UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"DESARROLLO DE UN CONTROL PID WAVENET"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

(MÓDULO CONTROL Y ROBÓTICA)

PRESENTA:

JOAQUÍN MANRIQUE DE LA CRUZ

Director de Tesis:

Ing. Alberto Ramiro Garibay Martínez.

CIUDAD UNIVERSITARIA, MÉXICO, D.F. 2010

Agradecimientos

Al Creador, por un universo que vale la pena estudiar desde muchos puntos de vista, y por la voluntad de dedicar la vida a estudiarlo.

A mis padres, Joaquín y Juanita, que me hicieron posible y con ello hicieron posible todo lo demás. Es por su amor, su apoyo, su cariño, su comprensión, confianza y dedicación que he llegado hasta donde estoy, que sé que puede llegar aún más lejos, y que tengo la oportunidad de escribir todo esto.

A mi familia, por todo lo que representan, y lo que llevo conmigo de todos ustedes.

A mi Fellowship: Kenji, Armi, Pan, Yáñez, Cowen, Confi, porque ya que todo esto era posible, es gracias a ustedes que fue interesante. Por saber ser amigos, hermanos e inspirarme a ser mejor para ser un poquito más como ustedes.

A Elizabeth, por tu cariño, tu apoyo, y por la luz que de otro modo no brillaría para mí.

A todos mis profesores, en especial al Profe Ray y al Maestro Mario, por su vocación de hacerme un hombre menos ignorante, por todo lo que me enseñaron y también por lo que no.

A la UNAM, por darme los medios para terminar esta etapa de mis estudios y comenzar con la siguiente.

Agradecimientos al M. en I. José Alberto Cruz Tolentino por su apoyo en la realización de este trabajo.

Índice

Capítulo 1
Introducción

En la práctica de la ingeniería comúnmente existe la necesidad de llevar un sistema físico hasta un punto determinado con cierto grado de exactitud. Un sistema de control es el conjunto de técnicas y herramientas que guían un sistema físico hasta las condiciones deseadas. Dentro de la vida cotidiana son cada vez más los aspectos en que se han visto involucrados los sistemas de control, ya sea en la producción industrial, sistemas de transporte, líneas de ensamble, sistemas de seguridad, etc. En algunos de éstos, la precisión y certidumbre de los procesos realizados es de vital importancia, y en ellas el control juega un papel fundamental.

El fundamento del control automático consiste en calcular una acción correctiva en función de la diferencia entre el estado actual del proceso y el estado deseado. Esto se conoce como control realimentado y está conformado por tres bloques básicos: un proceso, planta o sistema a controlar, un sensor y un controlador, como se muestra en la Figura 1.1.

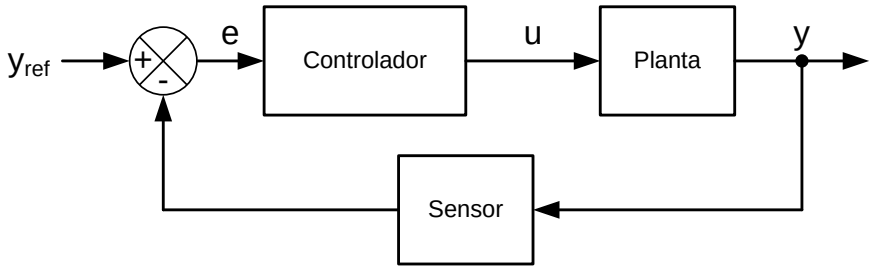


Figura 1.1. Esquema del control realimentado.

El controlador es un conjunto de elementos (de tipo mecánico, eléctrico, hidráulico, neumático, entre otros) que recibe en su entrada el error de referencia y ejecuta una ley de control para producir la señal correctiva de entrada del proceso.

El control PID es el controlador industrial más utilizado, empleado por su buen desempeño y facilidad de implementación. El acrónimo PID proviene de las tres partes que lo componen: una acción proporcional, una acción integral y una acción derivativa: el componente proporcional ajusta la ganancia en relación directa con la magnitud del error, el componente integral elimina el error en estado estacionario y el componente derivativo mejora la velocidad de respuesta [9].

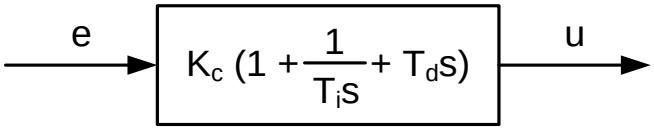


Figura 1.2. Diagrama de bloques de un controlador PID.

Para conseguir un buen rendimiento y evitar comportamientos indeseables en el sistema es preciso ajustar los parámetros del controlador hasta que el desempeño del proceso se considere satisfactorio [19]. A este procedimiento se le conoce como sintonización del controlador y puede realizarse por prueba y error, o siguiendo uno de un conjunto de algoritmos de sintonización que permiten encontrar de manera sistemática valores adecuados de los parámetros del controlador. Existen diversos métodos de diseño de controladores, que requieren ya sea el modelo del proceso o su respuesta en frecuencia para calcular las ganancias óptimas del controlador. Aún así se requiere hacer pruebas de campo para realizar ajustes finales, sobre todo si la información del proceso es incompleta o inexacta. De manera alternativa, se han planteado controladores adaptables, que tienen la capacidad de encontrar valores adecuados durante la operación automática.

El presente trabajo propone el desarrollo e implementación de un controlador tipo PID sintonizado empleando una red wavenet. Las redes wavenet se han utilizado con anterioridad para aproximación de funciones e identificación de procesos cuyas funciones de transferencia –el modelo matemático que describe el comportamiento del proceso– son desconocidas. En trabajos previos ([2] y [17]) se han empleado algoritmos wavenet para sintonizar controladores en plantas con modelos no lineales.

El propósito de utilizar un PID wavenet es que los parámetros del PID respondan a posibles cambios de la planta o a funciones no lineales con cambios no previstos. Un método autosintonizado es una consideración importante de diseño de sistemas para construir controles adaptables de un *sistema desconocido que varía lentamente*. La idea básica en el control adaptable es estimar los parámetros desconocidos de la planta y correspondientemente ajustar en línea los parámetros del controlador, basados en las

señales medidas del sistema, empleando los parámetros estimados en el cálculo de la entrada de control [6].

El algoritmo PID wavenet aquí presentado se desarrolla empleando simulaciones numéricas en el entorno de desarrollo LabVIEW. Posteriormente, se evalúa su desempeño en tiempo real mediante la implementación en un módulo compactRIO y el simulador de procesos PCS327.

1.1 Estudio del estado del arte

Las primeras menciones de teoría wavelet aparecen en el trabajo de Haar en 1909. Gran parte del desarrollo se realizó durante la década de 1930 por varios grupos de investigación en el tema de la representación de funciones empleando funciones base variantes de escala. David Marr produjo un algoritmo para procesar imágenes empleando wavelets en la década de 1980. En esta misma década Meyer construyo wavelets continuamente diferenciables (en contraste con las wavelets de Haar). Daubechies se basó en el trabajo de Mallat para construir una base ortonormal de funciones, que se ha convertido en el fundamento de las aplicaciones más recientes de las wavelets. [7]

Entre los usos más comunes de las transformadas wavelet están el análisis multirresolución y aproximación de funciones e identificación de parámetros.

Las redes neuronales artificiales se desarrollaron como un modelo matemático generalizado del aprendizaje humano basado en neuronas reales [15]. Esta forma de modelado fue estudiada originalmente por McCulloch y Pitts [12]. Este modelo permite almacenar información y reconocer patrones. Lippman las utilizó en 1987 para reconocimiento de patrones [11]. En la década de 1990 se investigó la descripción matemática de modelos de redes neuronales, su arquitectura y algoritmos de aprendizaje ([3], [4], [8] y [16]). Las redes neuronales ven uso en diversos campos, tales como las telecomunicaciones y reconocimiento de patrones en medicina e ingeniería.

La aplicación conjunta de las dos técnicas anteriores da lugar a las redes wavenet: redes neuronales que utilizan wavelets como funciones de activación, como en [19], que se presenta un control PID autosintonizable, [10] se utiliza una red wavenet para identificación de parámetros y otra red para la sintonización del controlador. También se le utiliza en [17] para controlar un generador eólico. En [2] se describe un método similar al presentado en esta tesis para sintonizar un PID multiresolución.

1.2 Planteamiento del problema

La implementación de un controlador PID wavenet para su ejecución a nivel académico en la tarea de controlar un sistema cuya función de transferencia es desconocida.

1.3 Objetivos

Ejecutar un controlador PID wavenet eficientemente en un equipo de operación en tiempo real.

1.4 Justificación

Se ha visto que el algoritmo PID wavenet y las tecnologías de redes neuronales tienen un gran potencial a desarrollar dentro de varios campos en el área del control. Las redes neuronales se han empleado para la identificación de sistemas físicos y en el auxilio de sistemas de control, cálculo de propiedades de estructuras y análisis de señales. También se les ha empleado en conjunto con las redes neuronales wavenet, como puede verse en la documentación consultada en la elaboración de este trabajo. En las aplicaciones específicas del control PID Wavenet, el algoritmo aún está poco documentado en cuanto a aplicaciones a nivel industrial o más allá de lo académico, lo cual es un espacio de trabajo que puede ser explorado más a profundidad y dar lugar a nuevas investigaciones.

El trabajo presente busca contribuir con mediciones experimentales y evidencia de funcionamiento en equipo de grado industrial.

Capítulo 2
Antecedentes

En este capítulo se explican las principales herramientas matemáticas en que se basa este trabajo. La primera sección se ocupa del controlador PID discreto, un controlador comúnmente empleado. La sección 2.2 trata sobre las funciones wavelet, funciones matemáticas que se emplean para el análisis de datos. La sección 2.3 explica las herramientas conocida como redes neuronales artificiales, construcciones matemáticas empleadas para aproximación y reconocimiento de funciones que pueden actualizarse mientras ocurren en el tiempo. En la última parte se explica la construcción de la red wavenet, combinando las dos herramientas anteriores.

2.1 Teoría de control PID

Un controlador PID es una unidad diseñada para crear un lazo de control estable y alcanzar el desempeño deseado mediante tres tipos de acciones básicas de corrección de error: proporcional, integral y derivativa, que dan origen al mnemónico PID. Cada una de estas acciones resuelve un problema específico relativo al error.

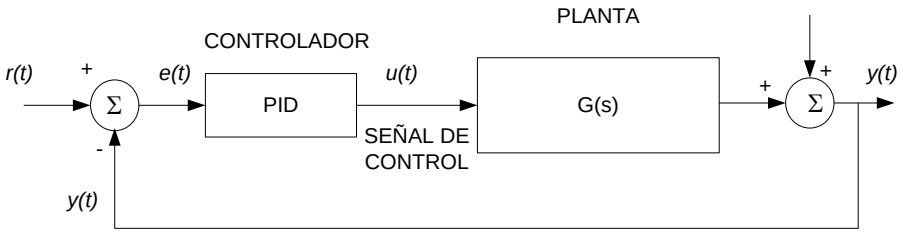


Figura 2.1. Esquema del controlador PID.

La acción proporcional es la principal respuesta del controlador ante las perturbaciones del lazo de control, ofrece una acción correctiva proporcional al tamaño del error. Si bien existe en todo el lazo de control, su presencia no garantiza que el cambio de valor de la

variable controlada coincida exactamente con el valor deseado para cualquier cambio de carga.

La función de la acción integral es eliminar el offset generado por la acción proporcional y asegurar que la salida del proceso corresponde a la señal de referencia. La acción proporcional genera un error en estado estacionario, pues una acción correctiva proporcional al tamaño del error no necesariamente conduce al valor de referencia.

La acción integral se agrega para asegurar que cuando el error acumulado es positivo se ofrece una señal de control que reduce el tamaño de la salida y cuando el error acumulado es negativo se ofrece una señal de control que aumenta la salida, sin importar qué tan pequeño es el error [5].

La acción derivativa mejora la velocidad de respuesta del controlador. Las anteriores acciones responden al error presente; la acción derivativa extrapola el error futuro como la tangente de la curva del error y hace la señal correctiva proporcional al error predicho.

El comportamiento del controlador PID es consecuencia de la aplicación paralela de estas tres acciones:

$$u(t) = K_c \left[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right] \tag{2.1}$$

donde K_c es la ganancia proporcional, T_i el tiempo integral y T_d el tiempo derivativo.

A partir de la ecuación (2.1) se puede obtener la función de transferencia siguiente:

$$u(s) = K_c \left[1 + \frac{1}{T_i s} + T_d s \right] E(s) \tag{2.2}$$

Es posible observar que la ganancia proporcional actúa sobre el total de la respuesta. Las constantes integral y derivativa representan el peso relativo de cada una de estas acciones respecto a la acción proporcional, originando los siguientes parámetros:

$$K_i = \frac{K_c}{T_i}, \quad y \quad K_d = K_c T_d \tag{2.3}$$

2.2 Funciones wavelet

El término wavelet (ondeleta u ondita) se refiere a un grupo de funciones que presentan una pequeña oscilación y alcanzan rápidamente cero. Es posible emplear conjuntos de wavelets para aproximar una señal utilizando versiones trasladadas y dilatadas o contraídas en el tiempo para representar mejor la señal. De esta manera, son herramientas de descomposición de funciones que permiten el análisis en diferentes niveles de frecuencia y tiempo de una función. Una función wavelet es una función $\psi \in L^2$ que cumple con la siguiente condición de admisibilidad:

$$c_h = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty \tag{2.4}$$

donde $\Psi(\omega)$ es la transformada de Fourier de la función $\psi(t)$.

La función wavelet generalmente se conoce como wavelet madre, porque a partir de ésta se generan funciones "wavelet hijas" a partir de operaciones de dilatación y traslación,

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \quad a > 0; \quad a, b \in \mathbb{R} \tag{2.5}$$

donde $\psi_{a,b}(t)$ es la wavelet hija, a es el factor de dilatación-contracción, que varía la duración de la wavelet, y b es un desplazamiento en el tiempo. $\frac{1}{\sqrt{a}}$ es un factor de normalización que permite que

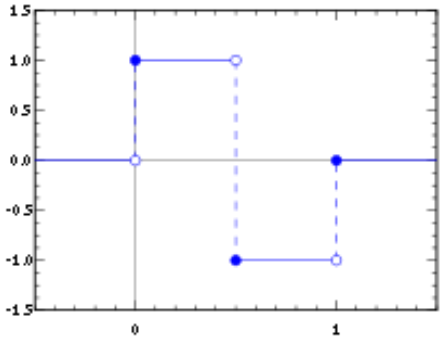
$$\|\psi_{a,b}\| = \|\psi\|, \quad \forall\, a, b \in \mathbb{R}.$$

En la Tabla 2.1 se presentan algunos ejemplos de wavelets comúnmente empleadas.

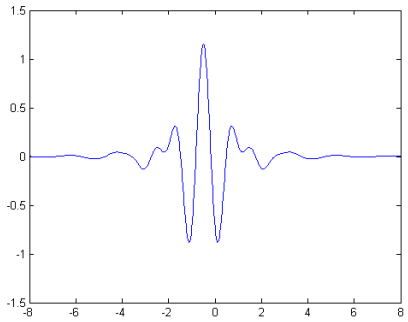
Wavelet	Función
Haar	$\psi(t) = \begin{cases} 1, & si\ 0 \le t \le \frac{1}{2}, \\ -1, & si\ \frac{1}{2}, \le t \le 1, \\ 0, & en\ cualquier\ otro\ caso \end{cases}$

Meyer	$\psi(w) = \begin{cases} (2\pi)^{-\frac{1}{2}} e^{\frac{iw}{2}} \sin\left(\frac{\pi}{2} v\left(\frac{3}{2\pi}\lvert w\rvert - 1\right)\right), & si \ \frac{2\pi}{3} \leq \lvert w\rvert \leq \frac{4\pi}{3}, \\ (2\pi)^{-\frac{1}{2}} e^{\frac{iw}{2}} \cos\left(\frac{\pi}{2} v\left(\frac{3}{4\pi}\lvert w\rvert - 1\right)\right), & si \ \frac{4\pi}{3} \leq \lvert w\rvert \leq \frac{8\pi}{3}, \\ 0, \quad si \ \lvert w\rvert \notin \left[\frac{2\pi}{3}, \frac{8\pi}{3}\right] \end{cases}$
Morlet	$\psi(t) = e^{-\frac{t^2}{2}} \cos(5t)$
Mexican Hat	$\psi(t) = \frac{3}{\sqrt{3}} \pi^{-\frac{1}{4}} (1 - t^2) e^{\left(\frac{1}{2}t^2\right)}$
RASP1	$\psi(t) = \frac{t}{t^2 + 1}$

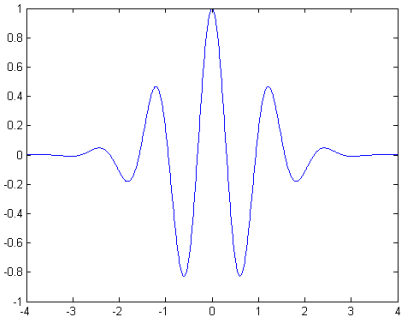
Tabla 2.1. Algunas wavelets madre.



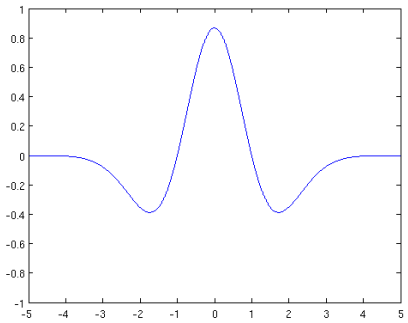
Wavelet Haar



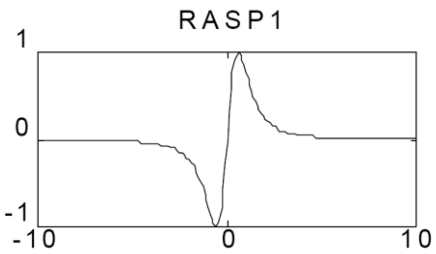
Wavelet Meyer



Wavelet Morlet



Wavelet Mexican Hat



Wavelet RASP1

Figura 2.2. Algunas wavelets comunes.

Las funciones wavelet descritas anteriormente forman la base de la transformada wavelet. La transformada wavelet es un operador que transforma una función integrándola con versiones modificadas de una función, en este caso, una wavelet madre.

La transformada wavelet de una función $f \in L^2$ respecto a una wavelet admisible está definida como:

$$W_f(a,b) = \int_{-\infty}^{\infty} f(t)\,\psi\left(\frac{t-b}{a}\right) dt$$

para $a \neq 0$, donde $\psi(\cdot)$ es la función wavelet. El efecto de traslación provoca que la transformada wavelet realice un efecto de ampliación en fenómenos de alta frecuencia y de breve duración. Conforme nos movemos de escalas mayores a menores, se pueden conseguir mejores representaciones de estos momentos en las señales.

2.3 Redes neuronales artificiales

Una red neuronal artificial es un modelo matemático adaptable que emula el funcionamiento de una neurona biológica por medio de la interconexión de nodos de proceso simple (neuronas artificiales).

La estructura de una red neuronal simple de L neuronas es la siguiente:

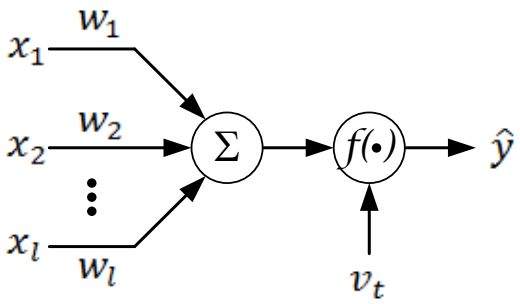


Figura 2.3. Estructura básica de una red neuronal.

donde x_l es la l-ésima entrada a la neurona, w_l es el peso asociado a dicha entrada, $f(\cdot)$ es la función de activación de la neurona, $\hat{y}$ la señal de salida y v_t la señal de bias o umbral.

La función de salida de la neurona es función de la suma ponderada de las entradas:

$$\hat{y} = f\left(\sum_{l=1}^{L} w_l x_l - v_t\right) \tag{2.6}$$

Esto es importante porque podemos establecer una función de aprendizaje a partir del gradiente de $\hat{y}$ para permitir que los pesos de cada conexión se actualicen.

Se define una función de medida del error E como:

$$E = \frac{1}{2}(y - \hat{y})^2 \tag{2.7}$$

donde y es la función que se desea aproximar o función objetivo y $\hat{y}$ es la salida de la red neuronal.

De acuerdo con el método del gradiente inverso, obtenemos Δw como la derivada parcial del error respeto a w.

$$\Delta w(k) = -\frac{\partial E}{\partial w} \tag{2.8}$$

El cambio de w para la siguiente iteración se obtiene de la siguiente manera:

$$w(k+1) = w(k) + \mu_w \Delta w(k) \tag{2.9}$$

Donde μ_w es la velocidad de aprendizaje de la neurona.

2.4 Teoría wavenet

La teoría wavenet combina los dos conceptos vistos previamente, teoría de las redes neuronales artificiales y funciones wavelet para generar un algoritmo de aproximación de funciones. La figura 2.4 a continuación muestra la arquitectura básica de una red wavenet adaptable:

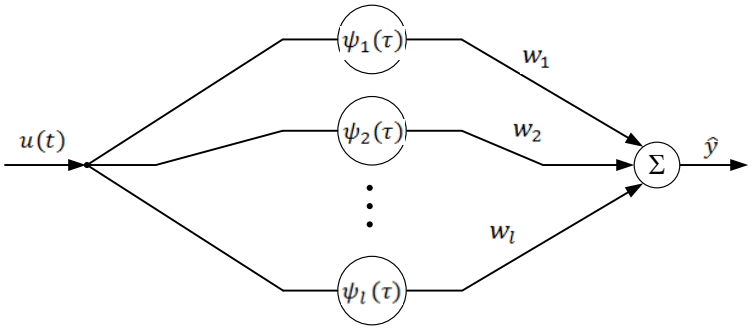


Figura 2.4. Diagrama de una red neuronal wavenet.

En la red wavenet $u(t)$ es la señal de entrada, $\hat{y}(t)$ la señal de salida de la red, $\psi_l(\tau)$ es la función wavelet de activación de la l-ésima neurona y w_l es el peso de conexión entre las neuronas.

Una aproximación razonable se consigue si las regiones de tiempo de la señal están cubiertas adecuadamente por las L ventanas de la red neuronal. La señal aproximada por la red es

$$\hat{y}(t) = u(t)\sum_{l=1}^{L} w_l \psi_l(\tau), \quad \hat{y}, u, w \in \mathbb{R} \tag{2.10}$$

Se definen los siguientes vectores:

$$\mathbf{A}(k) \triangleq [a_1(k) \quad a_2(k) \quad \dots \quad a_l(k) \quad \dots \quad a_{L-1}(k) \quad a_L(k)]^T \tag{2.11}$$
$$\mathbf{B}(k) \triangleq [b_1(k) \quad b_2(k) \quad \dots \quad b_l(k) \quad \dots \quad b_{L-1}(k) \quad b_L(k)]^T \tag{2.12}$$
$$\mathbf{W}(k) \triangleq [w_1(k) \quad w_2(k) \quad \dots \quad w_l(k) \quad \dots \quad w_{L-1}(k) \quad w_L(k)]^T \tag{2.13}$$
$$\mathbf{\Psi}(\tau) \triangleq [\psi_1(\tau) \quad \psi_2(\tau) \quad \dots \quad \psi_l(\tau) \quad \dots \quad \psi_{L-1}(\tau) \quad \psi_L(\tau)]^T \tag{2.14}$$

Y con ello es posible representar la ecuación (2.10) como

$$\hat{y} = u(t)\mathbf{\Psi}^T(\tau)\mathbf{W}(k) \tag{2.15}$$

Adicionalmente, se puede lograr una aproximación doble colocando un filtro de respuesta infinita al impulso en cascada con la red neuronal. El filtro IIR se emplea como un segundo nivel de identificación y aproximación de la señal.

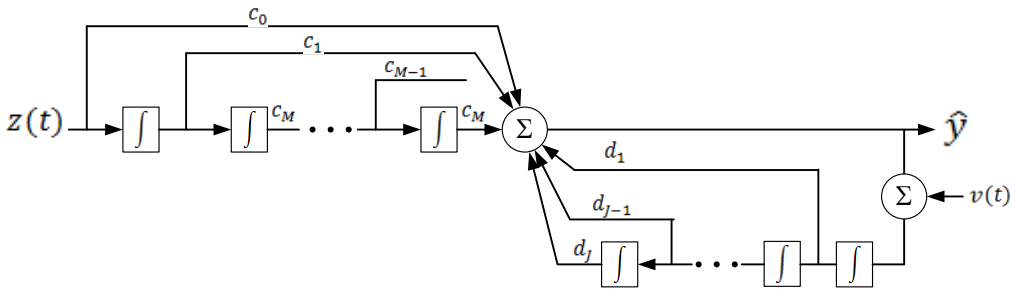


Figura 2.5. Diagrama del filtro IIR. $z(t)$ es la señal de entrada, $\hat{y}(t)$ es la señal de salida, c y d son los coeficientes del filtro, $v(t)$ es la señal de bias.

La estructura definitiva de la red se muestra a continuación:

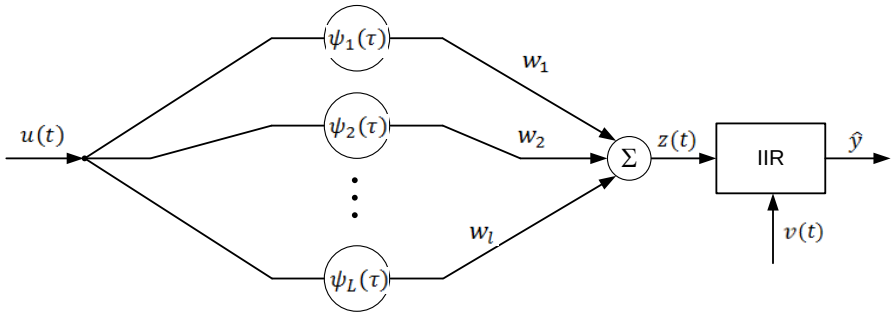


Figura 2.6. Red neuronal en cascada con un filtro IIR.

Se definen los vectores $\mathbf{C}$ y $\mathbf{D}$ de coeficientes del filtro, el vector $\mathbf{Z}$ que contiene las salidas actual y anteriores de la red neuronal y el vector $\hat{\mathbf{Y}}$ que contiene las salidas actual y anteriores del filtro IIR.

$$\mathbf{C}(k) \triangleq [c_1(k) \quad c_2(k) \quad \dots \quad c_m(k) \quad \dots \quad c_{M-1}(k) \quad c_M(k)]^T \tag{2.16}$$

$$\mathbf{D}(k) \triangleq [d_1(k) \quad d_2(k) \quad \dots \quad d_j(k) \quad \dots \quad d_{J-1}(k) \quad d_J(k)]^T \tag{2.17}$$

$$\mathbf{Z}(k) \triangleq [z(k) \quad z(k-1) \quad \dots \quad z(k-m) \quad \dots \quad z(k-M+1) \quad z(k-M)]^T \tag{2.18}$$

$$\hat{\mathbf{Y}}(k) \triangleq [\hat{y}(k) \quad \hat{y}(k-1) \quad \dots \quad \hat{y}(k-j) \quad \dots \quad \hat{y}(k-J+1) \quad \hat{y}(k-J)]^T \tag{2.19}$$

donde

$$z(k) = \mathbf{\Psi}^T(\tau)\mathbf{W}(k) \tag{2.20}$$

La función de salida de la red, la función de aproximación $\hat{y}$ se puede expresar de la siguiente forma:

$$\hat{y} = u(k)\mathbf{C}^T(k)\mathbf{Z}(k) + v(k)\mathbf{D}^T(k)\hat{\mathbf{Y}}(k) \tag{2.21}$$

Es posible actualizar los parámetros de la red wavenet para reducir el error entre la estimación y la señal deseada, mejorando la señal de aproximación $\hat{y}$. Se emplea un algoritmo de mínimos cuadrados medios por medio de la minimización de la función de costo o función de error E (que es una función de energía) a lo largo de todo el tiempo.

Los coeficientes son actualizados como se muestra en la ecuación (2.9) de acuerdo con la siguiente ley:

$$\begin{aligned} \mathbf{A}(k+1) &= \mathbf{A}(k) + \mu_A \Delta\mathbf{A} \\ \mathbf{B}(k+1) &= \mathbf{B}(k) + \mu_B \Delta\mathbf{B} \\ \mathbf{W}(k+1) &= \mathbf{W}(k) + \mu_W \Delta\mathbf{W} \\ \mathbf{C}(k+1) &= \mathbf{C}(k) + \mu_C \Delta\mathbf{C} \\ \mathbf{D}(k+1) &= \mathbf{D}(k) + \mu_D \Delta\mathbf{D} \end{aligned} \tag{2.22}$$

μ es la velocidad de aprendizaje de cada parámetro.

Capítulo 3
Módulo CompactRIO de National Instruments

CompactRIO es una plataforma de adquisición de datos y control impulsada por tecnología de entrada y salida reconfigurable (reconfigurable I/O, RIO). CompactRIO combina un procesador embebido en tiempo real, un arreglo de compuertas lógicas programables (FPGA por sus siglas en inglés) de alto rendimiento y módulos de entrada/salida intercambiables. Los módulos de E/S se conectan directamente al FPGA, proporcionando personalización de bajo nivel para temporización y procesamiento de señales de E/S. El FPGA se conecta a un procesador embebido en tiempo real vía un bus PCI de alta velocidad. Mediante la plataforma de desarrollo LabVIEW es posible la transferencia de datos entre los módulos de E/S y el FPGA, y entre el FPGA y el procesador embebido para análisis en tiempo real, procesamiento posterior, registro de datos o comunicación a un servidor conectado en red.

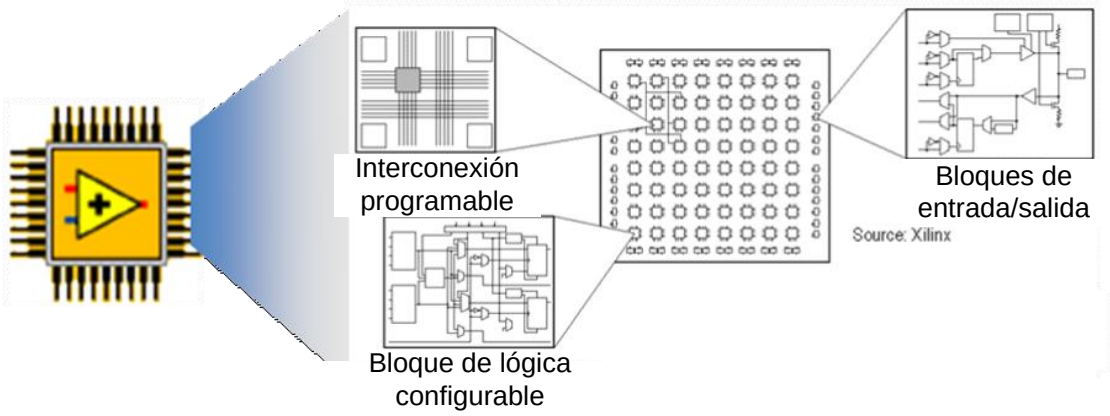


Figura 3.1. Enfoque LabVIEW FPGA.

3.1 Plataforma de desarrollo CompactRIO

El sistema de circuitos FPGA de RIO es un motor de computación reconfigurable de procesamiento paralelo que ejecuta aplicaciones integradas de LabVIEW determinísticamente. CompactRIO permite implementar sistemas de control PID analógicos de lazos múltiples a razones de 100 [kmuestras/s] y sistemas de control digital a razones de lazo de hasta 1 [Mmuestras/s] y evaluar escalones de lógica booleana de menos de 25 [ns].

Los principales componentes de un sistema de LabVIEW FPGA son el Módulo LabVIEW FPGA y el hardware RIO. La programación del código en LabVIEW ofrece la ventaja de que la representación del tiempo, concurrencia y paralelismo del hardware del FPGA es intuitivo al momento de implementar. Adicionalmente, se puede integrar el hardware del FPGA con otros componentes de sistema de medición y control usando VIs de Interfaz de LabVIEW FPGA.

Un sistema de control tradicional cuenta con tres etapas:

- El software de la aplicación corriendo en Windows o en un sistema operativo de tiempo real (RTOS).
- El software driver y las funciones para interactuar con el hardware.
- El hardware de Entrada/Salida.

El esquema anterior varía cuando se usa un sistema basado en LabVIEW FPGA. El software de la aplicación se ejecuta en Windows o en un RTOS y la interfaz con el hardware se realiza por medio de un driver National Instruments RIO (NI-RIO). Es necesario utilizar hardware para interaccionar con el proceso, que en este caso son dispositivos de la serie C, y la funcionalidad de este dispositivo es modificable usando LabVIEW FPGA. En síntesis, se trabaja en una plataforma FPGA de Entrada/Salida (I/O) que ofrece buen rendimiento, flexibilidad de aplicaciones y diseño de hardware a nivel de tarjeta.

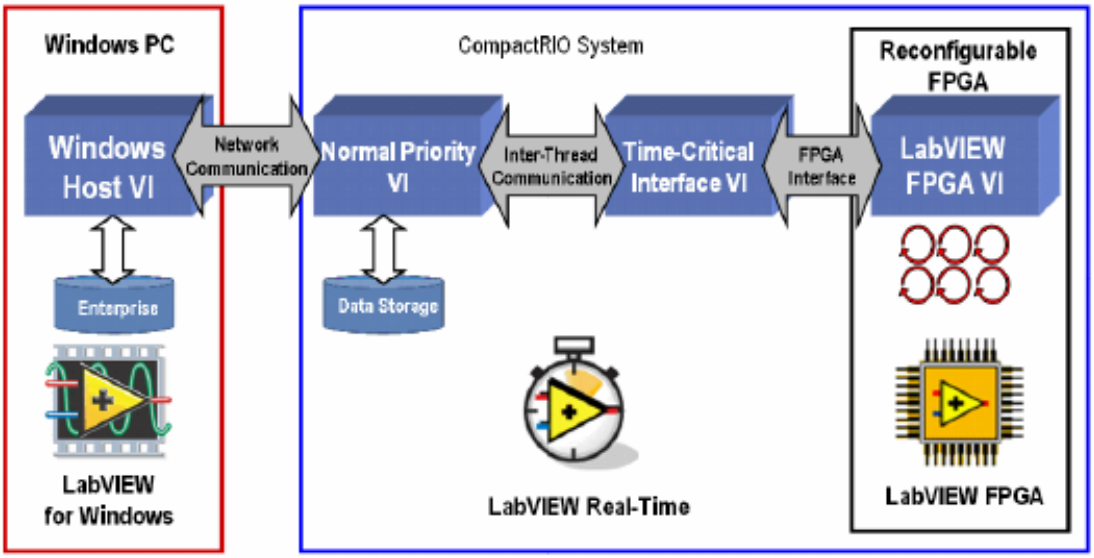


Figura 3.2. Paradigma de programación LabVIEW FPGA.

Los controladores embebidos CompactRIO ofrecen ejecución autónoma para aplicaciones determinísticas de LabVIEW Real-Time. CompactRIO opera con un procesador industrial de 200 [MHz] con capacidad de efectuar operaciones de punto fijo de procesamiento de señales y análisis para ciclos de control de 1 kHz. Los controladores NI CompactRIO tienen entradas de doble suministro de 11 a 30 [V_{DC}], un consumo de 7 a 10 [W] y un rango de temperatura de -40 a 70 [°C]. Los módulos de E/S de CompactRIO tienen un aislamiento de hasta 2,300 [V_{RMS}] (tolerancia) y un aislamiento de 250 [V_{RMS}] (continuo). Cada componente viene con una variedad de certificaciones y graduaciones internacionales de seguridad, ambientales y de compatibilidad.

CompactRIO está diseñado para utilizar las herramientas de desarrollo gráfico de LabVIEW para adaptar el hardware reconfigurable a una variedad de industrias y aplicaciones. El sistema incluye el controlador en tiempo real cRIO-9002 con procesadores industriales de punto flotante de 200 [MHz], el chasis cRIO-9102 reconfigurable de ocho ranuras con FPGA de 1 millón de compuertas, el módulo de ocho Entradas Analógicas 9201 que acepta un rango de ±10 [V], y finalmente el módulo de 4 Salidas Analógicas con un rango de salida de voltaje de ±10 [V].

En esta tesis, se programó el hardware de CompactRIO usando LabVIEW 2009 en Windows XP, el Módulo de LabVIEW Real-Time y el Módulo de LabVIEW FPGA.

3.2 Instalación y configuración

Esta sección describe el proceso de configuración del módulo compactRIO y los pasos previos a la implementación del programa que se describe en el capítulo 5.

En primer lugar se asigna un nombre al sistema compactRIO para su identificación en red con la PC. Se aceptan los valores sugeridos por el sistema, aplicando la configuración predeterminada y se reinicia el sistema antes de establecer la comunicación.

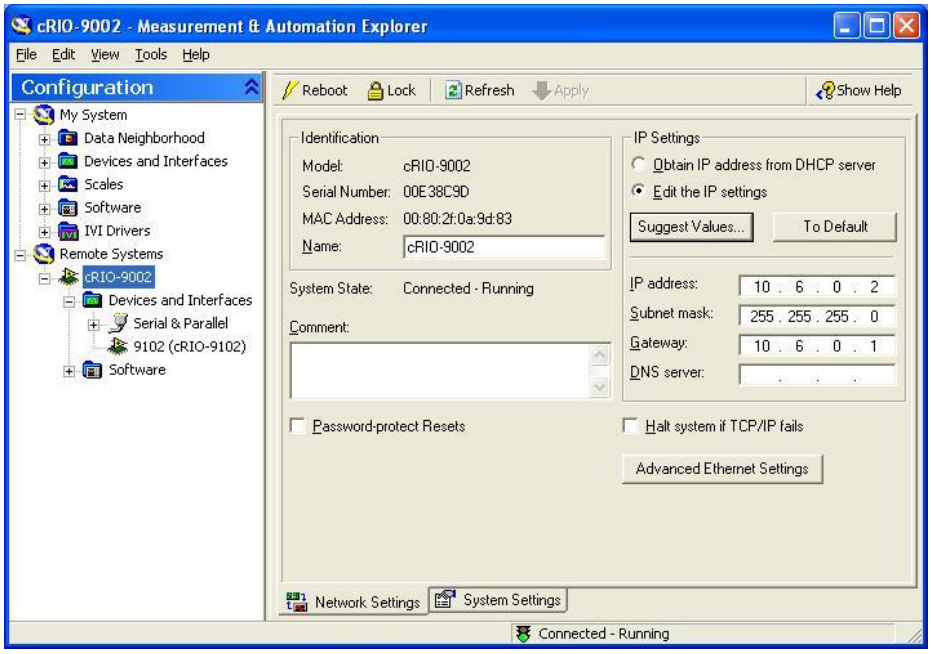


Figura 3.3. Measurement & Automation Explorer, mostrando la configuración para el módulo cRIO-9002.

Se crea un proyecto en blanco. En la ventana *Project Explorer* se abre el menú contextual en el icono del proyecto y se selecciona el comando *New>Targets and devices*, para agregar el dispositivo CompactRIO a la estructura.

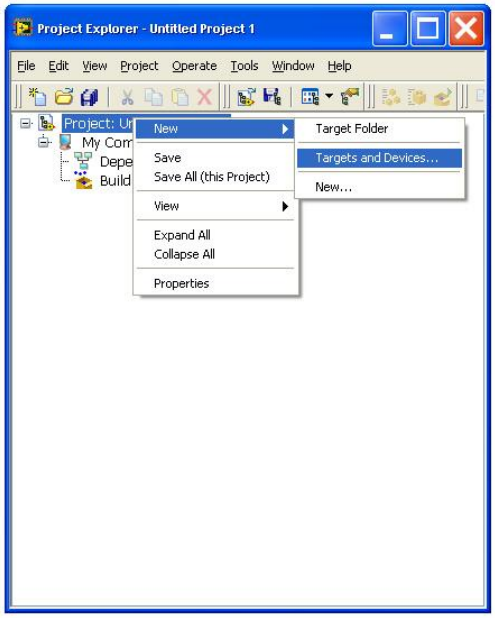


Figura 3.4. Explorador de proyecto, agregando un nuevo dispositivo.

La ventana desplegada permite la selección de un dispositivo de tiempo real entre varias opciones y varias maneras de detección. En el primer grupo de opciones se escoge *Existing target or device*, y del menú de opciones se expande la carpeta Real-Time CompactRIO para seleccionar el modelo del controlador, cRIO-9002.

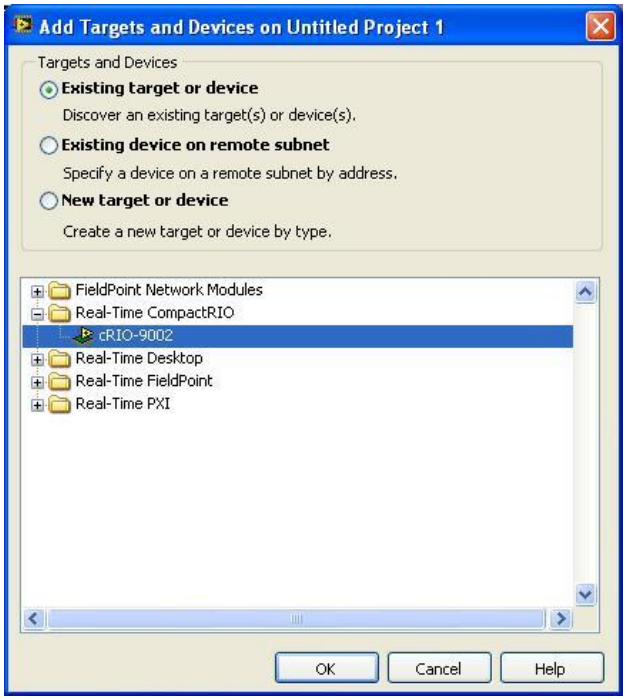


Figura 3.5. Selección del nuevo dispositivo, cRIO-9002.

Una vez que el controlador está en la estructura del proyecto, se agregan los módulos de entrada y salida analógica desplegando el menú contextual del FPGA Target y seleccionando el comando *New>C Series Modules*. Nuevamente se obtiene un menú para seleccionar entre los módulos conectados al chasís. Los módulos que se agregan para este proyecto son NI 9201, con 8 entradas analógicas de ±10 [V] y NI 9263, con 4 salidas analógicas de ±10 [V].

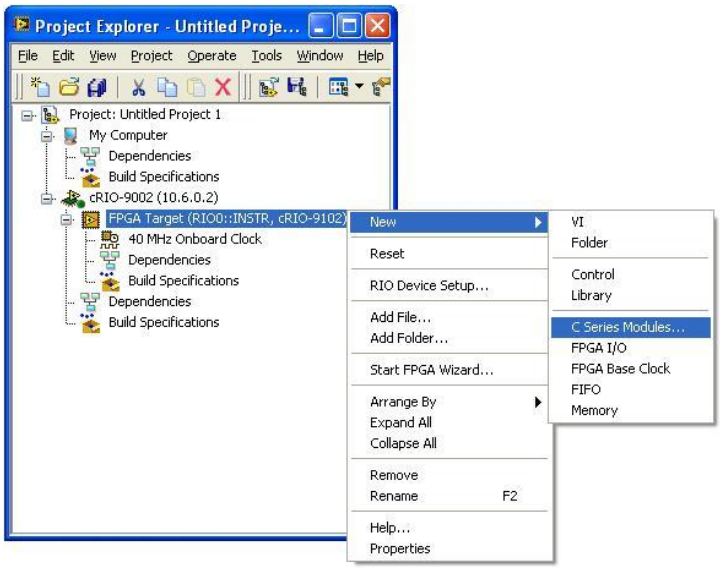


Figura 3.6. Agregando módulos de la serie C.

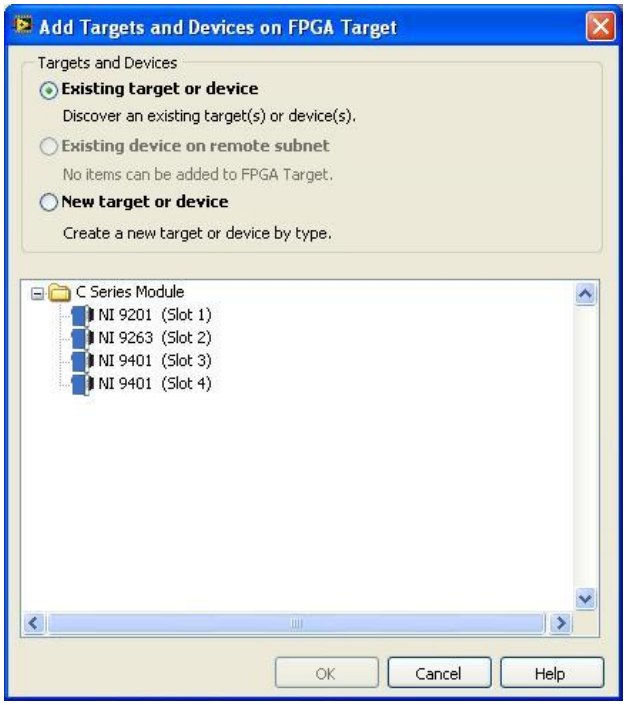


Figura 3.7. Selección del módulo NI 9201.

El módulo 9201 tiene ocho puertos de entrada de datos disponibles. A continuación se agregan las entradas que se emplearan para la ejecución del programa. Esto se hace llamando el menú contextual sobre el ícono del módulo 9201, seleccionando el comando New>FPGA I/O.

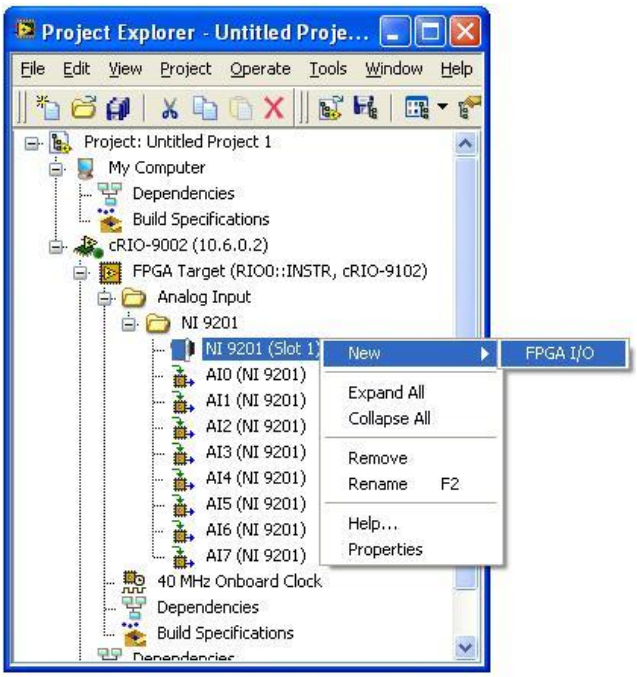


Figura 3.8. Agregando entradas analógicas al explorador de proyecto.

En la ventana siguiente se selecciona Analog Input (Entrada Analógica) y se expande el árbol para mostrar el módulo previamente agregado (NI 9201).

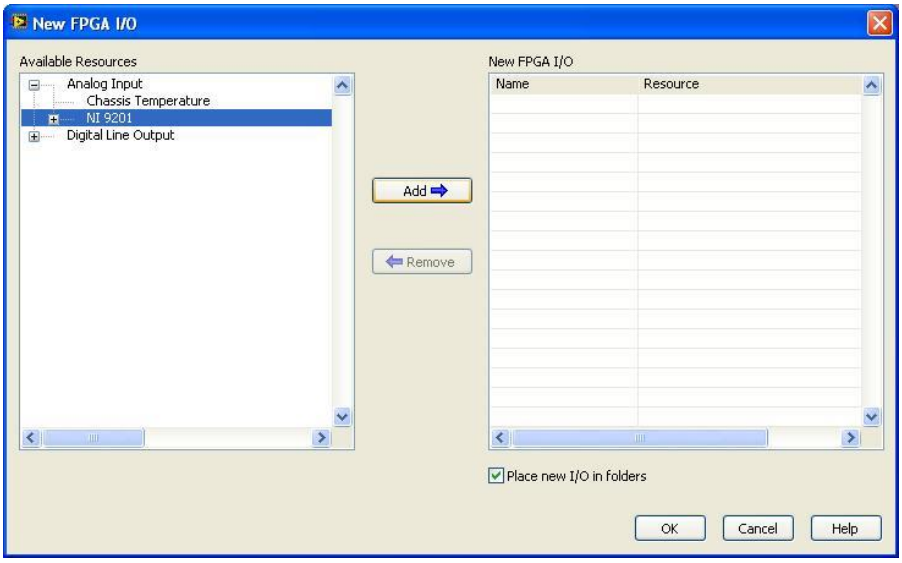


Figura 3.12. Selección del dispositivo de entrada analógica.

Se selecciona el módulo NI 9201 y click en Add para agregar todas las entradas que se tiene disponibles para ese módulo.

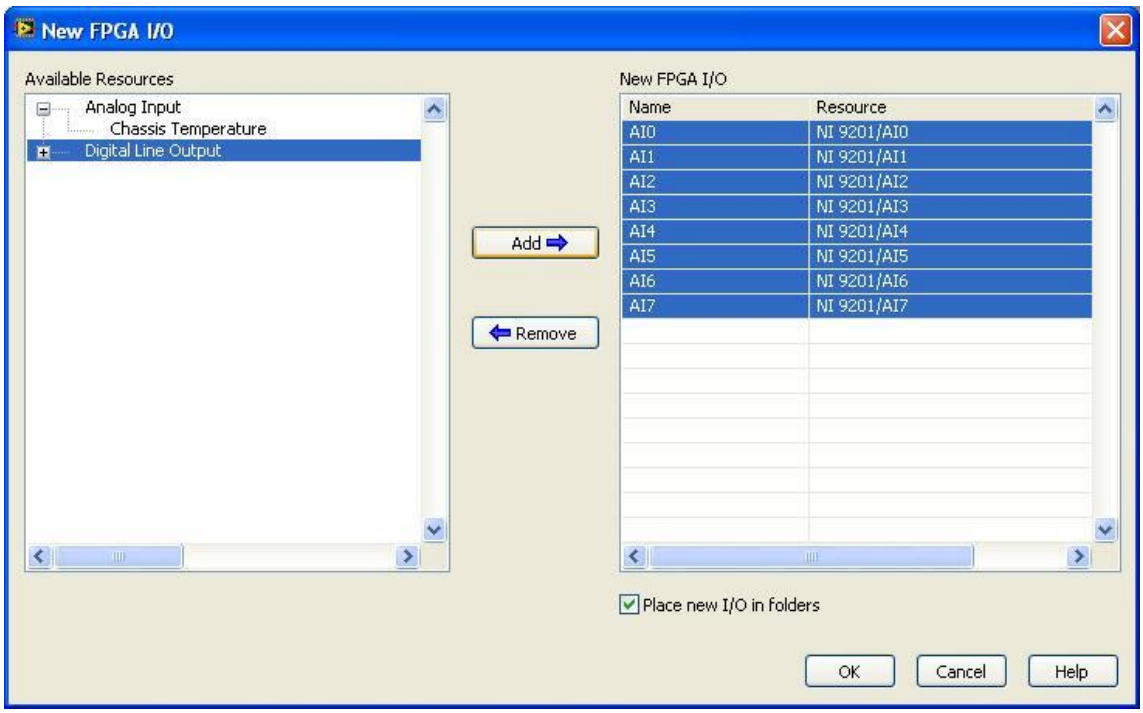


Figura 3.13. Selección de la entrada analógica.

Para agregar las salidas analógicas se realiza el mismo procedimiento que para la entrada selecionando el módulo NI 9263.

Capítulo 4
Desarrollo del control PID wavenet

El control PID wavenet está conformado por tres bloques funcionales principales: el controlador PID, la red neuronal wavenet y el algoritmo de sintonización. El controlador PID es un PID discreto que genera la señal de control que lleva la planta hasta la referencia, y reduce el efecto de perturbaciones en el ciclo de control principal. La red neuronal wavenet se encarga de aproximar localmente el comportamiento de la planta desconocida, y genera una señal de estimación de la salida de la planta. El algoritmo de sintonización, a través de un algoritmo de minimización del error, calcula los nuevos valores para las ganancias del controlador y los parámetros de la red neuronal. El desarrollo a continuación está basado en el algoritmo desarrollado en [2].

4.1 Algoritmos del control PID

Un control PID clásico actúa sobre la señal de error *e* aplicando tres acciones correctivas diferentes para generar una señal de control, como se muestra en la ecuación (4.1),

$$u(t) = k_p e(t) + k_I \int_0^t e(t)dt + k_D \frac{de(t)}{dt} \tag{4.1}$$

donde k_P, k_I y k_D son las ganancias proporcional, integral y derivativa, respectivamente. El término integral tiende a capturar la información de baja frecuencia y afecta el error en estado estacionario, mientras que el término derivativo responde a la información de alta frecuencia y afecta el estado transitorio de la señal de salida de la planta. En el caso del tiempo discreto, la ley de control PID se expresa como:

$$u(k) = k_p e(k) + k_I \sum_{i=0}^{k} e(k) + k_D[e(k) - e(k-1)] \tag{4.2}$$

La expresión integral de la ley de control del PID discreto se escribe como:

$$\Delta u(k) = u(k) - u(k-1) \tag{4.3}$$

Sustituyendo (4.2) en (4.3) se obtiene

$$\Delta u(k) = k_p e(k) + k_I \sum_{i=0}^{k} e(k) + k_D[e(k) - e(k-1)] - k_p e(k-1) - k_I \sum_{i=0}^{k-1} e(k-1) - k_D[e(k-1) - e(k-2)]$$

$$\Delta u(k) = k_p[e(k) - e(k-1)] + k_I e(k) + k_D[e(k) - 2e(k-1) + e(k-2)] \tag{4.4}$$

Finalmente

$$u(k) = u(k-1) + k_p[e(k) - e(k-1)] + k_I e(k) + k_D[e(k) - 2e(k-1) + e(k- \tag{4.5}$$

Es la ley de control que se implementa en el controlador.

4.2 Algoritmo wavenet

El controlador PID se auxilia de un algoritmo de sintonización basado en una red wavenet que proporciona el ajuste a los valores de cada ganancia, y permite compensar características no lineales y dinámicas desconocidas dentro de los procesos.

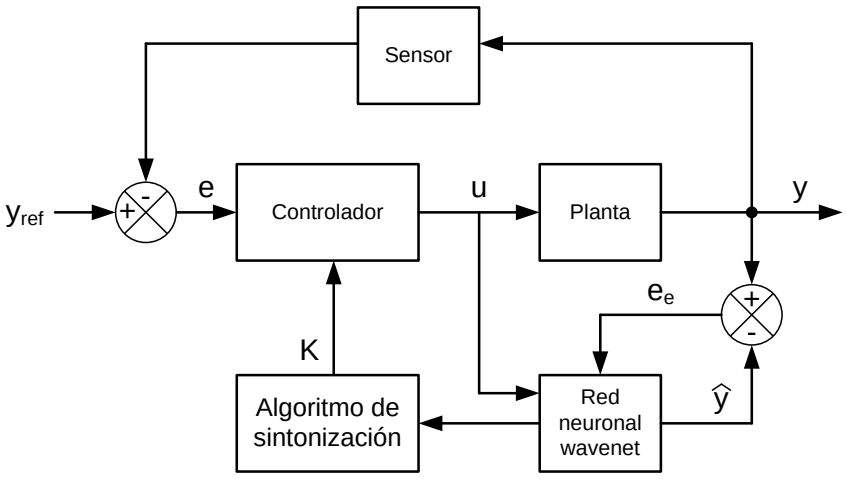


Figura 4.1. Diagrama de bloques del controlador PID wavenet.

Algoritmo de sintonización:

Del mismo modo que en el capítulo 2, se definen los siguientes vectores:

$$\mathbf{A}(k) \triangleq [a_1(k) \quad a_2(k) \quad \dots \quad a_l(k) \quad \dots \quad a_{L-1}(k) \quad a_L(k)]^T \tag{4.6}$$
$$\mathbf{B}(k) \triangleq [b_1(k) \quad b_2(k) \quad \dots \quad b_l(k) \quad \dots \quad b_{L-1}(k) \quad b_L(k)]^T \tag{4.7}$$
$$\mathbf{W}(k) \triangleq [w_1(k) \quad w_2(k) \quad \dots \quad w_l(k) \quad \dots \quad w_{L-1}(k) \quad w_L(k)]^T \tag{4.8}$$
$$\mathbf{\Psi}(k) \triangleq [\psi_1(\tau) \quad \psi_2(\tau) \quad \dots \quad \psi_l(\tau) \quad \dots \quad \psi_{L-1}(\tau) \quad \psi_L(\tau)]^T \tag{4.9}$$
$$\mathbf{C}(k) \triangleq [c_1(k) \quad c_2(k) \quad \dots \quad c_m(k) \quad \dots \quad c_{M-1}(k) \quad c_M(k)]^T \tag{4.10}$$
$$\mathbf{D}(k) \triangleq [d_1(k) \quad d_2(k) \quad \dots \quad d_j(k) \quad \dots \quad d_{J-1}(k) \quad d_J(k)]^T \tag{4.11}$$
$$\mathbf{Z}(k) \triangleq [z(k) \quad z(k-1) \quad \dots \quad z(k-m) \quad \dots \quad z(k-M+1) \quad z(k-M)]^T \tag{4.12}$$
$$\hat{\mathbf{Y}}(k) \triangleq [\hat{y}(k-1) \quad \hat{y}(k-2) \quad \dots \quad \hat{y}(k-j) \quad \dots \quad \hat{y}(k-J+1) \quad \hat{y}(k-J)]^T \tag{4.13}$$

Donde $\tau(k) = \frac{(k-b)}{a}$ y $\psi_l(\tau) = \psi\left(\frac{k-b_l(k)}{a_l(k)}\right)$ para $l = 1,2,\dots,L,\ L \in \mathbb{Z}$ es el número de neuronas en la capa de entrada de la red neuronal.

La función de estimación de la red neuronal está dada en (2.21) como:

$$\hat{y} = u(k)\mathbf{C}^T(k)\mathbf{Z}(k) + v(k)\mathbf{D}^T(k)\hat{\mathbf{Y}}(k).$$

La ley de control $u(k)$ está dada en (4.5), y $z(k)$ está dada en (2.20) como:

$$z(k) = \mathbf{\Psi}^T(\tau)\mathbf{W}(k).$$

Para minimizar el error se emplea un algoritmo de mínimos cuadrados medios, estableciendo una función de medida del error (que es una función de energía). Por medio del método del gradiente de pasos descendentes se minimiza el tamaño de este error actualizando las variables $\mathbf{A}(k)$, $\mathbf{B}(k)$, $\mathbf{W}(k)$, $\mathbf{C}(k)$ y $\mathbf{D}(k)$ de la red wavenet y los coeficientes k_p, k_i y k_d del controlador.

Se define la siguiente función de medida del error entre la señal de referencia y_{ref} y la señal de salida de la red wavenet $\hat{y}$:

$$E_{est} = \frac{1}{2}(y_{ref} - \hat{y})^2 \tag{4.14}$$

La sintonización del controlador PID wavenet requiere el cálculo de los gradientes de (2.22). Los coeficientes del controlador se actualizan de la misma manera. Definimos el vector:

$$\mathbf{K}(k) = [k_p \quad k_i \quad k_d]^T \tag{4.15}$$

y la expresión para la regla de aprendizaje de $\mathbf{K}$ es:

$$\mathbf{K}(k+1) = \mathbf{K}(k) + \mu_{\mathbf{K}}\Delta\mathbf{K} \tag{4.16}$$

Como se describió en la ecuación (2.22) y (4.16), para los obtener vectores actualizados $\mathbf{A}(k+1)$, $\mathbf{B}(k+1)$, $\mathbf{W}(k+1)$, $\mathbf{C}(k)+1$, $\mathbf{D}(k+1)$ y $\mathbf{K}(k+1)$ se requiere conocer la variación del error respecto a cada una de las variables. Estas variaciones se representan a continuación. El signo menos en la derivada parcial se introduce siguiendo el método explicado en la sección 2.3.

$$\Delta\mathbf{W}(k) = -\frac{\partial E_{est}}{\partial \mathbf{W}(k)} = \left[\frac{\partial E_{est}}{\partial w_1} \quad \frac{\partial E_{est}}{\partial w_2} \quad \cdots \quad \frac{\partial E_{est}}{\partial w_l}\right] \tag{4.17}$$

$$\Delta\mathbf{B}(k) = -\frac{\partial E_{est}}{\partial \mathbf{B}(k)} = \left[\frac{\partial E_{est}}{\partial b_1} \quad \frac{\partial E_{est}}{\partial b_2} \quad \cdots \quad \frac{\partial E_{est}}{\partial b_l}\right] \tag{4.18}$$

$$\Delta\mathbf{A}(k) = -\frac{\partial E_{est}}{\partial \mathbf{A}(k)} = \left[\frac{\partial E_{est}}{\partial b_1} \quad \frac{\partial E_{est}}{\partial b_2} \quad \cdots \quad \frac{\partial E_{est}}{\partial b_l}\right] \tag{4.19}$$

$$\Delta\mathbf{C}(k) = -\frac{\partial E_{est}}{\partial \mathbf{C}(k)} = \left[\frac{\partial E_{est}}{\partial b_1} \quad \frac{\partial E_{est}}{\partial b_2} \quad \cdots \quad \frac{\partial E_{est}}{\partial b_l}\right] \tag{4.20}$$

$$\Delta\mathbf{D}(k) = -\frac{\partial E_{est}}{\partial \mathbf{D}(k)} = \left[\frac{\partial E_{est}}{\partial b_1} \quad \frac{\partial E_{est}}{\partial b_2} \quad \cdots \quad \frac{\partial E_{est}}{\partial b_l}\right] \tag{4.21}$$

$$\Delta\mathbf{K}(k) = -\frac{\partial E_{est}}{\partial \mathbf{K}(k)} = \left[\frac{\partial E_{est}}{\partial b_1} \quad \frac{\partial E_{est}}{\partial b_2} \quad \cdots \quad \frac{\partial E_{est}}{\partial b_l}\right] \tag{4.22}$$

Cálculo de la variación $\Delta\mathbf{W}(k)$:

Por la regla de la cadena

$$\frac{\partial E_{est}}{\partial w_l} = \left(\frac{\partial E_{est}}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial w_l}\right) \tag{4.23}$$

Se obtiene la derivada parcial

$$\frac{\partial E_{est}}{\partial \hat{y}} = -\left(y_{ref} - \hat{y}\right) \tag{4.24}$$

Sustituyendo (4.24) y (2.21) en (4.23)

$$\frac{\partial E_{est}}{\partial w_l} = -\left(y_{ref} - \hat{y}\right)\frac{\partial}{\partial w_l}\left(u\mathbf{C}^T\mathbf{Z} + v\mathbf{D}^T\hat{\mathbf{Y}}\right) \tag{4.25}$$

Como únicamente $\mathbf{Z}$ depende de $\mathbf{W}$

$$\frac{\partial E_{est}}{\partial w_l} = -\left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \frac{\partial}{\partial w_l} \mathbf{Z} \tag{4.26}$$

Si

$$\frac{\partial}{\partial w_l} \mathbf{Z} = \left[\frac{\partial z(k)}{\partial w_l} \quad \frac{\partial z(k-1)}{\partial w_l} \quad \dots \quad \frac{\partial z(k-m)}{\partial w_l} \quad \dots \quad \frac{\partial z(k-M+1)}{\partial w_l} \quad \frac{\partial z(k-M)}{\partial w_l}\right]^T \tag{4.27}$$

derivando (2.20) respecto a w_l se obtiene

$$\frac{\partial z(k)}{\partial w_l} = \psi_l(k) \tag{4.28}$$

Se define un vector de wavelets pasadas $\mathbf{\Psi}_l$:

$$= [\psi_l(k) \quad \psi_l(k-1) \quad \dots \quad \psi_l(k-m) \quad \dots \quad \psi_l(k-M)]^T \tag{4.29}$$

La ecuación del gradiente para cada w_l resulta:

$$\frac{\partial E_{est}}{\partial w_l} = -\left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \mathbf{\Psi}_l \tag{4.30}$$

Cálculo de la variación del vector de traslaciones, $\Delta\mathbf{B}$:

$$\Delta \mathbf{B}(k) = -\frac{\partial E_{est}}{\partial \mathbf{B}(k)} \tag{4.31}$$

Por la regla de la cadena

$$\frac{\partial E_{est}}{\partial b_l} = \left(\frac{\partial E_{est}}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial b_l}\right) \tag{4.32}$$

Sustituyendo (4.24) y (2.21) en (4.32) se obtiene la siguiente expresión:

$$\frac{\partial E_{est}}{\partial b_l} = -\left(y_{ref} - \hat{y}\right) \frac{\partial}{\partial b_l}\left(u \mathbf{C}^T \mathbf{Z} + v \mathbf{D}^T \hat{\mathbf{Y}}\right) \tag{4.33}$$

Como únicamente $\mathbf{Z}$ depende de $\mathbf{B}$, se simplifica la ecuación

$$\frac{\partial E_{est}}{\partial b_l} = -\left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \frac{\partial}{\partial b_l} \mathbf{Z} \tag{4.34}$$

sea

$$\frac{\partial}{\partial b_l} \mathbf{Z} = \left[\frac{\partial z(k)}{\partial b_l} \quad \frac{\partial z(k-1)}{\partial b_l} \quad \cdots \quad \frac{\partial z(k-m)}{\partial b_l} \quad \cdots \quad \frac{\partial z(k-M)}{\partial b_l}\right]^T \tag{4.35}$$

derivando (2.20) respecto a b_l se obtiene

$$\frac{\partial z(k)}{\partial b_l} = \frac{\partial}{\partial b_l} \mathbf{\Psi}(k)^T \mathbf{W}(k) \tag{4.36}$$

$$\frac{\partial z(k)}{\partial b_l} = \frac{\partial \psi_l(k)}{\partial b_l} w_l(k) \tag{4.37}$$

Se define el siguiente vector auxiliar

$$\mathbf{\Psi}_{bl} = \left[\frac{\partial \psi_l(k)}{\partial b_l} \quad \frac{\partial \psi_l(k-1)}{\partial b_l} \quad \cdots \quad \frac{\partial \psi_l(k-m)}{\partial b_l} \quad \cdots \quad \frac{\partial \psi_l(k-M)}{\partial b_l}\right]^T \tag{4.38}$$

La ecuación del gradiente para cada b_l resulta:

$$\frac{\partial E_{est}}{\partial b_l} = -\left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \mathbf{\Psi}_{bl}(k) w_l(k) \tag{4.39}$$

Obtención de la variación del vector de dilataciones A, $\Delta\mathbf{A}(k)$:

Aplicando la regla de la cadena

$$\frac{\partial E_{est}}{\partial a_l} = \left(\frac{\partial E_{est}}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial a_l}\right) \tag{4.40}$$

Sustituyendo (4.24) y (2.21) en (4.40)

$$\frac{\partial E_{est}}{\partial a_l} = -\left(y_{ref} - \hat{y}\right) \frac{\partial}{\partial a_l}\left(u \mathbf{C}^T \mathbf{Z} + v \mathbf{D}^T \hat{\mathbf{Y}}\right) \tag{4.41}$$

Como únicamente $\mathbf{Z}$ depende de $\mathbf{A}$, se simplifica la ecuación

$$\frac{\partial E_{est}}{\partial a_l} = -\left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \frac{\partial}{\partial a_l} \mathbf{Z} \tag{4.42}$$

Sea

$$\frac{\partial}{\partial a_l} \mathbf{Z} = \left[\frac{\partial z(k)}{\partial a_l} \quad \frac{\partial z(k-1)}{\partial a_l} \quad \cdots \quad \frac{\partial z(k-m)}{\partial a_l} \quad \cdots \quad \frac{\partial z(k-M)}{\partial a_l}\right]^T \tag{4.43}$$

La derivada parcial de (2.20) respecto a a_l resulta

$$\frac{\partial z(k)}{\partial a_l} = \frac{\partial}{\partial a_l} \mathbf{\Psi}(k)^T \mathbf{W}(k) \tag{4.44}$$

$$\frac{\partial z(k)}{\partial a_l} = \frac{\partial \psi_l(k)}{\partial a_l} w_l(k) \tag{4.45}$$

Además, se conoce que

$$\frac{\partial \psi_l(k)}{\partial a_l} = \tau_l \frac{\partial \psi_l(k)}{\partial b_l} \tag{4.46}$$

Sustituyendo (4.46) en (4.45)

$$\frac{\partial z(k)}{\partial a_l} = \tau_l \frac{\partial \psi_l(k)}{\partial b_l} w_l(k) \tag{4.47}$$

Por lo tanto

$$\frac{\partial}{\partial a_l} \mathbf{Z} = \tau_l \frac{\partial}{\partial b_l} \mathbf{Z} \tag{4.48}$$

Y

$$\frac{\partial E_{est}}{\partial a_l} = -\left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \tau_l \frac{\partial}{\partial b_l} \mathbf{Z} \tag{4.49}$$

De este modo, la ecuación del gradiente para cada a_l resulta:

$$\frac{\partial E_{est}}{\partial a_l} = -\tau_l \left(y_{ref} - \hat{y}\right) u \mathbf{C}^T \mathbf{\Psi}_{bl}(k) w_l(k) \tag{4.50}$$

O, lo que es lo mismo:

$$\frac{\partial E_{est}}{\partial a_l} = \tau_l \frac{\partial E_{est}}{\partial b_l} \tag{4.51}$$

Cálculo de la variación $\Delta\mathbf{C}(k)$ del vector C, que se compone por los elementos $\frac{\partial E_{est}}{\partial c_m}$.

$$\frac{\partial E_{est}}{\partial c_m} = \left(\frac{\partial E_{est}}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial c_m}\right) \tag{4.52}$$

Sustituyendo (4.24) y (2.21) en (4.52) se obtiene

$$\frac{\partial E_{est}}{\partial c_m} = -\left(y_{ref} - \hat{y}\right)\frac{\partial}{\partial c_m}\left(u\mathbf{C}^T\mathbf{Z} + v\mathbf{D}^T\hat{\mathbf{Y}}\right) \tag{4.53}$$

Como únicamente el término $u\mathbf{C}^T\mathbf{Z}$ depende de $\mathbf{C}$, la ecuación se simplifica

$$\frac{\partial E_{est}}{\partial c_m} = -\left(y_{ref} - \hat{y}\right)u\left(\frac{\partial}{\partial c_m}\mathbf{C}^T\right)\mathbf{Z} \tag{4.54}$$

La ecuación del gradiente para cada c_m resulta:

$$\frac{\partial E_{est}}{\partial c_m} = -\left(y_{ref} - \hat{y}\right)uz(k-m) \tag{4.55}$$

Cálculo de la variación $\Delta\mathbf{D}(k)$del vector D, que se define como:

$$\Delta\mathbf{D}(k) = -\frac{\partial E_{est}}{\partial \mathbf{D}(k)}, \tag{4.56}$$

formado por los elementos $\frac{\partial E_{est}}{\partial d_j}$.

$$\frac{\partial E_{est}}{\partial d_j} = \left(\frac{\partial E_{est}}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial d_j}\right) \tag{4.57}$$

Sustituyendo (4.24) y (2.21) en (4.57)

$$\frac{\partial E_{est}}{\partial d_j} = -\left(y_{ref} - \hat{y}\right)\frac{\partial}{\partial d_j}\left(u\mathbf{C}^T\mathbf{Z} + v\mathbf{D}^T\hat{\mathbf{Y}}\right) \tag{4.58}$$

Como únicamente el término $v\mathbf{D}^T\hat{\mathbf{Y}}$ depende de $\mathbf{D}$, la ecuación se simplifica

$$\frac{\partial E_{est}}{\partial d_j} = -(y_{ref} - \hat{y})v\left(\frac{\partial}{\partial d_j}\mathbf{D}^T\right)\hat{\mathbf{Y}} \tag{4.59}$$

La ecuación del gradiente para cada d_j resulta:

$$\frac{\partial E_{est}}{\partial d_j} = -(y_{ref} - \hat{y})v\hat{y}(k-j) \tag{4.60}$$

Cálculo de la variación $\Delta\mathbf{K}(k)$ del vector de ganancias:

$$\Delta\mathbf{K}(k) = -\frac{\partial E_{est}}{\partial\mathbf{K}(k)} \tag{4.61}$$

compuesto por los elementos $\frac{\partial E_{est}}{\partial k_i}$.

$$\frac{\partial E_{est}}{\partial k_i} = \left(\frac{\partial E_{est}}{\partial \hat{y}}\right)\left(\frac{\partial \hat{y}}{\partial k_i}\right) \tag{4.62}$$

Sustituyendo (4.24) y (2.21) en (4.62)

$$\frac{\partial E_{est}}{\partial k_i} = -(y_{ref} - \hat{y})\frac{\partial}{\partial k_i}(u\mathbf{C}^T\mathbf{Z} + v\mathbf{D}^T\hat{\mathbf{Y}}) \tag{4.63}$$

Ya que únicamente el término $u(k)$ depende de $\mathbf{K}$, la ecuación se simplifica y el gradiente para cada ∂k_i resulta:

$$\frac{\partial E_{est}}{\partial k_i} = -(y_{ref} - \hat{y})\frac{\partial u}{\partial k_i}\mathbf{C}^T\mathbf{Z} \tag{4.64}$$

Capítulo 5
Programación del algoritmo en LabVIEW

En este capítulo se describen las funciones que se emplearon para implementar el control PID wavenet en LabVIEW. El algoritmo wavenet fue implementado en LabVIEW para efectuar simulaciones numéricas a nivel de PC en Windows. Posteriormente, se desarrolla la programación de las funciones de adquisición y salida de datos del módulo CompactRIO y el código se implementó en el FPGA para efectuar pruebas con el simulador de procesos PCS327. Los algoritmos programados son los obtenidos en el capítulo anterior para el controlador PID y la red wavenet.

5.1 Programación del algoritmo y código en LabVIEW

La implementación del controlador PID Wavenet se realiza empleando tres programas intercomunicados. El programa básico controla la adquisición y emisión de datos en el FPGA. Un segundo programa, ejecutado en el RT Host, es el propio PID wavenet y se encarga de generar la señal de control que requiere la planta. El tercer programa se ejecuta en Windows, y se encarga de las funciones de despliegue y almacenamiento de los datos generados durante la ejecución. Estos tres programas funcionan simultáneamente, y se comunican mediante el uso de nodos de entrada/salida y variables compartidas.

5.1.1. Programación en el FPGA cRIO-9102

La implementación del programa requiere comunicación con los módulos de entrada y salida del módulo cRIO-9102 para la lectura y envío de datos. El programa que se describe a continuación obtiene los datos de calibración de las tarjetas, los cuales se usan para

hacer las conversiones de valores binarios a valores nominales de voltaje requeridos para el control, como el primer paso en la programación del código.

El programa emplea una estructura tipo secuencia en que primero se declara que no se tienen los datos de calibración listos, después se obtienen los datos de calibración con un FPGA I/O Property node y finalmente se avisa al programa de control que ya se tienen los datos listos.

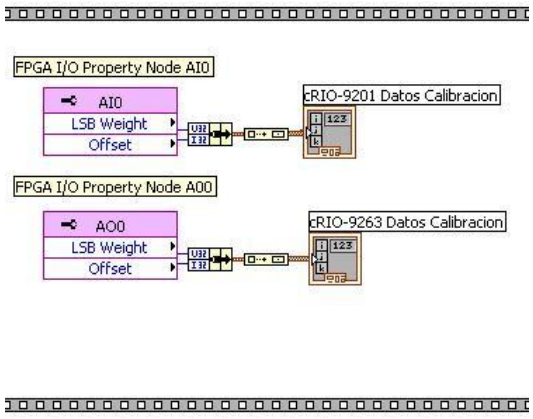


Figura 5.1. Nodos de propiedad FPGA I/O.

El segundo paso en la secuencia es el ciclo de lectura y escritura de datos. Dentro de un ciclo while se introduce una nueva secuencia de tres pasos: un retraso de 10 [ns], las funciones de lectura y escritura y una interrupción que reinicia el ciclo. El retraso se agrega como un elemento de seguridad. En este programa se emplea una entrada analógica de datos, la AI0 del módulo 9201, que envía la lectura del estado de la planta al programa de control. La señal de control generada por el programa de control se transmite a la planta a través de la salida analógica AO0 del módulo 9263. El último elemento del ciclo es una interrupción, empleada para avisar al programa de control que los datos nuevos están disponibles, que se envían los que se tenían, y se declara la condición “ready” para el envío y recepción de una nueva pareja de datos.

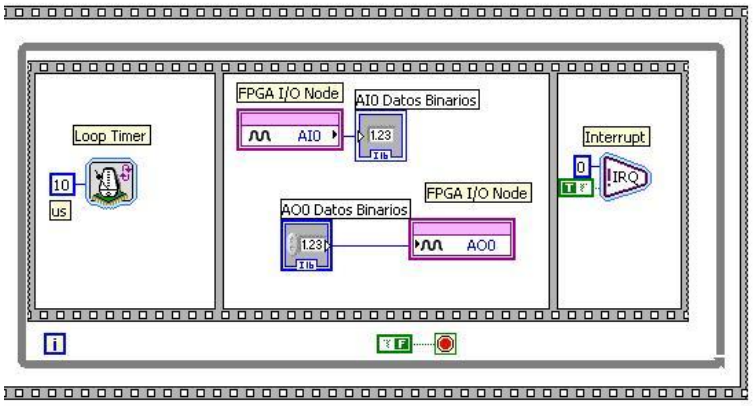


Figura 5.2. Ciclo de lectura y escritura de datos.

Este programa se compila y posteriormente se carga en el FPGA.

5.1.2. Programación en el HOST cRIO 9002

La rutina principal de control se ejecuta en el HOST cRIO9002. De inicio se requiere una etapa de comunicación con las variables que se leen y escriben en el FPGA, así como variables compartidas entre el HOST y la interfaz con el usuario en Windows. Este VI contiene también los algoritmos de control expuestos en el Capítulo 4, cuya programación es explicada a continuación.

En el diagrama de bloques del programa se crea una referencia al VI en el FPGA, definiendo el archivo de bits usado para determinar los controles disponibles en el FPGA y el dispositivo que se va a usar como execution target. La referencia a un VI debe cerrarse forzosamente para liberar recursos de hardware y evitar errores en la implementación de un programa posterior. La función Close FPGA VI Reference detiene el VI que corre en el dispositivo RIO y limpia la memoria usada por la interfaz del FPGA.

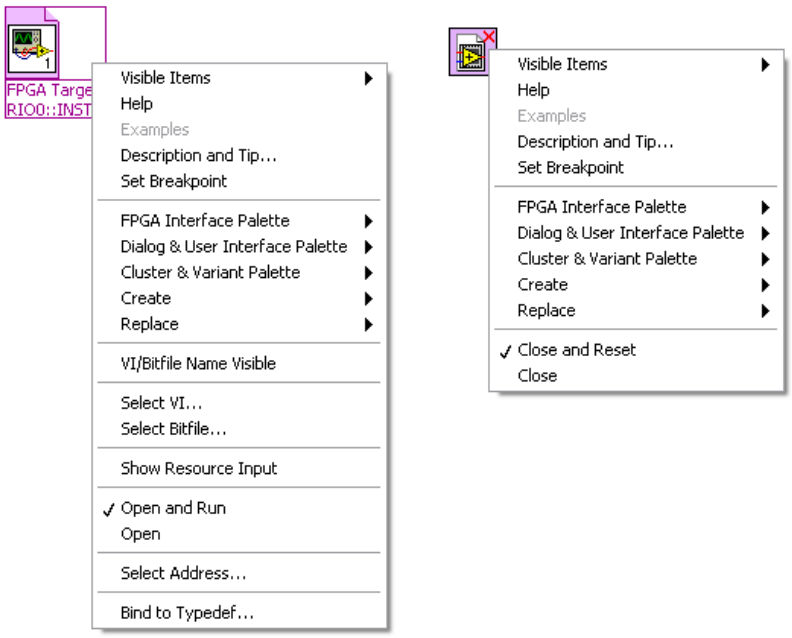


Figura 5.3. Open FPGA VI Reference y Close FPGA VI Reference.

La manipulación de datos para generar la señal de control y comunicarla al FPGA requiere una función de lectura/escritura: Read/Write Control (FPGA Interface).

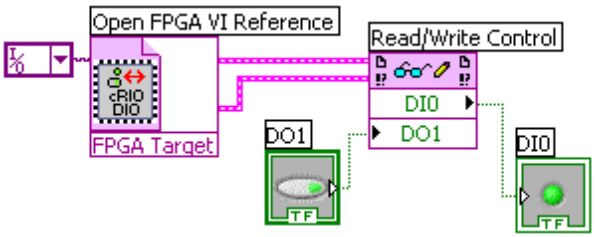
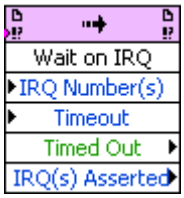
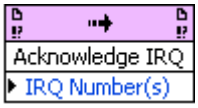


Figura 5.4. Read/Write Control.

El ciclo de control emplea una interrupción para sincronizar el VI del FPGA con el VI del HOST. La función Invoke Node (FPGA Interface), configurada en Wait on IRQ, espera el número de interrupción establecido en el VI del FPGA (o genera una señal de error si se excede el tiempo establecido de espera). Esta función avisa cuando recibe la interrupción y después envía una señal de aviso. Posteriormente se manda un reconocimiento por medio de otro Invoke Node configurado en Acknowledge IRQ para avisar a la fuente de interrupciones que se ha terminado el procesamiento de los datos actuales y que el VI de control en el HOST está listo para recibir un nuevo bloque de información.





Wait on IRQ Acknowledge IRQ

Figura 5.5. Funciones de interrupción.

El módulo cRIO 9201 entrega valores binarios tras efectuar una lectura de voltaje. Antes de procesar los datos es necesario convertirlos en valores nominales de voltaje. La conversión se efectúa por medio de un VI que recibe como parámetros de entrada el modelo de la tarjeta 9201, los datos de calibración de la tarjeta, y el valor binario proveniente de la misma; la salida es el valor nominal de voltaje que entra al control PID como variable de proceso. La Figura 5.6 muestra la ubicación de este VI en el diagrama de bloques.

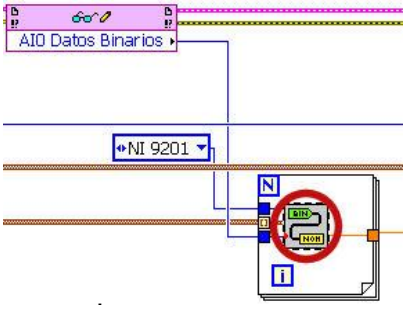


Figura 5.6. VI Binary to Nominal.vi en el ciclo de control.

De manera análoga, el módulo de salida cRIO 9263 requiere sean enviados valores binarios, por lo que es necesario convertir los valores nominales de voltaje a valores binarios. Para esto se incluye el subVI Nominal to Binary.vi que recibe como parámetros de entrada el modelo del módulo al cual se van a mandar estos datos, los datos de calibración del módulo, y el valor nominal de voltaje que se quiere convertir. La Figura 5.7 contiene la forma de incluir este VI en el ciclo de control.

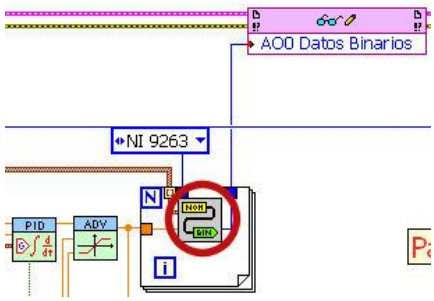


Figura 5.7. VI Binary to Nominal.vi en el ciclo de control.

El valor de voltaje obtenido de la planta se utiliza en la ley de control descrita en (4.5), que se implementa como se muestra en la Figura 5.8. La señal leída del voltaje de la planta se compara con la función de referencia (definida dentro de este mismo programa) para generar la señal de error con que actúa el PID. Se emplean nodos de corrimiento en la estructura iterativa del programa para contener los valores pasados del error de seguimiento y de la entrada a la planta.

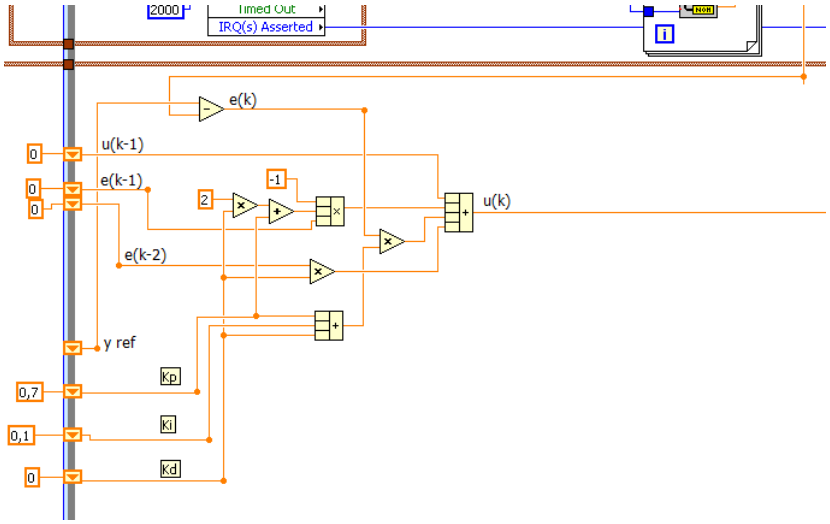


Figura 5.8. Implementación de la ley de control en el VI.

Se crea una variable compartida del tipo arreglo de 1-D en la estructura del proyecto, llamada Variables Control, cuya función es almacenar los valores de parámetros y variables de control para la comunicación con la interfaz de usuario. Este arreglo contiene la iteración (tick) del ciclo de control, el valor de referencia, la variable del proceso, la señal de control, una señal que indica si el ciclo de control está trabajando dentro del ciclo de 5 [ms] asignado a la tarea, el valor de la estimación de la red wavenet y las ganancias del controlador PID.

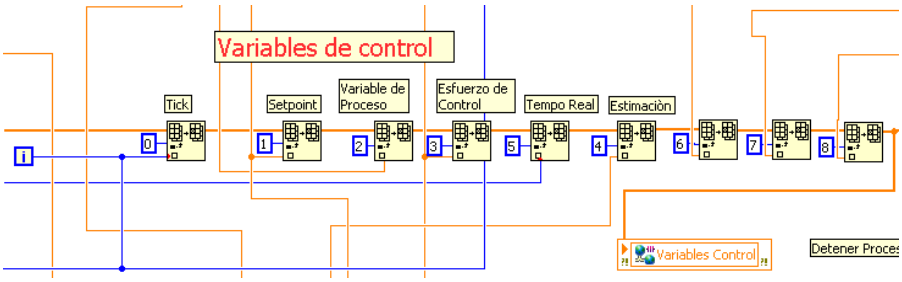


Figura 5.9. Variables de control en el programa en el FPGA.

La red wavenet empleada para actualizar los parámetros del controlador recibe la lectura del voltaje del proceso como una de las variables de entrada que requiere para la sintonización. La red wavenet requiere los vectores A,B y W característicos de la red wavenet, el vector Ψ y su derivada, dos parámetros C y D del filtro IIR y la salida anterior de la wavenet z[k-1]. Todo esto se incluye dentro de un ciclo FOR con un control numérico para elegir la cantidad de épocas de aprendizaje que realiza la red.

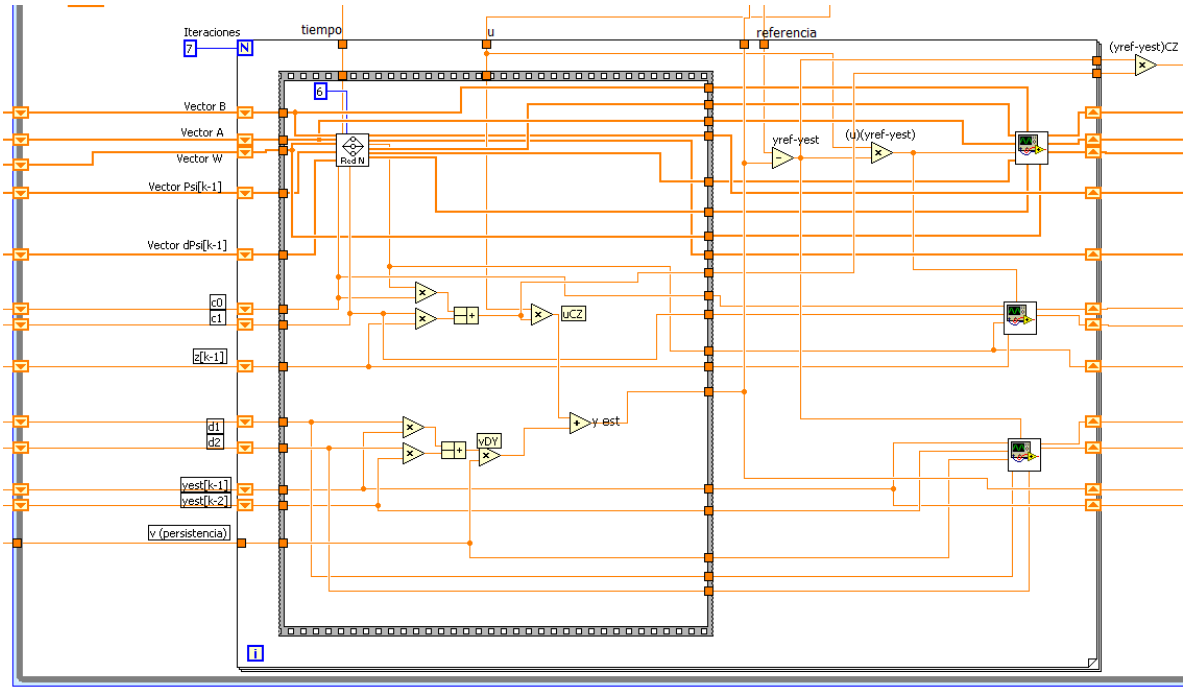


Figura 5.10. Implementación del algoritmo wavenet.

Dentro de este ciclo se incluye un subVI que incluye la red, y varias estructuras que forman parte del algoritmo de aprendizaje. La Figura 5.11 muestra el código de la red wavenet. Se utiliza un ciclo FOR, en el cual las iteraciones están determinadas por la cantidad de neuronas, para calcular la aproximación preliminar z[k] y cada valor de los vectores auxiliares del algoritmo de actualización de la red neuronal. Dentro de este ciclo

FOR existe un subprograma que calcula el valor de cada señal wavelet y su derivada parcial respecto a b.

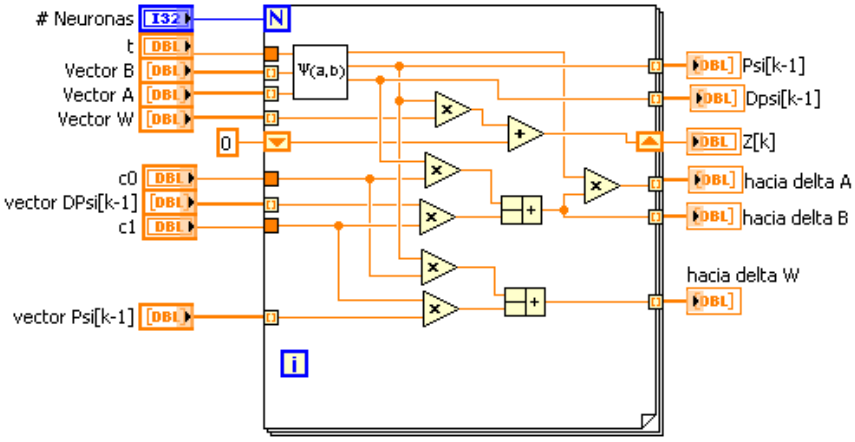


Figura 5.11. SubVI Red wavenet.

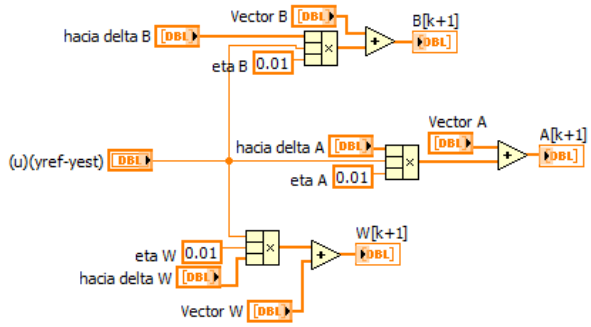


Figura 5.12. Subrutina de aprendizaje de los vectores A, B y W de la red neuronal.

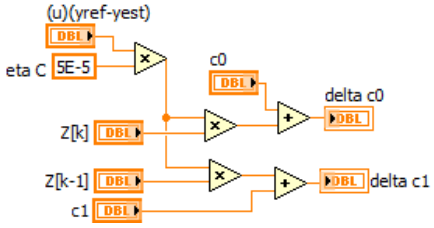


Figura 5.13. Subrutina de aprendizaje de los parámetros C del filtro IIR.

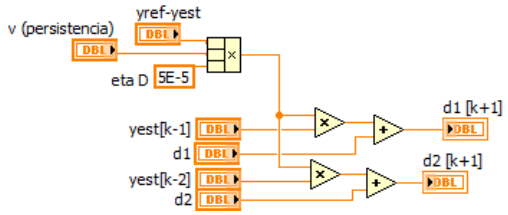


Figura 5.14. Subrutina de aprendizaje de los parámetros D del filtro IIR.

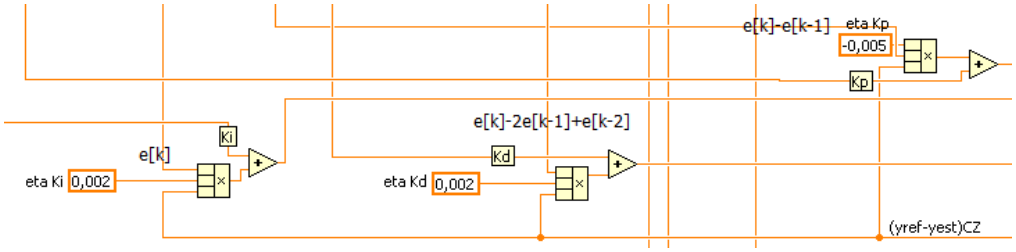


Figura 5.15. Subrutina de aprendizaje de las ganancias del control PID.

El diagrama de bloques del código completo se anexa a continuación. Se destacan las tres estructuras principales: el control PID descrito en primer lugar, la etapa de comunicación de datos por medio de variables compartidas y la red neuronal wavenet. La combinación de estos tres elementos es la implementación del algoritmo descrito en el Capítulo 4.

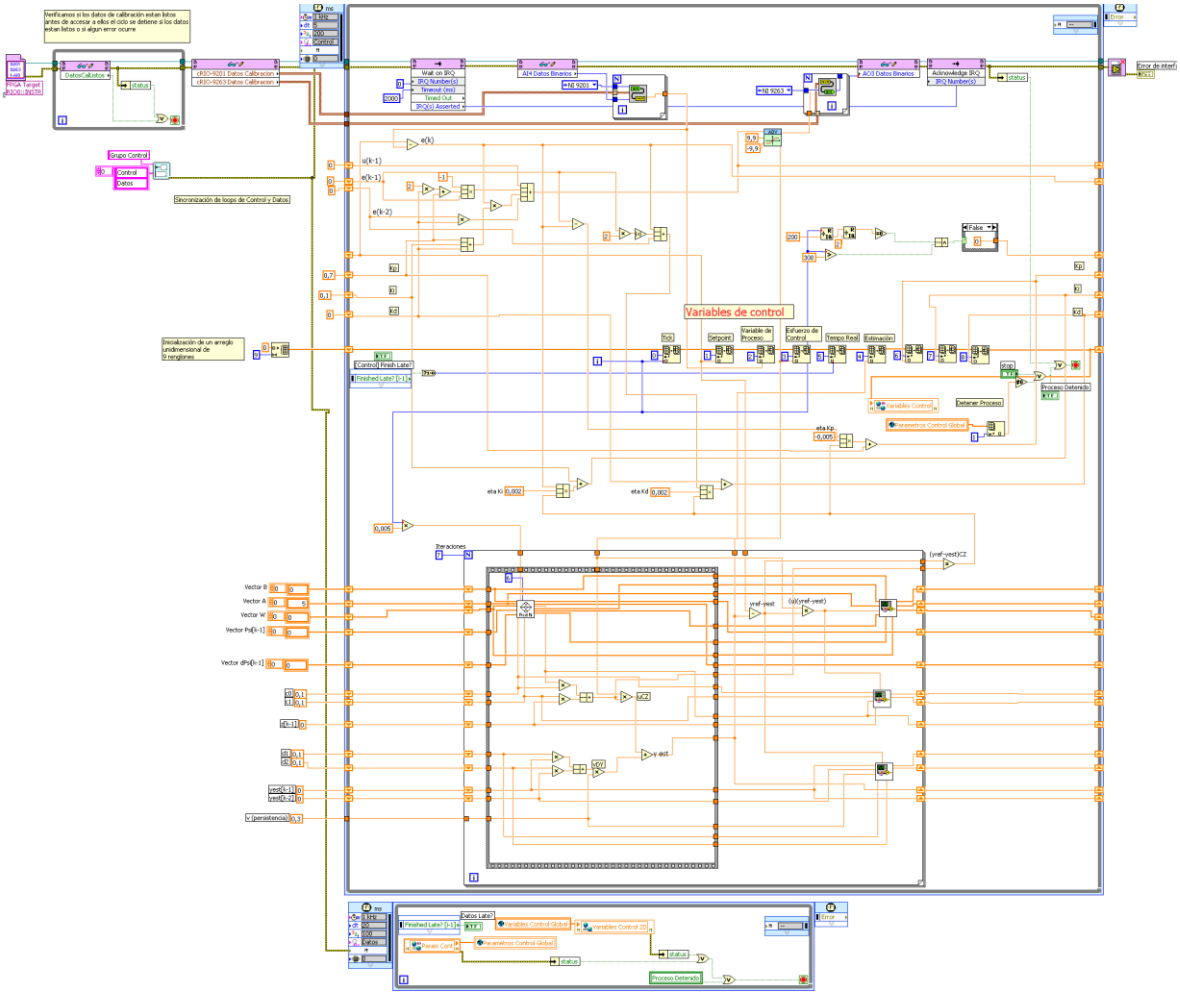


Figura 5.16. Diagrama de bloques del VI de control en el HOST cRIO.

5.1.3 Programación de la interfaz de usuario

La interfaz con el usuario se ejecuta en Windows. Cumple tres propósitos principales: recibir del usuario la referencia, desplegar los datos obtenidos y escribirlos en un archivo de texto. La Figura 5.17 muestra el panel de usuario, provisto de un control numérico para cambiar la señal de referencia:

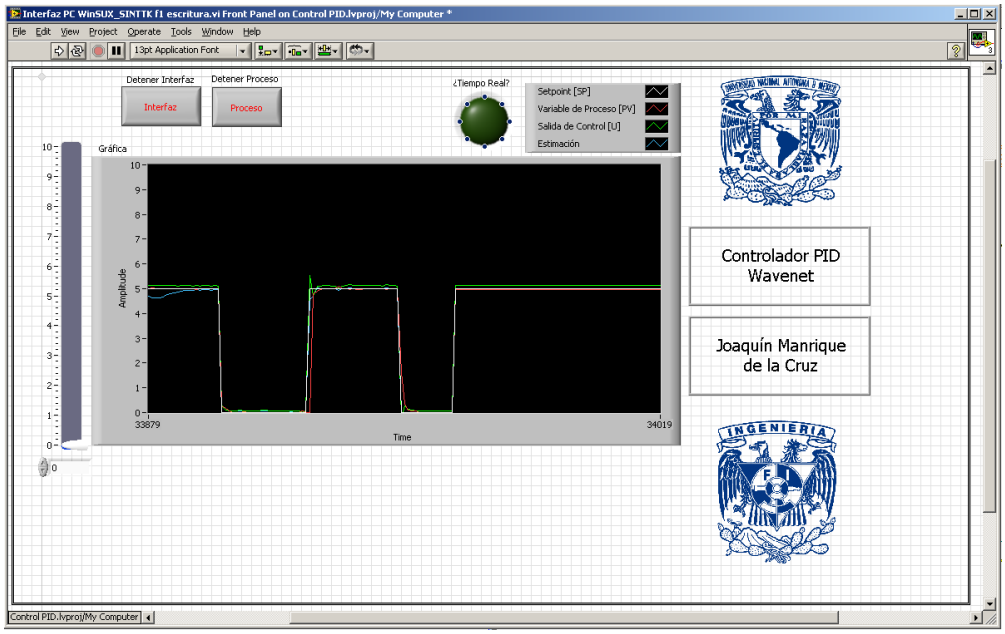


Figura 5.17. Panel de la interfaz de usuario.

Esta VI recibe tres entradas diferentes del usuario: la señal de referencia, la señal de paro del controlador en el HOST cRIO y la señal de paro de la interfaz. La señal de referencia y la señal de paro del controlador se transmiten por medio de variables compartidas. Del VI en el HOST cRIO se leen la señal de control, la variable del proceso y la señal generada por la wavenet, todas para su escritura en un archivo de texto, del cual posteriormente se reconstruye la gráfica de comportamiento del sistema. Aunque este programa tiene la capacidad de que el usuario controle la variable de referencia, en las pruebas se deshabilita esta opción y la señal de referencia se genera directamente en el FPGA.

Se utiliza la variable global Param Cont para transmitir dos señales al programa en el FPGA: señal de referencia y el comando de detener el proceso.

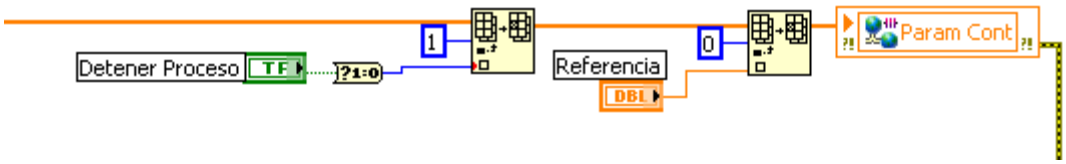


Figura 5.18. Parámetros de la interfaz de usuario.

El programa lee la variable compartida Variables Control, que contiene los valores de la señal de referencia, la lectura del proceso, la señal de control, la estimación de la wavenet y la señal de confirmación de tiempo real. Estas variables, generadas por el FPGA se despliegan en una gráfica en el panel frontal de la aplicación en windows.

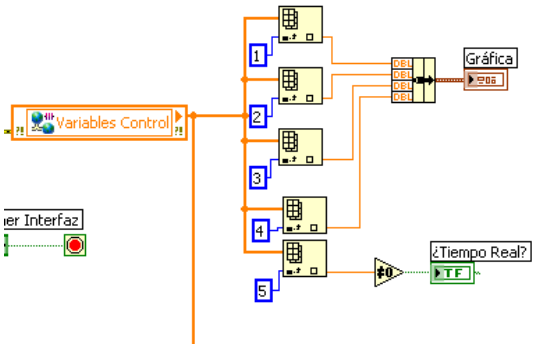


Figura 5.19. Lectura y despliegue de datos del FPGA.

Las lecturas de esta variable se almacenan en un archivo de texto. Las gráficas del comportamiento del controlador, que se presentan el capítulo siguiente, se reconstruyeron en Matlab a partir de estos archivos .TXT.

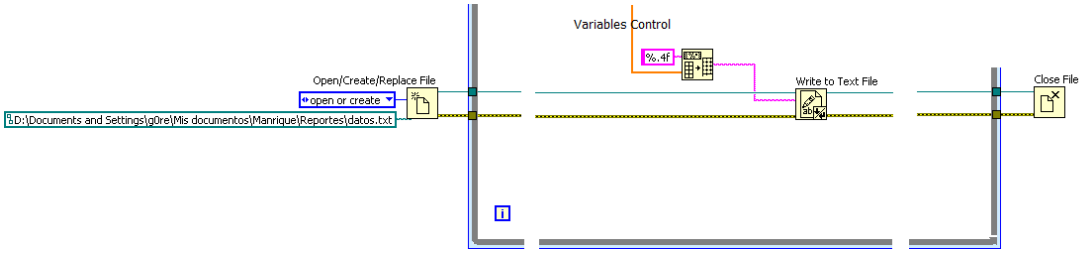


Figura 5.20. Comandos de escritura de datos a archivo.

La integridad del código se muestra en la Figura 5.21 a continuación:

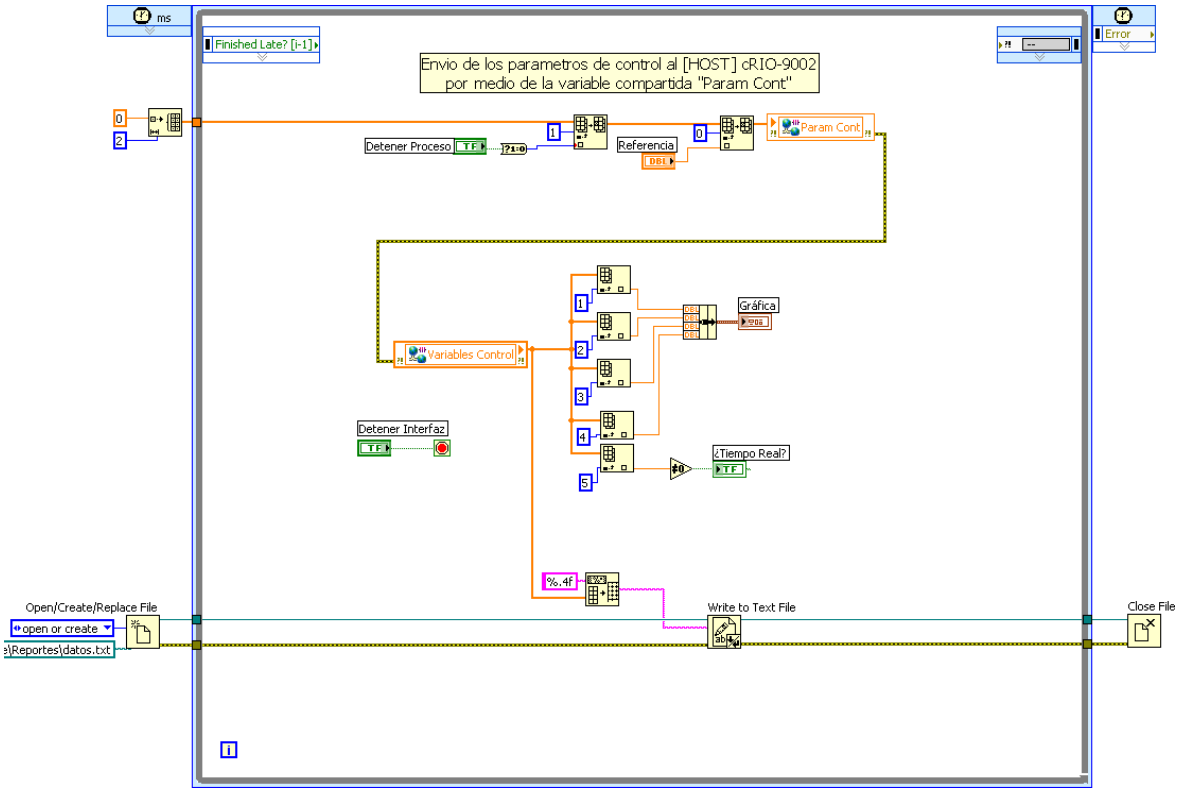


Figura 5.21. Diagrama de bloques de la interfaz de usuario.

Capítulo 6
Resultados de simulación y experimentales

En este capítulo se muestran los resultados de las simulaciones numéricas ejecutadas en el entorno de desarrollo de LabView y los resultados de las pruebas efectuadas implementando el código en el módulo compactRIO y utilizando el simulador de procesos PCS327.

El propósito de las simulaciones es verificar la funcionalidad del código programado y la obtención de valores iniciales adecuados para los parámetros variables de la red. Como se realizó en el Capítulo 4, la red neuronal está caracterizada por los vectores A, B, C, D, W, así como el número de neuronas, la cantidad de épocas de aprendizaje y la función wavelet seleccionada como función de activación de las neuronas.

Durante la implementación en tiempo real del algoritmo se busca hacer una evaluación cualitativa del desempeño del control y conseguir un bajo tiempo de ejecución.

En las pruebas se busca controlar una planta y sintonizar el controlador durante la operación. Ambas pruebas se llevan a cabo asumiendo que se desconoce la función de transferencia de la planta.

6.1 Simulaciones en LabVIEW

Experimento 1: Control PID Wavenet para controlar un programa (VI) que simula un proceso de primer orden.

En estas pruebas se emplea una red neuronal de 10 neuronas. Los parámetros variables en este experimento son los valores iniciales de los vectores de la wavenet, así como las velocidades de aprendizaje y las ganancias iniciales del PID. Se emplea como wavelet madre la función RASP1 (RAtional Second order Poles):

$$\psi(\tau) = \frac{\tau}{\tau^2 + 1}$$

La red wavenet puede efectuar varias épocas de aprendizaje dentro de un ciclo de control. Para determinar una cantidad razonable de épocas, el experimento se repitió varias veces, seleccionando la cantidad de épocas que ofrece un porcentaje de error de 1% después del primer escalón, lo cual lleva a un lazo de control más rápido.

Épocas	Tiempo [s]
1	0.4404
3	0.442
5	0.4431
7	0.4443
10	.4457
15	.4469
20	.447
25	.4459
30	0.4444
35	0.5011
40	0.5027

Tabla 6.1 Épocas de aprendizaje y tamaño del error.

La diferencia en tiempo no resulta significativa en este experimento debido a que el ciclo de control tiene una duración de 50 [ms]. Entre 5 y 10 iteraciones ofrecen una respuesta razonablemente suave, mientras que más de 10 introducen oscilaciones. Las pruebas que se reportan a continuación se efectuaron con 7 épocas de aprendizaje.

La tabla 6.2 contiene los valores iniciales de los parámetros variables de la red.

	Valores iniciales	Valores finales
A	[0 5 10 15 20 25 30 35 40 45]	[5.8759 5.6956 4.6653 4.8752 3.8466 3.5537 3.6975 4.3631 4.6799 4.8209]
B	[5 5 5 5 5 5 5 5 5 5]	[-0.8593 4.6319 8.8246 14.4029 18.4711 23.4791 30.7175 35.2248 40.0813 45.0354]
C	[0.1 0.1]	[0.4321 0.4319]
D	[0.1 0.1]	[0.3357 0.3356]
W	[0 0 0 0 0 0 0 0 0 0]	[4.9341 1.2617 1.1629 0.2721 0.4134 -1.6550 -2.8799 -1.1802 -0.5647 -0.3094]
Kp	0.7	0.7035
Ki	0.7	0.7204
Kd	-0.2	-0.2292

Tabla 6.2. Valores iniciales y finales de la wavenet.

La señal de entrada a este sistema fueron tres escalones de 5 V con duración de 5 segundos. La Figura 6.1 muestra la respuesta del sistema a esta señal de entrada.

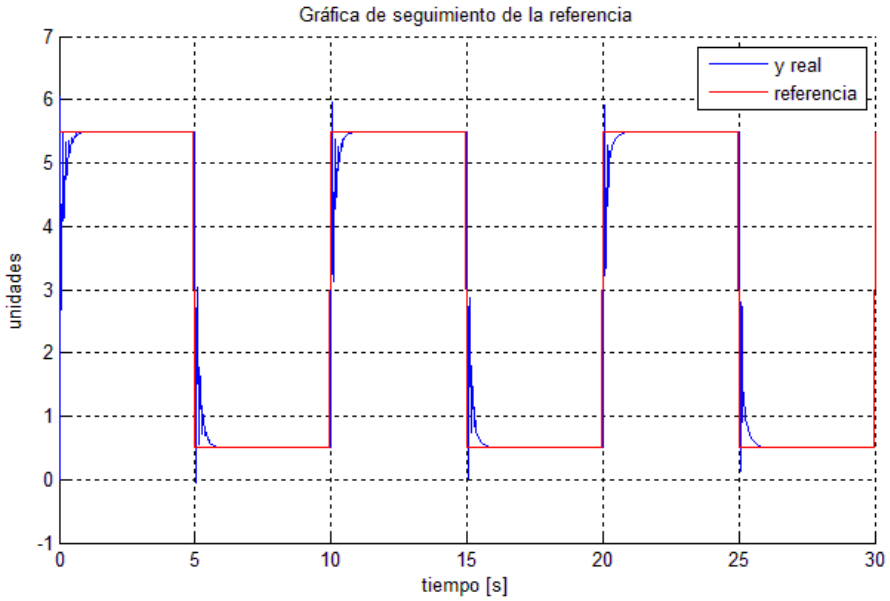


Figura 6.1. Respuesta del sistema.

A continuación se muestra la variación de los parámetros de control durante la ejecución del programa.

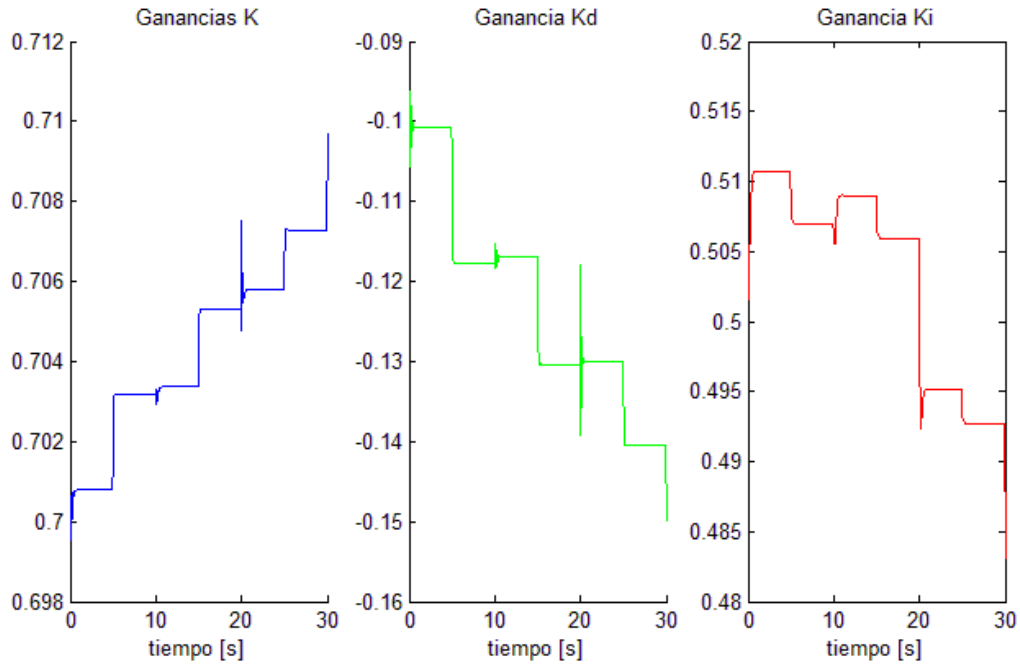


Figura 6.2. Variación de los parámetros del PID durante el tiempo de ejecución.

En este experimento, Kd se vuelve negativa

Experimento 2: En este experimento la función de entrada es una señal senoidal con un periodo de 5 s y se introduce una variación de la carga del sistema proporcional a la señal de entrada. El sistema opera durante 30 s.

Nuevamente se hace una comparación de la cantidad de épocas de aprendizaje con la disminución del tiempo en que se alcanza un error del 1%. El menor tiempo de reducción del error se obtiene utilizando cinco épocas de aprendizaje en cada ciclo de control.

Épocas	Tiempo
1	1.8858
5	1.8838
10	1.8842
20	1.8846
30	1.8846
40	1.8866

Tabla 6.3. Reducción del error.

La respuesta del sistema a esta entrada es

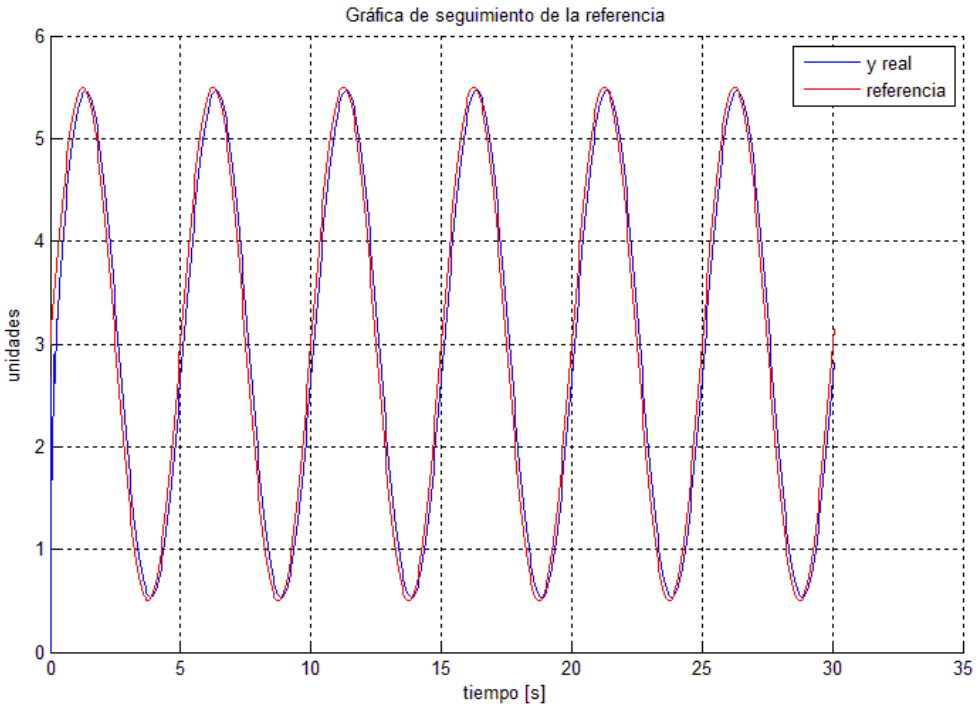


Figura 6.3 Respuesta del sistema a la señal senoidal.

La variación de los parámetros del controlador se ilustra a continuación.

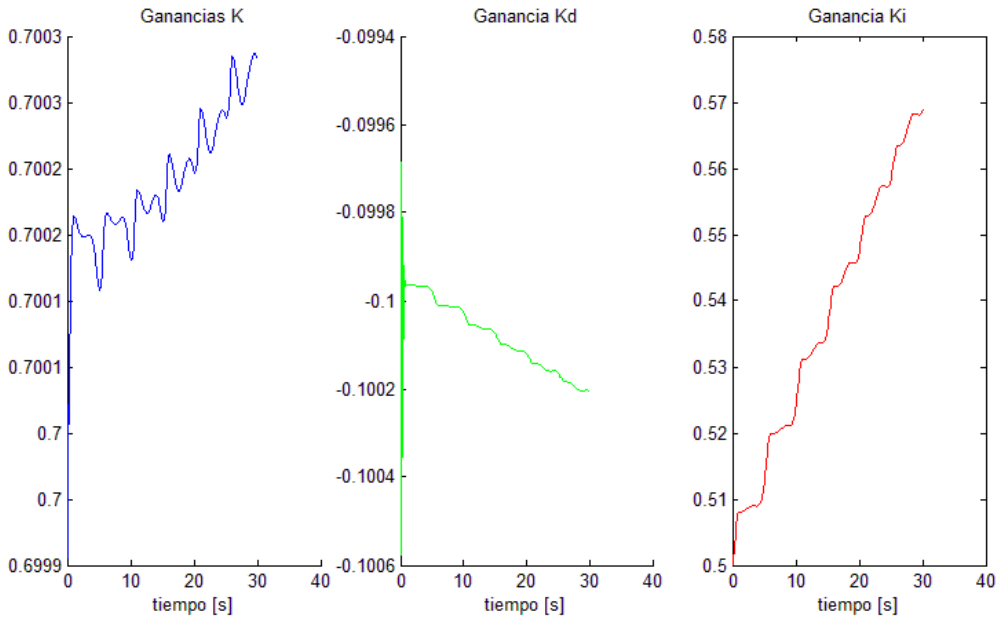


Figura 6.4 Variación de los parámetros del controlador.

6.2 Pruebas con el simulador de procesos

El simulador de procesos PCS327 es un equipo que permite simular procesos de primer, segundo y tercer orden. Estos procesos tienen constantes de tiempo ajustables de ya sea 10 ms o 1 s. También es posible agregar retrasos en el tiempo o perturbaciones a la entrada o salida del proceso. El simulador de procesos permite generar una planta compuesta por tres bloques cuya función de transferencia es $G(s) = \frac{100}{s+100}$. El equipo se muestra a continuación en la figura 6.5.

A partir de las simulaciones reportadas previamente se obtuvo un conjunto de valores aceptables para las variables de la wavenet y el PID. La implementación en compact RIO aprovecha de estos resultados, así como busca que el código pueda ejecutarse en el menor tiempo de procesamiento posible. Con este propósito, se reduce el tamaño de algunos de los vectores de la red, para reducir el tiempo de procesamiento.

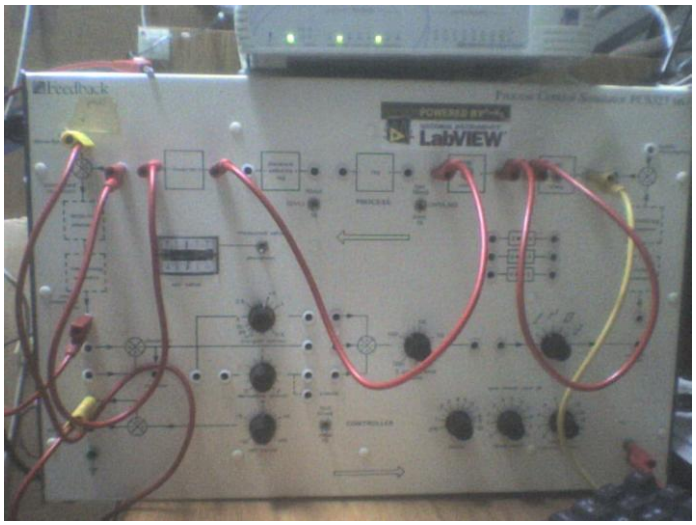


Figura 6.5. El simulador de procesos PCS327.

En estas pruebas no se leen las variables de la wavenet. La introducción de las rutinas de lectura y escritura al código del programa generan retrasos importantes en el tiempo de ejecución. Las lecturas efectuadas son de las ganancias del controlador PID y de las variables de control (variable manipulada, variable controlada, referencia y estimación de la wavenet). El mínimo tiempo que se consiguió para el ciclo de control fue de 5 ms. Las lecturas se realizan cada 30 ms.

Primer experimento: Se configura el módulo PCS327 como una planta de tercer orden con constante de tiempo de 10 ms.

Se emplea una red con siete neuronas y siete épocas de aprendizaje en cada ciclo de control.

	Valores iniciales
A	[0 5 10 15 20 25 30]
B	[5 5 5 5 5 5 5]
C	[0.1 0.1]
D	[0.1 0.1]
W	[0 0 0 0 0 0 0]
Kp	0.7
Ki	0.2
Kd	0.2

Tabla 6.4. Parámetros iniciales del PID Wavenet.

La señal de referencia son tres escalones de 5 V con duración de 1 s aplicados a los 2, 4 y 6 segundos de operación.

La figura 6.6 muestra el seguimiento de la señal de referencia, así como la señal de control y la aproximación hecha por la red wavenet.

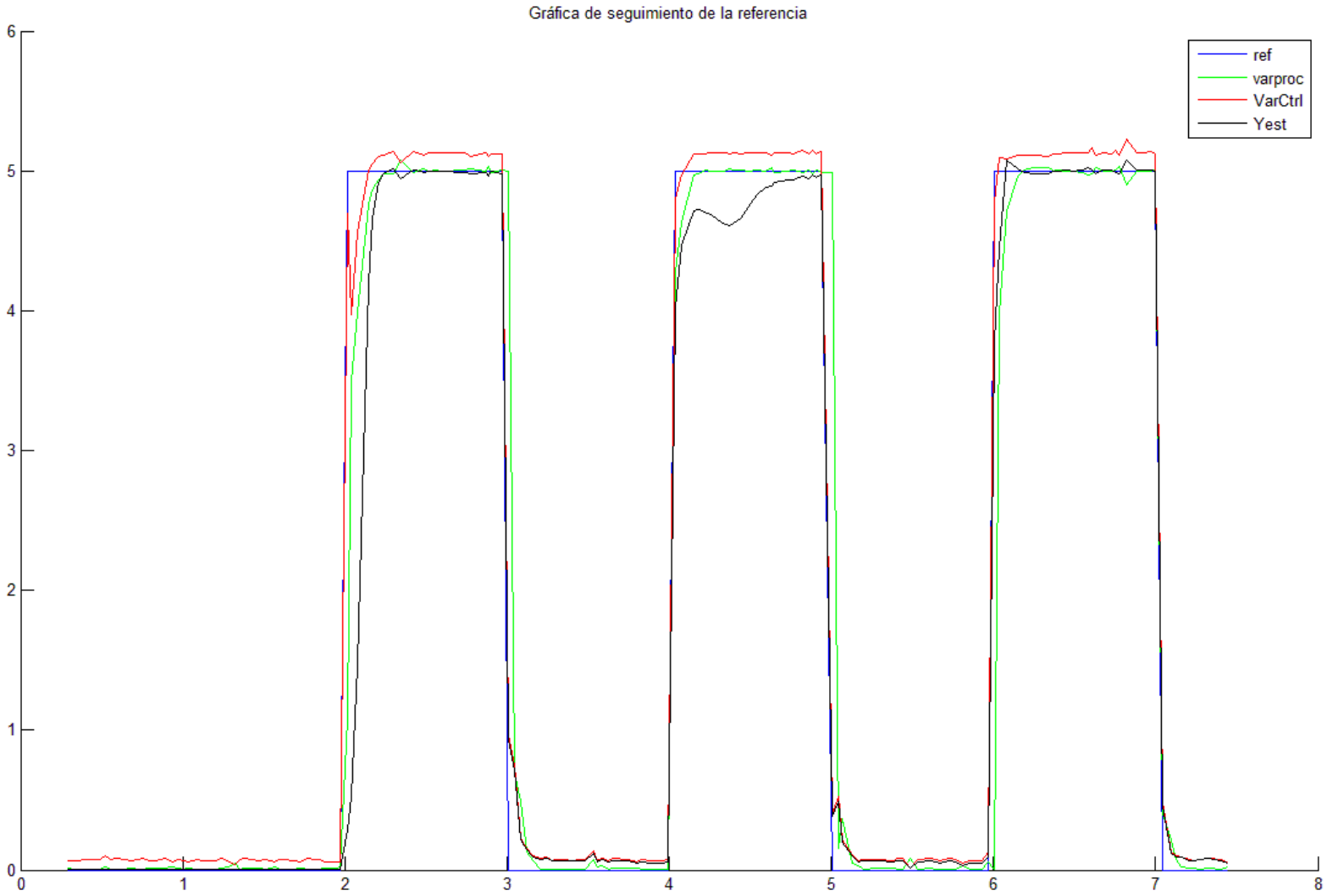


Figura 6.6. Respuesta del sistema con control PID Wavenet.

Se puede observar que existe una ligera oscilación de la variable manipulada en cada flanco de subida y de bajada.

La variación de las ganancias del controlador se muestra a continuación:

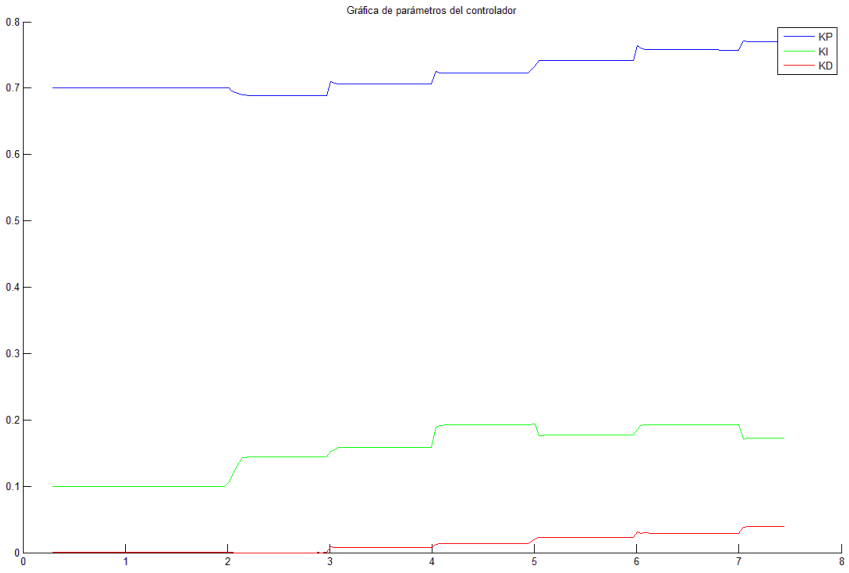


Figura 6.7. Variación de las ganancias del PID Wavenet.

Segundo experimento: Se configura el módulo PCS327 como un proceso de 2do orden con constante de tiempo de 10 ms. Se emplean las mismas condiciones iniciales que en el experimento anterior, y la misma señal de entrada. La duración de este experimento es de 16 segundos.

La figura 6.8 contiene el valor de la señal de referencia, la variable del proceso, la señal de control y la estimación de la wavenet para este experimento.

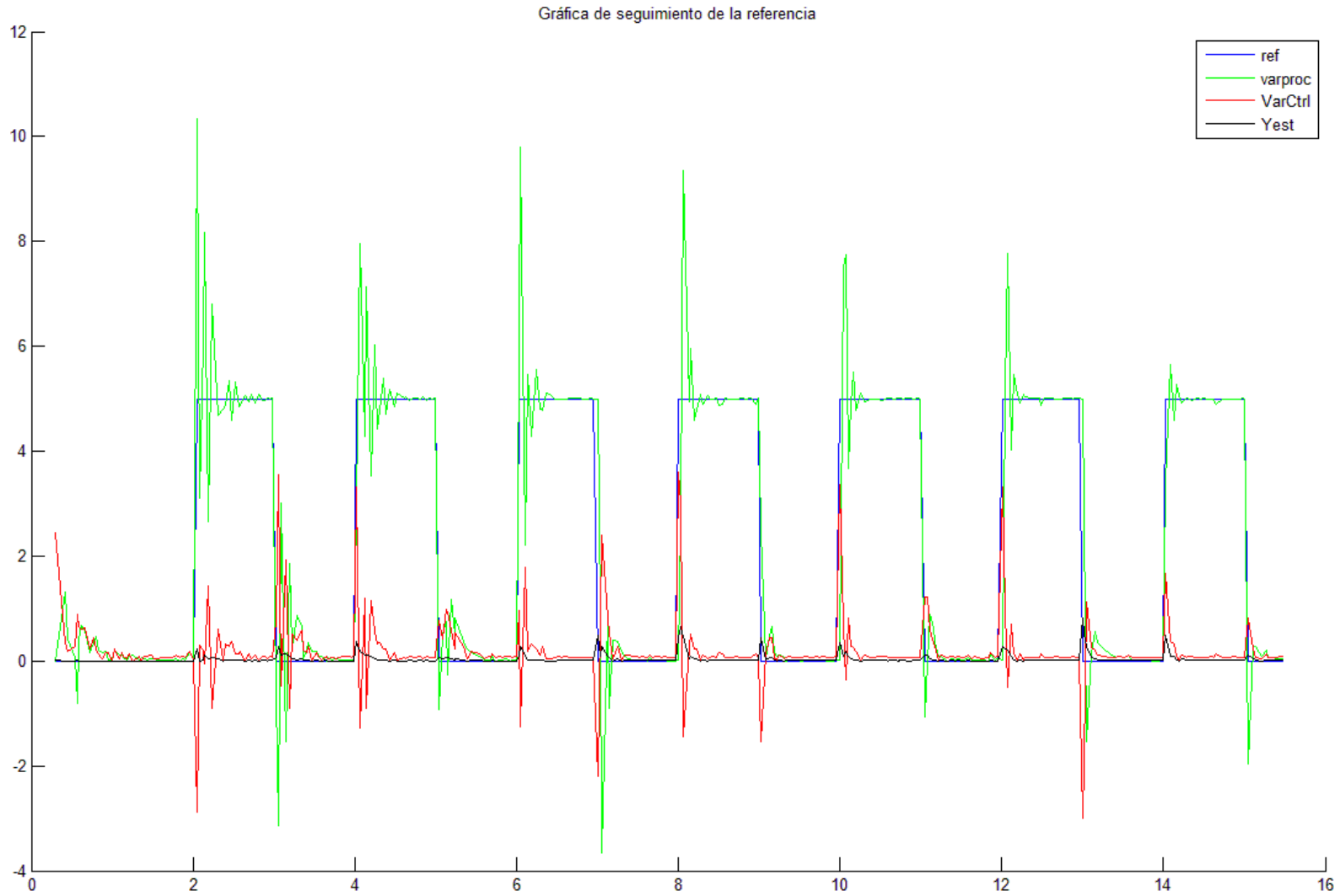


Figura 6.8. Comportamiento del sistema de segundo orden.

Se puede observar que la respuesta del sistema presenta un gran sobrepaso y un gran tiempo de asentamiento, que se reducen después de varios ciclos de control sin llegar a desaparecer completamente.

Las ganancias del controlador se ponen a continuación. Cabe notar que el tamaño de las oscilaciones disminuye a medida que Kp y Ki disminuyen y Kd aumenta ligeramente. Estos valores tienden a continuar su tendencia a disminuir. En experimentos similares con velocidades de aprendizaje mayores para Ki el sistema se inestabiliza en cuanto Ki adquiere un valor negativo.

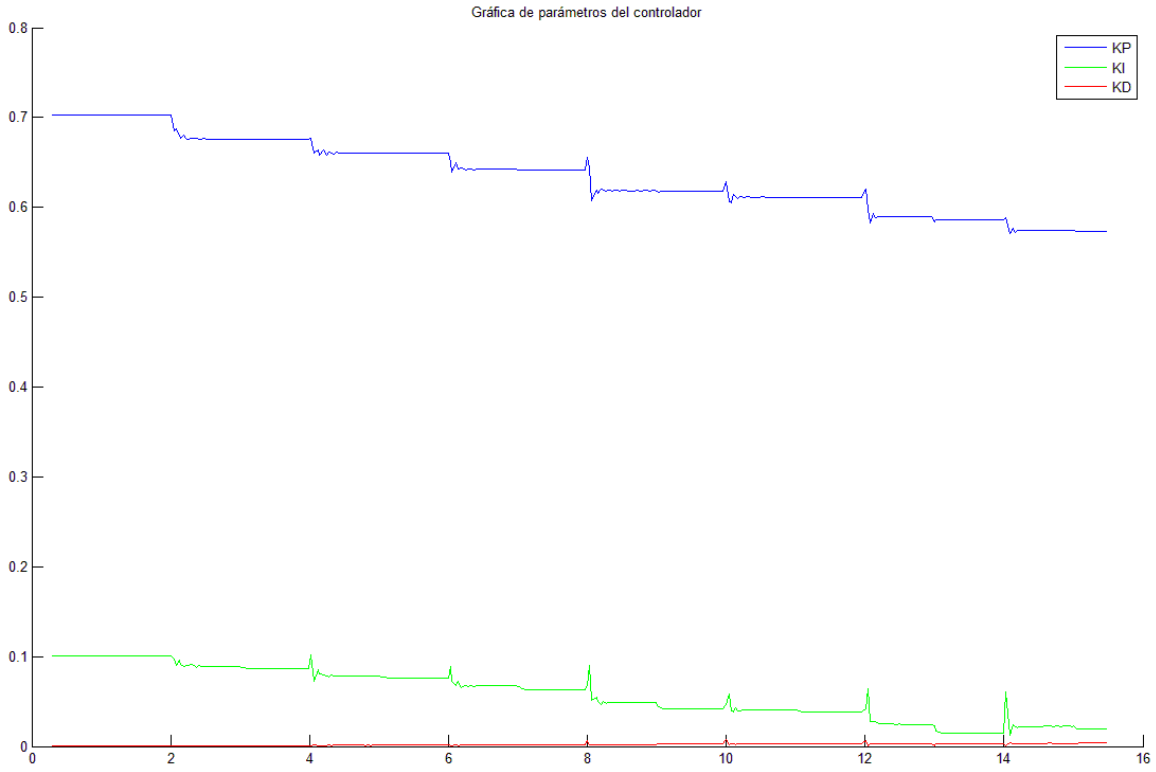


Figura 6.9. Variación de las ganancias del controlador.

Tercer experimento: En operación, se cambia el proceso de tercer orden a segundo orden, nuevamente a tercer orden y finalmente a segundo orden. Este experimento tiene como objetivo analizar el desempeño del controlador ante procesos que experimentan cambios bruscos en su función de transferencia.

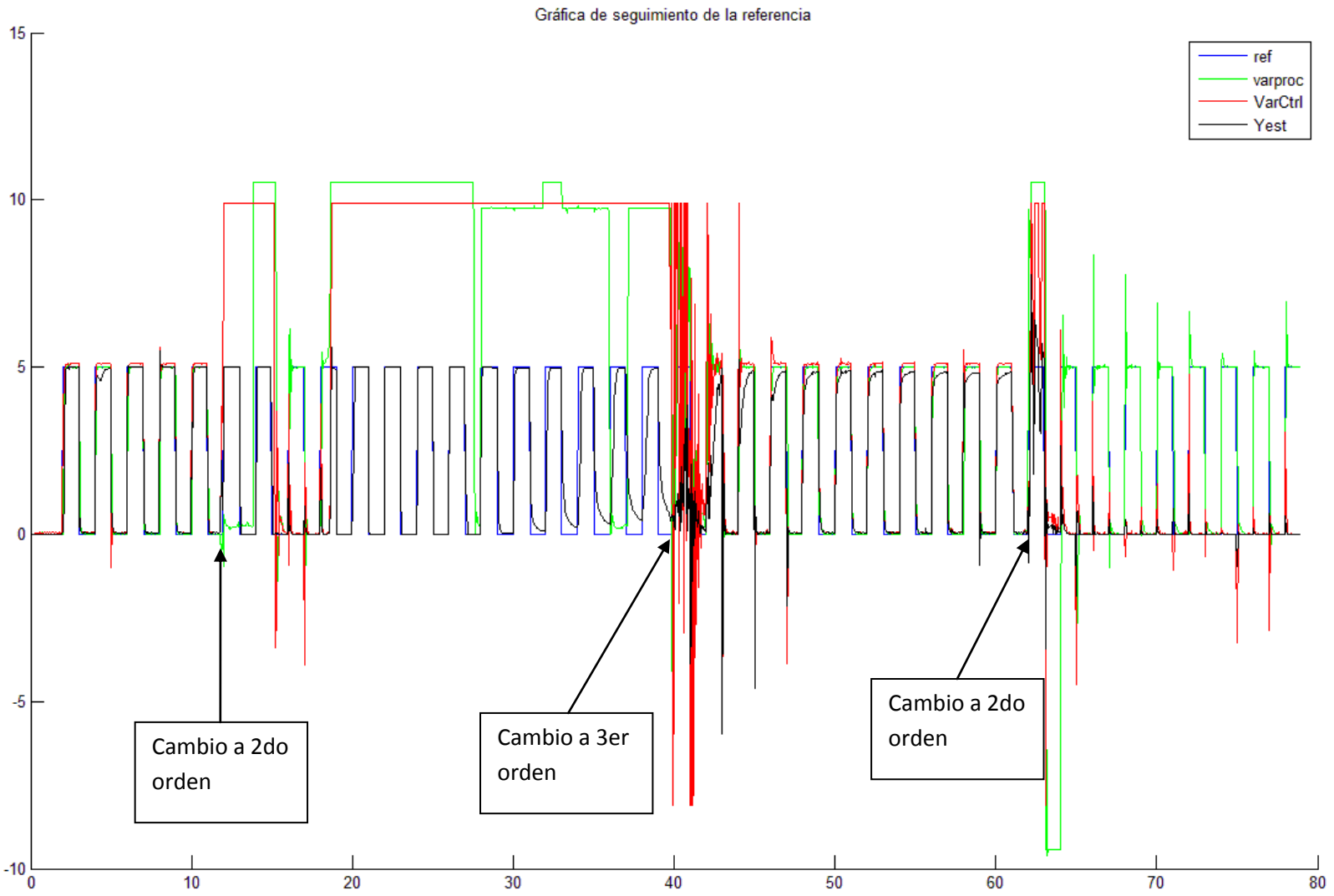


Figura 6.19. Comportamiento del sistema que cambia de orden.

El sistema muestra un comportamiento que puede considerarse adecuado cuando la planta controlada es de tercer orden. En cuanto se realiza el cambio de orden de la planta, el sistema se satura, presenta fuertes oscilaciones y se satura nuevamente. Al retornarlo a tercer orden, toma 6 [s] al sistema reducir las oscilaciones y presentar nuevamente un comportamiento suave sin sobrepasos. Al cambiar una vez más a una planta de segundo orden, el sistema ya entrenado presenta el comportamiento visto en la prueba anterior, reduciendo los sobrepasos y oscilaciones sin suprimirlas completamente. En las gráficas 6.19 y 6.20 se indican los instantes en que se realizó el cambio en la dinámica del proceso.

A continuación se muestra el cambio de las ganancias del controlador:

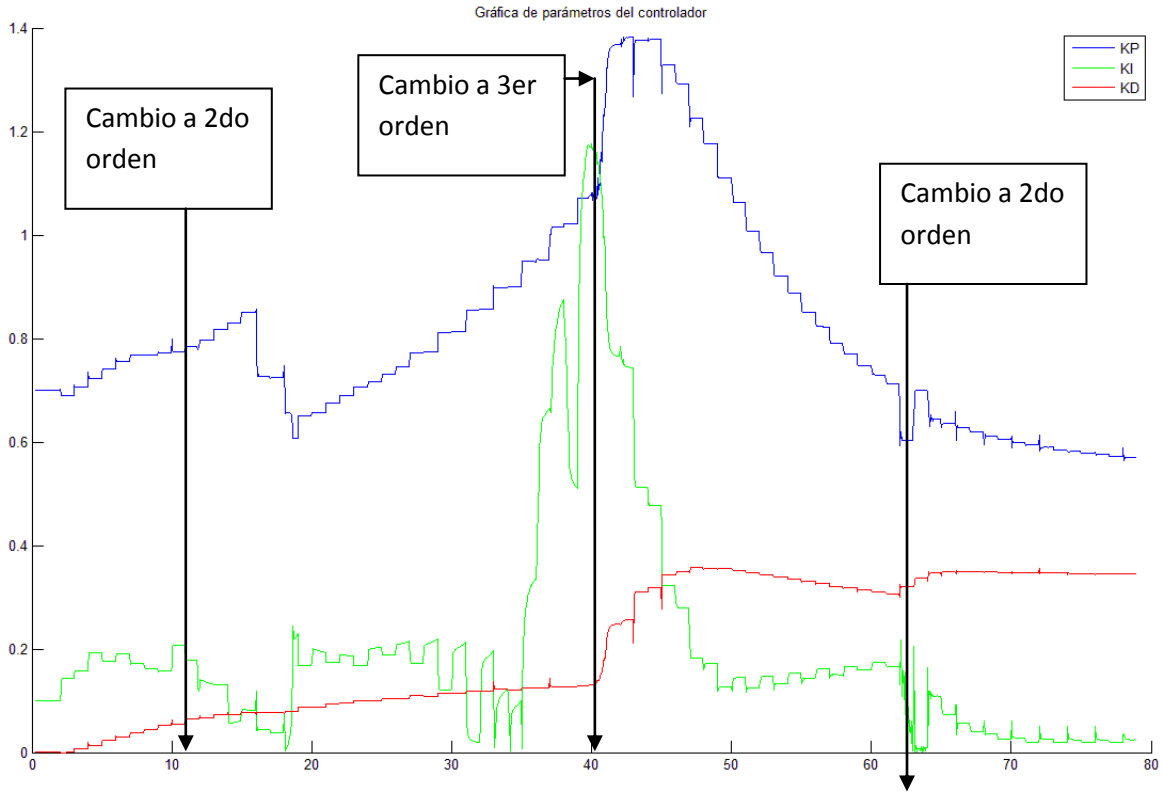


Figura 6.20. Ganancias del PID.

Cuarto experimento: El proceso se cambia de segundo a tercer orden en operación, dejando transcurrir varios escalones completos para el entrenamiento de la red wavenet.

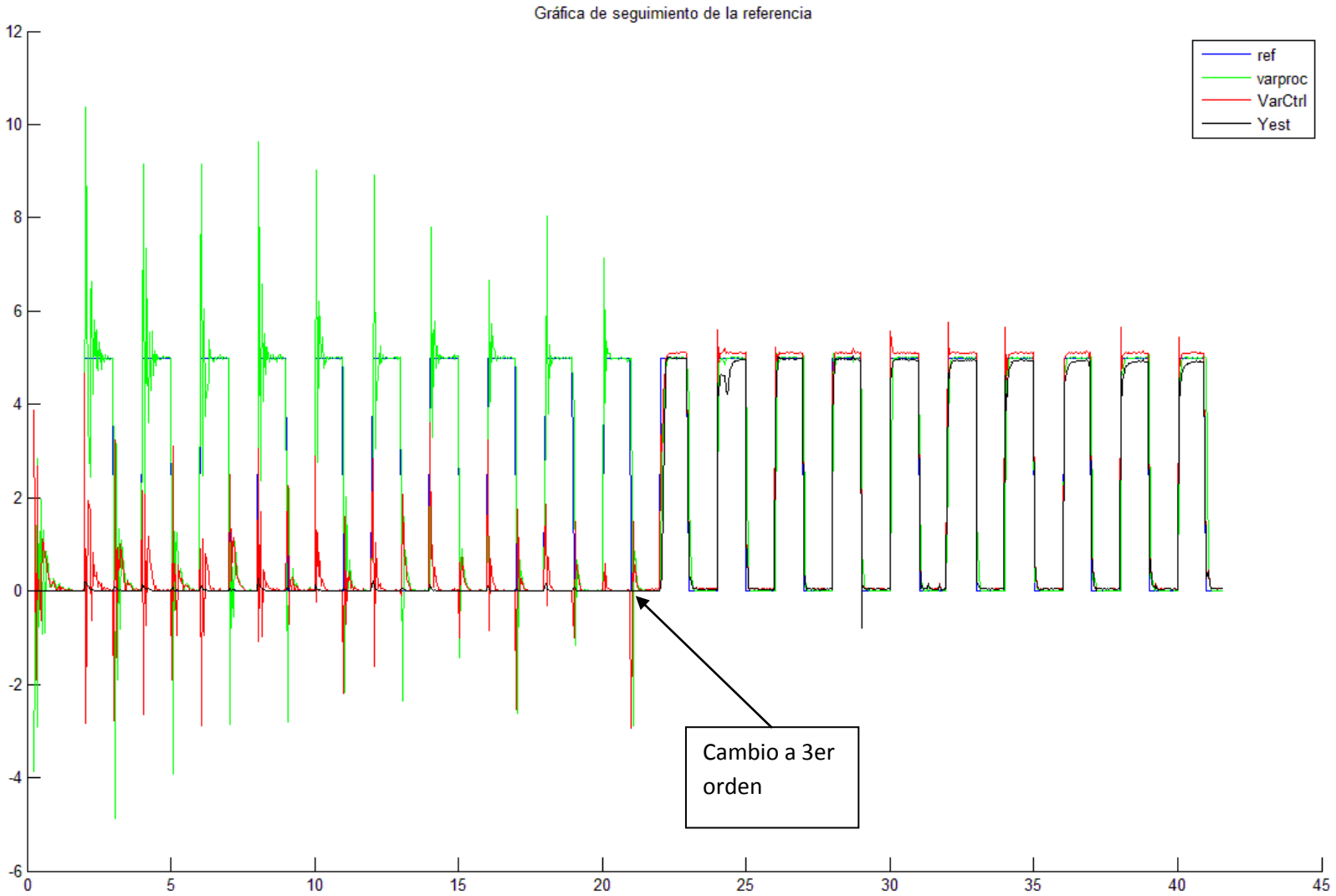


Figura 6.21. Comportamiento del sistema que cambia de segundo a tercer orden.

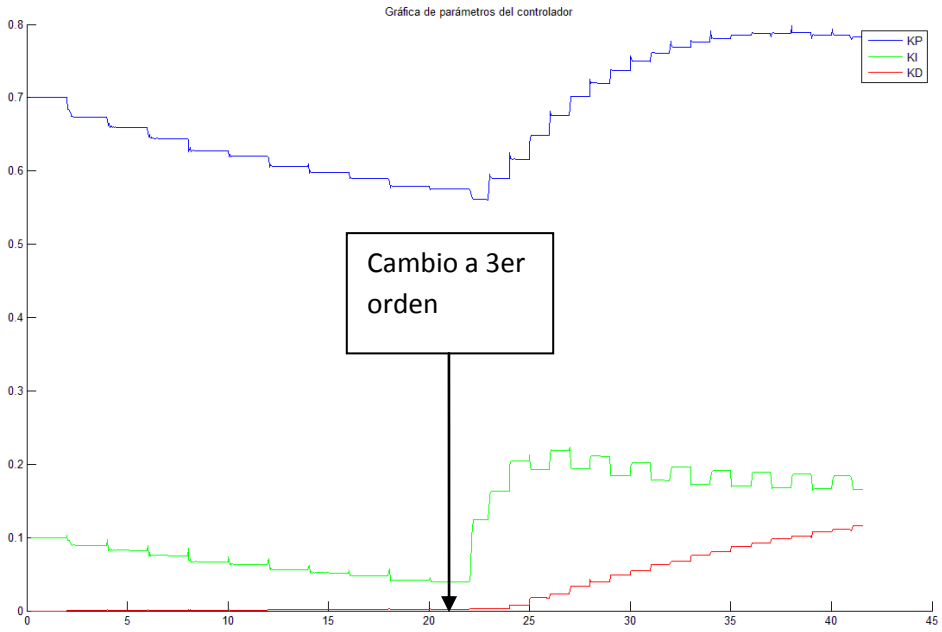


Figura 6.22. Ganancias del PID.

Los resultados obtenidos confirman lo del experimento anterior, donde las ganancias del controlador PID para el sistema de 3r orden son inestables para el sistema de segundo orden. El sistema, sin embargo, busca estabilizarse. Se observa que las ganancias del controlador se ajustan para provocar que el error se reduzca grandemente.

Capítulo 7
Conclusiones

La implementación del control PID Wavenet no resulta conviene para procesos que cambian bruscamente. Es posible que la introducción de dinámicas extrañas en un tiempo muy breve no dé oportunidad al controlador de ajustar sus parámetros antes de que el sistema muestre comportamientos inesperados o indeseables. Adicionalmente, como los métodos de ajuste son basados en derivadas parciales, se pueden introducir factores de ajuste muy grandes cuando el sistema experimenta cambios importantes en un periodo muy breve de tiempo.

PID Wavenet no aumenta significativamente el costo de implementar en un microncontrolador o una computadora de uso industrial. Si bien una gran cantidad de los procesos industriales se pueden resolver utilizando controladores tipo PID, PID wavenet ofrece una alternativa de control para procesos que presenten cambios, y en los que se desee un control ajustable en operación.

Como se esperaba de acuerdo al planteamiento del controlador adaptable, PID Wavenet ajusta la planta para cargas variables.

Tener algún conocimiento del comportamiento del sistema es una gran ayuda en la implementación de un PID ajustable. Es importante una selección adecuada de los parámetros iniciales para prevenir comportamientos erráticos mientras el sistema se ajusta. Depende de las velocidades de aprendizaje del controlador qué tan fino o brusco será el ajuste de las ganancias.

Cuando se requiere controlar procesos cuyas dinámicas no cambian, se puede emplear PID Wavenet para la aproximación del modelo de la planta, y es conveniente limitar la acción de sintonización después de un cierto tiempo, para prevenir una sobresintonización de los parámetros.

Bibliografía

[1] K. Ästrom & T. Hagglund. *PID Controllers: Theory, Design and Tuning*. International. Estados Unidos. 2007.

[2] J. A. Cruz Tolentino. *Diseño y aplicaciones de un controlador PID wavelet.* Tesis de maestría. Universidad Autónoma Metropolitana. México. 2009.

[3] L. Fausett. *Fundamentals of neural networks: Architecture, algorithms and applications.* Prentice Hall, 1994.

[4] J. A. Freeman. *Simulating neural networks with mathematica.* Reading, MA: Addison-Wesley, 1994.

5[5] A. Garibay Martinez. *Control en tiempo real de procesos dinámicos rápidos empleando módulos de tecnología FPGA*. Reporte de proyecto terminal de licenciatura. Universidad Autónoma Metropolitana. México 2007.

[6] L. Gaviphat. *Adaptive Self-Tuning Neuro Wavelet Network Controllers.* Tesis doctoral. Estados Unidos. 1997.

[7] A. Graps. *An introduction to wavelets.* IEEE Computational Science and Engineering, Summer 1995, vol. 2, num. 2.

[8] J. Hertz, A. Krogh & R.G. Palmer. *Introduction to the theory of neural computation.* Santa Fe Institute Studies in the Sciences of Complexity (vol. 1). Redwood City, CA: Addision-Wesley. 1991

[9] M. Johnson & M. Moradi. *PID Control, New Identification and Design Methods.* Springer-Verlag. Reino Unido. 2005.

[10] H. Li, H. Jin, and C. Guo. *PID Control Based on Wavelet Neural Network Identification and Tuning and Its Application to Fin Stabilizer.* Proceedings of the IEEE International Conference on Mechatronics and Automation, pp. 1907-1911, Niagara Falls, Canada, July 2005.

[11] R. P. Lippmann. *An introduction to computing with neural nets.* IEEE ASSP Magazine , 4–22. 1987

[12] W. McCulloch & W. Pitts. *A Logical Calculus of Ideas Imminent in Nervous Activity.* Bull. Math. Biophys, Vol. 5, pp. 115-133, 1943.

[13] National Instruments Corporation. *Getting Started with CompactRIO and LabVIEW.* 2008

[14] National Instruments Corporation. *Real Time Development Fundamentals Course Manual.* 2008.

[15] M.A. Razi & K. Athappilly. Expert Systems with Applications 29 (2005) 65–74)

[16] D.E. Rumelhart, G.E. Hinton & R.J. Williams. *Learning representations by back-propagating error*. Nature, 323, 533–536. Reprinted in Anderson and Rosenfeld [1988], pp. 696–699. 1986.

[17] Sedighizadeh & Rezazadeh, *Adaptive PID Control of Wind Energy Conversion Systems Using RASP1 Mother Wavelet Basis Function Networks*. Proceedings of World Academy of Science, engineering and Technology, Vol. 37. 2008.

[18] D. Seborg, T. Edgar & D. Mellichamp. *Process Dynamics and Control*. Wiley. Estados Unidos. 1989.

[19] Y. Tong, Q. Dao, and F. Xu. *AC Motor Control Based on Wavelet Network*. Proceeding of the Third International Conference on Machine Learning and Cybernetics, pp. 861-865, Shangai, China, August 2004.