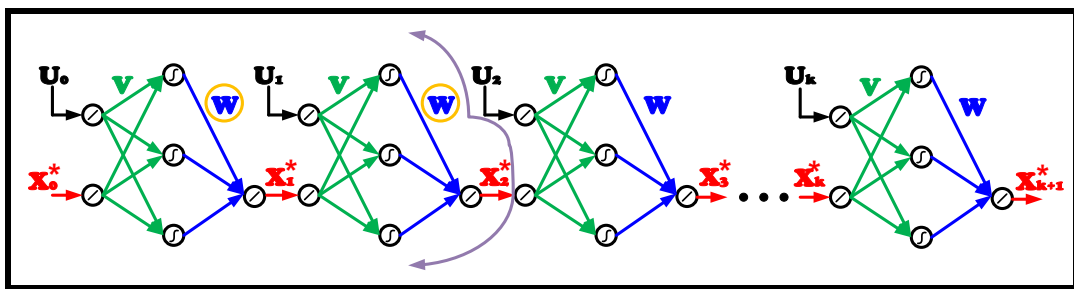


Figura 17: Derivada total de $\frac{dx_2^*}{dw}$
Fuente: Elaboración propia.

$$\frac{dx_2^*}{dw} = \frac{dx_2^*}{dw} + \frac{dx_2^*}{dx_1^*} \cdot \frac{dx_1^*}{dw} \dots\dots\dots (30)$$

Recordando que $\frac{dx_1^*}{dw} = \frac{dx_1^*}{dw}$ de la ecuación (29).

Para hallar $\frac{dx_3^*}{dw}$, procederemos de la misma forma, apuntando que la retropropagación de x_3^* depende de los pesos w de la tercera, segunda y primera etapa (que son iguales), por lo tanto su derivada total será la derivada simple de x_3^* respecto a w en la tercera etapa a la cual se suman las derivadas parciales que llegan hasta la primera etapa, pasando por la segunda. Indicando que la suma de las derivadas parciales anteriores, es la derivada total de la etapa anterior. Así se muestra la figura 18 y la ecuación (31), que representan $\frac{dx_3^*}{dw}$.

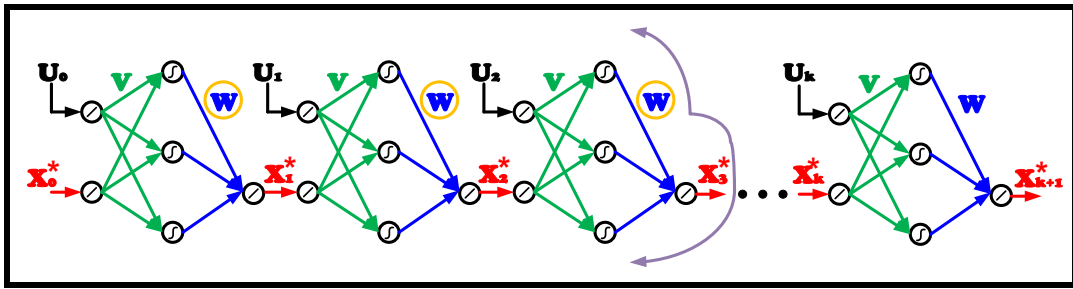


Figura 18: Derivada total de $\frac{\overline{dx}_3^*}{dw}$
 Fuente: Elaboración propia

$$\frac{\overline{dx}_3^*}{dw} = \frac{dx_3^*}{dw} + \frac{dx_3^*}{dx_2^*} \cdot \frac{dx_2^*}{dw} + \frac{dx_3^*}{dx_2^*} \cdot \frac{dx_2^*}{dx_1^*} \cdot \frac{dx_1^*}{dw},$$

$$\frac{\overline{dx}_3^*}{dw} = \frac{dx_3^*}{dw} + \frac{dx_3^*}{dx_2^*} \cdot \left(\frac{dx_2^*}{dw} + \frac{dx_2^*}{dx_1^*} \cdot \frac{dx_1^*}{dw} \right),$$

$$\frac{\overline{dx}_3^*}{dw} = \frac{dx_3^*}{dw} + \frac{dx_3^*}{dx_2^*} \cdot \left(\frac{\overline{dx}_2^*}{dw} \right) \dots\dots\dots (31)$$

El proceso se debe realizar hasta presentar todos los patrones de entrenamiento (k), pero de las ecuaciones (29), (30) y (31), desarrolladas anteriormente, podemos generalizar:

$$\frac{\overline{dx}_{k+1}^*}{dw} = \frac{dx_{k+1}^*}{dw} + \frac{dx_{k+1}^*}{dx_k^*} \cdot \left(\frac{\overline{dx}_k^*}{dw} \right) \dots\dots\dots (32)$$

La ecuación (32) se aplica a cada w_{jk} y es denominada la fórmula recursiva para el cálculo de las derivadas totales parciales.

Aplicando este procedimiento y reemplazando en la ecuación (29), ya se puede calcular $\frac{\overline{dJ}}{dw}$ de un modo recursivo por medio de la ecuación (32) para la actualización de los pesos w .

Para culminar, la actualización de los pesos sinápticos de la capa de salida viene dada por la ecuación (33):

$$w_{jk} = w_{jk} - \eta \frac{\overline{dJ}}{dw_{jk}} \dots\dots\dots (33)$$

Para los pesos sinápticos de la capa oculta, se procede de la misma manera, solo que se tiene que considerar la ecuación (28), y cambiar los términos w por v en las ecuaciones (32) y (33), quedando las ecuaciones (34) y (35):

$$\frac{dx_{k+1}^*}{dv} = \frac{dx_{k+1}^*}{dv} + \frac{dx_{k+1}^*}{dx_k^*} \cdot \left(\frac{dx_k^*}{dv}\right) \dots\dots\dots (34)$$

$$v_{jk} = v_{jk} - \eta \frac{dJ}{dv_{jk}} \dots\dots\dots (35)$$

Ya que todas las funciones y/o ecuaciones están definidas en la red neuronal mostrada en la figura 13, ya es factible encontrar todas las derivadas involucradas en las ecuaciones mostradas en las ecuaciones desde (29) hasta (35).

Ahora vamos a describir las matrices involucradas para una red neuronal dinámica que involucra dos salidas en la última capa, según muestra la figura 19, aplicando la teoría explicada referente al entrenamiento de redes neuronales dinámicas. A partir de acá, se puede generalizar para más entradas y salidas en la red neuronal.

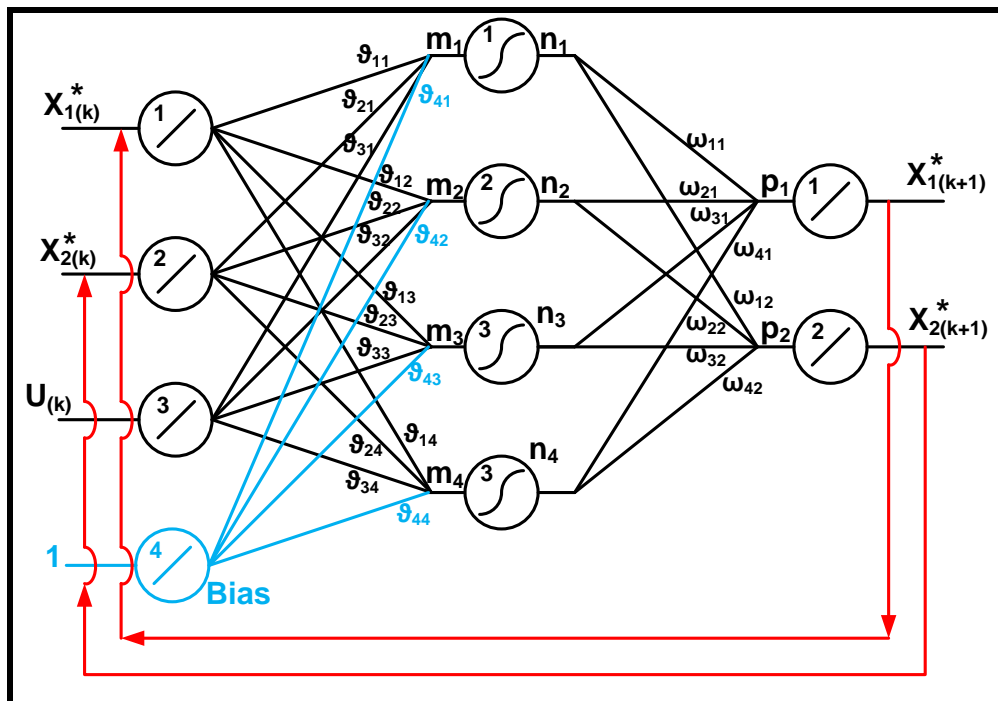


Figura 19: Estructura red neuronal dinámica con dos salidas.
Fuente: Elaboración propia.

Teniendo como ejemplo la estructura de la figura 19, procederemos a desarrollar las matrices involucradas:

Matriz de entrada:

$$X_{(k)}^* = \begin{bmatrix} x_{1(k)}^* \\ x_{2(k)}^* \\ u_k \\ 1 \end{bmatrix} \dots\dots\dots (36)$$

Dimensión: (# neuronas capa de entradas x 1) → (4 x 1)

Matriz pesos sinápticos capa oculta:

$$V = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ v_{41} & v_{42} & v_{43} & v_{44} \end{bmatrix} \dots\dots\dots (37)$$

Dimensión: (# neuronas capa de entradas x # neuronas capa oculta) → (4 x 4)

Matriz pesos sinápticos de la capa de salida:

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} \dots\dots\dots (38)$$

Dimensión: (# neuronas capa de oculta x # neuronas capa salida) → (4 x 2)

Matriz M (suma de la multiplicación de los pesos de la capa oculta con los valores de las entradas):

$$M = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \dots\dots\dots (39)$$

$$M = V^T X_{(k)}^* \dots\dots\dots (40)$$

Dimensión: (# neuronas capa de oculta x 1) → (4 x 1)

Matriz N salida de la capa oculta:

$$N = \begin{bmatrix} f(m_1) \\ f(m_2) \\ f(m_3) \\ f(m_4) \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix} \dots\dots\dots (41)$$

Dimensión: (# neuronas capa de oculta x 1) → (4 x 1)

Siendo $f(m_j)$ la función de activación sigmoidea tipo 2 en la capa oculta.

$$f(m_j) = \frac{2}{1 + e^{-m_j}} - 1 \dots\dots\dots (42)$$

Matriz $X_{(k+1)}^*$, matriz de salida, capa de salida:

$$X_{(k+1)}^* = \begin{bmatrix} x_{1(k+1)}^* \\ x_{2(k+1)}^* \end{bmatrix} \dots\dots\dots (43)$$

$$X_{(k+1)}^* = W^T N \dots\dots\dots (44)$$

Dimensión: (# neuronas capa de salida x 1) → (2 x 1)

Matriz $\frac{dN}{dM}$, matriz derivada de la función de transferencia de la capa oculta:

$$\frac{dN}{dM} = \begin{bmatrix} \frac{dn_1}{dm_1} & 0 & 0 & 0 \\ 0 & \frac{dn_2}{dm_2} & 0 & 0 \\ 0 & 0 & \frac{dn_3}{dm_3} & 0 \\ 0 & 0 & 0 & \frac{dn_4}{dm_4} \end{bmatrix} \dots\dots\dots (45)$$

Siendo la derivada de la ecuación (42):

$$\frac{dn_j}{dm_j} = \frac{d(fm_j)}{dm_j} = \frac{(1 - n_j) \cdot (1 + n_j)}{2} \dots\dots\dots (46)$$

Dimensión: (# neuronas capa oculta x # neuronas capa oculta) → (4 x 4).

Obsérvese que es una matriz diagonal.

Matriz $\frac{dX_{1(k+1)}^*}{dW}$, matriz derivada de la primera salida respecto a los pesos sinápticos de la capa de salida:

$$\frac{dX_{1(k+1)}^*}{dW} = \begin{bmatrix} n_1 & 0 \\ n_2 & 0 \\ n_3 & 0 \\ n_4 & 0 \end{bmatrix} \dots\dots\dots (47)$$

Dimensión (# neuronas capa oculta x # neuronas capa salida) → (4 x 2).

Comando: [N zeros(#neuronas capa oculta, 1)]

Matriz $\frac{dX_{2(k+1)}^*}{dW}$, matriz derivada de la segunda salida respecto a los pesos sinápticos de la capa de salida:

$$\frac{dX_{2(k+1)}^*}{dW} = \begin{bmatrix} 0 & n_1 \\ 0 & n_2 \\ 0 & n_3 \\ 0 & n_4 \end{bmatrix} \dots\dots\dots (48)$$

Dimensión (*# neuronas capa oculta x # neuronas capa salida*) → (4 x 2).

Comando: `[zeros(#neuronas capa oculta, 1) N]`

Matriz $\frac{dX_{1(k+1)}^*}{dV}$, matriz derivada de la primera salida respecto a los pesos sinápticos de la capa oculta:

$$\frac{dX_{1(k+1)}^*}{dV} = \begin{bmatrix} x_{1(k)}^* \\ x_{2(k)}^* \\ u \\ 1 \end{bmatrix} \cdot [w_{11} \quad w_{21} \quad w_{31} \quad w_{41}] \cdot \begin{bmatrix} \frac{dn_1}{dm_1} & 0 & 0 & 0 \\ 0 & \frac{dn_2}{dm_2} & 0 & 0 \\ 0 & 0 & \frac{dn_3}{dm_3} & 0 \\ 0 & 0 & 0 & \frac{dn_4}{dm_4} \end{bmatrix} \dots\dots\dots (49)$$

La matriz $[w_{11} \quad w_{21} \quad w_{31} \quad w_{41}]$ es la retropropagación de $x_{1(k+1)}$ hacia las capas anteriores.

Quedando la expresión final de la ecuación (49) como:

$$\frac{dX_{1(k+1)}^*}{dV} = X_{(k)}^* \cdot [W(:,1)]^T \cdot \frac{dN}{dM} \dots\dots\dots (50)$$

Dimensión: (*# neuronas capa entrada x # neuronas capa oculta*) → (4 x 4).

Matriz $\frac{dX_{2(k+1)}^*}{dV}$, matriz derivada de la segunda salida respecto a los pesos sinápticos de la capa oculta:

$$\frac{dX_{2(k+1)}^*}{dV} = \begin{bmatrix} x_{1(k)}^* \\ x_{2(k)}^* \\ u \\ 1 \end{bmatrix} \cdot [w_{12} \quad w_{22} \quad w_{32} \quad w_{42}] \cdot \begin{bmatrix} \frac{dn_1}{dm_1} & 0 & 0 & 0 \\ 0 & \frac{dn_2}{dm_2} & 0 & 0 \\ 0 & 0 & \frac{dn_3}{dm_3} & 0 \\ 0 & 0 & 0 & \frac{dn_4}{dm_4} \end{bmatrix} \dots (51)$$

La matriz $[w_{12} \quad w_{22} \quad w_{32} \quad w_{42}]$ es la retropropagación de $x_{2(k+1)}$ hacia las capas anteriores.

Quedando la expresión final de (51) como:

$$\frac{dX_{2(k+1)}^*}{dV} = X_{(k)}^* \cdot [W(:,2)]^T \cdot \frac{dN}{dM} \dots (52)$$

Dimensión (# neuronas capa entrada x # neuronas capa oculta) $\rightarrow (4 \times 4)$.

Matriz $\frac{dX_{(k+1)}^*}{dX_{(k)}^*}$, llamada la matriz “jacobiana”, en este caso por tener la red neuronal dos salidas, esta matriz tendrá un dimensión de (2×2) .

De la figura 20 se muestra que cada salida tiene que retropropagarse por todas las capas, empleando sus respectivas funciones y derivadas, hasta llegar a las entradas pasadas correspondientes. $X_{1(k+1)}^* \rightarrow X_{1(k)}^*$, $X_{1(k+1)}^* \rightarrow X_{2(k)}^*$, $X_{2(k+1)}^* \rightarrow X_{1(k)}^*$, $X_{2(k+1)}^* \rightarrow X_{2(k)}^*$.

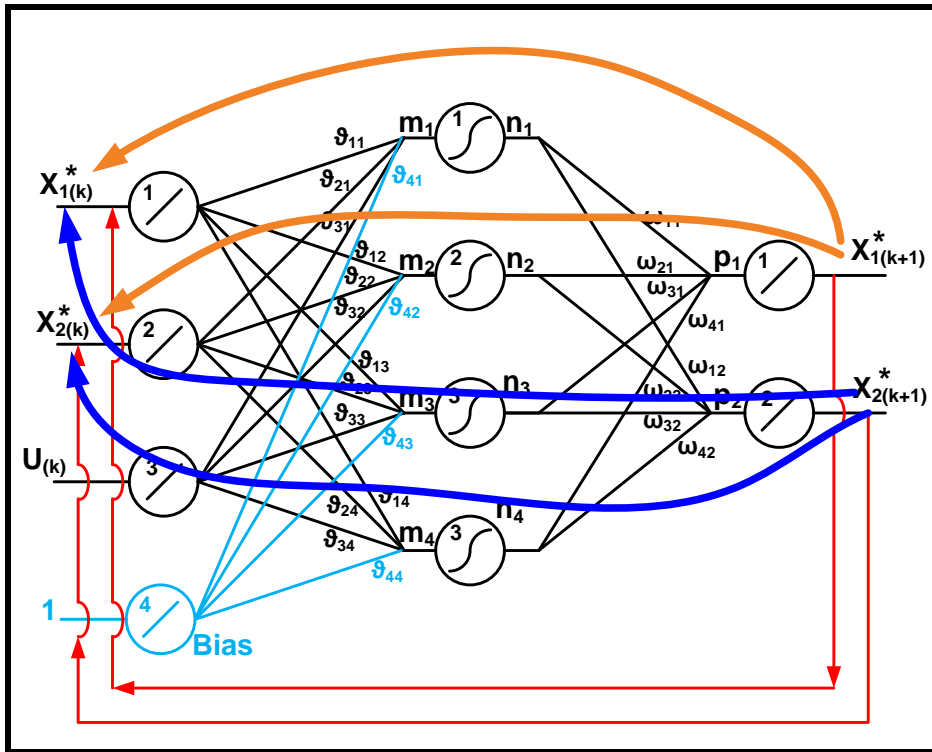


Figura 20: Retropropagación de las salidas hacia las entradas pasadas, para hallar la matriz jacobiana.
Fuente: Elaboración propia

$$\frac{dX^*_{(k+1)}}{dX^*_{(k)}} = \begin{bmatrix} \frac{dX^*_{1(k+1)}}{dX^*_{1(k)}} & \frac{dX^*_{1(k+1)}}{dX^*_{2(k)}} \\ \frac{dX^*_{2(k+1)}}{dX^*_{1(k)}} & \frac{dX^*_{2(k+1)}}{dX^*_{2(k)}} \end{bmatrix} \dots \dots \dots (53)$$

Desarrollando la expresión (53) para cada uno de sus filas-columnas, se tiene:

$$\frac{dX^*_{(k+1)}}{dX^*_{(k)}} = \begin{bmatrix} W_{11} & W_{21} & W_{31} & W_{41} \\ W_{12} & W_{22} & W_{32} & W_{42} \end{bmatrix} \cdot \begin{bmatrix} \frac{dn_1}{dm_1} & 0 & 0 & 0 \\ 0 & \frac{dn_2}{dm_2} & 0 & 0 \\ 0 & 0 & \frac{dn_3}{dm_3} & 0 \\ 0 & 0 & 0 & \frac{dn_4}{dm_4} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \\ v_{13} & v_{23} \\ v_{14} & v_{24} \end{bmatrix}$$

Quedando finalmente:

$$\frac{dX^*_{(k+1)}}{dX^*_{(k)}} = W^T \cdot \frac{dN}{dM} \cdot [V\{1: (\# \text{ neuronas de entrada} - 1), : \}]^T \dots \dots \dots (54)$$

Dimensión: (# neuronas capa salida x # n# neuronas capa salida) → (2 x 2).

Ahora para la red neuronal que emplea dos salidas en la capa de salida, la actualización de los pesos se basa en la función de costo, que para este caso es:

$$J = \frac{1}{2} (X_{(1)}^* - X_{(1)})^T \cdot (X_{(1)}^* - X_{(1)}) + \frac{1}{2} (X_{(2)}^* - X_{(2)})^T \cdot (X_{(2)}^* - X_{(2)}) + \frac{1}{2} (X_{(n)}^* - X_{(n)})^T \cdot (X_{(n)}^* - X_{(n)}) \quad \dots\dots\dots (55)$$

Con las derivadas totales de la función de costo respecto a los parámetros adaptables:

$$\frac{\overline{dJ}}{\overline{dV}} = (X_{(1)}^* - X_{(1)})^T \frac{\overline{dX_{(1)}}}{\overline{dV}} + (X_{(2)}^* - X_{(2)})^T \frac{\overline{dX_{(2)}}}{\overline{dV}} + (X_{(n)}^* - X_{(n)})^T \frac{\overline{dX_{(n)}}}{\overline{dV}} \quad \dots\dots (56)$$

$$\frac{\overline{dJ}}{\overline{dW}} = (X_{(1)}^* - X_{(1)})^T \frac{\overline{dX_{(1)}}}{\overline{dW}} + (X_{(2)}^* - X_{(2)})^T \frac{\overline{dX_{(2)}}}{\overline{dW}} + (X_{(n)}^* - X_{(n)})^T \frac{\overline{dX_{(n)}}}{\overline{dW}} \quad \dots\dots (57)$$

Indicando que $X_{(k)}^*$ es una matriz de 2 x 1, que representa las dos salidas de la red neuronal.

$$X_{(k)}^* = \begin{bmatrix} X_{1(k)}^* \\ X_{2(k)}^* \end{bmatrix} \quad \dots\dots\dots (58)$$

Y

$$X_{(k+1)}^* = \begin{bmatrix} X_{1(k+1)}^* \\ X_{2(k+1)}^* \end{bmatrix} \quad \dots\dots\dots (59)$$

A esta arquitectura de dos salidas, también se le puede aplicar la fórmula recursiva indicada en la expresión (32),

$$\frac{\overline{dx_{(k+1)}^*}}{\overline{dw}} = \frac{dx_{(k+1)}^*}{dw} + \frac{dx_{(k+1)}^*}{dx_{(k)}^*} \cdot \left(\frac{\overline{dx_k^*}}{\overline{dw}} \right) \quad \dots\dots\dots (60)$$

Considerando todas las matrices indicadas desde las expresiones (36) al (59), recordando que $\frac{dx^{*(k+1)}}{dx^{*(k)}}$ es la denominada matriz ‘jacobiana’.

Finalmente, luego de aplicar todos los cálculos mostrados, se procede a actualizar los pesos sinápticos, considerando las ecuaciones (33) y (35), es decir:

$$w_{jk} = w_{jk} - \eta \frac{\bar{d}J}{dw_{jk}},$$

$$v_{jk} = v_{jk} - \eta \frac{\bar{d}J}{dv_{jk}}.$$

A continuación se detalla el procedimiento del modelamiento con una red neuronal dinámica con dos salidas:

1. Generar las señales de entrada y salida de la función a aproximar o del sistema que se quiere modelar.
2. Escalar las entradas y salidas definidas en el paso 1, dentro del rango de operación de las funciones de activación de la red neuronal [0 1], [-1 1], etc. Una vez escaladas las entradas y salidas, ya se tiene definido los denominados patrones de entrenamiento del sistema.
3. Diseñar la red neuronal dinámica, considerando la arquitectura mostrada en la figura 2.12, en la cual se tiene que asignar el número de neuronas en la capa de entrada (definir si se considera bias o no), en la capa oculta y en la capa de salida. Como se trata de una red neuronal dinámica, las entradas deben considerar además las salidas anteriores al sistema, $(x_{k-1}^*, x_{k-2}^*, etc.)$. Así como definir la funciones de activación a utilizar en cada una de las capas.
4. Asignar valores iniciales aleatorios a los pesos sinápticos V, W pequeños, que pueden variar entre [0.1, 0.3] por ejemplo.
5. Asignar un valor inicial al ratio de aprendizaje, que puede variar entre [0.01, 0.1], considerando que un valor muy alto puede provocar la no convergencia del problema (saltos muy grandes) y un valor muy pequeño puede provocar mucha demora en la convergencia del mismo.

6. Definir el número de iteraciones a cumplir por el proceso de entrenamiento o un criterio de error mínimo a alcanzar para finalizar el proceso (por ejemplo error de mínimos cuadrados 'lse'). Iniciando con $k = 1$ como indicador de cada iteración.
7. Iniciar el proceso iterativo, considerando $\frac{dJ}{dV} = 0, \frac{dJ}{dW} = 0$, así como las derivadas totales $\frac{\overline{dJ}}{dV} = 0, \frac{\overline{dJ}}{dW} = 0$.
8. Colocar a la entrada de la red cada patrón de entrenamiento. Iniciando por el primer patrón.
9. Realizar los cálculos para la capa de entrada, para la capa oculta y la capa de salida para el patrón presentado, considerando las funciones de activación ya definidas en el punto 1.
10. Realizar el cálculo de las derivadas simples indicadas en las ecuaciones (45) y (46); de las derivadas totales de las salidas $(x_{1(k)}^*, x_{2(k)}^*)$ respecto a las variables adaptables (v, w) indicadas en las ecuaciones (47) al (52), empleando la expresión recursiva indicada en (60) y del jacobiano indicada en las ecuaciones (53) y (54).
11. El valor obtenido en la capa de salida de la red neuronal, hay que compararlo con el valor de salida que se desea obtener para ese patrón (salida escalada del sistema, punto 2). Es decir, se tiene que hallar el error mediante la ecuación (25).
12. Acumular las derivadas totales de la función de costo respecto a las variable adaptables $(\frac{\overline{dJ}}{dV}, \frac{\overline{dJ}}{dW})$, empleando las derivadas totales indicadas en (56) y (57). Es decir:

$$\frac{\overline{dJ}}{dW} = \frac{\overline{dJ}}{dW} + (X_{(1)}^* - X_{(1)})^T \frac{\overline{dX_{(1)}}}{dW} + (X_{(2)}^* - X_{(2)})^T \frac{\overline{dX_{(2)}}}{dW},$$

$$\frac{\overline{dJ}}{dV} = \frac{\overline{dJ}}{dV} + (X_{(1)}^* - X_{(1)})^T \frac{\overline{dX_{(1)}}}{dV} + (X_{(2)}^* - X_{(2)})^T \frac{\overline{dX_{(2)}}}{dV}.$$

13. La salida de red neuronal, será una de las entradas a la red en el instante siguiente.
($x_{in} = x_{out}$)

14. Si se termina de presentar todos los patrones de entrenamiento, ir al paso 15 de lo contrario presentar el próximo patrón de entrenamiento y volver al paso 9.

15. Dividir los valores acumulados hallados en el paso 12 entre el número total de patrones de entrenamiento presentados, es decir:

$\frac{\overline{\frac{dJ}{dW}}}{n_{patrones}}$ y $\frac{\overline{\frac{dJ}{dV}}}{n_{patrones}}$. Que resulta ser el promedio de todas las derivadas halladas en el paso 12.

16. Hallar los nuevos pesos sinápticos mediante:

$$W = W - \eta \cdot \frac{\overline{\frac{dJ}{dW}}}{n_{patrones}},$$

$$V = V - \eta \cdot \frac{\overline{\frac{dJ}{dV}}}{n_{patrones}},$$

17. Si se alcanza el número de iteraciones definidas o un criterio de error mínimo, se culmina con el entrenamiento, de lo contrario aumentar $k = k + 1$ y retornar al paso 7.

A continuación se muestran las gráficas 21 y 22, que son el resultado de aplicar el modelamiento de un sistema con redes neuronales estáticas y dinámicas (los datos son tomadas de la planta a presentar en el siguiente capítulo referente a la identificación del sistema).

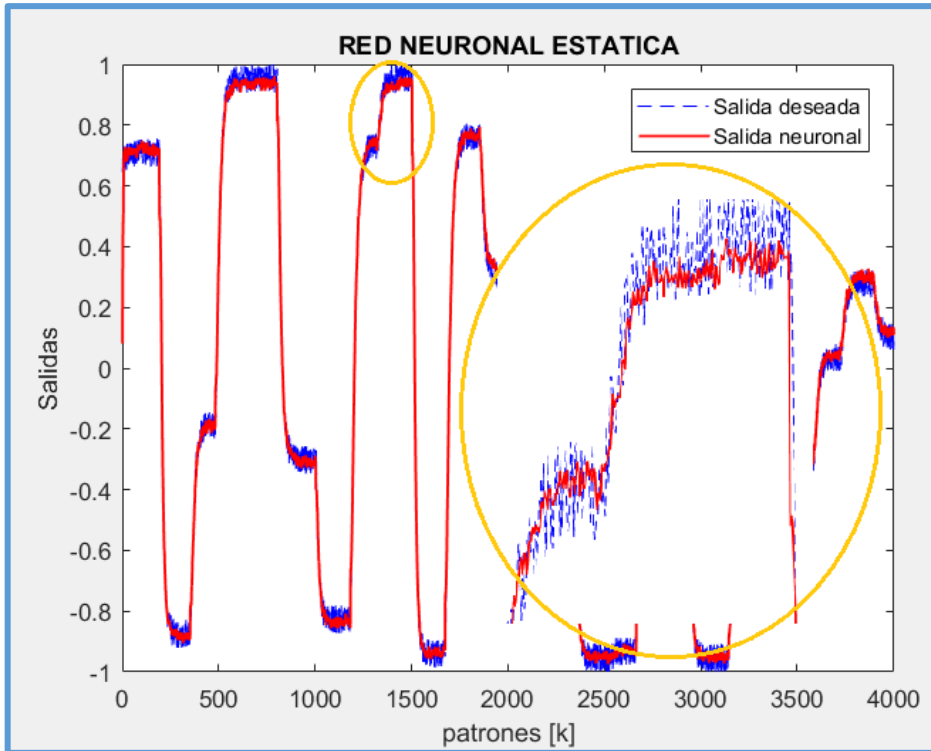


Figura 21: Respuesta de la identificación con una red neuronal estática, se puede observar que identifica hasta el ruido presente en el sistema.

Fuente: Elaboración propia.

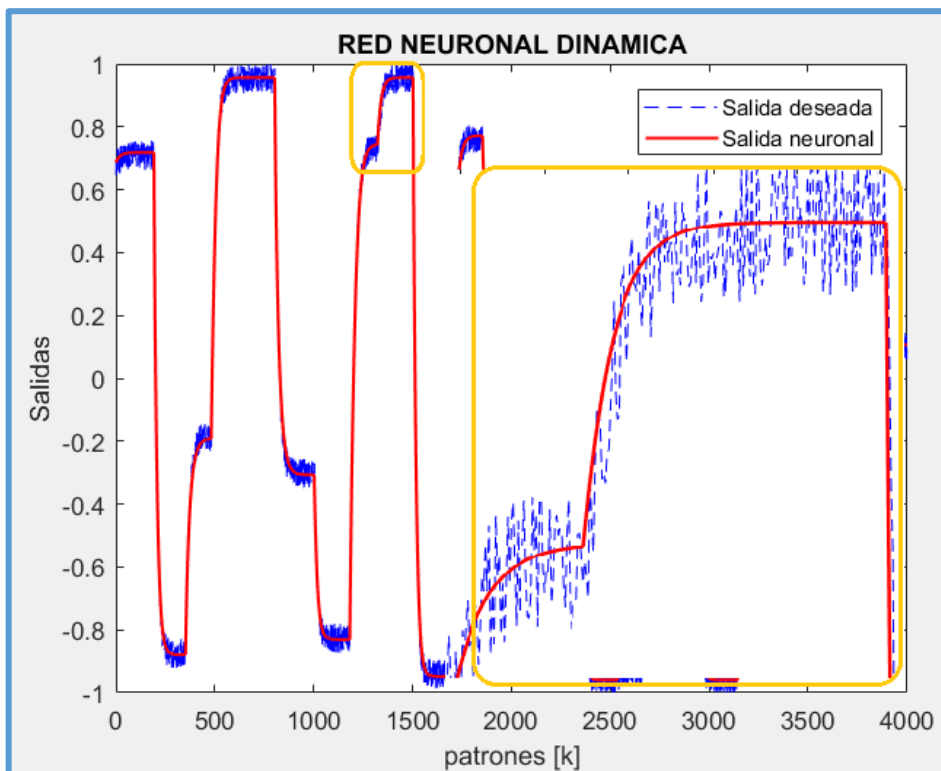


Figura 22: Respuesta de la identificación con una red neuronal dinámica, se puede observar que elimina al ruido durante el modelamiento.

Fuente: Elaboración propia.

Como se puede ver, la red neuronal dinámica, tiene mayor inmunidad al ruido, este es el punto fuerte de este tipo de redes, ya que este tipo de modelamiento ayudará a la robustez del sistema cuando se requiera controlar. En contra parte la red neuronal estática identifica inclusive el ruido presentado en el sistema, pero tiene a favor que la convergencia de la identificación es más rápida comparada a la red dinámica. Para este caso, hubo un tiempo de convergencia de 15 minutos para la red estática y de 80 minutos para la red dinámica. En este trabajo se va a utilizar la identificación o modelamiento del sistema empleando redes neuronales dinámicas, por dos motivos:

Se quiere presentar un nuevo método de identificación de sistemas utilizando redes neuronales dinámicas, el cual se presenta como un aporte en este tema, que es muy poco tratado.

La inmunidad al ruido que presenta este tipo de modelamiento ayudará a mejorar la robustez del control de la planta, ya que no considerará al ruido presente que pueda distorsionar la salida del sistema.

En el siguiente capítulo se explicará con mayor detalle el procedimiento utilizado para la identificación de la planta con redes neuronales dinámicas y el control PID autosintonizado del mismo, utilizando redes neuronales y control adaptativo.

CAPÍTULO III. DISEÑO DEL CONTROLADOR CON AUTOSINTONÍA PID BASADO EN MODELO Y CONTROL ADAPTATIVO

3.1. Identificación de sistemas con redes neuronales dinámicas.

3.1.1. Descripción del proceso de la planta con ganancia variable usada para la identificación del sistema.

La planta a ser usada como objeto de estudio (para identificación de la planta) en la presente tesis es una planta de neutralización de pH, cuyo modelo es tomado del trabajo de tesis realizado por Hau (2009). En dicho trabajo muestra la identificación de la planta usando el método Hammerstein polinomial para la etapa no lineal y el modelo ARX para la etapa lineal. La presente tesis tomará la forma del modelo hallado en Hau (2009), la cual servirá para representar a nuestra planta, a ella se le inyectarán las señales aleatorias de entrada para generar las señales de salida y éstas serán usadas para la identificación del sistema con redes neuronales dinámicas, para posteriormente desarrollar el controlador autosintonizable PID usando redes neuronales (Nelson & Yolanda (2006)).

En Hau (2009) se usa soluciones ácidas y bases para el control del pH, obteniendo sal y agua como producto final. La solución ácida que empleó fue ácido clorhídrico y la solución base fue el hidróxido de sodio (NaOH). El reactor en la cual se realiza la mezcla de ambas soluciones tiene un volumen constante de 1.8 litros y se asume que no hay pérdidas de calor por la reacción exotérmica que se produce al mezclar las soluciones.

Considera el control de pH, manipulando el flujo de la solución base cuyo flujo debe variar entre 0.384 l/min hasta 0.0034 l/min.

El flujo de la solución ácida se considera constante a un valor de 0.4 l/min.

Las concentraciones molares tanto en la solución base y ácida se considera de 0.0005 moles. En la figura 23 se muestra el reactor utilizado para el modelamiento de la planta.



Figura 23: Reactor usado para la neutralización de pH.

Fuente: Jorge Luis Neyra Hau Yon. Control Predictivo no Lineal basado en modelación Hammerstein polinomial aplicado a un módulo de pH. Universidad de Piura 2009.

En la etapa de identificación del sistema, se justifica el trabajo en la zona en la cual el pH varia hasta un valor de 3.8, ya que a partir de allí, se marca una alta no linealidad de la planta, (entre un pH de 4 hasta un pH 10), tal como se indica en la figura 24, zona en la cual, según menciona en Hau (2009), se tendría que trabajar con unos sistemas más elaborados.

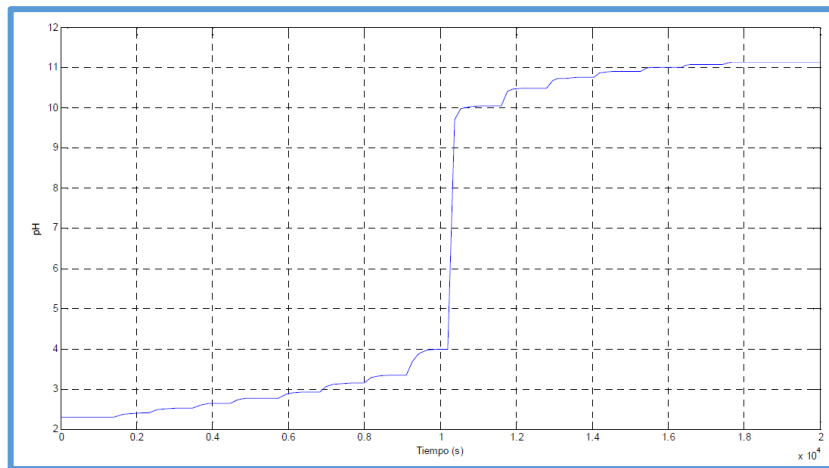


Figura 24: Respuesta de la planta a una señal escalón del caudal de NaOH.

Fuente: Jorge Neyra Hau Yon. Control Predictivo no Lineal basado en modelación Hammerstein polinomial aplicado a un módulo de pH. Universidad de Piura 2009.

Durante el proceso de identificación, en Hau (2009), se presentaron diversas entradas tipo escalón de igual magnitud como se muestra en la figura 25, las cuales al ser presentadas al sistema, determinan a la salida un valor de pH con ganancia estática variable tal como se observa en la figura 26. Se observa que mientras aumenta el valor

de pH, esta se hace más sensible a pequeños cambios del hidróxido de sodio, reconociéndose que este sistema es un sistema no lineal.

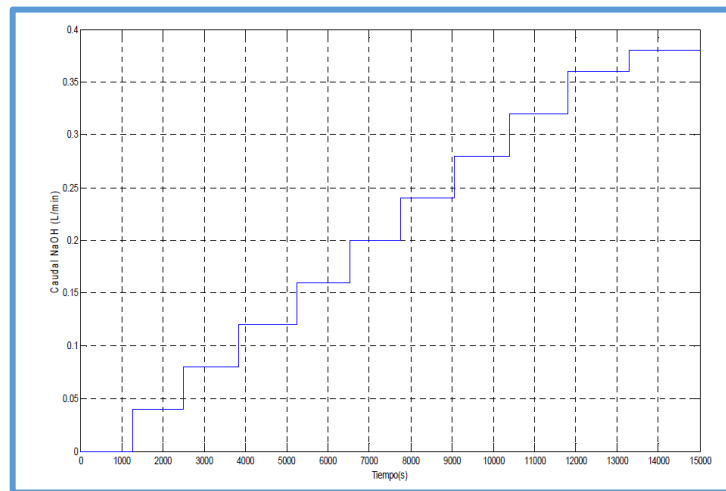


Figura 25: Señales tipo escalón inyectadas a la entrada del sistema.

Fuente: Jorge Luis Neyra Hau Yon. Control Predictivo no Lineal basado en modelación Hammerstein polinomial aplicado a un módulo de pH. Universidad de Piura 2009.

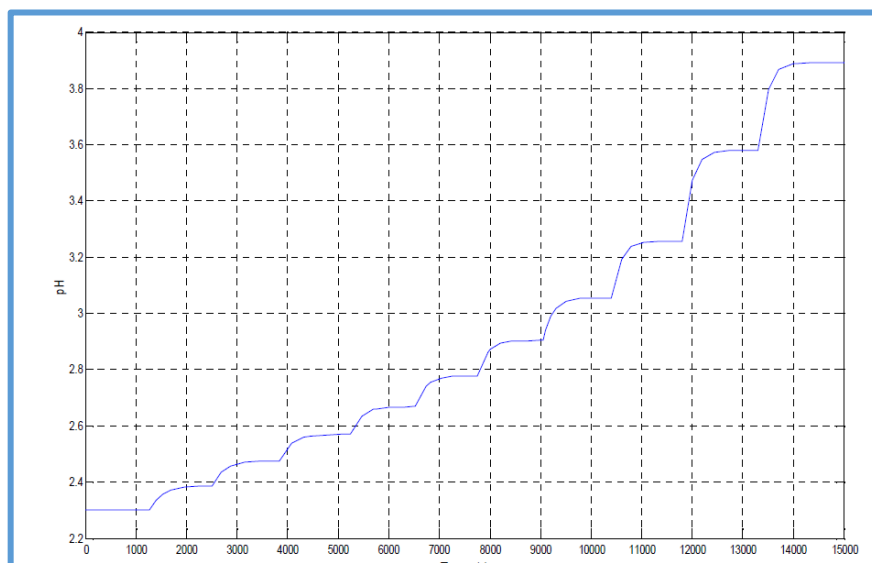


Figura 26: Respuesta del sistema a las entradas escalón. Se observándose una ganancia estática variable.

Fuente: Jorge Luis Neyra Hau Yon. Control Predictivo no Lineal basado en modelación Hammerstein polinomial aplicado a un módulo de pH. Universidad de Piura 2009.

En la etapa de identificación, mostrada en Hau (2009), se utiliza el método Hammerstein polinomial para la identificación de la no linealidad del sistema en su capítulo 3. Allí, presenta una cantidad de señales tipo escalón, las cuales son presentadas al sistema (mientras más señales, más preciso resultará el polinomio); el polinomio obtenido que representa la no linealidad es obtenido por la reducción de mínimos cuadrados que relaciona las entradas presentadas y las salidas obtenidas.

Aplicando el algoritmo y procedimiento correcto, obtiene el polinomio indicado en la expresión (61).

$$P(x) = 0.0021x^7 - 0.034x^6 + 0.21x^5 - 0.57x^4 + 0.3x^3 + 1.98x^2 - 4.4x + 2.899 \quad \dots\dots (61)$$

Y para la identificación de la dinámica lineal utiliza la estructura ARX la cual halla una función de transferencia que represente de una forma adecuada a esta dinámica. Para ello presenta una señal pseudoaleatoria al ingreso de la planta PRS, con la cual se obtienen señales de salida de pH. Estos datos de entrada y salida luego son tratadas para la identificación del sistema usando la arquitectura ARX, resultando el siguiente modelo expresado en (62):

$$F(z^{-1}) = \frac{0.01161z^{-1} - 0.01507z^{-2}}{1 - 1.62z^{-1} + 0.6456z^{-2}} \quad \dots\dots\dots (62)$$

Finalmente, el modelo no lineal, como el lineal, que representan el comportamiento dinámico de la planta puede ser graficado según la figura 27:

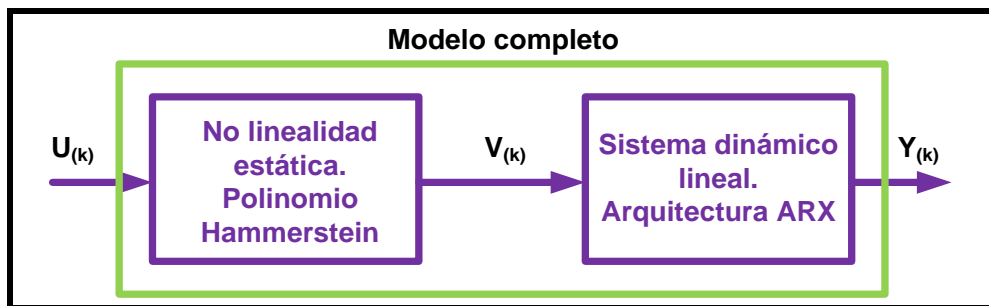


Figura 27: Diagrama de bloques del modelo completo de la planta pH.
Fuente: Elaboración propia.

Como se explicó en un inicio, se tomará el modelo desarrollado en Hau (2009), la cual, igualmente presentará un polinomio no lineal y una función de transferencia lineal y este modelo en su conjunto será nuestra planta a identificar y controlar usando redes neuronales dinámicas y control adaptativo, con la diferencia que se considerará ruido a la salida de la misma.

Transformando la ecuación (62) a tiempo continuo con el método mantenedor de orden cero – ‘zoh’ y utilizando el comando d2c de Matlab, con tiempo de muestreo igual a 5 segundos, (tiempo obtenido heurísticamente como se explica en párrafos siguientes), resulta la función de transferencia mostrada en la expresión (63):

$$G(s) = \frac{-0.0006704s + 0.001321}{s^2 + 0.08752s + 0.001253} \quad (63)$$

Polinomio Hammerstein que representa la ganancia estática variable (parte no lineal):

$$P(x) = 0.0021x^7 - 0.034x^6 + 0.21x^5 - 0.57x^4 + 0.3x^3 + 1.98x^2 - 4.4x + 2.899 \quad \dots (64)$$

Nuestra planta con la cual vamos a trabajar, basada en los conceptos anteriores, es la mostrada en la figura 28.

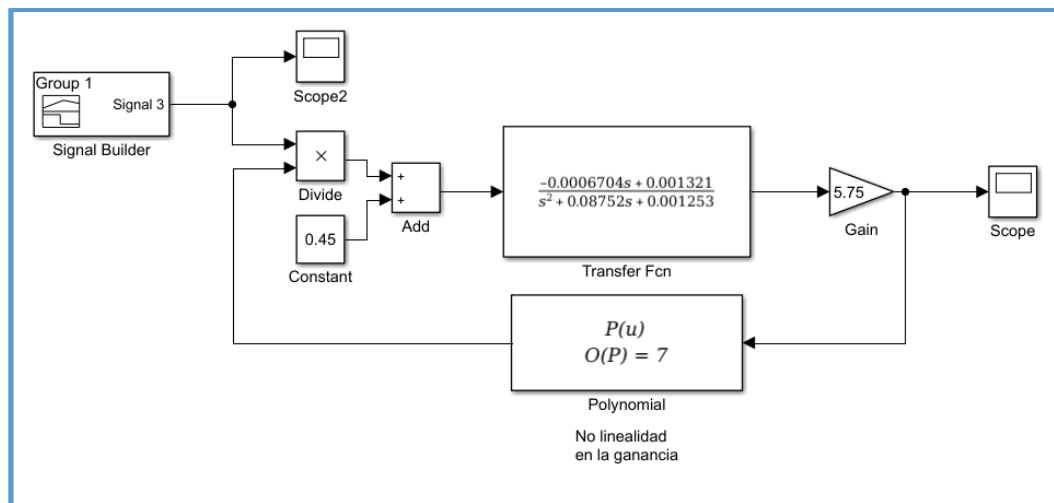


Figura 28: Planta a modelar y controlar en el presente trabajo.

Fuente: Elaboración propia.

En la figura 30, se muestra los valores de la ganancia estática ante entradas tipo escalón de la figura 29, como se ve, esta ganancia es variable, lo que marca la no linealidad de la planta.

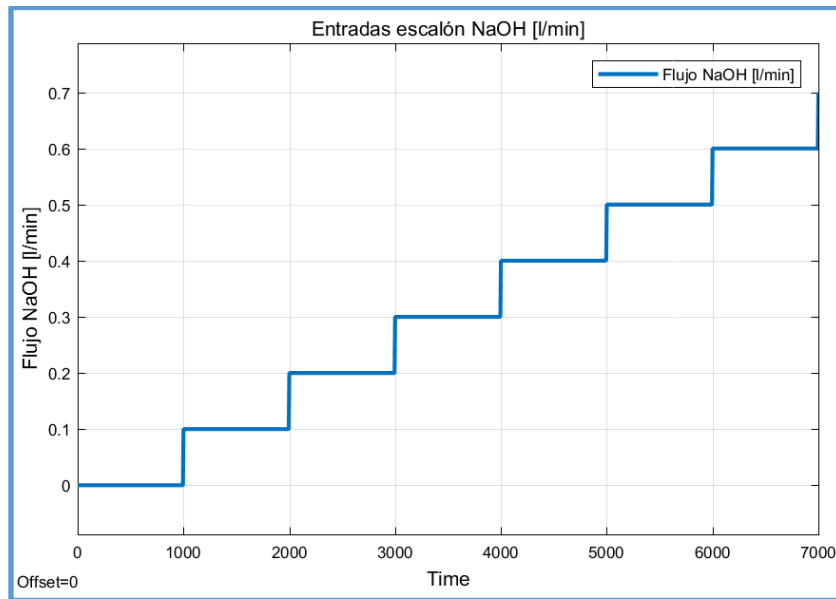


Figura 29: Entradas tipo escalón al ingreso de la planta.
Fuente: Elaboración propia.

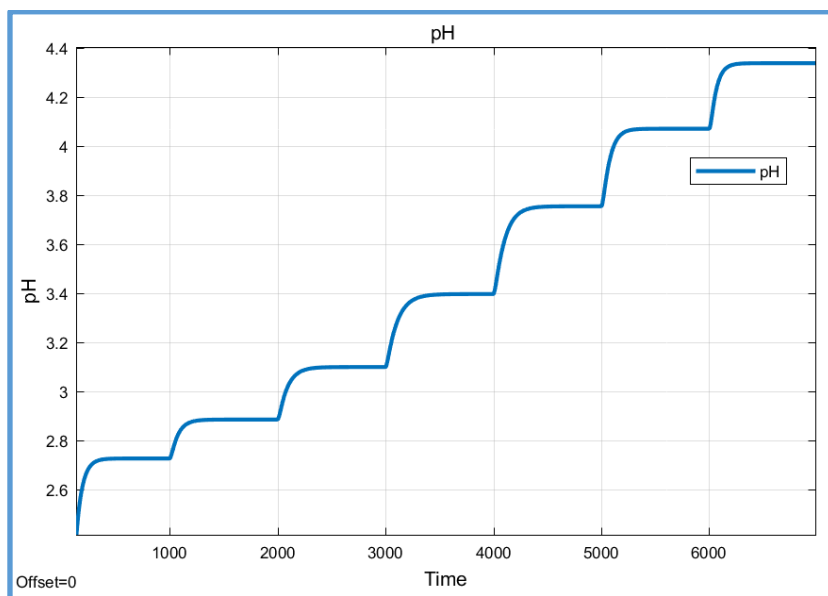


Figura 30: Respuesta pH de la planta, se observa la ganancia estática variable.
Fuente: Elaboración propia.

A continuación, en las figura 31 y 32 se muestra la señal de entrada tipo escalón y la respuesta de la planta ante esta señal, respectivamente.

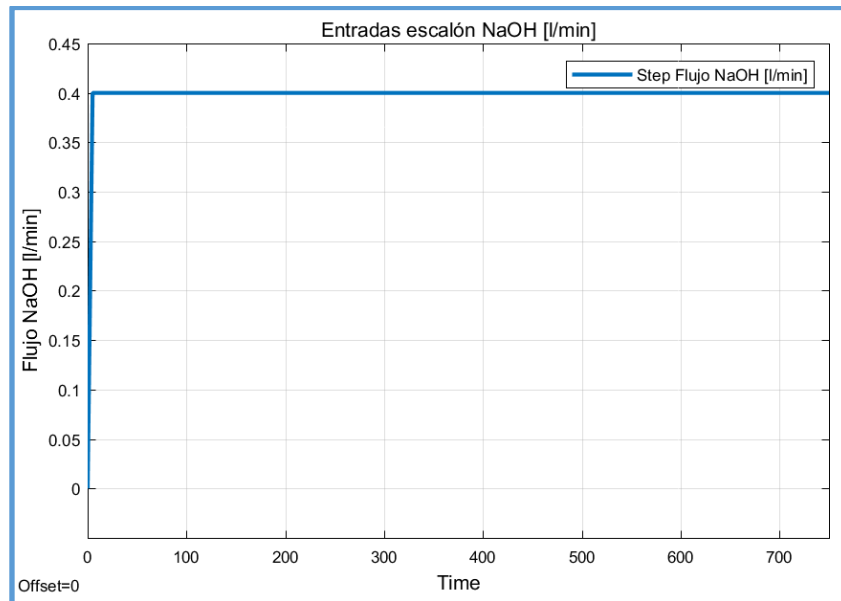


Figura 31: Señal tipo escalón presentado al sistema.
Fuente: Elaboración propia.

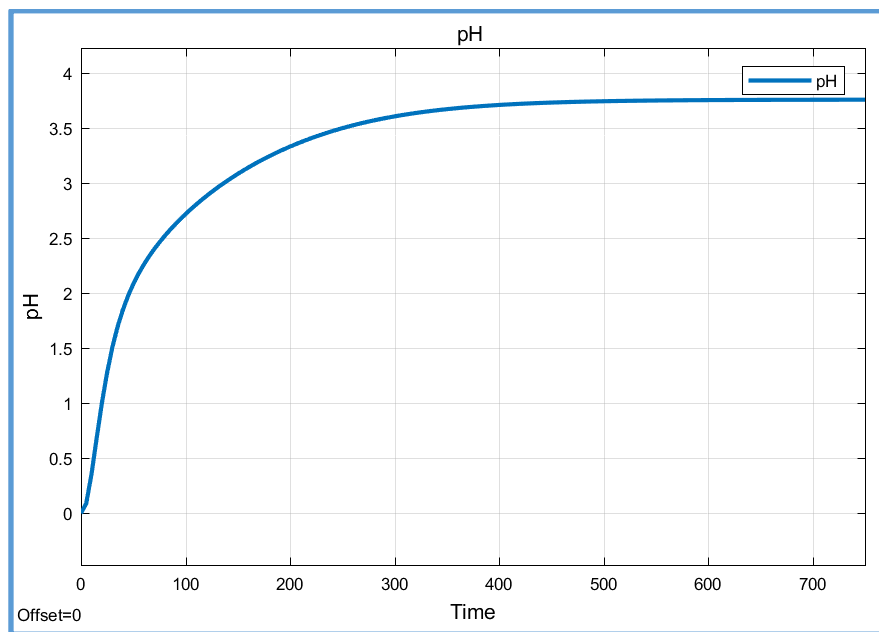


Figura 32: Respuesta de la planta a la señal tipo escalón.
Fuente: Elaboración propia.

Como regla general se acepta que el tiempo de muestreo sea menor que $[1/15, 1/20]$ del 80% del valor estable del sistema. De la figura 32 se observa que el tiempo estimado en alcanzar el 80% del valor estable es de 130seg, por lo que el tiempo estimado a considerar sería $\frac{130}{20} = 6.5$ seg. Considerándose **5 segundos** como el tiempo de muestreo para la identificación y diseño del controlador de la planta. Es decir:

$$T_s = 5 \text{ seg.} \dots\dots\dots (65)$$

Siendo T_s el tiempo de muestreo.

La cantidad de datos de entrada y salida fueron 4000, en un lapso de 5.5 horas.

3.1.2. Obtención del modelo del proceso aplicando redes neuronales dinámicas e identificación de sistemas.

Para la obtención del modelo del sistema, se aplicará la teoría de redes neuronales dinámicas explicada en el capítulo 2, para ello, el procedimiento a seguir será:

Generar una señal pseudo aleatoria que contenga la mayor cantidad de datos en un tiempo no muy prolongado. Esta señal será aplicada al ingreso de la planta de la figura 28, con esto, se obtendrá las señales de salida de nuestra planta, teniendo como referencia la figura 33, cabe resaltar que a la salida de la planta de la figura 33 se ha añadido una señal de ruido, que hace que la planta se acerque más a una real. Además de demostrar la fortaleza de la red neuronal dinámica ante sistemas con ruido. Una vez obtenidos los datos de entrada y salida, estos serán aplicados a la arquitectura de la red neuronal dinámica mostrada en la figura 33. El modelo se obtendrá aplicando la teoría mostrada en el capítulo 2, empleando las ecuaciones desde la 36 a la 60, considerando para esto la cantidad de neuronas en la capa de entrada igual a 4 ($N_1 = 4$), la cantidad de neuronas en la capa oculta igual a veinte ($N_2 = 20$) y la cantidad de neuronas en la capa de salida igual a 2 ($N_3 = 2$) es decir el modelamiento con una red neuronal dinámica con dos salidas.

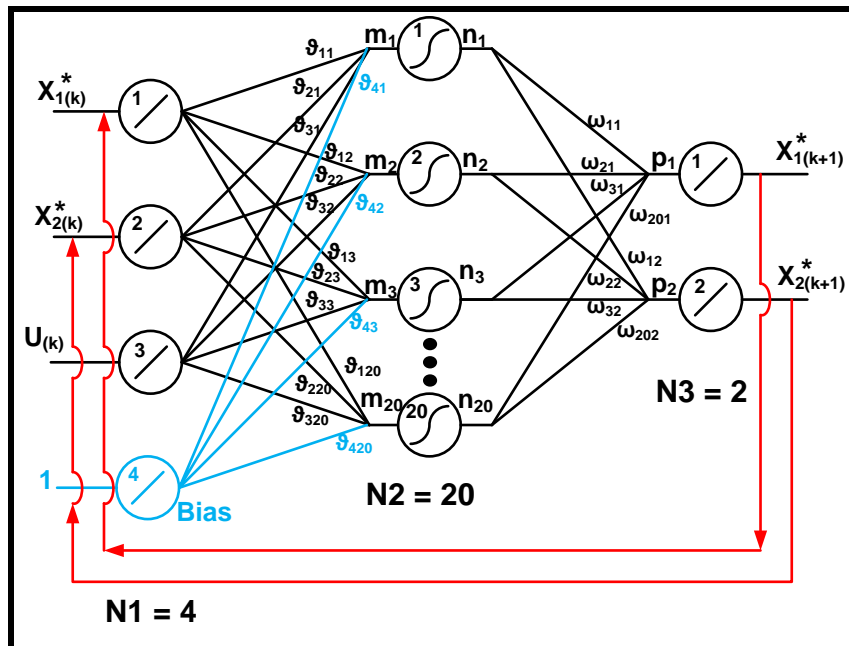


Figura 33: Arquitectura de la red neuronal a utilizar para la identificación del sistema (4, 20, 2)
Fuente: Elaboración propia.

En el anexo 1, se muestra el script elaborado en Matlab para el modelamiento del sistema indicado. Para mostrar cuán cercano es la planta modelada frente a la planta “real” (figura 28), se utilizará el concepto de VAF, la cual, indica que si el VAF es $\geq 97\%$, el modelamiento es aceptado (Babuška, Setnes, Molloy y Van der Veen. (2001).), lo mismo para la validación del modelo. El VAF representa la varianza en porcentaje en la comparación de dos señales temporales. Este índice, el cual se define mediante la ecuación (66), indica que cuanto más cercano al valor de 100% del valor del VAF, mayor es la precisión del modelo identificado.

$$\text{VAF} = \left[1 - \frac{\text{var}(y - y_m)}{\text{var}(y)} \right] \times 100\% \quad \dots\dots\dots (66)$$

Donde:

y , es la variable real del proceso.

y_m , es la variable estimada por el modelo neuronal.

En la figura 34, se muestra el sistema a identificar y en las figuras 35 y 35B se muestra la señal PRBS a inyectar a sistema para la identificación de la planta (señal de entrada ‘u’). Las figuras 36 y 36B muestran la respuesta de la planta modelada versus la planta

sujeta a la identificación, observando que el modelo neuronal rechaza el ruido presente en el sistema. La tabla 1 muestra los valores del VAF y RMSE obtenidos, de la cual, estos valores indican una buena identificación del modelo, por lo que el modelo es aceptado, siendo el próximo paso, la validación del mismo.

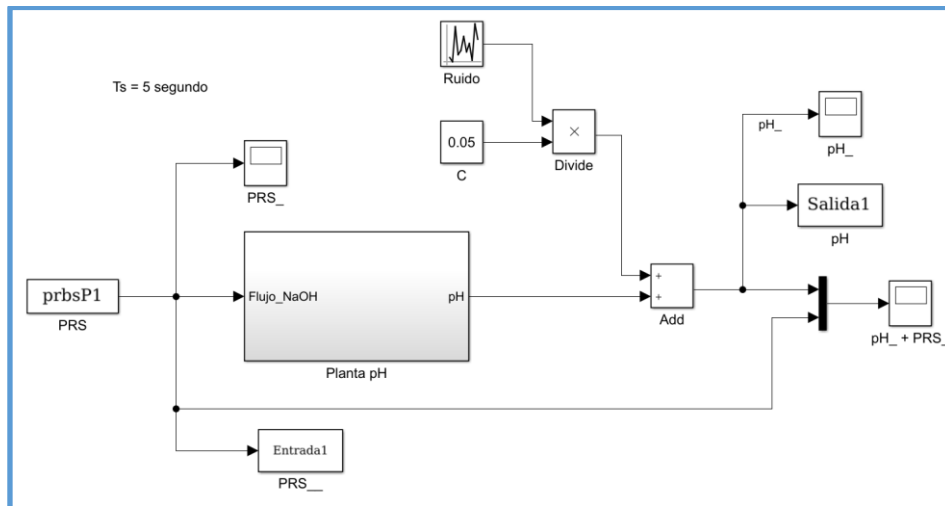


Figura 34: Señales de entrada y salida para la identificación del sistema. Se observa la generación de ruido a la salida.

Fuente: Elaboración propia.

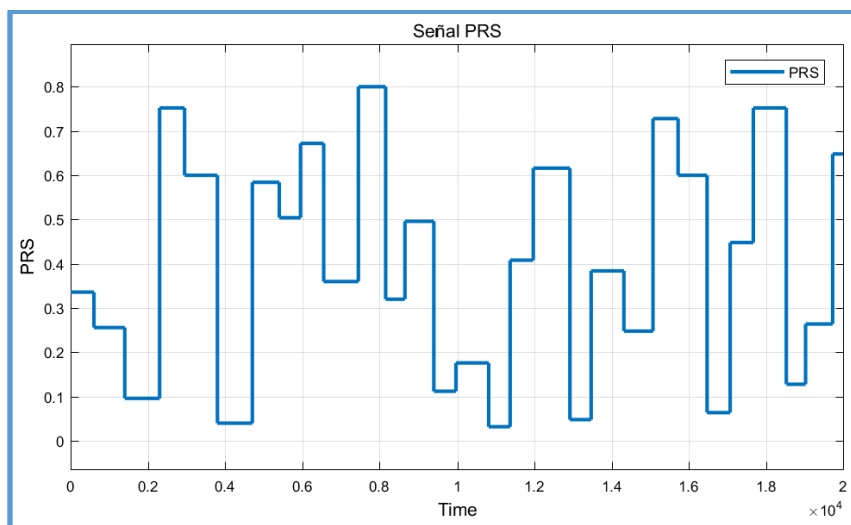


Figura 35: Señal PRBS sin escalar empleada para generar la salida de pH para la identificación del sistema.

Fuente: Elaboración propia.

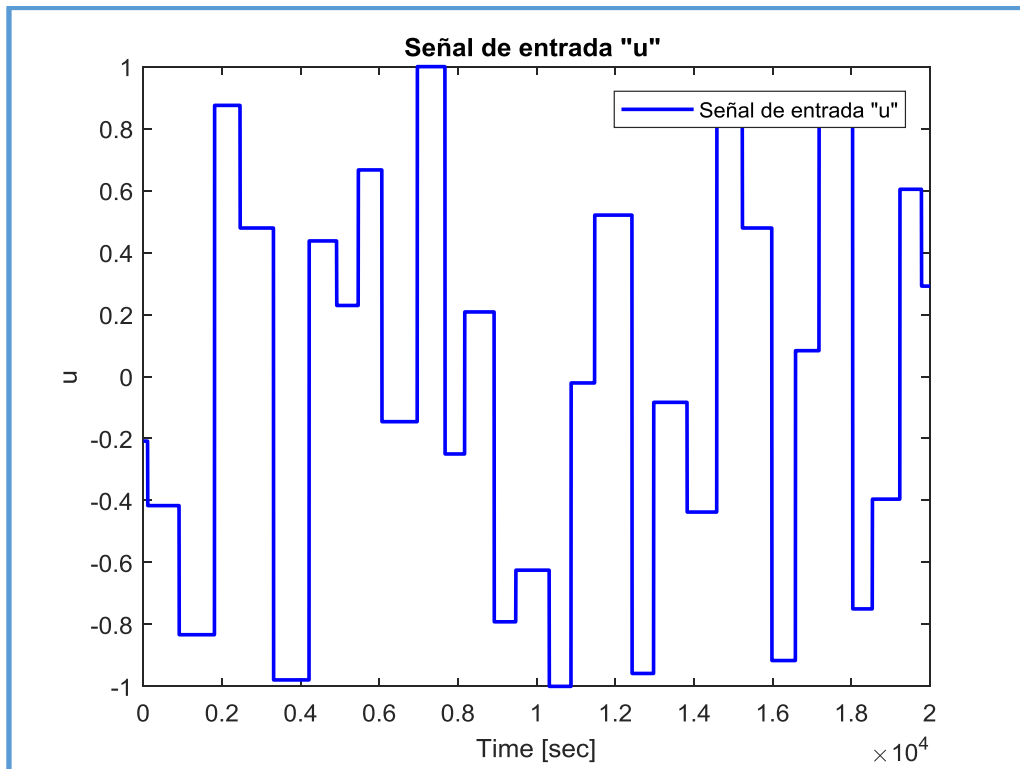


Figura 35B. Señal PRS escalada empleada para generar la salida de pH para la identificación del sistema en la red neuronal, valor comprendido entre [-1, +1].

Fuente: Elaboración propia.

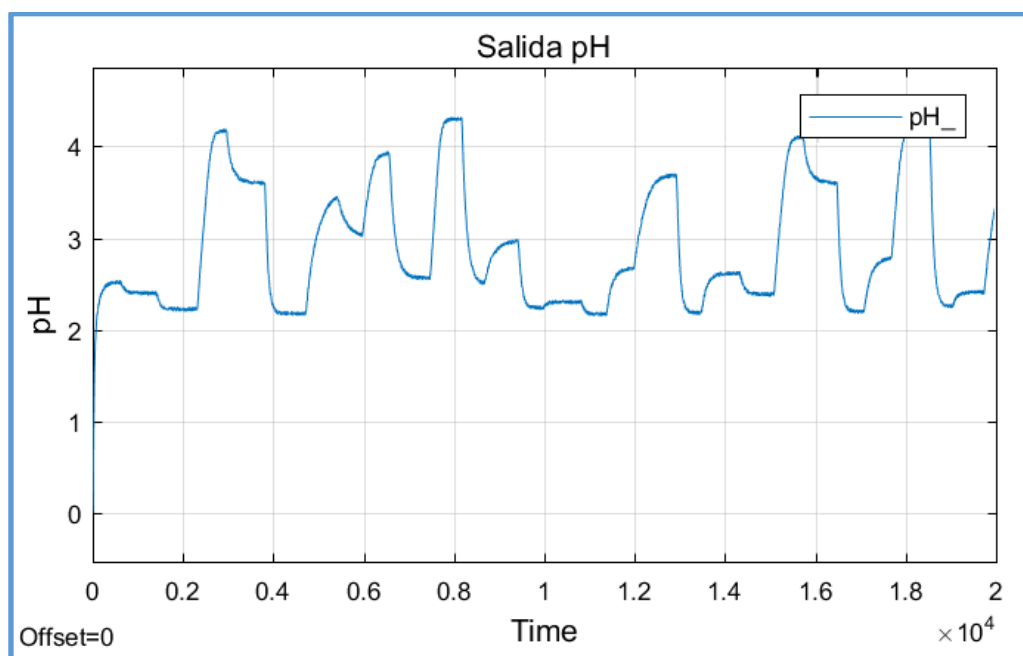


Figura 36: Salida de la planta sin escalar. Valor comprendido de pH entre 2.5 y 4.5.

Fuente: Elaboración propia.

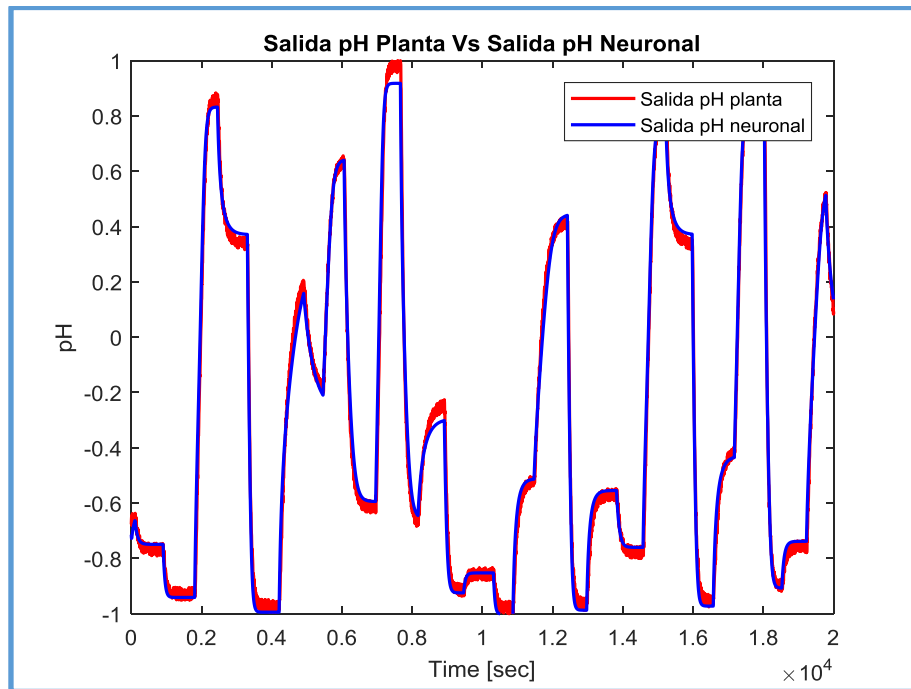


Figura 36B. Comparación salida pH de la planta VS salida pH del modelo neuronal escaladas, se observa que el modelo neuronal rechaza el ruido.

Fuente: Elaboración propia.

Tabla 1: Indicadores de proximidad del modelo neuronal hallado.

VAF	RMSE	FIT
99.7319	0.0342	0.9482

Fuente: Elaboración propia.

De la gráfica 36 se observa que el valor de pH a controlar estará comprendida entre los valores [2.5, 4.5], es decir $\text{pH} = [2.5, 4.5]$.

3.1.3. Validación del modelo.

Para la validación del modelo, al igual que para la identificación del sistema, se generará una señal pseudoaleatoria, distinta a la usada para la identificación. La cual, será presentada al ingreso del modelo obtenido en el paso anterior y a la planta “real”. Las salidas de ambos sistemas, luego serán comparadas utilizando el VAF (ecuación 65). Si el VAF resulta $\geq 97\%$, el modelo será aceptado y es con este modelo que se trabajará para el diseño del controlador PID autosintonizado.

En la figura 37 se muestra la señal pseudoaleatoria generada para la validación de la planta. La figura 38 muestra los resultados obtenidos a la salida del modelo obtenido y

de la planta a modelar, remarcando que el modelo obtenido con la red neuronal dinámica cancela el ruido presente en el sistema, es decir, tendrá mayor resistencia al ruido. La tabla 2 muestra los valores del VAF y RMSE obtenidos, de la cual, estos valores indican que hubo una buena identificación del modelo realizado en el paso anterior. Con esto, este modelo será utilizado para el diseño del controlador PID autosintonizado.

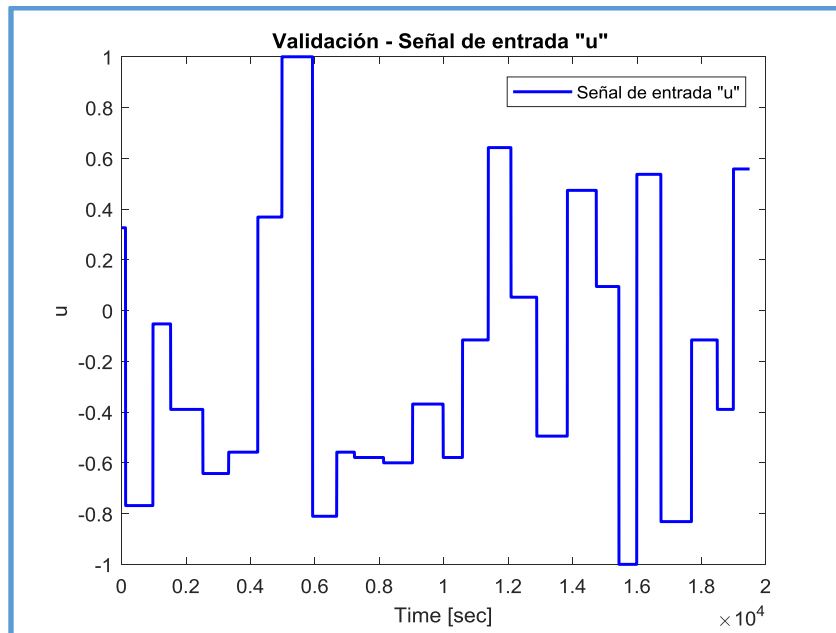


Figura 37: Señal PRS empleada para generar la salida de pH para la validación del sistema.
Fuente: Elaboración propia.

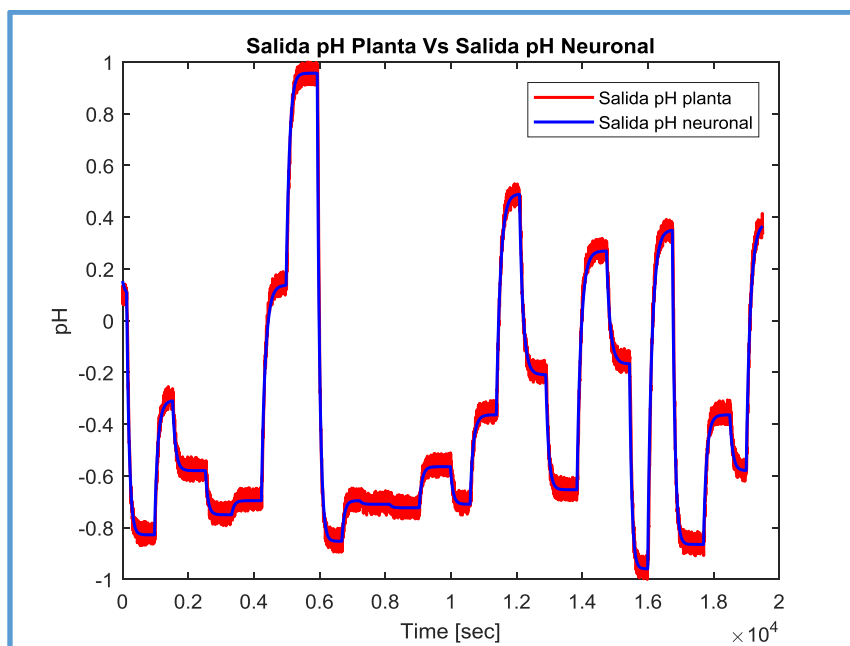


Figura 38: Comparación salida pH de la planta VS salida pH del modelo neuronal de los datos tomados para la validación observándose que el modelo neuronal es más inmune al ruido.

Fuente: Elaboración propia.

Tabla 2: Indicadores de proximidad del modelo neuronal hallado después de la validación.

VAF	RMSE	FIT
99.4990	0.0339	0.9282

Fuente: Elaboración propia.

Donde claramente se observa la gran proximidad que tiene el modelo hallado frente a la planta estudiada.

Concluyendo que el modelo desarrollado es aceptable.

3.2. Diseño e implementación del controlador con autosintonía PID basado en modelo.

3.2.1. Control adaptativo basado en modelo de referencia (MRAC).

Un control adaptativo como concepto, se puede definir, como aquel controlador que cambia sus parámetros de control cuando la planta a controlar cambia alguna de sus condiciones de operación (como su dinámica de operación o alguna perturbación presente en ella), para mantener a la planta dentro de los parámetros elegidos por el operador.

Si bien el cambio de parámetros puede darse de forma manual o automática en el controlador, se define que en un controlador adaptativo, estos cambios suceden de forma automática, tal como se indica en Pirabakaran y Becerra (2004).

El esquema de un control adaptativo básico, se muestra en la figura 39, en la cual se pueden identificar 02 lazos: un lazo externo, el cual tiene la realimentación negativa clásica el cual actúa como regulador y, un lazo interno, encargado de medir el índice de funcionamiento del sistema, el cual es comparado con un índice deseado, la diferencia entre ambas (error) genera la adaptación de los parámetros del controlador por medio de un algoritmo de optimización.

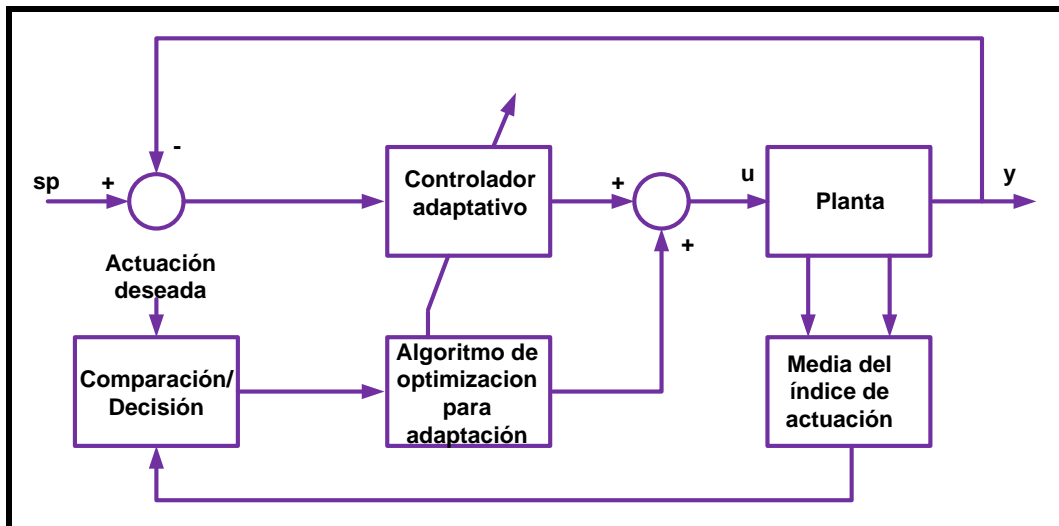


Figura 39: Controlador adaptativo básico.

Fuente: Elaboración propia.

De acuerdo a como se desarrollen los bloques mostrados en la figura 39, dan como resultado diversos tipos de controladores adaptativos, pudiéndose encontrar dos grupos marcados: controladores adaptativos basado en modelo de referencia (MRAC) y los reguladores autoajustable (STR).

Los controladores adaptativos basados en modelo de referencia, intentan seguir y/o alcanzar una señal de entrada definida, es decir seguir el comportamiento en un lazo cerrado de un modelo de referencia previamente definido como se presenta en Pirabakaran & Becerra (2004).

Los controladores adaptativos autoajustables, buscan llegar a un control óptimo, sujeto a algún tipo de controlador y a la obtención de los parámetros del proceso.

El presente trabajo, para el diseño del controlador PID autoajustable, se basará en la arquitectura de un controlador adaptativo basado en modelo de referencia (MRAC), por lo que el controlador adaptativo autoajustable no será tratado en adelante.

Como se indicó, el control adaptativo basado en modelo de referencia (MRAC) fue propuesto originalmente para resolver un problema de control, de la cual, sus especificaciones están dados en términos de un modelo de referencia, y es este modelo quien indica como la salida del proceso debe ante una señal de referencia. Siendo esto, el principal objetivo del control adaptativo.

La idea principal, se muestra en la figura 40, el cual es el controlador MRAC original, propuesto por Whitaker en 1958. El sistema que se muestra en la figura 39, consiste de dos lazos: el lazo interno, el cual es el lazo común realimentado, conformado por el controlador y el proceso. Siendo los parámetros del controlador ajustados por el lazo externo, que busca de alguna manera hacer que el error 'e' entre la salida del proceso 'y' y la salida del modelo 'y_m' sea lo más pequeño posible. El lazo externo es denominado el lazo de adaptación como se indica en Astrom & Wittenmark (1989).

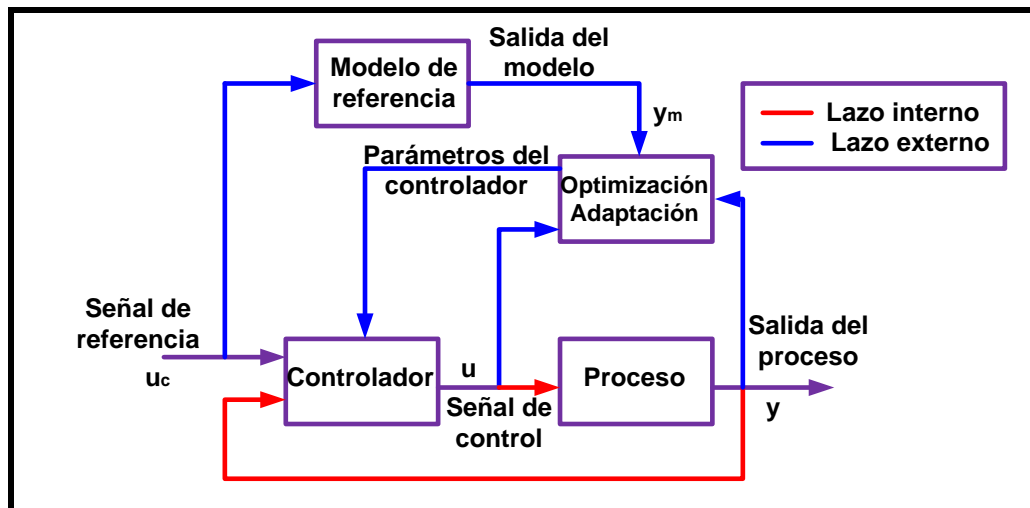


Figura 40: Diagrama de bloques de con control adaptativo basado en modelo de referencia.
Fuente: Elaboración propia.

Con lo descrito en referencia al controlador adaptativo basado en modelo (MRAC), este concepto formará parte de la arquitectura para el diseño del controlador PID autosintonizado, el cual se presentará a continuación.

3.2.2. Autosintonía PID usando redes neuronales y control adaptativo basado en modelo de referencia.

3.2.3. Arquitectura del sistema.

La arquitectura en la cual se basa el presente trabajo se muestra en la figura 41.

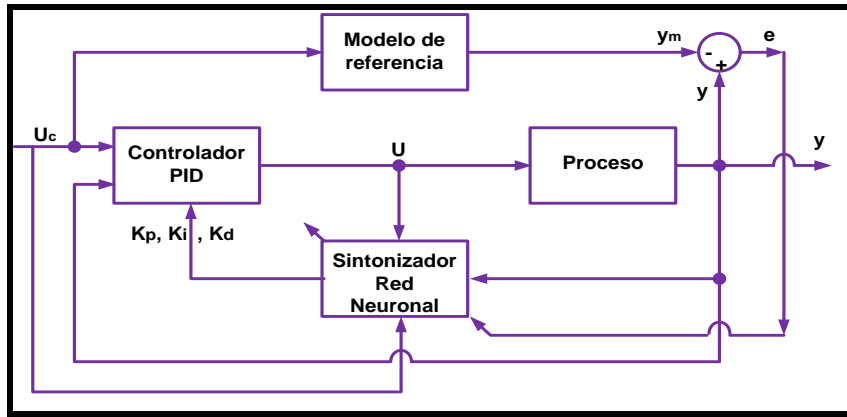


Figura 41: Arquitectura autosintonía PID con redes neuronales y control adaptativo.
Fuente: Elaboración propia.

De esta arquitectura (figura 41) hay que resaltar el bloque sintonizador basado en una red neuronal estática (capítulo 2), de la cual, las salidas de esta red neuronal serán los parámetros del controlador PID (parámetros adaptables) ,proporcional ΔK_p , integral ΔK_i , derivativo ΔK_d , y sus entradas serán seleccionadas de acuerdo al problema presentado, en este caso sus entradas serán: $[U(n), U(n - 1), y(n), y(n - 1), U_c(n), U_c(n - 1)]$, tal como se muestra en la figura 42.

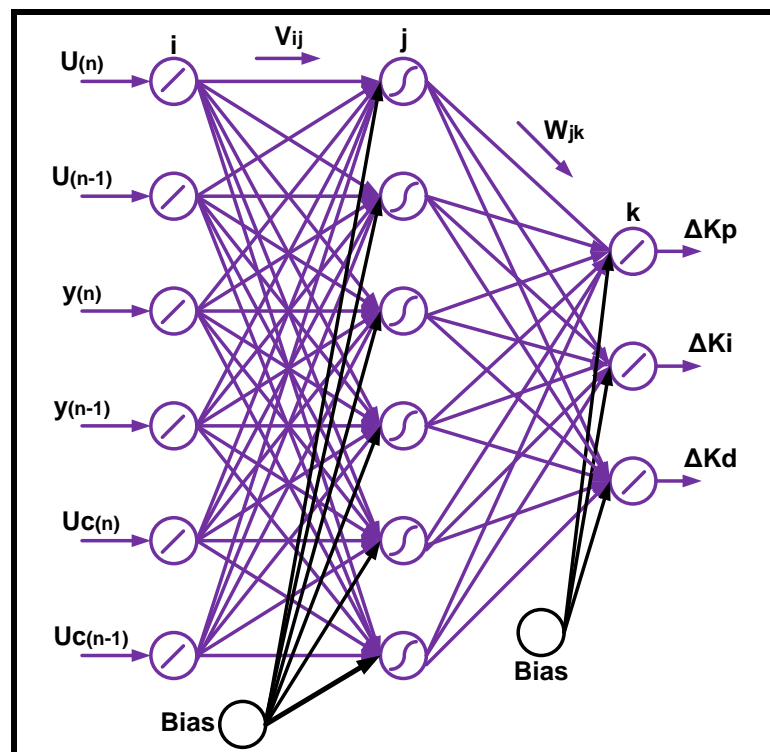


Figura 42: Red neuronal para la autosintonía PID (Sintonizador neuronal)
Fuente: Elaboración propia.

El método a desarrollar en el presente trabajo, calcula los parámetros del controlador en línea (proporcional ΔK_p , integral ΔK_i , derivativo ΔK_d). Pero, para determinar estos

parámetros, como es sabido cuando se trabaja con una red neuronal, es necesario tener los patrones de entrada y salida ya conocidos a priori. En este caso, los patrones de salida no son conocidos de forma anticipada (no se conocen los patrones de salida $\Delta K_p, \Delta K_i, \Delta K_d$), por lo que un método estándar de entrenamiento supervisado como es el método de retropropagación de errores (back-propagation), no puede ser aplicado, como se muestra en Yang & Wu (2016). Motivo por el cual, este método sufrirá algunos cambios, siendo uno de ellos: que los pesos de la red neuronal serán calculados para minimizar la función de costo indicada en la ecuación (67) utilizando el algoritmo gradiente descendiente. Observándose que la función de costo no está en función del error obtenido entre la salida estimada por la red neuronal y la salida del sistema o proceso como era el caso tratado en el capítulo 2 referente a una red neuronal estática, sino que el error, ahora está dado por la diferencia entre la salida del modelo de referencia y la salida del proceso.

$$J = \frac{1}{2} e^2 = \frac{1}{2} (y - y_m)^2 \quad \dots\dots\dots (67)$$

Donde:

y , salida del proceso

y_m , Salida del modelo de referencia.

La función de transferencia del modelo de referencia, está dada por:

$$G_m = \frac{Y_m(s)}{U_c(s)} = \frac{\omega_n^2}{s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2} \quad \dots\dots\dots (68)$$

Que tiene una ganancia unitaria en estado estacionario

Donde:

ω_n , es la frecuencia natural.

ζ , es la razón de amortiguamiento.

Recordando que el modelo de referencia determina la salida deseada para una entrada determinada.

Como se indicó, el modelo del error 'e' es calculado usando la salida del proceso y la salida del modelo de referencia (expresión 66), luego el error 'e' es usado para actualizar los pesos sinápticos de la red neuronal indicada como el bloque sintonizador red neuronal de la figura 3.18. Para el propósito de actualizar estos pesos sinápticos, el método retropropagación de errores además sufrirá los siguientes cambios siguiendo los criterios indicados a continuación.

El algoritmo de control PID en tiempo discreto utilizado será el de tipo velocidad, mencionando que otro algoritmo PID es el de tipo posición la cual no se tratará en esta tesis, quedando solo como referencia.

La función de transferencia de un controlador PID está dada por la expresión (69), (Khan & Rapal (2006)):

$$G_c(s) = \frac{U(s)}{E(s)} = k_c \cdot \left[1 + \frac{1}{T_i s} + T_d s \right] \quad \dots\dots\dots (69)$$

Donde K_c , T_i y T_d son denominados la ganancia proporcional, el tiempo integral y el tiempo derivativo respectivamente.

La expresión (69) puede ser escrita en el dominio del tiempo como se indica en la expresión (70):

$$u(t) = k_c \left[e(t) + \frac{1}{T_i} \int_{-\infty}^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right] \quad \dots\dots\dots (70)$$

Discretizando la expresión (70) usando el modo trapezoidal para integración numérica, se obtiene la ecuación (71):

$$u(n) = u(n - 1) + K_p [e(n) - e(n - 1)] + K_i e(n) + K_d [e(n) - 2e(n - 1) + e(n - 2)] \quad \dots\dots\dots (71)$$

Donde:

K_p, K_i y K_d , son las ganancias proporcional, integral y derivativa respectivamente.

$u(n)$, la entrada a la planta en el instante nT_s , siendo T_s el tiempo de muestreo, indicado en la expresión (65).

$e(n)$, es el error del control.

Para el ajuste de los parámetros K_p, K_i y K_d , se considerará una red neuronal de tres capas, tal como se muestra en la figura 43, la capa oculta trabajará con una función de activación sigmoidea tipo 2 (ecuación 6). Siendo el número de neuronas de la entrada igual a 6, no se considera bias ($N_1 = 6$), el número de neuronas en la capa oculta igual a 12 ($N_2 = 12$), y el número de neuronas de la capa de salida igual a 3 ($N_3 = 3$). El número de neuronas de la capa de salida corresponde a las ganancias del controlador PID (variables adaptativas en línea durante el control del proceso). En cambio, el número de neuronas de la capa de entrada como en la capa oculta fueron dimensionados por prueba y error, obteniéndose resultados satisfactorios como se demostrará más adelante.

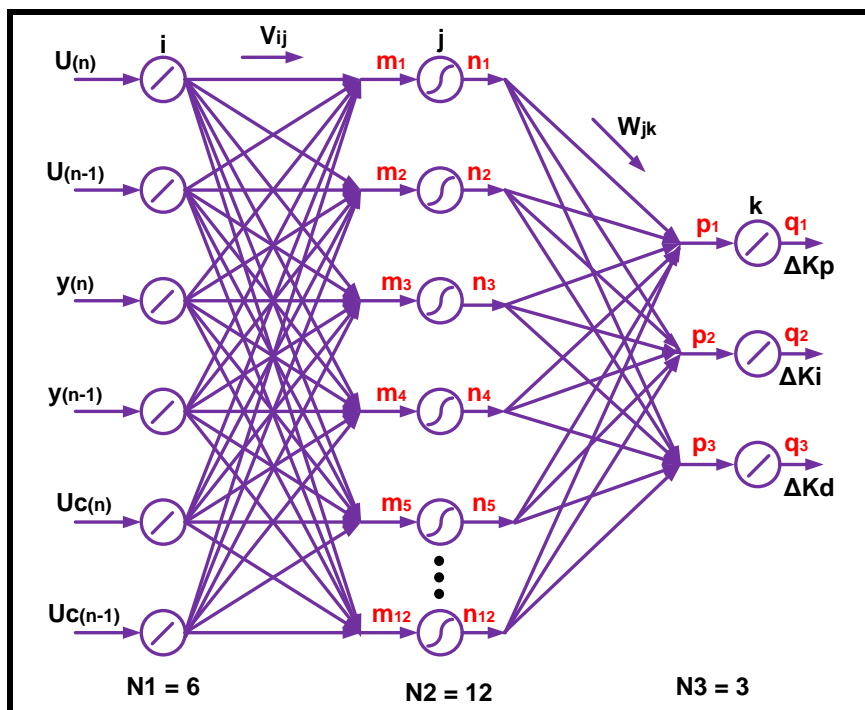


Figura 43: Sintonizador neuronal. Red neuronal estática para hallar los parámetros PID
Fuente: Elaboración propia.

Recordando que la función de costo a usarse es la de la ecuación (67).

Ahora, para la actualización de los pesos sinápticos de la capa oculta y la de la salida, se emplearán las siguientes ecuaciones, teniendo en cuenta la figura 44, de esta figura, el bloque NN emulador es el modelo hallado mediante la red neuronal dinámica en la etapa de identificación del proceso:

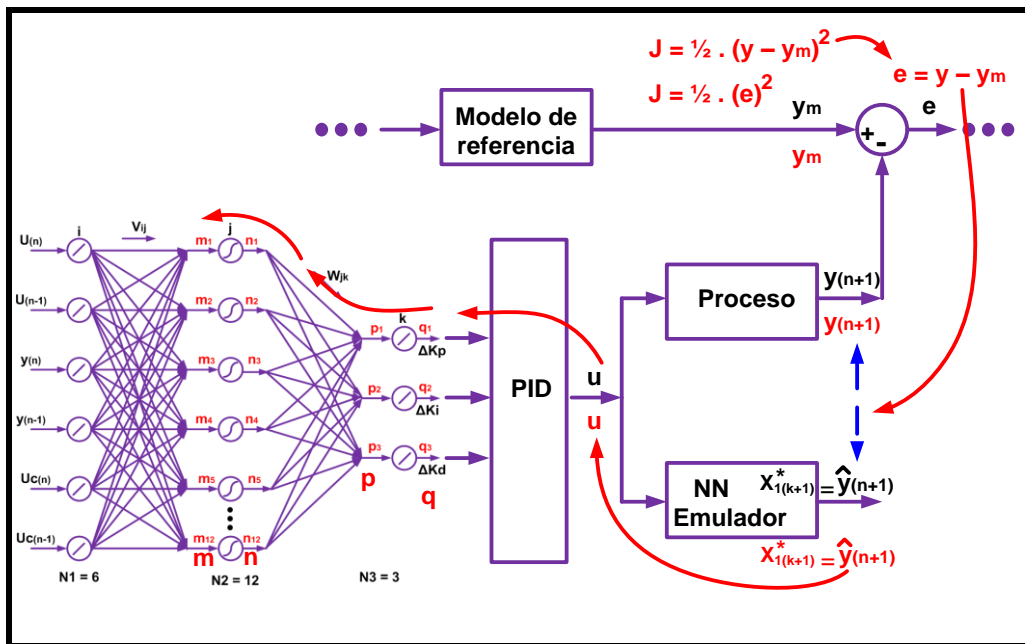


Figura 44: Forma adoptada para la retropropagación en la red neuronal sintonizadora.
Fuente: Elaboración propia.

Para los pesos de la capa de salida (W):

$$\Delta w_{jk}(n + 1) = \eta \cdot \delta_w \cdot o_j + \alpha \Delta w_{jk}(n) \dots \dots \dots (72)$$

Siendo:

η , ratio de aprendizaje.

δ_w , gradiente retropropagación capa salida (W), $\frac{dJ}{dp}$ (figura 44)

o_j , salida de la capa anterior.

α , constante de momento.

Para los pesos de la capa oculta (V):

$$\Delta v_{ij}(n+1) = \eta \cdot \delta_v \cdot o_v + \alpha \Delta v_{ij}(n) \quad \dots\dots\dots (73)$$

Siendo:

η , ratio de aprendizaje.

δ_v , gradiente retropropagación capa oculta (V), $\frac{dJ}{dm}$ (figura 44)

o_v , salida de la capa anterior.

α , constante de momento.

De las ecuaciones (72) y (73), desarrollaremos los gradientes de retropropagación del sistema, guiándonos de la figura 3.22:

$$\delta_w = \frac{dJ}{dp} = \frac{dJ}{de} \cdot \frac{de}{dy} \cdot \frac{dy}{du} \cdot \frac{du}{dq} \cdot \frac{dq}{dp} \quad \dots\dots\dots (74)$$

$$\delta_v = \frac{dJ}{dm} = \frac{dJ}{de} \cdot \frac{de}{dy} \cdot \frac{dy}{du} \cdot \frac{du}{dq} \cdot \frac{dq}{dp} \cdot \frac{dp}{dn} \cdot \frac{dn}{dm} \quad \dots\dots\dots (75)$$

Resolviendo:

$$\frac{dJ}{de} = e \quad \dots\dots\dots (76)$$

$$\frac{de}{dy} = 1 \quad \dots\dots\dots (77)$$

$$\frac{dy}{du} \quad \dots\dots\dots (78)$$

Será el denominado “jacobiano” de la planta, cuyo cálculo será desarrollado en el subtítulo siguiente.

$$\frac{du}{dq} \dots\dots\dots (79)$$

Es la derivada que involucra al desarrollo del algoritmo PID, presentado en la ecuación (61) donde hacemos $K_p = q_1$, $K_i = q_2$, $K_d = q_3$.

Entonces de la ecuación (61), se tiene:

$$\frac{du}{q_1} = e(n - 1) - e(n - 2) \dots\dots\dots (80)$$

$$\frac{du}{q_2} = e(n - 1) \dots\dots\dots (81)$$

$$\frac{du}{q_3} = e(n - 1) - 2 e(n - 2) + e(n - 3) \dots\dots\dots (82)$$

$$\frac{dq}{dp} = 1 \dots\dots\dots (83)$$

La función de activación de la capa de salida es lineal $p = q$.

$$\frac{dp}{dn} = W \dots\dots\dots (84)$$

$$\frac{dn}{dm} = \frac{df(m)}{dm} = \frac{(1 - n) \cdot (1 + n)}{2} \dots\dots\dots (85)$$

Que es la derivada de la función sigmoidea mostrada en la ecuación (86).

3.2.4. El “ Jacobiano”

De las gradientes de retropropagación de la ecuaciones (74) y (75), la derivada pendiente en desarrollar es: $\frac{dy}{du}$, denominado el “jacobiano de la planta”.

Esta expresión $(\frac{dy}{du})$, no está disponible para calcularse en la planta “real”, por lo que podemos definirlo como desconocido.

Para solucionar este inconveniente, se introduce un emulador que “imite” las entradas y salidas de la planta “real”. Este emulador será el modelo desarrollado mediante redes neuronales dinámicas (Capítulo 2), en el cual se utilizó las entradas y salidas (patrones) de la planta “real” y la arquitectura mostrada en la figura 33.

A este modelo se le denominó NN emulador y es representado en la figura 44. El NN emulador, servirá para calcular el “jacobiano” del sistema, pues, de este NN emulador, se conocen todos los parámetros y todas las ecuaciones involucradas en su desarrollo.

El cálculo del “jacobiano” y su forma de uso que este tiene para la actualización de los pesos sinápticos del sintonizador neuronal, es otra modificación del método de retropropagación de errores estándar que se conoce.

Como sabemos, el NN emulador, tiene la arquitectura mostrada en la figura 33, el cual tiene cuatro neuronas en la capa de entrada (una de ellas es el bias), veinte neuronas en la capa oculta y dos neuronas en la capa de salida. La salida $X_1^*(k + 1)$, representará el valor estimado de la salida de la planta “real”. ($X_2^*(k + 1)$ será desestimada).

La salida $X_1^*(k + 1)$, puede ser escrita como:

$$X_1^*(k + 1) = f(u_{(k)}, x_{1(k)}^*, x_{2(k)}^*),$$

el cual la podemos reescribir como:

$$\hat{y}(n + 1) = f(u_{(n)}, y_{(n)}, y_{(n-1)}).$$

Usando el NN emulador, el “jacobiano” puede ser calculado de la siguiente manera, teniendo como guía la figura 45:

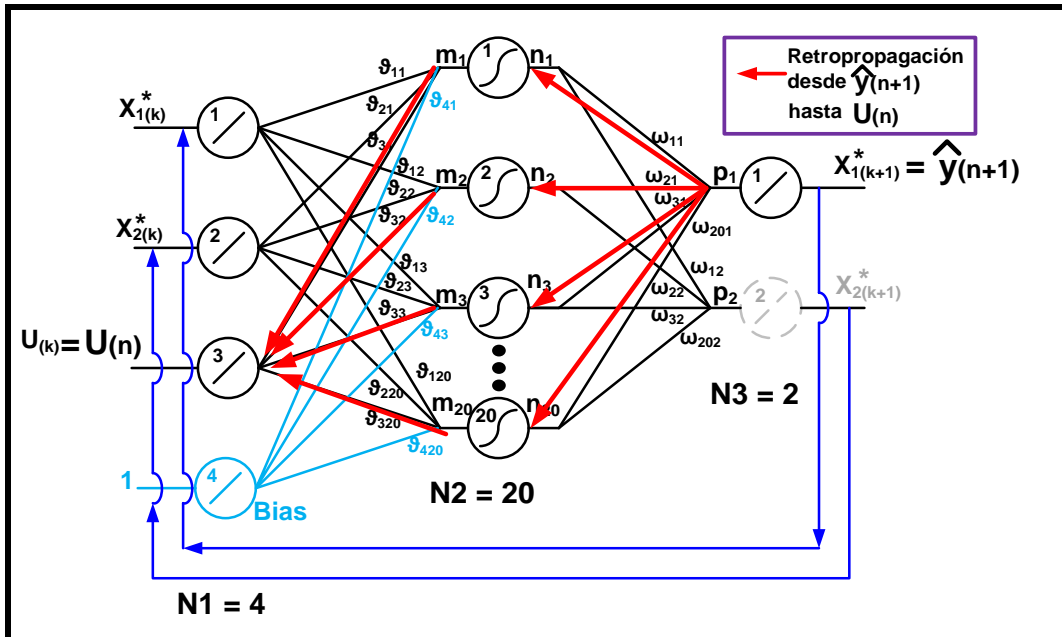


Figura 45: NN emulador. Retropropagación de la salida $\hat{y}(n+1)$ estimada hasta la entrada $U(n)$, para el cálculo del “jacobiano”.

Fuente: Elaboración propia.

$$\frac{d\hat{y}}{du} = \frac{d\hat{y}}{dp} \cdot \frac{dp}{dn} \cdot \frac{dn}{dm} \cdot \frac{dm}{du} \quad \dots \quad (86)$$

Desarrollando las derivadas de la ecuación (86) del NN emulador, una a una:

$$\frac{d\hat{y}}{dp} = 1 \quad \dots \quad (87)$$

La función de activación de la capa de salida es lineal.

$$\frac{dp}{dn} = W \quad \dots \quad (88)$$

$$\frac{dn}{dm} = \frac{df(m)}{dm} = \frac{(1-n) \cdot (1+n)}{2} \quad \dots \quad (89)$$

Que es la derivada de la función sigmoidea mostrada en la ecuación (76).

$$\frac{dm}{du} = V(\text{\#posición entrada de } u, :)^T \dots\dots\dots (90)$$

De la figura 18 se observa que la entrada de u está en la tercera ubicación, V es la expresión (37). Lo que la ecuación (90) se reescribe como:

$$\frac{dm}{du} = V(3, :)^T \dots\dots\dots (91)$$

V está basada en la expresión (37), que resulta de la arquitectura usada en el presente trabajo. El “jacobiano” de la arquitectura empleada en (16) resulta ser de dimensión (1x1).

Expresándolo de forma matricial, el “jacobiano” se calcula con la expresión indicada en (92):

$$\text{jacob} = \frac{d\hat{y}}{du} = V(3, :). \begin{bmatrix} \frac{dn_1}{dm_1} & 0 & \dots & 0 \\ 0 & \frac{dn_2}{dm_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{dn_{20}}{dm_{20}} \end{bmatrix} \cdot W \dots\dots\dots (92)$$

Hasta acá se involucra el NN emulador, cuyo modelo fue hallado mediante redes neuronales dinámicas.

Con el cálculo del “jacobiano” ya se tienen completadas todas las derivadas a utilizar para calcular las gradientes de retropropagación mostradas en las ecuaciones (74) y (75).

Actualización de pesos sintonizador neuronal.

Con el cálculo del jacobiano ya desarrollado, las gradientes de retropropagación mostradas en las ecuaciones (73) y (74), quedarían en forma matricial:

$$\delta_w = e .* \text{jacob} .* \begin{bmatrix} \frac{du}{dq_1} \\ \frac{du}{dq_2} \\ \frac{du}{dq_3} \end{bmatrix} \dots\dots\dots (93)$$

$$\delta_v = \frac{dN}{dM} .* (W * \delta_w) \dots\dots\dots (94)$$

En este caso $\frac{dN}{dM}$ se deriva de la ecuación (18), considerando la arquitectura del sintonizador neuronal graficado en la figura 3.21, $\frac{dN}{dM}$ queda como:

$$\frac{dN}{dM} = \begin{bmatrix} \frac{dn_1}{dm_1} \\ \frac{dn_2}{dm_2} \\ \vdots \\ \frac{dn_{12}}{dm_{12}} \end{bmatrix} \dots\dots\dots (95)$$

Y W con dimensión:

(12 x 3) → (#neuron. capa oculta x #neuron. capa salida)

Ahora, para la actualización de los pesos, necesitamos hallar $\frac{dJ}{dW}$ y $\frac{dJ}{dV}$, el cual como regla práctica se puede calcular como:

salida de la capa anterior * gradiente de retropropagacion,

Para $\frac{dJ}{dW}$ sería:

$$\frac{dJ}{dW} = N * (\delta_w)^T \dots\dots\dots (96)$$

En este caso N viene de la ecuación (5), considerando la arquitectura del sintonizador neuronal graficado en la figura 43, N queda como:

$$N = \begin{bmatrix} f_1(m_1) \\ f_2(m_2) \\ \vdots \\ f_{12}(m_{12}) \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_{12} \end{bmatrix} \dots\dots\dots (97)$$

Para $\frac{dJ}{dV}$ queda:

$$\frac{dJ}{dV} = X_{in} * (\delta_v)^T \dots\dots\dots (98)$$

Siendo:

$$X_{in} = \begin{bmatrix} u^{(n)} \\ u^{(n-1)} \\ y^{(n)} \\ y^{(n-1)} \\ u_c^{(n)} \\ u_c^{(n-1)} \end{bmatrix}.$$

Finalmente, la actualización de los pesos sinápticos W y V .

Para actualizar W de forma matricial, basándonos en la ecuación (72):

$$W(n + 1) = W(n) - \eta * \frac{dJ}{dW} + \alpha * \Delta W(n) \dots\dots\dots (99)$$

Del mismo modo ara actualizar V de forma matricial, basándonos en (73):

$$V(n + 1) = V(n) - \eta * \frac{dJ}{dV} + \alpha * \Delta V(n) \dots\dots\dots (100)$$

Siendo:

η , ratio de aprendizaje.

α , constante de momento.

Recordando que la actualización de W y V se realizará en línea.

La figura 46 muestra la arquitectura final del sistema con el controlador PID autosintonizado a utilizarse en el presente trabajo, mostrándose las escalas a utilizar y la posición que tienen el NN emulador en el diseño del controlador.

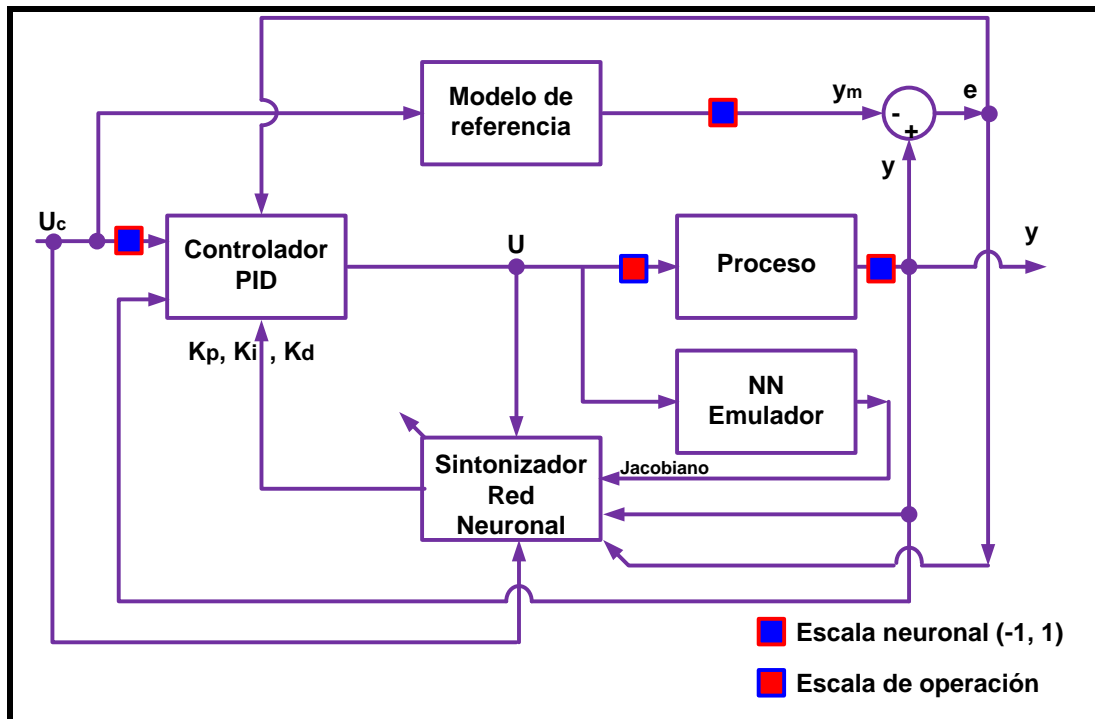


Figura 46: Arquitectura del sistema con controlador PID autosintonizado.

Fuente: Elaboración propia.

3.2.5. Implementación del controlador PID autosintonizado usando redes neuronales y control adaptativo en Simulink – Matlab.

A continuación presentaremos el desarrollo del control PID autosintonizado usando redes neuronales y control adaptativo, para lo cual utilizaremos el software Simulink – Matlab 2017.

Los bloques involucrados en el diseño del controlador serán presentados a continuación:

Bloque NN-emulador:

Es el bloque identificado con redes neuronales dinámicas (capítulo 2), los pesos sinápticos son hallados en el script presentado en el anexo 1, estos pesos son trasladados a Simulink. Este bloque se basa en la arquitectura mostrada en la figura 33. En la figura 47 se muestra el bloque NN emulador.

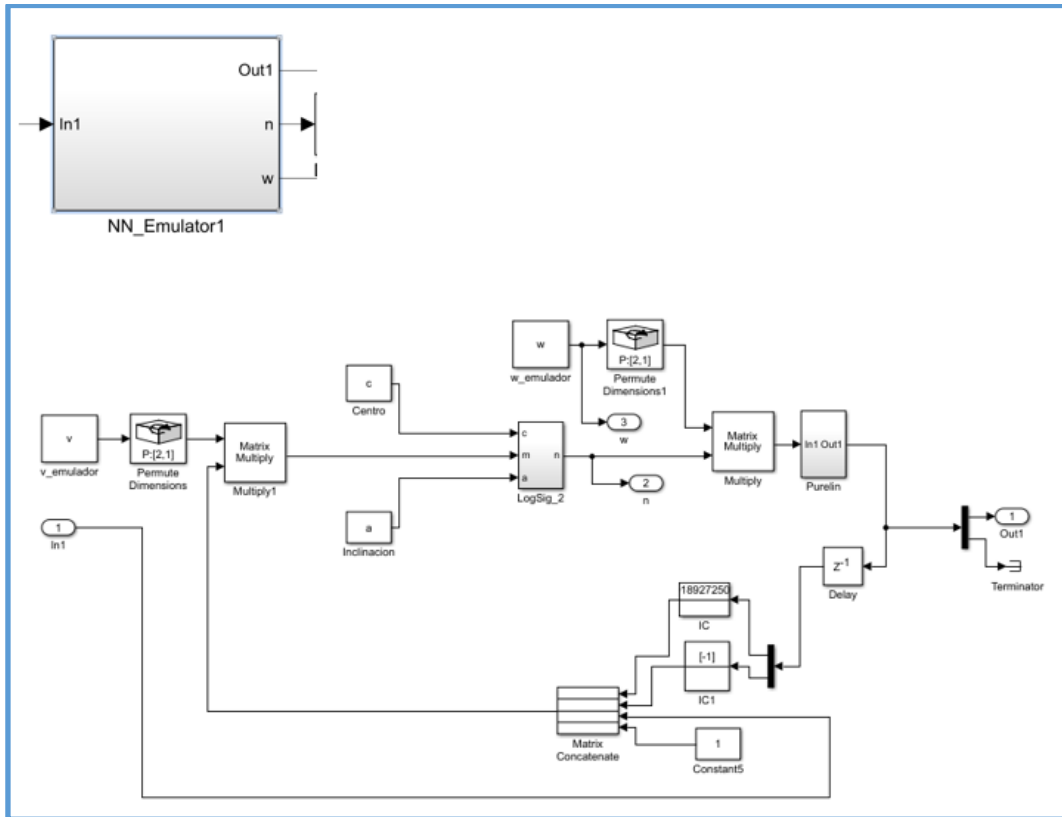


Figura 47: Bloque NN emulador

Fuente: Elaboración propia.

Bloque planta:

Este bloque representa la planta “real” a controlar indicados en las ecuaciones (63) y (64), basado en una planta de control de neutralización de pH. En la figura 48 se muestra el bloque planta.

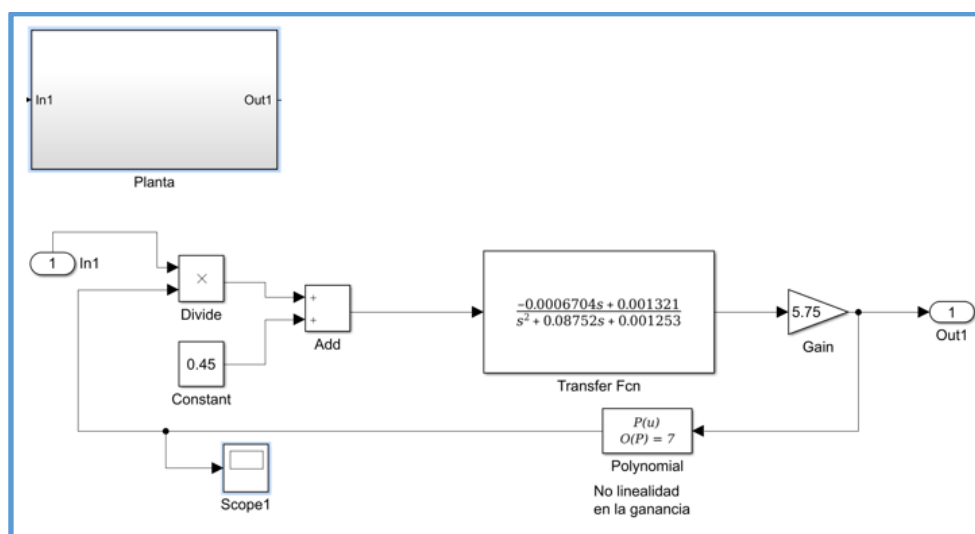


Figura 48: Bloque Planta, que representa al sistema a controlar.

Fuente: Elaboración propia.

Bloque PID:

Es el bloque que representa al controlador PID (de tipo velocidad), indicado en la ecuación (71), cuyos parámetros K_p , K_i y K_d son adaptativos y calculados por el sintonizador neuronal. El cálculo y la actualización de estos parámetros se realizan en línea. En la figura 49 se muestra el bloque PID.

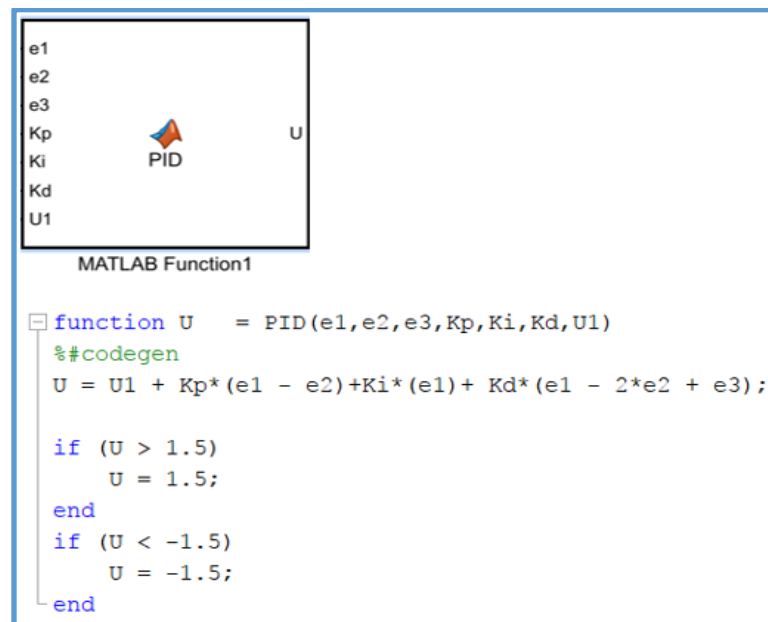


Figura 49: Bloque PID, K_p , K_i y K_d son calculados por el sintonizador neuronal.
Fuente: Propia.

Bloque modelo de referencia.

Es el bloque que representa al modelo de referencia, que es quien indica de cómo debe comportarse la señal de salida del proceso, ecuación (68). En este caso, es una función de transferencia de segundo orden, donde $\omega_n = 0.012$ y $\zeta = 0.9$. Estos valores pueden sufrir variaciones dependiendo de la respuesta que se desea. En la figura 50 se muestra el bloque de modelo de referencia.

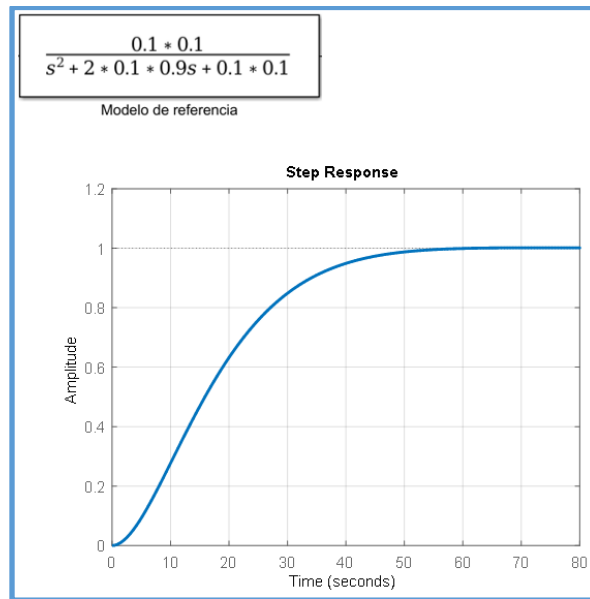


Figura 50: Bloque modelo de referencia.
Fuente: Elaboración propia.

Bloque “Jacobiano”

Es el bloque que desarrolla el cálculo del jacobiano resultante de la ecuación (92), cuyo resultado es entregado al sintonizador neuronal para la adaptación K_p , K_i y K_d que serán utilizados luego por el controlador PID. En la figura 51 se muestra el bloque jacobiano.

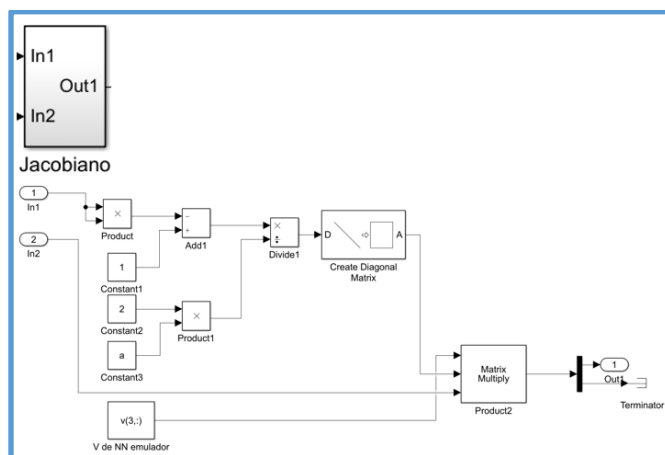


Figura 51: Bloque “jacobiano”.
Fuente: Elaboración propia.

Bloque $\frac{du}{dq}$

Bloque que desarrolla las derivadas de la salida del controlador PID, ‘u’ respecto a los parámetros del mismo K_p , K_i y K_d . Tal como se indica en la ecuaciones (79), (80) y (81). En la figura 52 se muestra el bloque du/dq .

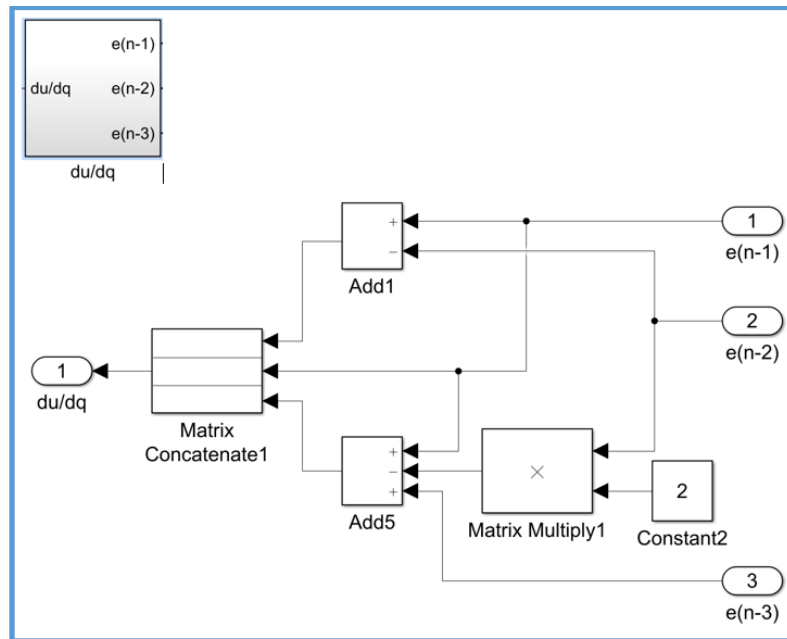


Figura 52: Bloque du/dq
Fuente: Elaboración propia.

Bloques de desarrollo del sintonizador neuronal:

En la figura 53 se muestra el desarrollo del sintonizador neuronal, basado en la arquitectura mostrada en la figura 43. Este desarrollo se basa en todas las ecuaciones involucradas que convergen en las ecuaciones (79) y (80). En la figura 53 se muestra el desarrollo del sintonizador neuronal.

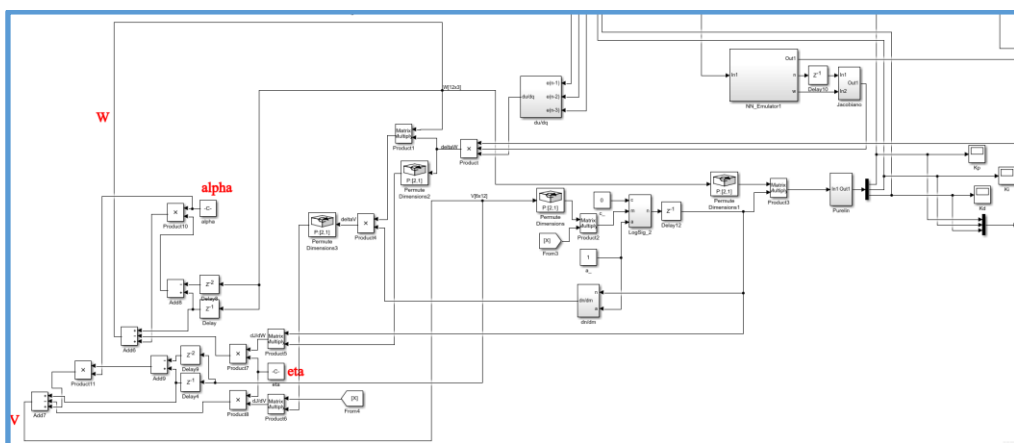


Figura 53: Desarrollo sintonizador neuronal.
Fuente: Elaboración propia.

En el anexo 2, se muestra el sistema completo del sistema y del controlador PID autosintonizado con redes neuronales y control adaptativo.

3.2.6. Resultados.

En la figura 54 se muestra la respuesta de la planta (pH) ante una señal set point variable y periódica (señal cuadrada). Se observa que durante el primer periodo, el sistema realiza el aprendizaje para la actualización de los pesos sinápticos para realizar el seguimiento de la señal set point, ya luego a partir del segundo periodo, la etapa de aprendizaje ha culminado y se observa que la señal pH hace un seguimiento bastante aceptable a la señal de set point.

Se considera:

$\eta = 0.0025$, ratio de aprendizaje.

$\alpha = 0.1$, constante de momento.

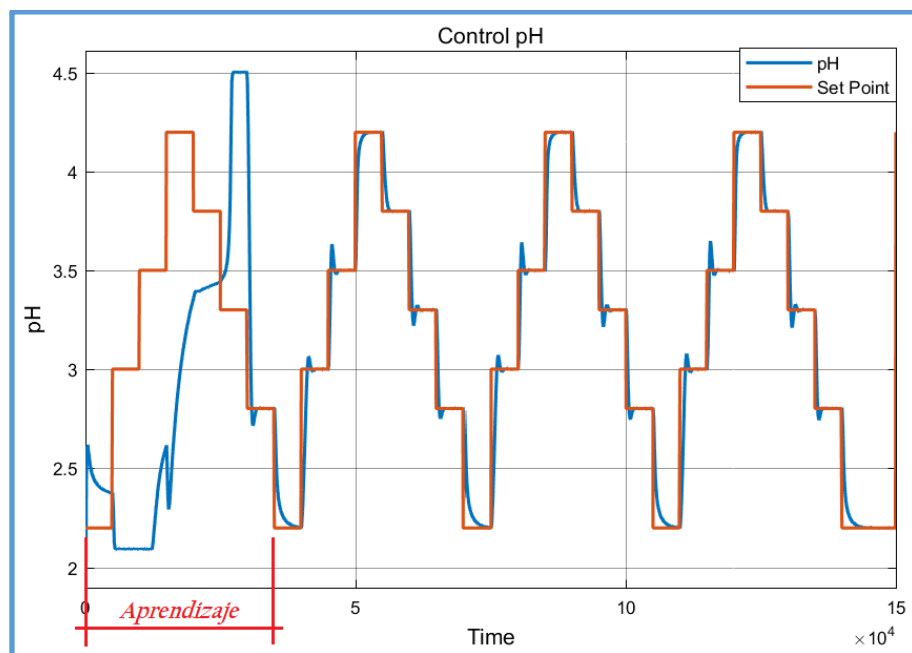


Figura 54: Salida de la planta pH vs. Set point

Fuente: Elaboración propia.

En la figura 55 de igual modo, se presenta una señal periódica, con frecuencia 1×10^{-6} Hz, al igual que el paso anterior, se observa la etapa de aprendizaje y posteriormente el sistema ya entrenado puede controlar de forma adecuada al sistema.

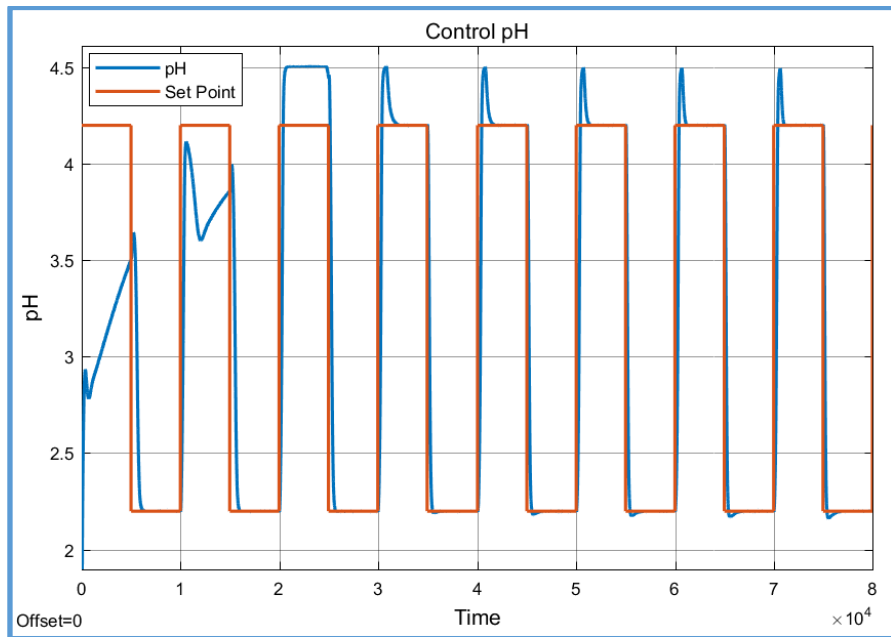


Figura 55: Salida de la planta pH vs. Set point periódico

Fuente: Elaboración propia.

En la figura 56, se presentan las respuestas del sistema a distintas entradas tipo escalón, observándose además el ruido presente en el sistema. Pudiéndose el correcto seguimiento del sistema a las señales de set point.

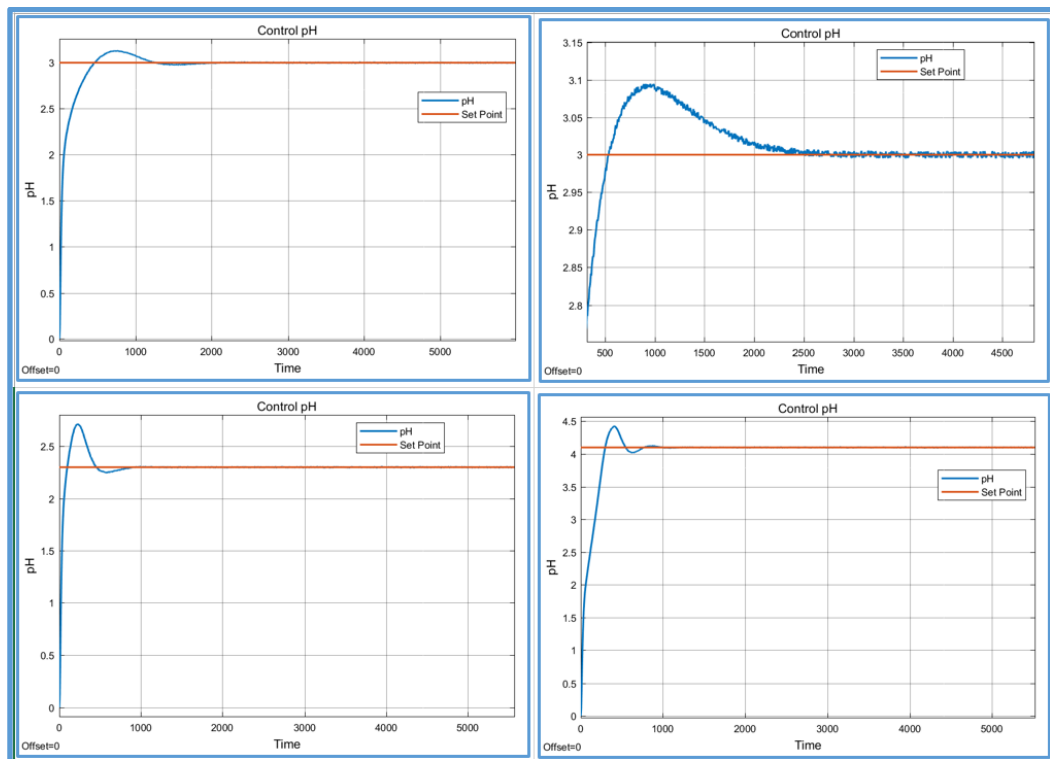


Figura 56: Respuesta del sistema a distintas señales tipo escalón.

Fuente: Elaboración propia.

3.2.7. Efectos de los valores iniciales de los pesos sinápticos.

En el controlador diseñado, se considera la inicialización de los pesos sinápticos del sintonizador neuronal de forma aleatoria, estos datos iniciales determinarán de forma directa la velocidad de aprendizaje del controlador PID autosintonizado y esto a su vez la convergencia del mismo. Si los valores iniciales de los pesos sinápticos no son adecuados, el sistema demorará en aprender y por lo tanto a converger. En la figura 57 se muestra este efecto a la misma señal de set point mostrada en la figura 54. En esta se puede apreciar la demora del aprendizaje del sistema, no permitiendo un seguimiento adecuado del valor controlado durante tres periodos.

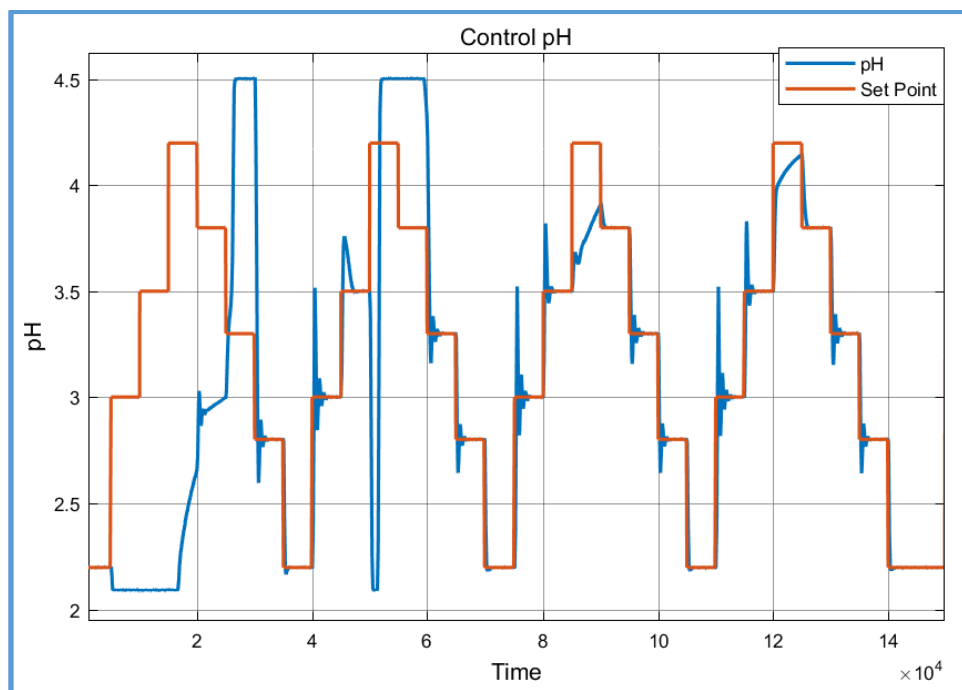


Figura 57: Efecto de elección de valores iniciales de los pesos sinápticos

Fuente: Elaboración propia.

3.2.8. Efecto del modelo de referencia.

El modelo de referencia como se indicó en un inicio, determina el comportamiento de la respuesta del sistema. Para la planta en estudio en cuestión, esta determinará la velocidad de respuesta del sistema. Es decir si el modelo de referencia tiene un tiempo de establecimiento alto el controlador también demorará en converger, lo mismo si sucede de manera contraria.

En las figuras 58 y 59 se muestran los efectos del modelo de referencia respecto a sus tiempos de establecimiento. Resaltando las diferencias del tiempo de establecimiento entre ellas.

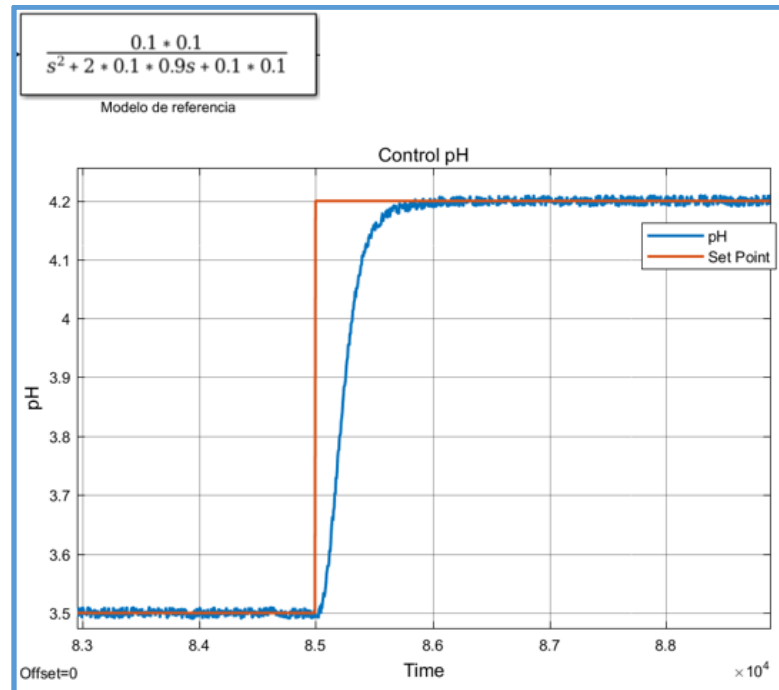


Figura 58: Respuesta ante el modelo de referencia con un tiempo de establecimiento de 500 seg.
Fuente: Elaboración propia.

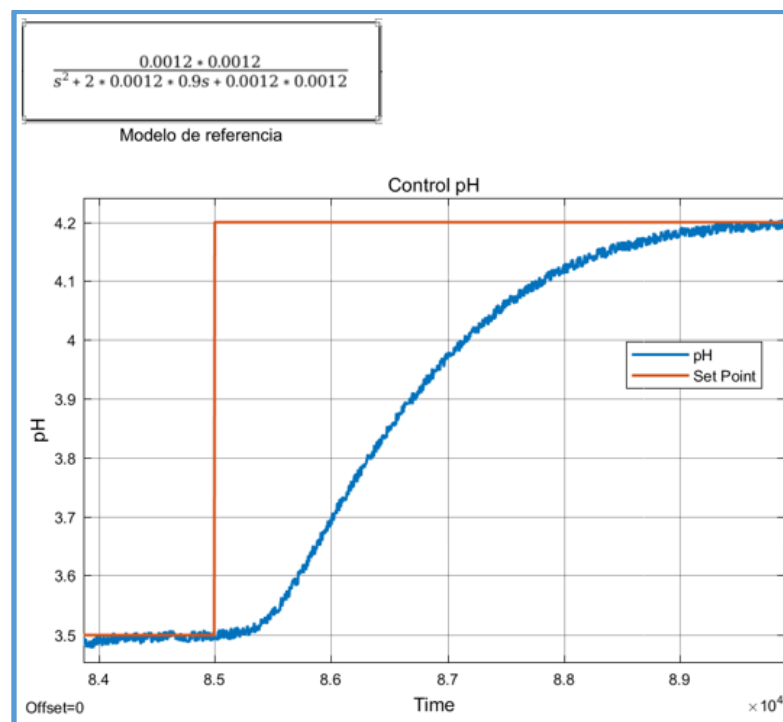


Figura 59: Respuesta ante el modelo de referencia con un tiempo de establecimiento de 4000 seg.
Fuente: Elaboración propia.

CAPÍTULO IV: ANÁLISIS DE ROBUSTEZ Y PROPUESTA DE IMPLEMENTACION DEL CONTROLADOR DISEÑADO

4.1. Controlador PID autosintonizado con redes neuronales y controla adaptativo frente a una perturbación.

En la figura 60 se muestra la perturbación generada a la salida del sistema, consta de una amplitud de 0.2, que representa el 10% del rango de operación del sistema, esta perturbación puede ser la variación del flujo de la solución ácida o algún cambio imprevisto externo al sistema, se observa como después de presentar dicha perturbación el controlador hace que el sistema se estabilice a la consigna indicada (set point), esto, en un tiempo aproximado de 400 segundos, que es el tiempo inherente al sistema (dinámica del sistema).

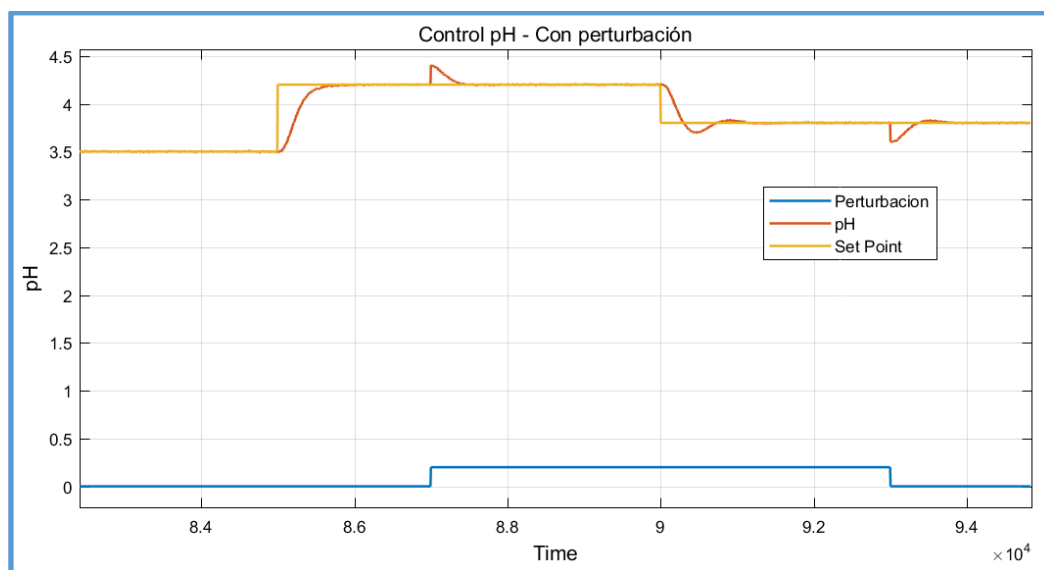


Figura 60: Respuesta del sistema ante una perturbación.
Fuente: Elaboración propia.

4.2. Comparación del controlador PID autosintonizado con redes neuronales y control adaptativo versus un controlador PID clásico.

A continuación se presenta la comparación entre un PID clásico y el controlador PID autosintonizado diseñado. El controlador PID convencional será el de tipo velocidad. Se presentará como señal de consigna (set point) la señal periódica presentada en la figura

62, la misma, se presenta al sistema controlado con el PID autosintonizado. Luego, las señales resultantes serán comparadas, teniendo como indicador de proximidad entre ambos sistemas la integral del error absoluto (IAE) y la integral del tiempo por error absoluto (ITAE).

Antes describiremos brevemente los conceptos de IAE y del ITAE.

La Integral del error absoluto (IAE), y la integral del tiempo por el error absoluto (ITAE) son índices de desempeño basado en la señal del error (e) presente en un sistema, la cual, si hablamos de sistemas de control, el error (e) se define como la diferencia entre el valor de consigna y la salida del sistema (valor real). Así:

La integral del error absoluto (IAE) se define como:

$$IAE = \int_0^{Tf} |e(t)|dt \dots\dots\dots(101)$$

Y,

La integral del tiempo por el error absoluto (ITAE), se define como:

$$ITAE = \int_0^{Tf} t|e(t)|dt \dots\dots\dots(102)$$

Donde:

Tf, el tiempo hasta donde se desea evaluar ambas señales.

e, el error medido definido como (sp – salida real).

La figura 61 muestra el sistema a ser controlado con un PID convencional y en la figura 62 se muestra la respuesta del sistema a dicho control frente a la consigna (set point) y perturbación mostrada anteriormente.

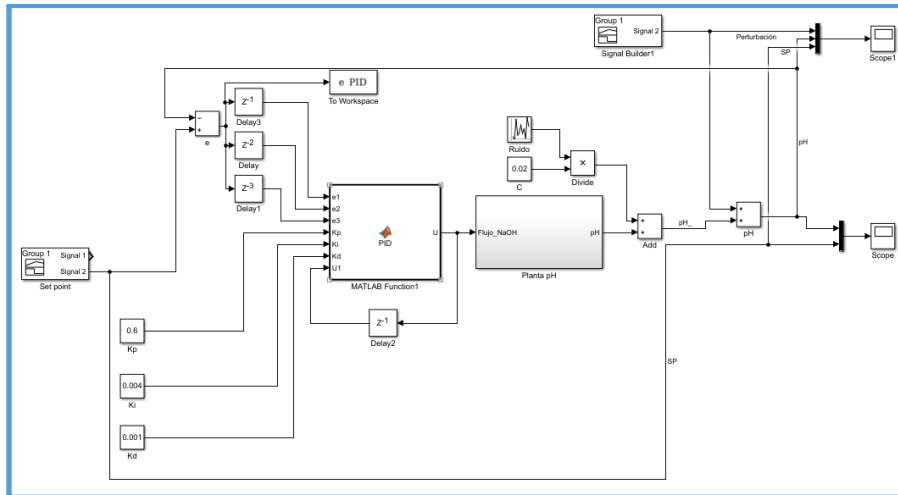


Figura 61: Sistema pH controlado con PID convencional
Fuente: Elaboración propia.

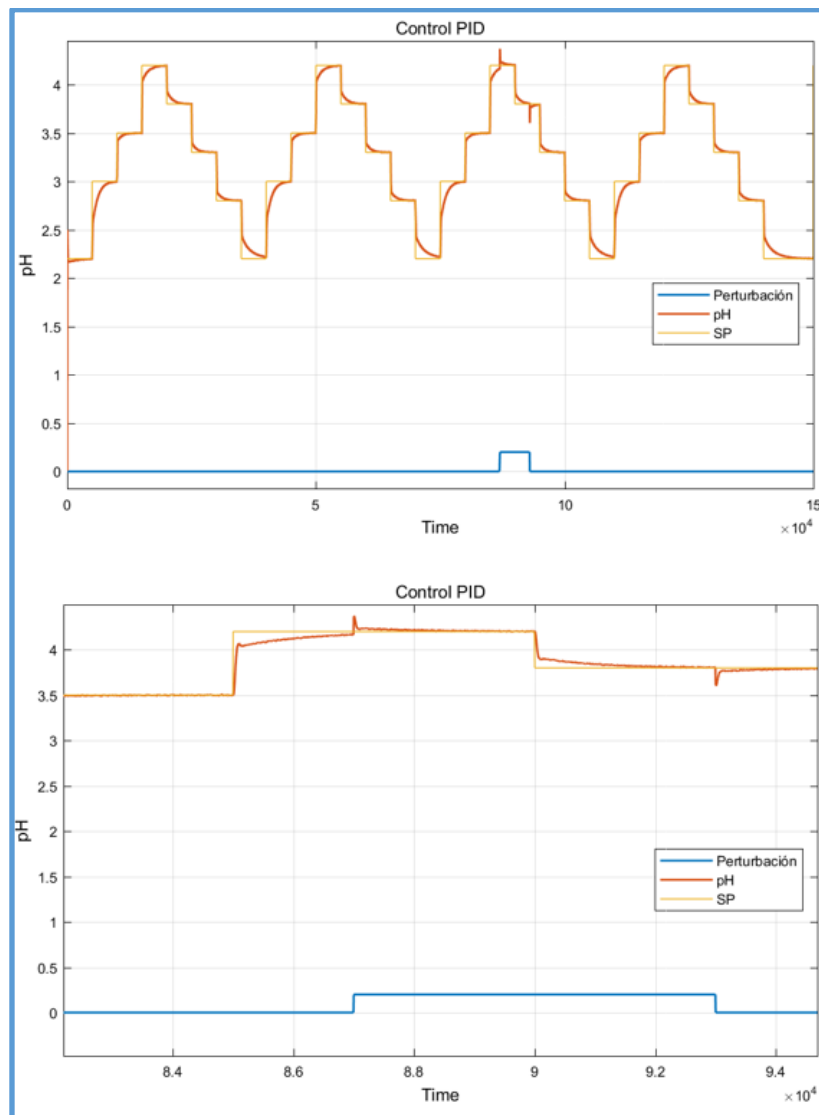


Figura 62: Respuesta del sistema a consignas variables y presencia de perturbación.
Fuente: Elaboración propia.

La tabla 3 muestra los valores de ITAE e IAE para el sistema desarrollado con controlador PID neuronal, y para el sistema controlado con un PID convencional, observándose un menor valor para el sistema controlado con un PID con redes neuronales y control adaptativo.

Lo que significa que el PID autosintonizado con redes neuronales y control adaptativo presenta menos error que el sistema controlado con un PID convencional.

Tabla 3: Indicadores ITAE e IAE de los sistemas con un PID autosintonizado con red neuronal y un PID convencional.

ITAE PID NN ADAPTATIVO	ITAE PID
4.36e+08	4.93e+08
IAE PID NN ADAPTATIVO	IAE PID
6443.9817	6877.7112

Fuente: Elaboración propia.

4.3. Propuesta de implementación práctica del controlador diseñado

Como se indicó en el capítulo 3, la planta se basa en el control de neutralización de pH desarrollado en Hau (2009).

En la cual desarrolla un control predictivo para la neutralización de pH.

Con la obtención de nuestra planta desarrollada en el capítulo 3, observamos que presentan grandes similitudes, por lo que podemos determinar que la instrumentación que se utiliza en Hau (2009), es también válida para nuestra planta cuestión de estudio.

En la figura 63 se lista la instrumentación y equipamiento que se sugiere para el control de pH de la planta cuestión de estudio.







REACTOR	BOMBA PROPORCIONAL SOLUC. ÁCIDA
	
Capacidad 1.8 litros, mezcla homogénea	Marca Walchem. Mod.:EZB30D2VC Flujo de operación: 0.4 l/min (constante)
BOMBA PROPORCIONAL SOLUC. BASE	AGITADOR MAGNÉTICO
	
Marca Walchem. Mod.:EZB30D2VC Actuador del sistema 4-20 mA.	Marca VELP. Mod.: AGE. 40Watts Para agitación del reactor.
ELECTRODO pH	TRANSMISOR INDICADOR pH
	
Marca: Hanna. Mod.: 1210B Transductor pH - volt	Marca: Hanna. Mod.: HI8510 Rango: 0-14, resol. 0.001. Out: 4-20mA

Figura 63: Instrumentación sugerida para el control del sistema.

Fuente: Elaboración propia.

4.4. Diagrama P&ID para el control del sistema

En la figura 64 se muestra el diagrama P&ID considerado para el control del sistema, se observa que el flujo de solución ácida en algún momento puede ser considerado como una perturbación.

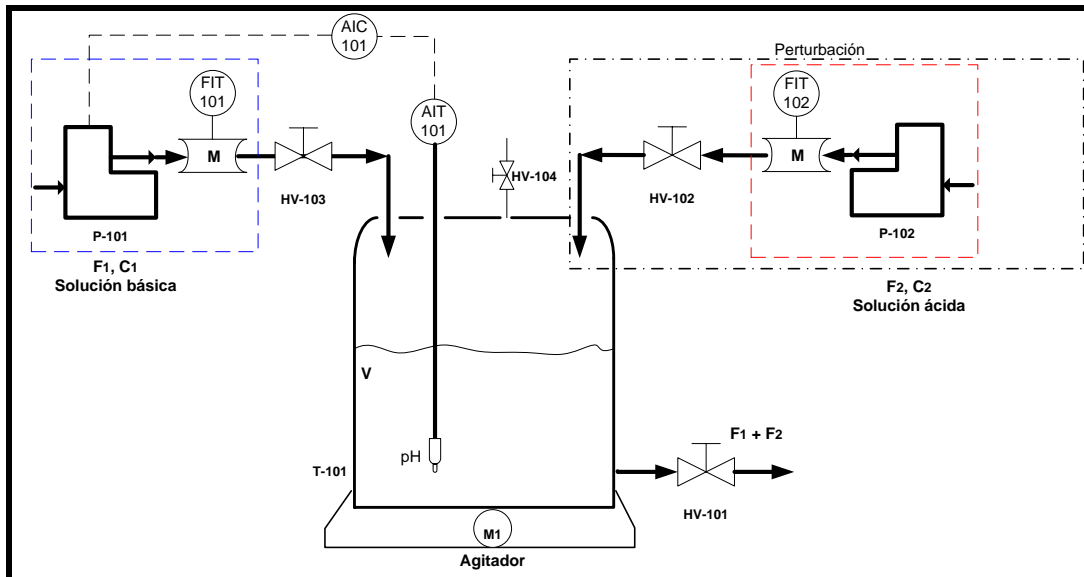


Figura 64: Diagrama P&ID para el control del sistema.

Fuente: Elaboración propia.

4.5. Sistema de control y arquitectura de control.

En la tabla 4 se muestra el listado del sistema de control a considerar para el control del sistema cuestión de estudio.

Tabla 4: Listado de los componentes del sistema de control sugerido (continúa).

SISTEMA CONTROLLOGIX – ROCKWELL AUTOMATION		
1756-L72	1	ControlLogix 5570 Controller with 4 MB Memory
1756-IA16I	1	79-132 VAC Isolated Input 16 Pts (36 Pin)
1756-OA16I	1	74-265 VAC Isolated Output 16 Pts (36 Pin)
1756-IF8IH	1	Isolated Analog Input Module - 8-Channel HART
1756-OF8IH	1	Isolated Analog Output Module - 8-Channel HART
1756-EN2T	1	EtherNet 10-100M Interface Module (TCP/IP connections)
1756-A7	1	7 Slot ControlLogix Chassis

Tabla 4: Listado de los componentes del sistema de control sugerido.

SISTEMA CONTROLLOGIX – ROCKWELL AUTOMATION		
1756-PA72	1	ControlLogix, 85-265 VAC Power Supply (10 Amp @ 5V)
1756-TBCH	1	36 Pin Screw Clamp Block With Standard Housing
1756-TBS6H	1	36 Pin Spring Clamp Block With Standard Housing
1585J-M8TBJM-1M9	1	Male RJ45 to Male RJ45 Patchcord, Unshielded Twisted Pair, Teal TPE Cable, 1.9 m

Fuente: Elaboración propia.

La arquitectura a considerar se muestra en la figura número 65, donde se indica el uso del servidor OPC, el cual servirá como enlace entre el PAC de Rockwell y Matlab el cual contiene el algoritmo de control. Con esto se tiene que el PAC se utilizará como una fuente de adquisición de datos.

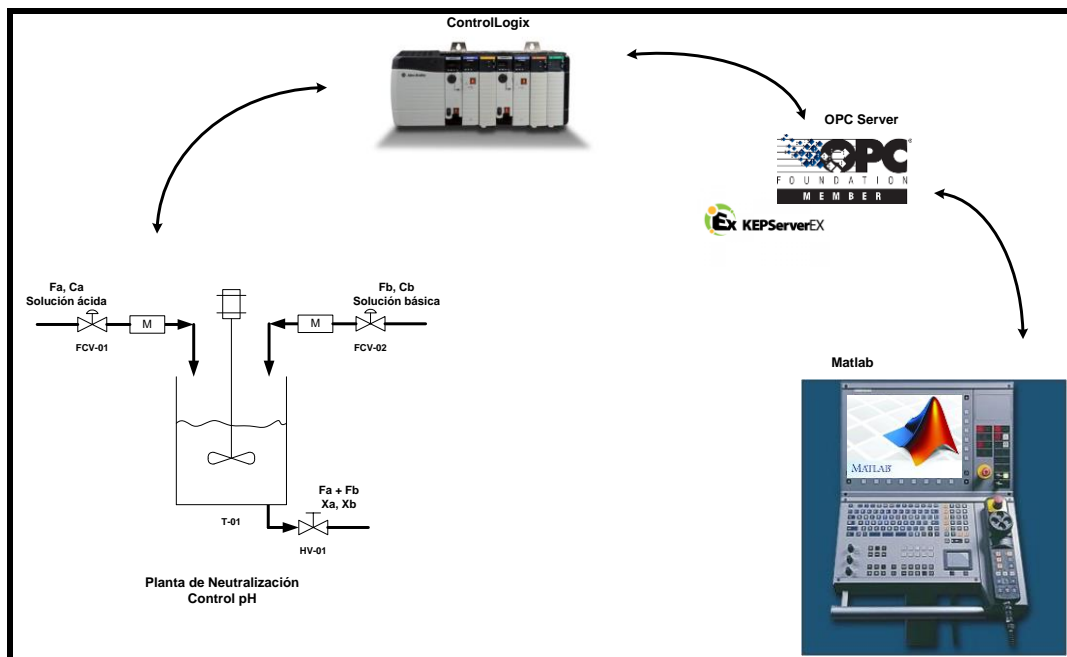


Figura 65: Arquitectura para el control de la planta de neutralización de pH

Fuente: Elaboración propia.

CONCLUSIONES

1. Se diseñó el controlador con auto sintonía PID basado en un modelo de redes neuronales dinámicas y control adaptativo, el cual también involucró el uso de teoría de redes neuronales estáticas, y por los resultados obtenidos, se concluye que es una alternativa viable para el control de plantas que presentan no linealidades o dinámicas complejas en su estructura.
2. Se identificó el modelo de una planta de segundo orden con discontinuidades en su ganancia; haciendo uso de redes neuronales dinámicas para conocer el comportamiento dinámico del sistema. Por los resultados obtenidos e índices utilizados (VAF) este método resulta óptimo para el desarrollo de sistemas no lineales.
3. En el desarrollo del controlador PID auto sintonizado con redes neuronales y control adaptativo, se desarrolló la teoría de este tipo de control (adaptativo) basado en modelo (modelo neuronal) para realizar la sintonía automática de los parámetros K_p , K_i y K_d del controlador PID. Se concluye que el control adaptativo, ayuda a mejorar la respuesta del sistema ante una consigna presentada, es decir en ella se define el comportamiento dinámico que debe tener la salida del sistema, como es el tiempo de establecimiento, sobre pico, amortiguamiento, etc.
4. En la implementación del controlador PID auto sintonizado con redes y control adaptativo, los algoritmos aplicados en una planta simulada de segundo orden con discontinuidades en su ganancia se pudo observar las mejoras obtenidas con este algoritmo en comparación con un PID clásico. Esto porque el controlador implementado incorpora funciones no lineales en su optimización, pudiendo operar en un rango bastante amplio de trabajo, y no limitarse en rangos lineales como lo hace el control PID convencional.
5. Se concluye y demuestra que las redes neuronales dinámicas, ofrecen mejor respuesta en la identificación de sistemas ruidosos, pues estas incorporan el error existente dentro de la etapa de aprendizaje de la red. Con el inconveniente que su convergencia es más lenta que una red neuronal estática.

6. Las redes neuronales estáticas, usadas en la identificación de sistemas, “siguen” de forma fiel a la señal original, tanto, que si la señal original presenta ruido o algunas variaciones externas, la red tomará dichas señales para su entrenamiento, incorporando estas señales en el modelo final pudiendo ocasionar una convergencia lenta o la no convergencia del sistema. Pero, la ventaja de la red neuronal estática, frente a la dinámica es que converge de una forma más rápida, siendo recomendados, por este motivo, para el control en línea de plantas industriales, denominándose “neurocontroladores”.
7. En el desarrollo del bloque sintonizador neuronal, una buena elección del valor del ratio de aprendizaje (η), de la constante de momento (α) y de los pesos iniciales de los pesos sinápticos (V, W) determinará una convergencia rápida entregando parámetros óptimos muy aceptables. Lo mismo sucede con la red neuronal dinámica. Por lo que se concluye que son estos parámetros determinantes en el aprendizaje de la red neuronal.
8. De los dos primeros puntos, se define: la red neuronal dinámica fue usada para la identificación del sistema que es considerada no lineal, y la red neuronal estática fue usada para sintonizar los parámetros Kp, Ki y Kd en línea.
9. En el sistema estudiado, para hallar los parámetros PID del controlador en línea, se necesita del modelo de la planta, ya que estos deben alcanzar sus objetivos bajo cálculos matemáticos, los cuales son definidos en el modelo identificado (NN emulador), por lo tanto se concluye que toda planta identificada con redes neuronales u otro método ayuda a desarrollar distintos algoritmos de optimización para alcanzar objetivos de control, en esta tesis se desarrolló la autosintonización de los parámetros Kp, Ki y Kd de un controlador PID.
10. Dentro del desarrollo del controlador, y basados en todas las ecuaciones y funciones del sistema modelado (NN emulador), un desarrollo novedoso es el de calcular el “jacobiano” del modelo hallado (NN emulador), el cual ayuda a determinar parámetros óptimos de control a usarse con distintos algoritmos (control predictivo, control fuzzy, etc.)

11. En plantas que presentan no linealidades, una alternativa real y óptima es el uso de controladores avanzados, los cuales usan algoritmos de optimización bastante exactos, con el desarrollo de la presente tesis, se concluye que estos controladores avanzados tienen una mayor eficiencia que un control PID convencional y abarcan un mayor rango de operación. Tal como se demuestra con los índices calculados en el capítulo 4 (ITAE, IAE).
12. Se concluye que los controladores avanzados, como el desarrollado en el presente trabajo, ofrecen mejor eficiencia a perturbaciones externas, tal como se pudo demostrar en el capítulo 4. En algunos casos, inclusive también se modela la perturbación para poder considerar esta señal dentro del controlador, para así tener una mejor respuesta frente a señales no deseadas.
13. Con las simulaciones realizadas, se puede concluir que el dimensionamiento y el planteamiento para la implementación práctica del control de pH en la planta de neutralización estudiada, cumple con las exigencias que se requiere para el control efectivo del sistema.

RECOMENDACIONES

1. El uso de este tipo de controladores avanzados, como es el desarrollado en el presente trabajo, involucra un conocimiento de la dinámica de la planta, pues sin esta información será imposible llegar a los objetivos de control asignados, por lo que se recomienda tener una base sólida en la teoría de la identificación de sistemas y los modelos que este utiliza (ARX, ARMAX, CARIMA, etc.), pues los controladores avanzados se basan mucho en estos modelos.
2. Pese a que el controlador desarrollado en el presente trabajo responde bien ante señales de perturbación, es recomendable el modelamiento del mismo, pues permite aumentar la robustez del controlador, ya que este es capaz de identificar la perturbación y anticiparse a este para realizar un control más eficiente.
3. En este caso en el que la planta estudiada no presenta alta no linealidad, se puede realizar un control bastante aceptable con un controlador PID clásico (aun así, el controlador PID autosintonizado muestra mayor rapidez en hallar parámetros K_p , K_i y K_d óptimos) pero dentro de un rango determinado de operación que es menor al presentado. Pero en plantas altamente no lineales (por ejemplo en todo el rango de operación del pH en una planta de neutralización 0- 14), el controlador PID convencional no lograría hacer un control aceptable, es ahí donde la performance del controlador avanzado diseñado mostrará una mayor eficiencia. Por lo que se recomienda continuar con la investigación y desarrollo de este controlador PID autosintonizado en plantas altamente no lineales.
4. En la propuesta de implementación práctica, se muestra el uso de un servidor OPC que sirve de enlace entre el PAC y el controlador (Matlab), pero, ahora, con el desarrollo tecnológico que tiene hoy en día, el costo computacional ya no es un tema crítico, los controladores presentan mejores prestaciones; por lo que se recomienda la investigación y desarrollo del algoritmo desarrollado dentro de un sistema embebido, como por ejemplo dentro de un PLC.
5. Asimismo, volviendo al desarrollo tecnológico de hoy en día, es más común escuchar del desarrollo de redes neuronales, control fuzzy, algoritmos genéticos,

etc., dentro de controladores avanzados (por ejemplo el predictivo), por lo que se recomienda y se conmina a esto a los estudiantes, a la investigación de este campo, pues el desarrollo de la inteligencia artificial está tomando un envión tremendo en estos tiempos, y todo ingeniero, debe estar preparado para esto, a enfrentar el futuro.

REFERENCIAS BIBLIOGRÁFICAS

- Acosta M. y Zuloaga C. (2000). Tutorial sobre redes neuronales aplicadas en ingeniería eléctrica y su implementación en un sitio web. Universidad tecnológica de Pereira. Facultad de ingeniería eléctrica. Pereira. Colombia.
- Åström, K. J., & Wittenmark, B. (1989). Adaptive control. Reading, MA: Addison-Wesley.
- Babuška, R., Setnes, M., Mollov, S. & Van der Veen, P. (2001). Fuzzy modeling and identification toolbox: for use with matlab. mathworks. The Netherlands.
- Brío, B. M., & Molina, A. S. (2001). Redes neuronales y sistemas borrosos. Madrid: RA-MA.
- Cardenas, A. M., Razuri, J. G., Sundgren, D., & Rahmani, R. (2013). Autonomous Motion of Mobile Robot Using Fuzzy-Neural Networks. 2013 12th Mexican International Conference on Artificial Intelligence. doi:10.1109/micai.2013.15
- Castillo, P., Castellano, J., Merelo, J., & Prieto, A. (2007). Diseño de Redes Neuronales Artificiales mediante Algoritmos Evolutivos. *Inteligencia Artificial*, 5(14). doi:10.4114/ia.v5i14.758
- Gomm, J. (1996). Control of pH in-line using a neural predictive strategy. UKACC International Conference on Control. *Control* 96. doi:10.1049/cp:19960699
- Gupta, M., Rao, D., & Wood, H. (1994). Learning and adaptive neural controller. [Proceedings] 1991 IEEE International Joint Conference on Neural Networks. doi:10.1109/ijcnn.1991.170744
- Hadjiski, M., Boshnakov, K., & Galibova, M. (2002). Neural networks based control of pH neutralization plant. Proceedings First International IEEE Symposium Intelligent Systems. doi:10.1109/is.2002.1042565

- Haykin, S. (1998). *Neural networks: A comprehensive foundation*. New York, NY: Macmillan.
- Hau Yon, Jorge. (2009). *Control predictivo no lineal basado en modelación Hammerstein polinomial aplicado a un módulo de pH*. Universidad de Piura. Piura. Perú.
- Khan, A., & Rapal, N. (2006). *Fuzzy PID Controller: Design, Tuning and Comparison with Conventional PID Controller*. 2006 IEEE International Conference on Engineering of Intelligent Systems. doi:10.1109/iceis.2006.1703213
- Lee, K. (2006). *System-Type Neural Network Architectures for Power Systems*. The 2006 IEEE International Joint Conference on Neural Network Proceedings. doi:10.1109/ijcnn.2006.1716313
- Moran, Antonio. (2012). *Systems Modeling and Control Using Dynamic Neural Networks*. Conference: 15th Latin American Control Conference. Lima. Perú.
- Nelson, S. C., & Yolanda, A. G. (2006). *Redes neuronales: Conceptos fundamentales y aplicaciones a control automático*. Madrid: Pearson/Prentice-Hall.
- Omatu, S., Yos, M., & Kosaka, T. (2009). *PID Control of Speed and Torque of Electric Vehicle*. 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences. doi:10.1109/advcomp.2009.31
- Pham, D. T., & Liu, X. (1995). *Neural networks for identification, prediction, and control*. London: Springer-Verlag.
- Pirabakaran, K., & Becerra, V. (2002). *Pid Autotuning Using Neural Networks and Model Reference Adaptive Control*. IFAC Proceedings Volumes, 35(1), 451-456. doi:10.3182/20020721-6-es-1901.00728

- Sung, S., Lee J. & Lee I. (2009) Proportional-Integral-Derivative Controller Tuning. Process Identification and PID Control, 151-199. doi:10.1002/9780470824122.ch5
- Volna, E. (2015). Multilayer Neural Networks & Backpropagation Rules. Backpropagation Neural Networks and Their Applications, 239-1284. doi:10.4018/978-1-4666-9708-9.les1
- Wang, J., Zhang, C., Jing, Y., & An, D. (2007). Study of Neural Network PID Control in Variable-frequency Air-conditioning System. 2007 IEEE International Conference on Control and Automation. doi:10.1109/icca.2007.4376371
- Yang, Y., & Wu, Q. (2016). A neural network PID control for PH neutralization process. 2016 35th Chinese Control Conference (CCC). doi:10.1109/chicc.2016.7553893
- Yeh, Z. (1994). An adaptive neural net controller design. Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN94). doi:10.1109/icnn.1994.374628

Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas (continúa).

```

dJda_t = zeros(nm,1);
x = dataoutesc(1,:); % Solo al principio como estado inicial
x = x';
% Inicio de presentación de los patrones de entrada - salida
for k = 1:ndata-1
    in_red = [ x
              u(k,1)
              1]; % % % Bias
    m = v'*in_red;
    n = 2.0./(1 + exp(-(m-c)./a)) - 1;
%   n = m; % % % Lineal
    out_red = w'*n;
    outpesc(k,:) = out_red';
    dndm = diag((1 - n.*n)./(2*a));
%   dndm = diag(ones(nm,1)); % Lineal.

% Cálculo de derivadas simples
dy1dw_s = [ n zeros(nm,1) ];
dy2dw_s = [ zeros(nm,1) n ];
dy1dv_s = in_red*w(:,1)*dndm;
dy2dv_s = in_red*w(:,2)*dndm;
dy1dc_s = w(:,1) .* ((n.*n-1)./(2.0.*a));
dy2dc_s = w(:,2) .* ((n.*n-1)./(2.0.*a));
dy1da_s = w(:,1) .* ((n.*n-1).*(m-c)./(2*a.*a));
dy2da_s = w(:,2) .* ((n.*n-1).*(m-c)./(2*a.*a));

% Jacobiano
jacob = w'*dndm*(v(1:ne-1,:))';

% Cálculo de derivadas totales
dy1dw_t = dy1dw_s + jacob(1,1).*dy1dw_t + jacob(1,2).*dy2dw_t;
dy2dw_t = dy2dw_s + jacob(2,1).*dy1dw_t + jacob(2,2).*dy2dw_t;
dy1dv_t = dy1dv_s + jacob(1,1).*dy1dv_t + jacob(1,2).*dy2dv_t;
dy2dv_t = dy2dv_s + jacob(2,1).*dy1dv_t + jacob(2,2).*dy2dv_t;
dy1dc_t = dy1dc_s + jacob(1,1).*dy1dc_t + jacob(1,2).*dy2dc_t;
dy2dc_t = dy2dc_s + jacob(2,1).*dy1dc_t + jacob(2,2).*dy2dc_t;
dy1da_t = dy1da_s + jacob(1,1).*dy1da_t + jacob(1,2).*dy2da_t;
dy2da_t = dy2da_s + jacob(2,1).*dy1da_t + jacob(2,2).*dy2da_t;
out_des = dataoutesc(k+1,:);
out_des = out_des';
er = (out_red - out_des);
erJ = (out_red - out_des).^1;
error(k,1) = er(1,1);
%   q1 = 1; q2 = 0; % Solo se mide la primera variable
q1 = 1; q2 = 1; % Se miden las dos variables
dJdw_t = dJdw_t + q1*erJ(1,1).*dy1dw_t + q2*erJ(2,1).*dy2dw_t;
dJdv_t = dJdv_t + q1*erJ(1,1).*dy1dv_t + q2*erJ(2,1).*dy2dv_t;
dJdc_t = dJdc_t + q1*erJ(1,1).*dy1dc_t + q2*erJ(2,1).*dy2dc_t;
dJda_t = dJda_t + q1*erJ(1,1).*dy1da_t + q2*erJ(2,1).*dy2da_t;

```

Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas (continúa).

```

    ersum2 = ersum2 + er.^2;
    x = out_red; % Notar que la salida se convierte en entrada
end
dJdw_t = dJdw_t/ndata;
dJdv_t = dJdv_t/ndata;
dJdc_t = dJdc_t/ndata;
dJda_t = dJda_t/ndata;
dw = dJdw_t;
dv = dJdv_t;
dc = dJdc_t;
da = dJda_t;

alpha = 0.75; %%%%%%%%%%%Factor momento

w1=w;
w = w - eta*dw + alpha*(w-w2);
w2 = w1;

v1 = v;
v = v - eta*dv + alpha*(v-v2);
v2 = v1;

c = c - etac*dc;
a = a - etaa*da;
dw_old = dw;
dv_old = dv;
ersum2total = sum(ersum2);
cont = cont + 1;

rmse(cont,1) = norm(error)/sqrt(ndata-1);
rmse(cont,1)
end
nt = length(u);
dt = 5;
tt = 0:1:(nt-1);
tt = dt*tt';
tt1 = tt(1:nt-1,1);

figure(1);
plot(tt,dataoutesc(:,1),'-r','LineWidth',1.5);
hold on;
plot(tt1,outputesc(:,1),'-b','LineWidth',1.5);
title('Salida pH Planta Vs Salida pH Neuronal');
xlabel('Time [sec]');
ylabel('pH');
legend('Salida pH planta', 'Salida pH neuronal');
axis([ 0 20000 -1 +1]);
hold off;
figure(2);

```

Anexo 1: Script Matlab que desarrolla la identificación del sistema con redes neuronales dinámicas.

```
plot(tt,u,'-b','LineWidth',1.5);  
title('Señal de entrada "u" ');  
xlabel('Time [sec]');  
ylabel('u');  
legend('Señal de entrada "u"');  
axis([ 0 20000 -1 +1]);
```

```
save reddbp2 ne nm ns v w v2 w2 c a; %Se guarda los pesos entrenados para futuras iteraciones
```

```
vaf = (1-(var(zesc(2:end,1) - outputesc(:,1))./var(zesc(2:end,1))))*100;  
rmse = rmse(cont-1,1);  
fit = goodnessOfFit(outputesc(:,1), zesc(2:end,1) , 'NRMSE');
```

```
fprintf('\n          VAF | RMSE | FIT \n');  
fprintf('-----\n');  
fprintf('          %2.4f | %2.4f | %2.4f \n', vaf, rmse, fit);
```

Fuente: Elaboración propia.

Anexo 2: Script de Matlab que desarrolla la validación del sistema.

```
figure(1);
plot(tt,dataoutesc(:,1),'-r','LineWidth',1.5);
hold on;
plot(tt,outputesc(:,1),'-b','LineWidth',1.5);
title('Salida pH Planta Vs Salida pH Neuronal');
xlabel('Time [sec]');
ylabel('pH');
legend('Salida pH planta', 'Salida pH neuronal');
axis([ 0 20000 -1 +1]);
hold off;

figure(2);
plot(tt,u,'-b','LineWidth',1.5);
title('Validación - Señal de entrada "u" ');
xlabel('Time [sec]');
ylabel('u');
legend('Señal de entrada "u"');
axis([ 0 20000 -1 +1]);

vaf = (1-(var(zesc(2:end,1) - outputesc(:,1))./var(zesc(2:end,1))))*100;
rmse = rmse(cont-1,1);
fit = goodnessOfFit(outputesc(:,1), zesc(2:end,1) ,'NRMSE');

fprintf('\n          VAF |  RMSE |  FIT  \n');
fprintf('-----\n');
fprintf('          %2.4f |  %2.4f |  %2.4f \n', vaf, rmse, fit);
```

Fuente: Elaboración propia.

Anexo 3: Script Matlab que desarrolla la identificación del sistema con redes neuronales estáticas (continúa).

```

% Entrenamiento batch - Salida vectorial
% Basico

clear;
clc;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load ('Entrada1.mat');
load ('Salida1.mat');

u = Entrada1(100:4101,1); %Entrada
nu = length(u);
z1 = Salida1(100:4101,1);

% Escalamiento
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xmax = max(abs(u));
xmin = min(abs(u));
pmax = 1;
pmin = -1;
uesc(:,1)=((pmax-pmin).*u + ((pmin*(xmax - xmin)).*ones(nu,1)) - ((xmin*(pmax-
pmin)).*ones(nu,1)) )./(xmax - xmin).*ones(nu,1));

ymax1 = max(abs(z1));
ymin1 = min(abs(z1));
zesc(:,1) = ((pmax-pmin).*z1(:,1) + ((pmin*(ymax1 - ymin1)).*ones(nu,1)) - ((ymin1*(pmax-
pmin)).*ones(nu,1)) )./(ymax1 - ymin1).*ones(nu,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u = uesc;
dataoutesc = zesc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nv = length(uesc);
x1 = uesc;
x2(1,1) = 0;
x2(2:nv,1) = zesc(1:nv-1,1);
x3(1,1) = 0;
x3(2:nv,1) = x2(1:nv-1,1);
x4(1,1) = 0;
x4(2:nv,1) = x3(1:nv-1,1);
x5(1,1) = 0;
x5(2:nv,1) = x4(1:nv-1,1);
x6(1,1) = 0;
x6(2:nv,1) = x5(1:nv-1,1);

```


Anexo 3: Script Matlab que desarrolla la identificación del sistema con redes neuronales estáticas.

```
dJdw = dJdw + n*er';
dJdv = dJdv + in * (dndm.*(w*er))';
end
rmse(iter,1) = norm(error)/sqrt(nx);
rmse(iter,1)
w = w - eta*dJdw/nx;
v = v - eta*dJdv/nx;
end

figure(1);
plot(yesc(:,1),'--b');
hold on;
plot(y(:,1),'-r','LineWidth',1.1);
title('RED NEURONAL ESTATICA');
legend('Salida deseada', 'Salida neuronal');
xlabel('patrones [k]');
ylabel('Salidas');
axis([0 nu -1 +1]);

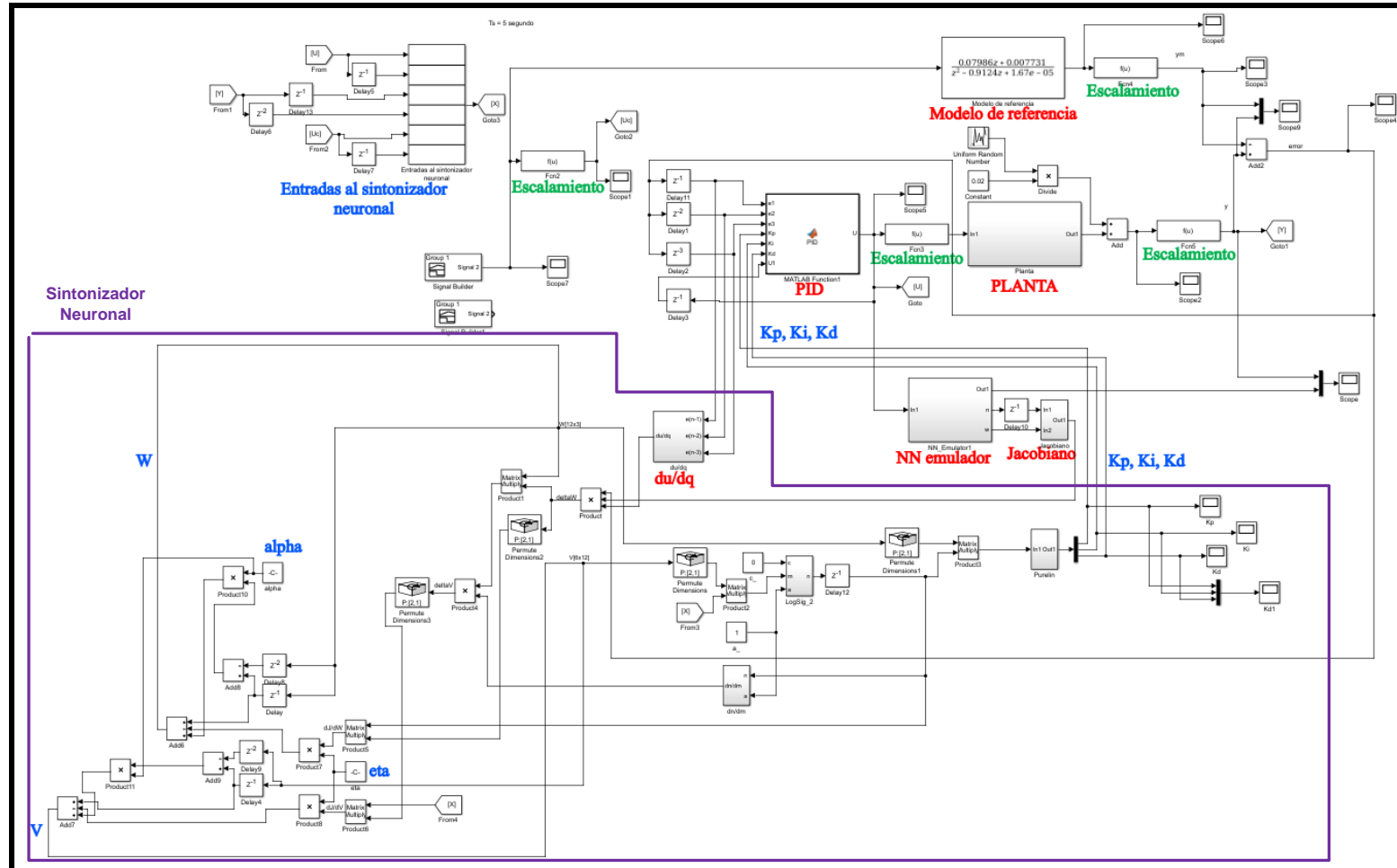
save redstat v w;

vaf = (1-(var(yesc - y)./var(yesc)))*100;
rmse = rmse(iter,1);
fit = goodnessOfFit(y, yesc , 'NRMSE');

fprintf('\n      VAF | RMSE | FIT \n');
fprintf('-----\n');
fprintf('      %2.4f | %2.4f | %2.4f \n', vaf, rmse, fit);
```

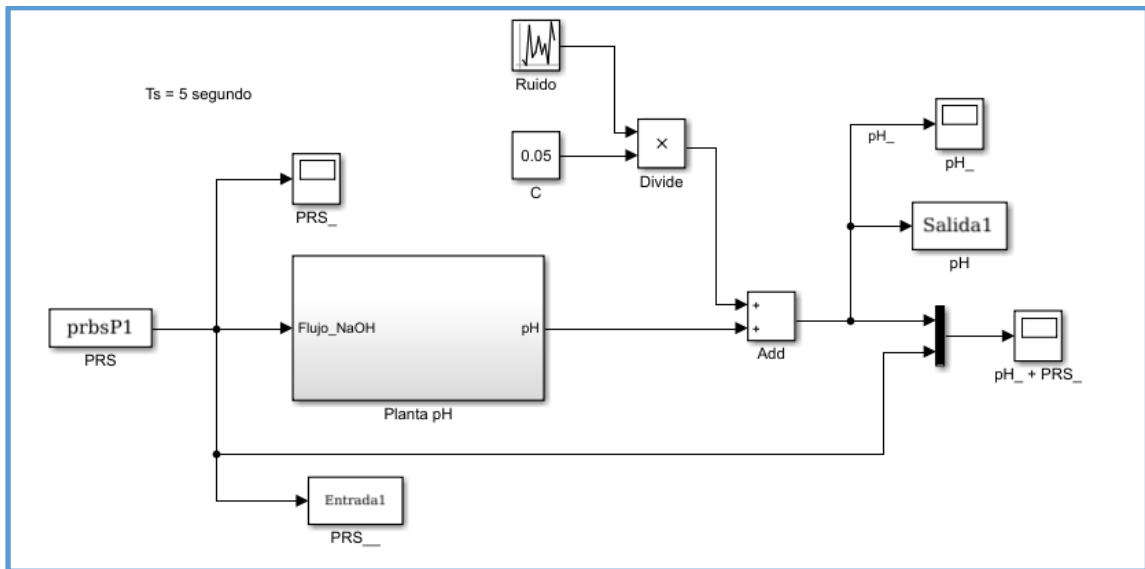
Fuente: Elaboración propia.

Anexo 4: Desarrollo del Controlador PID autosintonizado con redes neuronales dinámicas y control adaptativo.



Fuente: Elaboración propia

Anexo 5: Planta desarrollada en Simulink para la identificación del sistema.



Fuente: Elaboración propia.