DESARROLLO DE UN CONTROLADOR PREDICTIVO BASADO EN MODELO PARA PLATAFORMA INDUSTRIAL

José Carlos Oliden Semino

Piura, 20 de mayo de 2016

FACULTAD DE INGENIERÍA

Área Departamental de Ingeniería Mecánico Eléctrica

UNIVERSIDAD DE PIURA

FACULTAD DE INGENIERÍA

"DESARROLLO DE UN CONTROLADOR PREDICTIVO BASADO EN MODELO PARA PLATAFORMA INDUSTRIAL"

Tesis para optar el Grado de Master en Ingeniería Mecánico-Eléctrica con mención en Automática y Optimización.

JOSÉ CARLOS OLIDEN SEMINO

Asesor: Mgtr. José José Manrique Silupú

Piura, mayo 2016.

A mi familia y a mis amigos.

Prólogo

Este trabajo de tesis se centra en el desarrollo de un controlador predictivo basado en modelos (MPC), se ha implementado en un módulo de cuatro tanques acoplados usando como herramienta de identificación un controlador lógico programable (PLC), también se presenta en control de un modelo de secador de disco rotatorio usado en la industria de harina de pescado. Este trabajo ha sido realizado dentro de la línea de investigación de Control Avanzado del laboratorio de Sistemas Automáticos de Control de la Universidad de Piura.

Siendo el control predictivo una estrategia de control muy exitosa en procesos de gran escala, de dinámica compleja y con múltiples variables restringidas; y el PLC un dispositivo probado exitosamente en ambientes industriales. El uso de estos dispositivos para embeber controladores avanzados en procesos de menor escala significaría una mejora del desempeño. Este hecho ha motivado a realización de esta tesis.

Este trabajo también presenta las dificultades y algunas pautas para la implementación de un controlador predictivo en un PLC. Se ha desarrollado una estructura de programa en la que se incluyen los bloques de datos, las funciones y los bloques de organización, destinados a una futura implementación del controlador.

Algunos trabajos previos al inicio de la maestría relacionados con control predictivo fueron presentados en el Congreso Latinoamericano de Control Automático en el año 2012 (CLCA12) y en *European Control Conference* 2013 (ECC13). Durante el desarrollo de la maestría se publicaron artículos en *European Control Conferences* 2014 (ECC14) y en el Congreso Salesiano de Ciencia, Tecnología e Innovación para la Sociedad 2015 (CITIS15) y se asistió a la Escuela de Posgrado de Invierno en Santa Fe, Argentina, donde se estudiaron los temas: Control Predictivo, Modelación de Sistemas, Optimización no lineal, Sistemas con retardo y Sistemas de múltiples entradas y múltiples salidas.

Quiero agradecer a mi asesor José José Manrique Silupú, que se ha convertido en un amigo en los años que llevo trabajando en el laboratorio de SAC, con sus consejos y aliento me ha ayudado a hacer realidad esta tesis, a los profesores William Ipanaqué Alama, Juan Carlos Soto Bohórquez y a todos aquellos que con su apoyo han contribuido con esta obra. Así también quisiera agradecer al Consejo Nacional de Ciencia Tecnología e Innovación Tecnológica (CONCYTEC), por otorgarme la beca para realizar mis estudios de maestría.

Resumen

El presente trabajo de tesis plantea la implementación de un controlador predictivo basado en modelos, en una plataforma industrial tomando como ejemplo de aplicación una planta piloto de cuatro tanques acoplados. Se ha realizado el control del modelo de un módulo de cuatro tanques y el control predictivo de un modelo de un secador de disco rotatorio usado en la industria de harina de pescado.

El control predictivo ha sido exitosamente aplicado como controlador supervisor, la presente tesis plantea llevar las ventajas de este controlador a controladores de campo, esto mejoraría el desempeño del proceso. Una de las principales dificultades en la implementación de los controladores predictivo es la potencia de cálculo requerida, se plantea que es posible mediante el uso de algoritmos simplificados, resolver el problema de programación cuadrática que plantea el control predictivo en su formulación.

El capítulo 1 se enfoca en el desarrollo teórico del control predictivo, se describen sus fundamentos y algoritmos, también se hace una descripción de los algoritmos clásicos. En el capítulo 2 se desarrollan las estrategias de programación cuadrática más usadas para aplicaciones con control predictivo en sistemas con recursos computacionales limitados. En el capítulo 3 se presenta la modelación y el control mediante simulación del modelo de secador de disco rotatorio usando la estrategia *Extended Predictive Self-Adaptive Control* (NEPSAC), en el capítulo 4, el control del modelo de cuatro tanques acoplados usando *Model Predictive Control* (MPC), mientras el capítulo 5 se realizó la descripción del módulo y la implementación de la prueba de identificación.

Las pruebas en simulación han mostrado un buen desempeño de los controladores predictivos EPSAC y MPC. Queda como un trabajo interesante para realizar en el futuro la implementación en el módulo real de cuatro tanques acoplados y la implementación real del control en un módulo de secador de disco rotatorio.

Índice General

Índice de Figuras

Introducción

El presente trabajo de tesis plantea la implementación de un controlador predictivo basado en modelos que se ha implementado en un modelo de cuatro tanques acoplados. Se realizó la implementación en Matlab, la prueba de identificación se realizó en un PLC VIPA 314 usando en el programa STEP 7, y los lenguajes de programación SCL, AWL y KOP. Se han realizado simulaciones con el fin de sintonizar el control MPC (*Model Predictive Control*) de la planta de cuatro tanques acoplados. Se ha implementado en simulación el controlador predictivo en una planta SISO (Single Input Single Output) de secado de harina de pescado usando el enfoque NEPSAC (*Nolinear Extended Predictive Self-Adaptive Control*).

Los controladores predictivos ha sido ampliamente usados en la industria de procesos aplicado como controlador supervisor [1] y definiendo los puntos de operación de los controladores locales. La presente tesis plantea llevar las ventajas de este controlador a controladores de campo, esto mejoraría el desempeño del proceso y por consiguiente una mejor en calidad y ahorro en costos.

Una de las principales dificultades en la implementación de los controladores predictivo es la potencia de cálculo requerida. Se plantea que es posible mediante el uso de algoritmos simplificados resolver el problema de programación cuadrática que plantea el control predictivo en su formulación.

Este trabajo fue elaborado en la Universidad de Piura, en el laboratorio de Sistemas automáticos de control, bajo la línea de investigación de Control Avanzado de Procesos. Como publicaciones previas al inicio de la maestría relacionados con el tema de tesis tenemos dos trabajos presentados en el Congreso Latinoamericano de Control Automático 2012 (CLCA12) [2], [3] y uno en *European Control Conferences* 2013 (ECC13) [4]. Así también se ha publicado en *European Control Conferences* 2014 (ECC14) [5] y en el Congreso Salesiano de Ciencia, Tecnología e Innovación para la Sociedad 2015[6]. Durante la maestría se asistió a la Escuela de Posgrado de Invierno en Santa Fe, Argentina, donde se dictaron los cursos de Control de Sistemas con Retardo, Control de Sistemas MIMO,

Introducción a la Modelación y Simulación de Sistemas, Optimización: Teoría, Métodos y Aplicaciones e Introducción al control Predictivo.

Los capítulos 1 y 2 se centran en el desarrollo teórico pertinente, el primer capítulo desarrolla la teoría del control predictivo, se describen sus fundamentos y algoritmos clásicos. En el capítulo 2 se desarrollan las estrategias de programación cuadrática más usadas para aplicaciones con control predictivo en sistemas con recursos computacionales limitados. En el capítulo 3 se presenta la modelación y el control mediante simulación del modelo de secador de disco rotatorio usando la estrategia *Extended Predictive Self-Adaptive Control* (NEPSAC), en el capítulo 4 se realza la simulación y el control del modelo de cuatro tanques acoplados usando *Model Predictive Control* (MPC), mientras el capítulo 5 se realiza la implementación de la prueba de identificación en el módulo del control predictivo y una descripción de una estructura de programa para la operación manual, prueba de identificación y control automático del módulo, desarrollas en STEP7 y Wincc.

Capítulo 1.
Control Predictivo Basado en Modelos.

1.1 Introducción

El control predictivo basado en modelos, conocido como MPC o MBPC de sus siglas en inglés (*Model preditive control* o *Model based predictive control*), es una de las técnicas avanzadas de control que ha tenido un impacto importante en la industria. Las razones de su éxito son:

- Puede manejar problemas de control de múltiples entradas y múltiples salidas (sistemas MIMO con sigas en inglés) de una manera sencilla y tomando en cuenta la interacción en el sistema. Incluso es capaz de manejar sistemas con mayor número de salidas que de entradas.
- Su formulación toma en cuenta las restricciones en las entradas del proceso, la variación de las entradas y las salidas. Puede operar en condiciones cercanas a las restricciones, que lleva en general a condiciones más óptimas. El incluir las restricciones en la función de optimización permite usar las variables del sistema en todo su rango de operación permitido, incluso saturarlas cuando es necesario. Un mejor control permite la optimización de las consignas.
- Tienen una buena respuesta para sistemas con dinámica compleja tales como: sistemas con retardo, sistemas de respuesta inversa, sistemas inestables y de fase no mínima.

Presenta algunas desventajas como:

- Requiere un modelo dinámico del sistema y de las perturbaciones para predecir el comportamiento del sistema. Esto es fundamental ya que condiciona el desempeño del controlador.
- Se requiere un algoritmo de optimización on-line para la implementación del controlador. Esto lleva a tener altos costos en la implementación.

- Se suele aplicar en sistemas de dinámica lenta debido al costo computacional que requiere.

1.2 Estrategia del control predictivo.

El Control Predictivo Basado en Modelos no es una técnica de control particular, sino un conjunto de controladores que tienen las siguientes ideas en común:

a. El uso de un modelo del sistema para predecir su comportamiento futuro.
b. El uso de una función de costo.
c. Uso de la estrategia de horizonte deslizante.

En base a estas ideas se desprenden las diferentes familias de controladores predictivos. El control predictivo usa un modelo explícito del proceso que se desea controlar llamado modelo de predicción, el cual se utiliza para predecir el comportamiento del sistema en el tiempo a lo largo de un horizonte temporal, llamado horizonte de predicción (H_p). De esta manera la única salida del proceso en el caso de sistemas SISO converge a un valor de referencia que llamaremos y_{sp} como se muestra en la Figura 1.

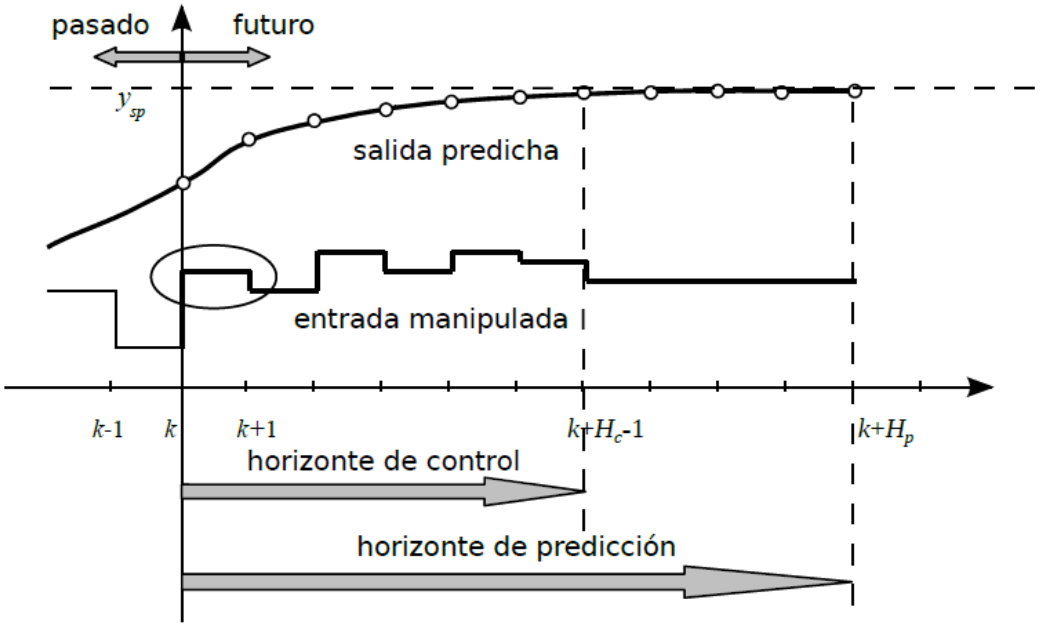

Figura 1. Control predictivo de un sistema SISO. Fuente: [7]

El problema de control se reduce a la solución de un problema de optimización con restricciones a lo largo del horizonte de predicción. De esta manera se obtiene un vector de entradas óptimas que tienen la longitud del horizonte de control (H_c). Solo se aplica a la planta el primer valor de este vector (estrategia de horizonte deslizante), esto se resuelve en cada instante de muestreo.

La estructura del controlador predictivo se representa en el diagrama de bloques de la Figura 2. Se presenta un bloque que contiene el modelo de la planta y un bloque optimizador donde se incluyen las restricciones y una función de costo, ambos bloques forman el controlador predictivo marcado en lineales discontinuas. El caso de un controlador predictivo con modelo en espacio de estados además hay un bloque para la estimación de estados.

La referencia externa (Ref) se introduce en el controlador y se restan de las salidas futuras predichas por el modelo (Ym). Esa diferencia llamada error (e) se introducen en el optimizador que calcula una entrada óptima (u) y se introduce en el proceso. El proceso actualiza las lecturas de salidas pasadas y presentes (Yp) estos nuevos valores son introducidos en el controlador que mediante el modelo repite el cálculo de las salidas futuras para un nuevo periodo de muestreo. En el caso de un modelo en espacio de estados, los valores pasados de las entradas y los valores pasados y presentes de las salidas son utilizados por el estimador para el cálculo de los estados y salidas futuras del modelo.

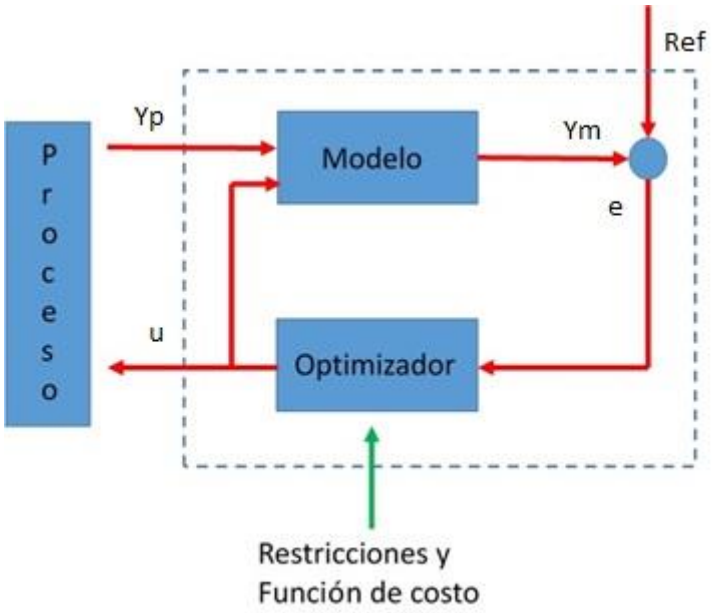

Figura 2. Estructura del controlador predictivo. Fuente: Elaboración Propia.

1.3 Modelo de predicción.

La formación de los modelos de predicción se basa en los modelos usados. Los modelos deben representar las dinámicas esenciales de los procesos y deben dar una respuesta adecuada ante cambios de las variables manipulables y las perturbaciones.

Algunas familias de controladores predictivos como GPC (*Generalized Predictive Control*) usan un modelo de respuesta impulso. Otros como el DMC (*Dinamic Matrix Control*) usan un modelo de respuesta escalón. Los controladores formulados en la academia han usado tradicionalmente modelos en espacio de estados por su facilidad de manejo ante sistemas MIMO. A continuación se describen estos modelos.

1.3.1 Modelo de predicción basado en la respuesta impulso.

Se tiene el modelo de respuesta impulso discreto de un sistema SISO o también llamado modelo de convolución.

$$y(k) = \sum_{i=0}^{\infty} g_i u(k-i) = g_0 u(k) + g_1 u(k-1) + g_2 u(k-2) + \cdots \qquad (1.1)$$

Donde $i = 0,1,2,\ldots$ y g_i son los coeficientes de la respuesta impulso. Los coeficientes de la respuesta impulso tienden a 0, por lo que se toma la suma truncada hasta N. Tomando $g_0 = 0$, la ecuación 1.1 se convierte en la ecuación (1.2).

$$y(k) = G(z^{-1})u(k) \tag{1.2}$$

Donde:

$$G(z^{-1}) = g_1 z^{-1} + g_2 z^{-2} + g_3 z^{-3} + \cdots + g_N z^{-N} \tag{1.3}$$

Tomando la ecuación 1.1 se obtiene el modelo de predicción.

$$\hat{y}(k+j|k) = \sum_{i=0}^{\infty} g_i u(k+j-i|k) \tag{1.4}$$

Donde se obtiene el valor de $\hat{y}$ estimado para el instante $k+j$, a partir del k.

Algunas características de este modelo son:

- Esta limitado a sistemas estables.
- Tiene muchos parámetros.
- Poco sensible a errores.
- No requiere conocer de antemano la estructura del modelo.

1.3.2 Modelo de predicción basado en la respuesta escalón.

Se tiene el modelo de respuesta escalón discreto de un sistema SISO.

$$y(k) = \sum_{i=0}^{\infty} h_i \Delta u(k-i) = h_0 \Delta u(k) + h_1 \Delta u(k-1) + h_2 \Delta u(k-2) + \cdots \tag{1.5}$$

Donde $i = 0,1,2,\ldots$ y h_i son los coeficientes de la respuesta escalón. Se toma la suma truncada hasta N. Tomando $h_0 = 0$, la ecuación 1.5 se convierte en:

$$y(k) = H(z^{-1})\Delta u(k) \tag{1.6}$$

Donde:

$$H(z^{-1}) = h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + \cdots + h_N z^{-N} \tag{1.7}$$

Tomando la ecuación 1.5 se obtiene el modelo de predicción.

$$\hat{y}(k+j|k) = \sum_{i=0}^{\infty} h_i \Delta u(k+j-i|k) \tag{1.8}$$

Donde se obtiene el valor de $\hat{y}$ para el instante $k+j$ desde k.

Las características de este modelo al igual que el anterior son:

- Esta limitado a sistemas estables.
- Tiene muchos parámetros.
- Poco sensible a errores.
- No requiere conocer de antemano la estructura del modelo.

1.3.3 Modelo de predicción basado espacio de estados.

Se tiene el modelo en espacio de estados en discreto.

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) \tag{1.9}$$

Tomando la ecuación 1.5 se obtiene el modelo de predicción.

$$x(k+j+1|k) = Ax(k+j|k) + Bu(k+j|k)$$
$$y(k+j|k) = Cx(k+j|k) \tag{1.10}$$

Donde se obtiene el valor de y para el instante $k+j$ desde k.

1.4 Función de costo.

Los valores futuros de las variables de control $u(k), u(k+1), u(k+2), \ldots$ son calculados para que la predicción de la salida siga una referencia. Los errores en la predicción tienen que ser mínimos y se calcula con la siguiente ecuación:

$$\hat{e} = \begin{matrix} Trayetoria\ de \\ referencia \end{matrix} - \begin{matrix} salidas \\ predichas \end{matrix} (\hat{y}(k+j))$$

En control predictivo las funciones de costo suelen ser, cuadráticas, lineales y no lineales.

1.4.1 Función de costo cuadrática.

El problema de la programación cuadrática se caracteriza porque tiene una función de costo cuadrática y restricciones lineales, en notación matricial tiene la siguiente forma:

$$\min x^T Q x + cx$$
$$s.a.\ x \geq 0$$
$$Ax \geq b \tag{1.11}$$

En control predictivo se suele tomar la siguiente función de costo.

$$\min \sum_{j=H_1}^{H_2} \alpha(j)[\hat{y}(k+j) - r(k+j)]^2 + \sum_{j=0}^{H_c-1} \beta(j)[\Delta u(k+j)]^2 \tag{1.12}$$

Donde H_1 es el inicio del horizonte de predicción, H_2 es el final del horizonte de predicción, $\alpha(j)$ es el peso de los errores futuros y $\beta(j)$ es el peso de los incrementos futuros de la variable manipulable. Los parámetros antes mencionados incluyendo el horizonte de control (H_c) son parámetros de la sintonización del controlador predictivo.

1.5 Control Predictivo con Matriz Dinámica.

El control predictivo con matriz dinámica o más conocido como *Dynamic Matrix Control* (DMC), fue diseñado en primer lugar por los ingenieros de la *Shell Oil* [8]. Actualmente, se encuentra en una gran variedad de aplicaciones en diferentes industrias y forma parte de muchos paquetes comerciales.

Se emplea un modelo de respuesta escalón para la formulación, este asume que la perturbación es constante para todo el horizonte de predicción. Este modelo tiene la limitación de solo ser poder usarse en plantas estables.

Se tiene el modelo de predicción.

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} h_i \Delta u(k+j-i|k) + \sum_{i=j+1}^{\infty} h_i \Delta u(k+j-i|k) + n(t+j) \qquad (1.13)$$

Donde $n(t+j)$ es la predicción de la perturbación, cuyo modelo es:

$$n(t+j) = y_p - \sum_{i=1}^{\infty} h_i \Delta u(k-i|k) \qquad (1.14)$$

Donde y_p es la salida medida del proceso.

Reemplazando la ecuación (1.14) en (1.13).

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} h_i \Delta u(k+j-i|k) + \sum_{i=j+1}^{\infty} h_i \Delta u(k+j-i|k) + y_p - \sum_{i=1}^{\infty} h_i \Delta u(k-i|k) \qquad (1.15)$$

Se propone la siguiente igualdad.

$$y^*(k+j|k) = \sum_{i=j+1}^{\infty} h_i \Delta u(k+j-i|k) + y_p - \sum_{i=1}^{\infty} h_i \Delta u(k-i|k) \qquad (1.16)$$

Reagrupando los términos de las sumatorias se obtiene:

$$y^*(k+j|k) = y_p + \sum_{i=1}^{\infty} \left(h_{j+i} - h_i\right) \Delta u(k-i|k) \qquad (1.17)$$

Donde la sumatoria es el efecto acumulado de los valores pasados de y, que se puede reemplazar por el valor pasado de y^*. Para sistemas asintóticamente estables, se asume que en el instante N, $h_{k+i} - h_k \approx 0$.

Luego se tiene lo siguiente:

$$y^*(k+j|k) = y_p + \sum_{i=1}^{N} \left(h_{j+i} - h_i\right) \Delta u(k-i|k) \qquad (1.18)$$

Reemplazando 1.18 en 1.15 resulta:

$$\hat{y}(k+j|k) = y^*(k+j|k) + \sum_{i=0}^{j} h_i \Delta u(k+j-i|k) \qquad (1.19)$$

El primer elemento ($y^*(k+j|k)$) se nombra como la respuesta libre y el segundo elemento $\sum_{i=0}^{j} h_i \Delta u(k+j-i|k)$ se le llama respuesta forzada.

Si asumimos que $H_p < N$, escrita de forma matricial se tiene:

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} y^*(k+1|k) \\ \vdots \\ y^*(k+H_p|k) \end{bmatrix} + \mathcal{H} \begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+H_c-1|k) \end{bmatrix} \tag{1.20}$$

Donde:

$$\mathcal{H} = \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ h_2 & h_1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ h_{H_c} & h_{H_c-1} & \cdots & h_1 \\ \vdots & \vdots & \cdots & \vdots \\ h_{H_c} & h_{H_c-1} & \cdots & h_{H_p-H_c+1} \end{bmatrix} \tag{1.21}$$

En la función de costo se pueden considerar dos partes, una que compara los valores predichos a lo largo del horizonte de predicción con la referencia y el otro que pesa las variaciones de la variable de control. Las funciones de costo más usadas son la que considera solo la primera parte (Ec. 1.22) y otra que considera a las dos (Ec. 1.23).

$$\min J = \min \sum_{i=1}^{H_p} [\hat{y}(k+i|k) - r(k+i)]^2 \tag{1.22}$$

$$\min J = \min \sum_{i=1}^{H_p} [\hat{y}(k+i|k) - r(k+i)]^2 + \sum_{i=0}^{H_c-1} \lambda [\Delta u(k+i)]^2 \tag{1.23}$$

1.6 Extended Prediction Self-Adaptive Control (EPSAC).

Esta estrategia de control predictivo fue creada por De Keyser y Van Cauwenberghe en [9]. Las primeras aplicaciones fueron presentadas en [10], [11] y [12].

La estrategia EPSAC considera en su formulación un modelo del proceso de la siguiente forma:

$$y(t) = x(t) + n(t) \tag{1.24}$$

Donde $y(t)$ es la salida medida del proceso, $x(t)$ es la salida real del proceso y $n(t)$ la perturbación. El control requiere una predicción de las futuras salidas, en un horizonte de predicción de H_p. Dado por:

$$y(t+k|t) = x(t+k|t) + n(t+k|t) \tag{1.25}$$

La salida futura puede ser descrita como la contribución de dos partes:

$$y(t+k|t) = y_{base}(t+k|t) + y_{optimo}(t+k|t) \tag{1.26}$$

$y_{base}(t+k|t)$ es el efecto de las entradas pasadas, $u_{base}(t+k|t)$ la secuencia futura de control y los disturbios.

$y_{óptimo}(t+k|t)$ es el efecto de las acciones de control $\delta u(t+k|t) = \delta u(t+k|t) - u_{base}(t+k|t)$, en un horizonte de control H_C.

El modelo de la perturbación es el siguiente:

$$n(t)\frac{C(q^{-1})}{D(q^{-1})}\xi(t) \tag{1.27}$$

Donde:

$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_c q^{-c} \tag{1.28}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + d_2 q^{-2} + \cdots + d_d q^{-d} \tag{1.29}$$

La salida optimizada se puede expresar como una ecuación de convolución en tiempo discreto, para la respuesta impulso. En notación matricial se expresa como sigue:

$$Y_{óptimo} = GU \tag{1.30}$$

Donde:

$$Y_{óptimo} = \left[y_{óptimo}(t+N_1|t), \ldots, y_{óptimo}(t+N_2|t)\right]^T$$

$$U = [\delta u(t|t), \ldots, \delta u(t+N_u-1|t)]^T$$

$$G = \begin{bmatrix} h_{N_1} & h_{N_1-1} & h_{N_1-2} & \ldots & h_{N_1-N_u+1} \\ h_{N_1+1} & h_{N_1} & h_{N_1-1} & \ddots & h_{N_1-N_u+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{N_2} & h_{N_2+1} & h_{N_1+2} & \cdots & h_{N_2-N_u+1} \end{bmatrix} \tag{1.31}$$

Entonces la salida predicha queda expresada matricialmente como:

$$Y = \bar{Y} + GU \tag{1.32}$$

Donde:

$$Y = [y(t+N_1|t), \ldots, y(t+N_2|t)]^T \tag{1.33}$$

$$\bar{Y} = [y_{base}(t+N_1|t), \ldots, y_{base}(t+N_2|t)\ ]^T \tag{1.34}$$

Si se tiene la predicción de la salida se puede optimizar la señal de control U, minimizando la función de costo J.

$$\min J = \min \sum_{k=N_1}^{N_2} [r(t+k|t) - y(t+k|t)]^2 + \sum_{k=0}^{H_u-1} \lambda[u(t+k|t)]^2 \tag{1.35}$$

Donde λ es el peso de la variable de control y $r(t+k|t)$ es la referencia. Minimizando la función de costo se obtiene U en notación matricial.

$$U = [G^T G - \lambda I]^{-1} G^T [R - \bar{Y}]$$

$$R = \begin{bmatrix} r(t+N_1|t) \\ \vdots \\ r(t+N_2|t) \end{bmatrix} \tag{1.36}$$

1.7 Generalized Predictive Control (GPC).

El Generalize Predictive Control (GPC) fue diseñado por Clarke, Mohtadi y Tuffs. Su formulación e interpretación se desarrolló por primera vez en [13] y [14]; y sus propiedades básicas en [15].

La formulación considera un modelo de la forma:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + n(t) \tag{1.37}$$

La variable $u(t)$ es la entrada al proceso, $y(t)$ es la variable de salida y $n(t)$ es el disturbio. Los polinomios A y B que contienen el operador de retardo q^{-1} tienen la siguiente forma.

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb} \tag{1.38}$$

Para el GPC el modelo usado para el disturbio $n(t)$ en el proceso es de la siguiente forma:

$$n(t) = \frac{C(q^{-1})\xi(t)}{\Delta} \tag{1.39}$$

Reemplazando 1.39 en 1.37 se obtiene:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{C(q^{-1})\xi(t)}{\Delta} \tag{1.40}$$

A este modelo se le conoce como Controlled Auto-Regressive Integrated Moving-Average (CARIMA).

Donde:

$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_{nc} q^{-nc} \tag{1.41}$$

$\xi(t)$: Es una secuencia aleatoria no correlacionada.

Δ: Es el operador diferencia $1 - q^{-1}$.

Asumiendo $C(q^{-1}) = 1$.

Se obtiene la ecuación (1.42):

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{\xi(t)}{\Delta} \tag{1.42}$$

Para hallar la predicción de la salida $y(t+j)$ para j pasos adelante se usa la siguiente ecuación.

$$1 = E_j(q^{-1})A\Delta + q^{-1}F_j(q^{-1}) \tag{1.43}$$

Obtenidos los E_j y F_j de manera recursiva por medio de la ecuación diofántina se reemplazan en la ecuación (1.44) para obtener la estimación de las salidas futuras. Se asume que la mejor predicción para $\xi(t+j)$ es 0.

$$y(t+j) = E_j B\Delta u(t+j-1) + F_j y(t) \tag{1.44}$$

Se considera la siguiente función de costo a minimizar:

$$J = \sum_{j=N_1}^{N_2} [y(t+j) - w(t+j)]^2 + \sum_{j=1}^{N_2} \lambda(j)[\Delta u(t+j-1)]^2 \qquad (1.45)$$

Donde:

N_1: Es el horizonte de predicción mínimo.

N_2: Es el horizonte de predicción máximo.

$\lambda(j)$: Es el peso de variable de control.

Se tiene el siguiente producto:

$G_j = E_j B$

La ecuación (1.44) se puede expresar de manera matricial.

$$\hat{y} = G\bar{u} + f \qquad (1.46)$$

Donde $N_1 = 1$ y $N_2 = N$:

$$\hat{y} = [\hat{y}(t+1), \dots, \hat{y}(t+N)\ ]^T$$
$$\bar{u} = [\Delta u(t), \dots, \Delta u(t+N-1)\ ]^T$$
$$f = [f(t+1), \dots, f(t+N)]^T \qquad (1.47)$$

Y la matriz G es triangular superior:

$$G = \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ g_j & g_{j-1} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ g_N & g_{N-1} & \cdots & g_1 \end{bmatrix} \qquad (1.48)$$

Expresando de manera matricial la función de costo se obtiene:

$$J = (G\bar{u} + f - \mathrm{w})^T (G\bar{u} + f - \mathrm{w}) + \lambda \bar{u}^T \bar{u} \qquad (1.49)$$

Donde:

$$\mathrm{w} = [w(t+1), \dots, \mathrm{w}(t+N)]^T \qquad (1.50)$$

Considerando que el sistema no tiene restricciones se puede obtener una solución explicita del controlador.

$$\bar{u} = (G^T G + \lambda I)^{-1} G^T (w - f) \qquad (1.51)$$

1.8 Formulación de control predictivo en variables de estados.

Se tiene el modelo en espacio de estados discreto en la siguiente ecuación:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y = Cx(k) \tag{1.52}$$

A continuación se realizó el cálculo de las predicciones de los estados del modelo para diferentes instantes en el futuro.

$x(k+1) = Ax(k) + Bu(k)$

$x(k+2) = Ax(k+1) + Bu(k+1) = A^2x(k) + ABu(k) + Bu(k+1)$

$x(k+3) = Ax(k+2) + Bu(k+2) = A^3x(k) + A^2Bu(k) + ABu(k+1) + Bu(k+2)$

Generalizando para un instante j de predicción se tiene para los estados:

$$x(k+j) = A^j x(k) + \sum_{i=1}^{j} A^{j-i}Bu(k+i-1) \tag{1.53}$$

Se realiza el mismo procedimiento para las salidas y se obtiene la siguiente generalización:

$$y(k+j) = CA^j x(k) + \sum_{i=1}^{j} CA^{j-i}u(k+i-1) \tag{1.54}$$

Los estados del modelo en general no son medibles por lo que se usan estimadores de estados. Estos usan los valores pasados de las entradas, la salida del proceso (y_p) y los estados estimados anteriormente.

$$\hat{x}(k) = A\hat{x}(k-1) + Bu(k-1) + L\left[y_p(k) - C\left(A\hat{x}(k-1) + Bu(k-1)\right)\right]$$
$$\hat{y}(k) = C\hat{x}(k) \tag{1.55}$$

Entonces las predicciones de las salidas se calculan reemplazando los estados por sus estimaciones.

$$\hat{y}(k+j) = CA^j \hat{x}(k) + \sum_{i=1}^{j} CA^{j-i}u(k+i-1) \tag{1.56}$$

Para estimar los estados hay varios criterios que se pueden tomar para escoger los valores de L. En el presente trabajo se usó el filtro de Kalman.

Las predicciones de las salidas del sistema desde el instante 1 al instante j, expresadas en forma matricial son:

$$\hat{y} = \begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k+2) \\ \vdots \\ \hat{y}(k+j) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^j \end{bmatrix} \hat{x}(k) + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{j-1}B & CA^{j-2}B & \cdots & CB \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+j-1) \end{bmatrix} \tag{1.57}$$

De forma compacta tenemos:

$$\hat{y} = G\hat{x} + Fu \tag{1.58}$$

Donde:

$$G = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^j \end{bmatrix}$$

$$F = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{j-1}B & CA^{j-2}B & \cdots & CB \end{bmatrix} \tag{1.59}$$

Se define la función de costo.

$$J = \sum_{i=N_1}^{N_2} [\hat{y}(t+i) - w(t+i)]^2 + \sum_{i=1}^{N_2} \lambda(i)[u(t+i-1)]^2 \tag{1.60}$$

La formulación anterior no garantiza error nulo en estado estacionario. Esto debido a que el modelo no es perfecto. Entonces, el valor de u que minimiza la función de costo en estado estacionario puede no coincidir con el que hace la salida igual a la referencia.

Para obtener un error en estacionario nulo a pesar del error de modelado, el modelo y la función de costo deben seleccionarse correctamente. Además el estimador debe tener acción integral. Una manera de solucionar este problema es usar un modelo en forma de velocidad, que a continuación se describe.

Teniendo en cuenta la ecuación (1.52) se realiza un cambio en los estados del modelo a forma de velocidad tomando en cuenta los siguientes cambios:

$$\Delta x(t) = x(t) - x(t-1)$$

$$\Delta u(t) = u(t) - u(t-1)$$

Se consideran los siguientes estados:

$$z(t) = \begin{bmatrix} \Delta x(t) \\ y(t) \end{bmatrix} \tag{1.61}$$

Entonces el modelo queda expresado como:

$$\begin{bmatrix} \Delta x(t+1) \\ y(t+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \begin{bmatrix} \Delta x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} B \\ CB \end{bmatrix} \Delta u(t)$$

$$y(t) = [0 \quad I] \begin{bmatrix} \Delta x(t) \\ y(t) \end{bmatrix} \tag{1.62}$$

Las nuevas matrices del modelo serían las siguientes:

$$A_v = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix}$$

$$B_v = \begin{bmatrix} B \\ CB \end{bmatrix}$$

$$C_v = [0 \quad I] \tag{1.63}$$

Entonces el modelo se puede reescribir reemplazando las ecuaciones (1.61) y (1.63) en la (1.62). Se obtiene el modelo en espacio de estados discreto para los nuevos estados.

$$z(t+1) = A_v z(t) + B_v \Delta u(t)$$

$$y = C_v z(t) \tag{1.64}$$

Se reformula el problema de programación cuadrática con sigue:

$$\min_{\Delta u(t+i)} J = \sum_{i=N_1}^{N_2} [C_v z(t+i) - w_v(t+i)]^2 + \sum_{i=1}^{N_2} \lambda(i)[\Delta u(t+i-1)]^2 \tag{1.65}$$

Donde:

$$z(t+1) = A_v z(t) + B_v \Delta u(t)$$

$$y = C_v z(t)$$

$$w_v = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ w \end{bmatrix} \tag{1.66}$$

Capítulo 2.
Programación Cuadrática.

2.1 Introducción.

De manera general un problema de optimización puede definirse como sigue:

$$\underset{x}{\text{optimizar}}\, Z = z(x) \tag{2.1}$$

Sujeto a:

$$h(x) = 0 \tag{2.2}$$

$$g(x) \leq 0 \tag{2.3}$$

Objetivo de la optimización es encontrar las variables x que optimicen la función objetivo Z, mientras se mantienen ciertos límites definidos por las restricciones de igualdad y desigualdad [16].

En este capítulo se describirá de manera resumida algunos de los algoritmos de programación cuadrática más usados en PLCs. Se hará una breve descripción del problema de programación cuadrática, así también se describirá el algoritmo hildreth, que es uno de los más usados para resolver el problema de programación cuadrática en PLCs.

2.2 Definición matemática de programación cuadrática.

Matemáticamente la programación cuadrática se define:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x + x^T g$$

$$s.t \qquad Gx \geq b \tag{2.4}$$

Donde, la matriz *Hessiana* está definida por:

$$H \in S^n \stackrel{\text{def}}{=} \{M \in \mathbb{R}^{nxn}, M = M^T\} \tag{2.5}$$

El vector gradiente $g \in \mathbb{R}^n$, la matriz de restricciones $G \in \mathbb{R}^{mxn}$, el vector de restricciones $b \in \mathbb{R}^m$.

2.3 Algoritmos de los métodos y algoritmos más usados en PLCs.

Algunos de los métodos y algoritmos más usados en la programación cuadrática son los siguientes: *Pivoting Algorithms*, *Reduced Gradient Algorithm*, *Interior Point Methods*, *Active Set Methods* [17] y el Algoritmo de hildreth. De estos los más usados en aplicaciones con PLCs son *Interior Point Methods*, *Active Set Methods* [18]y el Algoritmo de hildreth.

2.3.1 Algoritmos *Active Set* (Algoritmos de Conjunto Activo).

Los artículos [19] y [20] usan el algoritmo qpOASES ([18]) que es un algoritmo de Conjunto Activo muy usado en problemas de optimización en control predictivo.

Los métodos de conjunto activo se basan en el hecho que un problema de optimización con restricciones, algunas de estas se encuentran inactivas en el punto óptimo y pueden ser ignoradas y otras están activas [17]. Si se conociera cuáles de estas restricciones están en este último grupo se podría reducir la dimensión del problema manteniendo las restricciones activas como igualdades y descartando las otras temporalmente [21].

Ciertamente, al inicio del algoritmo no se sabe cuáles de las restricciones están activas, por lo que estos métodos deben tener una estrategia para escoger un conjunto de restricciones activas que debe cambiar con cada iteración [17], así como también un punto factible de inicio de las iteraciones.

Para explicar el algoritmo se ha seguido en la referencia [17]:

Considerando el problema de programación cuadrática de la siguiente forma:

$$\begin{aligned} P: \quad & \min_x \frac{1}{2} x^T Q x + c^T x \\ s.a. \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.6}$$

Las condiciones de optimalidad de KKT (Karush–Kuhn–Tucker) para programación cuadrática son:

$$\begin{aligned} & Ax = b \\ & x \geq 0 \\ & Qx + c - A^T p - s = 0 \\ & s \geq 0 \\ & x_j s_j = 0, \qquad j = 1, \dots, n \end{aligned} \tag{2.7}$$

Suponiendo que la solución óptima es x y se realiza la partición siguiente:

$$x = (x_\beta, x_\eta) \tag{2.8}$$

Donde el primer elemento x_β de la ecuación (2.8) es relativamente más pequeño que el segundo elemento y las variables $x_\beta \geq 0$ y $x_\eta = 0$.

Reescribiendo los parámetros del problema de optimización se obtienen:

$$Q = \begin{bmatrix} Q_{\beta\beta} & Q_{\beta\eta} \\ Q_{\eta\beta} & Q_{\eta\eta} \end{bmatrix}$$

$$A = [A_\beta \quad A_\eta]$$

$$c = \begin{bmatrix} c_\beta \\ c_\eta \end{bmatrix} \tag{2.9}$$

Luego se asume que las variables x_j donde $j \in \eta$, son ceros en el punto de solución óptima. Entonces se pueden eliminar momentáneamente del problema y resolver el siguiente más pequeño.

$$P_\beta: \ \min_{x_\beta} \frac{1}{2} x_\beta^T Q_{\beta\beta} x_\beta + c_\beta^T x_\beta$$

$$s.a. \qquad A_\beta x_\beta = b$$

$$x_\beta \geq 0 \tag{2.10}$$

Siendo la solución óptima x_β^* y se satisfacen las condiciones de optimalidad de KKT.

$$A_\beta x_\beta^* = b$$

$$x_\beta^* \geq 0$$

$$Q_{\beta\beta} x_\beta^* + c_\beta - A_\beta^T p^* - s_\beta = 0$$

$$s_\beta \geq 0$$

$$\left(x_\beta^*\right)^T s_\beta = 0, \tag{2.11}$$

La solución anterior x_β^* se introduce como una solución factible para el problema completo haciendo:

$$x = (x_\beta, x_\eta) = (x_\beta^*, 0) \tag{2.12}$$

Tomando $x_\eta = 0$ se obtiene una solución factible del problema original pero no necesariamente la óptima. Para determinarlo se usan las condiciones de KKT, haciendo $x = (x_\beta^*, 0)$ y $p = p^*$.

$$\begin{bmatrix} Q_{\beta\beta} & Q_{\beta\eta} \\ Q_{\eta\beta} & Q_{\eta\eta} \end{bmatrix} \begin{bmatrix} x_\beta^* \\ 0 \end{bmatrix} + \begin{bmatrix} c_\beta \\ c_\eta \end{bmatrix} - \begin{bmatrix} A_\beta^T \\ A_\eta^T \end{bmatrix} p^* = \begin{bmatrix} s_\beta \\ s_\eta \end{bmatrix} \geq 0$$

$$[A_\beta \quad A_\eta]\begin{bmatrix} x_\beta^* \\ 0 \end{bmatrix} = b$$

$$\left(x_\beta^*, 0\right) \geq 0$$

$$\left(x_\beta^*\right)^T s_\beta = 0, 0^T s_\eta = 0 \tag{2.13}$$

Las condiciones anteriores son satisfechas inmediatamente excepto $s_\eta \geq 0$ que está definido como:

$$s_\eta = Q_{\eta\beta} x_\beta^* + c_\eta - A_\eta^T p^*$$

Entonces se presentan dos casos:

- Si $s_\eta \geq 0$ la solución $\left(x_\beta^*, 0\right)$ es la solución óptima del problema original.
- Si $s_j < 0$ para cualquier $j \in \eta$, existe una solución mejor

El algoritmo *Active Set* para resolver el problema de programación cuadrática es el siguiente.

a. Definir el conjunto de índices $\{1, .., n\} = \beta \cup \eta$.
b. Resolver el problema reducido:

$$P_\beta: \ \min_{x_\beta} \frac{1}{2} x_\beta^T Q_{\beta\beta} x_\beta + c_\beta^T x_\beta$$

$$s.a. \qquad A_\beta x_\beta = b$$

$$x_\beta \geq 0 \tag{2.14}$$

c. Calcular:

$$s_\eta = Q_{\eta\beta} x_\beta^* + c_\eta - A_\eta^T p^* \tag{2.15}$$

Si $s_\eta \geq 0$ entonces $\left(x_\beta^*, 0\right)$ es la solución óptima del problema original. Si $s_j < 0$ para cualquier $j \in \eta$, se cambia el conjunto de índices de los conjuntos β y η y se vuelve al paso b.

2.3.2 Métodos de punto interior (*Interior-Point Methods*).

Es usado para resolver problemas de optimización lineal y cuadrática sin importar el tamaño, usando entre 25 y 80 iteraciones [22]. Este método se hizo conocido a partir del desarrollo del *Karmarkar's interior-point method* y es hasta ahora un área de investigación muy activa [17]. A continuación se plantea el algoritmo, se seguirá el documento [17]. Se presenta el problema de optimización como sigue.

$$P: \ \min_x \frac{1}{2} x^T Q x + c^T x$$

$$s.a. \qquad Ax = b$$

$$x \geq 0 \tag{2.16}$$

Se define el problema dual usando la construcción lagrangiana.

$$DP: \quad \min_{p,s,x} b^T p - \frac{1}{2} x^T Q x$$
$$s.a. \qquad A^T p + s - Qx = c$$
$$s \geq 0 \tag{2.17}$$

El algoritmo es el siguiente:

a. Dados los valores iniciales (x^0, s^0, p^0) que satisfacen las condiciones $x^0 > 0$, $s^0 > 0$ y $\mu^0 > 0$ y r q satisface la condición $0 < r < 1$ y $\epsilon > 0$, donde ϵ es la tolerancia que se usa en el criterio de parada, el valor típico es $\epsilon = 10^{-8}$.

b. Se verifican los criterios de parada.

$$\|Ax^k - b\| \leq \epsilon$$
$$\|-Qx^k + A^T p^k + s^k - c\| \leq \epsilon$$
$$(s^k)^T x^k \leq \epsilon \tag{2.18}$$

Si se cumplen se detiene el algoritmo, sino continúa el algoritmo.

c. Se calcula.

$$\mu = \left(\frac{1}{10}\right)\left(\frac{(x^k)^T (s^k)}{n}\right) \tag{2.19}$$

d. Se resuelve el sistema de ecuaciones de Newton.

$$A\Delta x = b - Ax^k =: r_1$$
$$-Q\Delta x + A^T \Delta p + \Delta s = c + Qx^k - A^T p^k - s^k =: r_2$$
$$S^k \Delta x + X^k \Delta s = \mu e - X^k S^k e =: r_3 \tag{2.20}$$

e. Determinar el tamaño de salto.

$$\theta_P = min\left\{1, r \min_{\Delta x_j < 0}\left\{\frac{x_j^k}{-\Delta x_j}\right\}\right\}$$
$$\theta_D = min\left\{1, r \min_{\Delta s_j < 0}\left\{\frac{s_j^k}{-\Delta s_j}\right\}\right\}$$
$$\theta = min\{\theta_P, \theta_D\} \tag{2.21}$$

f. Actualización de variables.

$$(x^{k+1}, p^{k+1}, s^{k+1}) \leftarrow (x^{k+1}, p^{k+1}, s^{k+1}) + \theta(\Delta x, \Delta p, \Delta s) \tag{2.22}$$

Se actualiza y se retorna al paso (b).

2.3.3 Algoritmo hildreth.

Este algoritmo es uno de lo más usados para aplicaciones con control predictivo embebidos en PLCs, algunos ejemplos de estas aplicaciones se presentan en [23] donde se muestra una aplicación en control de temperatura; en [24] se realizó el control predictivo de una planta piloto de columna de destilación en un PLC siemens de la serie 300. Un estudio comparativo entre los algoritmos Hildreth y qpOASES se realizó en [25].

Siguiendo la formulación del algoritmo de Hildreth (ver [26]), se parte de la ecuación (2.1)

$$\max_{x\in\mathbb{R}^n} x^T Cx + x^T g$$
$$s.t \qquad Ax \geq b \tag{2.23}$$

Por el teorema de Kuhn y Tucker [26] se obtiene el problema equivalente.

$$\min_z \max_x x^T Cx + x^T g - z^T(Ax - b)$$
$$s.t \qquad z \geq 0 \tag{2.24}$$

La maximización de x se puede hallar por diferenciación:

$$x^* = \frac{1}{2} G^{-1}(A^T z - g) \tag{2.25}$$

Reemplazando en la ecuación (2.24) y haciendo el cambio de variables de las ecuaciones (2.26) y (2.27).

$$H = -\frac{1}{4} AC^{-1}A^T \tag{2.26}$$

$$d = \frac{1}{2} gC^{-1}A^T + b^T \tag{2.27}$$

Se obtiene el siguiente problema equivalente:

$$\min_z z^T Hz + dz^T$$
$$s.t \qquad z \geq 0 \tag{2.28}$$

En procedimiento de cálculo del algoritmo es el siguiente:

El primer valor de z ($z^{(0)}$) se escoge arbitrariamente con la condición $z^{(0)} \geq 0$. Los valores siguientes $z^{(p)}$ se calculan con las siguientes relaciones.

$$z_i^{(p+1)} = max\left(0, w_i^{(p+1)}\right) \tag{2.29}$$

$$w_i^{(p+1)} = -\frac{1}{H_{ii}}\left(\sum_{j=1}^{i-1} H_{ij} z_j^{(p+1)} + \sum_{j=i+1}^{m} H_{ij} z_j^{(p)} + \frac{d_i}{2}\right) \tag{2.30}$$

Una vez hallado el valor de z óptimo usando el criterio $\left|z^{(p+1)} - z^{(p)}\right| \leq \varepsilon$ se reemplaza en la ecuación (2.25) para hallar el x^* óptimo.

Capítulo 3.
Simulación de un controlador predictivo usando la metodología EPSAC de un secador rotatorio de harina de pescado.

3.1 Introducción

En este capítulo se realizó la aplicación en un simulador de planta de una entrada y una salida de un secador de disco rotatorio usado en la industria de harina de pescado. Se realizó la simulación y sintonización del control predictivo de esta planta.

3.2 Control del modelo de secador de disco rotatorio.

Para el control se usará el modelo de secador propuesto en [5]. Un esquema general de este secador se presenta en la Figura 3. El modelo está dividido en tres partes, balance de energía y masa en el vapor de alimentación, balance de energía y masa en la cámara de secado y balance de masa de la concentración de harina de pescado.

3.2.1 Descripción del modelo

El secador rota disco, está compuesto por un cilindro de acero ligeramente inclinado con un diámetro entre 0.3-5m y de largo entre 5-15m, el sólido ingresa por uno de los extremos del secador y se moverá a través de los discos hacia el otro extremo.

Hipótesis del modelo:

- La velocidad de entrada de la harina es uniforme.
- El tamaño de las partículas de los sólidos es uniforme.
- No se asumen pérdidas de vapor por los discos ni por las paredes del secador.
- La dispersión de los sólidos dentro del secador es despreciable.
- No existen reacciones químicas en el proceso.
- La energía específica dentro del volumen de control es uniforme.

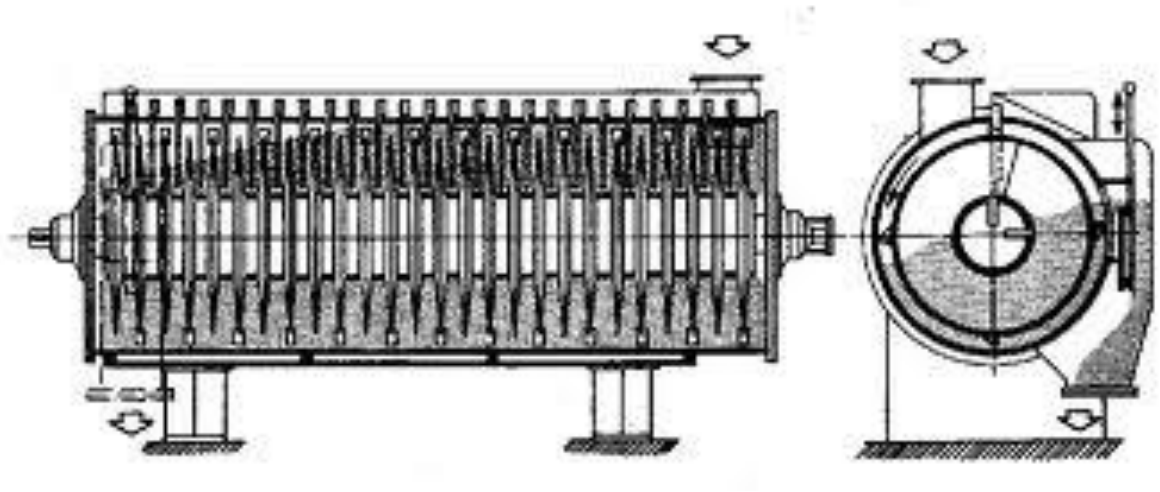

Figura 3 Diagrama General de un secador de disco rotatorio. Fuente: [5]

El modelo matemático se ha divido en tres partes:

Balance de masa y energía del vapor:

Balance de masa:

$$\frac{dm_v}{dt} = F_v - F_c \tag{3.1}$$

Como hipótesis se ha asumido que no existen pérdidas por las paredes ni por los discos. Por lo tanto, todo el vapor que ingresa al secador será expulsado como condensado.

$$F_v = F_c \tag{3.2}$$

Balance de energía:

$$\frac{d(m_v e_v)}{dt} = F_v h_v - F_c h_c - \dot{Q}_{in} \tag{3.3}$$

Despejando de la ecuación $\dot{Q}_{in}$ de la ecuación (3.3) para el estado estacionario tenemos:

$$\dot{Q}_{in} = F_v h_v - F_c h_c \tag{3.4}$$

Balance de masa y energía de la harina:

Balance de masa:

$$\frac{dm_f}{dt} = F_{f\_in} - F_{f_{out}} - W_{w_{evp}} \tag{3.5}$$

Balance de energía:

$$\frac{d(m_f e_f)}{dt} = F_{f\_in} h_{f\_in} - F_{f_{out}} h_{f_{out}} + \dot{Q}_{in} - W_{w_{evp}} h_{W_{evp}} \tag{3.6}$$

Balance de masa de la concentración:

Balance de la concentración:

$$\frac{d(m_f X_f)}{dt} = F_{f\_in} X_{f\_in} - F_{f_{out}} X_{f_{out}} \tag{3.7}$$

Reemplazando la ecuación (3.5) en la ecuación (3.7) y considerando la masa de la harina en el secador constante se obtiene la siguiente ecuación:

$$\frac{dX_f}{dt} = \frac{F_{f_{in}}(X_{f_{in}} - X_{f_{out}}) + W_{w_{evp}} X_{f_{out}}}{m_f} \tag{3.8}$$

Despejando $W_{w_{evp}}$ de la ecuación (3.6) se tiene:

$$W_{w\_evp} = \frac{F_{f\_in}(h_{f\_in} - h_{f\_out}) + \dot{Q}_{in}}{h_{w\_evp} - h_{f\_out}} \tag{3.9}$$

En todas estas ecuaciones se ha utilizado la siguiente nomenclatura:

F_v: Flujo másico de vapor [kg/s].

F_c: Flujo másico de condensado [kg/s].

W_{w_evp}: Flujo másico de agua evaporada [kg/s].

$\dot{Q}_{in}$: Calor por unidad de tiempo entregado [W].

e_v: Energía total por unidad de masa del sistema.

m_v: Masa de vapor [kg].

h_v: Entalpia específica de vapor [J/kg].

h_c: Entalpia específica del condensado [J/kg].

h_{w_evp}: Entalpía específica del agua evaporada [J/kg].

m_f: Masa de la harina [kg].

X_{f_in}: Concentración de sólidos en la entrada.

X_{f_out}: Concentración de sólidos a la salida.

h_{f_in}: Entalpia específica de la harina a la entrada [J/kg].

h_{f_out}: Entalpia específica de harina a la salida [J/kg].

T_v: Temperatura de vapor [°C].

T_c: Temperatura de condensado [°C].

T_{v_sat}: Temperatura de saturación del vapor [°C].

T_{f_in}: Temperatura de harina la entrada [°C].

T_{f_out}: Temperatura de harina a la salida [°C].

F_{f_in}: Flujo másico de la harina a la entrada [kg/s].

F_{f_out}: Flujo de másico de la harina a la salida [kg/s].

P_v: Presión de vapor [Pa].

Cp_f= Calor específico de la harina [kJ/kg°C].

La entalpia específica de vapor es:

$$h_v = 2.5x10^6 + 1813T_{v\_sat} + 0.47T_{v\_sat}^2 - 0.011T_{v_{sat}}^3 + 2090\left(T_v - T_{v_{sat}}\right)\left[\frac{J}{kg}\right] \quad (3.10)$$

La entalpia específica del condensado se calcula por la ecuación:

$$h_c = 1500 + 4122T_c + (0.55T_c)^2\left[\frac{J}{kg}\right] \quad (3.11)$$

Temperatura de saturación T_{sat}, en función de la presión de vapor P_v se expresa como:

$$T_{sat} = \frac{2147}{10.76 - \log P_v} \quad (3.12)$$

$$Cp_f = 1x\%water + 0.52x\%solid + 0.4x\%fat \quad (3.13)$$

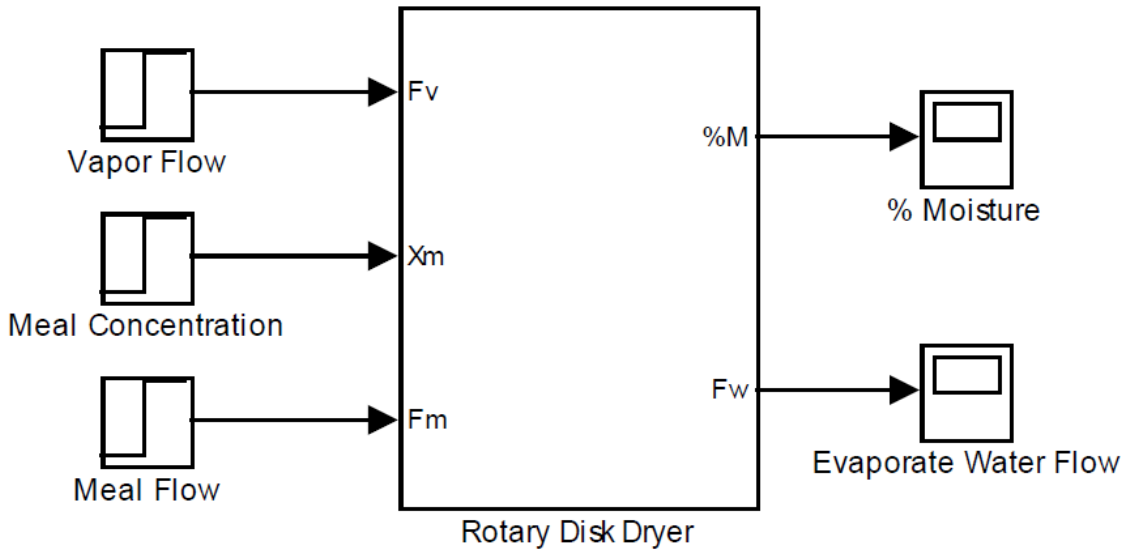

Figura 4 Esquema del modelo en simulink del secador de disco rotatorio. Fuente: [5]

3.2.2 Sintonización del controlador por simulación.

Se realizó el control predictivo del modelo de secador de harina de pescado con el enfoque EPSAC, tomando como periodo de muestreo dos minutos. Se usó el entorno e modelación por bloques simulink (ver Figura 4) para la programación del modelo y un archivo en Matlab para la simulación del control.

La variable manipulable es el flujo de vapor (Kg/s) y la variable de control es la humedad de salida del secador (%). Se realizó la sintonización el modelo a través de la siguiente estrategia:

- Se fija el horizonte de control (H_c) lo que deja como parámetros de sintonización al horizonte de predicción y el peso de la variable de control.
- Se escoge un horizonte de predicción grande (mayor del tiempo de establecimiento) y se ajusta el peso de la variable de control de tal manera que se pueda reducir el tiempo de respuesta a un tiempo adecuado para el proceso.
- Luego se reduce el horizonte de predicción y se reajusta el peso de la variable de control, se sigue esta estrategia hasta encontrar la respuesta adecuada a la aplicación y el proceso.

Se ha usado esta metodología para sintonizar el controlador. Se inicia con un horizonte de predicción de 20 y un peso de la variable de control de 1. Se ha introducido un ruido blanco de desviación estándar de 0.1, para tener condiciones más cercanas a la realidad.

Como se observa en la Figura 5 al tener un horizonte de control de 1, lambda no afecta mucho al control, esto se debe a que al ser un horizonte de control muy corto afecta muy poco en el cálculo de la variable de control. La respuesta ante disturbios de tipo ruido blanco es adecuada en los tres casos. Se observa también que el tiempo de respuesta del sistema es alto por lo que se reduce el horizonte de predicción a 10 y se prueba con diferentes lambdas.

En la Figura 6 se observa que para horizontes de predicción de 10 lambda no afecta el desempaño del controlador. Se realizó una última prueba para horizonte de predicción de 5 y diferentes lambdas. Se observa también que el controlador tiene una buena respuesta a ante un disturbio de tipo ruido blanco.

Los controladores con un horizonte de predicción (ver Figura 7) menor muestran una dinámica más rápida, pero la respuesta de la variable de control tiene una variación mayor, en el caso del controlador con horizonte de predicción de 5 las respuestas tienen las variaciones más violentas. Así mismo tienen una peor respuesta al rechazo del disturbio de ruido blanco.

Podemos concluir que la sintonización del controlador EPSAC más idónea para este proceso es con horizonte de predicción de 5, horizonte de control de 1, λ igual a 1.

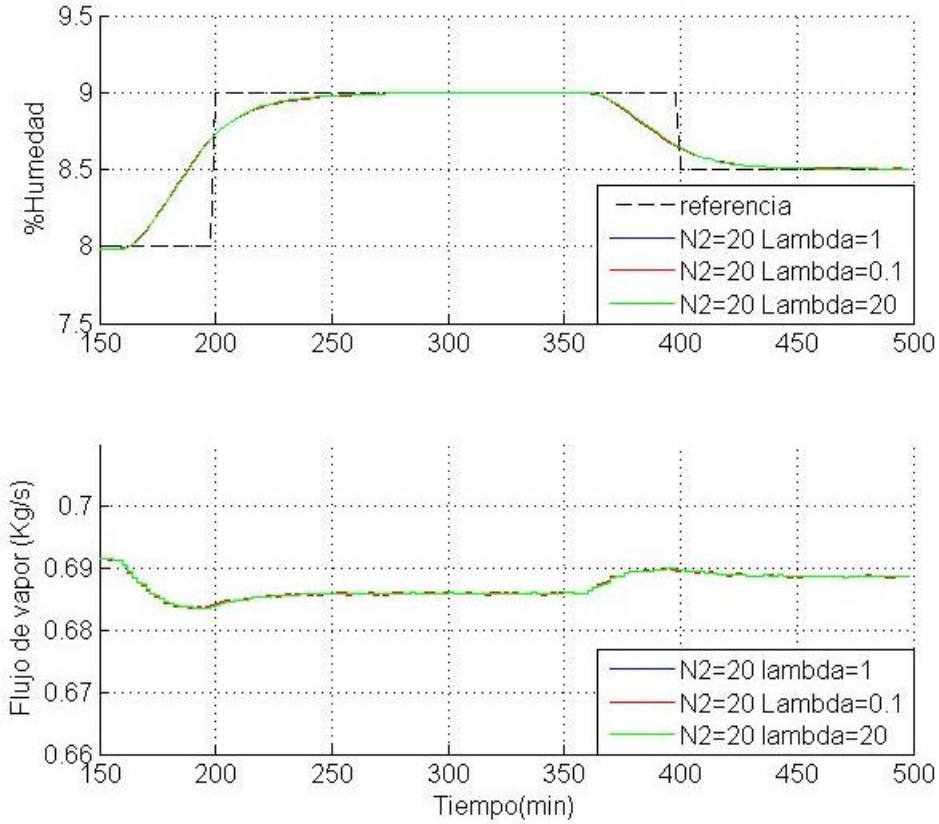

Figura 5 Primera prueba de simulación del control del secador de harina de pescado.

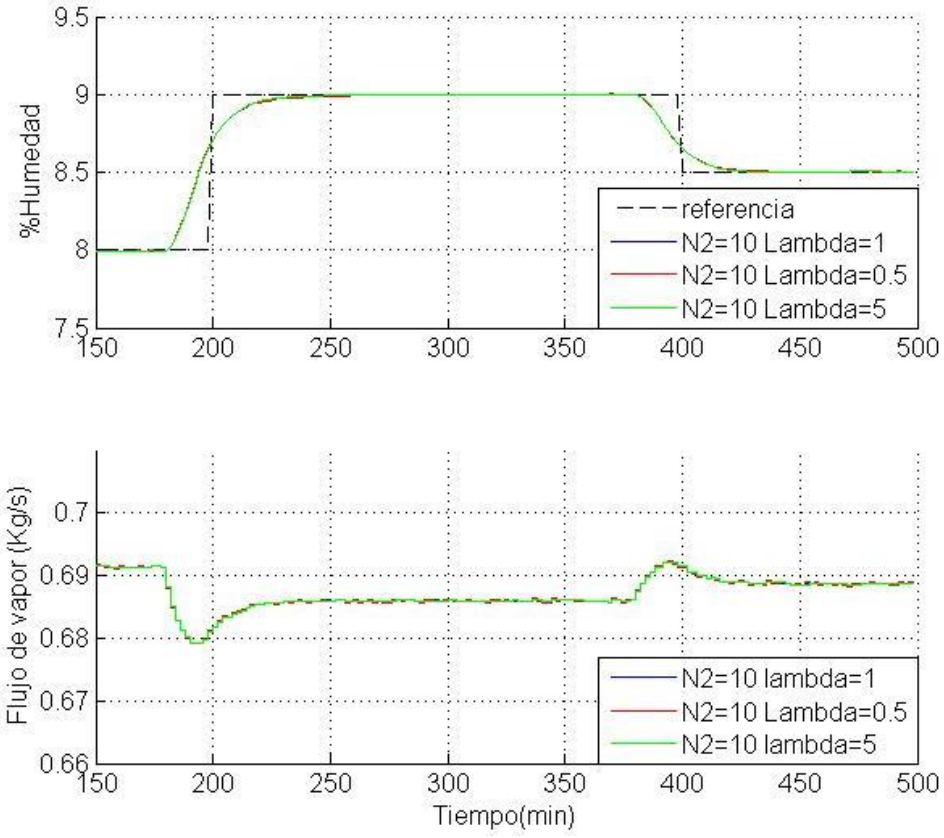

Figura 6 Segunda prueba de simulación del control del secador de harina de pescado.

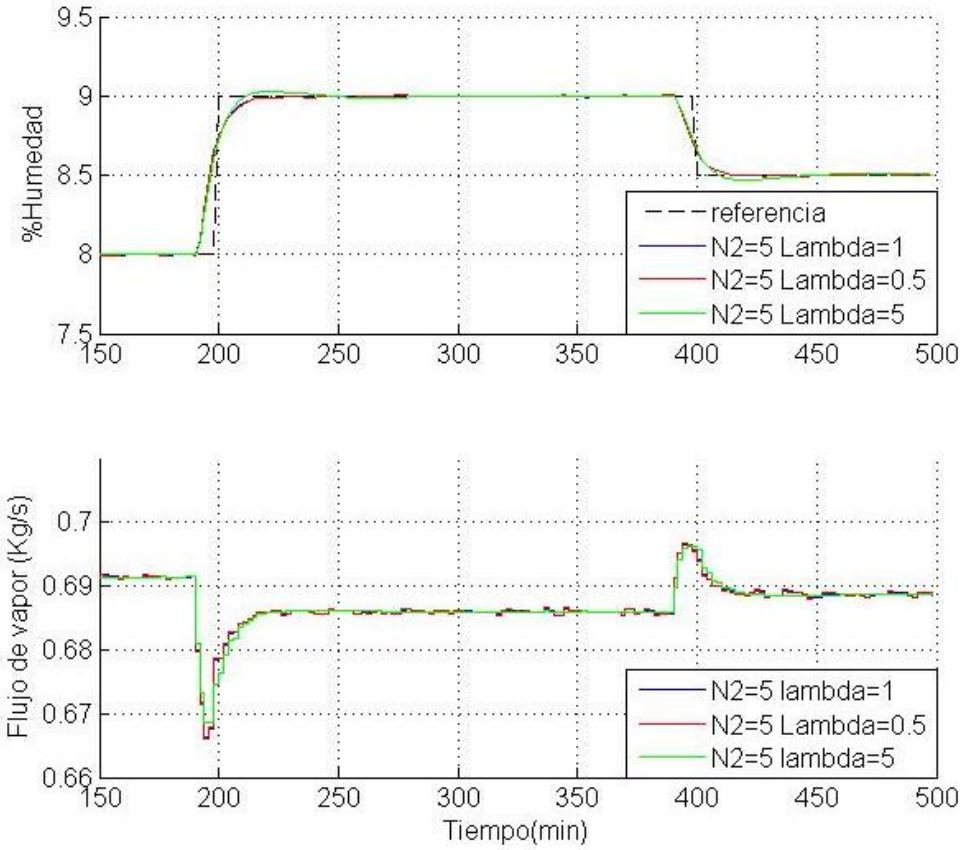

Figura 7 Tercera prueba de simulación del control del secador de harina de pescado.

Capítulo 4.
Control predictivo basado en modelos en espacio de estados (MPC) del modelo de módulo de cuatro tanques

4.1 Control del modelo de cuatro tanques.

4.1.1 Modelo del módulo de cuatro tanques.

Para la modelación de este módulo de varias entradas y salidas se han usado las ecuaciones de balance de masa, y la ecuación de velocidad de descarga por un agujero de un tanque abierto. Se ha seguido la formulación de los modelos usados en [27], [28] y [29], pero con una configuración de conexiones de tanques diferentes. En las ecuaciones 4.1 y 4.3 se muestran estas relaciones para el caso de un tanque (ver Figura 8).

$$\frac{dM}{dt} = \dot{m}_e - \dot{m}_s \tag{4.1}$$

Donde $\dot{m}_e$ y $\dot{m}_s$ son los valores de los flujos másicos de entrada y de salida respectivamente. La variable M es la masa de agua en el tanque. Considerando que el tanque es de área transversal constante con respecto a su altura y dividiendo la ecuación (4.1) entre la densidad del agua se puede reescribir de la siguiente manera, A es el área del tanque y h es la altura del tanque, como se muestra a continuación:

$$A\frac{dh}{dt} = Q_e - Q_s \tag{4.2}$$

Donde Q_e y Q_s son los valores de los flujos volumétricos de entrada y de salida respectivamente.

Se utiliza la ecuación de la velocidad de descarga (V_d) por un agujero de un tanque abierto.

$$V_d = \sqrt{2gh} \tag{4.3}$$

El valor de g es la aceleración de la gravedad.

La ecuación (4.3) se multiplica por el área del agujero de descarga (a) y por el factor de descarga (C_d) y se obtiene la ecuación (4.4) del caudal de salida Q_s.

$$\mathrm{Q_s} = \mathrm{X}C_d a\sqrt{2\mathrm{gh}} \tag{4.4}$$

El factor de descarga depende del diámetro y de la geometría del agujero y la altura descarga. Puede tomar valores entre 0 y 1. X es un factor de ajuste que se determinó experimentalmente.

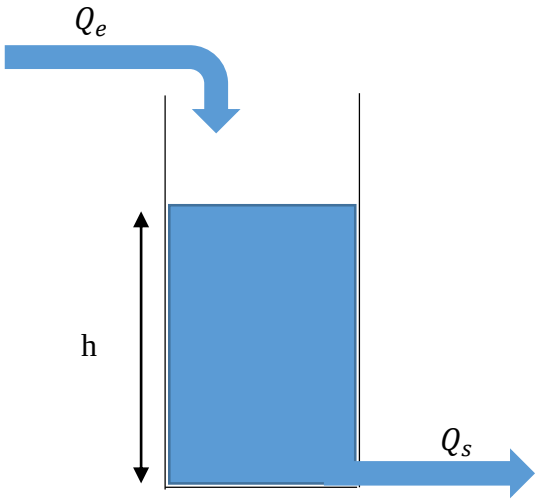

Figura 8. Representación de un tanque con una entrada y una salida. Fuente: Elaboración propia.

En la Figura 9 se muestra un diagrama simplificado del módulo de cuatro tanques, este consta de cuatro tanques en los cuales se mide el nivel, por medio de transmisores de nivel y un tanque de almacenamiento. Además cuenta con dos bombas de corriente continua.

Las ecuaciones siguientes describen el sistema:

$$A_1\frac{dh_1}{dt} = Q_{B2,T1} + Q_{T1,T3} - Q_{T1,T2}$$

$$A_2\frac{dh_2}{dt} = Q_{B1,T2} + Q_{T1,T2} - Q_{T2,T0}$$

$$A_3\frac{dh_3}{dt} = Q_{B1,T3} - Q_{T1,T3} - Q_{T3,T4}$$

$$A_4\frac{dh_4}{dt} = Q_{B2,T4} + Q_{T3,T4} - Q_{T4,T0} \tag{4.5}$$

Donde A_i y h_i son el área transversal y la altura de la columna de agua en el tanque i. El caudal $Q_{Bi,Tj}$ entre la bomba i y el tanque j y el caudal $Q_{Ti,Tj}$ entre el tanque i y el tanque j. El tanque 0 es el de almacenamiento como se ve en la Figura 9 .Para las simulaciones e implementación se usó la configuración con las válvulas V101, V203 y V103 cerradas, las demás estuvieron abiertas.

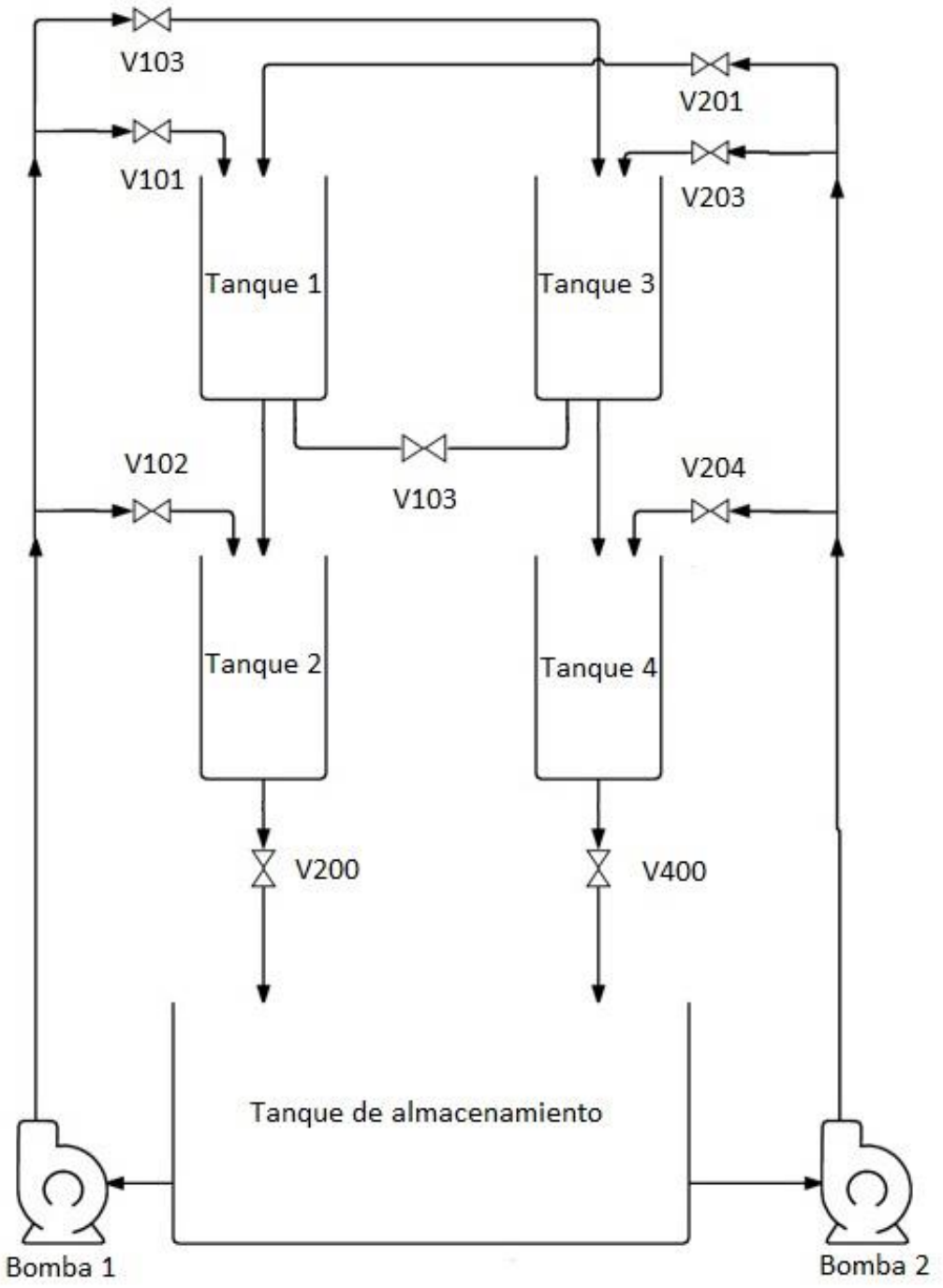

Figura 9. Diagrama del módulo de cuatro tanques. Fuente: Elaboración propia.

Las ecuaciones que describen los caudales $Q_{Bi,Tj}$ y $Q_{Ti,Tj}$ se describen a continuación:

$$Q_{Bi,Tj} = k_{Bi,Ti}\sqrt{w_{Bi}v_{Bi}^2 + k_{Bi}} \tag{4.6}$$

Los parámetros $w_{Bi,Tj}$, $k_{Bi,Tj}$ y K_{Ti} son parámetros que aproximan el caudal de la bomba i que recibe del tanque j. La relación entre caudal de la bomba y tensión de alimentación se ha obtenido heurísticamente.

En la ecuación (4.7), C_d es el coeficiente de descarga del tanque i hacia el tanque j, que se ha asumido como 0.64 para todas las tuberías de descarga, y a_i es el área del agujero de descarga. $X_{Ti,Tj}$ es un factor de ajuste que se determine experimentalmente.

$$Q_{Ti,Tj} = X_{Ti,Tj} C_d a_i \sqrt{2gh_i} \tag{4.7}$$

Para el caso $i = 1$ y $j = 3$ la ecuación cambia a (4.8), porque la dirección del caudal entre estos dos tanques cambian con la diferencia de alturas, El caudal fluye desde el tanque de mayor altura hacia el de menor altura .Al estar la válvula V103 (ver Figura 9) cerrada el caudal de la ecuación (4.8) se hace cero para la configuración actual.

$$Q_{T3,T4} = signo(h_1 - h_3) X_{T1,T3} a_3 \sqrt{2g|h_1 - h_3|} \tag{4.8}$$

4.1.1.1 Ajuste de parámetros del modelo no lineal.

Para la estimación de los parámetros del modelo se utilizó "prediction error minimization algorithm (pem)" disponible en la librería de optimización de Matlab como la función pem.m. En la Tabla 4-1 se muestran los parámetros estimados para el modelo no lineal. Las pruebas experimentales para el cálculo de estos parámetros se realizaron en una interfaz gráfica programada en Wincc. Se describe esta interfaz en el Capítulo 5.

Tabla 4-1 Parámetros estimados para el modelo no lineal. Fuente: Elaboración propia.

Símbolo	Nombre	Valor
A_1	Área del tanque 1	100 cm
A_2	Área del tanque 2	100 cm
A_3	Área del tanque 3	100 cm
A_4	Área del tanque 4	100 cm
a_1	Área de tubería de descarga del tanque 1	0.6362 cm^2
a_2	Área de tubería de descarga del tanque 2	0.6362 cm^2
a_3	Área de tubería de descarga del tanque 3	0.6362 cm^2
a_4	Área de tubería de descarga del tanque 4	0.6362 cm^2
C_d	Coeficiente de descarga de las tuberías	0.64
x_{12}	Parámetro de ajuste del caudal de descarga del tanque 1 al 2	1.1844
x_{2a}	Parámetro de ajuste del caudal de descarga del tanque 2 al tanque del almacenamiento.	10377
x_{34}	Parámetro de ajuste del caudal de descarga del tanque 3 al 4	1.2846
x_{4a}	Parámetro de ajuste del caudal de descarga del tanque 4 al tanque del almacenamiento.	1.1032
w_2	Parámetro de ajuste del caudal de la bomba 2	49.8846
k_2	Parámetro de ajuste del caudal de la bomba 2	-15809
w_1	Parámetro de ajuste del caudal de la bomba 1	49.8619
k_1	Parámetro de ajuste del caudal de la bomba 1	-14327
k_{12}	Parámetro de ajuste del caudal de la bomba 1 al tanque 2	0.1561
k_{13}	Parámetro de ajuste del caudal de la bomba 1 al tanque 3	1
k_{21}	Parámetro de ajuste del caudal de la bomba 2 al tanque 1	1
k_{24}	Parámetro de ajuste del caudal de la bomba 2 al tanque 4	0.0487

Por lo tanto el modelo queda descrito por las siguientes ecuaciones diferenciales:

$$A_1 \frac{dh_1}{dt} = k_{21}\sqrt{w_2 v_2^2 + k_2} - x_{12} C_d a_1 \sqrt{2gh_1}$$

$$A_2 \frac{dh_2}{dt} = k_{12}\sqrt{w_1 v_1^2 + k_1} + x_{12} C_d a_1 \sqrt{2gh_1} - x_{2a} C_d a_2 \sqrt{2gh_2}$$

$$A_3 \frac{dh_3}{dt} = k_{13}\sqrt{w_1 v_1^2 + k_1} - x_{34} C_d a_3 \sqrt{2gh_3}$$

$$A_4 \frac{dh_4}{dt} = k_{24}\sqrt{w_2 v_2^2 + k_2} + x_{34} C_d a_3 \sqrt{2gh_3} - x_{4a} C_d a_4 \sqrt{2gh_4} \qquad (4.9)$$

En el Anexo A1 se presenta el modelo de cuatro tanques acoplado, programado como una función de Matlab.

4.1.1.2 Modelo en espacio de estados.

Teniendo el modelo en espacios de estados:

$$\dot{x}(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) \qquad (4.10)$$

Reemplazando variables y extendiendo los vectores del modelo se obtiene lo siguiente:

$$\begin{bmatrix} \dot{x}_1(k+1) \\ \dot{x}_2(k+1) \\ \dot{x}_3(k+1) \\ \dot{x}_3(k+1) \end{bmatrix} = \begin{bmatrix} 0.9054 & 0.0015 & 0.0032 & 0.0069 \\ -0.0837 & 0.8904 & -0.0111 & -0.0098 \\ -0.1647 & -0.1520 & 0.9023 & 0.0287 \\ -0.1123 & 0.1520 & -0.0674 & 0.8227 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} + \begin{bmatrix} 0.0065 & 0.0014 \\ 0.0058 & -0.0041 \\ 0.0108 & -0.0025 \\ 0.0038 & 0.0115 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} 27.5115 & -98.5564 & 52.4028 & -23.3574 \\ 65.6542 & 0.7833 & -30.3947 & -10.2595 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{bmatrix} \qquad (4.11)$$

El modelo anterior se ha calculado por medio de identificación en espacio de estados usando la función n4sid disponible en Matlab y los datos fueron obtenidos mediante una prueba de identificación en el módulo (Ver Capítulo 5).

4.1.2 Sintonización del controlador por simulación.

Se realizaron seis pruebas para diferentes horizontes de predicción y valores de peso sobre la variación de la variable de control. Se han tomado en cuenta las restricciones en las señales de control y en sus variaciones, además también en las señales de salida. Se han tomado como salidas las alturas de los tanques inferiores (tanques 2 y 4) y como entradas las tensiones de las bombas. El programa para el controlador predictivo se ha elaborado en Matlab (ver Anexo A2).

Las pruebas se hicieron para seguimiento de referencia y ante rechazo de disturbios. Los disturbios se producen sobre una de las salidas al variar la referencia de la otra.

En el primer grupo de pruebas se fijó lambda y se varió el horizonte de predicción en 20, 30 y 50. Como se detalla a continuación:

Prueba 1	Prueba 2	Prueba 3
N2=20; Nu=3; Lambda=5.0;	N2=30; Nu=3; Lambda=5.0;	N2=50; Nu=3; Lambda=5.0;

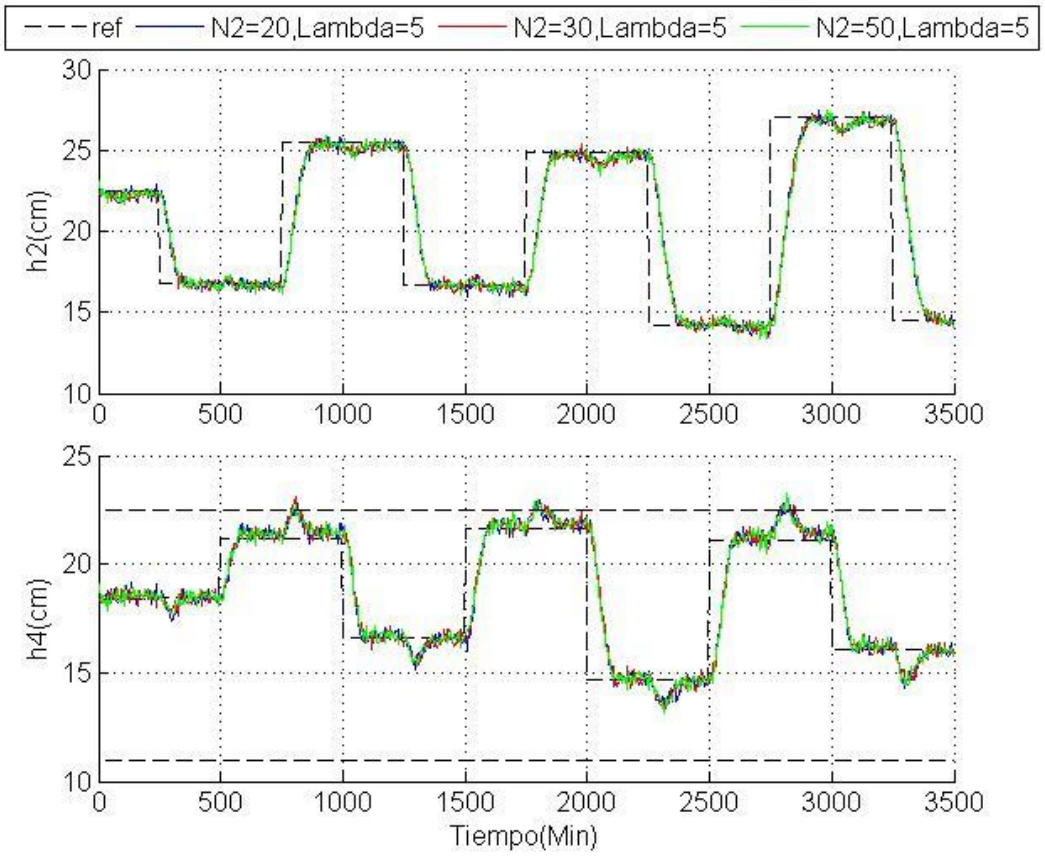

Figura 10. Salidas de las pruebas 1, 2 y 3. Fuente: Elaboración propia

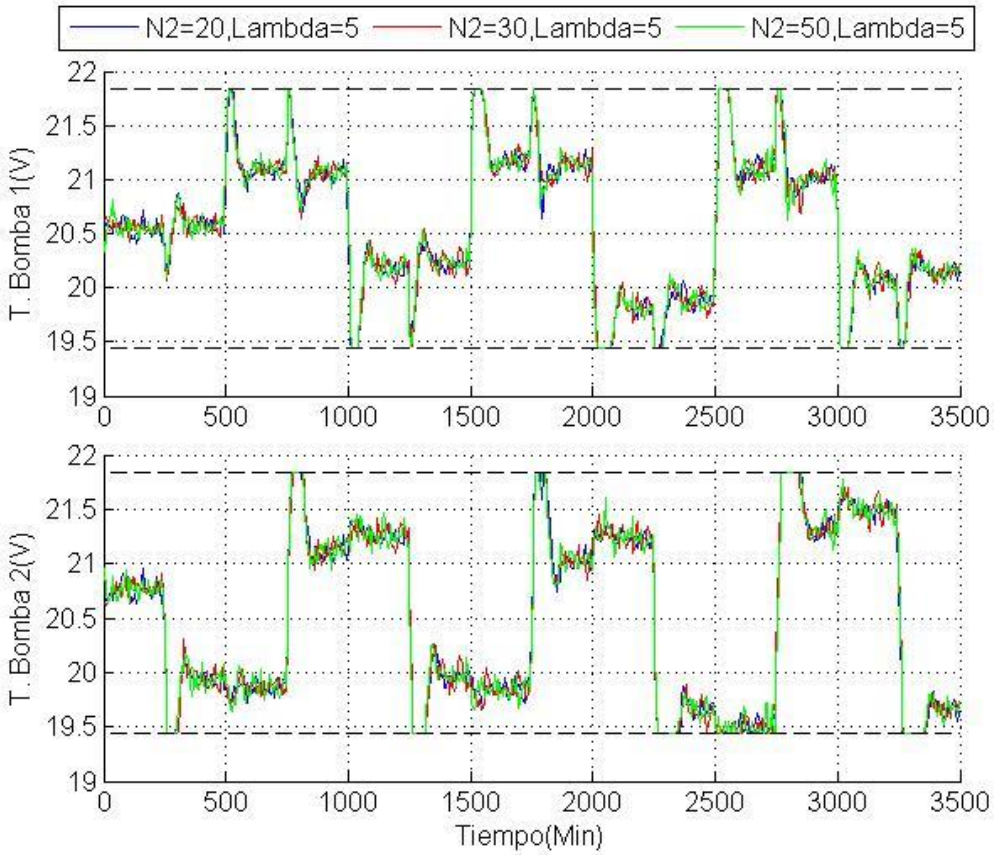

Figura 11. Entradas de las pruebas 1, 2 y 3. Fuente: Elaboración propia.

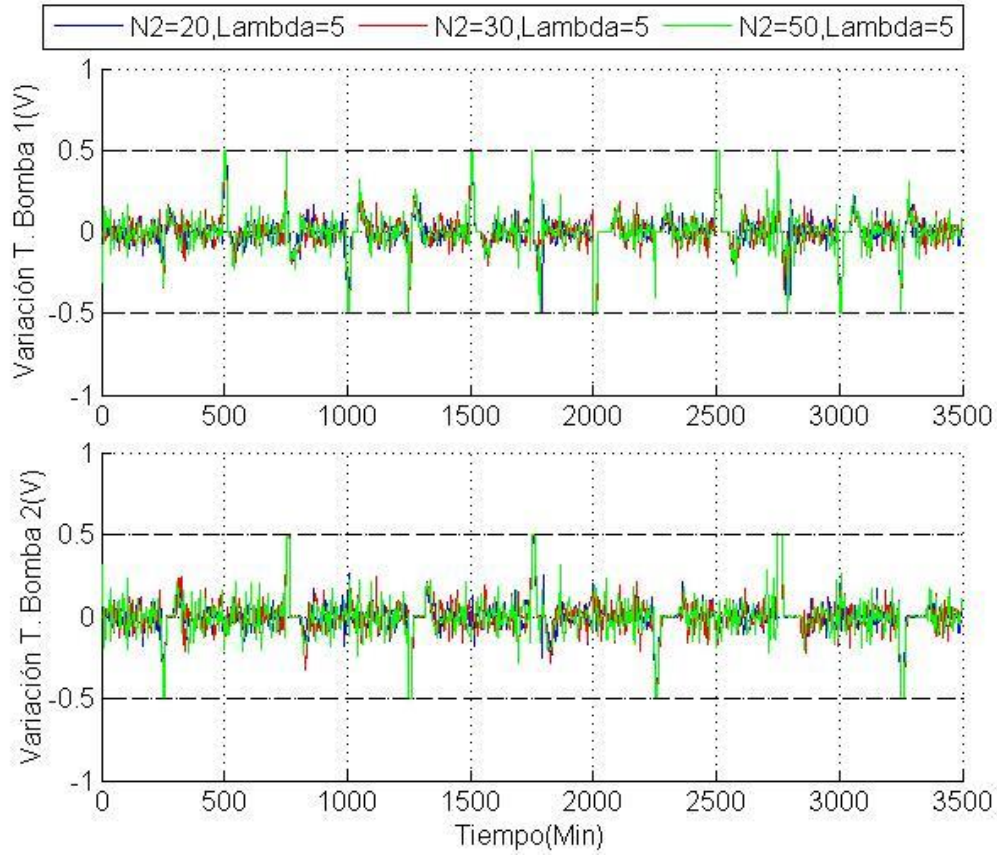

Figura 12. Variaciones de las entradas de las pruebas 1, 2 y 3. Fuente: Elaboración propia.

En el segundo grupo de pruebas se fijó el horizonte de predicción y se varió lambda en 2, 10 y 20. Como se detalla a continuación.

Prueba 4	Prueba 5	Prueba 6
N2=30; Nu=3; Lambda=2.0;	N2=30; Nu=3; Lambda=10.0;	N2=30; Nu=3; Lambda=20.0;

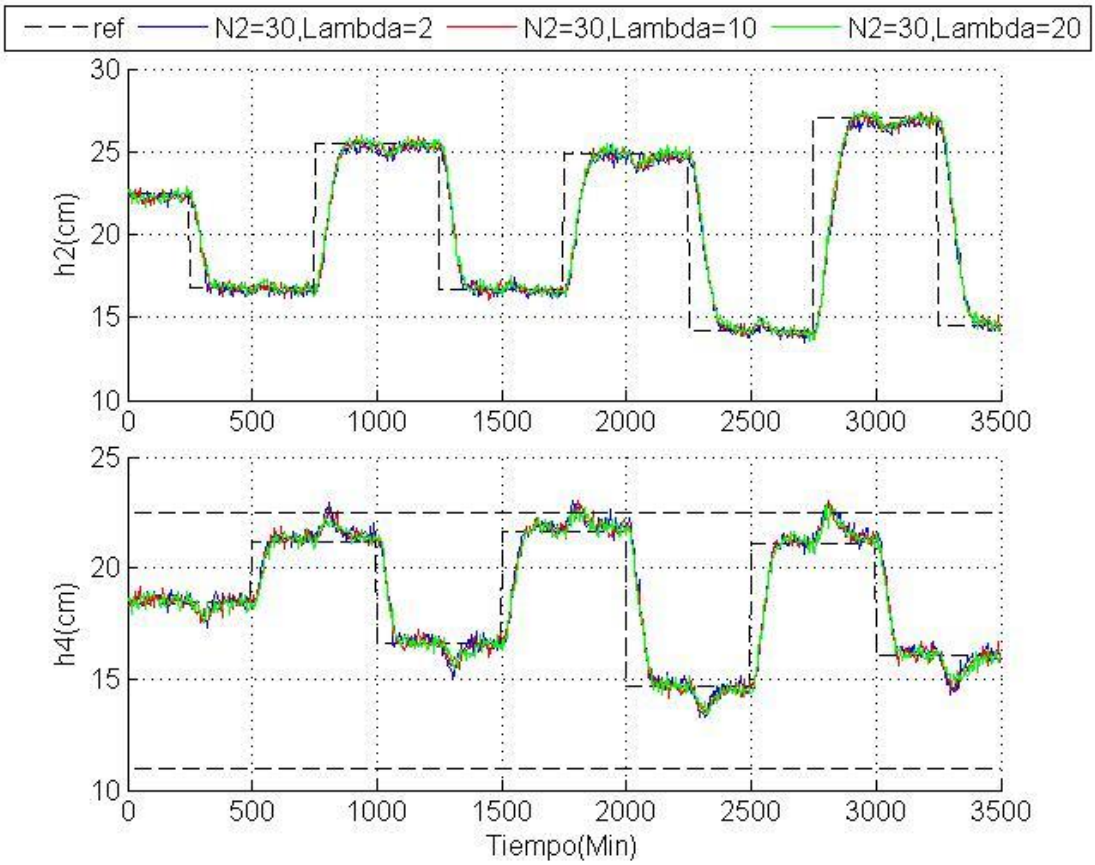

Figura 13. Salidas de las pruebas 4, 5 y 6. Fuente: Elaboración propia.

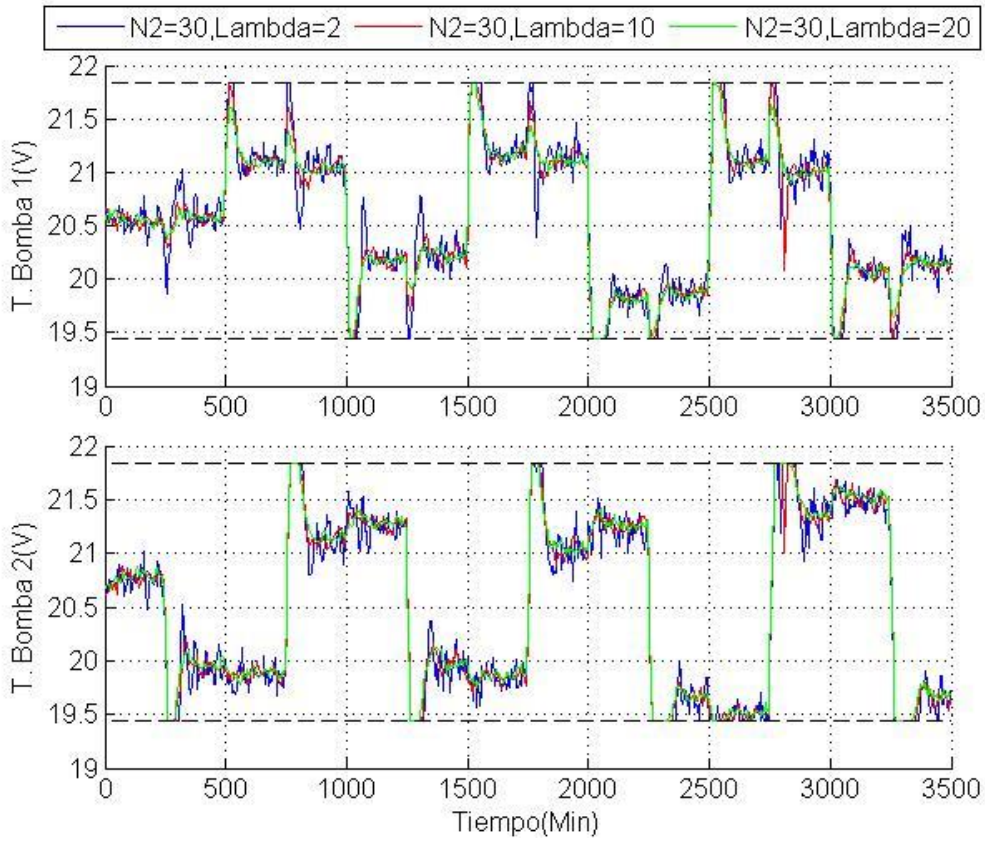

Figura 14. Entradas de las pruebas 4, 5 y 6. Fuente: Elaboración propia.

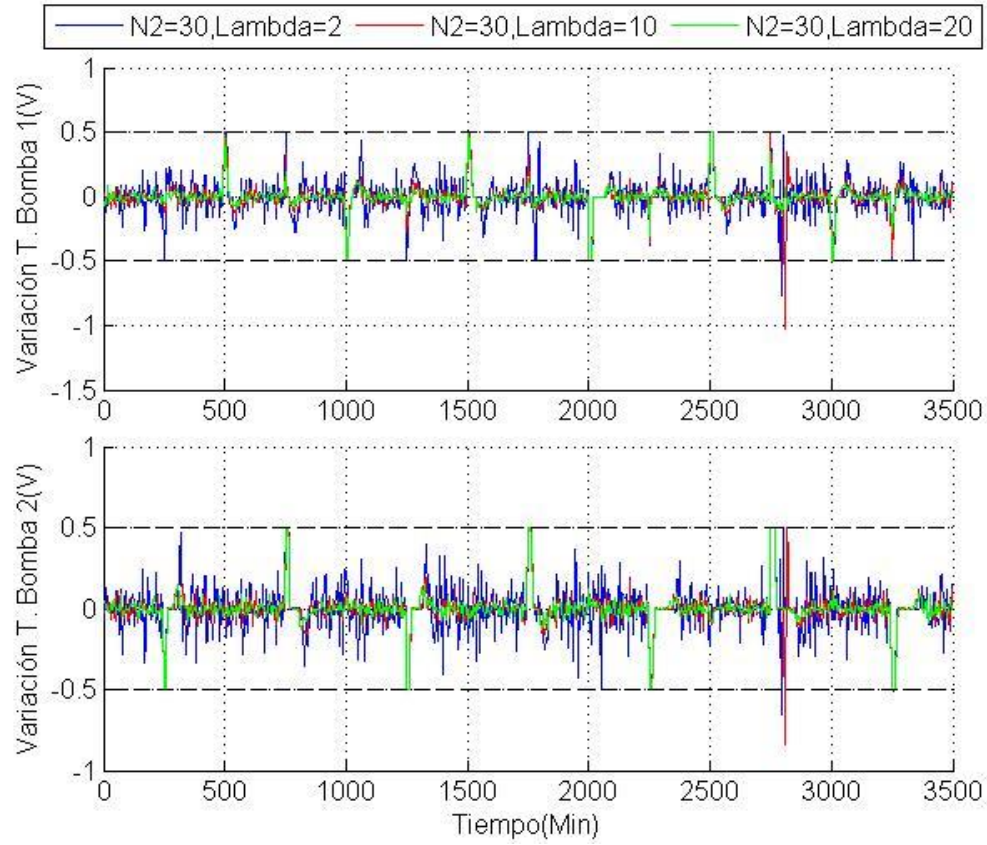

Figura 15. Variaciones de las entradas de las pruebas 4, 5 y 6. Fuente: Elaboración propia.

En cuanto a tiempo de respuestas ante cambios de referencias, presencia de disturbios y violación de las restricciones la Prueba 6 presenta mejores resultados. Sin embargo muestra el problemas en la violación de restricciones en la salida (ver Figura 13). Este problema también se presenta en las demás pruebas (ver Figura 10 y Figura 13).

En las simulaciones el problema de programación cuadrática ha tenido una convergencia lenta. En algunos tiempos de muestreo como en los casos 4 y 5 el número de iteraciones superó en número máximo de iteraciones de 1000. En las pruebas 5 y 3 el número de iteraciones no superó las 300 y en los casos 1 y 6 no superó las 200.

Desde el punto de vista de la implementación en el PLC, las sintonizaciones 1 y 6 tienen la mayor probabilidad de ser exitosas.

Capítulo 5.
Implementación de sistema de identificación en PLC

5.1 Descripción del hardware.

5.1.1 PLC VIPA CPU 314 6CG03.

Para el desarrollo de las pruebas en el módulo experimental se usó un PLC VIPA-314-6CG03 (ver Figura 16), cuenta con un CPU 314SC/DPM - SPEED7 technology, 24 entradas digitales, 16 salidas digitales, 4 entradas analógicas, 1 entrada para PT100 y 2 salidas analógicas. Tiene una memoria de trabajo de 128 kBytes expandible hasta 1 Mbyte (50% para programa / 50% para datos). Información técnica más detallada se encuentra en [30].

Figura 16. PLC VIPA 314 6CG03. Fuente: Elaboración propia.

5.1.2 Descripción del módulo de cuatro tanques.

Este módulo consta de cuatro tanques de acrílico de sección cuadrada de 10 cm de lado y 60 cm de altura, distribuidos como se muestra en la Figura 17. Cada uno tiene instalado un transmisor de presión que se usará para estimar el nivel de cada tanque. Consta también de un tanque de almacenamiento de acero inoxidable de 30x40 cm de base y 30 cm de altura, conectado a dos bombas DC 24V.

Los tanques superiores descargan hacia los tanques inferiores del mismo lado. A su vez estos descargan hacia el tanque de almacenamiento. Existe también una tubería de conexión entre los tanques superiores.

Figura 17. Fotografía del módulo de cuatro tanques acoplados. Fuente: Elaboración propia.

Estas bombas de la marca *Johnson Pump*, del tipo CM 30P7-1 (ver Figura 18), tienen las siguientes características, caudal de 22.5 l/min a 15 kPa, Potencia de 26 W y un diámetro de salida de 20 mm.

La bomba del lado izquierdo se conecta a los dos tanques del lado izquierdo y al tanque superior del lado derecho, mientras que la del lado derecho a los dos tanques del lado derecho y al tanque superior del lado izquierdo como se muestra en la Figura 17.

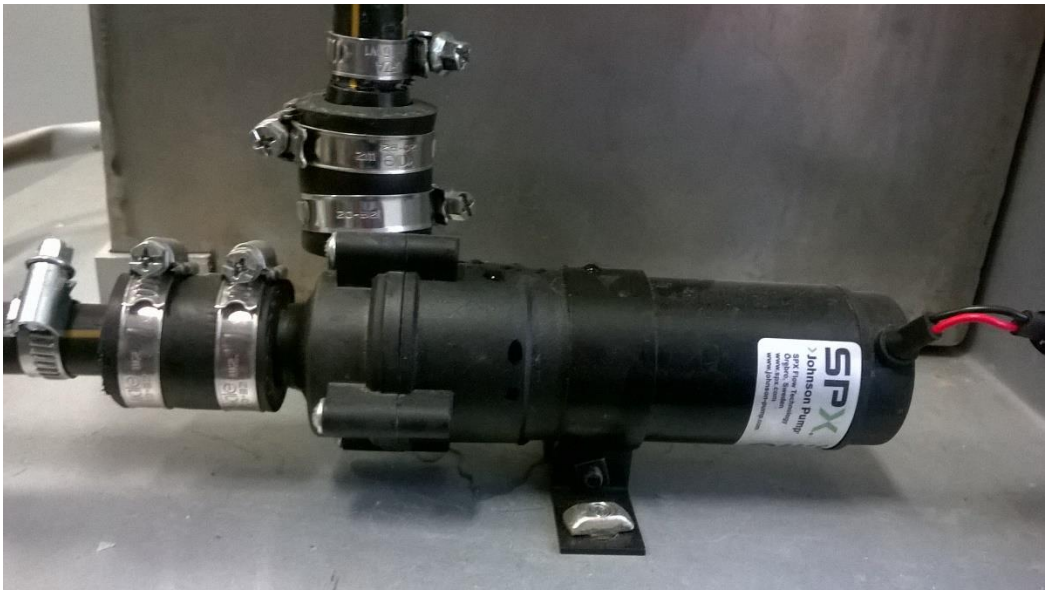

Figura 18. Bomba del lado izquierdo del módulo de cuatro tanques acoplados. Fuente: Elaboración propia.

En los cuatro tanques de acrílico están instalados transmisores de presión marca WIKA, modelo S-10 (ver Figura 19), que tiene las siguientes características: presión entre 0 y 0.1 bar, salida de 4-20 mA y alimentación de 10 a 30 V DC.

Figura 19. Transmisor de presión WIKA modelo S-10. Fuente: Elaboración propia.

5.2 Softwares usados.

Para la programación del PLC se usó el STEP 7 professional 2006 SR6, El cual nos permite programar en lenguaje de texto estructurado, lenguaje que se necesita para resolver el problema de programación cuadrática. Se usó el SIMATIC WinCC V7.0 SP2 para el desarrollo del SCADA, por ser de fácil integración con el STEP 7 professional 2006 SR6.

5.3 Programación del PLC e interfaz hombre-máquina.

A continuación se hará una descripción de los aspectos más importantes de la programación del PLC y de la interfaz de hombre-máquina.

5.3.1 Bloques de programación

Para la implementación del controlador predictivo en el PLC se han usado los siguientes bloques de programación. Se han incluido en esta sección los bloques de datos, funciones y bloques de organización necesarios para la implementación futura de un controlador predictivo basado en modelos (MPC). A continuación se presenta una descripción de los bloques usados.

5.3.1.1 Bloques de datos

Para un manejo más ordenado de la programación se han dividido los datos en seis bloques de acuerdo a su función.

- DATA_BLOCK DB1:
  Almacena los parámetros del controlador predictivo y el número de entradas, salidas y estados (Ver Anexo B1).
- DATA_BLOCK DB2:
  Almacena las matrices necesarias para la programación cuadrática y la actualización de matrices (Ver Anexo B2).
- DATA_BLOCK DB3:
  Almacena los datos el proceso, es decir, los valores de las entradas y salidas (Ver Anexo B3).
- DATA_BLOCK DB4:
  Se almacenan los BITs para el manejo del HMI (Ver Anexo B4).
- DATA_BLOCK DB5:
  Se almacenan los parámetros para el ajuste lineal de las entradas al proceso, alturas (Ver Anexo B5).
- DATA_BLOCK DB6:
  Almacena los datos necesarios para realizar la prueba de identificación (Ver Anexo B6).

5.3.1.2 Funciones

- Función FC1:
  Realiza la solución del problema de programación cuadrática para el cálculo de las entradas óptimas (Ver Anexo B7).
- Función FC2:
  Realiza la actualización de las matrices para el problema de programación cuadrática de la función FC1 (Ver Anexo B8).

- Función FC11:
  Función que realiza el escalamiento de una señal de sensor (Ver Anexo B12).
- Función FC12:
  Función que realiza el des-escalamiento de una señal de salida (Ver Anexo B13).
- Función FC13:
  Función que ajusta a una recta el valor de escalados de salida de la función FC11 (Ver Anexo B11).
- Función FC14:
  Función que realiza el escalamiento y ajuste a una recta de todos los sensores del módulo (Ver Anexo B14).
- Función FC15:
  Función que realiza el des-escalamiento de todas las salidas a las bombas (Ver Anexo B15).
- Función FC20:
  Función que genera las entradas escalones para la prueba de identificación del módulo (Ver Anexo B10).
- Función FC21:
  Función que genera las entradas PRBS para la prueba de identificación del módulo (Ver Anexo B11).

5.3.1.3 Bloques de organización

- Bloque de organización OB1:
  Bloque principal de organización de programa. Se programó las lógicas de inicio, parada y activación de estados (Modo Manual, Identificación y Automático) Ver Anexo B16.
- Bloque de organización OB35:
  Bloque de organización de programa que se ejecuta cíclicamente cada cinco segundos. Se programó la actualización de las salidas, las funciones que corresponden al modo automático (FC1 y FC2) y las funciones que corresponde al modo de prueba de identificación (FC20 y FC21).Ver Anexo B17.

5.3.2 Diseño de la interfaz con el usuario.

La interfaz se ha divido en tres secciones, la operación manual, la prueba de identificación y operación automática. A continuación se describen:

Modo Manual:

La interfaz (ver Figura 20) presenta cuatro visualizadores del nivel de líquido, dos recuadros de ingreso de la tensión de las bombas, botones de inicio y paro. Además los botones, auto e ident, que nos permiten cambiar de modo de operación.

Modo de prueba de identificación.

La interfaz (ver Figura 21) presenta cuatro visualizadores del nivel de líquido, cuadros de ingreso de la tensión nominal de las bombas, tensiones máximas y mínimas, botones de inicio y paro; y los botones, manual y auto, que nos permiten cambiar de modo de operación.

Además se tiene un selector del tipo de entrada que se desea para la identificación, se puede escoger entre una señal PRBS y una señal de escalones a diferentes frecuencias.

Modo automático.

La interfaz (ver Figura 22) presenta cuatro visualizadores del nivel de líquido, botones de inicio y paro; y los botones, manual e ident, que nos permiten cambiar de modo de operación. Además un botón de parámetros que abre otra ventana (ver Figura 23) para introducir los parámetros del controlador.

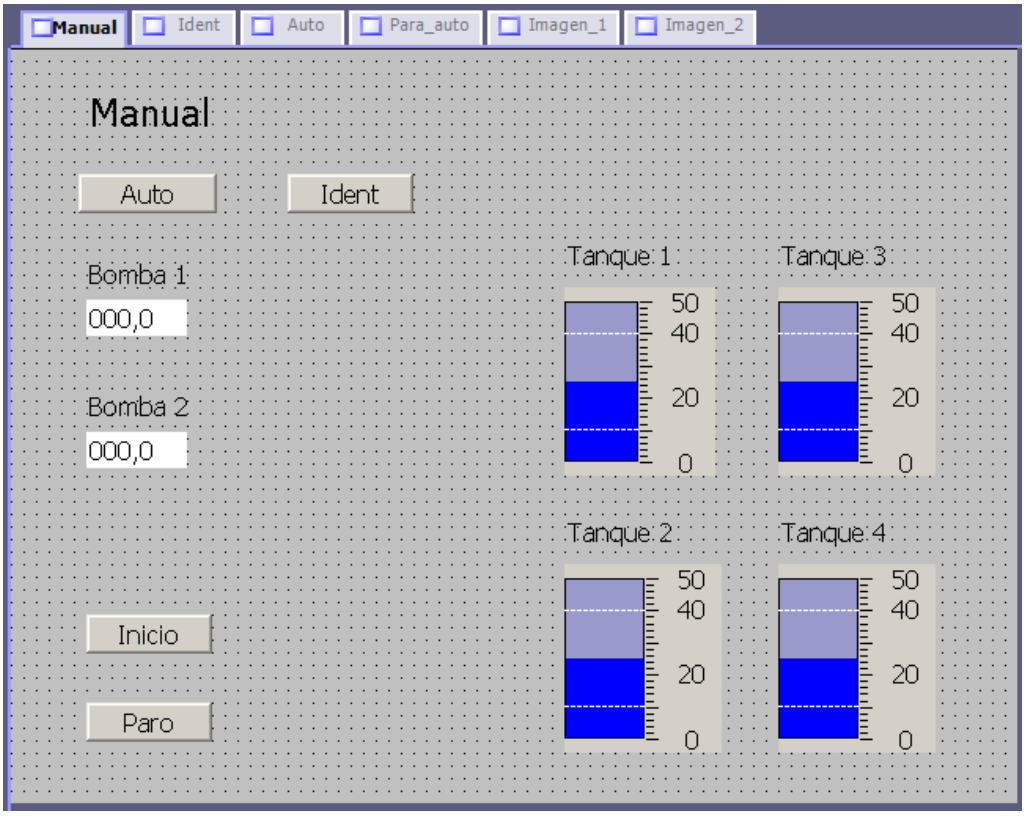

Figura 20. Interfaz gráfica del modo manual. Fuente: Elaboración propia.

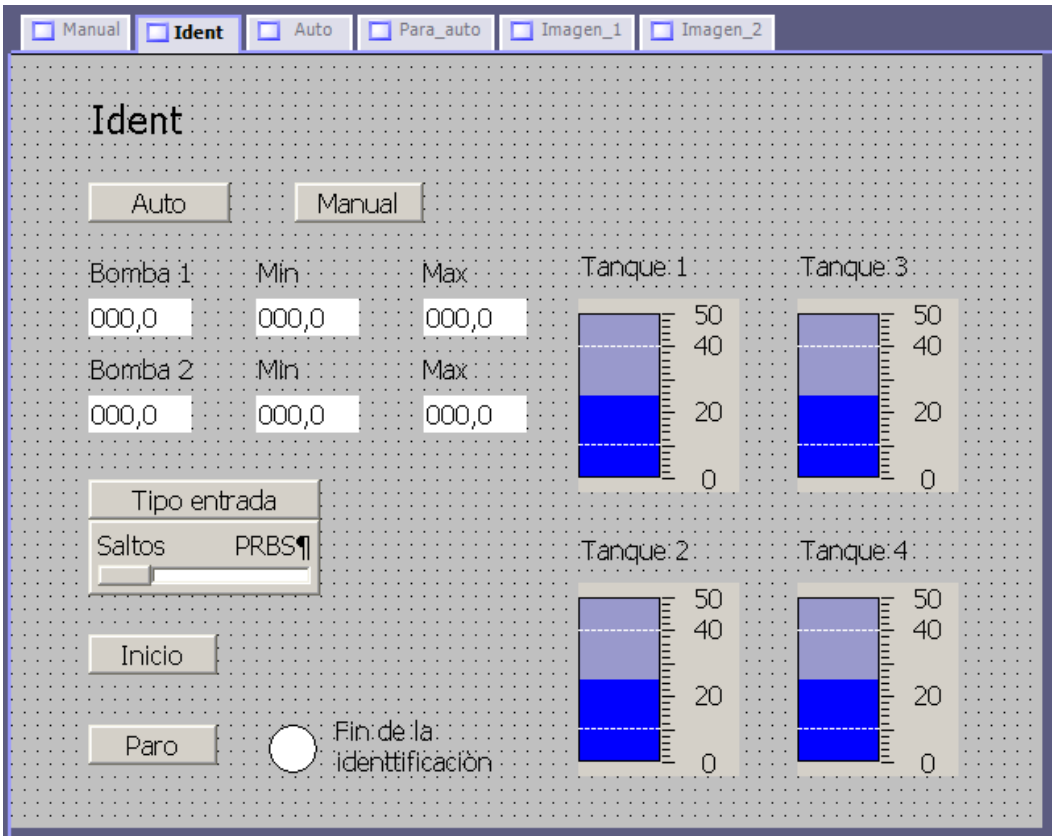

Figura 21. Interfaz gráfica modo prueba de identificación. Fuente: Elaboración propia.

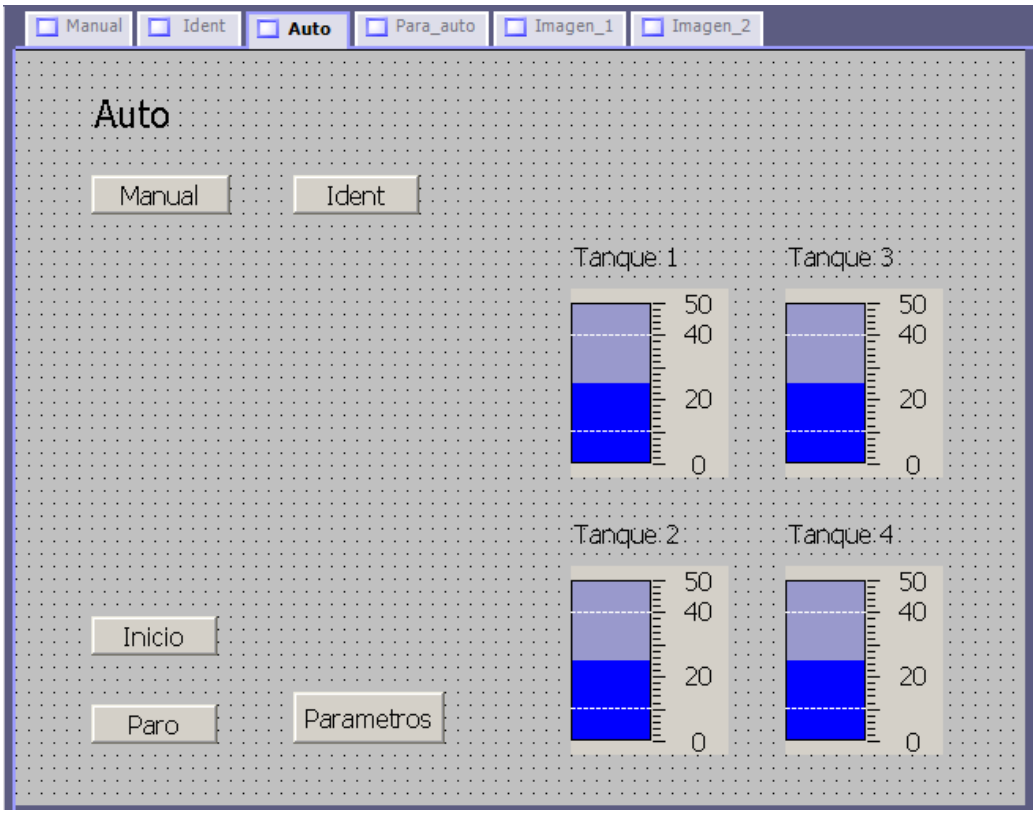

Figura 22. Interfaz gráfica modo automático. Fuente: Elaboración propia.

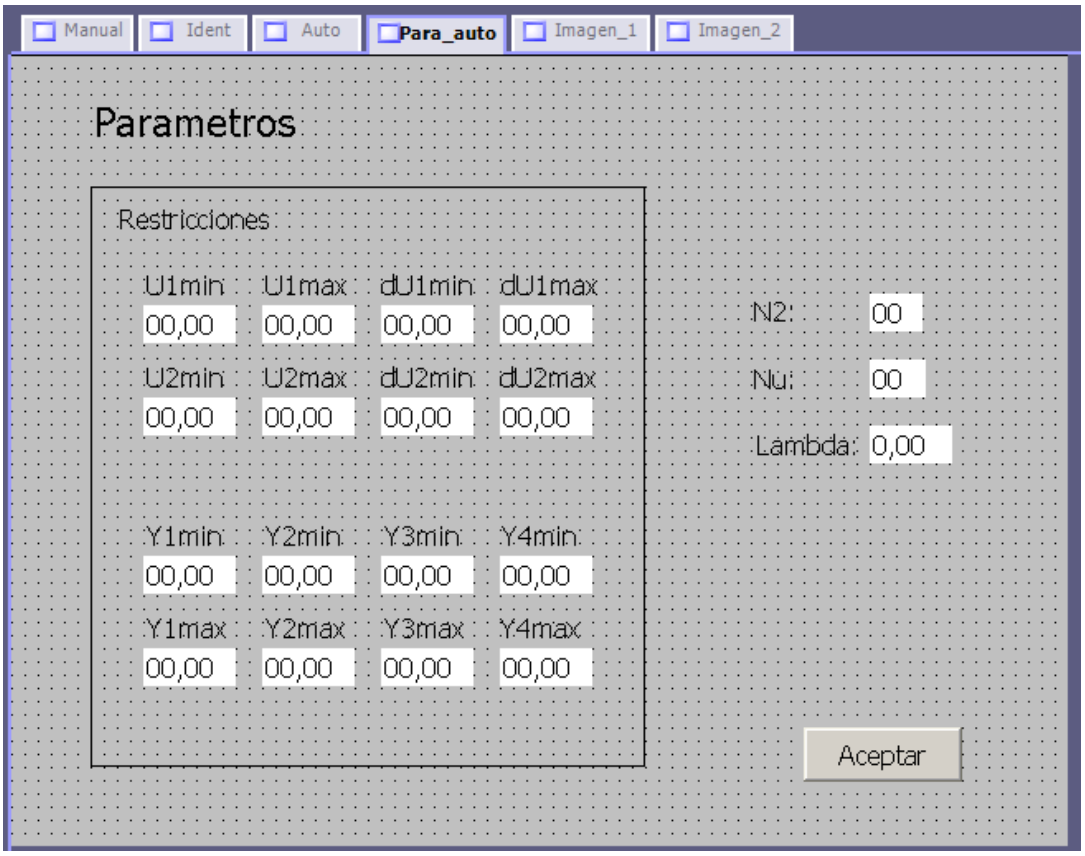

Figura 23. Interfaz de ingreso de parámetros del controlador. Fuente Elaboración propia.

5.4 Implementación y resultados de identificación.

Para la identificación y validación del modelo se han usado señales *pseudorandom binary sequence* (PRBS) que se muestran en la Figura 24, señales que fueron centradas en el punto 20.64 V para ambas bombas y tomando límite máximo +1.2 V y mínimo -1.2 V respecto al valor central. Se ha tomado la primera mitad de los datos para identificar y la segunda para validar el modelo. Se realizó una interfaz gráfica en Wincc para la introducción de la señales a las bombas de manera automática.

Se realizó una identificación en espacio de estados usando la función n4sid.m disponible en Matlab. Los resultados para identificación se muestran en la Figura 25 y en la Figura 26 los resultados de validación.

Se han tomado como entradas las tensiones de las bombas del módulo y como salidas las alturas del tanque dos y cuatro. El mejor ajuste de estos datos se obtiene con un modelo de cuatro estados, siendo para la altura del tanque dos, 61.48% para identificación y 61.71% para validación; la altura del tanque cuatro tuvo un ajuste de 84.22% y 84.13% para identificación y validación respectivamente. Para modelos de mayor grado la mejora no era significativa.

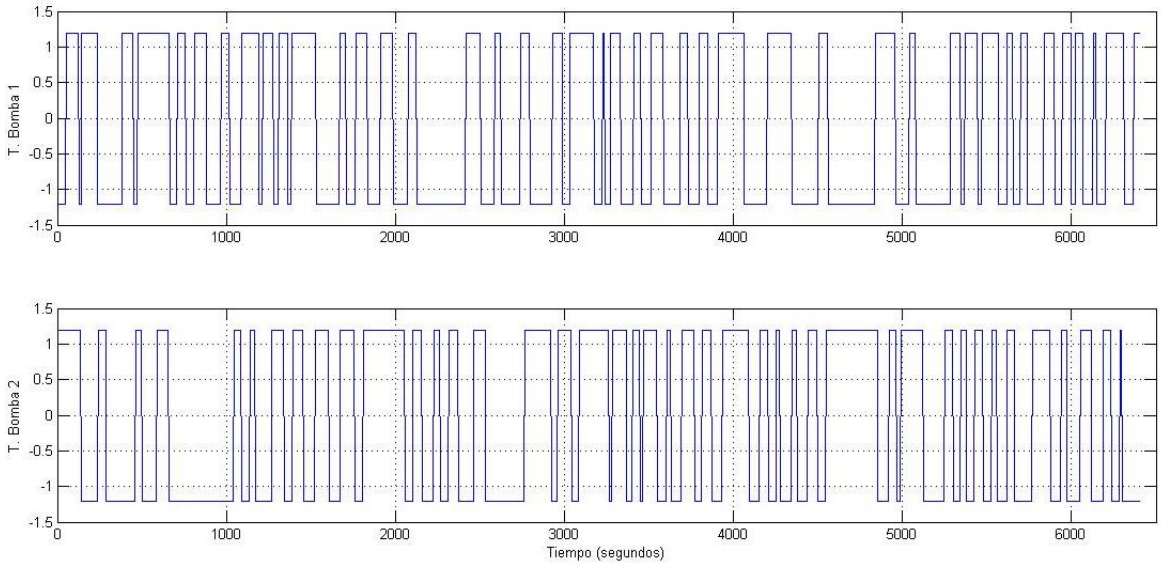

Figura 24 Entradas para la identificación y validación del modelo en espacio de estados. Fuente: Elaboración propia.

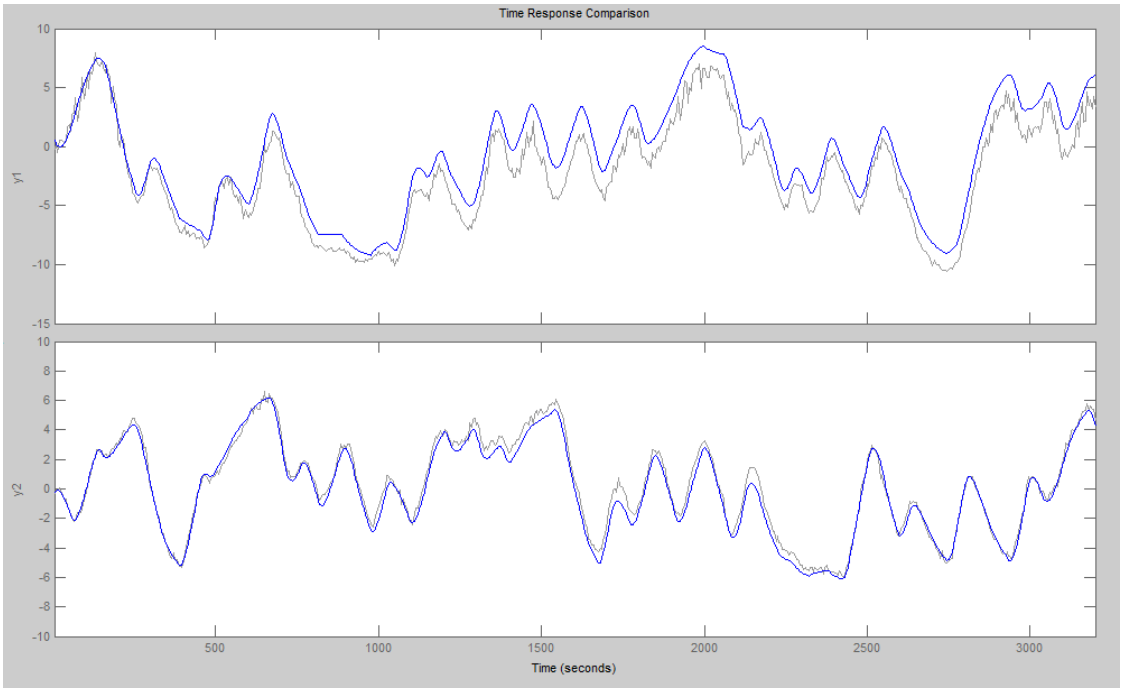

Figura 25 Gráficas comparativas de identificación del modelo en espacio de estados FIT h2-61.48% y h4-84.22%. Fuente: Elaboración propia.

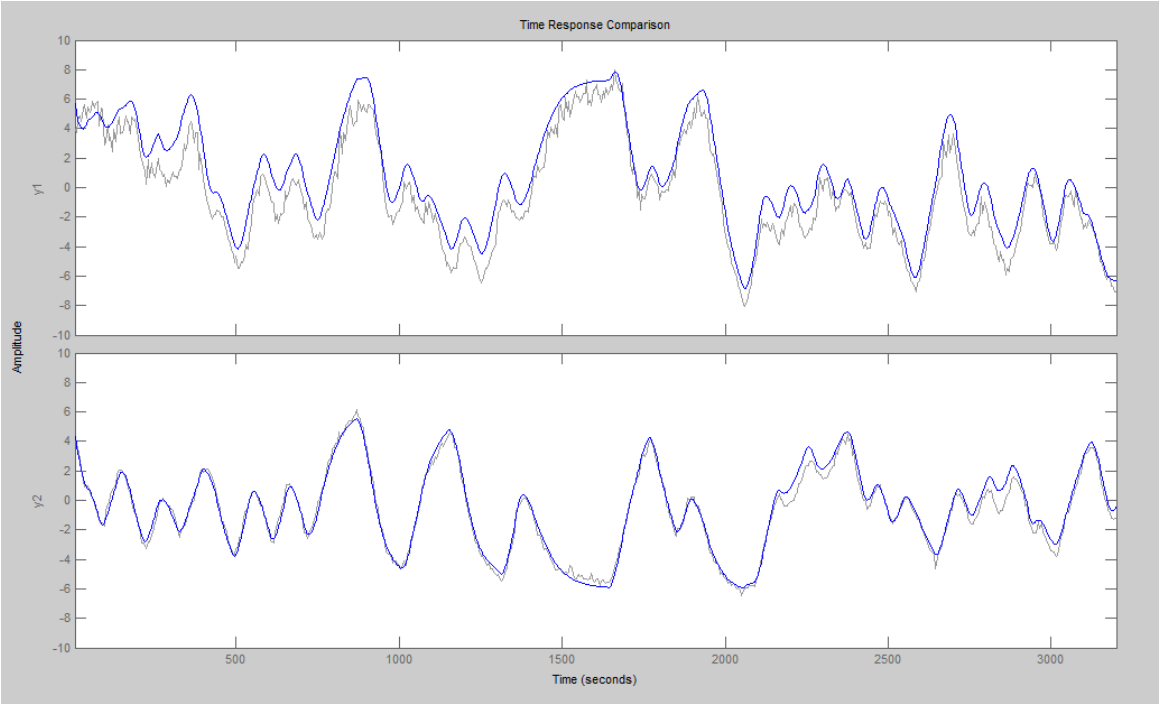

Figura 26 Gráficas comparativas de validación del modelo en espacio de estados FIT h2-61.71% y h4-84.13%. Fuente: Elaboración propia.

Conclusiones

El control predictivo basado en modelos es una estrategia de control avanzado que permite manejar sistemas múltiples entradas y múltiples salidas, tomando en cuenta sus restricciones de entrada, variación de la entrada y salida. Presenta la ventaja de incluir parámetros de sintonización intuitivos, que permiten al usuario sintonizar una velocidad de respuesta a lazo cerrado adecuada.

El modelo matemático basado en principios físicos del secador de disco rotatorio de harina de pescado ha sido exitosamente implementado en un controlador predictivo EPSAC. Se ha obtenido un buen desempeño, esto permitiría mejorar la calidad de harina en cuanto a su valor de humedad, reducir los costos en consumos de vapor y evitar los reprocesamiento del producto. Trabajos futuros incluirían la implementación de esta estrategia de control en un módulo de laboratorio.

Se ha realizado en control avanzado siguiendo la metodología de control predictivo MPC, usando un modelo en espacio de estados extendido del módulo de cuatro tanques acoplados que se ha aplicado a un modelo de principios físicos de dicho módulo. Trabajos futuros permitirán realiza el control de módulo real con un PLC.

El uso de modelos matemáticos para el diseño de controladores es una herramienta que permite reducir tiempos en las pruebas experimentales, conocer con cierta profundidad el comportamiento del sistema estudiado y reducir costos económicos al requerir menos pruebas experimentales.

La calidad de los controladores diseñados usando como herramienta un modelo de la planta y el desempeño de los controladores predictivo, si bien tienen muchas ventajas, dependen del modelo.

Se ha realizado la implementación de sistema para las pruebas de identificación múltiple entrada y múltiple salida del módulo de cuatro tanques acoplados, lo que permitió ajustar los parámetros del modelo de principios físicos e identificar un modelo en espacio de estados.

Se ha realizado el diseño de la estructura del programa que permitirá implementar el controlador predictivo MPC en la planta real. Esta estructura incluye bloques de datos, funciones y bloques de organización programados en STEP7.

Los estudios de maestría y el desarrollo de la tesis han permito de realizar publicaciones en congresos internacionales, un artículo en Francia, uno en Suiza y uno en Ecuador, que me permitió obtener experiencia en la redacción de artículos en congresos de alta exigencia.

Durante la maestría se realizó un viaje para asistir a la Escuela de Posgrado de Invierno, que se llevó a cabo en el Universidad Nacional del Litoral, Santa Fe, Argentina. Se estudiaron temas que ayudaron en la elaboración de la tesis, y me dio experiencia en una universidad extrajera.

Referencias

[1] R. Haber, R. Bars y U. Schmitz, Predictive Control in Process Engineering, Singapore: WILEY-VCH, 2011.

[2] W. Ipanaqué, C. Vallejos, R. De Keyser, A. Dutta, J. Oliden y R. Saavedra, «Modelo y Control Predictivo no Lineal de un Secador de Disco Rotatorio para la Producción de Harina de Pescado,» de *XV Congreso Latinoamericano de Control Automático*, Lima, 2012.

[3] W. Ipanaqué, R. De Keyser, A. Dutta, J. Oliden y J. Manrique, «Control no Lineal Iterativo Predictivo de un Evaporador en Obtención de Bio-etanol,» de *XV Congreso Latinoamericano de Control Automático*, Lima, 2012.

[4] W. Ipanaqué, J. Oliden, J. Manrique, A. Hernández, A. Dutta y R. De Keyser, «Nonlinear Predictive Control of an Evaporator for Bioethanol production,» de *European Control Conference*, Zurich, 2013.

[5] A. Hernández, R. De Keyser, J. Oliden y W. Ipanaqué, «Modeling and Nonlinear Model Predictive Control of a Rotary Disc Dryer for Fishmeal Production,,» de *European Control Conference* , strasbuorg, 2014.

[6] W. Ipanaqué, P. Gutarra, J. Manrique y J. Oliden, «Modeling and comparison of PID, GPC and MPC with state space controllers in a Coupled Tanks System,» de *Congreso Salesiano de Ciencia, Tecnología e Innovación para la Sociedad*, Guayaquil, 2015.

[7] E. J. Adam, *Conceptos Básicos de Control Predictivo,* Santa Fe, Argentina: Escuela de posgrado de invierno de la facultad de ingeniería química de la universidad nacional del litoral, Departamento de Ingeniería Química – FIQ – UNL, 2014.

[8] C. R. Cutler y B. L. Ramaker, «Dynamic Matrix Control- a computer algorithm,» *Proc. JACC,* 1980.

[9] R. M. C. De Keyser y A. R. Van Cauwenberghe, «Self-tuning predictive control,» *Journal A*, pp. 22, 167-174, 1981.

[10] R. M. C. De Keyser y A. R. Van Cauwenberghe, «Aplication of self-tuning predictive control,» *Journal A,* pp. 23, 1-10, 1982a.

[11] R. M. C. De Keyser y A. R. Van Cauwenberghe, «Typical application possibilities for self-tuning predictive control,,» *Proc. IFAC Symp on Identification and System Parameter Estimation,* 1982b.

[12] R. M. C. De keyser y A. R. Van Cauwenberghe, «Extende prediction adaptative control,» *Proc. IFAC Symp on Identification and System Parameter Estimation,* 1985.

[13] D. W. Clarke, C. Mohtadi y P. S. Tuffs, «Generalized Predictive Control Part I- The Basic Algorithm,» *Automatica,* pp. Vol. 23, 2, 137-148, 1987a.

[14] D. W. Clarke, C. Mohtadi y P. S. Tuffs, «Generalized Preidictive Control Part II- Extensions and Interpretation,» *Automatica,* pp. Vol. 23, 2, 149-160, 1987b.

[15] D. W. Clarke y C. Mohtadi, «Properties of Generalized Predictive Control,» *Automatica,* pp. Vol. 25, 6, 859-875, 1989.

[16] U. Diwekar, Introduction to applied optimization, New York: Springer, 2008.

[17] J. Sun, R. Freund y T. Magnanti, «MIT open course ware,» [En línea]. Available: http://ocw.mit.edu/courses/sloan-school-of-management/15-094j-systems-optimization-models-and-computation-sma-5223-spring-2004/lecture-notes/14solving_qp_art.pdf. [Último acceso: 1 10 2015].

[18] H. J. Ferreau, H. G. Bock y M. diehl, «An online active set strategy to overcome the limitations of explicit MPC,» *International Journal of Robust and Nonlinear Control,* vol. 18, nº 8, pp. 816-830, 2008.

[19] F. Bonne, M. Alamir y P. Bonnay, «Cornell University Library,» 24 6 2014. [En línea]. Available: http://arxiv.org/pdf/1406.6281v1.pdf. [Último acceso: 20 10 2015].

[20] P. Seman, M. Juhás, L. Tkác y M. Honek, «New possibilities of industrial programming software Swing-up and Stabilization of Reaction Wheel Inverted Pendulum via model predictive control using PLC,» de *International Conference on Process Control (PC)*, Štrbské Pleso, 2013.

[21] R. Fletcher, Practical Methods of Optimization, New York: JOHN WILEY & SONS, 1987.

[22] R. Freund, «MIT open course ware,» [En línea]. Available: http://ocw.mit.edu/courses/sloan-school-of-management/15-084j-nonlinear-programming-spring-2004/lecture-notes/lec14_int_pt_mthd.pdf. [Último acceso: 2 11 15].

[23] B. Huyck, L. Callebaut, F. Logist, H. . J. Ferreau, M. Diehl, J. De Brabanter, J. Van Impe y B. De Moor, «Implementation and Experimental Validation of Classic MPC on Programmable Logic Controllers,» de *20th Mediterranean Conference on Control & Automation (MED)*, Barcelona, 2012.

[24] B. Huyck, J. De Brabanter, B. De Moor, J. F. Van Impe y F. Logist, «Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study,» *ControlEngineeringPractice,* vol. 28, pp. 34-48, 2014.

[25] B. Huyck, H. J. Ferreau, M. Diehl, J. De Brabanter, J. F. Van Impe, B. De Moor y F. Logist, «Towards Online Model Predictive Control on a Programmable Logic Controller: Practical Considerations,» *Mathematical Problems in Engineering,* 2012.

[26] C. Hildreth, «A quadratic programming procedure,» *Naval Research Logistics Quarterly,* vol. 4, nº 1, p. 79–85, 1957.

[27] I. Alvarado, D. Limon, D. Muñoz de la Peña, J. M. Maestre, M. A. Ridao, H. Scheu, W. Marquardt, R. R. Negenborn, B. De Schutter, F. Valencia y J. Espinosa, «A comparative analysis of distributed MPC

techniques applied to the HD-MPC four-tank benchmark,» *Journal of Process Control*, vol. 21, nº 1, p. 800–815, 2011.

[28] A. Ferramosca, D. Limon, I. Alvarado y E. F. Camacho, «Cooperative distributed MPC for tracking☆,» *Automatica*, vol. 49, nº 1, p. 906–914, 2013.

[29] F. Ferrese, Q. Dong, S. Biswas y J. Batcho, «Decentralized Control of Coupled Nonlinear Dynamic Systems with Application to Quadruple-Tank Process,» de *Industrial Electronics Society, IECON 2014 - 40th Annual Conference of the IEEE*, Dallas, TX, 2014.

[30] VIPA_GmbH, «SPEED7 - CPU SC | 314-6CG03 | Manual,» 30 11 2009. [En línea]. Available: http://www.vipa.com/en/products/control-systems/300s/cpus/?tx_vipaproducts_pi1[tab]=1&cHash=1ed530b27ad5fc4dbf2a98e8b8cc9517. [Último acceso: 01 09 2010].

Anexo A

A1. Código en modelo4tanques.m del modelo de cuatro tanques acoplados:

```
%% Modelo de 4 tanque acoplados
% José Carlos Oliden Semino
% Universidad de Piura
% maestria en Ingeniería Mecánico Eléctrica
% con mención en automática y optimización.
% Febrero 2016.
function [dh] = modelo4tanques(t,h,u)
%MODELO4TANQUE modelo de 4 tanques  acoplados.
dh=zeros(4,1);
%% constantes
% area transversal de los tanques.
A1=100;
A2=100;
A3=100;
A4=100;
% area de orificios.
Cd=0.64; % coeficiente de descarga
a1=(Cd*pi*(0.9)^2)/4;
a2=(Cd*pi*(0.9)^2)/4;
a3=(Cd*pi*(0.9)^2)/4;
a4=(Cd*pi*(0.9)^2)/4;
% costantes fisicas
g=9.81*100;
%parametros estimados con pruebas sobre el módulo
x12=1.1844;
x2a=1.0377;
x34=1.2846;
x4a=1.1032;
```

```
w21=49.8846;
w13=49.8619;
k21= -1.5809e+04;
k12=0.1561;
k13=-1.4327e+04;
k24=0.0487;
%% ecuaciones dinámicas
dh(1)=(sqrt(w21*u(2)^2+k21)-x12*a1*sqrt(2*g*h(1)))/A1;
dh(2)=(k12*sqrt(w13*u(1)^2+k13)+x12*a1*sqrt(2*g*h(1))-x2a*a2*sqrt(2*g*h(2)))/A2;
dh(3)=(sqrt(w13*u(1)^2+k13)-x34*a3*sqrt(2*g*h(3)))/A3;
dh(4)=(k24*sqrt(w21*u(2)^2+k21)+x34*a3*sqrt(2*g*h(3))-x4a*a4*sqrt(2*g*h(4)))/A4;
return
```

A2. Código en ControlMPC_2_tanques.m del control del modelo de cuatro tanques acoplados:

```
%% Modelo de 4 tanque acoplados
% José Carlos Oliden Semino
% Universidad de Piura
% Maestria en Ingeniería Mecánico Eléctrica
% con mención en automática y optimización.
% Febrero 2016.
% se controlan los tanques inferiores 2 y 4
%%
clear all;
%% cargar modelo en espacio de estados
load('modelo_ident_SS');
%% parametros de simulacion
Tk_fin=700;
Ts=5;
%% Parametros del modelo.
N_entradas=2; % Numero de entradas
N_salidas=2; % Numero de salidas
N_estados=4; % Numero de estados
% Numero de estados modelo velocidad
N_est_velocidad=N_salidas+N_estados;
Un=[20.64;20.64];% valores nominales de entrada.
% Matrizes del modelo en espacio de estados
A=sys1.A;
B=sys1.B;
C=sys1.C;
yn=Y0';% valor nominal de salidas y
% calculo Matriz modelo de velocidad.
Av=[A zeros(N_estados,N_salidas);C*A eye(N_salidas)];
Bv=[B;C*B];
Cv=[zeros(N_salidas,N_estados) eye(N_salidas,N_salidas)];
%% Parametros de controlador.
N2=20;
```

```
N1=1;
N=N2-N1+1;
Nu=3;
Lambda=5.0;
%% Inicializacion de martrices, vectores y parametros.
% matrices del proceso
U11=zeros(Tk_fin,N_entradas);
dU11=zeros(Tk_fin,N_entradas);% matriz de entradas al modelo (respecto a Un)
ymp=zeros(Tk_fin,4);% matriz de salidas planta
ym=zeros(Tk_fin,4);% matriz de salidas modelo
zm=zeros(Tk_fin,N_est_velocidad);% matriz de estados modelo
z_estimado=zeros(Tk_fin,N_est_velocidad);% matriz de estados estimados
Yest=zeros(Tk_fin,N_salidas);
ref=zeros(Tk_fin,N_salidas); % vectores de referencia salidas.
W=repmat([22.5;18.4],N,1);%referencias salidas completas
W1=zeros(Tk_fin,N_salidas);
Yest(1,:)=[yn(2);yn(4)];
% Matrices programacion cuadratica
G=zeros(N_salidas*N,N_est_velocidad);
F=zeros(N_salidas*N,N_entradas*Nu);
F_aux11=zeros(N_salidas*N,N_entradas*N);
Ares=zeros(4*Nu*N_entradas+2*N*N_salidas,Nu*N_entradas);
Bres=zeros(4*Nu*N_entradas+2*N*N_salidas,1);
%% Calculo de matrices y vectores constantes.
% Matrices para resolver le programacion cuadratica
G(1:N_salidas,:)=Cv*Av;
F_aux11(1:N_salidas,1:N_entradas)=Cv*Bv;
for j=2:N
   G((j-1)*N_salidas+1:j*N_salidas,:)=Cv*Av^j;
end
for j=2:N2
   for i=1:j
   F_aux11((j-1)*N_salidas+1:j*N_salidas,(i-1)*N_entradas+1:i*N_entradas)=Cv*Av^(j-
i)*Bv;
   end
end
F=F_aux11(1:N_salidas*N,1:N_entradas*Nu);
H=2*(F'*F+Lambda*eye(N_entradas*Nu,N_entradas*Nu));%Hessiano
ff=2*(z_estimado(1,:)*G'*F-(W-repmat([yn(2);yn(4)],N,1))'*F);
% Restriciones
dUmin=[-0.5;-0.5];
dUmax=[0.5;0.5];
Umin=[19.44;19.44];%respecto al valor de la planta
Umax=[21.84; 21.84];%respecto valor para la planta
ymin=[10;11];%respecto al valor del modelo
ymax=[30;22.5];%respecto al valor del modelo

Ares(1:Nu*N_entradas,:)=eye(Nu*N_entradas);
Ares(Nu*N_entradas+1:2*Nu*N_entradas,:)=-eye(Nu*N_entradas);
Bres(1:Nu*N_entradas)=repmat(dUmax,Nu,1);
```

```
Bres(Nu*N_entradas+1:2*Nu*N_entradas)=-repmat(dUmin,Nu,1);

Ares(2*Nu*N_entradas+1:3*Nu*N_entradas,:)=tril(repmat(eye(N_entradas,N_entradas),N
u,Nu));
Ares(3*Nu*N_entradas+1:4*Nu*N_entradas,:)=-
tril(repmat(eye(N_entradas,N_entradas),Nu,Nu));

Bres(2*Nu*N_entradas+1:3*Nu*N_entradas)=repmat(Umax-U11(1,:)',Nu,1);
Bres(3*Nu*N_entradas+1:4*Nu*N_entradas)=-repmat(Umin-U11(1,:)',Nu,1);

Ares(4*Nu*N_entradas+1:4*Nu*N_entradas+N*N_salidas,:)=F;
Ares(4*Nu*N_entradas+N*N_salidas+1:end,:)=-F;
%los z estimados se actualiza cada tiempo de muestreo
Bres(4*Nu*N_entradas+1:4*Nu*N_entradas+N*N_salidas)=repmat(ymax-
[yn(2);yn(4)],N,1)-G*z_estimado(1,:)';
Bres(4*Nu*N_entradas+N*N_salidas+1:end)=-repmat(ymin-
[yn(2);yn(4)],N,1)+G*z_estimado(1,:)';

% Para el filtro de Kalman
Xem1=zeros(N_est_velocidad,1);
Pkm1=ones(N_est_velocidad,N_est_velocidad)*0.01;
Q=eye(N_est_velocidad)*0.000005;%ones(10,10)*0.01;
R=eye(N_salidas);R(1,1)=5.0;R(2,2)=3.5;%R(3,3)=0.5;R(3,3)=0.5;
Xem=zeros(Tk_fin,N_est_velocidad);
Xe=zeros(Tk_fin,N_est_velocidad);

%% simulacion
yini=yn;% valor inicial de la salida del modelo de la planta
U11(1,:)=[20.64,20.64];
dU11(1,:)=[0 0];
ymp(1,:)=yn';
W1(1,:)=W(1:N_salidas,1)';

for k=2:Tk_fin
    %% Simulacion de la planta real
     Uent=U11(k-1,:);
     [tmp1,ymp1]=ode45(@(tmp1,ymp1)modelo4tanques(tmp1,ymp1,Uent),[0 Ts],yini);
     % actualización del valor inicial del modelo de la planta
     yini=ymp1(end,:);
     ymp(k,:)=ymp1(end,:)+random('norm',0,0.2);
     %% Estimación de estados
     zm(k,:)=Av*Xem1+Bv*dU11(k-1,:)';
     Pkm=Av*Pkm1*Av'+Q;
     Kk=(Pkm*Cv')/(Cv*Pkm*Cv'+R);
     z_estimado(k,:)=zm(k,:)'+Kk*([ymp(k,2);ymp(k,4)]-[yn(2);yn(4)]-Cv*zm(k,:)');
     Pk=(eye(N_est_velocidad)-Kk*Cv)*Pkm;
     Xem1=zm(k,:)';
     Pkm1=Pk;
     Yest(k,:)=Cv*z_estimado(k,:)'+[yn(2);yn(4)];
    %% Actualización de matrices
```

```
    if k>50
       W=repmat([16.8;18.44],N,1); end;
    if k>100
       W=repmat([16.8;21.2],N,1); end;
    if k>150
       W=repmat([25.5;21.2],N,1); end;
    if k>200
       W=repmat([25.5;16.6],N,1); end;
    if k>250
       W=repmat([16.7;16.6],N,1); end;
    if k>300
       W=repmat([16.7;21.6],N,1); end;
    if k>350
       W=repmat([24.9;21.6],N,1); end;
    if k>400
       W=repmat([24.9;14.7],N,1); end;
    if k>450
       W=repmat([14.2;14.7],N,1); end;
    if k>500
       W=repmat([14.2;21.1],N,1); end;
    if k>550
       W=repmat([27.0;21.1],N,1); end;
    if k>600
       W=repmat([27.0;16.1],N,1); end;
    if k>650
       W=repmat([14.5;16.1],N,1); end;

     H=2*(F'*F+Lambda*eye(N_entradas*Nu,N_entradas*Nu));%Hessiano
     ff=2*(z_estimado(k,:)*G'*F-(W-repmat([yn(2);yn(4)],N,1))'*F);

     %restriciones
     Ares(1:Nu*N_entradas,:)=eye(Nu*N_entradas);
     Ares(Nu*N_entradas+1:2*Nu*N_entradas,:)=-eye(Nu*N_entradas);
     Bres(1:Nu*N_entradas)=repmat(dUmax,Nu,1);
     Bres(Nu*N_entradas+1:2*Nu*N_entradas)=-repmat(dUmin,Nu,1);


Ares(2*Nu*N_entradas+1:3*Nu*N_entradas,:)=tril(repmat(eye(N_entradas,N_entradas),N
u,Nu));
     Ares(3*Nu*N_entradas+1:4*Nu*N_entradas,:)=-
tril(repmat(eye(N_entradas,N_entradas),Nu,Nu));
     % Un se reemplaza por u(t-1) en la simulacion
     Bres(2*Nu*N_entradas+1:3*Nu*N_entradas)=repmat(Umax-U11(k-1,:)',Nu,1);
     Bres(3*Nu*N_entradas+1:4*Nu*N_entradas)=-repmat(Umin-U11(k-1,:)',Nu,1);

     Ares(4*Nu*N_entradas+1:4*Nu*N_entradas+N*N_salidas,:)=F;
     Ares(4*Nu*N_entradas+N*N_salidas+1:end,:)=-F;
     %los z estimados se actualiza cada tiempo de muestreo
     Bres(4*Nu*N_entradas+1:4*Nu*N_entradas+N*N_salidas)=repmat(ymax-
[yn(2);yn(4)],N,1)-G*z_estimado(k,:)';
```

```
    Bres(4*Nu*N_entradas+N*N_salidas+1:end)=-repmat(ymin-
[yn(2);yn(4)],N,1)+G*z_estimado(k,:)';

    eta=QPhild(H,ff',Ares,Bres);
    dU11(k,:)=eta(1:2);
    U11(k,:)=U11(k-1,:)+dU11(k,:);
    W1(k,:)=W(1:N_salidas,1)';
end
%% graficar
tt=[0:5:5*(length(ymp)-1)];
figure(1);
subplot(4,1,1)
plot(tt,ymp(:,1),'g');
legend('h1(cm)');
hold on;
legend('h1(cm)');
ylabel('h1(cm)');
grid on;
subplot(4,1,2)
plot(tt,ymp(:,2),'g');
hold on;plot(tt,W1(:,1),'k');
plot(tt,repmat(ymax(1)',Tk_fin,1),'r--');
plot(tt,repmat(ymin(1)',Tk_fin,1),'b--');
legend('h2(cm)','referencia');
ylabel('h2(cm)');
grid on;
subplot(4,1,3)
plot(tt,ymp(:,3),'g');
hold on
legend('h3(cm)');
ylabel('h3(cm)');
grid on;
subplot(4,1,4)
plot(tt,ymp(:,4),'g');
hold on;plot(tt,W1(:,2),'k');
plot(tt,repmat(ymax(2)',Tk_fin,1),'r--');
plot(tt,repmat(ymin(2)',Tk_fin,1),'b--');
legend('h4(cm)','referencia');
xlabel('Tiempo(segundos)')
ylabel('h4(cm)');
grid on;
figure(2);
subplot(2,1,1)
plot(tt,U11(:,1),'g');hold on;
plot(tt,repmat(Umax(1)',Tk_fin,1),'r--');
plot(tt,repmat(Umin(1)',Tk_fin,1),'b--');
legend('T. Bomba 1(V)')
ylabel('T. Bomba 1(V)');
grid on;
subplot(2,1,2)
```

```
plot(tt,U11(:,2),'g');hold on;
plot(tt,repmat(Umax(2)',Tk_fin,1),'r--');
plot(tt,repmat(Umin(2)',Tk_fin,1),'b--');
legend('T. Bomba 2(V)')
ylabel('T. Bomba 2(V)');
xlabel('Tiempo(segundos)');
grid on;
figure(3);
subplot(2,1,1)
plot(tt,dU11(:,1),'g');hold on;
plot(tt,repmat(dUmax(1)',Tk_fin,1),'r--');
plot(tt,repmat(dUmin(1)',Tk_fin,1),'b--');
legend('Desviación T. Bomba 1(V)')
ylabel('Desviación T. Bomba 1(V)');
grid on;
subplot(2,1,2)
plot(tt,dU11(:,2),'g');hold on;
plot(tt,repmat(dUmax(2)',Tk_fin,1),'r--');
plot(tt,repmat(dUmin(2)',Tk_fin,1),'b--');
legend('Desviación T. Bomba 2(V)')
ylabel('Desviación T. Bomba 2(V)');
xlabel('Tiempo(segundos)');
grid on;
```

Anexo B

B1. Bloque de datos DB1 en lenguaje SCL.

```
DATA_BLOCK DB1
// Parámetros del controlador.
STRUCT
    Entradas: INT;
    Salidas: INT;
    Estados: INT;
    EstadosVel: INT;
    N1: INT;
    N2: INT;
    Nu: INT;
    Lambda: REAL;
    Ynominal: ARRAY [1..4, 1..1] OF REAL;
    Unominal: ARRAY [1..2, 1..1] OF REAL;
    Ymax: ARRAY [1..4, 1..1] OF REAL;
```

```
    Ymin: ARRAY [1..4, 1..1] OF REAL;
    Umax: ARRAY [1..2, 1..1] OF REAL;
    Umin: ARRAY [1..2, 1..1] OF REAL;
    dUmax: ARRAY [1..2, 1..1] OF REAL;
    dUmin: ARRAY [1..2, 1..1] OF REAL;
END_STRUCT
BEGIN
Entradas:=2;
Salidas:=4;
Estados:=10;
EstadosVel:=14;
N1:=1;
N2:=15;
Nu:=3;
Lambda:= 1.0;
END_DATA_BLOCK
```

B2. Bloque de datos DB2 en lenguaje SCL.

```
DATA_BLOCK DB2
// Bloque de datos de matrices....
STRUCT
H: ARRAY [1..6, 1..6] OF REAL;
Haux: ARRAY [1..6, 1..6] OF REAL;
Hinv: ARRAY [1..6, 1..6] OF REAL; //H inversa
AUX1: ARRAY [1..6, 1..48] OF REAL; //Ares*inv(H)
P: ARRAY [1..48, 1..48] OF REAL; //P=Ares*(Ares´*inv(H))
d1: ARRAY [1..48, 1..1] OF REAL; // d=(Ares*(f/H)+b)
Lambda: ARRAY [1..48, 1..1] OF REAL;
Lambda_P: ARRAY [1..48, 1..1] OF REAL;
```

```
AUX2: ARRAY [1..6, 1..1] OF REAL;
Ftruc: ARRAY [1..12, 1..6] OF REAL;
Wtruc: ARRAY [1..12, 1..1] OF REAL;
WtF: ARRAY [1..1, 1..6] OF REAL;
Z: ARRAY [1..14, 1..1] OF REAL;
GtF: ARRAY [1..14, 1..6] OF REAL;
f: ARRAY [1..1, 1..6] OF REAL;
Ares: ARRAY [1..48, 1..6] OF REAL;//reducida
Bres: ARRAY [1..48, 1..1] OF REAL;//reducida
du: ARRAY [1..2,1..1] OF REAL;//variaciòn de u
Xem1: ARRAY [1..14, 1..1] OF REAL;
Pkm1: ARRAY [1..14, 1..14] OF REAL;
Q: ARRAY [1..14, 1..14]  OF REAL;
R: ARRAY [1..4,1..4] OF REAL;
Kk: ARRAY [1..14, 1..4] OF REAL;
//Matrices del modelo de velocidad
Av: ARRAY [1..14, 1..14] OF REAL;
Bv: ARRAY [1..14, 1..2] OF REAL;
Cv: ARRAY [1..4,1..14] OF REAL;
   END_STRUCT
BEGIN
H [1,1]:= 1.0;H [1,2]:= 0.0;H [1,3]:= 0.0;H [1,4]:= 0.0;H [1,5]:= 0.0;H [1,6]:= 0.0;
H [2,1]:= 0.0;H [2,2]:= 1.0;H [2,3]:= 0.0;H [2,4]:= 0.0;H [2,5]:= 0.0;H [2,6]:= 0.0;
H [3,1]:= 0.0;H [3,2]:= 0.0;H [3,3]:= 1.0;H [3,4]:= 0.0;H [3,5]:= 0.0;H [3,6]:= 0.0;
H [4,1]:= 0.0;H [4,2]:= 0.0;H [4,3]:= 0.0;H [4,4]:= 1.0;H [4,5]:= 0.0;H [4,6]:= 0.0;
H [5,1]:= 0.0;H [5,2]:= 0.0;H [5,3]:= 0.0;H [5,4]:= 0.0;H [5,5]:= 1.0;H [5,6]:= 0.0;
H [6,1]:= 0.0;H [6,2]:= 0.0;H [6,3]:= 0.0;H [6,4]:= 0.0;H [6,5]:= 0.0;H [6,6]:= 1.0;

END_DATA_BLOCK
```

B3. Bloque de datos DB3 en lenguaje SCL.

```
DATA_BLOCK DB3
//Datos del proceso
   STRUCT
   En1: REAL;
   En2: REAL;
   En3: REAL;
   En4: REAL;
   Sal1: REAL;
   Sal2: REAL;
   END_STRUCT
BEGIN
En1:= 0.0;
En2:= 0.0;
En3:= 0.0;
En4:= 0.0;
Sal1:= 0.0;
Sal2:= 0.0;
END_DATA_BLOCK
```

B4. Bloque de datos DB4 en lenguaje SCL.

```
DATA_BLOCK DB4
// Bits de operación de HMI.
   STRUCT
   Operacion_Manual: BOOL;
   Operacion_Identificacion: BOOL;
   Operacion_Automatico: BOOL;
   START_Manual: BOOL;
```

```
   START_Identificacion: BOOL;
   START_Automatico: BOOL;
   STOP_Manual: BOOL;
   STOP_Identificacion: BOOL;
   STOP_Automatico: BOOL;
   IND_STOP_Identificacion: BOOL;
   ENT_TIPO_Ident: BOOL;
   END_STRUCT
BEGIN
END_DATA_BLOCK
```

B5. Bloque de datos DB5 en lenguaje SCL.

```
DATA_BLOCK DB5
//Bloque de Ajuste de Entradas analógicas
   STRUCT
   A1: REAL;
   A2: REAL;
   A3: REAL;
   A4: REAL;
   B1: REAL;
   B2: REAL;
   B3: REAL;
   B4: REAL;
   END_STRUCT
BEGIN
A1:=1.037;
A2:=1.031;
A3:=1.068;
A4:=1.053;
```

```
B1:=2.56;
B2:=2.98;
B3:=-1.94;
B4:=-3.3;
END_DATA_BLOCK
```

B6. Bloque de datos DB6 en lenguaje SCL.

```
DATA_BLOCK DB6
//Bloque de datos para identificación
   STRUCT
   Saltos: ARRAY[1..21, 1..2] OF INT;
   Saltos1: ARRAY[1..87,1..2] OF INT;
   con1: INT;
   con2: INT;
   Unom1: REAL;
   Unom2: REAL;
   Umin1: REAL;
   Umax1: REAL;
   Umin2: REAL;
   Umax2: REAL;
      END_STRUCT
BEGIN
Saltos[1,1]:=80;Saltos[2,1]:=85;Saltos[3,1]:=90;Saltos[4,1]:=110; Saltos[5,1]:=130;
Saltos[6,1]:=140;Saltos[7,1]:=150;Saltos[8,1]:=190;Saltos[9,1]:=230;Saltos[10,1]:=310;
Saltos[11,1]:=390;Saltos[12,1]:=470;Saltos[13,1]:=550;Saltos[14,1]:=560;Saltos[15,1]:=
570;
Saltos[16,1]:=610;Saltos[17,1]:=650;Saltos[18,1]:=670;Saltos[19,1]:=690;Saltos[20,1]:=6
95;
 Saltos[21,1]:=700;
```

```
Saltos[1,2]:=80;Saltos[2,2]:=120;Saltos[3,2]:=160;Saltos[4,2]:=165;Saltos[5,2]:= 170;

Saltos[6,2]:=250;Saltos[7,2]:=330;Saltos[8,2]:=350;Saltos[9,2]:=370;Saltos[10,2]:=380;

Saltos[11,2]:=390;Saltos[12,2]:=400;Saltos[13,2]:=410;Saltos[14,2]:=490;Saltos[15,2]:=5
70;

Saltos[16,2]:=590;Saltos[17,2]:=610;Saltos[18,2]:=615;Saltos[19,2]:=620;Saltos[20,2]:=6
60;

Saltos[21,2]:= 700;

con1:= 0;con2:= 0;

Saltos1[1,1]:=200;Saltos1[2,1]:=210;Saltos1[3,1]:=227;Saltos1[4,1]:=230;Saltos1[5,1]:=2
51;

Saltos1[6,1]:=282;Saltos1[7,1]:=296;Saltos1[8,1]:=302;Saltos1[9,1]:=344;Saltos1[10,1]:=
354;

Saltos1[11,1]:=365;Saltos1[12,1]:=375;Saltos1[13,1]:=392;Saltos1[14,1]:=410;Saltos1[15,
1]:=421;

Saltos1[16,1]:=436;Saltos1[17,1]:=459;Saltos1[18,1]:=463;Saltos1[19,1]:=479;Saltos1[20,
1]:=485;

Saltos1[21,1]:=497;Saltos1[22,1]:=501;Saltos1[23,1]:=534;Saltos1[24,1]:=563;Saltos1[25,
1]:=573;

Saltos1[26,1]:=585;Saltos1[27,1]:=600;Saltos1[28,1]:=616;Saltos1[29,1]:=633;Saltos1[30,
1]:=652;

Saltos1[31,1]:=664;Saltos1[32,1]:=728;Saltos1[33,1]:=746;Saltos1[34,1]:=764;Saltos1[35,
1]:=772;

Saltos1[36,1]:=798;Saltos1[37,1]:=810;Saltos1[38,1]:=839;Saltos1[39,1]:=852;Saltos1[40,
1]:=862;

Saltos1[41,1]:=893;Saltos1[42,1]:=904;Saltos1[43,1]:=907;Saltos1[44,1]:=914;Saltos1[45,
1]:=927;

Saltos1[46,1]:=945;Saltos1[47,1]:=955;Saltos1[48,1]:=968;Saltos1[49,1]:=984;Saltos1[50,
1]:=1004;

Saltos1[51,1]:=1015;Saltos1[52,1]:=1029;Saltos1[53,1]:=1040;Saltos1[54,1]:=1053;Saltos
1[55,1]:=1088;

Saltos1[56,1]:=1117;Saltos1[57,1]:=1149;Saltos1[58,1]:=1184;Saltos1[59,1]:=1196;Saltos
1[60,1]:=1209;

Saltos1[61,1]:=1244;Saltos1[62,1]:=1257;Saltos1[63,1]:=1284;Saltos1[64,1]:=1301;Saltos
1[65,1]:=1310;

Saltos1[66,1]:=1354;Saltos1[67,1]:=1368;Saltos1[68,1]:=1374;Saltos1[69,1]:=1390;Saltos
1[70,1]:=1395;
```

Saltos1[71,1]:=1417;Saltos1[72,1]:=1428;Saltos1[73,1]:=1436;Saltos1[74,1]:=1445;Saltos1[75,1]:=1454;

Saltos1[76,1]:=1476;Saltos1[77,1]:=1489;Saltos1[78,1]:=1499;Saltos1[79,1]:=1510;Saltos1[80,1]:=1517;

Saltos1[81,1]:=1527;Saltos1[82,1]:=1540;Saltos1[83,1]:=1544;Saltos1[84,1]:=1556;Saltos1[85,1]:=1580;

Saltos1[86,1]:=1592;Saltos1[87,1]:=1600;

Saltos1[1,2]:=200;Saltos1[2,2]:=229;Saltos1[3,2]:=252;Saltos1[4,2]:=261;Saltos1[5,2]:=299;

Saltos1[6,2]:=307;Saltos1[7,2]:=328;Saltos1[8,2]:=343;Saltos1[9,2]:=427;Saltos1[10,2]:=436;

Saltos1[11,2]:=447;Saltos1[12,2]:=454;Saltos1[13,2]:=476;Saltos1[14,2]:=493;Saltos1[15,2]:=504;

Saltos1[16,2]:=516;Saltos1[17,2]:=532;Saltos1[18,2]:=550;Saltos1[19,2]:=565;Saltos1[20,2]:=584;

Saltos1[21,2]:=594;Saltos1[22,2]:=647;Saltos1[23,2]:=659;Saltos1[24,2]:=671;Saltos1[25,2]:=686;

Saltos1[26,2]:=694;Saltos1[27,2]:=705;Saltos1[28,2]:=718;Saltos1[29,2]:=737;Saltos1[30,2]:=753;

Saltos1[31,2]:=804;Saltos1[32,2]:=838;Saltos1[33,2]:=846;Saltos1[34,2]:=864;Saltos1[35,2]:=874;

Saltos1[36,2]:=912;Saltos1[37,2]:=917;Saltos1[38,2]:=936;Saltos1[39,2]:=944;Saltos1[40,2]:=953;

Saltos1[41,2]:=957;Saltos1[42,2]:=975;Saltos1[43,2]:=987;Saltos1[44,2]:=993;Saltos1[45,2]:=1007;

Saltos1[46,2]:=1023;Saltos1[47,2]:=1033;Saltos1[48,2]:=1045;Saltos1[49,2]:=1059;Saltos1[50,2]:=1095;

Saltos1[51,2]:=1108;Saltos1[52,2]:=1119;Saltos1[53,2]:=1129;Saltos1[54,2]:=1134;Saltos1[55,2]:=1149;

Saltos1[56,2]:=1157;Saltos1[57,2]:=1170;Saltos1[58,2]:=1183;Saltos1[59,2]:=1193;Saltos1[60,2]:=1238;

Saltos1[61,2]:=1255;Saltos1[62,2]:=1261;Saltos1[63,2]:=1275;Saltos1[64,2]:=1285;Saltos1[65,2]:=1290;

Saltos1[66,2]:=1319;Saltos1[67,2]:=1347;Saltos1[68,2]:=1357;Saltos1[69,2]:=1368;Saltos1[70,2]:=1376;

Saltos1[71,2]:=1387;Saltos1[72,2]:=1398;Saltos1[73,2]:=1408;Saltos1[74,2]:=1415;Saltos1[75,2]:=1428;

Saltos1[76,2]:=1438;Saltos1[77,2]:=1461;Saltos1[78,2]:=1483;Saltos1[79,2]:=1498;Saltos1[80,2]:=1505;

Saltos1[81,2]:=1523;Saltos1[82,2]:=1538;Saltos1[83,2]:=1553;Saltos1[84,2]:=1563;Saltos1[85,2]:=1574;

Saltos1[86,2]:=1577;Saltos1[87,2]:=1600;

END_DATA_BLOCK

B7.Función en la que se resuelve el problema de programación cuadrática en lenguaje SCL.

```
FUNCTION FC1 : INT
// Algoritmo Hildreth
VAR_TEMP
   // variables temporales
   I: INT;
   J: INT;
   K: INT;
   L: INT;
   N: INT;
   DIVI: REAL;
   MUL: REAL;
   LA: REAL;
   AL: REAL;
   W: REAL;
END_VAR
//Halla respuesta sin restricciones.
  //Calculo de inversa de H.
     //llenar invH de respuesta.
     FOR I:= 1 TO DB1.Entradas*DB1.Nu DO
        FOR J:= 1 TO DB1.Entradas*DB1.Nu DO
```

```
      IF I = J THEN
        DB2.Hinv[I,J]:=1.0;
      ELSE
        DB2.Hinv[I,J]:=0.0;
      END_IF;
    END_FOR;
  END_FOR;
  FOR I:= 1 TO DB1.Entradas*DB1.Nu DO
    FOR J:= 1 TO DB1.Entradas*DB1.Nu DO
      DB2.Haux[I,J]:=DB2.H[I,J];
    END_FOR;
  END_FOR;
 FOR I:= 1 TO DB1.Entradas*DB1.Nu DO
       DIVI:= DB2.Haux[I,I];
    FOR J:= I TO DB1.Entradas*DB1.Nu DO
       DB2.Haux[I,J]:=DB2.Haux[I,J]/DIVI;
    END_FOR;
    FOR J:= 1 TO DB1.Entradas*DB1.Nu DO
      DB2.Hinv[I,J]:=DB2.Hinv[I,J]/DIVI;
    END_FOR;
    FOR J:= I+1 TO DB1.Entradas*DB1.Nu DO
       MUL:= DB2.Haux[J,I];
      FOR K:= I TO DB1.Entradas*DB1.Nu DO
       DB2.Haux[J,K]:=DB2.Haux[J,K]-MUL*DB2.Haux[I,K];
      END_FOR;
      FOR K:= 1 TO DB1.Entradas*DB1.Nu DO
       DB2.Hinv[J,K]:=DB2.Hinv[J,K]-MUL*DB2.Hinv[I,K];
      END_FOR;
    END_FOR;
  END_FOR;
```

```
    FOR K:=DB1.Entradas*DB1.Nu TO 2 BY -1 DO
      FOR I:=K-1 TO 1 BY -1 DO
        FOR J:=1 TO DB1.Entradas*DB1.Nu DO
          DB2.Hinv[I,J]:=DB2.Hinv[I,J]-DB2.Hinv[K,J]*DB2.Haux[I,K];
        END_FOR;
      END_FOR;
    END_FOR;
  //hallar eta=-invH*f  Respuesta sin restriciones dUCompleto
   FOR I:=1 TO DB1.Entradas*DB1.Nu DO
    FOR J:=1 TO DB1.Entradas*DB1.Nu DO
      DB2.dU[J,1]:=DB2.Hinv[I,J]*DB2.f[1,J]*(-1);
    END_FOR;
   END_FOR;
   //Verificar restricciones
   K:=0;
   FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
    MUL:=0.0;
    FOR J:=1 TO DB1.Nu*DB1.Entradas DO
       MUL:=MUL+DB2.Ares [I,J]*DB2.dU[J,1];
    END_FOR;
    IF  MUL >DB2.Bres[I,1]THEN
      K:=K+1;
    END_IF;
   END_FOR;
   IF K > 0 THEN
      RETURN;
   END_IF;
 //Multlicar Ares*inv(H)
 FOR J:=1 TO DB1.Nu*DB1.Entradas DO
  FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
```

```
    FOR K:=1 TO DB1.Nu*DB1.Entradas DO
        DB2.AUX1[J,I]:=DB2.Hinv[J,K]*DB2.Ares[I,K];
    END_FOR;
  END_FOR;
END_FOR;
//Multlicar P=Ares*(Ares´*inv(H))
FOR J:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
  FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
    FOR K:=1 TO DB1.Nu*DB1.Entradas DO
       DB2.P[J,I]:=DB02.Ares[J,K]*DB2.AUX1[K,I];
    END_FOR;
  END_FOR;
END_FOR;
//calculo de d=(Ares*(f/H)+b)
 FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
    FOR K:=1 TO DB1.Nu*DB1.Entradas DO
        ;DB2.d1[I,1]:=DB2.Ares[I,K]*DB2.du[K,1]*(-1);
    END_FOR;
        ;DB2.d1[I,1]:=DB2.d1[I,1]+DB2.Bres[I,1];
 END_FOR;
 //Inicialiacion de Lambda
 FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
   DB2.Lambda[I,1]:=0.0;
 END_FOR;
 AL:=10.0;
 //Iteraciones.
 FOR L:=1 TO 38 DO
   DB2.Lambda_P:=DB2.Lambda;
   //Calculo de W
   FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
```

```
    FOR J:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
      W:=DB2.P[I,J]*DB2.Lambda[I,1];
    END_FOR;
    W:=W-DB2.P[I,I]*DB2.Lambda[I,1];
    W:=W+DB2.d1[I,1];
    LA:=(-1)*W/DB2.P[I,I];
    W:=W+DB2.P[I,1];
    //Calculo Lambda(i,1)= max(0,LA);
    IF LA > 0.0 THEN
      DB2.Lambda[I,1]:=LA;
    ELSE
      DB2.Lambda[I,1]:=0.0;
    END_IF;
  END_FOR;
  FOR I:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
    AL:=(DB2.Lambda[I,1]-DB2.Lambda_P[I,1])*(DB2.Lambda[I,1]-
DB2.Lambda_P[I,1]);
  END_FOR;
  IF AL < 0.0000001 THEN
   EXIT;
  END_IF;
 END_FOR;
 FOR I:=1 TO DB1.Nu*DB1.Entradas DO
  DB2.AUX2[I,1]:=0.0;
  FOR J:=1 TO 2*DB1.Nu*DB1.Salidas+4*DB1.Nu*DB1.Entradas DO
    DB2.AUX2[I,1]:=DB2.AUX2[I,1]-DB2.AUX1[I,J]*DB2.Lambda[J,1];
  END_FOR;
    DB2.dU[I,1]:=DB2.dU[I,1]+DB2.AUX2[I,1];
 END_FOR;
    FC1 := 100;
END_FUNCTION
```

B8. Función que actualiza las matices de programación cuadrática en lenguaje SCL.

```
FUNCTION FC2 : INT
// Actualización de Matrices de entrada de FC1  QP.
// H, f, Ares, Bres.
VAR_TEMP
   // variables temporales
   I: INT;
   J: INT;
   K: INT;
END_VAR
  //calculo de H
  FOR I:=1 TO DB1.Nu*DB1.Entradas DO
    FOR J:=1 TO DB1.Nu*DB1.Entradas DO
      FOR K:=1 TO DB1.Nu*DB1.Salidas DO
       IF I = J THEN
         DB2.H[I,J]:=2*(DB2.F[K,I]*DB2.F[K,J]+DB1.Lambda);
       ELSE
         DB2.H[I,J]:= 2*(DB2.F[K,I]*DB2.F[K,J]);
       END_IF;
      END_FOR;
    END_FOR;
  END_FOR;
  //calculo WtF
  FOR I:=1 TO DB1.Nu*DB1.Entradas DO
    FOR K:= 1 TO DB1.Salidas*DB1.Nu DO
      DB2.WtF[1,I]:=DB2.Wtruc[J,1]*DB2.Ftruc[J,I];
    END_FOR;
  END_FOR;
```

```
  //calculo de f
  FOR J:=1 TO DB1.Nu*DB1.Entradas DO
    DB2.f[1,J]:=0.0;
    FOR K:=1 TO DB1.EstadosVel DO
      DB2.f[1,J]:= DB2.f[1,J]+DB2.Z[K,1]*DB2.GtF[K,I];
    END_FOR;
    DB2.f[1,J]:=2*(DB2.f[1,J]-DB2.WtF[1,J]);
  END_FOR;
   FC2 := 100;
END_FUNCTION
```

B9. Función que ajusta la lectura de los sensores en lenguaje SCL.

```
FUNCTION FC13 : INT
//Ajuste de señales de entrada.
//Ajusta la señal escalada de los sensores a valores calibrados a una recta.
// Y=Ax+B
VAR_INPUT
   IN: REAL;
   A: REAL;
   B: REAL;
END_VAR
VAR_OUTPUT
   Sal: REAL;
END_VAR
VAR_TEMP
END_VAR
   Sal:=A*IN+B;
   FC13 := 100;
END_FUNCTION
```

B10.Función FC20 que genera las secuencias de entrada para la identificación en lenguaje SCL. (Señal de escalones).

```
FUNCTION FC20 : INT
//función que maneja la identificación en el bloque cíclico.
VAR_TEMP


END_VAR
  //0
  IF DB6.Con1 = 0 THEN
    DB3.Sal1:= DB6.Unom1;
  END_IF;
  IF DB6.Con1 = 0 THEN
    DB3.Sal2:= DB6.Unom2;
  END_IF;
  // 1
  IF DB6.Con1 = DB6.Saltos[1,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con1 = DB6.Saltos[1,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  //2
  IF DB6.Con1 = DB6.Saltos[2,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con1 = DB6.Saltos[2,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  //3
  IF DB6.Con1 = DB6.Saltos[3,1] THEN
```

```
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[3,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//4
IF DB6.Con1 = DB6.Saltos[4,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[4,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//5
IF DB6.Con1 = DB6.Saltos[5,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[5,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//6
IF DB6.Con1 = DB6.Saltos[6,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[6,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//7
IF DB6.Con1 = DB6.Saltos[7,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
```

```
IF DB6.Con1 = DB6.Saltos[7,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//8
IF DB6.Con1 = DB6.Saltos[8,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[8,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//9
IF DB6.Con1 = DB6.Saltos[9,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[9,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
// 10
IF DB6.Con1 = DB6.Saltos[10,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[10,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//11
IF DB6.Con1 = DB6.Saltos[11,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[11,2] THEN
  DB3.Sal2:= DB6.Umax2;
```

```
END_IF;
//12
IF DB6.Con1 = DB6.Saltos[12,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[12,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//13
IF DB6.Con1 = DB6.Saltos[13,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[13,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//14
IF DB6.Con1 = DB6.Saltos[14,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[14,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//15
IF DB6.Con1 = DB6.Saltos[15,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[15,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//16
```

```
IF DB6.Con1 = DB6.Saltos[16,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[16,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//17
IF DB6.Con1 = DB6.Saltos[17,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[17,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//18
IF DB6.Con1 = DB6.Saltos[18,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con1 = DB6.Saltos[18,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//19
IF DB6.Con1 = DB6.Saltos[19,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con1 = DB6.Saltos[19,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//20
IF DB6.Con1 = DB6.Saltos[20,1] THEN
  DB3.Sal1:= DB6.Umin1;
```

```
   END_IF;
   IF DB6.Con1 = DB6.Saltos[20,2] THEN
      DB3.Sal2:= DB6.Umin2;
   END_IF;
   //21
   IF DB6.Con1 = DB6.Saltos[21,1] THEN
      DB3.Sal1:= 0.0;
   END_IF;
   IF DB6.Con1 = DB6.Saltos[21,2] THEN
      DB3.Sal2:= 0.0;
      DB6.Con1:=0;
      DB4.IND_STOP_Identificacion:= true;
      DB4.STOP_Identificacion:= true;
   END_IF;
   DB6.Con1:= DB6.Con1+1;
   ;
   FC20 := 100;
END_FUNCTION
```

B11. Función FC21 que genera las secuencias de entrada para la identificación en lenguaje SCL. (Señal de PRBS).

```
FUNCTION FC21 : INT
//función que maneja la identificación en el bloque cíclico.
VAR_TEMP

END_VAR
   IF DB6.Con2 = 0 THEN
      DB3.Sal1:= DB6.Unom1;
      DB3.Sal2:= DB6.Unom2;
   END_IF;
```

```
// 1
IF DB6.Con2 = DB6.Saltos1[1,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[1,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//2
IF DB6.Con2 = DB6.Saltos1[2,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[2,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//3
IF DB6.Con2 = DB6.Saltos1[3,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[3,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//4
IF DB6.Con2 = DB6.Saltos1[4,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[4,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//5
IF DB6.Con2 = DB6.Saltos1[5,1] THEN
```

```
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[5,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//6
IF DB6.Con2 = DB6.Saltos1[6,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[6,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//7
IF DB6.Con2 = DB6.Saltos1[7,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[7,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//8
IF DB6.Con2 = DB6.Saltos1[8,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[8,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//9
IF DB6.Con2 = DB6.Saltos1[9,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
```

```
IF DB6.Con2 = DB6.Saltos1[9,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
// 10
IF DB6.Con2 = DB6.Saltos1[10,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[10,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//11
IF DB6.Con2 = DB6.Saltos1[11,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[11,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//12
IF DB6.Con2 = DB6.Saltos1[12,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[12,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//13
IF DB6.Con2 = DB6.Saltos1[13,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[13,2] THEN
  DB3.Sal2:= DB6.Umax2;
```

```
END_IF;
//14
IF DB6.Con2 = DB6.Saltos1[14,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[14,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//15
IF DB6.Con2 = DB6.Saltos1[15,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[15,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//16
IF DB6.Con2 = DB6.Saltos1[16,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[16,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//17
IF DB6.Con2 = DB6.Saltos1[17,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[17,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//18
```

```
IF DB6.Con2 = DB6.Saltos1[18,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[18,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//19
IF DB6.Con2 = DB6.Saltos1[19,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[19,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//20
IF DB6.Con2 = DB6.Saltos1[20,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[20,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
// 21
IF DB6.Con2 = DB6.Saltos1[21,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[21,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//22
IF DB6.Con2 = DB6.Saltos1[22,1] THEN
  DB3.Sal1:= DB6.Umax1;
```

```
END_IF;
IF DB6.Con2 = DB6.Saltos1[22,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//23
IF DB6.Con2 = DB6.Saltos1[23,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[23,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//24
IF DB6.Con2 = DB6.Saltos1[24,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[24,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//25
IF DB6.Con2 = DB6.Saltos1[25,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[25,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//26
IF DB6.Con2 = DB6.Saltos1[26,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[26,2] THEN
```

```
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  //27
  IF DB6.Con2 = DB6.Saltos1[27,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[27,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  //28
  IF DB6.Con2 = DB6.Saltos1[28,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[28,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  //29
  IF DB6.Con2 = DB6.Saltos1[29,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[29,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  // 30
  IF DB6.Con2 = DB6.Saltos1[30,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[30,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
```

```
//31
IF DB6.Con2 = DB6.Saltos1[31,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[31,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//32
IF DB6.Con2 = DB6.Saltos1[32,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[32,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//33
IF DB6.Con2 = DB6.Saltos1[33,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[33,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//34
IF DB6.Con2 = DB6.Saltos1[34,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[34,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//35
IF DB6.Con2 = DB6.Saltos1[35,1] THEN
```

```
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[35,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  //36
  IF DB6.Con2 = DB6.Saltos1[36,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[36,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  //37
  IF DB6.Con2 = DB6.Saltos1[37,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[37,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  //38
  IF DB6.Con2 = DB6.Saltos1[38,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[38,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  //39
  IF DB6.Con2 = DB6.Saltos1[39,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
```

```
IF DB6.Con2 = DB6.Saltos1[39,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//40
IF DB6.Con2 = DB6.Saltos1[40,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[40,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
// 41
IF DB6.Con2 = DB6.Saltos1[41,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[41,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//42
IF DB6.Con2 = DB6.Saltos1[42,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[42,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//43
IF DB6.Con2 = DB6.Saltos1[43,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[43,2] THEN
  DB3.Sal2:= DB6.Umax2;
```

```
END_IF;
//44
IF DB6.Con2 = DB6.Saltos1[44,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[44,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//45
IF DB6.Con2 = DB6.Saltos1[45,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[45,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//46
IF DB6.Con2 = DB6.Saltos1[46,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[46,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//47
IF DB6.Con2 = DB6.Saltos1[47,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[47,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//48
```

```
IF DB6.Con2 = DB6.Saltos1[48,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[48,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//49
IF DB6.Con2 = DB6.Saltos1[49,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[49,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
// 50
IF DB6.Con2 = DB6.Saltos1[50,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[50,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//51
IF DB6.Con2 = DB6.Saltos1[51,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[51,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//52
IF DB6.Con2 = DB6.Saltos1[52,1] THEN
  DB3.Sal1:= DB6.Umax1;
```

```
END_IF;
IF DB6.Con2 = DB6.Saltos1[52,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//53
IF DB6.Con2 = DB6.Saltos1[53,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[53,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//54
IF DB6.Con2 = DB6.Saltos1[54,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[54,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//55
IF DB6.Con2 = DB6.Saltos1[55,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[55,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//56
IF DB6.Con2 = DB6.Saltos1[56,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[56,2] THEN
```

```
  DB3.Sal2:= DB6.Umin2;
END_IF;
//57
IF DB6.Con2 = DB6.Saltos1[57,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[57,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//58
IF DB6.Con2 = DB6.Saltos1[58,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[58,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//59
IF DB6.Con2 = DB6.Saltos1[59,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[59,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//60
IF DB6.Con2 = DB6.Saltos1[60,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[60,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
```

```
//61
IF DB6.Con2 = DB6.Saltos1[61,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[61,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//62
IF DB6.Con2 = DB6.Saltos1[62,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[62,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//63
IF DB6.Con2 = DB6.Saltos1[63,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[63,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//64
IF DB6.Con2 = DB6.Saltos1[64,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[64,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//65
IF DB6.Con2 = DB6.Saltos1[65,1] THEN
```

```
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[65,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//66
IF DB6.Con2 = DB6.Saltos1[66,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[66,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//67
IF DB6.Con2 = DB6.Saltos1[67,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[67,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//68
IF DB6.Con2 = DB6.Saltos1[68,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[68,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//69
IF DB6.Con2 = DB6.Saltos1[69,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
```

```
  IF DB6.Con2 = DB6.Saltos1[69,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  //70
  IF DB6.Con2 = DB6.Saltos1[70,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[70,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  // 71
  IF DB6.Con2 = DB6.Saltos1[71,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[71,2] THEN
    DB3.Sal2:= DB6.Umax2;
  END_IF;
  //72
  IF DB6.Con2 = DB6.Saltos1[72,1] THEN
    DB3.Sal1:= DB6.Umax1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[72,2] THEN
    DB3.Sal2:= DB6.Umin2;
  END_IF;
  //73
  IF DB6.Con2 = DB6.Saltos1[73,1] THEN
    DB3.Sal1:= DB6.Umin1;
  END_IF;
  IF DB6.Con2 = DB6.Saltos1[73,2] THEN
    DB3.Sal2:= DB6.Umax2;
```

```
END_IF;
//74
IF DB6.Con2 = DB6.Saltos1[74,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[74,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//75
IF DB6.Con2 = DB6.Saltos1[75,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[75,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//76
IF DB6.Con2 = DB6.Saltos1[76,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[76,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//77
IF DB6.Con2 = DB6.Saltos1[77,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[77,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//78
```

```
IF DB6.Con2 = DB6.Saltos1[78,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[78,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//79
IF DB6.Con2 = DB6.Saltos1[79,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[79,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
// 80
IF DB6.Con2 = DB6.Saltos1[80,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[80,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//81
IF DB6.Con2 = DB6.Saltos1[81,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[81,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//82
IF DB6.Con2 = DB6.Saltos1[82,1] THEN
  DB3.Sal1:= DB6.Umax1;
```

```
END_IF;
IF DB6.Con2 = DB6.Saltos1[82,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//83
IF DB6.Con2 = DB6.Saltos1[83,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[83,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//84
IF DB6.Con2 = DB6.Saltos1[84,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[84,2] THEN
  DB3.Sal2:= DB6.Umin2;
END_IF;
//85
IF DB6.Con2 = DB6.Saltos1[85,1] THEN
  DB3.Sal1:= DB6.Umin1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[85,2] THEN
  DB3.Sal2:= DB6.Umax2;
END_IF;
//86
IF DB6.Con2 = DB6.Saltos1[86,1] THEN
  DB3.Sal1:= DB6.Umax1;
END_IF;
IF DB6.Con2 = DB6.Saltos1[86,2] THEN
```

```
      DB3.Sal2:= DB6.Umin2;
   END_IF;
   //87
   IF DB6.Con2 = DB6.Saltos1[87,1] THEN
      DB3.Sal1:= 0.0;
   END_IF;
   IF DB6.Con2 = DB6.Saltos1[87,2] THEN
      DB3.Sal2:= 0.0;
      DB6.Con2:=0;
      DB4.IND_STOP_Identificacion:= true;
      DB4.STOP_Identificacion:= true;
   END_IF;
   DB6.Con2:= DB6.Con2+1;
   FC21 := 100;
END_FUNCTION
```

B12. Función FC11 que realiza el escalamiento de las señales del sensor en lenguaje AWL.

```
FUNCTION "Escalamiento" : VOID
TITLE =
VERSION : 0.1
VAR_INPUT
  ValorSensor : WORD ;       //Valor del sensor
  NivelSup : REAL ;   //Valor maximo del sensor
  NivelInf : REAL ;    //Valor minimo del sensor
END_VAR
VAR_OUTPUT
  ValorEscalado : REAL ;      //Valor escalado del sensor
END_VAR
VAR_TEMP
```

```
  ValorSensorReal : REAL ;  //Valor del sensor real
  Rango : REAL ;
END_VAR
BEGIN
NETWORK
TITLE =
     L     #ValorSensor;
     ITD   ;
     DTR   ;
     T     #ValorSensorReal;
     L     #NivelSup;
     L     #NivelInf;
     -R    ;
     T     #Rango;
     L     #ValorSensorReal;
     L     2.764800e+004;
     /R    ;
     L     #Rango;
     *R    ;
     L     #NivelInf;
     +R    ;
     T     #ValorEscalado;
END_FUNCTION
```

B13. Función FC12 que realiza el desescalamiento de las señales del actuador en lenguaje AWL.

```
FUNCTION "Desescalamiento" : VOID
TITLE =
//Desescalamiento de señales de salida. se escalara para una salida de 5v máximo
VERSION : 0.1
```

```
VAR_INPUT
  ValorSalReal : REAL ;
  NivelSup : REAL ;
  NivelInf : REAL ;
END_VAR
VAR_OUTPUT
  ValorSalEnt : WORD ;
END_VAR
VAR_TEMP
  Rango : REAL ;
  ValorAux : REAL ;
END_VAR
BEGIN
NETWORK
TITLE =
     L     #NivelSup;
     L     #NivelInf;
     -R    ;
     T     #Rango;
     L     #ValorSalReal;
     L     #NivelInf;
     -R    ;
     L     #Rango;
     /R    ;
     L     1.382400e+004;
     *R    ;
     T     #ValorAux;
     TRUNC ;
     T     #ValorSalEnt;
END_FUNCTION
```

B14. Bloque de función FC14 que realiza la adaptación de las señales de sensor en lenguaje AWL.

```
FUNCTION "Adaptacion entradas" : VOID
TITLE =
VERSION : 0.1
BEGIN
NETWORK
TITLE =Adaptacion entrada en altura 1
    CALL "Escalamiento" (
        ValorSensor           := "EA1",
        NivelSup              := 1.000000e+002,
        NivelInf              := 0.000000e+000,
        ValorEscalado         := MD   200);
    NOP   0;
NETWORK
TITLE =
    CALL "Ajuste a recta entradas" (
        IN                    := MD   200,
        A                     := "AjusteEn".A1,
        B                     := "AjusteEn".B1,
        RET_VAL               := MW   104,
        Sal                   := "VA_HMI".En1);
    NOP   0;
NETWORK
TITLE =Adaptacion entrada en altura 2
    CALL "Escalamiento" (
        ValorSensor           := "EA2",
        NivelSup              := 1.000000e+002,
        NivelInf              := 0.000000e+000,
```

```
        ValorEscalado           := MD   204);
    NOP   0;
NETWORK
TITLE =
    CALL "Ajuste a recta entradas" (
        IN                  := MD   204,
        A                   := "AjusteEn".A2,
        B                   := "AjusteEn".B2,
        RET_VAL                 := MW   106,
        Sal                 := "VA_HMI".En2);
    NOP   0;
NETWORK
TITLE =Adaptacion entrada en altura 3
    CALL "Escalamiento" (
        ValorSensor             := "EA3",
        NivelSup                := 1.000000e+002,
        NivelInf                := 0.000000e+000,
        ValorEscalado           := MD   208);
    NOP   0;
NETWORK
TITLE =
    CALL "Ajuste a recta entradas" (
        IN                  := MD   208,
        A                   := "AjusteEn".A3,
        B                   := "AjusteEn".B3,
        RET_VAL                 := MW   108,
        Sal                 := "VA_HMI".En3);
    NOP   0;
NETWORK
TITLE =Adaptacion entrada en altura 4
```

```
    CALL "Escalamiento" (
        ValorSensor          := "EA4",
        NivelSup             := 1.000000e+002,
        NivelInf             := 0.000000e+000,
        ValorEscalado        := MD   212);
    NOP   0;
NETWORK
TITLE =
    CALL "Ajuste a recta entradas" (
        IN                   := MD   212,
        A                    := "AjusteEn".A4,
        B                    := "AjusteEn".B4,
        RET_VAL              := MW   110,
        Sal                  := "VA_HMI".En4);
    NOP   0;
END_FUNCTION
```

B15. Función FC15 que realiza la adaptación de las señales de actuador en lenguaje AWL.

```
FUNCTION "Adaptacion salidas" : VOID
TITLE =
VERSION : 0.1
BEGIN
NETWORK
TITLE =Adaptacion de salida bomba 1
    CALL "Desescalamiento" (
        ValorSalReal         := "VA_HMI".Sal1,
        NivelSup             := 1.000000e+002,
        NivelInf             := 0.000000e+000,
        ValorSalEnt          := "SA1");
```

```
    NOP   0;
NETWORK
TITLE =Adaptacion de salida bomba 1
    CALL "Desescalamiento" (
        ValorSalReal          := "VA_HMI".Sal2,
        NivelSup              := 1.000000e+002,
        NivelInf              := 0.000000e+000,
        ValorSalEnt           := "SA2");
    NOP   0;
END_FUNCTION
```

B16. Bloque de organización OB1 en lenguaje AWL.

```
ORGANIZATION_BLOCK OB 1
TITLE = "Main Program Sweep (Cycle)"
VERSION : 0.1
VAR_TEMP
  OB1_EV_CLASS : BYTE ;//Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1 : BYTE ;     //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY : BYTE ;  //Priority of OB Execution
  OB1_OB_NUMBR : BYTE ;         //1 (Organization block 1, OB1)
  OB1_RESERVED_1 : BYTE ;       //Reserved for system
  OB1_RESERVED_2 : BYTE ;       //Reserved for system
  OB1_PREV_CYCLE : INT ;        //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE : INT ; //Minimum cycle time of OB1 (milliseconds)
  OB1_MAX_CYCLE : INT ;         //Maximum cycle time of OB1 (milliseconds)
  OB1_DATE_TIME : DATE_AND_TIME ;         //Date and time OB1 started
END_VAR
BEGIN
NETWORK
```

```
TITLE =
    U     M     10.0;
    =     M     10.1;
NETWORK
TITLE =Actualizacion de Señales de entrada sensores de presion
    CALL "Adaptacion entradas" ;
    NOP   0;
NETWORK
TITLE =Condiciones para inicar operacion manual
    UN    "BITs_ HMI".STOP_Manual;
    U(    ;
    O     "BITs_ HMI".START_Manual;
    O     M     50.0;
    )     ;
    U     "BITs_ HMI".Operacion_Manual;
    =     M     50.0;
NETWORK
TITLE =
    U     M     50.0;
    =     "BITs_ HMI".START_Manual;
NETWORK
TITLE =
    U     "BITs_ HMI".STOP_Manual;
    =     L     20.0;
    U     L     20.0;
    SPBNB _001;
    L     0.000000e+000;
    T     "VA_HMI".Sal1;
_001: NOP   0;
    U     L     20.0;
```

```
    SPBNB _002;
    L     0.000000e+000;
    T     "VA_HMI".Sal2;
_002: NOP   0;
    U     L     20.0;
    SPBNB _003;
    L     0;
    T     AW    10;
_003: NOP   0;
    U     L     20.0;
    SPBNB _004;
    L     0;
    T     AW    12;
_004: NOP   0;
NETWORK
TITLE =
    UN    "BITs_ HMI".STOP_Automatico;
    U(    ;
    O     "BITs_ HMI".START_Automatico;
    O     M     50.1;
    )     ;
    U     "BITs_ HMI".Operacion_Automatico;
    =     M     50.1;
NETWORK
TITLE =
    U     M     50.1;
    =     "BITs_ HMI".START_Automatico;
NETWORK
TITLE =
    U     "BITs_ HMI".STOP_Automatico;
```

```
      =     L     20.0;
      U     L     20.0;
      SPBNB _005;
      L     0.000000e+000;
      T     "VA_HMI".Sal1;
_005: NOP   0;
      U     L     20.0;
      SPBNB _006;
      L     0.000000e+000;
      T     "VA_HMI".Sal2;
_006: NOP   0;
      U     L     20.0;
      SPBNB _007;
      L     0;
      T     AW    10;
_007: NOP   0;
      U     L     20.0;
      SPBNB _008;
      L     0;
      T     AW    12;
_008: NOP   0;
NETWORK
TITLE =
      UN    "BITs_ HMI".STOP_Identificacion;
      U(    ;
      O     "BITs_ HMI".START_Identificacion;
      O     M     50.2;
      )     ;
      U     "BITs_ HMI".Operacion_Identificacion;
      =     M     50.2;
```

```
NETWORK
TITLE =
     U     M     50.2;
     =     "BITs_ HMI".START_Identificacion;
NETWORK
TITLE =Condiciones para llevar a cero las bombas
     U     "BITs_ HMI".STOP_Identificacion;
     =     L     20.0;
     U     L     20.0;
     SPBNB _009;
     L     0.000000e+000;
     T     "VA_HMI".Sal1;
_009: NOP   0;
     U     L     20.0;
     SPBNB _00a;
     L     0.000000e+000;
     T     "VA_HMI".Sal2;
_00a: NOP   0;
     U     L     20.0;
     SPBNB _00b;
     L     0;
     T     "Identificacion".Saltos1[87, 1];
_00b: NOP   0;
     U     L     20.0;
     SPBNB _00c;
     L     0;
     T     AW    10;
_00c: NOP   0;
     U     L     20.0;
     SPBNB _00d;
```

```
    L     0;
    T     AW    12;
_00d: NOP   0;
    U     L     20.0;
    SPBNB _00e;
    L     0;
    T     "Identificacion".Saltos1[87, 2];
_00e: NOP   0;
NETWORK
TITLE =inicializar contador con1 de identificacion
NETWORK
TITLE =
END_ORGANIZATION_BLOCK
```

B17. Bloque de organización OB35 en lenguaje AWL.

```
ORGANIZATION_BLOCK "CYC_INT5"
TITLE = "Cyclic Interrupt"
VERSION : 0.1
VAR_TEMP
  OB35_EV_CLASS : BYTE ;         //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event
class 1)
  OB35_STRT_INF : BYTE ;         //16#36 (OB 35 has started)
  OB35_PRIORITY : BYTE ;         //Priority of OB Execution
  OB35_OB_NUMBR : BYTE ;         //35 (Organization block 35, OB35)
  OB35_RESERVED_1 : BYTE ;       //Reserved for system
  OB35_RESERVED_2 : BYTE ;       //Reserved for system
  OB35_PHASE_OFFSET : WORD ;          //Phase offset (msec)
  OB35_RESERVED_3 : INT ;        //Reserved for system
  OB35_EXC_FREQ : INT ;  //Frequency of execution (msec)
  OB35_DATE_TIME : DATE_AND_TIME ;        //Date and time OB35 started
```

```
END_VAR
BEGIN
NETWORK
TITLE =
    UN    E      0.0;
    =     A      0.0;
NETWORK
TITLE =
    U     "BITs_ HMI".Operacion_Automatico;
    =     A      0.5;
NETWORK
TITLE =Actalizacion de matrice para control MPC FC1
    U     "BITs_ HMI".START_Automatico;
    SPBNB _001;
    CALL "Actualizacion para  FC1" (
        RET_VAL               := MW   100);
_001: NOP   0;
NETWORK
TITLE =Bloque de control MPC
    U     "BITs_ HMI".Operacion_Automatico;
    SPBNB _002;
    CALL "Control MPC" (
        RET_VAL               := MW   102);
_002: NOP   0;
NETWORK
TITLE =Calculo de las salidas hacia las bombas
    U     "BITs_ HMI".START_Identificacion;
    UN    "BITs_ HMI".ENT_TIPO_Ident;
    SPBNB _003;
    CALL "Ident Saltos" (
```

```
        RET_VAL             := MW   120);
_003: NOP   0;
NETWORK
TITLE =
    U    "BITs_ HMI".START_Identificacion;
    U    "BITs_ HMI".ENT_TIPO_Ident;
    SPBNB _005;
    CALL "Ident PRBS" (
        RET_VAL             := MW   122);
_005: NOP   0;
NETWORK
TITLE =
NETWORK
TITLE =
NETWORK
TITLE =
    U(   ;
    O    "BITs_ HMI".START_Automatico;
    O    "BITs_ HMI".START_Identificacion;
    O    "BITs_ HMI".START_Manual;
    )    ;
    SPBNB _006;
    CALL "Adaptacion salidas" ;
_006: NOP   0;
END_ORGANIZATION_BLOCK
```