UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

APLICACIÓN DE CONTROL DE MATRIZ DINÁMICA A PLANTA PILOTO

POR

Miguel Alonso Álvarez Burgos

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero(a) Civil Electrónico(a)

Profesor Guía
Juan Pablo Segovia Vera

Septiembre 2024
Concepción (Chile)

Agradecimientos

En primer lugar, quiero agradecer a mi familia por el apoyo incondicional que me brindaron durante toda mi etapa universitaria. A mi madre Marcela, por haberlo dado todo con tal de que no nos faltara nada, por haberme instado a perseguir una carrera universitaria, por ayudarme y guiarme con temas muy difíciles, por ser la mejor mamá del mundo. A mi hermano Rolando, por haberme acompañado en uno de los momentos más oscuros de mi vida, por las largas conversaciones, por los momentos agrios y los dulces, por haber sido mucho más que un hermano. A mi Padre Miguel, por el continuo apoyo económico, pese a no conversar mucho.

También quiero agradecer a mis amigos, en particular a Diego Gómez, Joaquín Lobos, Katerin Oyarzo, Esteban Rivas, Javier Cisternas, Nicolás Lobos y Cristóbal Moreira, por haberme acompañado, escuchado y apoyado. Por los buenos y malos momentos juntos, por las salidas y las oncecitas, por la distancia y la cercanía, por estar para mí cuando los necesité. A Matías Cuevas, por ser como un hermano y apoyarme de maneras que no se pueden describir. Y a María José Delgado, mi pareja durante el fin de este largo viaje, junto a su compañía y apoyo esta memoria de título fue posible.

A mi profesor guía, Juan Pablo Segovia, quien iniciara la llama de mi pasión por esta hermosa área de estudio. Gracias por comprender las situaciones que acomplejaron mi progreso durante este periodo.

Sumario

En el presente documento se detalla el desarrollo y la implementación del controlador Control de Matriz Dinámica (DMC) para el control del lazo de flujo de la planta piloto del Laboratorio de Control del Departamento de Ingeniería Eléctrica de la Universidad de Concepción. Para la identificación paramétrica del proceso se utiliza el Filtro de Kalman Extendido (EKF). Además, se desarrolla una interfaz gráfica para facilitar la interacción entre el operador de la planta y el algoritmo. Así, el presente proyecto consiste en estimar los parámetros de un proceso sujeto a ruido, para posteriormente aplicar control.

En primer lugar, se realiza un estudio bibliográfico del EKF. Luego, dicho algoritmo se implementa en una simulación en Matlab, con el fin de estimar los parámetros de un proceso de primer orden con retardo cuya medición, o salida, es afectada por ruido blanco. Al finalizar la sintonización del filtro se prosigue con la siguiente etapa.

En segundo lugar, se hace el estudio bibliográfico del DMC. Luego, es implementado en una simulación en Matlab, cuya entrada es la salida del EKF, es decir, los parámetros estimados del proceso de primer orden con retardo. Para finalizar esta etapa, se sintoniza el controlador para obtener una respuesta adecuada.

En tercer lugar, se implementa el controlador en un PLC, teniendo presente que estos son limitados en cuanto a su poder de procesamiento, y se desarrolla una interfaz gráfica. Por ende, tanto el EKF como una parte del algoritmo del DMC son ejecutados paralelamente al controlador en Matlab, siendo necesaria la comunicación entre el PLC y Matlab para la lectura y escritura de datos. Esta sección es probada y simulada utilizando Simulink para simular la planta piloto, Matlab para ejecutar el EKF y una parte del DMC, y RSLogix 5000 para simular el controlador, es decir, la parte final del DMC.

Finalmente, se hacen pruebas en la planta piloto y se validan las simulaciones. Se encuentra que el controlador regula de manera adecuada el proceso y que el filtro es capaz de estimar los parámetros del proceso con bajo error RMS.

Summary

The present document details the development and implementation of the Dynamic Matrix Control (DMC) controller for the flow loop control of the pilot plant in the Control Laboratory of the Department of Electrical Engineering at the University of Concepción. The Extended Kalman Filter (EKF) is used for the parametric identification of the process. Additionally, a graphical interface is developed to facilitate the interaction between the plant operator and the algorithm. Thus, the present project consists of estimating the parameters of a process subject to noise, in order to later apply control.

First, a literature review of the EKF is conducted. Then, this algorithm is implemented in a Matlab simulation to estimate the parameters of a first-order process with delay, whose measurement or output is affected by white noise. After tuning the filter, the next stage is carried out.

Second, a literature review of the DMC is conducted. Then, it is implemented in a Matlab simulation, where the input is the output of the EKF, i.e., the estimated parameters of the first-order process with delay. To conclude this stage, the controller is tuned to obtain an adequate response.

Third, the controller is implemented in a PLC, keeping in mind the processing power limitations of these devices, and a graphical interface is developed. Therefore, both the EKF and part of the DMC algorithm are executed in parallel to the controller in Matlab, making it necessary to establish communication between the PLC and Matlab for reading and writing data. This section is tested and simulated using Simulink to simulate the pilot plant, Matlab to execute the EKF and part of the DMC, and RSLogix 5000 to simulate the controller, i.e., the final part of the DMC.

Finally, tests are conducted on the pilot plant, and the simulations are validated. It is found that the controller adequately regulates the process and that the filter can estimate the process parameters with low RMS error.

Tabla de contenidos

Lista de tablas

Lista de figuras

Abreviaciones y nomenclatura

Mayúsculas

ARMA : autorregresivo de media móvil (AutoRegressive Moving Average).
DMC : control de matriz dinámica (Dynamic Matrix Control).
EKF : filtro de Kalman extendido (Extended Kalman Filter).
FAT : pruebas de atención en fábrica (Factory Acceptance Test).
GPC : control predictivo generalizado (Generalized Predictive Control).
HMI : interfaz humano-máquina (Human-Machine Interface).
KF : filtro de Kalman (Kalman Filter).
LA : lazo abierto.
LC : lazo cerrado.
LQG : lineal cuadrático gaussiano (Linear Quadratic Gaussian).
MISO : múltiples entradas única salida (Multiple Inputs Single Output).
MMSE : error cuadrático medio mínimo (Minimum Mean Squared Error)
OPC : OLE para control de procesos (OLE for Process Control).
PID : proporcional, integral y derivativo (Proporitonal, Interal, Derivative)
PLC : controlador lógico programable (Programmable Logic Controller).
PRBS : secuencia pseudo aleatoria de bits (Pseudo Random Bit Sequence).
RMSE : raíz del error cuadrático medio (Root Mean Squared Error).
SCADA : supervisión, control y adquisición de datos (Supervisory Control and Data Acquisition).
SISO : única entrada única salida (Single Input Single Output).
SS : estado estacionario (Steady State).
UKF : filtro de Kalman unscented (Unscented Kalman Filter).

Matrices

A : matriz de parámetros $(n \cdot n)$.
B : matriz de parámetros $(n \cdot p)$.
C : matriz de parámetros $(q \cdot n)$.
D : matriz de parámetros $(q \cdot p)$.
$\mathbf{F}_k$: matriz Jacobiana en el instante k.
$\mathbf{P}_k$: matriz de auto covarianza del proceso en el instante k.
$\mathbf{R}_k$: matriz de covarianza de ruido de medición en el instante k.
$\mathbf{Q}_k$: matriz de covarianza de ruido de proceso en el instante k.
G : matriz de coeficientes de la respuesta escalón.

Vectores

Y_k : colección de mediciones hasta el tiempo k.
x_k : vector de n variables de estados en el instante k.
y_k : vector de q variables de salida en el instante k.
u_k : vector de p variables de entrada en el instante k.
v_k : vector de ruido de medición en el instante k (p).
w_k : vector de ruido de proceso en el instante k (n).
φ : vector de parámetros.

1. Introducción

1.1. Introducción general

El Filtro de Kalman (KF) es un estimador de estados óptimo en el sentido de que este busca minimizar el RMSE de estimación. El algoritmo que se plantea corresponde al cálculo recursivo de la esperanza condicional, es decir, $\hat{x}_k = E[x_k|y_0, y_1, \dots, y_k]$ [1] y actualiza las estimaciones del estado y las covarianzas del error para refinar iterativamente las predicciones [3]. A diferencia de otros estimadores, como el observador de Luenberger, el KF considera que tanto el sistema como las mediciones poseen perturbaciones en la forma de ruido Gaussiano [1], efecto que debe ser considerado a la hora de estimar los estados del sistema. Por lo tanto, el KF presenta una ventaja sobre otros estimadores para tratar sistemas estocásticos.

La diferencia fundamental entre el KF y el Filtro de Kalman Extendido (EKF) es que el primero solo considera el caso lineal de los sistemas [2], mientras que el segundo abarca el caso no lineal mediante la linealización del sistema. El rendimiento del EKF es afectado tanto por el ruido de los estados como el de la medición [2].

Tanto el KF como el EKF son ampliamente usados en distintas aplicaciones donde se requiere predecir eventos, por ejemplo, en [3] se predice el nivel de un río para prevenir inundaciones que puedan perturbar a los habitantes de la zona colindante, y en [4] se busca predecir la posición de un objeto. Pese a que el EKF se plantea como un estimador de estados, este también puede ser empleado para estimar parámetros (por ejemplo, de un sistema de primer o segundo orden), mediante la conjunción de dichos parámetros al vector de estados del modelo con el que se trabaja el sistema [5], [6] y [7].

Un parámetro relevante por estimar es el tiempo muerto, es decir, el retardo que existe entre la aplicación de una entrada y la respuesta en la salida. Obviar este parámetro en un controlador que depende de un modelo matemático puede resultar en el deterioro del esquema de control utilizado, por ejemplo, provocando que el controlador sea inestable [6]. Además, en sistemas con una dinámica rápida, al considerar el retardo del controlador a la hora de calcular la salida de este se obtienen mejores respuestas [7].

Para el presente proyecto se usa el controlador Control de Matriz Dinámica (DMC). El DMC nace a partir de técnicas que representan la respuesta dinámica de procesos a través de un conjunto de coeficientes numéricos, así la matriz de coeficientes que describe la dinámica del sistema es la base para este algoritmo [9]. Como se menciona en [10], este controlador es utilizado en el control

de procesos en refinerías de petróleo y otras industrias. Algunos ejemplos de aplicación se muestran en [10], [11] y [12]. El motivo del uso de este controlador radica en la obtención de los parámetros estimados por el EKF, con los cuales se puede calcular la respuesta escalón del sistema y así generar predicciones explícitas en base a esta [12].

1.2. Trabajos previos

1.2.1. Del filtro

♣ B. M. Quine, "A derivative-free implementation of the extended Kalman filter," *Automatica*, vol. 42, no. 11, pp. 1927–1934, Nov. 2006, doi: 10.1016/j.automatica.2006.06.013

En este trabajo se muestra el desarrollo del algoritmo del Filtro de Kalman Extendido, se mencionan sus restricciones, las complicaciones de su implementación, y aplicaciones. Dentro de dichas aplicaciones se encuentra el área industrial, campo que compete a la presente memoria.

♣ Q. Li, R. Li, K. Ji and W. Dai, "Kalman Filter and Its Application," *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Tianjin, China, 2015, pp. 74-77, doi: 10.1109/ICINIS.2015.35.

En este trabajo se muestran las distintas variedades del KF (KF, EKF, UKF) y la aplicación donde es adecuado su uso. Además, se muestra una gran desventaja a la hora del uso del EKF, que corresponde a que si las matrices de covarianza (que se utilizan como parámetros de sintonización del filtro) no son estimadas adecuadamente, el error producido por esto es acarreado provocando que la estimación diverja.

♣ R. Adnan, F. A. Ruslan, A. M. Samad and Z. Md Zain, "Extended Kalman Filter (EKF) prediction of flood water level," *2012 IEEE Control and System Graduate Research Colloquium*, Shah Alam, Malaysia, 2012, pp. 171-174, doi: 10.1109/ICSGRC.2012.6287156.

En este trabajo se utiliza el EKF para predecir el nivel de agua de un río y así poder prever posibles inundaciones. Se discute la necesidad del uso de un estimador que sea capaz de lidiar con sistemas no lineales. Se utilizan datos reales para hacer las predicciones, las cuales son simuladas en Matlab. Se concluye que el EKF predice con éxito los niveles de agua durante las inundaciones con alta precisión, dado su bajo RMSE de predicción

♣ S. Yang and M. Baum, "Extended Kalman filter for extended object tracking," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017, pp. 4386-4390, doi: 10.1109/ICASSP.2017.7952985.

En este trabajo se utiliza el EKF para estimar la posición de objetos que requieren múltiples mediciones dada la complejidad de su forma, siendo esta aproximada por una elipse. Se utilizan mediciones para estimar tanto estados cinemáticos como de forma. Se tiene que el EKF logra adaptarse rápidamente a cambios en la forma del objeto, produciendo una buena estimación de su posición.

♣ T. Yoshimura, K. Konishi, and T. Soeda, "A modified extended Kalman filter for linear discrete-time systems with unknown parameters," *Automatica*, vol. 17, no. 4, pp. 657–660, Jul. 1981, doi: 10.1016/0005-1098(81)90041-8.

En este trabajo se desarrolla un filtro, basado en el algoritmo del KF, para la estimación de parámetros de un sistema lineal discreto. Se encuentra que, al omitir la covarianza entre los estados y los parámetros a estimar, además de agregar un compensador, se asegura la convergencia del algoritmo.

♣ X. Yang and O. Marjanovic, "LQG Control with Extended Kalman Filter for Power Systems with Unknown Time-Delays," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 3708–3713, Jan. 2011, doi: 10.3182/20110828-6-it-1002.03082.

En este trabajo se muestra el uso del algoritmo EKF, como estimador de parámetros, en conjunto con un controlador del tipo lineal cuadrático gaussiano (LQG) para controlar un proceso. Para la estimación del tiempo muerto se hace uso de la aproximación de Padé de primer orden para aproximar el retardo como un cociente de polinomios y así poder ser introducido a una función de transferencia cuyos parámetros son estimados con el EKF.

♣ A. Probst, O. Sawodny, and M.E. Magaña, "Using a Kalman filter and a Padé approximation to estimate random time delays in a networked feedback control system," *IET Control Theory and Applications*, vol. 4, no. 11, pp. 2263–2272, Nov. 2010, doi: 10.1049/iet-cta.2009.0339.

En este trabajo se aplica la aproximación de Padé para introducir el tiempo muerto al modelo del sistema a trabajar. Se utiliza el EKF para hacer predicciones acerca del estado del sistema y el retardo asociado a la salida dada una entrada. Se prueba el algoritmo en un sistema con una dinámica rápida y por tanto altamente dependiente de los retardos que pueda incluir el controlador. Se concluye que utilizar esta estrategia mejora el rendimiento de la regulación del sistema, puesto que el controlador escogido toma en consideración los retardos para calcular la su salida.

♣ Ch. Venkateswarlu and Rama Rao Karri, *Optimal State Estimation for Process Monitoring, Fault Diagnosis and Control*. Elsevier, 2022, sec. 4.4, pp. 65–73.

En esta sección del libro se trata la teoría del algoritmo para el EKF. Se explica la importancia de la elección de los parámetros iniciales del vector de predicción de estados $\hat{x}$, matriz de covarianza de los estados **P**, matriz de covarianza del ruido de proceso **Q** y matriz de covarianza del ruido de medición **R**. Además, se muestra hace mención del efecto que cada uno de estos parámetros tiene sobre el desempeño del filtro.

1.2.2. Del controlador

♣ C. R. Cutler and B. L. Ramaker, "Dynamic matrix control - A computer control algorithm," vol. 17, no. 17, p. 72, Jan. 1980, doi: 10.1109/jacc.1980.4232009.

En este trabajo se muestra el desarrollo del DMC para incorporar control del tipo *feedforward* y multivariable. Se menciona que al añadir el tiempo muerto del sistema al controlador se obtienen respuestas similares a las del controlador PID con la parte integral suprimida. Se hace control de la temperatura de un horno con controladores convencionales y con el DMC. Se concluye que el DMC proporciona una salida más suave y estable que las entregadas por los otros algoritmos.

♣ Z. Jun, M. Qingjin and Y. Hongliang, "The application of dynamic matrix control in the grate cooler," *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, 2017, pp. 2630-2633, doi: 10.1109/IAEAC.2017.8054501.

En este trabajo se aplica el DMC para el control de un equipo de una planta de cemento, el enfriador de rejilla. El modelo utilizado para representar al sistema es una ecuación de transferencia de primer orden con retardo, cuyos parámetros son estimados mediante mínimos cuadrados en Matlab. Se obtiene que el controlador logra operar de manera adecuada, contrastándolo con control manual o semi manual dado por operadores.

♣ Y. Hao and H. Yan, "Application of Dynamic Matrix Control in Liquid Level Control," *2020 IEEE 4th International Conference on Frontiers of Sensors Technologies (ICFST)*, Shanghai, China, 2020, pp. 59-62, doi: 10.1109/ICFST51577.2020.9294771.

En este trabajo se utiliza un controlador DMC para controlar el nivel de un estanque. El algoritmo es implementado directamente en el PLC. Sin embargo, y como se menciona, al requerir órdenes altos en las matrices, y dado que estas se necesita el cálculo de la inversa de estas, se necesita de un alto poder computacional.

♣ V. L. Chuong and T. N. Luan Vu, "Identification and Dynamic Matrix Control algorithm for

a heating process," *2017 International Conference on System Science and Engineering (ICSSE)*, Ho Chi Minh City, Vietnam, 2017, pp. 642-645, doi: 10.1109/ICSSE.2017.8030954.

En este trabajo se aplica el algoritmo DMC para el control de la temperatura de aire encapsulado en una cámara, mediante la regulación del voltaje aplicado a resistencias. Para la identificación del sistema se hace uso de una señal PRBS. El sistema es modelado como una función de transferencia continua, con un polo y un cero, y retardo incluido. Se obtiene un controlador que regula adecuadamente el sistema, además de soportar perturbaciones.

♣ A. Zheng, "Does nonlinear dynamic matrix control provide integral control?," *Computers & Chemical Engineering*, vol. 23, no. 11–12, pp. 1753–1756, Jan. 2000, doi: 10.1016/s0098-1354(99)00323-3.

En este trabajo se analiza la versión no lineal del algoritmo DMC (NDMC). Se muestran las condiciones necesarias para que tanto el DMC como el NDMC (ambos en su versión con horizonte de predicción finito puesto que un horizonte infinito no es realizable) tengan parte integral. En particular, se menciona la necesidad de una función objetivo con mínimo global. Se concluye que el NDMC puede tener parte integral para todo cambio de referencia mientras se mantengan las condiciones planteadas.

♣ W. Klopot, T. Klopot, and M. Metzger, "Adaptive and Non-Adaptive Dynamic Matrix Control for Conical Tank," *IFAC Proceedings Volumes*, vol. 42, no. 13, pp. 543–548, Jan. 2009, doi: 10.3182/20090819-3-PL-3002.00094.

En este trabajo se aplica el algoritmo DMC para controlar el nivel de un estanque cónico. Además, se comparan los resultados de dos maneras distintas de aplicar el algoritmo: en su versión adaptativa y la no adaptativa. Se obtiene que el controlador adaptativo tiene un rendimiento muy superior a la versión no adaptativa, lo cual viene dado de que un estanque cónico es altamente no lineal, lo que produce que los parámetros de su modelo, y por consiguiente la respuesta escalón, varíen significativamente.

♣ P. Lundström, J. H. Lee, M. Morari, and S. Skogestad, "Limitations of dynamic matrix control," *Computers & Chemical Engineering*, vol. 19, no. 4, pp. 409–421, Apr. 1995, doi: 10.1016/0098-1354(94)00063-t.

En este trabajo se mencionan las limitaciones del algoritmo DMC para su aplicación en planta, las cuales son: para conseguir un buen desempeño del controlador se necesita una excesiva cantidad de coeficientes de la respuesta escalón del sistema; el rendimiento del controlador es afectado por el

ruido; el controlador no es robusto. Se concluye que un DMC puede tener un rendimiento pobre independientemente de su sintonización.

♣ J. A. Rossiter, *A First Course in Predictive Control*. CRC Press, 2018, ch. 4, pp. 117–147.

En este libro se trata la teoría acerca de control predictivo generalizado (GPC). Se discute acerca de las distintas funciones objetivos que se pueden tener, su optimización y por qué la forma cuadrática es la más utilizada. Se trata el control DMC, que corresponde a una subclase de GPC.

1.2.3. Discusión

En la revisión bibliográfica se encuentra que los algoritmos que se aplican en el presente proyecto son realizables en la planta piloto, dada la aplicación que se busca. Se muestra que el controlador es apto para realizar control de manera adecuada, como se ve en [13] el control DMC cuenta con error en estado estacionario 0, es decir, provee control integral. Por otra parte, también se encuentra la desventaja que implica usar un modelo fijo para un sistema no lineal, puesto que las predicciones del controlador son hechas en base a este [14], además de requerir un alto poder computacional de acuerdo con la cantidad de predicciones que sean necesarias para realizar un control adecuado [15]. Se menciona en [9] que cualquier sistema que pueda ser descrito mediante un sistema de ecuaciones diferenciales lineales puede utilizar el DMC, y como se muestra más adelante, el proceso a trabajar puede ser descrito como tal.

Se encuentra que el desempeño del EKF para la identificación paramétrica depende en gran medida de la elección de las matrices de covarianza. En particular, en [8] se menciona que, al escoger valores pequeños para la matriz de covarianza de ruido de proceso, el filtro depende excesivamente del modelo escogido y no utiliza las mediciones del proceso para corregir los estados. Esto resulta de gran importancia si se desean recalcular los estados, o parámetros, del sistema en tiempo real.

1.3. Hipótesis de trabajo

- Es posible la implementación del EKF para estimación de parámetros del sistema en tiempo real e implementarlo en conjunto a un controlador DMC.
- Los parámetros estimados serán lo suficientemente precisos si hay excitación mínima o suficiente.
- El EKF será capaz de estimar la salida del proceso, incluso cuando se produzcan cambios de parámetros de acuerdo con la zona de operación de dicho proceso.

- El controlador regulará el sistema de manera adecuada.

1.4. Objetivos

1.4.1. Objetivo general

Implementación del controlador DMC en el PLC ubicado en el Laboratorio de Control del Departamento de Ingeniería Eléctrica para realizar control en tiempo real de flujo. Además, hacer estimación paramétrica del proceso a trabajar mediante el uso del algoritmo EKF.

1.4.2. Objetivos específicos

- Investigar y comprender los algoritmos y métodos actuales utilizados en la estimación del retardo variable dentro de modelos del tipo SISO.
- Buscar posibles mejoras en los métodos propuestos previamente.
- Implementar el EKF para la identificación paramétrica del proceso mediante su excitación con una señal PRBS.
- Implementar un DMC para regular flujo, en el PLC ubicado en la planta piloto.
- Realizar una conexión OPC para el envío de variables en tiempo real desde Matlab al PLC.
- Realización de pruebas de los algoritmos en planta piloto.

1.5. Alcances y limitaciones

- Los parámetros del sistema son calculados en tiempo real por el EKF.
- Los parámetros del DMC no son actualizados en tiempo real.
- La implementación se lleva a cabo en un PLC Allen Bradley ControlLogix 1756.
- La interfaz gráfica es desarrollada en FactoryTalk.
- Las pruebas son realizadas en la planta piloto.

1.6. Temario y metodología

En el capítulo 2 se muestra el desarrollo teórico de los algoritmos a implementar, tanto del EKF como el DMC. En el capítulo 3 se implementan estos algoritmos primeramente en simulación, para luego ser probados la planta piloto. En el capítulo 4 se muestra la interfaz gráfica diseñada para el proyecto, junto con una breve descripción de las pantallas que la componen; además, se muestra cómo se logra la conexión OPC entre Matlab y el PLC. En el capítulo 5 se prueban distintas configuraciones para el controlador y se muestra la dependencia de la sintonización para controlar

distintos niveles de flujo. En el capítulo 6 se muestran métricas de desempeño para la estimación paramétrica. En el capítulo 7 se analizan los resultados encontrados en los capítulos 4 y 5. Finalmente, en el capítulo 8 se muestran las conclusiones del trabajo.

La metodología adoptada en el presente trabajo corresponde a comenzar con el desarrollo de los algoritmos deseados en Matlab, para luego proceder a realizar pruebas de aceptación en fábrica (FAT) mediante la simulación de la planta en Simulink en conjunto con RSLogix. Finalmente, se implementan los algoritmos en el PLC y se analizan los resultados.

2. Desarrollo de algoritmos para filtro de Kalman en tiempo real y controlador DMC

2.1. Filtro de Kalman

Un estimador $\phi(y)$ es una función que mapea cada medición, y, a un valor estimado, $\hat{x}$. Se tiene que el estimador con error cuadrático medio mínimo (MMSE) para X dado $Y = y$, que minimiza la esperanza $E[|X - \phi(y)|^2]$ está dado por $\phi_{MMSE}(y) = E[X|Y = y]$. Por lo tanto, un estimador basado en esta regla es óptimo.

El Filtro de Kalman (KF) es un método recursivo para el cálculo de la regla antes enunciada. Otro punto importante por mencionar de este estimador es que no presenta sesgo, es decir, los estados estimados convergen al valor real. Además, el valor en estado estacionario no depende de las condiciones iniciales de $\hat{x}_0$ o P_0.

Dado un sistema estocástico, lineal, descrito por

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k + w_k \tag{2.1}$$

$$y_k = \mathbf{C}x_k + \mathbf{D}u_k + v_k \tag{2.2}$$

Donde ambos vectores de ruido son considerados como ruido Gaussiano, es decir, $w_k \sim \mathcal{N}(0, \mathbf{Q}_k)$, $v_k \sim \mathcal{N}(0, \mathbf{R}_k)$. Se asume que $w_k \perp v_j, \forall k, j$, y $(w_k \perp w_j, v_k \perp v_j), \forall k \neq j$. Es decir, no están correlacionados.

Dados $\hat{x}_k$ y $\mathbf{P}_k$, para calcular $\hat{x}_{k+1}$ y $\mathbf{P}_{k+1}$ se divide el algoritmo en dos partes: predicción y actualización de medición, tal como se muestra en el siguiente esquema:

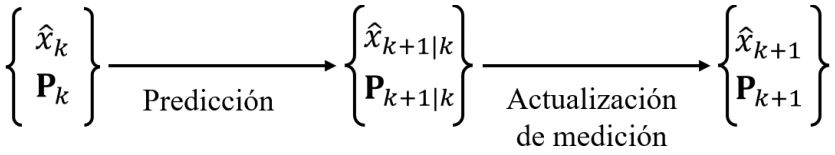

Figura 2.1: Esquema de predicción y actualización KF

A los valores obtenidos entre la etapa de predicción y la actualización se les llama usualmente valores a priori, puesto que es lo que el algoritmo estima. A los términos luego de la actualización de medición se les conoce como valores a posteriori.

2.1.1. Etapa de predicción

En esta etapa se computan $\hat{x}_{k+1|k}$ y $\mathbf{P}_{k+1}$ usando $\hat{x}_k$ y $\mathbf{P}_k$. Sea Y_k el vector con todas las mediciones hasta el tiempo k,

$$\begin{aligned}\hat{x}_{k+1|k} &= E[\hat{x}_{\mathrm{k+1}}|\mathrm{Y_k}] \\ &= E[\mathbf{A}x_k + \mathbf{B}u_k + w_k|Y_k] \\ &= \mathbf{A}E[x_k|Y_k] + \mathbf{B}u_k + E[w_k|Y_k]\end{aligned}$$

Se tiene que $\hat{x}_k = E[x_k|Y_k]$ y $E[w_k|Y_k] = 0$, por lo tanto

$$\hat{x}_{k+1|k} = \mathbf{A}\hat{x}_k + \mathbf{B}u_k \tag{2.3}$$

Luego,

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathrm{cov}(x_{k+1}, x_{k+1}|Y_k) \\ &= E\left[\left(x_{k+1} - \hat{x}_{k+1|\mathrm{k}}\right)\left(x_{k+1} - \hat{x}_{k+1|\mathrm{k}}\right)^T \middle| Y_k\right] \\ &= E[(\mathbf{A}x_k + \mathbf{B}u_k + w_k - \mathbf{A}\hat{x}_k + \mathbf{B}u_k)(\mathbf{A}x_k + \mathbf{B}u_k + w_k - \mathbf{A}\hat{x}_k + \mathbf{B}u_k)^T|Y_k] \\ &= E[\mathbf{A}(x_k - \hat{x}_k)\mathbf{A}^T|Y_k] + E[w_k w_k^T|Y_k] + E[\mathbf{A}(x_k - \hat{x}_k)w_k^T|Y_k] + E[w_k(x_k - \hat{x}_k)\mathbf{A}^T|Y_k] \\ &= \mathbf{A}E[(x_k - \hat{x}_k)|Y_k]\mathbf{A}^T + E[w_k w_k^T|Y_k] + E[\mathbf{A}(x_k - \hat{x}_k)w_k^T|Y_k] + E[w_k(x_k - \hat{x}_k)\mathbf{A}^T|Y_k]\end{aligned}$$

con $E[\mathbf{A}(x_k - \hat{x}_k)w_k^T|Y_k] = E[w_k(x_k - \hat{x}_k)\mathbf{A}^T|Y_k] = 0$, $E[w_k w_k^T|Y_k] = \mathbf{Q}_k$ y $E[(x_k - \hat{x}_k)|Y_k] = \mathbf{P}_k$. Por lo tanto,

$$\mathbf{P}_{k+1|\mathrm{k}} = \mathbf{A}\mathbf{P}_k\mathbf{A}^{\boldsymbol{T}} + \mathbf{Q}_k \tag{2.4}$$

2.1.2. Etapa de actualización de medición

En esta etapa se busca calcular $\hat{x}_{k+1} = E[\hat{x}_{k+1}|Y_{k+1}] = E[x_{k+1}|Y_k, y_{k+1}]$ utilizando los resultados de la etapa previa. Se utiliza el hecho de que las variables x_{k+1} y y_{k+1} son Gaussianas, y por tanto pueden ser escritas como conjuntamente Gaussianas, es decir,

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix}|Y_k \sim \mathcal{N}\left(\begin{pmatrix} E[x_{k+1}|Y_k] \\ E[y_{k+1}|Y_k] \end{pmatrix}, \begin{bmatrix} \mathrm{cov}(x_{k+1}, x_{k+1}|Y_k) & \mathrm{cov}(x_{k+1}, y_{k+1}|Y_k) \\ \mathrm{cov}(y_{k+1}, x_{k+1}|Y_k) & \mathrm{cov}(y_{k+1}, y_{k+1}|Y_k) \end{bmatrix}\right)$$

Se usa

$$E[x_{k+1}|Y_k, y_{k+1}] = E[x_{k+1}|Y_k] + \mathrm{cov}(x_{k+1}, y_{k+1}|Y_k)\mathrm{cov}(y_{k+1}, y_{k+1}|Y_k)^{-1}(y_{k+1} - E[y_{k+1}|Y_k])$$

Donde

$$E[x_{k+1}|Y_k] = \hat{x}_{k+1|k}$$

$$E[y_{k+1}|Y_k] = \mathbf{C}\hat{x}_{k+1|k} + \mathbf{D}u_{k+1}$$

$$\mathrm{cov}(x_{k+1}, y_{k+1}|Y_k) = \mathbf{P}_{k+1|k}\mathbf{C}^T$$

$$\mathrm{cov}(y_{k+1}, y_{k+1}|Y_k)^{-1} = \left(\mathbf{C}P_{k+1|k}\mathbf{C}^T + \mathbf{R}_k\right)^{-1}$$

Por lo tanto, y definiendo

$$K_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}^T\left(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \mathbf{R}_k\right)^{-1} \tag{2.5}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_{k+1}\left(y_{k+1} - \mathbf{C}\hat{x}_{k+1|k} - \mathbf{D}u_{k+1}\right) \tag{2.6}$$

Finalmente, se actualiza la matriz de covarianza. Se usa

$$\text{cov}(x_{k+1}, x_{k+1}|Y_{k+1}, y_{k+1}) =$$
$$\text{cov}(x_{k+1}, x_{k+1}|Y_k) - \text{cov}(x_{k+1}, y_{k+1}|Y_k)\text{cov}(y_{k+1}, y_{k+1}|Y_k)^{-1}\text{cov}(y_{k+1}, x_{k+1}|Y_k)$$

Donde

$$\text{cov}(x_{k+1}, x_{k+1}|Y_k) = \mathbf{P}_{k+1|k}$$
$$\text{cov}(x_{k+1}, y_{k+1}|Y_k)\text{cov}(y_{k+1}, y_{k+1}|Y_k)^{-1} = K_{k+1}$$
$$\text{cov}(y_{k+1}, x_{k+1}|Y_k) = \mathbf{C}\mathbf{P}_{k+1|k}$$

Por lo tanto,

$$\mathbf{P}_{k+1} = (\mathbf{I} + K_{k+1}\mathbf{C})\mathbf{P}_{k+1|k} \tag{2.7}$$

2.1.3. Resumen del algoritmo

Para su uso práctico, es necesario ajustar los índices correspondientes a tiempos futuros en las ecuaciones (2.4)-(2.6). El KF queda representado por el siguiente diagrama:

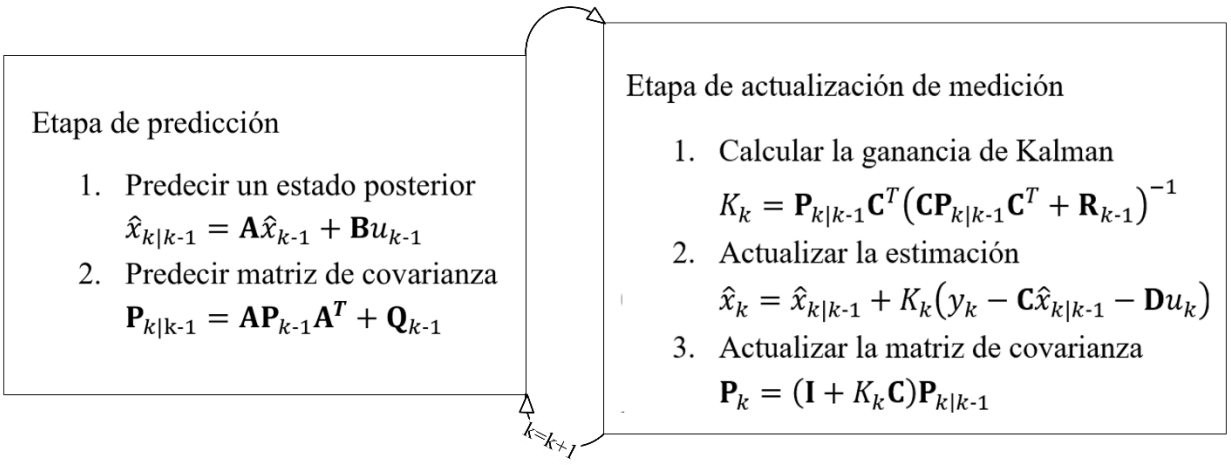

Figura 2.2: Algoritmo del KF

2.2. Filtro de Kalman Extendido

Tal como dice el nombre de este filtro, es una extensión del KF diseñado para tratar con sistemas no lineales. Al igual que el KF, el Filtro de Kalman Extendido (EKF) se compone de dos etapas: predicción y actualización de medición. Además, sigue el mismo esquema planteado en la Figura 2.1.

Dado un sistema estocástico, no lineal, descrito por

$$x_{k+1} = f(x_k, u_k) + w_k \tag{2.8}$$

$$y_k = h(x_k, u_k) + v_k \tag{2.9}$$

Donde ambos vectores de ruido son considerados como ruido Gaussiano, es decir, $w_k \sim \mathcal{N}(0, \mathbf{Q}_k)$, $v_k \sim \mathcal{N}(0, \mathbf{R}_k)$. Se asume que $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$, $x_0 \perp w_k$, $x_0 \perp v_k$, $w_k \perp v_j, \forall k, j$, y $w_k \perp w_j, v_k \perp v_j, \forall k \neq j$.

Dados $\hat{x}_k = E[x_k|Y_k]$ y $\mathbf{P}_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T|Y_k]$, se desea calcular $\hat{x}_{k+1|k} = E[x_{k+1}|Y_k]$ y $\mathbf{P}_{k+1|k} = E\left[\left(x_{k+1} - \hat{x}_{k+1|k}\right)\left(x_{k+1} - \hat{x}_{k+1|k}\right)^T\middle|Y_k\right]$.

2.2.1. Predicción vía linealización

Para seguir con el esquema planteado en la sección anterior, es necesario linealizar las ecuaciones (2.8) y (2.9). El EKF linealiza en torno al estado estimado actual, $\hat{x}_k$, y la entrada, u_k. Por lo tanto,

$$f(x_k, u_k) \approx f(\hat{x}_k, u_k) + \left.\frac{\partial f(x_k, u_k)}{\partial x}\right|_{\hat{x}_k, u_k} (x_k - \hat{x}_k)$$

Definiendo $\mathbf{F}_k \triangleq \left.\frac{\partial f(x_k, u_k)}{\partial x}\right|_{\hat{x}_k, u_k}$, el Jacobiano de $f(x_k, u_k)$. Así,

$$x_{k+1} \approx \mathbf{F}_k x_k + f(\hat{x}_k, u_k) - \mathbf{F}_k \hat{x}_k + w_k$$

Definiendo $\tilde{u}_k \triangleq f(\hat{x}_k, u_k) - \mathbf{F}_k \hat{x}_k$ para la ecuación anterior, se consigue un esquema idéntico al planteado durante el desarrollo del KF. Es decir, se obtiene que

$$x_{k+1} \approx \mathbf{F}_{\mathrm{k}} \mathrm{x}_{\mathrm{k}} + \tilde{u}_k + w_k$$

Ahora, el estimador queda dado por

$$\begin{aligned}
\hat{x}_{k+1|k} &= E[x_{k+1}|Y_k] \\
&= E[\mathrm{F}_{\mathrm{k}} \mathrm{x}_{\mathrm{k}} + \tilde{u}_k + w_k|Y_k] \\
&= \mathbf{F}_k E[x_k|Y_k] + \tilde{u}_k + 0 \\
&= \mathbf{F}_k \hat{x}_k + f(\hat{x}_k, u_k) - \mathbf{F}_k \hat{x}_k \\
&= f(\hat{x}_k, u_k)
\end{aligned}$$

Por lo tanto,

$$\hat{x}_{k+1|k} = f(\hat{x}_k, u_k) \tag{2.10}$$

Finalmente, siguiendo un desarrollo idéntico al mostrado en la sección 2.1.1, se llega a

$$\mathbf{P}_{k+1|k} = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^T + \mathbf{Q}_k \tag{2.11}$$

Que corresponde a reemplazar **A** por $\mathbf{F}_k$ en la ecuación (2.4).

2.2.2. Actualización de medición vía linealización

Se tienen $\hat{x}_{k+1|k} = E[x_{k+1}|Y_k]$ y $\mathbf{P}_{k+1} = E\left[\left(x_{k+1} - \hat{x}_{k+1|k}\right)\left(x_{k+1} - \hat{x}_{k+1|k}\right)^T \middle| Y_k\right]$. Se linealiza (2.9) para el tiempo $k+1$

$$y_{k+1} \approx h\left(\hat{x}_{k+1|k}, u_{k+1}\right) + \left.\frac{\partial h(x_{k+1}, u_{k+1})}{\partial x}\right|_{\hat{x}_{k+1|k}, u_{k+1}} \left(x_{k+1} - \hat{x}_{k+1|k}\right) + v_{k+1}$$

Definiendo $\mathbf{H}_{k+1} \triangleq \left.\frac{\partial h(x_{k+1}, u_{k+1})}{\partial x}\right|_{\hat{x}_{k+1|k}, u_{k+1}}$. Así,

$$y_{k+1} \approx \mathbf{H}_{k+1} x_{k+1} + h\left(\hat{x}_{k+1|k}, u_{k+1}\right) - \mathbf{H}_{k+1} \hat{x}_{k+1|k} + v_{k+1}$$

Y nuevamente, siguiendo el procedimiento descrito en la sección 2.1.2, se llega a

$$K_{k+1} = \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T \left(\mathbf{H}_{k+1} \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T + \mathbf{R}_{k+1}\right)^{-1} \tag{2.12}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + K_{k+1}\left(y_{k+1} - h\left(\hat{x}_{k+1|k}, u_{k+1}\right)\right) \tag{2.13}$$

$$\mathbf{P}_{k+1} = (\mathbf{I} + K_{k+1} \mathbf{H}_{\boldsymbol{k+1}}) \mathbf{P}_{k+1|k} \tag{2.14}$$

2.3. Aprovechamiento del EKF para estimación de parámetros

Dado el sistema descrito por las ecuaciones

$$\dot{x}(t) = f(x, u, \varphi) + w_x(t)$$
$$y(t) = h(x, u, \varphi) + v(t)$$

Con φ los parámetros a estimar del modelo. El objetivo es utilizar las entradas u_k y las mediciones y_k para estimar φ y $x(t)$, lo cual se logra considerando los parámetros como estados. Se impone $\dot{\varphi} = 0 + w_\varphi(t)$. Así, se define

$$\tilde{x}(t) \triangleq \begin{bmatrix} x(t) \\ \varphi \end{bmatrix}$$

Por lo tanto,

$$\dot{\tilde{x}}(t) = \begin{bmatrix} f(x, u, \varphi) \\ 0 \end{bmatrix} + \begin{bmatrix} w_x(t) \\ w_\varphi(t) \end{bmatrix}$$
$$= \tilde{f}(\tilde{x}, u) + \tilde{w}(t)$$
$$y(t) = h(\tilde{x}, u) + v(t)$$

La matriz de covarianza de ruido de proceso queda definida como

$$\widetilde{\mathbf{Q}}_k = \begin{bmatrix} \mathbf{Q}_{x,k} & 0 \\ 0 & \mathbf{Q}_{\varphi,k} \end{bmatrix}$$

Finalmente, para aplicar estimación de parámetros basta con discretizar el sistema obtenido y aplicar EKF mediante el cómputo de las ecuaciones (2.10)-(2.14)**.**

2.4. Controlador Control de Matriz Dinámica

El controlador Control de Matriz Dinámica (DMC) es del tipo control predictivo generalizado (GPC). Este controlador usa la respuesta escalón del proceso para estimar valores futuros y adelantarse a la respuesta de este en base a dichas estimaciones, la entrada actual y entradas futura. Así, este controlador tiene dos horizontes: de predicción y de control. A falta de valores futuros de la entrada, se puede asumir que el valor actual de esta se mantiene constante en el horizonte de predicción.

El algoritmo requiere de una estimación de salida y una función de costo. La estimación está dada por la respuesta escalón del sistema, mientras que la función de costo es relativamente arbitraria. Sin embargo, como se muestra en la revisión bibliográfica, es importante que esta tenga un mínimo global, por lo que es común el uso de funciones cuadráticas. Esta última es la encargada de ponderar distintas diferencias para regular la salida. La función de costo escogida hace uso del error, que corresponde la diferencia entre la salida actual y la salida deseada; y la actividad de la entrada, que corresponde a la diferencia entre la entrada actual y la entrada un tiempo atrás. Es decir, la función de costo queda definida como

$$J = \sum_{j=1}^{P} \delta(j)[\hat{y}(t+j|t) - r(t+j)]^2 + \sum_{j=1}^{N} \lambda(j)[\Delta u(t+j-i)]^2 \tag{2.15}$$

Siendo δ y λ pesos para el error de predicción y variación en la salida, respectivamente; P y N los horizontes de predicción y control, respectivamente; $\hat{y}$ el valor predicho mediante respuesta escalón; r la referencia. Notando que se hace una modificación, de acuerdo con lo revisado en [16], a la ecuación (4) mostrada en [14], agregando el término $\delta(j)$.

La representación matricial de (2.15) queda dada por

$$J = (\hat{y} - r)^T \boldsymbol{\delta} (\hat{y} - r) + \boldsymbol{\lambda} u^T u \tag{2.16}$$

Usando la ecuación de predicción en [14], que está dada por la respuesta escalón del sistema,

$$y(t+k|t) = \sum_{i=1}^{k} g_i \, \Delta u(t+k-i) + \sum_{j=1}^{M} \left(g_{j+k} - g_j\right) \Delta u(t-j) + y(t) \tag{2.17}$$

Donde M es un valor arbitrario, donde la respuesta escalón se considera invariante, por lo cual los términos siguientes son obviados.

La respuesta escalón, en representación matricial, queda dada por

$$\mathbf{G} = \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_P & g_{P-1} & \cdots & g_1 \end{bmatrix}$$

Por lo tanto, la ecuación (2.17), en representación matricial, y haciendo $y_f = \sum_{j=1}^{M} (g_{j+k} - g_j)\Delta u(t-j) + y(t)$, es

$$\hat{y} = \mathbf{G}\Delta u + y_f \tag{2.18}$$

Finalmente, se reemplaza (2.18) en (2.16) y se minimiza esta función respecto a Δu, es decir, se hace $\partial J / \partial \Delta u = 0$, resultando

$$\Delta u = \left(\mathbf{G}^T(\delta \mathbf{I})\mathbf{G} + (\lambda \mathbf{I})\right)^{-1} \mathbf{G}^T(\lambda \mathbf{I})^T \left(r - y_f\right)$$

O bien, haciendo

$$\mathbf{K}_c = \left(\mathbf{G}^T(\delta \mathbf{I})\mathbf{G} + (\lambda \mathbf{I})\right)^{-1} \mathbf{G}^T(\lambda \mathbf{I})^T \tag{2.19}$$

La ecuación que dicta la variación en la salida del controlador queda dada por

$$\Delta u = \mathbf{K}_c \left(r - y_f\right) \tag{2.20}$$

3. Evaluación de algoritmos en planta con ruido natural

3.1. Introducción

Los algoritmos expuestos en el capítulo Desarrollo de algoritmos para filtro de Kalman en tiempo real son probados en la planta piloto ubicada en el Laboratorio de Control del Departamento de Ingeniería Eléctrica de la Universidad de Concepción. Ambos algoritmos se utilizan para el control de flujo de la planta, mientras que el lazo de nivel queda abierto al operador.

El diagrama P&ID de la planta piloto se muestra en la siguiente imagen,

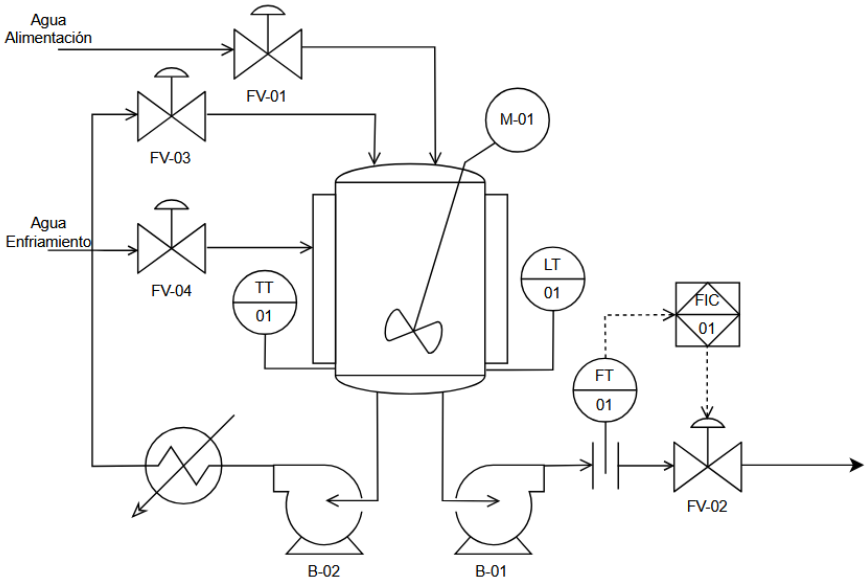

Figura 3.1: Diagrama P&ID de la planta piloto

La relación porcentaje de apertura de la válvula v/s flujo se puede ver en la Figura 3.2. Se puede observar que el flujo en zonas menores al 40% tiene una respuesta anómala al ser comparada con zonas superiores.

El flujo es considerado como un proceso de primer orden, y por consiguiente modelado como tal.

$$H(s) = \frac{k_p}{\tau s + 1} e^{-\theta s} \tag{3.1}$$

Donde k_p corresponde a la ganancia, τ la constante de tiempo, θ el tiempo muerto o retardo del proceso. Este último se aproxima como una función polinomial mediante la aproximación de Padé de primer orden, dada por

$$e^{-\theta s} \approx \frac{2 - \theta s}{2 + \theta s}$$

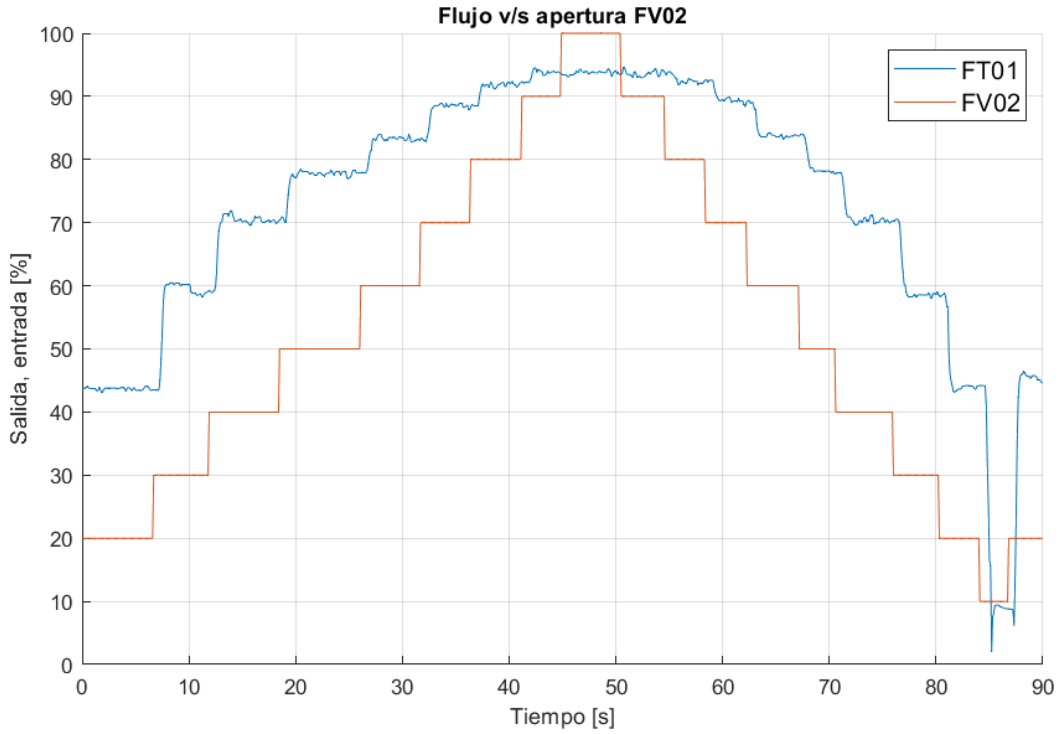

Figura 3.2: Relación apertura de flujo v/s apertura de válvula

Reemplazando esta aproximación en (3.1),

$$H(s) = \frac{y(s)}{u(s)} = \frac{2k_p - k_p\theta s}{\tau\theta s^2 + (\theta + 2\tau)s + 2}$$

Se usa la variable auxiliar $\gamma = 1/(\tau\theta)$ y se aplica la transformada de Laplace inversa, resultando la ecuación diferencial de segundo orden

$$\ddot{y}(t) = \gamma\left[-2y(t) - \dot{y}(t)(\theta + 2\tau) - k_p\theta\dot{u}(t) + 2k_p u(t)\right]$$

Como se desea implementar un algoritmo discreto, puesto que este es implementado en un PLC, se discretiza la ecuación anterior mediante el método de Euler, siendo t_0 el tiempo de muestreo. Así, y haciendo $x = [y \quad \dot{y} \quad k_p \quad \tau \quad \theta \quad \gamma] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6]$, se obtiene el vector discreto

$$x_{k+1} = \begin{bmatrix} x_1 + x_2 t_0 \\ x_2 + x_6 t_0(-2x_1 - x_2 x_5 - 2x_2 x_4 - x_3 x_5 du_k + 2x_3 u_k) \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

Del cual se obtiene su Jacobiano, dado por

$$\mathbf{F_k} = \begin{bmatrix} 1 & t_0 & 0 & 0 & 0 & 0 \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Con

$$f = \begin{bmatrix} f_{21} \\ f_{22} \\ f_{23} \\ f_{24} \\ f_{25} \\ f_{26} \end{bmatrix} = \begin{bmatrix} -2x_6t_0 \\ 1 + x_6t_0(-x_5 - 2x_4) \\ x_6t_0(-x_5du_k + 2u_k) \\ x_6t_0(-2x_2) \\ x_6t_0(-x_2 - x_3du_k) \\ t_0(-2x_1 - x_2x_5 - 2x_2x_4 - x_3x_5du_k + 2x_3u_k) \end{bmatrix}$$

Teniendo x_{k+1} y $\mathbf{F}_k$, se ejecuta el EKF. Los resultados obtenidos son mostrados en secciones siguientes.

Para la simulación del proceso, se discretiza la función de transferencia dada por la ecuación (3.1) mediante el uso de un retentor de orden 0, resultando el modelo ARMA dado por

$$y[k] = \alpha y[k-1] + \beta u[k-p-1] \tag{3.2}$$

Donde $\alpha = e^{-t_0/\tau}$, $\beta = k_p\left(1 - e^{-t_0/\tau}\right)$ y $p = \theta/t_0$ un número entero.

3.2. Evaluación de los algoritmos en Matlab

El código utilizado para la simulación en Matlab se muestra en el Anexo A.

3.2.1. Identificación paramétrica

A continuación, se muestran los resultados de la implementación del EKF para la identificación paramétrica del proceso representado por la ecuación (3.1), notando que al modelo simulado se le añade ruido. Para la estimación, tal como en [8], se utiliza una señal PRBS de magnitud ±5 para la excitación del sistema. Se simula con los parámetros $k_p = 1.2$, $\tau = 2.5$ y $\theta = 0.4$.

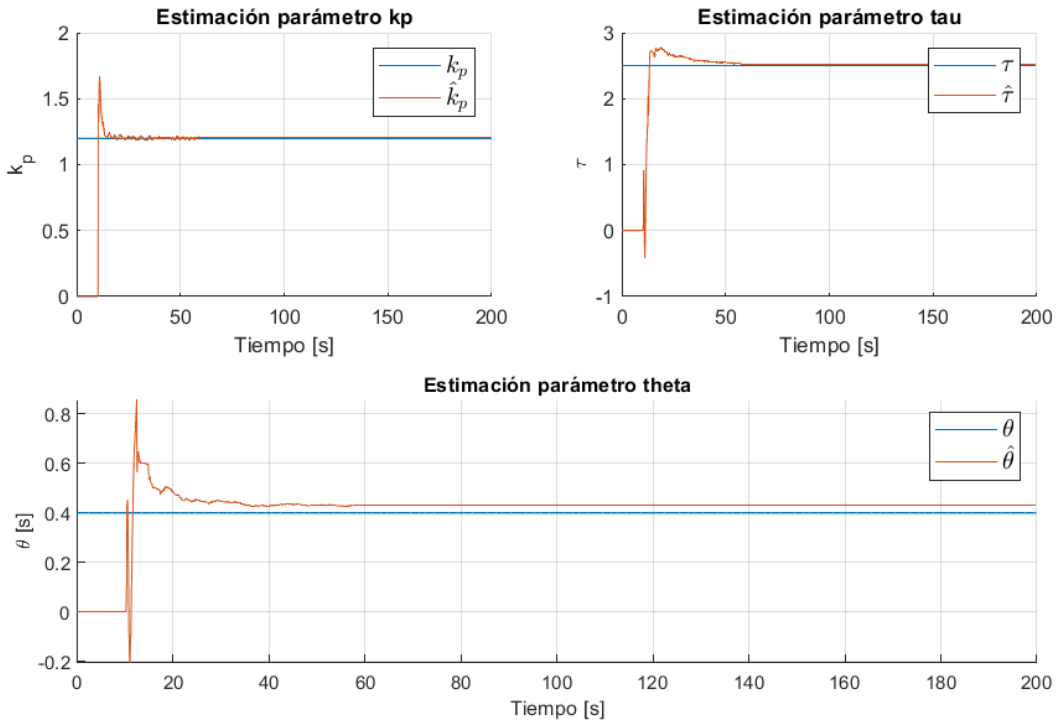

Figura 3.3: Resultados de la identificación paramétrica en simulación

3.2.2. Control DMC

Con los parámetros estimados se ejecuta el controlador, se ajustan los parámetros de sintonización hasta obtener una respuesta deseable, es decir, tiempo de asentamiento acotado y sobre paso menor al 10% del cambio de referencia. En la siguiente figura se muestra en los primeros 60 segundos la identificación de parámetros del sistema, posterior al segundo 70 comienza la acción de control.

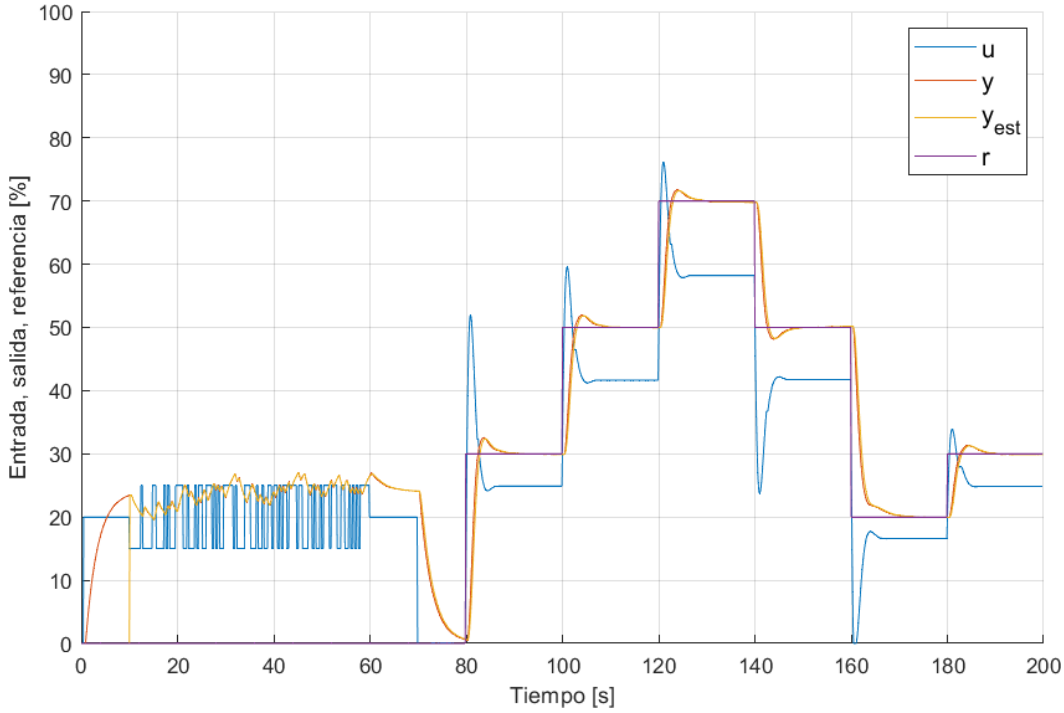

Figura 3.4: Respuesta del sistema con controlador DMC en simulación

3.3. Evaluación de algoritmos en simulación usando Simulink y RSLogix

Luego de comprobar que el algoritmo funciona adecuadamente en Matlab, se realizan las pruebas de atención en fábrica (FAT) mediante la simulación del proceso en Simulink y el uso de un PLC virtual para controlar el proceso. Se diseña una HMI para el proyecto con fines de sintonización.

En la siguiente figura se muestra el esquema Simulink de la planta correspondiente al P&ID de la Figura 2.1, con los lazos de flujo y nivel demarcados dentro del rectángulo en rojo. Se simula para $k_p = 1.26$, $\tau = 2.479$ y $\theta = 0.4$.

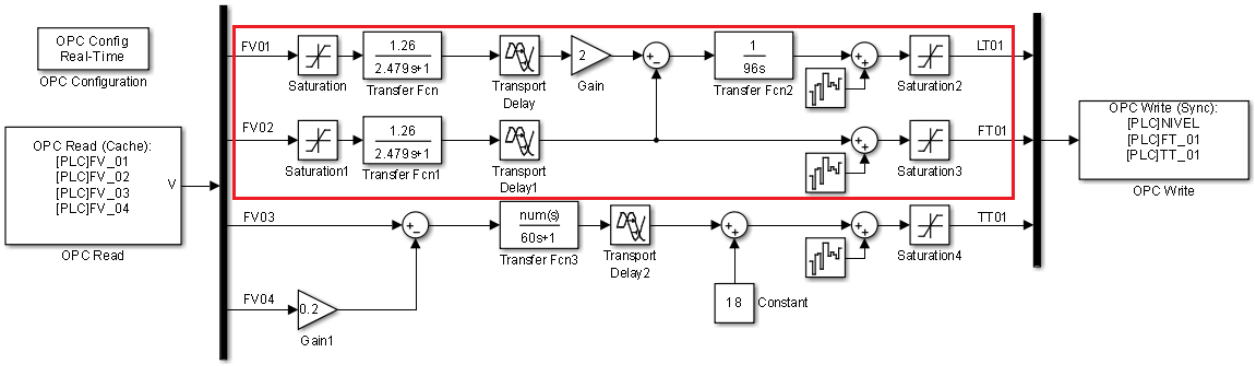

Figura 3.5: Esquema Simulink de la planta piloto

Para llevar a cabo la simulación fue necesario ejecutar dos sesiones de Matlab en paralelo, una encargada del modelo Simulink y otra para el script de los algoritmos y comunicación OPC. El resultado de la identificación paramétrica se muestra en la Figura 3.6. Los parámetros estimados corresponden a $\hat{k}_p = 1.26$, $\hat{\tau} = 2.36$ y $\hat{\theta} = 0.47$, es decir, se tienen RMSE del 0%, 11.9% y 7%, respectivamente. El EKF se ejecuta en segundo plano para detectar cambios en los parámetros.

Como matriz de covarianza de ruido de medición se utiliza el valor medio del ruido, es decir, $\mathbf{R} = 0.01$. Como matriz de covarianza de ruido de proceso se utiliza un valor obtenido empíricamente, $\mathbf{Q} = 5$.

En la Figura 3.7 se muestra el resultado sobre el sistema del controlador sintonizado, donde se hacen cambios de referencia de 10%, 20%, 30% y 60%.

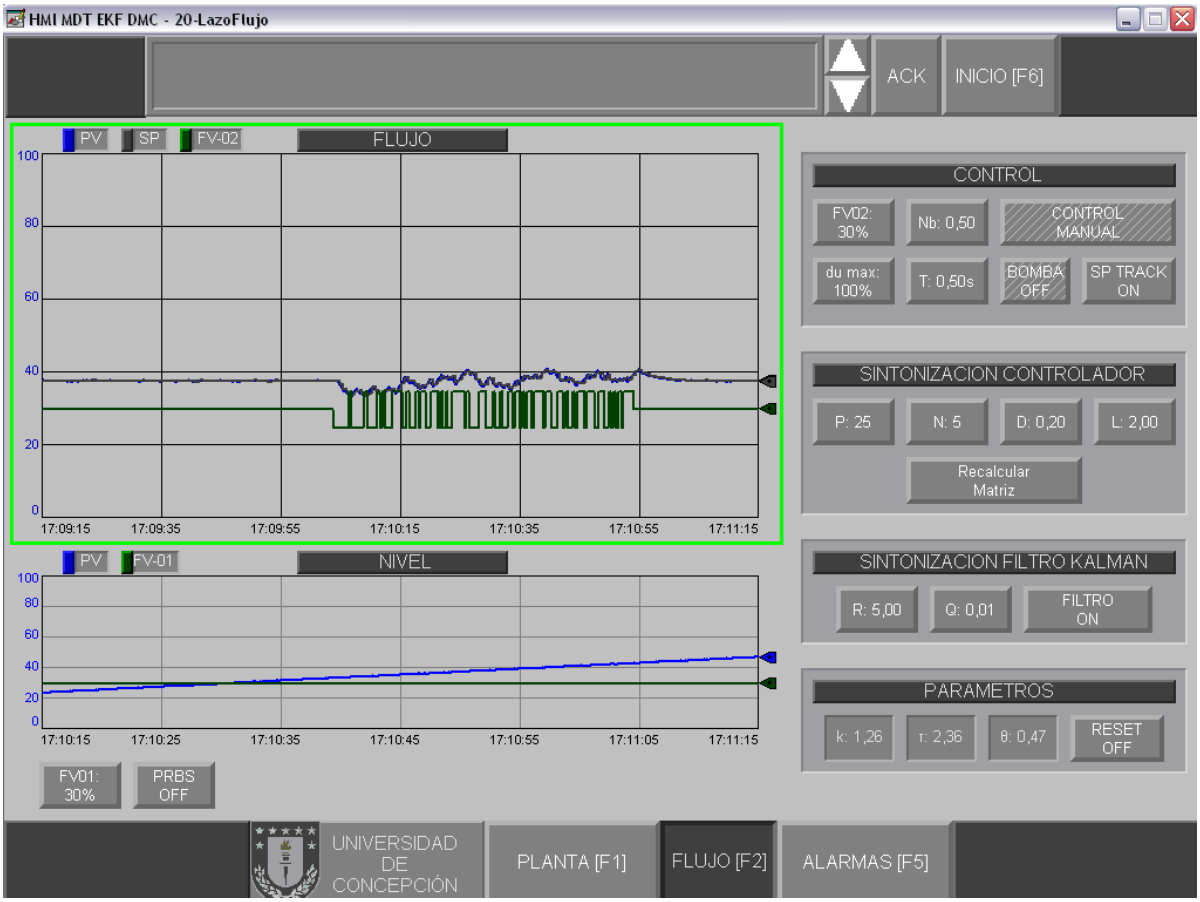

Figura 3.6: HMI, identificación paramétrica en planta virtual

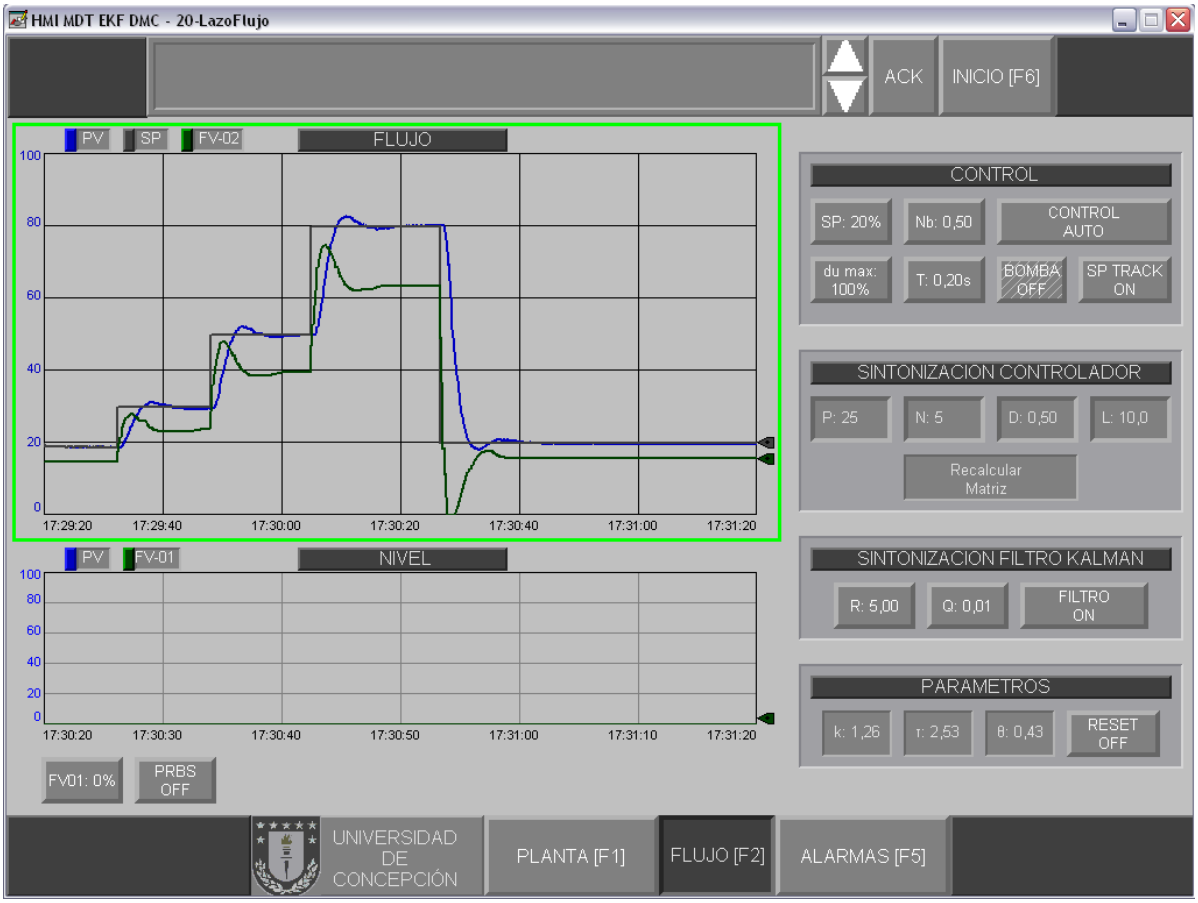

Figura 3.7: HMI, controlador sintonizado para planta virtual

3.4. Evaluación de algoritmos en planta piloto

3.4.1. Identificación paramétrica

Una vez verificado el funcionamiento correcto de los algoritmos en simulación, se procede a evaluar los algoritmos en la estimación de parámetros y control en una planta real. En primer lugar, se hace la identificación paramétrica excitando el sistema con una PRBS, lo cual se muestra en la Figura 3.8. Para obtener el valor a utilizar como matriz de covarianza de ruido de proceso del EKF se toma una muestra de mediciones, se calcula la diferencia entre muestras contiguas y se usa el valor promedio del valor absoluto de dichas diferencias ($\mathbf{Q} = 0.115$). Además, una de las suposiciones del EKF es que el ruido que afecta la señal tenga una distribución normal, o pueda ser aproximada por una. Estos últimos dos puntos se muestran en la Figura 3.9. La matriz de covarianza de ruido de proceso se escoge como un valor pequeño, obtenido empíricamente ($\mathbf{R} = 0.05$), para que las predicciones se ajusten de acuerdo con las mediciones y no sean altamente dependientes del modelo escogido.

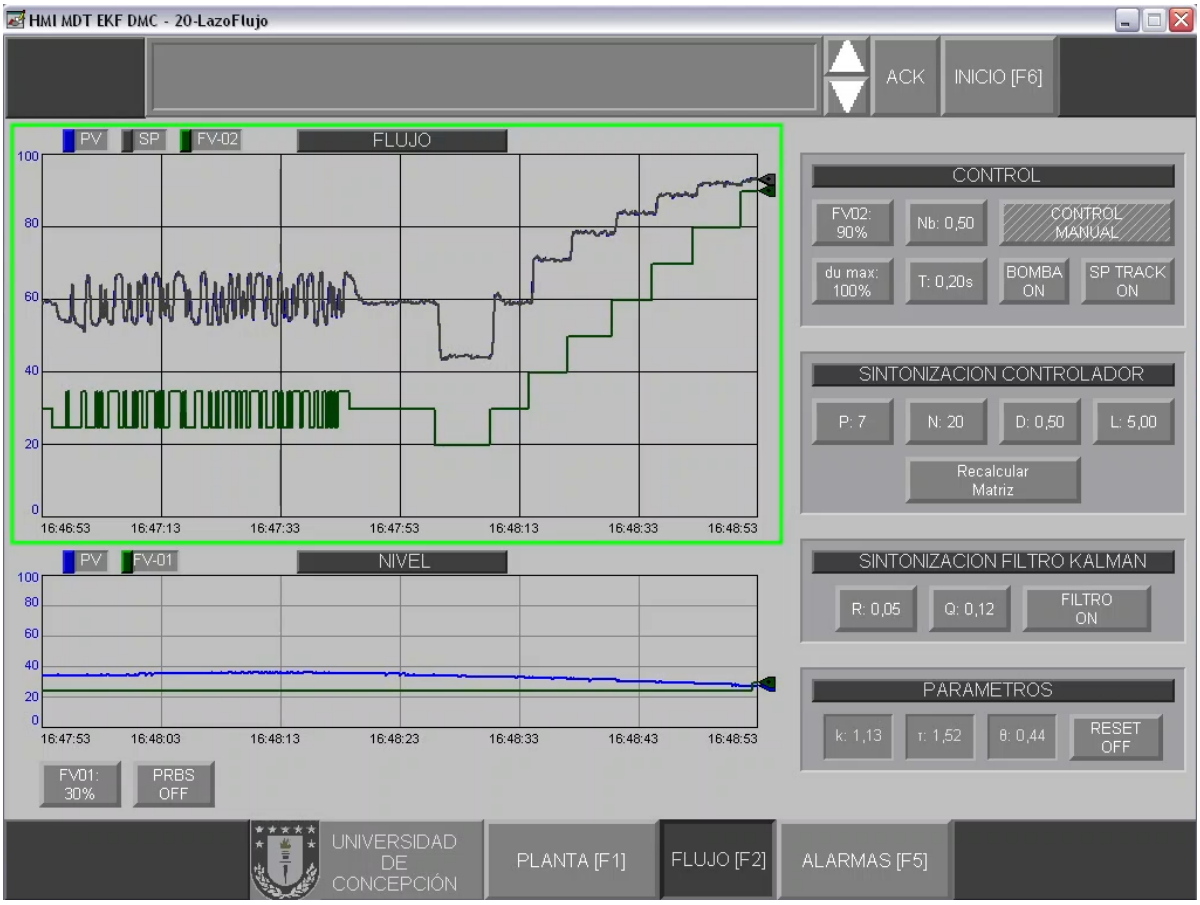

Figura 3.8: HMI, identificación paramétrica en planta piloto

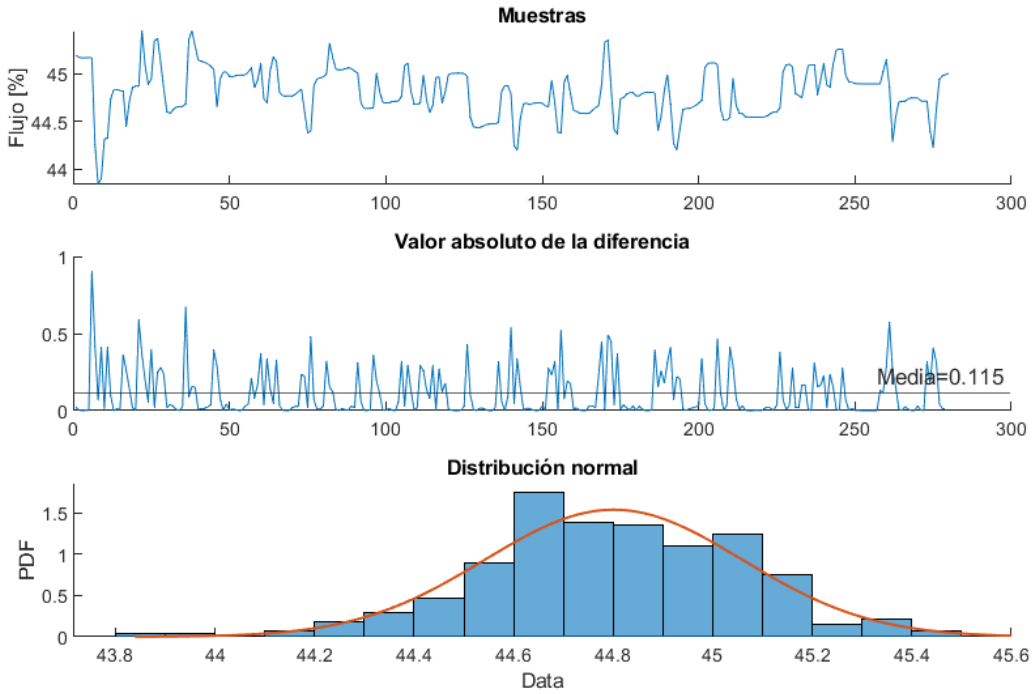

Figura 3.9: Análisis de ruido

En la siguiente figura se muestra la excitación del sistema y la estimación de parámetros, notando que hay un periodo de tiempo en el cual la estimación del EKF queda detenida. Lo anteriormente mencionado se puede observar entre los segundos 50 y 60 del gráfico de flujo.

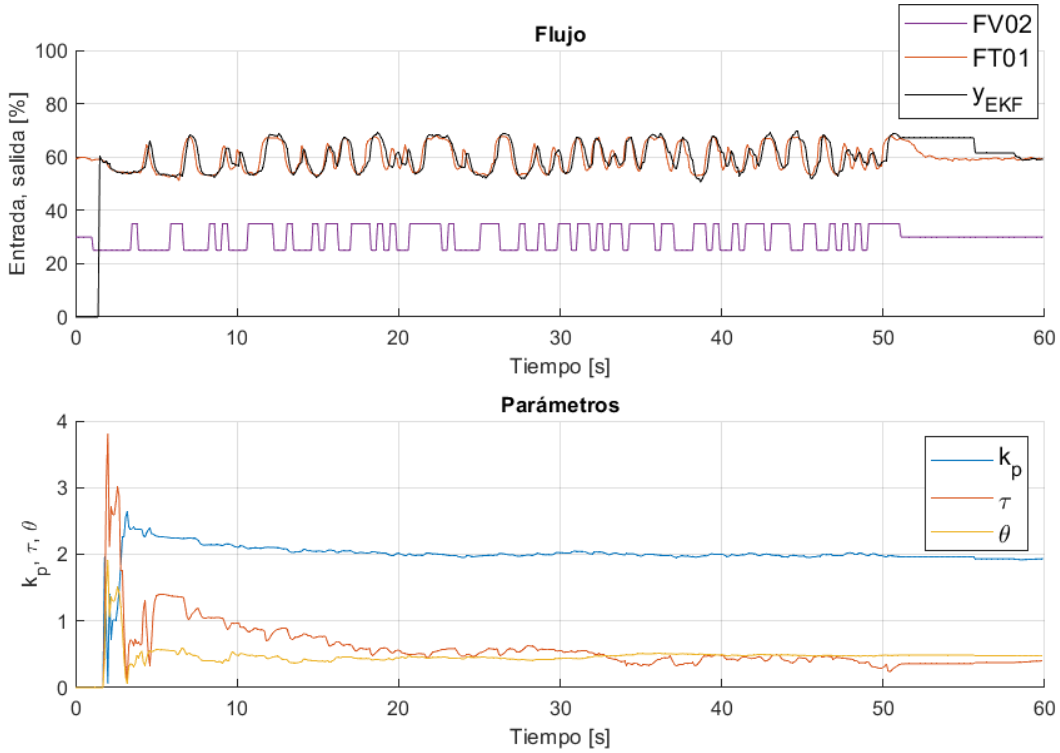

Figura 3.10: Identificación paramétrica para planta piloto

3.4.2. Controlador DMC

Finalmente, y al tener los parámetros estimados del proceso, se aplica el algoritmo del controlador DMC para controlar el flujo. Para la sintonización se escogen valores con el objetivo de obtener respuestas suaves, es decir, se le entrega menor ponderación al error y mayor ponderación a la diferencia entre valores de entrada. Además, como se tiene un proceso altamente no lineal, el horizonte de predicción se acota a un valor pequeño. En la siguiente figura se muestra la acción del controlador para flujo desde el 50%.

Figura 3.11: HMI, controlador sintonizado para planta piloto

4. Interfaz gráfica y conexión OPC

4.1. Interfaz gráfica

La interfaz gráfica es diseñada bajo la norma ISA101, en particular, siguiendo las guías mostradas en [17]. Se realizan tres pantallas: Planta, donde se muestra un esquema general de la planta piloto junto a las variables de proceso; Flujo, donde se encuentran tanto los parámetros de sintonización del DMC como del EKF, además de dos gráficos en los cuales se muestran variables relevantes relacionadas a flujo y nivel; Alarmas, donde se muestran las posibles alarmas relacionadas al proceso. A continuación, se muestran dichas pantallas.

En la Figura 4.1 se muestra la pantalla del esquema general de la planta piloto. En esta se consideran 4 elementos (demarcados por líneas rojas y con su correspondiente número también encerrado en rojo), de los cuales 3 son únicos para esta pantalla. El primero corresponde al diagrama de la planta piloto, donde se observa un gráfico de tendencia de nivel dentro del estanque. El segundo corresponde a un visor de las 3 variables de proceso del sistema junto a su respectiva referencia, donde se distinguen 5 zonas para cada indicador: la zona inferior gris oscuro corresponde a la alarma *Low-Low*, la zona inferior gris claro a la alarma *Low*, la zona azul a la zona de operación normal, la zona superior gris claro a la alarma *High*, la zona superior gris oscuro a la alarma *High-High*. El tercero corresponde a la botonera de bombas *booster* y el motor agitador. El cuarto elemento, y común en todas las pantallas, corresponde al panel de alarmas.

En la Figura 4.2 se muestran 6 elementos. El primer elemento corresponde a los gráficos de tendencias del flujo y nivel. El segundo es una botonera donde se pueden seleccionar parámetros como la banda de ruido (Nb), control automático o manual, la referencia (SP) cuando se encuentra en modo automático el controlador y la apertura de la válvula FV02 cuando se encuentra en modo manual, la variación máxima en la salida (du max), el tiempo de muestreo del controlador (T), la activación o desactivación de la bomba *booster* B01 y el seguimiento de referencia (SP tracking) cuando el controlador está en modo manual. El tercer elemento corresponde a la botonera para cambiar los parámetros de sintonización del DMC, junto con un botón para recalcular el vector dado por la ecuación (2.19). El cuarto elemento es una botonera para la sintonización del EKF, donde se pueden modificar tanto la matriz de covarianza del ruido de medición como la matriz de covarianza de ruido de proceso, y un botón para activar o desactivar el filtro. El quinto elemento es un visor de los parámetros calculados por el EKF, y un botón para reiniciar dichos parámetros. El sexto

elemento es una botonera para accionar la válvula de carga FV01 y activar o desactivar la señal de excitación.

En la Figura 4.3 se muestra la pantalla de alarmas. El único elemento es el panel de alarmas históricas y su botonera correspondiente.

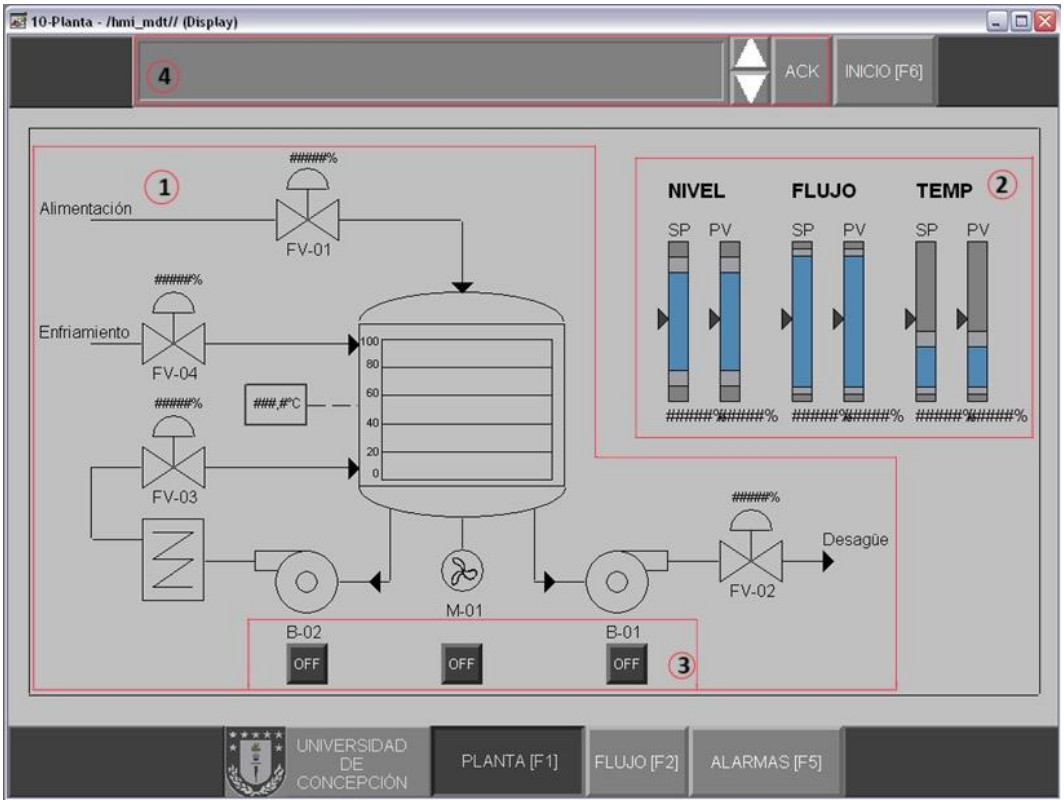

Figura 4.1: HMI, pantalla general de la planta

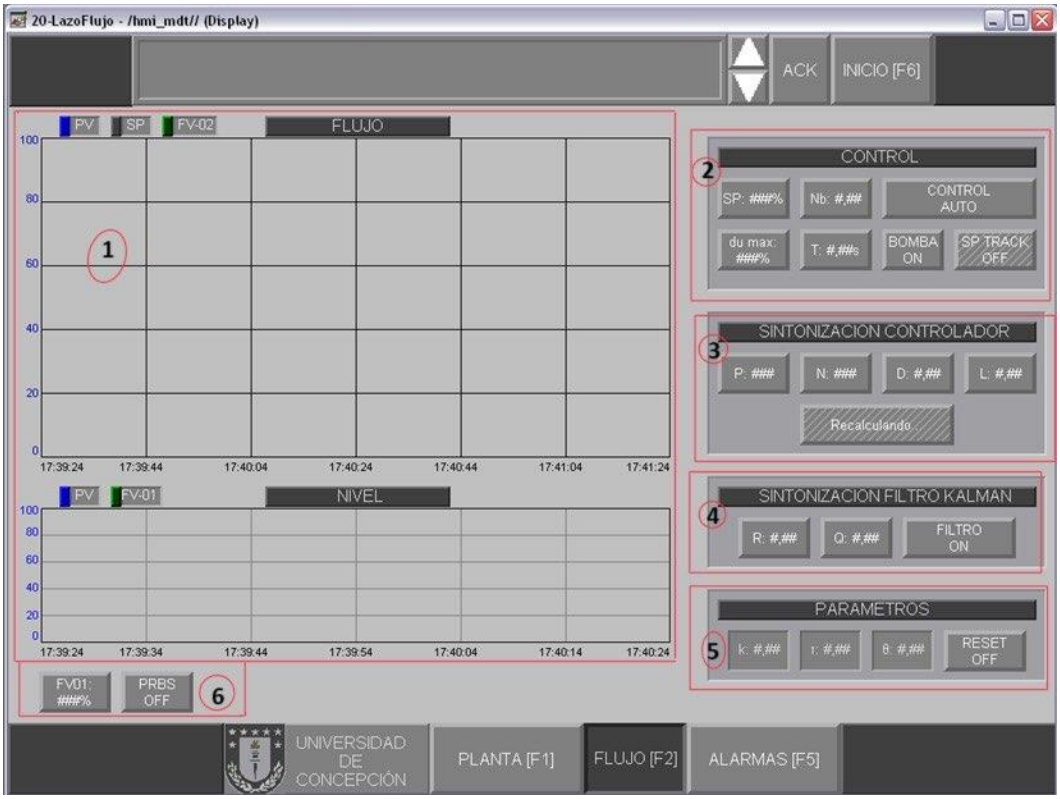

Figura 4.2: HMI, pantalla de variables relevantes, DMC y EKF

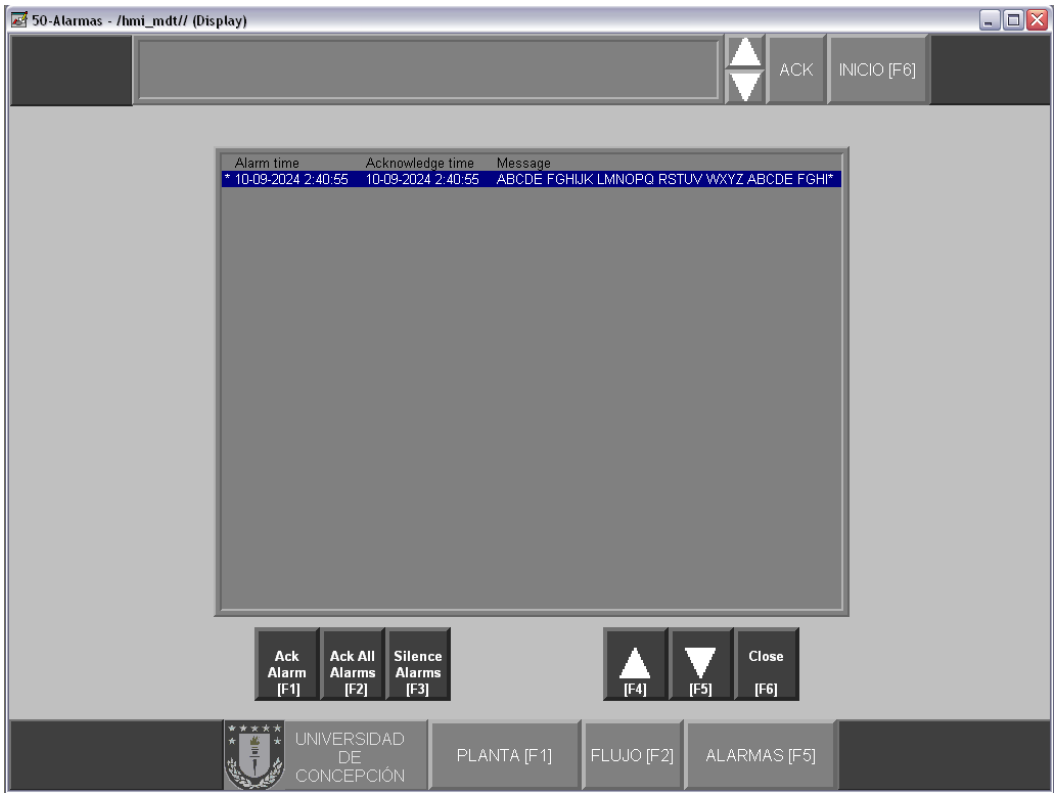

Figura 4.3: HMI, pantalla de alarmas

4.2. Conexión OPC

Un PLC tiene capacidades computacionales limitadas, por lo que el cálculo de la inversa de una matriz toma una cantidad relevante de tiempo en ser ejecutada en este. Por lo anterior, el PLC es destinado únicamente al cálculo de valores que sean estrictamente necesarios para el control en tiempo real. Es decir, tanto el algoritmo del EKF como el cálculo de la ecuación (2.19) son realizados en Matlab y enviados al PLC mediante RSLinx, un servidor OPC. Por lo tanto, en el PLC solo se lleva a cabo el cálculo de la ecuación (2.20) y se hace el control del actuador.

La conexión entre el PLC y Matlab se hace utilizando la librería [18]. Se usan los comandos `opcda`, `connect`, `additem` y `write`. Los primeros dos para establecer la conexión, el tercero para añadir tags a leer al ambiente de variables de Matlab y el último para escribir valores desde Matlab al PLC.

RSLinx se configura con el uso de distintos tópicos. Para las simulaciones se conecta con un chasis virtual que simula un PLC, mientras que en la planta piloto se conecta directamente al PLC mediante ethernet usando la dirección IP de este. A continuación, se muestra una captura de pantalla de este OPC y los distintos tópicos utilizados, siendo 152.74.22.42 la dirección IP del PLC. Algo a notar es que en simulación el controlador se encuentra en el segundo espacio del chasis, mientras que en la planta piloto este está ubicado en el primer espacio.

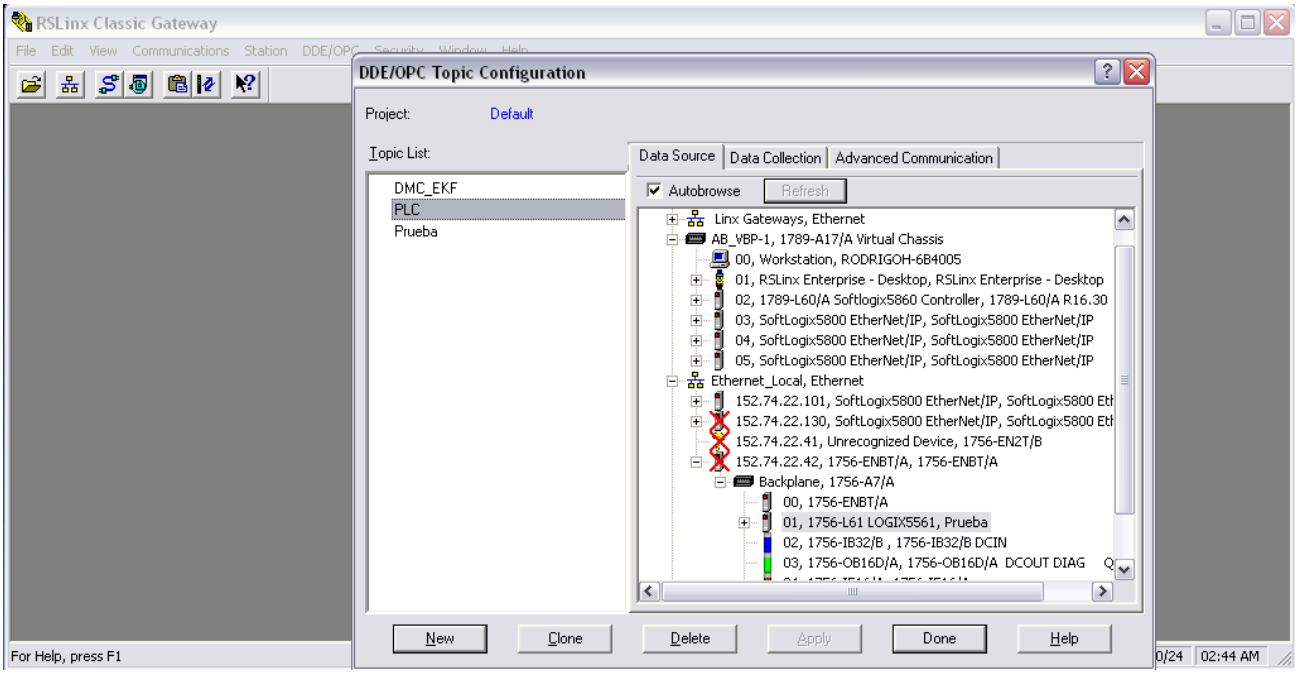

Figura 4.4: RSLinx

En la siguiente figura se muestra un esquema de la estrategia utilizada para llevar a cabo los cálculos computacionales intensivos que son enviados al PLC, el cálculo del valor necesario para el control en tiempo real en el PLC y finalmente la acción del PLC sobre el actuador, que en el caso de la planta piloto resulta ser una válvula de control.

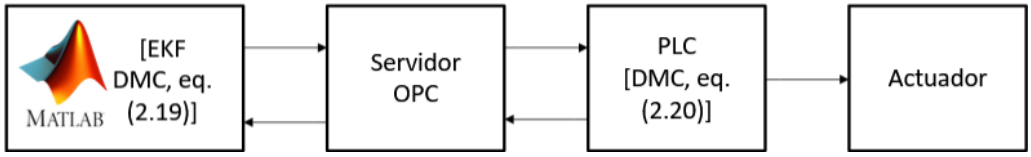

Figura 4.5: Esquema de conexión con Matlab

5. Pruebas comparativas

5.1. Introducción

El controlador elegido tiene cuatro parámetros de sintonización: horizonte de predicción, horizonte de control, ponderación del error y ponderación de la entrada. Cada uno de estos valores tiene un efecto en la acción de regulación.

El primero le permite al controlador predecir valores futuros del proceso de acuerdo con su respuesta escalón. El segundo tiene relación con el número de entradas que se predicen para la regulación del proceso. En cuanto a la implementación, se tiene que mientras más grande sea el valor de estos dos parámetros, se requiere un número mayor iteraciones para calcular la respuesta libre, en particular el horizonte de predicción es el que más añade iteraciones el cálculo. Por lo anterior, en la práctica no es viable utilizar valores muy elevados para el horizonte de predicción, al menos si se necesita un tiempo de muestreo pequeño para el controlador.

Los últimos dos parámetros tienen directa relación con la velocidad que se efectúa el actuador, puesto que le entregan un peso a la variación de la salida. Si la ponderación del error es elevada, la variación del actuador será elevada. Por otro lado, a mayor ponderación de la variación de la salida, esta tiene una respuesta más suave.

5.2. Sintonización con distintos parámetros

A continuación, se muestran distintas configuraciones con distintas respuestas para la salida del controlador, además del valor absoluto del error para cada instante de tiempo. En la siguiente tabla se muestran los valores utilizados para cada configuración. Además, se muestra el rango de operación desde el cual el controlador es estable, la constante de tiempo para la zona de 70% de flujo y el tiempo que le toma llegar al estado estacionario en 70%.

Tabla 5.1: Configuraciones para sintonización

Conf.	t_0	P	N	D	L	Rango Op.	τ_{rise}	SS
1	0.2s	5	15	0.5	5	50%-100%	1.0s	1.5s
2	0.2s	7	2	0.5	10	50%-100%	0.9	4s
3	0.2s	7	10	0.5	10	40%-100%	1.1s	3.7s
4	0.2s	7	20	0.5	10	40%-100%	1.0s	2.4s
5	0.2s	7	20	0.5	5	50%-100%	1.0s	1.8s
6	0.5s	25	10	0.5	5	30%-100%	1.6s	8.1s
7	0.1s	30	20	0.5	10	80%-100%	-	-

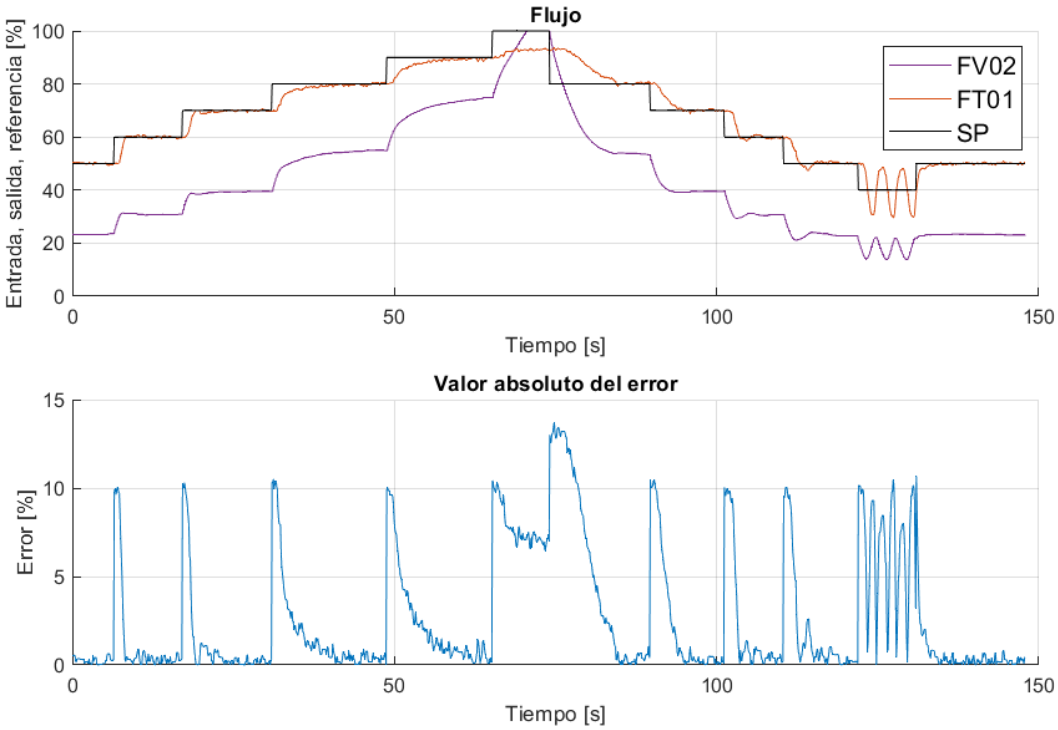

Figura 5.1: Resultado de sintonización, primera configuración

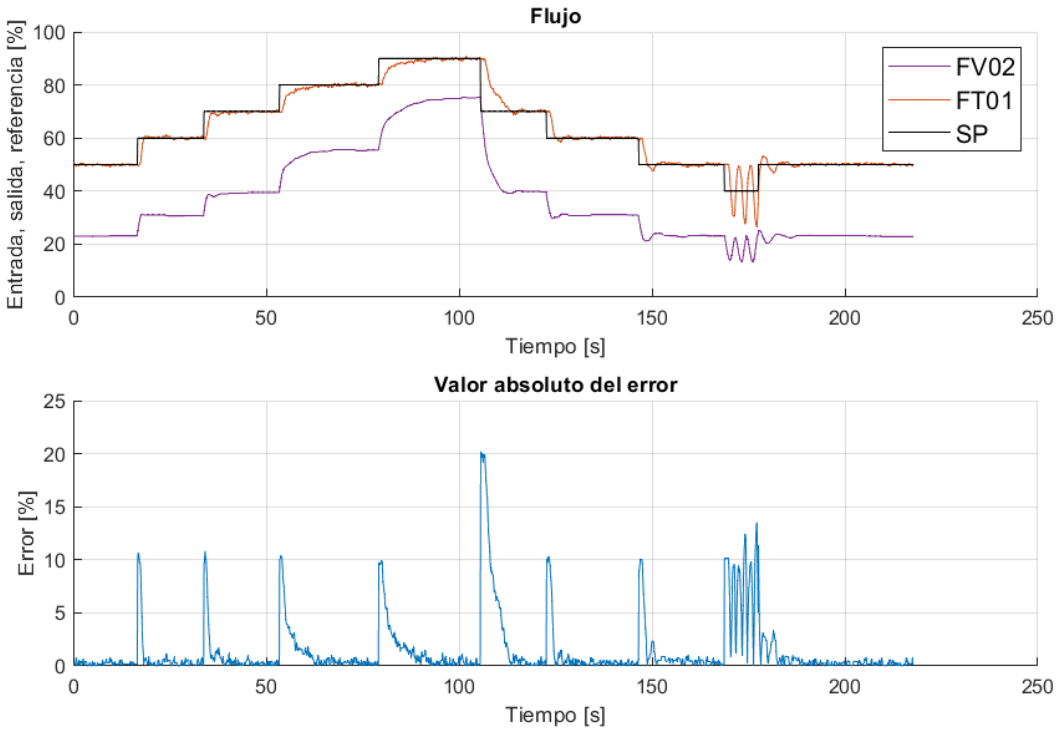

Figura 5.2: Resultado de sintonización, segunda configuración

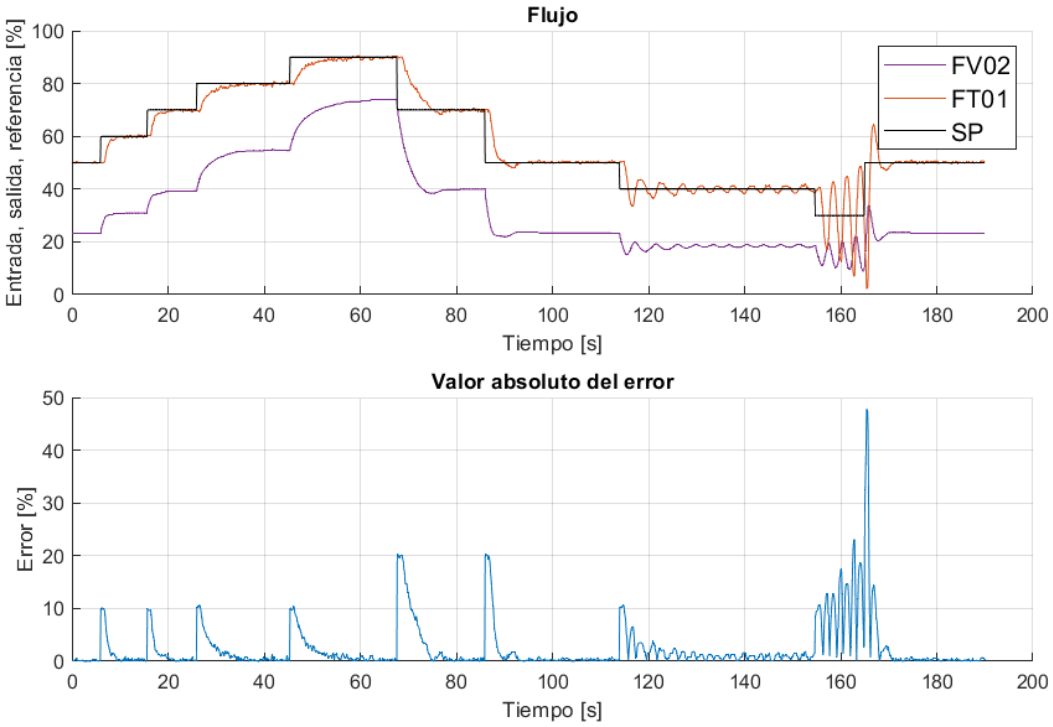

Figura 5.3: Resultado de sintonización, tercera configuración

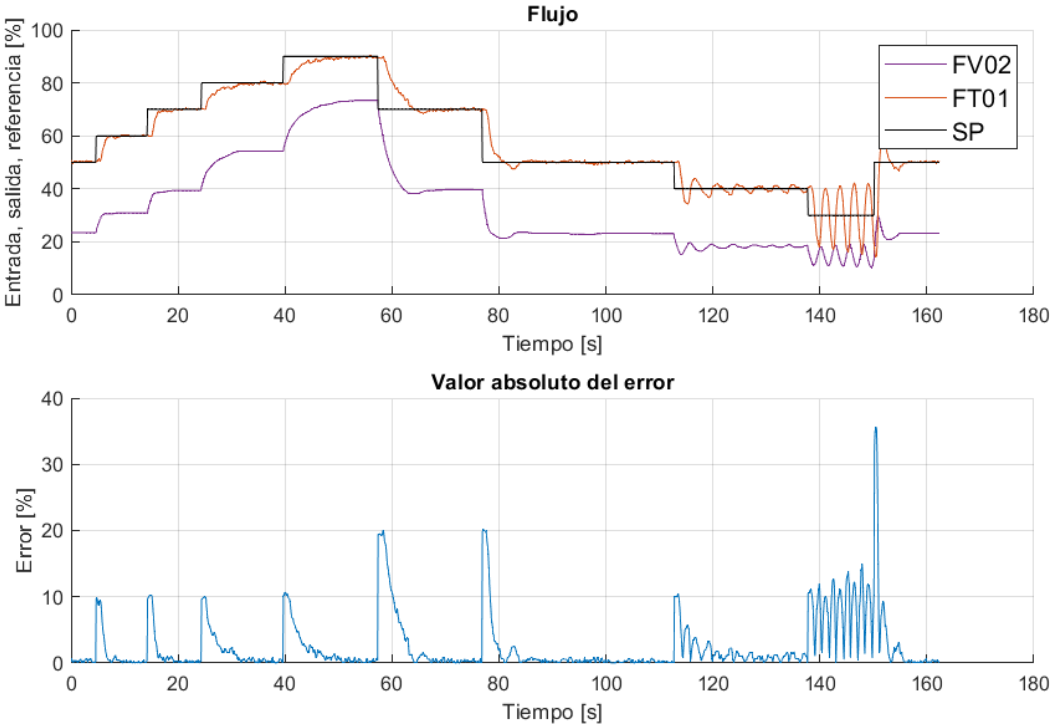

Figura 5.4: Resultado de sintonización, cuarta configuración

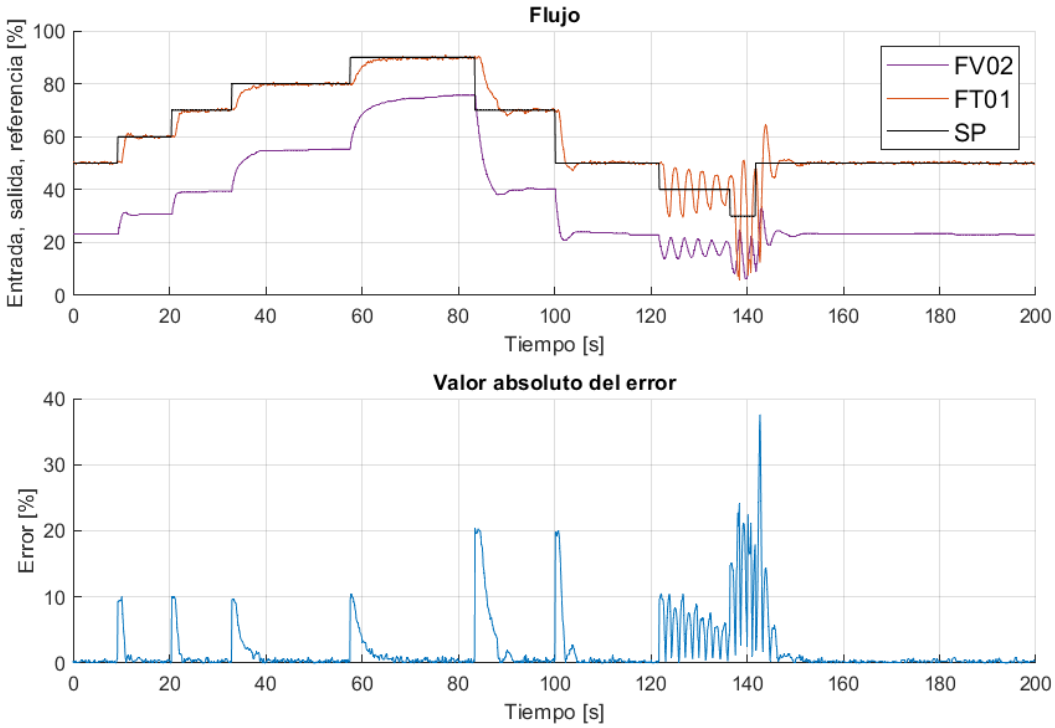

Figura 5.5: Resultado de sintonización, quinta configuración

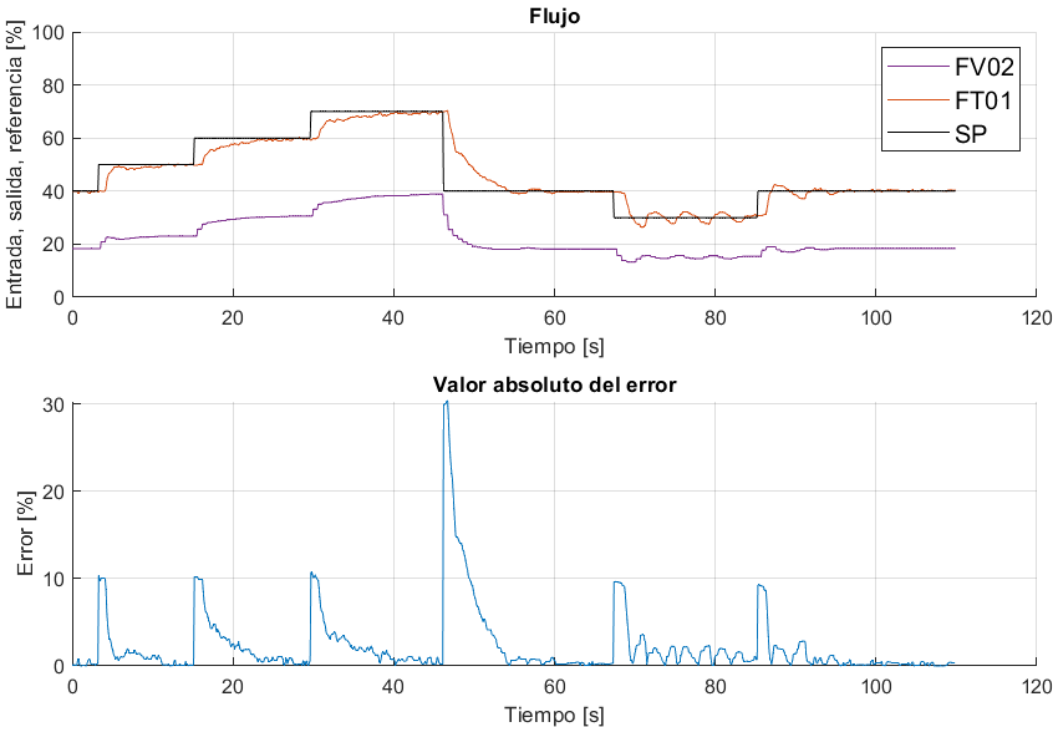

Figura 5.6: Resultado de sintonización, sexta configuración

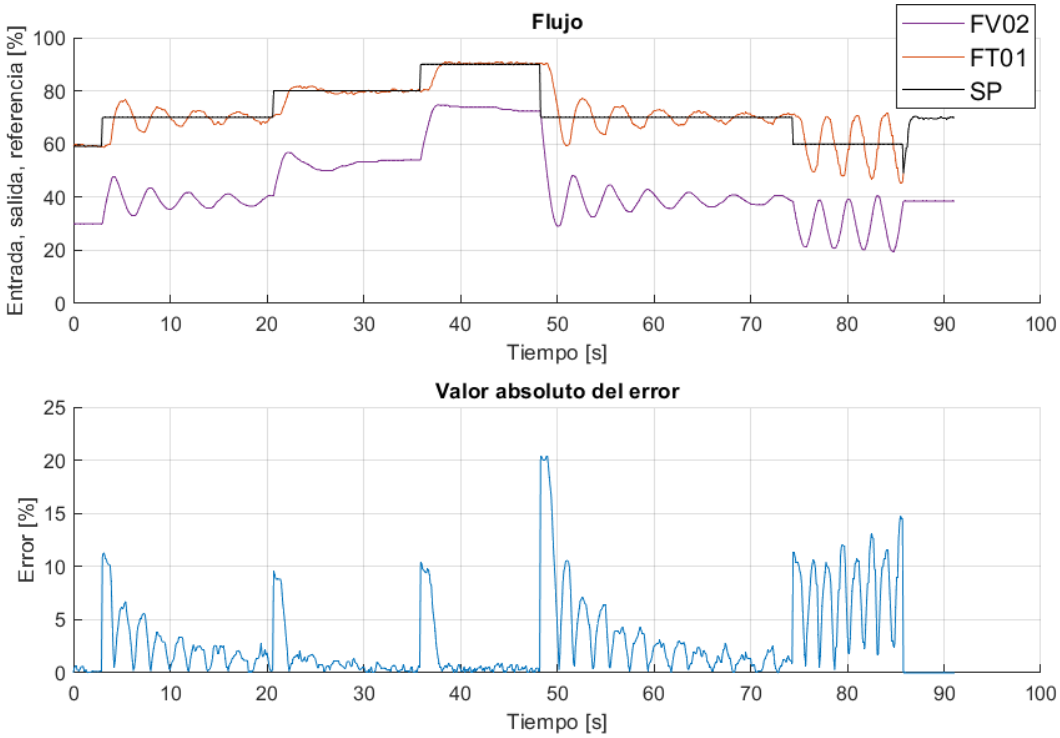

Figura 5.7: Resultado de sintonización, séptima configuración

6. Determinación de métricas de medición de desempeño del filtro

6.1. Introducción

A falta de un valor real de los parámetros estimados, para determinar el desempeño del filtro se utiliza el RMSE de predicción. Además, como se tiene un proceso no lineal, se evalúa el seguimiento de parámetros de acuerdo con la zona de trabajo del flujo.

6.2. RMSE de predicción

En la siguiente figura se muestra el error RMS de predicción del filtro al excitar con una señal PRBS al sistema.

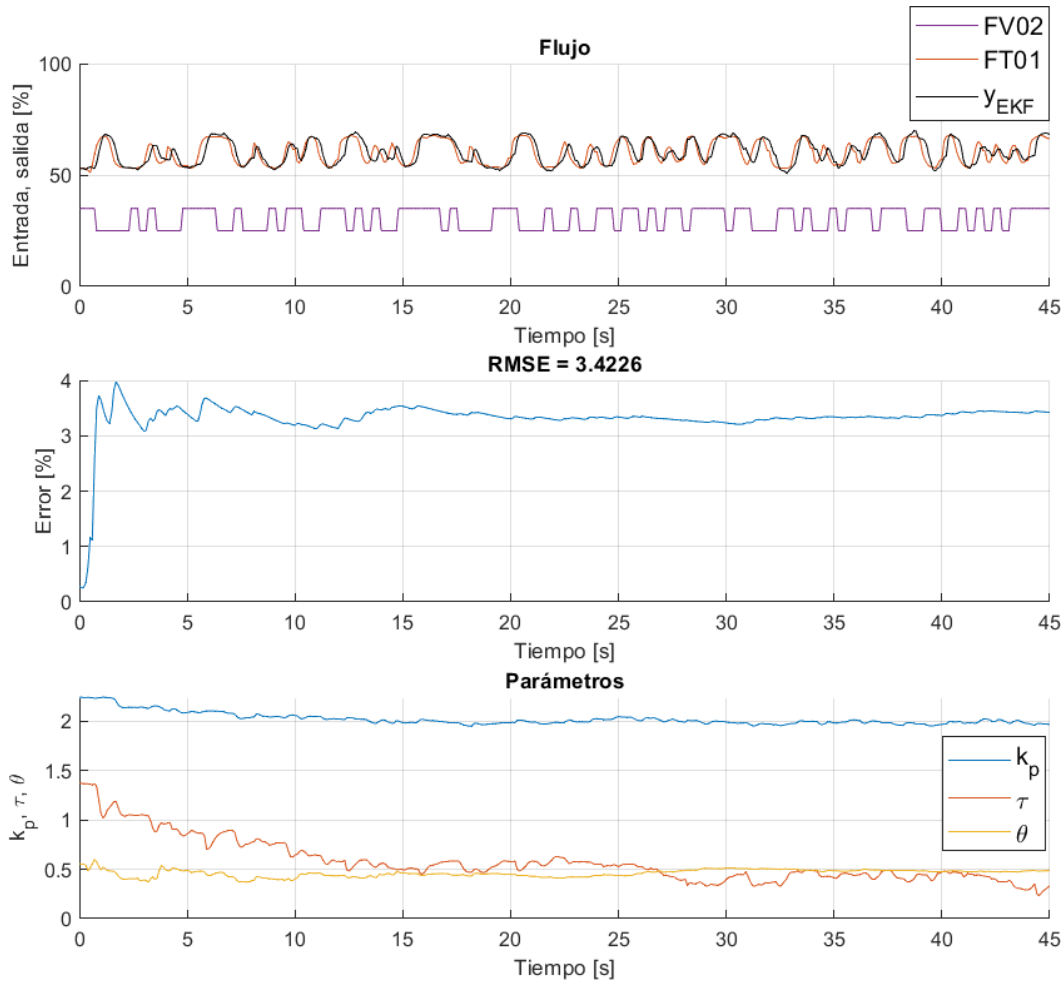

Figura 6.1: RMSE de predicción

6.3. Seguimiento de parámetros

Intuitivamente, observando el gráfico de flujo, se tiene que la ganancia del proceso disminuye mientras que su constante de tiempo aumenta, dado que la respuesta se demora más en llegar al

valor en estado estacionario a medida que aumenta el flujo. Además, el retardo del proceso no debiese ser afectado por la zona en la que se está trabajando. En la siguiente figura se muestran los parámetros estimados y el error RMS del filtro en distintas zonas de flujo.

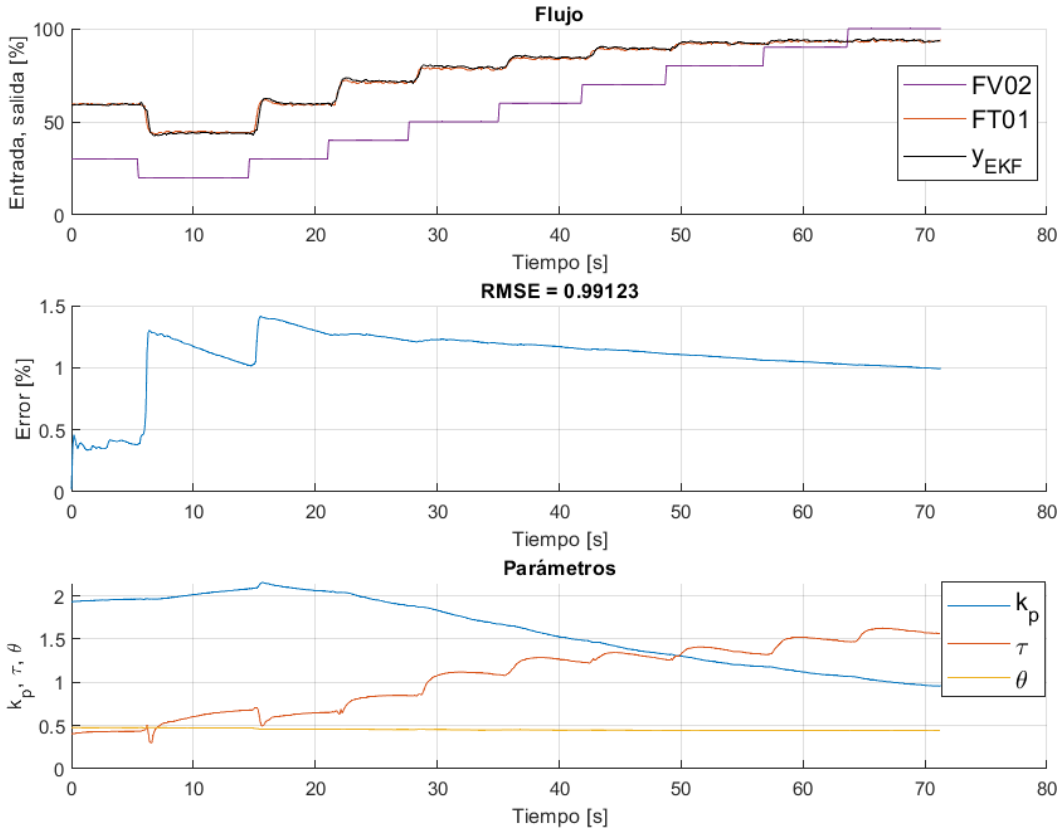

Figura 6.2: Seguimiento de parámetros

7. Análisis de resultados

7.1. Filtro

De acuerdo con lo expuesto en los capítulos previos, el EKF implementado para estimar la salida y parámetros de un sistema de primer orden con retardo resulta tener una buena métrica de desempeño. El RMSE de predicción resulta ser del 3.3% una vez finalizada la excitación del sistema y del 0.99% cuando el algoritmo se aplica continuamente durante cambios en el sistema. Además, se ve en la Figura 6.2 que el EKF es capaz de estimar continuamente los parámetros del sistema, dados distintos niveles de flujo.

En la Figura 3.10 se puede observar entre los segundos 50 y 60 un periodo donde el estimador deja de funcionar. Esto se debe a la escritura de datos secuencial de los valores calculados y enviados por Matlab, Por lo tanto, este controlador, por el número de cálculos que requiere para funcionar, no es factible implementarlo directamente en el PLC.

7.2. Controlador

Al revisar la Tabla 5.1 se puede ver que, de todas las configuraciones probadas en la planta, la primera resulta ser la que mejor rendimiento tiene (segunda menor constante de tiempo y menor tiempo en llegar al SS), considerando que solo se desee controlar flujo a partir del 50%. El efecto del controlador para esta configuración operando en la HMI se puede ver en la Figura 3.11. Para flujos menores al antes mencionado, se tiene que cambios muy pequeños en la salida tienen un impacto muy grande en la variable de interés, por lo cual, si se deseara controlar niveles menores de flujo se puede hacer cambios de parámetros del controlador. Este fenómeno se ve en la Figura 3.2. Se puede ver en las configuraciones 3, 4 y 6 que al modificar algunos parámetros se puede conseguir una respuesta estable, sacrificando velocidad en la respuesta.

8. Conclusiones y resultados

Se logran ejecutar los algoritmos deseados para realizar control en la planta piloto, haciendo una conexión OPC entre Matlab y el PLC. Se calculan en tiempo real los parámetros estimados por el filtro, valores también enviados en tiempo real al PLC. Además, el vector requerido por el controlador DMC también se logra enviar desde Matlab al PLC, pero no en tiempo real debido a la gran cantidad de valores requeridos y dado que este envío se realiza de manera secuencial en un bucle *for*.

Se obtiene que el EKF estima los parámetros del sistema de una manera lo suficientemente precisa dado el bajo RMSE de predicción. Además, el EKF es capaz de seguir los cambios de parámetros del proceso en todo el rango de operación de la válvula. Por lo tanto, el algoritmo resulta ser un buen estimador de parámetros para la aplicación que se le dio. Por otra parte, se menciona en la revisión bibliográfica que anular la matriz de covarianza de los parámetros a estimar produce que el algoritmo converja adecuadamente. Sin embargo, se encuentra que al hacer esto el estimador no es capaz de adaptarse al cambio de parámetros de acuerdo con la zona donde se trabaja. Así, se utilizan valores muy pequeños en lugar de nulos para la identificación paramétrica. Los parámetros escogidos para esta matriz durante la ejecución en tiempo real del EKF se pueden ver en el código anexado en A.2.

El controlador DMC regula el sistema de manera adecuada, produciendo respuestas sin sobre paso y con bajo tiempo de asentamiento. Sin embargo, este es muy dependiente de los parámetros con los cuales se calcula la respuesta escalón, produciendo que al controlar zonas con menor ganancia o mayor constante de tiempo se obtienen peores respuestas, es decir, mayor tiempo de asentamiento. Más aún, regular zonas con una ganancia más grande que con la que se obtiene la respuesta escalón produce que el proceso oscile de manera estable o marginalmente estable, incluso llegando a la inestabilidad. Por lo tanto, si se aplica el controlador en un sistema altamente no lineal este pudiese llegar a ser un regulador de bajo rendimiento. Sin embargo, para la aplicación de control de flujo de la planta piloto, el controlador logra su cometido.

8.1. Trabajos futuros

Como se menciona, el controlador puede no ser un buen regulador para sistemas no lineales, por lo que se plantea como trabajo futuro el desarrollo de una versión adaptativa en tiempo real del controlador. Además, en el presente informe no se realizó un análisis de estabilidad del sistema, por

lo que esto también queda como trabajo futuro. Finalmente, se puede mejorar el algoritmo utilizado para la escritura de datos desde Matlab al PLC, puesto que el filtro deja de funcionar mientras se realiza este proceso dado que el EKF y los cálculos necesarios para el DMC son ejecutados de manera secuencial.

Referencias

[1] B. M. Quine, “A derivative-free implementation of the extended Kalman filter,” *Automatica*, vol. 42, no. 11, pp. 1927–1934, Nov. 2006, doi: 10.1016/j.automatica.2006.06.013.

[2] Q. Li, R. Li, K. Ji and W. Dai, "Kalman Filter and Its Application," *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Tianjin, China, 2015, pp. 74-77, doi: 10.1109/ICINIS.2015.35.

[3] R. Adnan, F. A. Ruslan, A. M. Samad and Z. Md Zain, "Extended Kalman Filter (EKF) prediction of flood water level," *2012 IEEE Control and System Graduate Research Colloquium*, Shah Alam, Malaysia, 2012, pp. 171-174, doi: 10.1109/ICSGRC.2012.6287156.

[4] S. Yang and M. Baum, "Extended Kalman filter for extended object tracking," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017, pp. 4386-4390, doi: 10.1109/ICASSP.2017.7952985.

[5] T. Yoshimura, K. Konishi, and T. Soeda, “A modified extended Kalman filter for linear discrete-time systems with unknown parameters,” *Automatica*, vol. 17, no. 4, pp. 657–660, Jul. 1981, doi: 10.1016/0005-1098(81)90041-8.

[6] X. Yang and O. Marjanovic, “LQG Control with Extended Kalman Filter for Power Systems with Unknown Time-Delays,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 3708–3713, Jan. 2011, doi: 10.3182/20110828-6-it-1002.03082.

[7] A. Probst, O. Sawodny, and M.E. Magaña, “Using a Kalman filter and a Padé approximation to estimate random time delays in a networked feedback control system,” *IET Control Theory and Applications*, vol. 4, no. 11, pp. 2263–2272, Nov. 2010, doi: 10.1049/iet-cta.2009.0339.

[8] Ch. Venkateswarlu and Rama Rao Karri, *Optimal State Estimation for Process Monitoring, Fault Diagnosis and Control*. Elsevier, 2022, sec. 4.4, pp. 65–73.

[9] C. R. Cutler and B. L. Ramaker, “Dynamic matrix control - A computer control algorithm,” vol. 17, no. 17, p. 72, Jan. 1980, doi: 10.1109/jacc.1980.4232009.

[10] Z. Jun, M. Qingjin and Y. Hongliang, "The application of dynamic matrix control in the grate cooler," *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, 2017, pp. 2630-2633, doi:

10.1109/IAEAC.2017.8054501.

[11] Y. Hao and H. Yan, "Application of Dynamic Matrix Control in Liquid Level Control," *2020 IEEE 4th International Conference on Frontiers of Sensors Technologies (ICFST)*, Shanghai, China, 2020, pp. 59-62, doi: 10.1109/ICFST51577.2020.9294771.

[12] V. L. Chuong and T. N. Luan Vu, "Identification and Dynamic Matrix Control algorithm for a heating process," *2017 International Conference on System Science and Engineering (ICSSE)*, Ho Chi Minh City, Vietnam, 2017, pp. 642-645, doi: 10.1109/ICSSE.2017.8030954.

[13] A. Zheng, "Does nonlinear dynamic matrix control provide integral control?," *Computers & Chemical Engineering*, vol. 23, no. 11–12, pp. 1753–1756, Jan. 2000, doi: 10.1016/s0098-1354(99)00323-3.

[14] W. Klopot, T. Klopot, and M. Metzger, "Adaptive and Non-Adaptive Dynamic Matrix Control for Conical Tank," *IFAC Proceedings Volumes*, vol. 42, no. 13, pp. 543–548, Jan. 2009, doi: 10.3182/20090819-3-PL-3002.00094.

[15] P. Lundström, J. H. Lee, M. Morari, and S. Skogestad, "Limitations of dynamic matrix control," *Computers & Chemical Engineering*, vol. 19, no. 4, pp. 409–421, Apr. 1995, doi: 10.1016/0098-1354(94)00063-t.

[16] J. A. Rossiter, *A First Course in Predictive Control*. CRC Press, 2018, ch. 4, pp. 117–147.

[17] "Rockwell Automation Process HMI Style Guide White Paper." Available: https://literature.rockwellautomation.com/idc/groups/literature/documents/wp/proces-wp023_-en-p.pdf

[18] "(To be removed) Create OPC data access object - MATLAB opcda - MathWorks América Latina," *Mathworks.com*, 2024. https://la.mathworks.com/help/icomm/ug/opcda.html?searchHighlight=opcda&s_tid=srchtitle_support_results_1_opcda (accessed Aug. 15, 2024).

Anexo A. Códigos Matlab Filtro de Kalman Extendido

A.1. Simulación Matlab

```
clear all, close all, clc
%parametros
t0      = 0.1;              %tiempo de muestreo
kp           = 1.2;                 %ganacia del proceso
tau       = 2.5;              %constante de tiempo inicial
alpha   = exp(-t0/tau);          %A(z-1)
beta   = kp*(1-alpha);           %B(z-1)

tend    = 200;              %tiempo simulacion
N_EKF   = tend/t0;            %numero de puntos
theta   = 4;                %retardo en muestras (theta/T)
t       = 0:t0:tend-1*t0;
Nb = 0.5;
rd = 0.01;

sw_ctr_est = 0;
% inicializa prbs
for  i=1:7
      sh(i)=1.0;          % inicializa palabra prbs
end
delta_prbs = 4;
umax_prbs = 5;                  % Salida máxima para prbs
umin_prbs = -5;                 % Salida mínima para prbs
act_prbs= 0;        % carga periodo de actualización de prbs

%
sw_prbs = 10/t0;
len_prbs = 50/t0;
cnt_prbs = 0;
sw_EKF = 0;

%inicializacion EKF
%Filtro de Kalman
nx = 2;             %numero variables de estado
nu = 1;             %numero entradas
ny = 1;             %numero salidas
np = 4;             %numero parametros a estimar
nt = nx+np;
% xhat(1) = SP;
P = 1000000*eye(nt,nt);
H = zeros(1,nt);
H(1,1) = 1;
%  ------------ parametros sitonizacion ekf ------------  %
R = 0.5*eye(ny);    %ruido medicion
Q = 0.01*eye(nt);   %ruido proceso
Q(3,3) = 0.001;     %ruido kp
Q(4,4) = 0.001;      %ruido tau
Q(5,5) = 0.0;    %ruido theta
Q(6,6) = 0.0;   %ruido 1/(theta*tau)
Kgain = zeros(nt,1);
%  ------------------------------------------------  %
```

```
%  ------------------------------------------------------  %
%  ------------ parametros sitonizacion dmc ------------  %
% Define los ajustes del Control Preditivo
Pc=40;            % Horizonte de prediccion
Nc=10;             % Horizonte de control
lc=7;             % Ponderación actuador
dc=1*eye(Pc);     % Matriz de ponderacion del error
tDMC = 0.1;
n_coef = 25;
%  ------------------------------------------------------  %
%inicializacion variables
u = zeros(N_EKF,nu);    %inicializa entradas
y = zeros(N_EKF,ny);    %iniciliaza salidas
y_est = zeros(N_EKF,1);
xhat = zeros(nt,1);
wmem_Sp = zeros(N_EKF,1);
manaut = sw_prbs+len_prbs+100;
do = zeros(1,N_EKF);
do(manaut+100/t0:N_EKF) = +1; %perturbación
uk1 = 0;
cnt = 0;
SP= zeros(1,N_EKF);
SP(manaut+10/t0:manaut+30/t0) = 30;
SP(manaut+30/t0+1:manaut+50/t0) = 50;
SP(manaut+50/t0+1:manaut+70/t0) = 70;
SP(manaut+70/t0+1:manaut+90/t0) = 50;
SP(manaut+90/t0+1:manaut+110/t0) = 20;
SP(manaut+110/t0+1:end) = 30;
for k = 2+theta:N_EKF
    %proceso
    y(k) = alpha*y(k-1)+beta*u(k-theta-1)+rd*randn;
    %limitador
    if y(k) < 0, y(k) = 0; end

    ek_EKF      = y(k)-xhat(1);           %error
    y_est(k)   = xhat(1)+Kgain(1)*ek_EKF;
    xhat(1) = y_est(k);
    %entrada
    if k < manaut && ~sw_EKF
        u(k) = 20;
    end
    %PRBS
    if k > sw_prbs && k <= len_prbs+sw_prbs
        cnt_prbs = cnt_prbs + 1;
        if cnt_prbs == 1
            SP_aux = u(k);
            sw_EKF = 1;
        end
           act_prbs= act_prbs-1;   % decrementar contador de periodo
        % Genera PRBS
           if act_prbs <=0
                  act_prbs =delta_prbs; % carga valor de periodo de prbs
                  s=sh(7)+sh(6);
                  if s >=1.5 s=0; end
                  for j=2:7
                        jm=8-j;
```

```
                    sh(jm+1)=sh(jm);
                end
                sh(1)=s;
                u(k)=s*(umax_prbs-umin_prbs)+umin_prbs+20; % genera salida prbs
        else
            u(k)=u(k-1);
        end
        if k == sw_prbs+len_prbs
            u(k) = SP_aux;
            sw_EKF = 0;
        end
    end
    %controlador
    if k > manaut
        cnt = cnt+1;
        if cnt == 1
            % Modelo  de  Sistema
            alpha_est   = exp(-t0/tau_est);          %A(z-1)
            beta_est      = kp_est*(1-alpha_est);          %B(z-1)
            num=kp_est;                                % Numerador Continuo
            den=[tau_est 1];                     % Denominador Continuo
            ret=round(theta_est/t0)*tDMC;                                % Retardo del
sistema
            gp=tf(num,den);                      % Funcion de transferencia
            gp.outputdelay=ret;
            ftz=c2d(gp,tDMC);                       %Planta Discreta
            [B,A]=tfdata(ftz,'v');               %Divide Numerador en B y denominador en
A
            d=round(ret/tDMC);                       % atraso de tiempo discreto
            % Vector de coeficientes Gi
            gi=step(ftz,n_coef);       % Guardo la respuesta al escalón
            Nm=length(gi)-1;    % Valor donde trunco la sumatoria para la predicción del
modelo completo "M"
            duf=zeros(1,Nm); % (Delta U Free)
            % Calculo de la Matriz G
            G=zeros(Pc,Nc);
            G(:,1)=gi(1+d:Pc+d);
            for i=2:Nc
                for j=2:Pc
                    G(j,i)=G(j-1,i-1);
                end
            end
            % calcula matriz Mn Matriz
            Mn=inv(G'*dc*G + lc*eye(Nc))*G'*dc;
            %Calculo de controldor K1 (Primera fila de Mn)
            K1=Mn(1,:);
        end
        % CALCULO DE LA RESPUESTA LIBRE
        f=zeros(1,Pc); % Vetor f (free) Respuesta Libre
        for kk=1:Pc
        % monta un vector con las gkk+i - gkk
             for i=1:Nm-Pc
                 vect_g(i)=gi(kk+i)-gi(i);
             end
             for i=Nm-Pc+1:Nm
                 vect_g(i)=gi(Nm)-gi(i);
```

```
        end
        f(kk)=y(k)+vect_g*duf';  %Calculo de respuesta libre
    end
    %Calculo de la ley de control
    ee = abs(SP(k)-y(k));
    du = K1*(SP(k)-f');
    if ee < Nb
        u(k) = u(k-1);
    else
        u(k)=u(k-1)+du;
    end
    if u(k) < 0
        u(k) = 0;
    end
    if u(k) > 100
        u(k) = 100;
    end
    %actualiza duf
    wmem_du(k) = du;
    aux_2=duf(1:Nm-1);
    duf=[du aux_2];
end
%simulacion proceso
uk = u(k);
yk = y(k);
duk = (uk-uk1)/t0;          %derivada del error
%EKF
if sw_EKF
    Pk = P;
    %etapa de prediccion
    %discretizacion sistema primer orden con retardo
    dxhatk1_1 = xhat(2)*t0;
    dxhatk1_2 = (-2*xhat(1)-xhat(2)*xhat(5)-2*xhat(2)*xhat(4)-...
        xhat(3)*xhat(5)*duk+2*xhat(3)*uk)*xhat(6)*t0;
    xhatk1 = xhat + [dxhatk1_1;
                     dxhatk1_2;
                     0;
                     0;
                     0;
                     0];
    %A linealizada
    F21 = -2*xhat(6)*t0;
    F22 = 1-(2*xhat(4)+xhat(5))*xhat(6)*t0;
    F23 = (-xhat(5)*duk+2*uk)*xhat(6)*t0;
    F24 = -2*xhat(2)*xhat(6)*t0;
    F25 = (-xhat(2)-xhat(3)*duk)*xhat(6)*t0;
    F26 = (-2*xhat(1)-xhat(2)*xhat(5)-2*xhat(2)*xhat(4)-...
        xhat(3)*xhat(5)*duk+2*xhat(3)*uk)*t0;
    F = [1, t0, 0, 0, 0, 0;
         F21, F22, F23, F24, F25, F26;
         0, 0, 1, 0, 0, 0;
         0, 0, 0, 1, 0, 0;
         0, 0, 0, 0, 1, 0;
         0, 0, 0, 0, 0, 1];
    Pk1_est = F*Pk*F' + Q;
    %etapa de actualizacion de medicion
```

```
        Sk_EKF = H*Pk1_est*H'+ R;
        Kgain = Pk1_est*H'*1/Sk_EKF;

        %asignacion de resultados
        xhat = xhatk1+Kgain*ek_EKF;
        kp_est = xhat(3);
        tau_est = xhat(4);
        theta_est = xhat(5);
        phi_est = xhat(6);
        P = (eye(nt)-Kgain*H)*Pk1_est;
    end

    %memoria variables
    uk1         = uk;
    %memoria grafica
    wmem_x1(k)  = xhat(1);
    wmem_x2(k)  = xhat(2);
    wmem_kp(k)  = xhat(3);
    wmem_tau(k) = xhat(4);
    wmem_theta(k) = xhat(5)/t0;
    wmem_x6(k)  = xhat(6)*t0;
    wmem_trace(k) = trace(P);
end
kp_rmse = rmse(wmem_kp(end),kp)
tau_rmse = rmse(wmem_tau(end),tau)
theta_rmse = rmse(wmem_theta(end),theta)
phi_rmse = rmse(wmem_x6,1/(theta*tau))
sprintf('kp= %d\ntau= %d\ntheta= %d', kp_est,tau_est,theta_est)
%---------------------------------- PLOTS EKF ----------------------------------%
figure();
subplot(2,2,1)
title('Estimación parámetro kp');hold on;grid on;
plot(t,ones(1,N_EKF)*(kp));
plot(t,wmem_kp);
legend('$k_p$','$\hat{k}_p$','Interpreter', 'Latex', 'FontSize', 12);
ylabel('k_p',"Rotation",0)
xlabel("Tiempo [s]")

subplot(2,2,2);
title('Estimación parámetro tau');hold on;grid on;
plot(t,ones(1,N_EKF)*tau);
plot(t,wmem_tau);
legend('$\tau$','$\hat{\tau}$','Interpreter', 'Latex', 'FontSize', 12);
ylabel('\tau',"Rotation",0);
xlabel("Tiempo [s]")

subplot(2,2,[3,4])
title('Estimación parámetro theta');hold on;grid on;
plot(t,ones(1,N_EKF)*theta);
plot(t,wmem_theta);
legend('$\theta$','$\hat{\theta}$','Interpreter', 'Latex', 'FontSize', 12);
ylabel('t_0',"Rotation",0)
xlabel("Tiempo [s]")
set(gcf, 'Position',  [100, 100, 800, 500])
%---------------------------------- PLOTS EKF ----------------------------------%
figure();hold on; ylim([0 100]); grid on;
```

```
plot(t,u)
plot(t,y)
plot(t,y_est)
plot(t,SP);
legend({'u','y','y_{est}','r'}, 'FontSize', 12)
ylabel('Entrada, salida, referencia [%]')
xlabel("Tiempo [s]")
set(gcf, 'Position',  [100, 100, 800, 500])
```

A.2. Código de conexión OPC y cálculo de variables relevantes

```
%inicializacion EKF
nx = 2;              %numero variables de estado
nu = 1;              %numero entradas
ny = 1;              %numero salidas
np = 4;              %numero parametros a estimar
nt = nx+np;
H = zeros(1,nt);
H(1) = 1;

%inicializa variables en 0
P = 1000*eye(nt,nt);
Kgain = zeros(nt,1);
xhat = zeros(nt,1);
uk1 = 0;
F21 = 0;
F22 = 0;
F23 = 0;
F24 = 0;
F25 = 0;
F26 = 0;
F = zeros(nt,nt);

%Conexion PLC
t0 =0.1;
da = opcda('localhost','RSLinx Remote OPC Server');
connect(da);
grp = addgroup(da);
grp.UpdateRate = t0;
%ns = getnamespace(da);
%conName = strcat('[',ns(1).Name,']');
conName = '[PLC]';

%Lectura variables desde el PLC
PLC_sw_EKF = additem(grp, strcat(conName,'sw_EKF'));
PLC_FV_02 = additem(grp, strcat(conName,'FV_02'));
PLC_FT_01 = additem(grp, strcat(conName,'FT_01'));
PLC_sw_RF = additem(grp, strcat(conName,'sw_RF'));
PLC_sw_reset = additem(grp, strcat(conName,'sw_reset'));
PLC_kp = additem(grp,strcat(conName,'kp'));
PLC_tau = additem(grp,strcat(conName,'tau'));
PLC_theta = additem(grp,strcat(conName,'theta'));
PLC_yk_est_f = additem(grp,strcat(conName,'Program:MainProgram.yk_est_f'));

PLC_Pc = additem(grp,strcat(conName,'Pc'));
PLC_Nc = additem(grp,strcat(conName,'Nc'));
PLC_lc = additem(grp,strcat(conName,'lc'));
```

```
PLC_dc = additem(grp,strcat(conName,'dc'));
PLC_n_coefs = additem(grp,strcat(conName,'n_coefs'));
PLC_Tm_DMC = additem(grp,strcat(conName,'Tm_DMC'));
PLC_len_gi = additem(grp,strcat(conName,'Program:MainProgram.len_gi'));

PLC_noiseMeasurement = additem(grp,strcat(conName,'noiseMeasurement'));
PLC_noiseProcess = additem(grp,strcat(conName,'noiseProcess'));
PLC_traza = additem(grp,strcat(conName,'traza'));

Mr = PLC_noiseMeasurement.Value;
Mq = PLC_noiseProcess.Value;
R = Mr*eye(ny);    %ruido medicion
Q = Mq*eye(nt);   %ruido proceso
Q(3,3) = 0.00001;     %ruido kp
Q(4,4) = 0.0001;      %ruido tau
Q(5,5) = 0.0000;    %ruido theta
Q(6,6) = 0.0000;   %ruido 1/(theta*tau)
display('OK')

while 1
    tic
    sw_EKF = PLC_sw_EKF.Value;
    sw_RF = PLC_sw_RF.Value;
    sw_reset = PLC_sw_reset.Value;
    Tm_DMC = PLC_Tm_DMC.Value;
    n_coefs = PLC_n_coefs.Value;
    uk = PLC_FV_02.Value;
    yk = PLC_FT_01.Value;
    if sw_reset
        P = 1000*eye(nt,nt);
        Kgain = zeros(nt,1);
        xhat = zeros(nt,1);
        uk1 = 0;
        F21 = 0;
        F22 = 0;
        F23 = 0;
        F24 = 0;
        F25 = 0;
        F26 = 0;
        F = zeros(nt,nt);
        write(PLC_sw_reset, 0);
    end
    %EKF
    ek_EKF  = yk-xhat(1);           %error
    y_est   = xhat(1)+Kgain(1)*ek_EKF;
    xhat(1) = y_est;
    duk     = (uk-uk1)/t0;
    if sw_EKF
        Pk          = P;
        %etapa de prediccion
        %discretizacion sistema primer orden con retardo
        dxhatk1_1 = xhat(2)*t0;
        dxhatk1_2 = (-2*xhat(1)-xhat(2)*xhat(5)-2*xhat(2)*xhat(4)-...
            xhat(3)*xhat(5)*duk+2*xhat(3)*uk)*xhat(6)*t0;
        xhatk1 = xhat + [dxhatk1_1;
                         dxhatk1_2;
                         0;
                         0;
```

```
                    0;
                    0];

    %A linealizada
    F21 = -2*xhat(6)*t0;
    F22 = 1-(2*xhat(4)+xhat(5))*xhat(6)*t0;
    F23 = (-xhat(5)*duk+2*uk)*xhat(6)*t0;
    F24 = -2*xhat(2)*xhat(6)*t0;
    F25 = (-xhat(2)-xhat(3)*duk)*xhat(6)*t0;
    F26 = (-2*xhat(1)-xhat(2)*xhat(5)-2*xhat(2)*xhat(4)-...
        xhat(3)*xhat(5)*duk+2*xhat(3)*uk)*t0;
    F = [1, t0, 0, 0, 0, 0;
         F21, F22, F23, F24, F25, F26;
         0, 0, 1, 0, 0, 0;
         0, 0, 0, 1, 0, 0;
         0, 0, 0, 0, 1, 0;
         0, 0, 0, 0, 0, 1];
    Pk1_est = F*Pk*F' + Q;

    %etapa de actualizacion de medicion
    Sk_EKF = H*Pk1_est*H'+ R;
    Kgain = Pk1_est*H'*1/Sk_EKF;

    %asignacion de resultados
    xhat = xhatk1+Kgain*ek_EKF;
    kp_est = xhat(3);
    tau_est = xhat(4);
    theta_est = xhat(5);
    P = (eye(nt)-Kgain*H)*Pk1_est;

    %Escritura parametros estimados al PLC
    if ~isnan(kp_est), write(PLC_kp,kp_est); end
    if ~isnan(tau_est), write(PLC_tau, tau_est); end
    if ~isnan(theta_est), write(PLC_theta, theta_est); end
    if ~isnan(xhat(1)), write(PLC_yk_est_f, xhat(1)); end
end

if sw_RF
    %Lectura parametros sintonizacion controlador
    Pc = cast(PLC_Pc.Value,'single');
    Nc = cast(PLC_Nc.Value,'single');
    lc = cast(PLC_lc.Value,'single');
    dc = cast(PLC_dc.Value,'single');
    kp_est = cast(PLC_kp.Value,'single');
    tau_est = cast(PLC_tau.Value,'single');
    theta_est = cast(PLC_theta.Value,'single');
    Tm_DMC = cast(PLC_Tm_DMC.Value, 'single');
    %Modelo del sistema
    num=kp_est;
    den=[tau_est 1];
    ret=theta_est;
    gp=tf(num,den);             %Funcion de transferencia continua
    gp.outputdelay=ret;         %Retardo del sistema
    ftz=c2d(gp,Tm_DMC);         %Funcion de transferencia discreta
    d=round(ret/t0);            %Retardo del sistema discreto
    gi=step(ftz,n_coefs);       %Respuesta escalon
    len_gi=length(gi)-1;        %Valor M, donde la respuesta entra a SS
```

```
        write(PLC_len_gi, len_gi);
        % Calculo de la Matriz G
        if Pc > len_gi
            Pc = len_gi-1;        %Limita Pc maximo
            write(PLC_Pc, Pc);
        end
        G=zeros(Pc,Nc);
        G(:,1)=gi(1:Pc);
        for i=2:Nc
            for j=2:Pc
                G(j,i)=G(j-1,i-1);
            end
        end
        Mn=inv(G'*dc*eye(Pc)*G + lc*eye(Nc))*G'*dc*eye(Pc);
        K1=Mn(1,:);                  %Primera fila de la matriz Mn
        %Escritura vector K1 al PLC
        for r = 1:Pc
           strg = strcat(conName,'K1[',num2str(r-1),']');
           PLC_K1 = additem(grp, strg);
           write(PLC_K1, K1(r));
           delete(PLC_K1);
        end
        %Escritura vector gi al PLC
        for r = 1:len_gi+1
           strg = strcat(conName,'gi[',num2str(r-1),']');
           PLC_gi = additem(grp, strg);
           write(PLC_gi, gi(r));
           delete(PLC_gi);
        end
        %Seccion se ejecuta 1 vez
        write(PLC_sw_RF, 0);
    end
    if isnan(trace(P))
        disp('La traza es NaN, reset de parametros')
        write(PLC_sw_reset,1)
    end
    %Memoria
    uk1 = uk;
    %Detencion de acuerdo con el tiempo de muestreo
    tEnd = toc;
    pause(t0-tEnd)
end
```

Anexo B. Códigos de PLC y SCADA

B.1. Condiciones iniciales

```
sw_am_dmc := 0;
noiseBand := 0.5;
Tm_PLC := 0.1;
dumax := 100;
sw_sp_f := 1;

//Planta
FV_01 := 0;
FV_02 := 0;
FV_03 := 0;
FV_04 := 0;
Sw_B01 := 0;
Sw_B02 := 0;
Sw_M01 := 0;

//Condiciones iniciales KF
noiseMeasurement := 0.5;
noiseProcess := 0.117;
kp := 0;
tau := 0;
theta := 0;
(*FOR r := 0 TO 9 DO
  u[r] := 0;
END_FOR;*)

//Condiciones iniciales DMC
Tm_DMC := 0.2;
n_coefs := 20;
Pc := 15; //Horizonte de prediccion
```

```
Nc := 2;  //Horizonte de control
dc := 0.5;  //Pondera error
lc := 10;  //Pondera actuador

//Condiciones iniciales PRBS
FOR i_prbs := 0 TO 6 BY 1 DO
  sh[i_prbs] := 1;
END_FOR;
uvar_prbs := 0;
uvar1 := 0;
act_prbs := 1;
delta_prbs := 4;
t_prbs := 50;
t := 0;
sw_prbs := 0;
umax_prbs := 5;
umin_prbs := -5;
sw_ci := 0;
```

B.2. DMC

```
//Mediciones
uk_f := FV_02;
yk_f := FT_01;

du_f := 0;
IF sw_am_dmc THEN
  //Controlador
  //Calculo de respuesta libre
  FOR r := 0 TO Pc-1 BY 1 DO
        vg_dot_duf := 0;
        FOR k := 0 TO len_gi-Pc-1 BY 1 DO
              vg_dot_duf := vg_dot_duf+(gi[r+k]-gi[k])*duf[k];
```

```
            END_FOR;
            FOR k := len_gi-Pc TO len_gi-1 BY 1 DO
                  vg_dot_duf := vg_dot_duf+(gi[len_gi]-gi[k])*duf[k];
            END_FOR;
            du_f := du_f + K1[r]*(SP_f-(yk_f + vg_dot_duf));
  END_FOR;
  //Error
  ek_f := ABS(SP_f-FT_01);
  //Aplica banda de ruido
  IF ek_f < noiseBand THEN
            du_f := 0;
  END_IF;
  FOR r:=len_gi-1 TO 1 BY -1 DO
            duf[r] :=  duf[r-1];
  END_FOR;
  duf[0] := du_f;
  FV_02 := uk_f+du_f;

  //Limitador salida
  IF FV_02 < 0 THEN
            FV_02 := 0;
  ELSIF FV_02 > 100 THEN
            FV_02 := 100;
  END_IF;
ELSE
  FV_02 := uk_f;
  //Setpoint tracking
  IF sw_sp_f THEN
            SP_f := FT_01;
  END_IF;
END_IF;
```

```
//Memoria
yk1_f := yk_f;
```

UNIVERSIDAD DE CONCEPCION – FACULTAD DE INGENIERIA
RESUMEN DE MEMORIA DE TITULO

Departamento	: Departamento de Ingeniería Eléctrica
Carrera	: Ingeniería civil electrónica
Nombre del memorista	: Miguel Alonso Álvarez Burgos
Título de la memoria	: Aplicación de Control de Matriz Dinámica a planta piloto
Fecha de la presentación oral	: 17/10/2024
Profesor(es) guía	: Juan Pablo Segovia Vera
Profesor(es) revisor(es)	: José R. Espinoza C./Sergio N. Torres I.
Concepto	:
Calificación	:

Resumen (máximo 200 palabras)

En el presente proyecto se implementa un controlador DMC para regular flujo, en conjunto con el Filtro de Kalman Extendido para la estimación paramétrica del proceso.

Se utiliza el Filtro de Kalman Extendido puesto que se trabaja un sistema no lineal con sus mediciones corrompidas por ruido blanco. Este se usa para estimar los parámetros del modelo escogido para representar el sistema. La salida del filtro es ingresada al controlador para calcular la respuesta escalón del sistema y consiguientemente realizar regulación del flujo.

Dado que los PLCs, en general, poseen una capacidad computacional limitada, además de no proveer de algunas herramientas matemáticas necesarias para el cálculo de los valores requeridos para ejecutar el controlador, se opta por utilizar Matlab para realizar estos cálculos y enviarlos al PLC mediante la conexión a un servidor OPC. Así, el PLC solo computa lo requerido para hacer control en tiempo real.

Se obtiene que el filtro logra predecir de manera precisa los estados del sistema y el controlador regula de manera adecuada el flujo.