Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Modelado y control predictivo de una planta de regulación de nivel

Autor: Olivia María Neila Jiménez

Tutor: Daniel Limón Marruedo

Dpto. Ing. Sistemas y Automática.
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Modelado y control predictivo de una planta de regulación de nivel

Autor:
Olivia María Neila Jiménez

Tutor:
Daniel Limón Marruedo
Profesor Titular

Dpto. Ing. Sistemas y Automática.
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado: Modelado y control predictivo de una planta de regulación de nivel

Autor: Olivia María Neila Jiménez
Tutor: Daniel Limón Marruedo

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

En primer lugar, agradecer a Daniel Limón y a Juan Manuel Escaño por darme la gran oportunidad de desarrollar este proyecto. Agradecer a mi tutora y supervisora Samira Roshany–Yamchi toda su dedicación y trabajo para la realización de éste. Mencionar a todas esas personas que me han ayudado a desarrollarlo durante estos cinco meses en especial a Bartlomiej Gnyszka, Adolfo Sanchez, Niel Canty, Kritchai Witheephanich y Louis Cronin. Por último agradecer al centro Nimbus y a todo su personal la oportunidad y el buen trato recibido todo este tiempo.

Olivia María Neila Jiménez
Cork, 2015

Resumen

En este trabajo se modela una planta de regulación de nivel en la cual se implementa un Controlador Predictivo Generalizado (GPC) a través un Programmable Logic Controller (PLC). En primer lugar se hace uso de la herramienta de MATLAB de identificación de parámetros, Ident Toolbox, para obtener un modelo aproximado a un tanque de agua no lineal, en el cual se implementarán las técnicas citadas. Los resultados del GPC serán comparados con los resultados de un controlador Proporcional Integral (PI). Para conseguir un modelo más preciso se hace uso de la técnica de Gain Scheduling para aproximar el modelo no lineal a un conjunto de modelos lineales. Para la implementación del controlador GPC en el sistema a través del PLC SIMATIC S7-200 Siemens® se ha hecho uso de una conexión OPC entre el autómata y MATLAB® , donde se encuentra el algoritmo de control. Después se ha usado la herramienta de MATLAB® Hybrid Toolbox, para obtener una solución explícita del GPC la cual sea implementable en el propio autómata, donde finalmente se podrá comprobar que los resultados son tan buenos como los del controlador online.

Abstract

In this project a level regulation plant will be modelled and the Generalized Predictive Controller (GPC) will be implemented using a Programmable Logic Controller (PLC). To begin with, the Parameter Identification Toolbox of MATLAB® will be used to model a nonlinear water tank. The results will then be compared with the Proportional Integral (PI) controller results. In order to get the precise model, Gain Scheduling techniques will be used to approximate the nonlinear model of the tank with a family of linear models. All these methods have been implemented in PLC SIMATIC S7-200 Siemens® by connecting MATLAB® to the OPC server. After this the Hybrid Toolbox of MATLAB® will be used to get the explicit GPC which can be implemented in the PLC.

Índice Abreviado

Índice

1 Introducción

1.1 Introducción

El objetivo de este proyecto es el de comprobar bajo una planta real, las ventajas del control avanzado frente al control convencional. Es sabido que controladores PID tradicionales están involucrados en todo tipo de ingenierías e industrias, pero en las últimas décadas las técnicas avanzadas de control han ido tomando lugar y demostrando la gran capacidad que tienen no solo en cuestión de control, sino también de optimización.

Para hacer posible esta demostración se ha hecho uso de la planta que será usada en un proyecto como sistema de simulación en el centro Nimbus, en el Instituto Tecnológico de Cork, Irlanda. La planta actualmente se encuentra en proceso de ampliación para la configuración de ésta como el sistema de prueba del proyecto mayor que se desarrolla en el centro Nimbus.

El interés de este proyecto no queda solo en el desarrollo del controlador de la planta. La puesta en marcha de ésta también ha formado parte del proyecto, haciendo estudio de todo lo necesario para configurar un autómata y de muchos de los problemas que se pueden presentar en una instalación real.

Como último objetivo del proyecto se ha querido obtener una solución compacta del controlador GPC, es decir, conseguir implementar un controlador GPC con restricciones, no explícito, dentro del PLC S7-1200.

1.2 Estado del arte

El desarrollo de los conceptos de control modernos se remonta a la obra de Kalman a principios de 1960 con el regulador lineal cuadrático (LQR) diseñado para minimizar una función objetivo cuadrática sin restricciones de los estados y las entradas. El horizonte infinito dotó al algoritmo LQR con potentes propiedades estabilizantes. Sin embargo éste tuvo poco impacto en el desarrollo de la tecnología de control en las industrias de proceso. La razón de esto radica en la ausencia de restricciones en su formulación, las no linealidades de los sistemas reales, y sobre todo el conocimiento de control en procesos industriales en el momento, en el que los técnicos de instrumentación y los ingenieros de control o bien no tenían conocimiento del control óptimo o lo consideraban como poco práctico. Así, los primeros defensores del MPC para el control de procesos procedieron de forma independiente, atendiendo las necesidades de la industria. [4]

A finales de 1970 varios artículos anunciaron aplicaciones exitosas de MPC en la industria, principalmente los de Richalet (1978) que presentó el Modelo Predictivo de Control heurístico (más tarde conocido como Modelo de Control algorítmico (MAC)) y los de Cutler y Ramaker (1980) con Dynamic Matrix Control (DMC).

El tema común de estas estrategias fue la idea de utilizar un modelo dinámico del (respuesta de impulso en el primero y en el paso respuesta más adelante) proceso para predecir el efecto de las acciones de control futuras, que se determinaron mediante la minimización del error predicho sujeto a restricciones operativas. La optimización se repite en cada periodo de muestreo con información actualizada del proceso. Estas formulaciones eran algorítmicas y heurísticas y aprovecharon el aumento del potencial de las computadoras

digitales en el momento.

Más tarde en una segunda generación de MPC (QDMC; Garcia, Morshedi, 1986) empezó a utilizarse la programación cuadrática para resolver el problema de control óptimo de bucle abierto limitado donde el sistema es lineal, el coste es cuadrático y las restricciones de los estados y del control son definidas por desigualdades lineales.

Por otro lado, otra línea de trabajo surgió de manera independiente en torno a ideas de control adaptativo como desarrollo de estrategias esencialmente para procesos monovariables formulados con modelos de entrada/salida haciendo uso de la ecuación Diophantine. La primera iniciativa vino de Astron (1970) con el control de mínima varianza.

Con el fin de hacer frente a las plantas de fase no mínima una entrada de penalización fue introducida en la función objetivo y esto se convirtió en el control generalizado de varianza mínima (GVM). Más tarde la entrada fue reemplazada por el incremento en la señal de control para garantizar un error en régimen permanente cero.

Bajo la base de las ideas de GVM Clarke (1987) fue desarrollado el Control Predictivo Generalizado (GPC) el cual es hoy uno de los métodos más populares. También se desarrollaron versiones de espacio de estado de restricciones GPC.

La situación actual de aplicaciones de MPC en la industria está bien reflejada en la recopilación de Qin y Badgwell [6]. La mayoría de las aplicaciones son en procesos multivariables. Sorprendentemente, MPC ha tenido menor impacto en otro tipo de industrias, aunque estudios de 1993 sugieren que unas 20.000 aplicaciones podrían beneficiarse de esta técnica. El éxito actual del MPC en la industria se debe a tres razones principales [2]:

- La incorporación de un modelo explícito del proceso en los cálculos permite al controlador tratar con todas las características importantes de la dinámica del proceso.
- La consideración del comportamiento del proceso a lo largo de un horizonte futuro permite tener en cuenta el efecto de las perturbaciones en alimentación y prealimentación, permitiendo al controlador conducir la salida a la trayectoria de referencia deseada.
- La consideración de restricciones en la fase del diseño del controlador evita en lo posible su violación, resultando en un control más preciso en torno al punto Fundamentos óptimo de operación. La inclusión de restricciones es quizás la característica que más distingue al MPC respecto a otras metodologías.

Otra de las razones que han contribuido a que el MPC se haya convertido en un éxito comercial es el hecho de que existen unos 15 suministradores que instalan el producto llave en mano, permitiendo que medianas empresas puedan tener acceso a esta tecnología. Aparte de esto, los nuevos Sistemas de Control Distribuido empiezan a ofertar productos MPC genéricos que ofrecen al usuario la posibilidad de realizar futuras modificaciones sin depender de un producto cerrado.

2 Descripción del Proyecto

2.1 Descripción del sistema

Se dispone de dos tanques, con la misma dimensión, desarrollo y aplicación, Figura 2.1. En adelante se describe el proyecto realizado en un solo tanque, sabiendo que dicho trabajo es aplicado a ambos. Debe saberse que el sistema es diferente al de la Figura 2.1, ya que se han cambiado los sensores.

El sistema se basa en un tanque cilíndrico de 120cm de altura y 15cm de diámetro. Está equipado por una bomba-motor centrífuga que introduce el agua al tanque desde la parte superior de éste. En la parte inferior del tanque se dispone de un sensor de presión, el cual sirve para conocer la cantidad de agua presente dentro del tanque. Junto a éste se encuentra la salida de agua del tanque, en la parte inferior, que está controlada mediante una válvula manipulable.

Como medida de seguridad existe un segundo sensor, en la parte superior del tanque que indica si el nivel de agua supera la altura donde el sensor está situado.

La salida de agua del tanque tiene lugar en un depósito de agua que se encuentra en la parte inferior al tanque y almacena el agua saliente de éste. El volumen de este depósito es medido mediante un sensor ultrasónico.

Todos los dispositivos del sistema están conectados a un PLC, el cual ha sido programado para el funcionamiento del sistema. El esquema que sigue la planta es el representado en la Figura 2.2.

Dispositivos

1. **Bomba-Motor Centrifuga:**
   Se dispone de una bomba que bombea el agua desde el depósito situado en la parte inferior de la planta hasta la parte superior del tanque.

   Esta bomba está propulsada mediante un motor de inducción, el cual tiene controlado su frecuencia mediante un VFD (Velocity Frecuency Drive). El VFD se controla mediante dos entradas, una entrada analógica de 0 a 10V DC la cual fija la frecuencia en hercios de la bomba, y otra entrada digital ON-OFF la cual enciende o apaga la bomba. Ambas señales están conectadas y serán controladas desde el autómata, Siemens S7-1200.

   • Estandarización de la señal: El variador de frecuencia Allen-Bradley Power Flex 4 Variable Frecuency Drives (VFD) es usado para el control de la velocidad y de la función encendido-apagado de la bomba. La señal digital ON/OFF es fácil de interpretar y trabajar con ella, en cambio la señal analógica necesita ser tratada para trabajar con ella en hercios.

   Debe conocerse que las entradas y salidas analógicas del autómata deben ser siempre de 0-10 voltios. Cada señal está definida en una variable que puede tomar como valor de 0 a 27648. Esta cuenta nos limita la precisión de las señales, Figura 2.3.

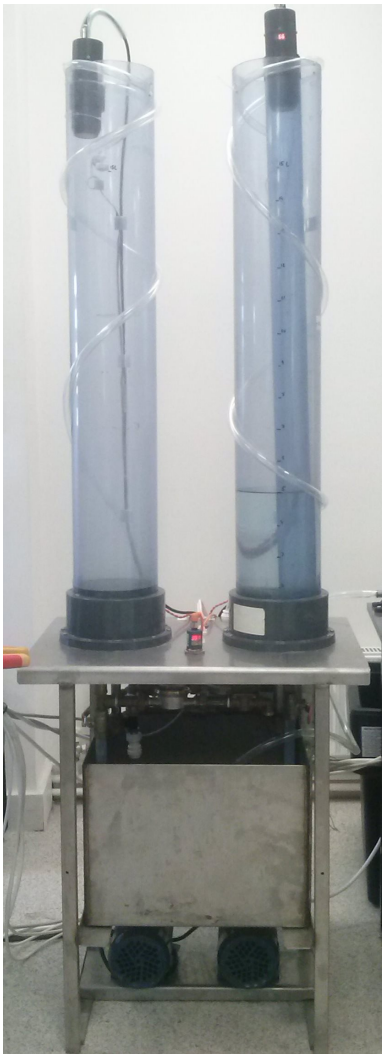

Figura 2.1 Planta de Pruebas.

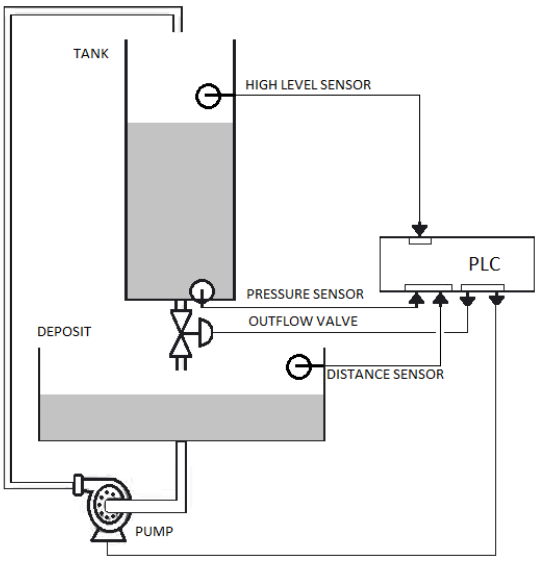

Figura 2.2 Esquema de la planta.

Se ha definido que 27counts equivale a 0.1Hz de modo que:
$27648/27 = 1024 \rightarrow$ Definiendo así frecuencia de 0 a 102,4 Hz con una precisión de 0.1/27 Hz.
Observando la bomba, a valores menores de 6.6Hz el caudal procedente de la bomba es cero, por lo cual se desplaza el rango de funcionamiento a 6.6Hz y 109Hz. Estos valores se configuran como máximo y mínimo en el VFD.

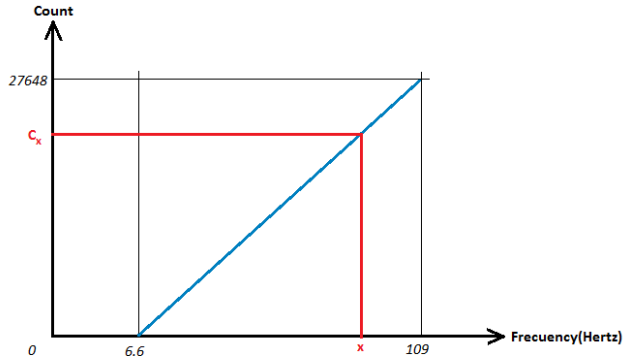

Figura 2.3 Escalado.

$$Count_{output} = \frac{f_{output} - f_{min}}{f_{max} - f_{min}} * Count_{max} \tag{2.1}$$

Donde f_{output} es la frecuencia calculada (Hertz), f_{min}=6.6Hz, f_{max}=109Hz, $Count_{max}$=27648 y $Count_{output}$ es la cuenta para la variable de salida del PLC.

2. **Válvula de Salida:**
La salida de agua está manipulada mediante una válvula motorizada, controlada mediante una señal de 0-10V DC desde un controlador Burkert que produce el pulso apropiado para fijar la apertura de ésta.

• Estandarización de la señal: al igual que para el VFD, la señal enviada a la válvula motorizada de salida de agua debe ser escalada con una relación entre (0-100) porcentaje de apertura y (0-10) Voltios para la señal.

$$Count_{output} = \frac{\%_{output} - \%_{min}}{\%_{max} - \%_{min}} * Count_{max} = \frac{\%_{output}}{\%_{max}} * Count_{max} \tag{2.2}$$

Donde $\%_{output}$ es el porcentaje de apertura deseado, $\%_{min}$=0, $\%_{max}$=100, $Count_{max}$=27648 y $Count_{output}$ es la cuenta para la variable de salida del PLC.

3. **Sensor de presión:**
El volumen del tanque es medido gracias al uso de un transductor de presión en la parte inferior del tanque. La señal que emite el sensor es de 4-20 mA, siendo 4mA equivalente a ausencia de presión o ausencia de líquido en el tanque. La presión máxima medible es de 5mWG.

• Estandarización de la señal: la señal enviada desde el sensor de presión es una señal analógica de 4-20 mA. Para leer esta señal en el PLC se ha añadido una resistencia de 500Ohm para que la magnitud de la señal recibida por el autómata esté en voltios, el rango de la señal se redefine entonces como 2-10V DC. Como se ha citado antes el rango de presión medible es de 0-5 mWG.
Teniendo en cuenta que: 1mWG=0.0981 bar, 1bar = 1kg/cm2, 1kg=1litre y calculado el valor máximo

requerido de presión: Volumen máximo=15 litros;
Volumen/Área Tanque = Presión ; $15/(7.5^2\pi) = 0.085$bar. Implica que se hace uso de menos del 20 % del rango del sensor.

Se conoce el volumen escalando la señal mediante el siguiente cálculo.
Para ello se han realizado dos medidas para conocer el valor del sensor cuando el tanque está vacío y lleno. Una vez conocidas ambas medidas, cualquier otra pueda escalarse de la siguiente manera:

$$Volume_{input} = \frac{C_{input} - C_{min}}{C_{max} - C_{min}} * Volume_{max} \tag{2.3}$$

Donde C_{input} es la cuenta recibida desde el sensor, $C_{min} = 5480$counts, $C_{max} = 9056$counts, $Volume_{max} =$ 15litres y $Volume_{output}$ es el volumen real medido.

4. **Sensor de distancia:**
Para medir el volumen del depósito de agua se hace uso de un sensor ultrasónico de distancia que mide la distancia desde la parte inferior de la mesa, donde los tanques están situados, al nivel de agua existente en el depósito. La señal que emite el sensor es una señal analógica de 0-10V DC.

• Estandarización de la señal: al igual que para el tanque, para conocer el volumen del deposito se necesita conocer la relación entre el voltaje del sensor y el volumen. Para ello se han realizado dos medidas para conocer el valor del sensor cuando el deposito esta vació y lleno. Una vez conocidas ambas medidas, cualquier otra pueda escalarse de la siguiente manera:

$$Volume_{input} = \frac{C_{input} - C_{min}}{C_{max} - C_{min}} * Volume_{max} \tag{2.4}$$

Donde C_{input} es la cuenta recibida desde el sensor $C_{min} = 9000$counts, $C_{max} = 4000$counts, $Volume_{max} =$ 30litres y $Volume_{output}$ es el volumen real medido.

5. **Sensor de posición:**
En la parte superior del tanque existe un sensor digital ON/OFF que se activa cuando el agua supera el nivel donde el sensor está situado. Este sensor se programara como sensor de seguridad de modo que cuando el sensor se active la bomba se apagara para evitar el rebosamiento del tanque.

6. **Panel de Control:**
El panel de control está construido tal y como se ve en la Figura 2.4. El cableado de cada aparato está localizado en el panel, todos ellos conectados al PLC, Programable Logic Controller. Tambien se encuentra conectada al PLC una pantalla HMI, Human Machine Interface. Todas las señales de cada sensor o actuador están conectadas al PLC para su conocimiento, control y modificación requerida.

7. **PLC:**
El controlador lógico programable, más conocido por sus siglas PLC, es una computadora que se define como un dispositivo electrónico digital que usa una memoria programable para guardar instrucciones y llevar a cabo funciones lógicas de configuración de secuencia, de sincronización, de conteo y aritméticas, para el control de maquinaria y procesos. El PLC utilizado en el proyecto es el mostrado el la Figura 2.5 SIMATIC S7-1200 Siemens [9], CPU 1214C.

8. **Pantalla HMI:**
La pantalla HMI, Human Machine Interface, también conocida como display o pantalla de operador se define como un elemento de comunicación entre el usuario y el proceso de la planta. La pantalla

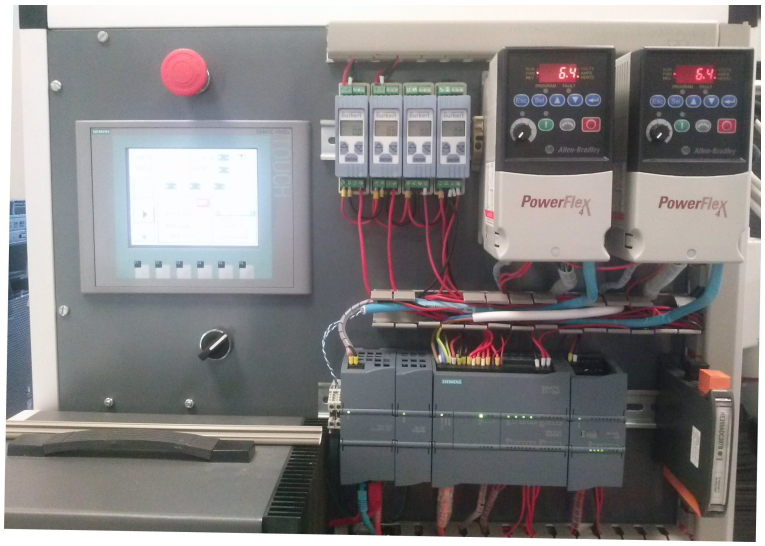

Figura 2.4 Panel de Control.

Figura 2.5 PLC Simatic S7-1200.

SIMATIC HMI de la que se dispone en la planta se comunica directamente con el PLC S7-1200 y se configuran intuitivamente desde TIA Portal, Figura 2.6. Esta pantalla sirve para interactuar con el proceso una vez esté programada para ello, desde la cual una vez programado el PLC se va a poder poner en marcha, cambiar modos de operación o apagar el sistema.

2.2 Descripción teórica MPC

2.2.1 Control Predictivo basado en Modelo MPC

El Control Predictivo Basado en Modelo [3] [7], Model (Based) Predictive Control (mbpc o mpc) constituye un campo muy amplio de métodos de control desarrollados en torno a ciertas ideas comunes e integra diversas disciplinas como control óptimo, control estocástico, control de procesos con tiempos muertos, control multivariable o control con restricciones. Estas ideas que aparecen en todo control predictivo son:

- Uso de un modelo para la predicción de la salida en un intervalo de tiempo futuro.
- Cálculo del conjunto de señales de control óptimo minimizando cierta función objetivo.
- Uso de una estrategia deslizante.

La estrategia utilizada por cualquier controlador de la familia de los controladores predictivos se caracterizan de la siguiente manera, representada en la Figura 2.7

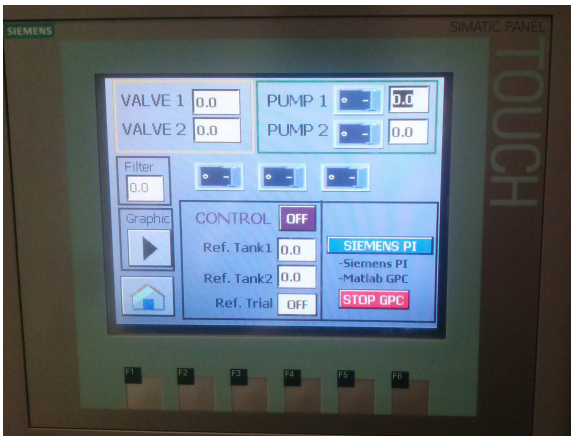

Figura 2.6 Pantalla Simatic HMI.

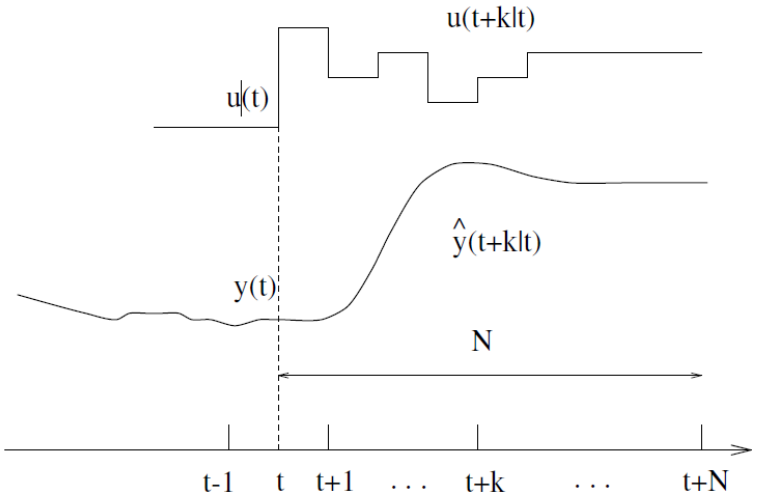

Figura 2.7 Estrategia del Control Predictivo.

- En cada instante t y usando el modelo se predicen las futuras salidas durante el horizonte de predicción N.
- Las predicciones $\hat{y}(t+k|t)$ para $k = 1...N$ dependen de los valores conocidos hasta el instante t (entradas y salidas pasadas) y de las señales de control futuras $u(t+k|t)$, $k = 0...N-1$ que se pretenden mandar al sistema que son las que se quieren calcular.
- La secuencia de señales de control futuras se calcula optimizando una función de coste que expresa la bondad de control. Esta función de coste suele ser una función cuadrática de los errores entre la salida predicha y la trayectoria de referencia, incluyendo en muchos casos el esfuerzo de control.
- Adicionalmente se hace alguna suposición como por ejemplo que la señal de control va a ser constante a partir de cierto instante. El resultado de la optimización es una secuencia óptima de actuaciones futuras:
  $u* = [u(t|t), u(t+1|t), u(t+2|t), ..., u(t+N-1|t)]^T$
- La señal u(t|t) se aplica y se desechan todas las demás $u(t+1|t)$,... , debido a que en el siguiente instante de muestreo ya se conoce $y(t+1)$ y se repite la minimización de la función de conste con el nuevo valor.
- Se calcula por tanto $u(t+1|t+1)$ (que en general sera diferente que $u(t+1|t)$ al calcularse con nueva información), haciendo uso del concepto de horizonte deslizante.

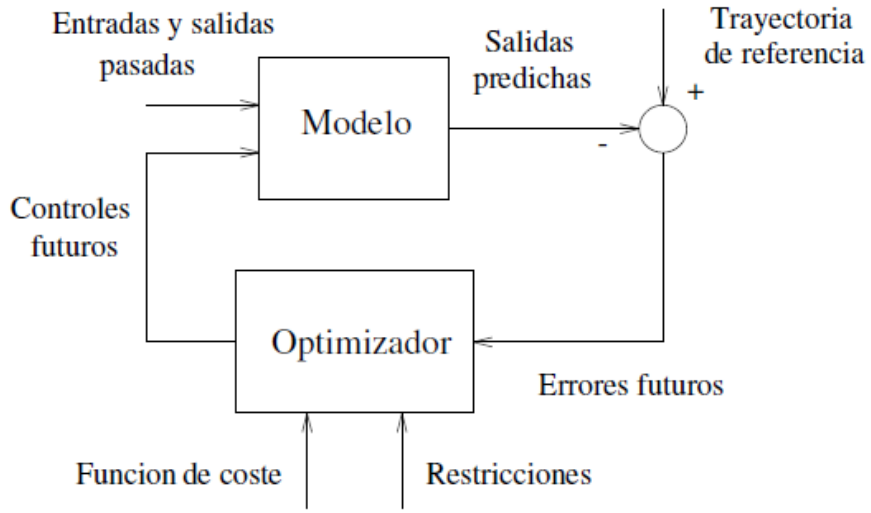

Figura 2.8 Estrategia del Control Predictivo.

Para llevar a cabo esta estrategia se usa una estructura como la de Figura 2.8. Se hace uso de un modelo para predecir las salidas futuras del proceso, basándose en las futuras señales de control propuestas. Estas señales son calculadas por el optimizador teniendo en cuenta la función de coste (donde aparece el futuro error de seguimiento) así como las restricciones. Por tanto el modelo juega un papel decisivo en el controlador. El modelo elegido debe ser capaz de capturar la dinámica del proceso para poder predecir las salidas futuras al mismo tiempo que debe ser sencillo de usar y de comprender.

El optimizador es parte fundamental de la estrategia pues proporciona las acciones de control. Si la función de coste es cuadrática, el mínimo se puede obtener como una función explícita de las entradas y salidas pasadas de la trayectoria de referencia. Sin embargo, cuando existen restricciones la solución debe ser calculada por métodos numéricos.

2.2.2 Elementos básicos

Modelo de predicción

El modelo de predicción es una de las claves del MPC ya que la fidelidad entre el modelo y el sistema real es lo que, en gran parte, caracteriza a un buen controlador predictivo. El uso del modelo del proceso viene determinado por la necesidad de predecir la salida en instantes futuros. Existen muchas técnicas para la obtención del modelo de un proceso y para el MPC se recomendará aquella que determine un modelo que describa de la mejor manera posible el proceso, de manera que éste pueda capturar al máximo la dinámica del proceso, que sea capaz de permitir el cálculo de las predicciones, y a la vez sea intuitivo y permita un análisis teórico.

Entre todos los tipos de modelos, para desarrollar el resto de ideas, se va a hacer uso de un modelo de función de transferencia ya que éste sera el usado posteriormente en la implementación del proyecto.

Se utiliza el concepto de función de transferencia: $G = A/B$. Donde la salida viene dada por:

$$\begin{aligned} A\left(z^{-1}\right) y(t) &= B\left(z^{-1}\right) u(t) \\ A\left(z^{-1}\right) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na} z^{na} \\ B\left(z^{-1}\right) &= 1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{nb} z^{nb} \end{aligned} \tag{2.5}$$

Por lo que la predicción vendrá dada por:

$$\hat{y}(t+k|t) = \frac{B\left(z^{-1}\right)}{A\left(z^{-1}\right)} u(t+k|k) \tag{2.6}$$

- Respuesta libre y forzada:
  Una característica importantes del MPC es el uso del concepto de la respuesta libre y respuesta forzada en la estrategia de control. La idea es expresar la secuencia de acciones de control como la suma de dos señales:

$$u(t) = u_l(t) + u_f(t) \tag{2.7}$$

Donde:

1. La señal $u_l(t)$ corresponde a las entradas pasadas (anteriores al instante t):

$$u_l(t-j) = u(t-j) \quad \text{para} \quad j = 1,2,\ldots$$
$$u_l(t+j) = u(t-1) \quad \text{para} \quad j = 0,1,2,\ldots$$

2. La señal $u_f(t)$ corresponde a las señales de control en los instantes futuros y vale cero en el pasado:

$$u_f(t-j) = 0 \qquad \text{para} \quad j = 1,2,\ldots$$
$$u_f(t+j) = u(t+j) - u(t-1) \quad \text{para} \quad j = 0,1,2,\ldots$$

Entonces la predicción de la secuencia de salida se separa también en dos partes.
Una de ellas, la respuesta libre, corresponde a la predicción de la salida cuando la variable manipulada se hace igual a $u_l(t)$, y la otra, la repuesta forzada, corresponde a la predicción de la salida cuando la señal de control es $u_f(t)$. Por lo tanto se define la respuesta libre como la evolución del proceso debido a su estado actual (incluido por tanto el efecto de acciones pasadas) mientras que la respuesta forzada es aquella debida a las acciones de control futuras.

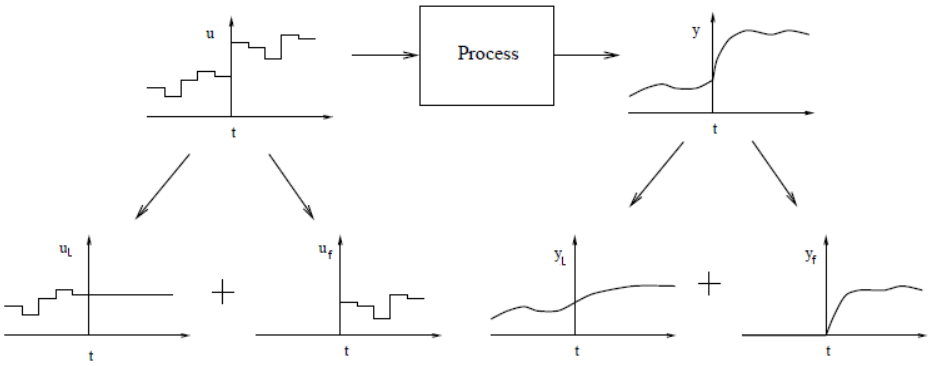

Figura 2.9 Respuesta libre y forzada.

Función objetivo

Los diversos algoritmos de MPC proponen distintas funciones de coste para la obtención de la ley de control. En general se persigue que la salida futura en el horizonte considerado siga a una determinada señal de referencia al mismo tiempo que se puede penalizar el esfuerzo de control requerido para hacerlo. La expresión general de tal función objetivo será:

$$J(N_1,N_2,N_c) = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2 \tag{2.8}$$

Donde:

- N_1 y N_2 definen el horizonte de predicción o intervalo de tiempo en el que se considera la evolución predicha del sistema.

- $N_1 > 1$ implica que no se tienen en cuenta los errores de seguimiento al comienzo. En caso de existencia de retraso, se optimiza a partir del primer error que se pueda minimizar $N_1 \geq (d+1)$.
- N_u define el horizonte de control o intervalo en el que se consideran incrementos sobre la acción de control. Si $N_u < N_2$ se suele suponer $\Delta u(t+j-1) = 0$ para $j > N_u$.
- $\delta(j)$ y $\lambda(j)$ son secuencias de ponderación que dan más o menos peso a los valores del error o del esfuerzo de la señal de control respectivamente. Una ponderación mayor del esfuerzo de la señal de control hace al control más lento pero a su vez más robusto.

2.2.3 Control Predictivo Generalizado GPC

El Control Predictivo Generalizado GPC es un método nacido en el mundo académico, propuesto por Clarke (1987), se ha convertido en uno de los métodos más populares en el ámbito del Control Predictivo académico como en el mundo industrial. Se ha empleado con éxito en numerosas aplicaciones industriales, mostrando buenas prestaciones a la vez que un cierto grado de robustez. La idea básica del GPC es calcular una secuencia de futuras acciones de control de tal forma que minimice una función de coste multipaso. El indice a minimizar es una función cuadrática que mide por un lado la distancia entre la salida predicha del sistema y una cierta trayectoria de referencia hasta el horizonte de predicción, y por otro el esfuerzo de control necesario para obtener dicha salida. El Control Predictivo Generalizado tiene muchas ideas en común con otros controladores predictivos ya que está basado en las mismas ideas pero posee a su vez algunas diferencias. Es capaz de proporcionar una solución explícita (en ausencia de restricciones), puede trabajar con procesos inestables o de fase no mínima e incorpora el concepto de horizonte de control así como la consideración en la función de coste de ponderación de los incrementos en las acciones de control. Las diversas posibilidades disponibles para el GPC conducen a una gran variedad de objetivos de control comparado con otras realizaciones, algunas de las cuales pueden ser consideradas como subconjuntos o casos límites del GPC.

Formulación del Control Predictivo Generalizado

El GPC emplea un modelo CARIMA para la predicción de la salida

$$A\left(z^{-1}\right)y(t) = B\left(z^{-1}\right)z^{-d}u(t-1) + C(z^{-1})\frac{e(t)}{\Delta} \tag{2.9}$$

donde la perturbación viene dada por un ruido blanco coloreado por el polinomio $C(z^{-1})$.

El algoritmo del Control Predictivo Generalizado consiste en aplicar una secuencia de señales de control que minimice una función de coste de la forma:

$$J(N_1,N_2,N_c) = \sum_{j=N_1}^{N_2}\delta(j)[\hat{y}(t+j\,|t) - w(t+j)]^2 + \sum_{j=1}^{N_u}\lambda(j)\,[\Delta u(t+j-1)]^2 \tag{2.10}$$

donde $\hat{y}(t+j|t)$ es la predicción óptima j pasos hacia delante de la salida del proceso con datos conocidos hasta el instante t, N_1 y N_2 son los horizontes mínimo y máximo de coste, N_u es el horizonte de control y $\delta(j)$ y $\lambda(j)$ son las secuencias de ponderación mientras que w(t+j) es la futura trayectoria de referencia. En muchas situaciones se considera $\delta(j)$ y $\lambda(j)$ constantes.

El objetivo es el cálculo de la futura secuencia de control u(t), u(t+1),... de tal manera que la salida futura del proceso y(t+j) permanezca próxima a w(t+j). Esto se logra minimizando $J(N_1,N_2,N_u)$.

- Predicción óptima

  Para obtener la predicción y(t+j) se hacen las siguientes transformaciones. Partiendo del modelo CARIMA:

  $$A\left(z^{-1}\right)y(t) = B\left(z^{-1}\right)z^{-d}u(t-1) + C(z^{-1})\frac{e(t)}{\Delta} \tag{2.11}$$

  Los polinomios $E_j(z^{-1})$ y $F_j(z^{-1})$ se definen, de grados j-1 y na, dividiendo el polinomio 1 entre $\tilde{A}(z^{-1}) = \Delta A(z^{-1})$hasta que se obtiene un resto que se pueda factorizar como $z-jF_j(z-1)$ y siendo

el cociente el polinomio $E_j(z-1)$. Esto implica que $E_j(z^{-1})$ y $F_j(z^{-1})$ satisfacen la relación de la ecuación diofántica:

$$1 = E_j(z^{-1})\tilde{A} + z^{-j}F_j(z^{-1}) \tag{2.12}$$

De este modo se puede transformar el modelo de la siguiente manera:
En primer lugar multiplicando el modelo por $E_j^{-1}z^j\Delta$ se obtiene:

$$\tilde{A}\left(z^{-1}\right)E_j(z^{-1})y(t+j) = E_j(z^{-1})B\left(z^{-1}\right)\Delta u(t+j-d-1) + E_j^{-1}e(t+j) \tag{2.13}$$

que usando la ecuación diofántica:

$$(1-z^{-j}F_j(z^{-1}))y(t+j) = E_j(z^{-1})B\left(z^{-1}\right)\Delta u(t+j-d-1) + E_j^{-1}e(t+j) \tag{2.14}$$

también transformable como:

$$y(t+j) = F_j(z^{-1})y(t) + E_j(z^{-1})B\left(z^{-1}\right)\Delta u(t+j-d-1) + E_j^{-1}e(t+j) \tag{2.15}$$

Es importante destacar que al ser de grado $E_j(z-1)\,j-1$, los términos de la expresión $E_j^{-1}e(t+j)$ son una combinación lineal de errores futuros, por lo que su valor esperado es cero, quedando así la mejor predicción como:

$$\hat{y}(t+j|t) = F_j(z^{-1})y(t) + G_j(z^{-1})\Delta u(t+j-d-1) \tag{2.16}$$

donde $G_j(z^{-1}) = E_j(z^{-1})B\left(z^{-1}\right)$.

Teniendo en cuenta que $N_1 \geq d+1$, el horizonte N_1 hasta N_2 se puede indicar como el intervalo desde d+1 hasta d+N. El conjunto de predicciones a calcular serán:

$$\begin{aligned}
\hat{y}(t+d+1|t) &= F_{d+1}(z^{-1})y(t) + G_j d+1(z^{-1})\Delta u(t) \\
\hat{y}(t+d+2|t) &= F_{d+2}(z^{-1})y(t) + G_j d+2(z^{-1})\Delta u(t+1) \\
&\vdots \\
\hat{y}(t+d+N|t) &= F_{d+N}(z^{-1})y(t) + G_j d+N(z^{-1})\Delta u(t+N-1)
\end{aligned} \tag{2.17}$$

El conjunto de predicciones se puede expresar en forma condensada:

$$y = Gu + F(z^{-1})y(t) + G'(z^{-1})\Delta u(t-1) \tag{2.18}$$

$$y = \begin{bmatrix} \hat{y}(t+d+1|t) \\ \hat{y}(t+d+2|t) \\ \vdots \\ \hat{y}(t+d+N|t) \end{bmatrix} u = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+N-1) \end{bmatrix} G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g_{N-1} & g_{N-1} & \cdots & g_0 \end{bmatrix}$$

$$F(z^{-1}) = \begin{bmatrix} F_{d+1}(z^{-1}) \\ F_{d+2}(z^{-1}) \\ \vdots \\ F_{d+N}(z^{-1}) \end{bmatrix} G'(z^{-1}) = \begin{bmatrix} z(G_{d+1}(z^{-1}) - g_0) \\ z^2(G_{d+2}(z^{-1}) - g_0 - g_1 z^{-1}) \\ \vdots \\ z^N(G_{d+N}(z^{-1}) - g_0 - g_1 z^{-1} - \cdots - g_{N-1}z^{-(N-1)}) \end{bmatrix}$$

El conjunto de predicciones queda definido como:

$$\begin{aligned}
y &= Gu + f \\
\text{con } f &= F(z^{-1})y(t) + G'(z^{-1})\Delta u(t-1)
\end{aligned} \tag{2.19}$$

- Obtención de la ley de control

  Tras definir la predicción óptima, la función de coste (2.10) puede reescribirse como:

$$J = (Gu + f - w)^T(Gu + f - w) + \lambda u^T u \tag{2.20}$$

La ecuación (2.20) se puede poner como:

$$J = \frac{1}{2}u^T Hu + bu + f_0 \tag{2.21}$$

donde:

$$\begin{aligned} H &= 2(G^T(\lambda_1 I^N)G + \lambda_2 I^{N_c}) \\ b &= 2\lambda_1(f-w)^T G \\ f_0 &= (f-w)^T(f-w) \end{aligned}$$

En el caso de que no existan restricciones, el mínimo de la ecuación (2.21) se da en:

$$u = -H^{-1}b^T \tag{2.22}$$

- Restricciones

  Una restricción es toda aquella limitación existente en un sistema. Las restricciones más comunes son:

$$\begin{array}{lccl} U_{min} \leq & u(t) & \leq U_{max} & \forall t \\ u_{min} \leq & u(t)-u(t-1) & \leq y_{max} & \forall t \\ y_{min} \leq & y(t) & \leq y_{max} & \forall t \end{array} \tag{2.23}$$

  Teniendo en cuenta la ecuación de predicción (2.19) y la secuencia de actuaciones **u** las restricciones pueden reescribirse como [8]:

$$\begin{array}{lccl} \overline{1}U_{min} \leq & T\mathbf{u}(t) + u(t-1)\overline{1} & \leq \overline{1}U_{max} & \forall t \\ \overline{1}u_{min} \leq & \mathbf{u} & \leq \overline{1}y_{max} & \forall t \\ \overline{1}y_{min} \leq & G\mathbf{u} + f & \leq \overline{1}y_{max} & \forall t \end{array} \tag{2.24}$$

  De modo que todas las restricciones puedan expresarse como:

$$Ru \leq c \tag{2.25}$$

  Previamente se ha definido la secuencia de actuaciones óptimas, cuando no existen restricciones, como el mínimo de la ecuación (2.21) , definido como la ecuación explícita (2.22).

  Con la adición de restricciones el problema general de control predictivo cambia y la secuencia de actuaciones óptimas se calculan resolviendo:

$$\begin{aligned} minJ &= \frac{1}{2}\overline{u}^T H\overline{u} + b\overline{u} \\ &\text{sujeto a} \quad Ru \leq c \end{aligned} \tag{2.26}$$

  Es decir, el problema consiste en la minimización de una función cuadrática con restricciones lineales, lo que se conoce como Programación Cuadrática, QP. En este caso no se puede encontrar una solución analítica como en el caso sin restricciones, sino que hay que recurrir a métodos iterativos. Esto conlleva a una carga de cálculo considerable, ya que se obtiene la solución resolviendo el algoritmo iterativo en cada periodo de muestreo.

3 Implementación del controlador

3.1 Conexión OPC

Para comunicar el autómata con MATLAB® se hace uso de un servidor OPC. El servidor OPC, instalado en un PC, se conecta con las variables existentes de una computadora, en este caso el autómata, y deja que sean variables de lectura o escritura desde un cliente OPC.
Para ello el esquema que sigue el sistema es mostrado en la Figura 3.1:

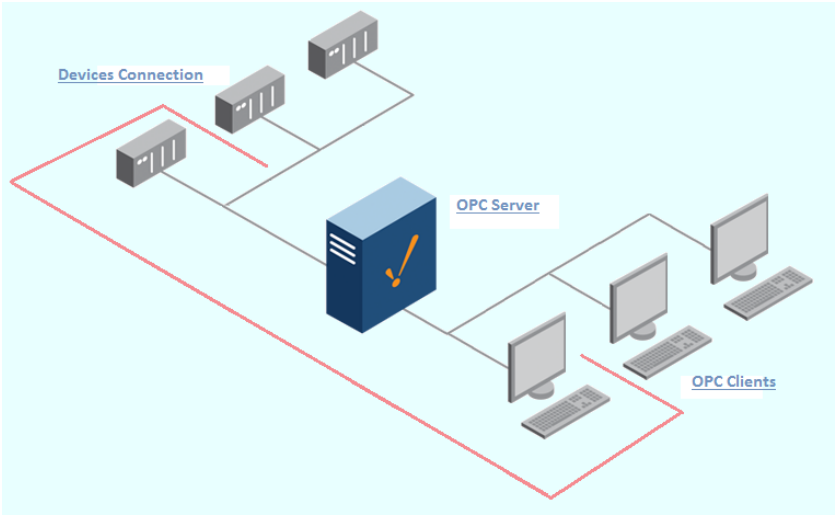

Figura 3.1 Esquema conexión OPC.

KepServerex ha sido el servidor OPC utilizado en el proyecto. La configuración de éste es sencilla:

1. Instalar el programa KepServerex OPC teniendo en cuenta durante su instalación que hay que añadir los Drivers necesarios, en el caso de la planta: Siemens Drivers.
2. Una vez instalado el programa, se configura la conexión del dispositivo, el autómata.
3. Como último paso, tras tener hecha la conexión del dispositivo, se añaden una por una las variables que necesiten ser compartidas con el cliente OPC.
4. Se puede comprobar el buen funcionamiento del OPC tan solo abriendo el cliente OPC que dispone el propio servidor, Figura 3.2.

La configuración de MATLAB® como Cliente OPC es sencilla ya que MATLAB® dispone de las herramientas necesarias para crear el cliente. La creación del cliente puede ser descrita según los pasos siguientes:

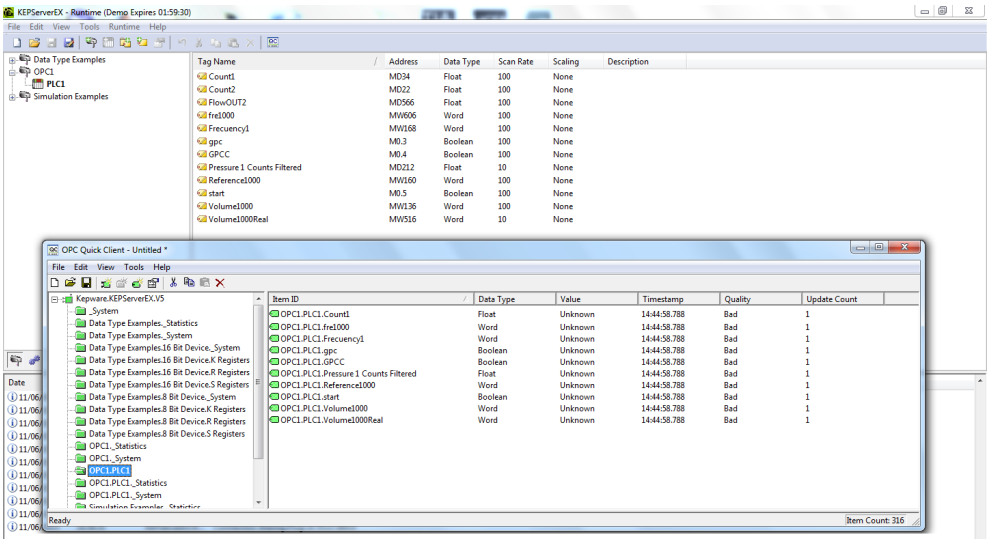

Figura 3.2 OPC Server and OPC Client.

1. En primer lugar para conocer información acerca de los servidores instalados en el equipo local se hace uso de la función:
   opcserverinfo('localhost')
2. A continuación se crea el acceso a los datos del servidor instalado en el equipo local mediante la función:
   da= opcda('Host','ServerID')
3. La conexión, después de crear el acceso entre el servidor y el equipo local, está desactivada. Para activarla hay que hacer uso de la función:
   Connect(da)
4. Una vez realizados estos pasos, si la conexión se realiza satisfactoriamente, el cliente OPC ya está creado y preparado para su uso.
5. Como último paso hay que definir en MATLAB® las variables del servidor OPC que se quieren usar. La función que es usada para esto es la siguiente:
   IObj = additem('GroupObj',ÍName')
   El nombre de cada variable debe coincidir con el definido previamente en el servidor.

Código 3.1 Código MATLAB® : Conexión MATLAB® Client - Server.

```
client=opcserverinfo('localhost');
da=opcda('127.0.0.1','Kepware.KEPServerEX.V5');
connect(da);
group1=addgroup(da);
y0_1 = additem(group1,'OPC1.PLC1.Volume');
u0_1 = additem(group1,'OPC1.PLC1.Frecuency');
r0_1 = additem(group1,'OPC1.PLC1.Reference');
y0_real = additem(group1,'OPC1.PLC1.VolumeReal');
GPC = additem(group1,'OPC1.PLC1.GPC');
start = additem(group1,'OPC1.PLC1.StartCopy');
```

3.2 Programación en TIA PORTAL del autómata Simatic S7-1200

El PLC Simatic S7-1200 [9] dispone de un software TIA PORTAL en el que se programan todas las conexiones y programas necesarios en el interior del autómata. A lo largo del proyecto se ha programado bastante en autómata, en este apartado se han querido citar tan solo algunas de las partes más importantes.

3.2.1 Variables de entrada y salida

- Las entradas analógicas al autómata se almacenan en variables tipo %IW. Tal y como se explicó en el apartado 2.1 cualquier entrada analógica al autómata es una señal entre 0 y 10 voltios, que se convierte en una cuenta entre 0 y 27648 en la variable %IW, correspondiendo 0 a 0 voltios y 27648 a 10 voltios. Dichas señales analógicas deben ser escaladas tanto para su lectura como para la escritura. En el caso de la entrada analógica del sensor de presión la transformación se ha realizado como se representa en la Figura 3.3.

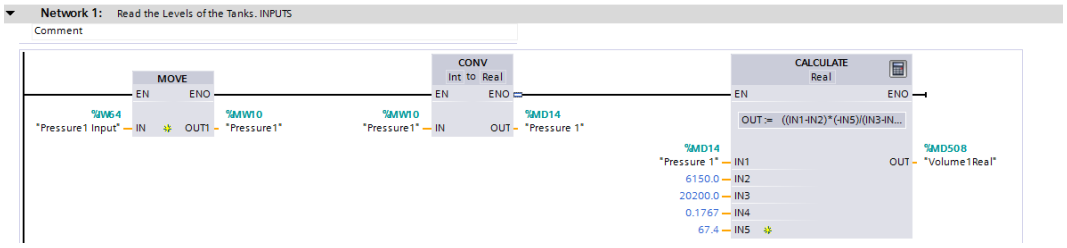

Figura 3.3 Señal de entrada. TIA PORTAL.

Donde "Pressure1 Input" es la señal de entrada al autómata (count), IN2 es C_{min}: valor cuando el volumen es cero, IN3 es C_{max}: valor cuando el volumen es máximo, IN4 es el área del tanque, IN5 es la altura cuando el volumen es máximo y "Volume1Real" es el volumen real tras ser escalado. Figura 3.3.

La conversión intermedia que se puede ver en la Figura 3.3 es para pasar la variable de tipo entero a variable tipo real. Este tipo de transformaciones son importantes y han sido bastante usadas en la programación del PLC.

- Las salidas analógicas del autómata se definen del tipo %QW y se programan de manera parecida, pero en orden inverso a las anteriores citadas. De la misma manera para la apertura de las válvulas como para la frecuencia de la bomba, el valor que se les quiere aplicar, ya sean hercios o porcentaje de apertura, debe ser escalado a su valor correspondiente entre 0 y 27648 y guardado en la variable de salida. Este proceso se puede ver en la Figura 3.4 y la Figura 3.5.

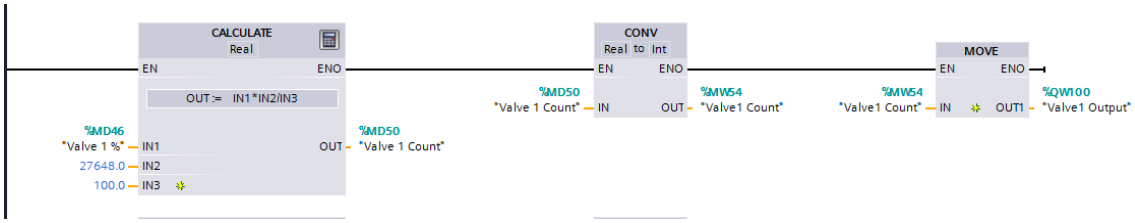

Figura 3.4 Señal de salida. TIA PORTAL.

Donde "Valve 1 %" es el valor del porcentaje de apertura que se quiere enviar a la válvula, "Valve1 Count": es el valor escalado a counts transformado a entero y "Valve1 Output" es la variable de salida del autómata donde se envía el valor a la válvula. Figura 3.4.

Donde "Pump1 Frec" es el valor en frecuencia que se quiere mandar al motor, "Pump 1 Count": es el valor escalado a counts transformado a entero y "Pump1 Output" es la variable de salida del autómata donde se envía el valor a la válvula. Figura 3.5.

- La programación de las entradas y salidas digitales resulta más sencilla ya que no necesitan de ninguna transformación. Dichas señales se encuentran en variables tipo %I0 y %Q0. Están variables son variables booleanas, las cuales solo tienen dos valores posibles, 0 y 1. Como se citó en el apartado 2.1 la entrada digital del sensor de volumen se utiliza como seguridad y en el caso de estar activa la bomba

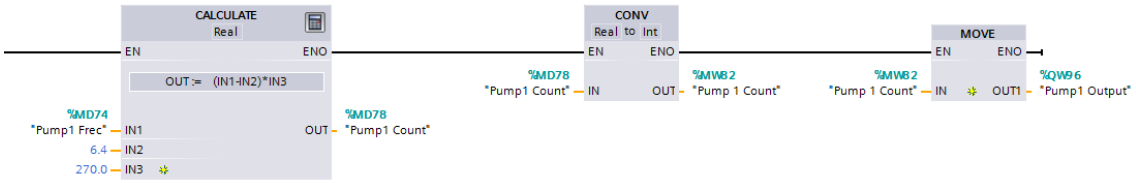

Figura 3.5 Señal de salida 2. TIA PORTAL.

debe paralizarse. La programación de estas señales se ha configurado de la siguiente manera, Figura 3.6.

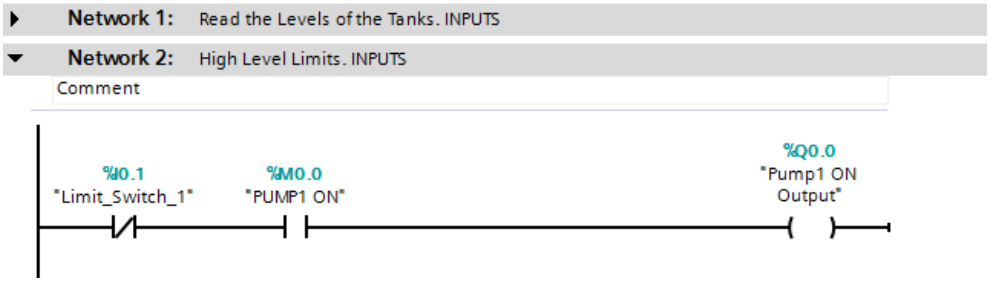

Figura 3.6 Señales digitales. TIA PORTAL.

Donde "Limit_Swith_1" es la variable digital de entrada y "Pump1 ON Output" es la variable digital de salida. Programándose así que en el caso de que se sobrepase el límite de agua permitido %I0.1=1 se apague la bomba definiendo %Q0.0=0. Figura 3.6.

Debe saberse que las variables de entrada y salida de señales al autómata están predefinidas y limitadas, variables %I (Inputs) y %Q (Outputs). El resto de variables usadas en el software para transformaciones o el resto de operaciones necesarias se definen en la memoria del PLC donde las variables se llaman %M, pueden ser de todos los tipos, y se han usado según han sido necesarias tipo booleanas, reales, enteras, etc.

3.2.2 HMI

El sistema también dispone de una pantalla HMI como se ha citado en el apartado 2.1. Desde TIA PORTAL se ha programado la pantalla, Figura 3.7 creando un acceso directo a ciertas variables del sistema, pudiendo desde ella encender las bombas, cambiar su frecuencia, definir el valor de la apertura de las válvulas de salida, constante de filtro, etc. También desde la pantalla HMI se puede activar al modo de control programado en el software, que activa los controladores GPC o PI según se quiera.

3.2.3 Controlador PID

En este proyecto se ha querido comparar un controlador tradicional PI con un controlador avanzado GPC con restricciones. Para ello se ha hecho uso del bloque PID que dispone el software TIA PORTAL y se ha utilizado la aplicación de auto–tuning que el bloque dispone, Figura 3.8 y Figura 3.9. El resultado ha sido un controlador PI con las siguientes constantes:

$$\begin{aligned} Kp &= 142.45 \\ Ti &= 6.64 \text{ sec} \\ Ts &= 1 \text{ sec} \end{aligned} \tag{3.1}$$

Éste ha sido el PI que se ha utilizado en el proyecto. Resulta interesante el uso de este controlador, ya que en muchísimos casos los parámetros de los PID usados en la industria son obtenidos con

Figura 3.7 Pantalla HMI.

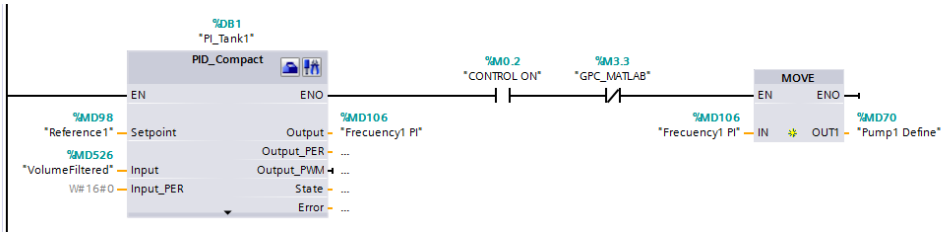

Figura 3.8 Bloque PID TIA PORTAL.

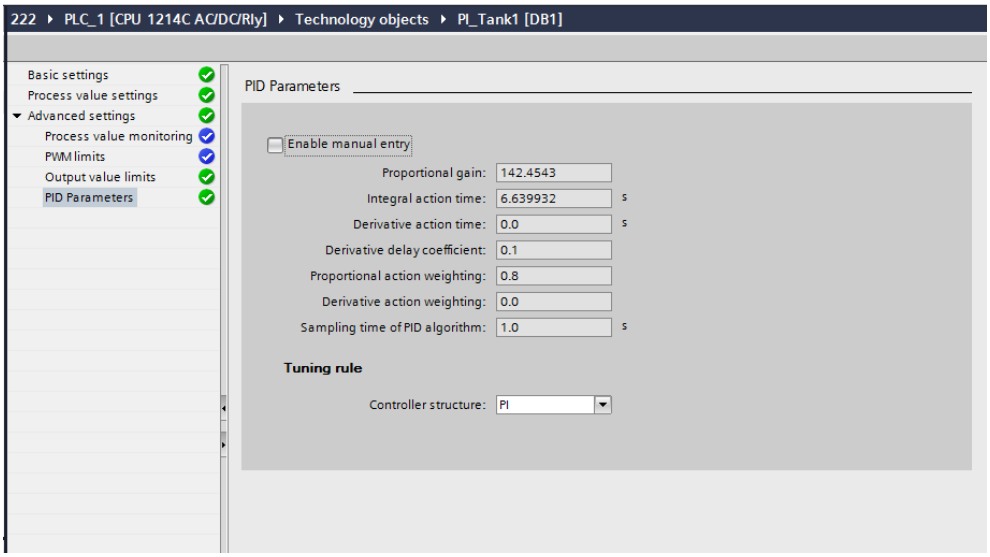

Figura 3.9 Parámetros PI TIA PORTAL.

aplicaciones de auto-tuning que tienen los autómatas donde el controlador se implementa, por lo que resulta interesante la comparación con un PID que se puede encontrar frecuentemente en la realidad.

3.2.4 Referencia

En este proyecto, es necesario un set point o referencia de la variable a controlar. En función de los requerimientos para el proyecto la referencia se ha definido de varias maneras.

- Desde la pantalla HMI, pudiendo cambiar el valor de la referencia en el momento que se quiera.

En este caso la referencia lee el valor de una variable %MW definida desde la pantalla HMI.

– Predefinida desde MATLAB® . Opción muy útil ya que se han creado ciertos programas de cambios de referencia. Estos programas han sido usados en continuas ocasiones haciendo así los ensayos comparables los unos con los otros ya que los cambios de setpoints eran los mismos. En este caso la referencia se encontraba en una variable %MW la cual estaba disponible en el OPC de modo que MATLAB® pudiera darle el valor que quisiera.

3.2.5 Filtro Paso Bajo

Tras el ensayo de la Figura 3.10 en el cual la válvula de salida estaba cerrada y la bomba apagada, se ha podido detectar la presencia de ruido en la señal de entrada al autómata. Este ruido es debido a la presencia magnética que produce el VFD, el cual se encuentra al lado del autómata y todos los cables adyacentes a él. Por lo cual se ha decido implementar un filtro de paso bajo que filtre la señal del sensor de presión.

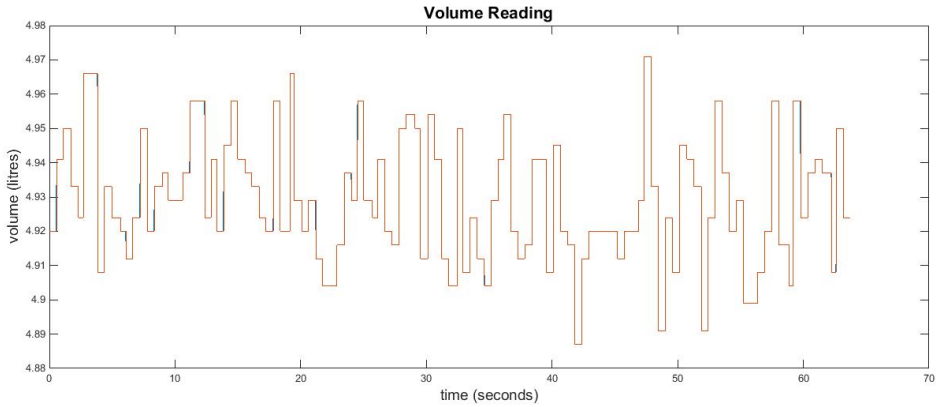

Figura 3.10 Ruido de la señal de entrada.

Un filtro de paso bajo corresponde a un filtro caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas, así todas las frecuencias se pueden presentar a la entrada, pero a la salida solo estarán presentes las que permita pasar el filtro. De la teoría se obtiene que un filtro de paso bajo de primer orden está caracterizado por la siguiente función de transferencia:

$$H(s) = k\frac{1}{1+\frac{s}{\omega_c}} \tag{3.2}$$

Donde la constante k es sólo una ponderación correspondiente a la ganancia del filtro, y ω_c corresponde a la frecuencia de corte propia del filtro, aquel valor de frecuencia para el cual la amplitud de la señal de entrada se atenúa 3 dB.

Para la programación del filtro en TIA PORTAL se ha implementado una formula discreta del filtro citado. Dicha formula es la siguiente, y su implementación puede verse en la Figura 3.11

$$y_{filtered}(t) = (1-k) * y(t) + k * y_{filtered}(t-1) \tag{3.3}$$

Donde "Volume1Real" es el volumen actual, "VolumeFiltered" es el volumen filtrado en el periodo anterior, "output" es el nuevo volumen actual filtrado y "filter k" la constante del filtro.

La constante del filtro K se ha elegido después de hacer varios ensayos del controlador en la planta. El controlador trabaja con el valor de la señal del volumen filtrado pero se analizan los resultados del volumen real, para no falsear los resultados. La elección del filtro se explica en el apartado 3.5.2.

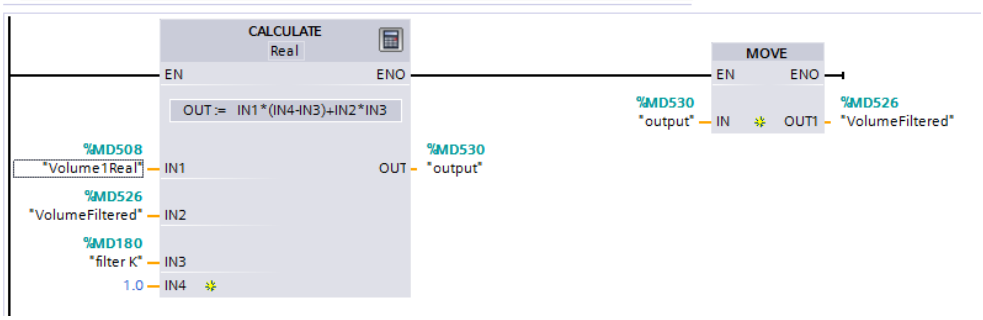

Figura 3.11 Filtro Paso Bajo TIA PORTAL.

3.3 Estudio de la linealidad del sistema

Para conocer la linealidad de una planta hace falta estudiar su dinámica. Se considera un sistema lineal cuando satisface el principio de superposición, que engloba las propiedades de proporcionalidad y aditividad. Que sea proporcional significa que cuando la entrada de un sistema es multiplicada por un factor, la salida del sistema también será multiplicada por el mismo factor. Por otro lado, que un sistema sea aditivo significa que si la entrada es el resultado de la suma de dos entradas, la salida será la resultante de la suma de las salidas que producirían cada una de esas entradas individualmente.

Para analizar la dinámica del sistema se ha realizado un ensayo en bucle abierto, donde la entrada se ha incrementado hasta en siete ocasiones con el mismo incremento, 0.5, Figura 3.12.

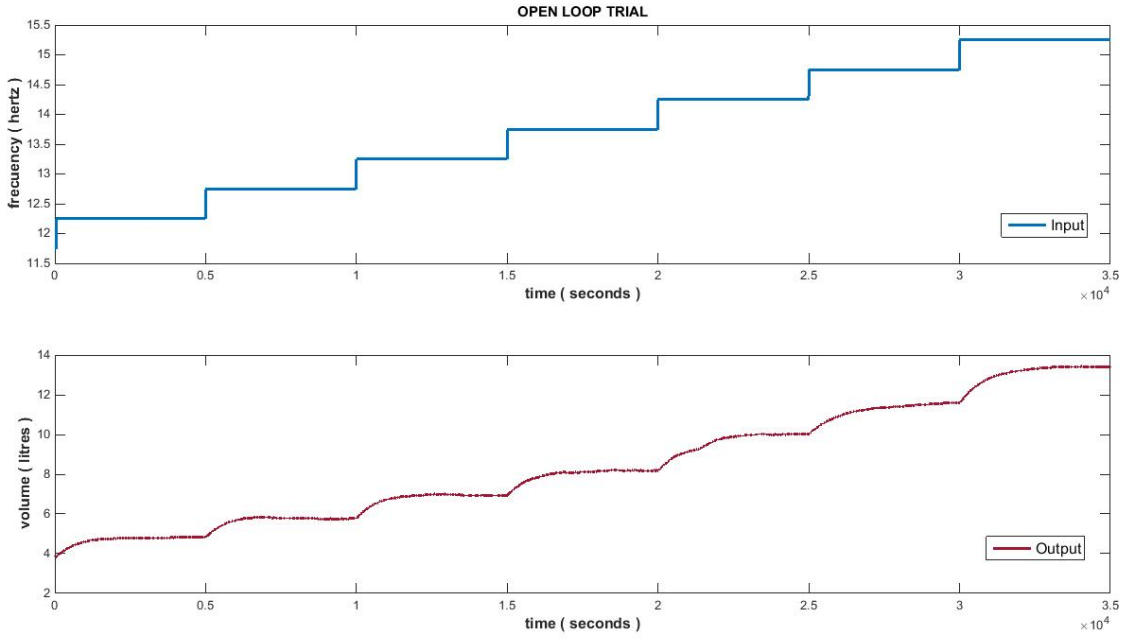

Figura 3.12 Ensayo en bucle abierto .

Dado el resultado de la prueba en bucle abierto puede decirse que el sistema no es lineal ya que para mismos incrementos en la entrada, la salida no varia igual.

$$\begin{aligned} Ganancia_{Step_2} &= \Delta y/\Delta u = 0.94/0.5 = 1.88 \\ Ganancia_{Step_6} &= \Delta y/\Delta u = 3.16/0.5 = 3.16 \end{aligned} \tag{3.4}$$

En segundo lugar se ha querido estudiar la causa de la no linealidad del sistema. La causa es debida a dos aspectos, Figura 3.13:

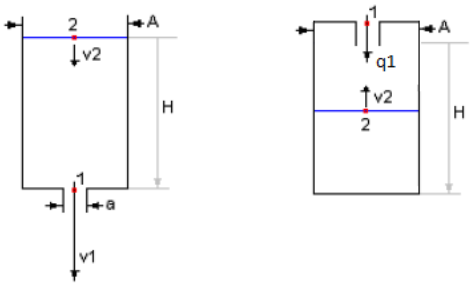

Figura 3.13 Esquema descarga depósito.

Donde a es el área de descarga, g es la gravedad, H es la altura del depósito, v1 y v2 son velocidades, A es la sección del depósito y q1 es el caudal de entrada de agua.

3.3.1 La dinámica de la salida de agua

Un depósito de descarga puede estudiarse como:

Se define la velocidad de salida del depósito como: $v_1 = \sqrt{2gh}[m/s]$. Por lo que el volumen del depósito queda definido como:

$$Volumen/A = v_2 = \frac{a\sqrt{2gh}}{A}[m/s] \tag{3.5}$$

Donde queda demostrado que el volumen del tanque no es linealmente dependiente de la salida de agua, identificando asì una de las causas de la no linealidad del sistema.

3.3.2 La dinámica de la entrada de agua

Para analizar la dinámica de la entrada del agua se deben estudiar dos conceptos:

1. Dinámica del depósito:

   La dinámica de la entrada de agua de un depósito como el de la Figura 3.13 se define como:

   $$Volumen/A = v_2 = \frac{q_1}{A}[m/s] \tag{3.6}$$

   Lo cual refleja que el volumen del tanque es linealmente dependiente del caudal de entrada. En este caso la entrada de agua no afectaría a la linealidad del Sistema.

   Pero la variable de entrada al sistema no es el caudal, si no la frecuencia de la bomba, por lo que se ha hecho un segundo estudio para analizar si la relación frecuencia - caudal es lineal o no.

2. Dinámica de la bomba:

   La entrada de agua al tanque no está siendo medida en caudal, si no que la variable a controlar es la frecuencia de la bomba. Para estudiar la dinámica de esta relación frecuencia - caudal de entrada, se ha hecho uso de un caudalímetro.

   A continuación se muestran los resultados de un ensayo en bucle abierto que se ha realizado para analizar la relación citada, Figura 3.14.

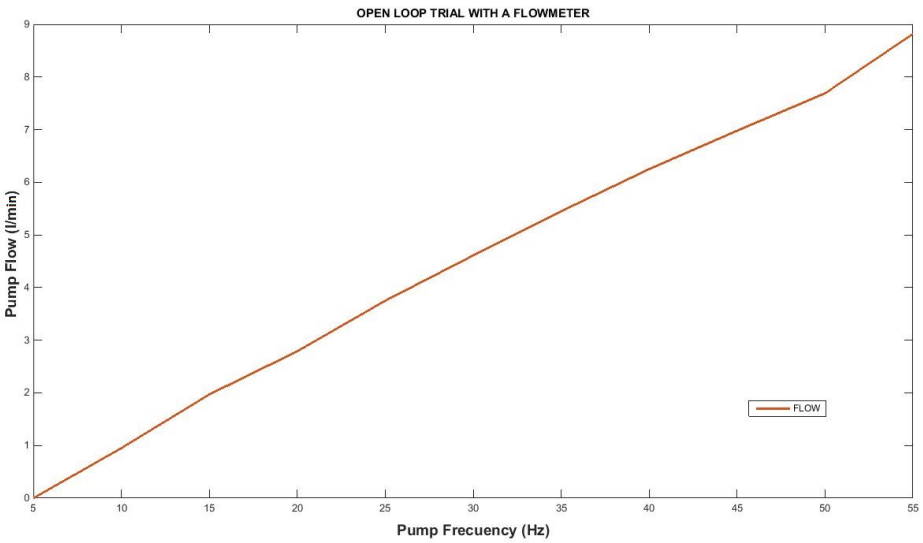

Figura 3.14 Ensayo en bucle abierto del caudal de entrada.

Analizando los resultados se puede afirmar que la relación es levemente no lineal y que por lo tanto la entrada de agua también es causa de la no linealidad de la planta. Aun así la no linealidad de esta relación es muy leve, y prácticamente lineal en la mayor parte del rango de trabajo. Por lo que se ha despreciado y no se ha tenido en cuenta en el diseño del controlador con Gain-Scheduling para corregir las no linealidades del sistema.

3.4 Modelado del sistema

Existen variadas maneras de modelar un sistema. En este proyecto la planta se ha modelado bajo ensayos en bucle abierto. Esta opción puede definirse como uno de los tipos de modelaje más universal para modelar sistemas, ya que no necesita saber de la física de la planta o sus equipos. La idea es la de modelar el sistema completo analizando ensayos en bucle abierto. Caracterizando exclusivamente la entrada y la salida del sistema.

Existen otras técnicas de modelaje de sistemas. Como por ejemplo el estudio de la física de la planta y su posterior linealización de las ecuaciones reales.

En adelante todos los ensayos que se han realizado se han hecho definiendo la variable de la válvula de salida como un parámetro fijo:

$$\text{Apertura de la válvula de salida}=73\,\%$$

Para modelar la dinámica de la planta se ha utilizado el ensayo de la Figura 3.15:

Una vez hecho el ensayo se ha identificado la dinámica utilizado la herramienta de MATLAB® : **System Identification Toolbox**

- Debido a que el sistema es no lineal como se ha estudiado en el apartado anterior es importante tener en cuenta que un solo modelo, que no es variable según el punto de trabajo, no es lo suficientemente bueno y real.
- Aun con esta particularidad, el sistema es levemente no lineal y bajo la hipótesis de que sí lo fuera los resultados no son malos. Por lo cual, en primer lugar, solo un modelo ha sido necesario. Éste ha sido basado en el escalón intermedio del ensayo en bucle abierto.
- Como segundo objetivo del proyecto, la no linealidad si se tiene en cuenta. La idea se basa en el uso de diferentes modelos según el punto de trabajo, por lo cual se va a hacer uso de varios

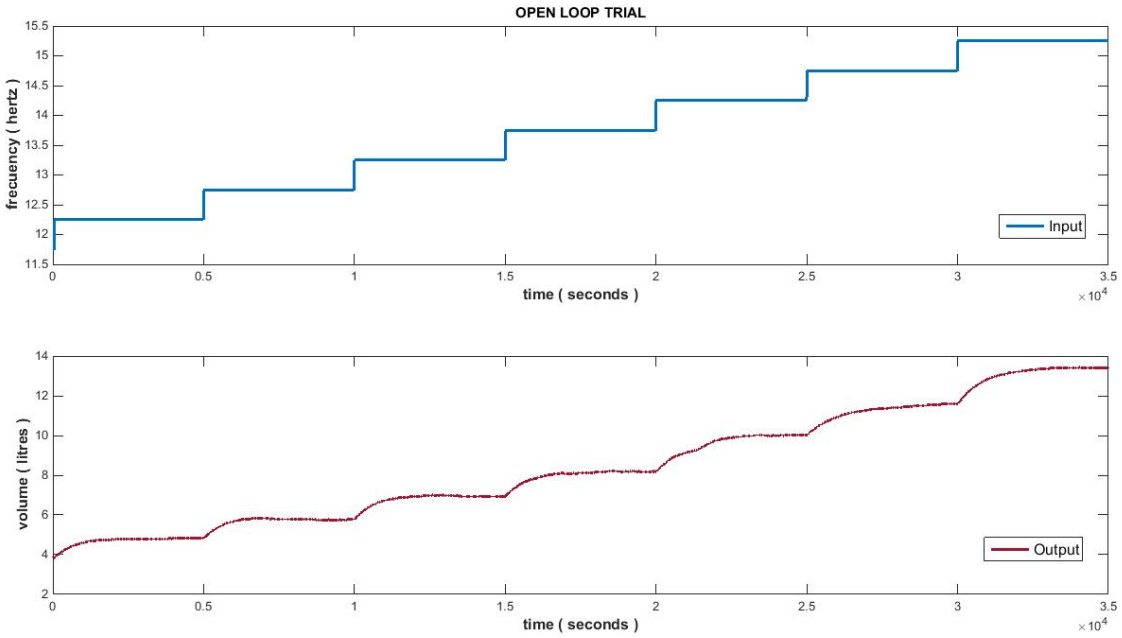

Figura 3.15 Ensayo en bucle abierto .

modelos. Para obtenerlos tan solo hace falta obtener diversos modelos de la herramienta de MATLAB® basados en diferentes escalón del ensayo en bucle abierto.

3.4.1 System Identification Toolbox

Esta herramienta es muy útil para modelar todo tipo de sistemas. Lo único que necesita es la suficiente información para modelar. Para ello se importan los datos del ensayo en bucle abierto. Se recomienda seguir los siguientes pasos para su uso:

1. Importar los datos de un ensayo en bucle abierto. Es recomendable importar un solo escalón donde partan de cero tanto la entrada como la salida. Por lo cual se debe restar el offset para obtener un cero en ambas variables al inicio del escalón.
2. Hay que elegir el tipo de modelo que se quiere obtener a partir de los datos. Puede elegirse entre modelos en función de transferencia, espacio de estados, etc. En el caso de la planta se ha escogido un modelo en función de transferencia.
3. En el caso de modelos en función de transferencia se deben definir el número de polos y ceros de éste. Para la planta, la elección del número de polos y ceros se ha definido realizando varias pruebas y se ha elegido un sistema con dos polos debido a que era el modelo que mejor se acopla al sistema, Figura 3.16.
4. Automáticamente la herramienta da el mejor modelo aproximado posible dado las características previas definidas de este, Figura 3.17.

El modelo obtenido que ha sido usado para la creación del controlador GPC es el siguiente:

- Modelo definido con el cuarto escalón del ensayo en bucle abierto de la Figura 3.15.

$$G_2(s) = \frac{0.01152}{s^2 + 3.507s + 0.004685} \tag{3.7}$$

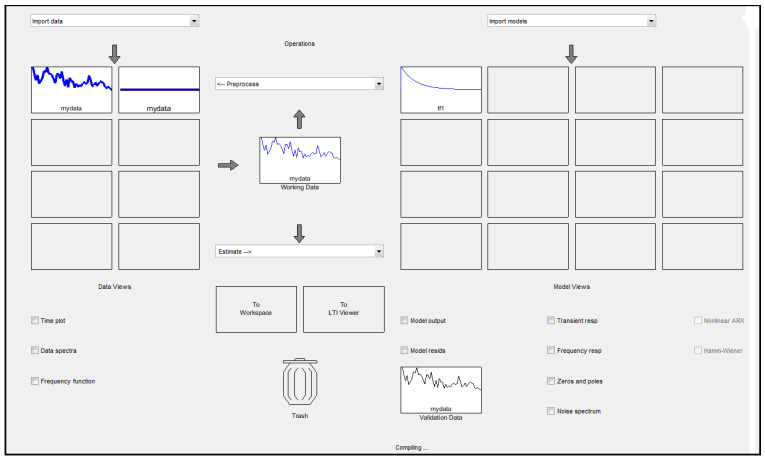

Figura 3.16 Ident Toolbox .

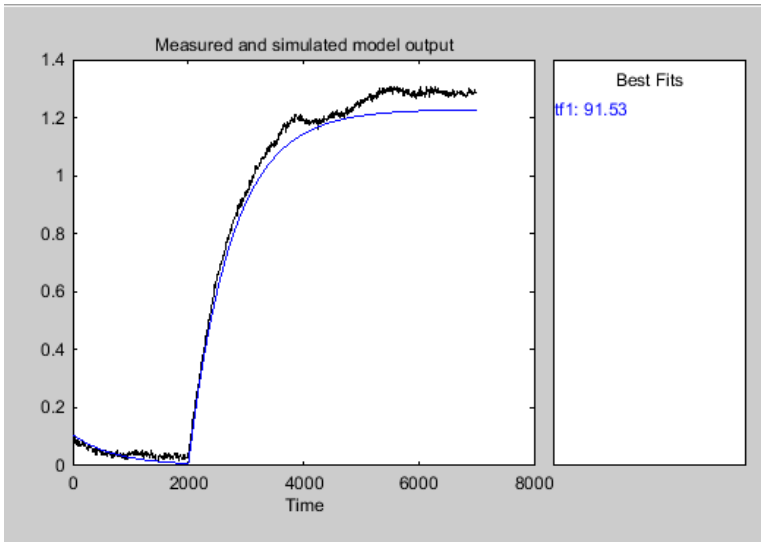

Figura 3.17 Result Ident Toolbox .

3.5 Controlador GPC

El objetivo de este proyecto es el de usar un control predictivo generalizado GPC para controlar el volumen de un tanque de agua a partir de la frecuencia del motor que bombea agua al tanque.
Como se ha explicado en el apartado 2.2.1 para crear un controlador GPC se necesita un modelo en función de trasferencia que modele el sistema, el cual ha sido definido como [3] [7]:

$$(s^2 + 3.507s + 0.004685)y(s) = (0.01152)u(s) \tag{3.8}$$

Tan bien una función de coste es necesaria para crear el controlador. La ecuación usada en el proyecto es la siguiente:

$$J(N_1,N_2,N_c) = \sum_{j=N_1}^{N_2} \delta(t)[\hat{y}(t+j|t) - ref(t+j)]^2 + \sum_{j=0}^{N_u-1} \lambda(t)[\Delta u(t+j)]^2 \tag{3.9}$$

En la cual $\delta(t) = \lambda_2$ y $\lambda(t) = \lambda$ se han definido constantes en el tiempo, pero debido a que no se tiene ningún requerimiento previo, se han dejado como variables a fijar posteriormente.

3.5.1 Restricciones

Las limitaciones existentes en el sistema juegan un papel importante en el controlador, por lo que deben tenerse en cuenta. Las restricciones existentes son las siguientes:

1. Volumen:
   El volumen máximo de trabajo se ha fijado en 15 litros siendo el mínimo 0 litros.

$$y_{min} = 0 \quad \leq \quad y(t) \quad \leq \quad 15 = y_{max} \quad \forall t$$

2. Frecuencia:
   Respecto a la bomba, para frecuencias menores de 6.6 Hercios ésta no bombea agua, fijándose así 6.6 Hercios como el valor mínimo de la frecuencia y el máximo se ha fijado en 75 Hercios ya que para valores mayores apenas se aprecia un incremento de caudal.

$$u_{min} = 6.6 \quad \leq \quad u(t) \quad \leq \quad 75 = u_{max} \quad \forall t$$

Para añadir las restricciones al problema de optimización se deben definir todas las restricciones de la siguiente forma:

$$R\mathbf{u} \leq c \tag{3.10}$$

Teniendo en cuenta la ecuación de predicción (2.19) y la secuencia de actuaciones **u** las restricciones pueden reescribirse como [8]:

$$\begin{array}{lccl} \bar{U}_{min} \leq & T\mathbf{u}(t) + \bar{1}^{N_c} u(t-1) & \leq \bar{U}_{max} & \forall t \\ \bar{y}_{min} \leq & G\mathbf{u} + f & \leq \bar{y}_{max} & \forall t \end{array} \tag{3.11}$$

Quedan definidas las restricciones según la ecuación (3.10) como:

$$\begin{bmatrix} +G \\ -G \\ +T \\ -T \end{bmatrix} \bar{u} \leq \begin{bmatrix} \bar{y}_{max} - F_x \bar{x} \\ -\bar{y}_{min} + F_x \bar{x} \\ \bar{U}_{max} - \bar{1}^{N_c} u(t-1) \\ -\bar{U}_{min} + \bar{1}^{N_c} u(t-1) \end{bmatrix} \tag{3.12}$$

con $T \in \Re^{N_c \times N_c}$, $\bar{y}_{max} \in \Re^{N}$, $\bar{y}_{min} \in \Re^{N}$, $\bar{U}_{max} \in \Re^{N_c}$ y $\bar{U}_{min} \in \Re^{N_c}$

Haciendo uso de MATLAB® se ha desarrollado un programa, que construye el problema de optimización tal y como se plantea en el apartado 2.2.3 y tiene la siguiente forma:

$$\begin{aligned} \min J &= \frac{1}{2} \bar{u}^T H \bar{u} + b\bar{u} \\ \text{sujeto a} \quad & Ru \leq c \end{aligned} \tag{3.13}$$

Para el funcionamiento de este código, descrito a continuación Código 3.2 , tan solo es necesario definir previamente:

-El modelo en función de transferencia. G(s)
-El tiempo de muestreo Ts
-Las constantes de ponderación λ y λ_2
-Los horizontes de predicción de control N y N_c

Código 3.2 Código MATLAB® : Código para la creación del problema de optimización QP.

```
% G(s) Model System
num=0.01152;
den=[1 3.507 0.004685];
sys=tf(num,den);


%Parámetros:
lambda=0.005; % DU COST CONSTANT
lambda2=50;  % ERROR COST CONSTANT
N=20;        % PREDICTION HORIZONT
Nc=20;       % CONTROL HORIZONT
Ts=1;        % Sample time: IMPORTANT! DISCRET TIME AND CONTROL TIME

sysd = c2d(sys,Ts,'zoh');
[fil,col]=size(sysd.num{1});
B=sysd.num{1}(2:col);
A=sysd.den{1};
nb=length(B)-1;
na=length(A)-1;


%Calculation of Fj
Avir=conv(A,[1 -1]);
F=zeros(N,na+1);
poluno=[1 zeros(1,na+1)];
F1aux=poluno-Avir;
F(1,:)=F1aux(1,2:na+2);
for j=1:(N-1)
    for i=0:na-1
        F(j+1,i+1)=F(j,i+2)-F(j,1)*Avir(1,i+2);
    end
    F(j+1,na+1)=-F(j,1)*Avir(1,na+2);
end

%Calculation of Ej
E=zeros(N,N);
E(1,1)=1;
for j=1:(N-1)
    E(j+1,:)=E(j,:);
    E(j+1,j+1)=F(j,1);
end

%Calculation of Gj=Ej*Denj
Gpols=zeros(N,N+nb);
for j=1:N
    Gpols(j,:)=conv(E(j,:),B);
end

%Calculation of G Matrix
G2=zeros(N,N);
for j=1:N
    kk=1;
    for i=j:-1:1
        G2(j,kk)=Gpols(j,i);
```

```
        kk=kk+1;
    end
end
G=G2(:,1:Nc);

%Calculation of G' Matrix
Gprima=[];
if nb>0
    Gprima=zeros(N,nb);
    for j=1:N
        Gprima(j,:)=Gpols(j,1+j:(j+nb));
    end
end

H=2*(G'*(lambda2*eye(N))*G+lambda*eye(Nc));

%Calculated each sample time
x=[y(k:-1:k-na);du(k-1:-1:k-nb)];
Fx=[F';Gprima']';
f=Fx*x;
b=2*lambda2*(f-ref(k:k+N-1))'*G;
```

Una vez planteado el problema a resolver, la solución de éste se ha obtenido, también usando MATLAB® , como se describe en el Código 3.3

Código 3.3 Código MATLAB® : Código para el cálculo de la secuencia de actuaciones u en el instante k.

```
% Unconstraint problem:
 [duk]=-inv(H)*b';
 du(k)=duk(1);
 u(k)=u(k-1)+du(k);

% Constraint problem:
 c1(1:Nc,1)=1;
 I=eye(Nc);
 T=tril(ones(Nc),0);
 R=[I;-I;T;-T;G;-G];

 c=[dumax;-dumin;umax-c1*u(k-1);-umin+c1*u(k-1);ymax-f;-ymin+f];
 [duk,min,flag]=quadprog(H,b,R,c);
 du(k)=duk(1);
 u(k)=u(k-1)+du(k);
```

3.5.2 Elección de parámetros

1. Sample Time:

   El tiempo de muestreo o sample time juega un papel doble en el controlador. En primer lugar el tiempo de muestreo es el tiempo que se utiliza para discretizar el modelo de función de transferencia. Esto conlleva, en segundo lugar, que dicho tiempo sea el tiempo de actuación del controlador.

   Como regla general un modelo discreto es suficientemente bueno usando un sample time menor o igual que la décima parte de la constante de tiempo de la planta. Analizando el ensayo en bucle

abierto de la Figura 3.12 se calcula que la constante de tiempo ronda alrededor de 700 segundos. Lo que supone que un sample time de 70 segundos sería el mayor de los sample time que se deben usar.

Varios ensayos se han realizado para ver como afecta al controlador una modificación del tiempo de muestreo, Figura 3.18 y Figura 3.19.

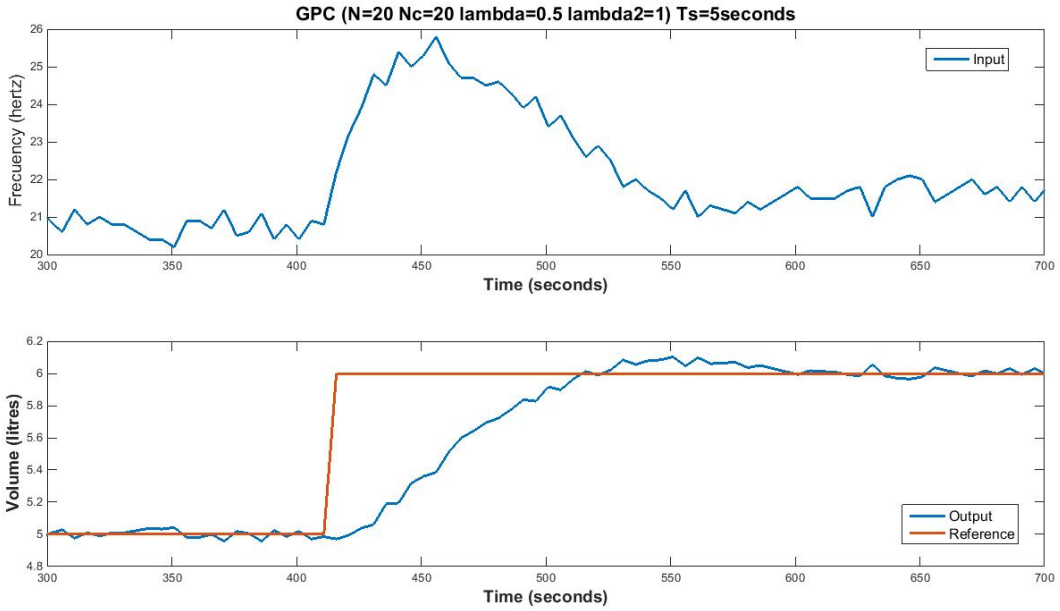

Figura 3.18 Ensayo GPC con $Ts = 5$segundos .

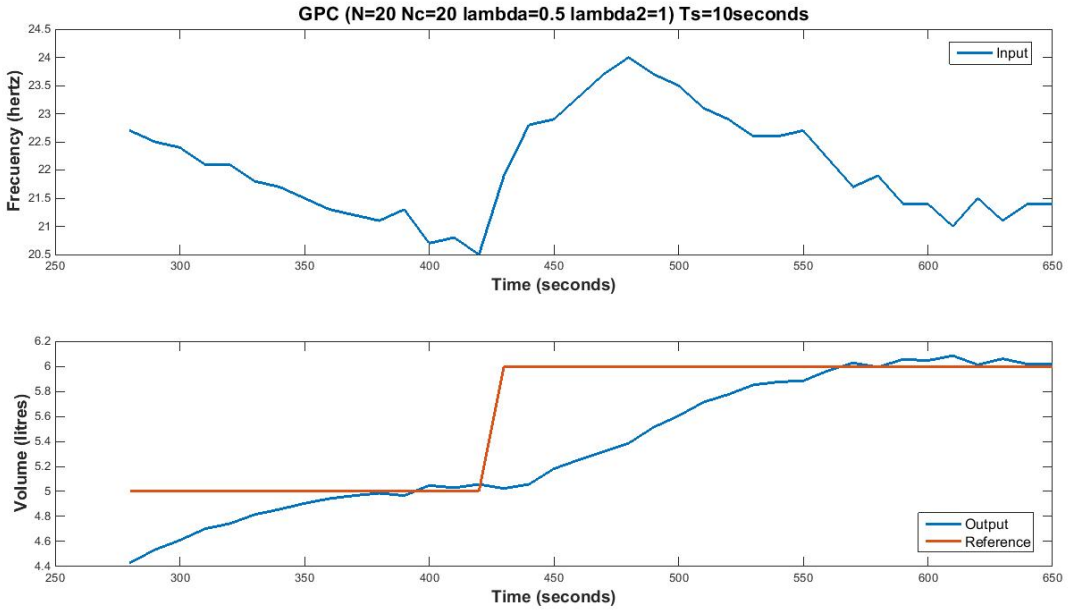

Figura 3.19 Ensayo GPC con $Ts = 10$segundos .

Analizando los resultados se comprueba que cuanto mayor es el tiempo de muestreo, más lenta y pero más suave es la respuesta.

2. Horizonte de predicción y horizonte de control N y N_c:

Para la elección del horizonte de predicción y de control también se han realizado varios ensayos para analizar como afecta al controlador, Figura 3.21 y Figura 3.20.

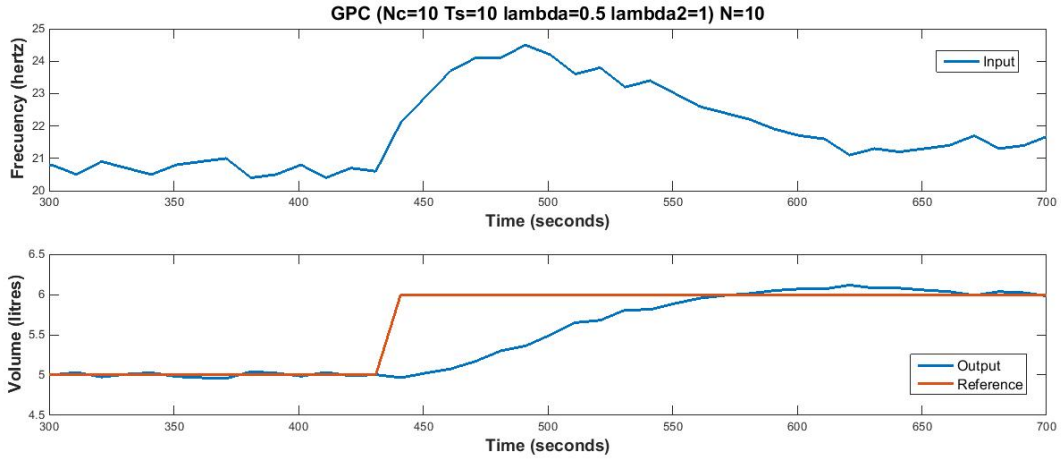

Figura 3.20 Ensayo GPC con $N = 10$.

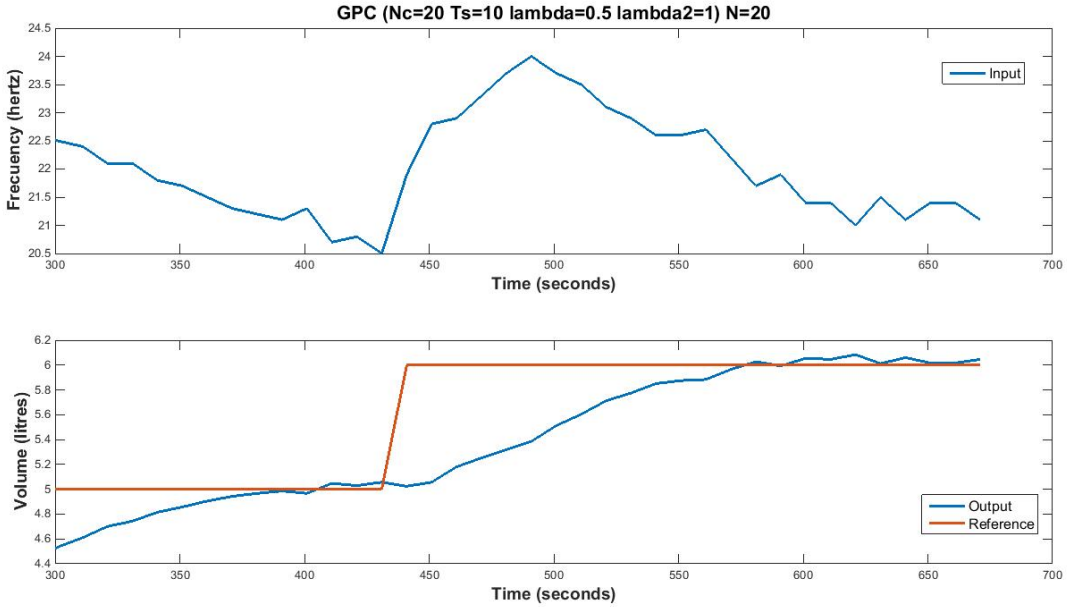

Figura 3.21 Ensayo GPC con $N = 20$.

Analizando los resultados se comprueba que cuanto mayor sea el horizonte de predicción mejor, ya que la sobreoscilación es mejor y es menos agresivo.

Aun así la elección de estos dos parámetros se ven limitados por el tiempo de muestreo. Esto se debe a que cuanto mayor son los horizontes, sobre todo el de control, mayores son los cálculos

que deben resolverse en el problema de optimización, lo que hace que el tiempo de la resolución de éste aumente. El tiempo de muestreo debe ser siempre mayor que el tiempo que tome el algoritmo en obtener la secuencia de acciones de control óptimas, Relación 1 Figura 3.22.

Por otro lado si se requiere que el horizonte tenga una dimensión de X segundos, cuanto menor sea el tiempo de muestreo mayor sera el valor de N, Relación 2 Figura 3.22.

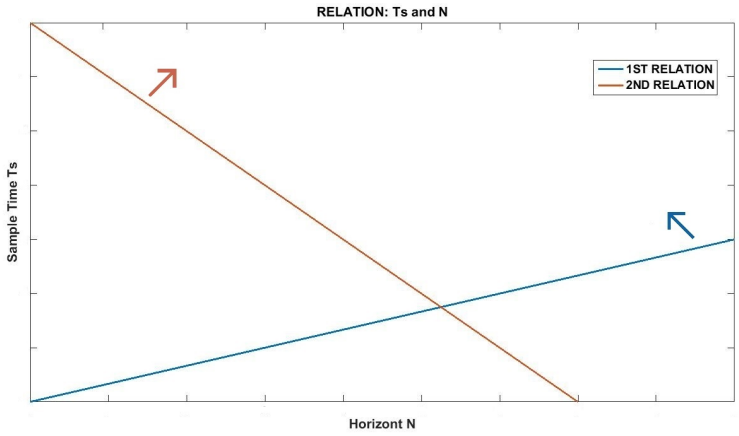

Figura 3.22 Relación N y Ts .

Por lo tanto se debe buscar un punto óptimo teniendo en cuenta estas dos situaciones.

3. Constantes de Ponderación λ_1 y λ_2:

Las constantes de ponderación suelen estar previamente fijadas, según sea el requisito real de la planta. En el proyecto como no están fijadas, se han usado como parámetros a escoger. Analizando la función de coste, al ser un problema con solo dos variables, lo que importa es la relación que haya entre ellas, pudiendo dejar una de ellas fija siempre. Varios ensayos se han realizado para ver como afecta al controlador una modificación de las constantes de ponderación, Figura 3.23 y Figura 3.24.

Analizando los resultados de los ensayos de la Figura 3.23 y la Figura 3.24 queda demostrado, como se esperaba, que cuanto menor sea λ y por lo tanto menor sea la relación λ/λ_2, más agresivo y más rápido es el controlador. Esto se debe a que λ es la constante que pondera el esfuerzo de la acción de control. Cuando menos se pondere al esfuerzo, más agresivo va a poder ser la acción de control y por lo tanto más rápida.

4. Constante de Filtro K:

Tal y como se explicó en el apartado 3.2.5, un filtro de paso bajo ha sido instalado en el autómata. Para la elección de su constante K se han realizado también varios ensayos, para analizar como afecta el filtro al controlador GPC. Los resultados han sido los siguientes, Figura 3.25, Figura 3.26 y Figura 3.27.

Cuando no se aplica ningún filtro, aparecen oscilaciones en régimen permanente debido a la existencia de ruido. El problema es que el controlador intenta corregir el ruido, sin éxito, y eso implica una oscilación permanente como se puede ver en la Figura 3.25.

Analizando los resultados de la Figura 3.26 y la Figura 3.27 se comprueba que cuanto mayor es el filtro mayor es la sobreoscilación ante un cambio de referencia.

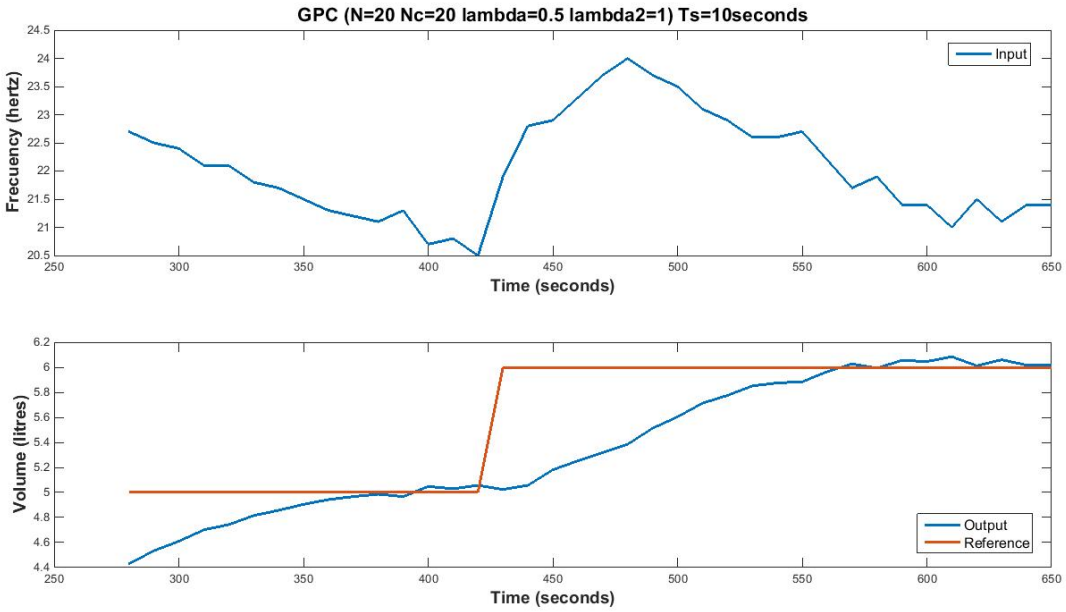

Figura 3.23 Ensayo GPC con $\lambda = 0.5$.

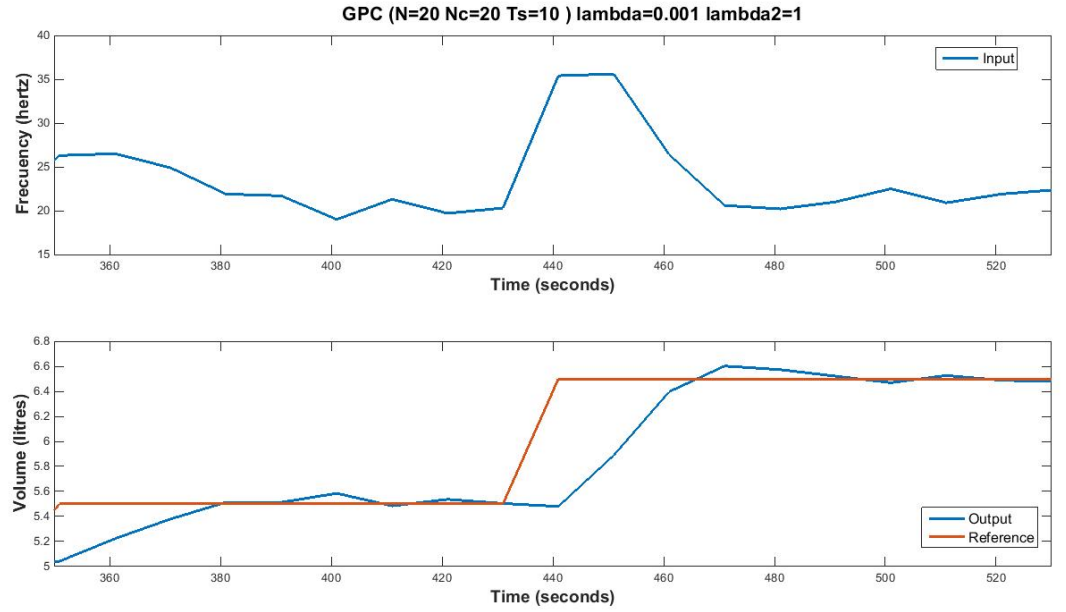

Figura 3.24 Ensayo GPC con $\lambda = 0.001$.

Por lo tanto se debe buscar un filtro que elimine el ruido en régimen permanente pero que no sea demasiado agresivo como para crear una gran sobreoscilación.

La constante de filtro tras estos ensayos escogida ha sido la siguiente y los resultados son los mostrados en la Figura 3.28 :

$$K_{filter} = 0.8$$

Una vez analizados todos los parámetros y conociendo como afectan al controlador, dichos parámetros pueden ser elegidos como se prefieran, es decir, en función de los requerimientos que se tengan. E

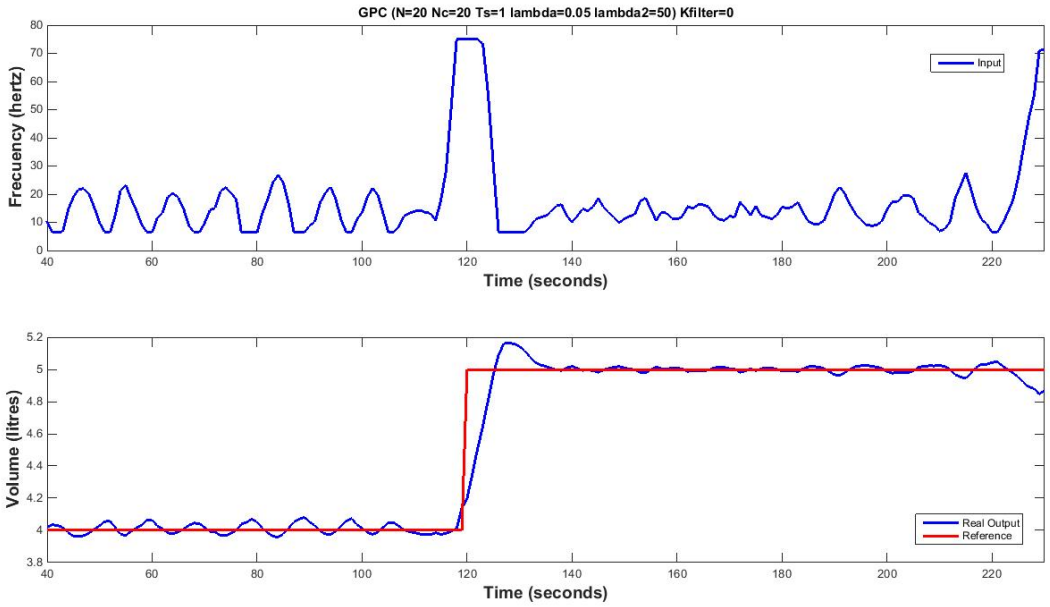

Figura 3.25 Ensayo GPC sin Filtro .

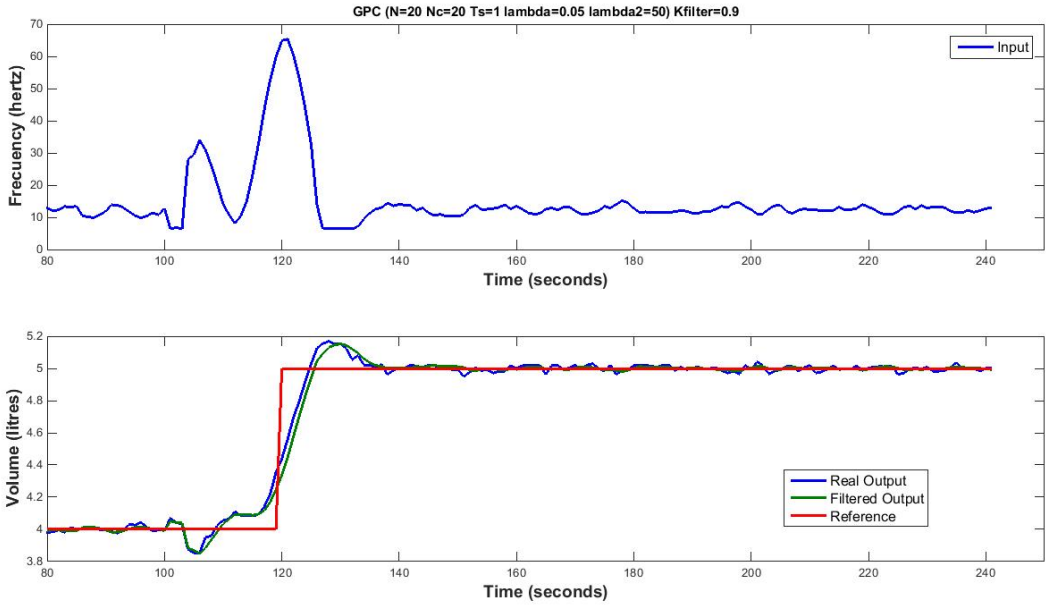

Figura 3.26 Ensayo GPC con Filtro k=0.9 .

si se quiere una respuesta rápida o una acción de control suave o se debe tener una sobreoscilación $\leq 5\,\%$Cambio de referencia.

- El tiempo de muestreo elegido ha sido 1 segundo, para utilizar el mismo que el fijado en el controlador PI.
- N y N_c han sido fijados en 20, ya que aumentándolos más el tiempo de cálculo sobrepasaba 1 segundo.
- Los valores de λ y λ_2 han sido fijados para obtener una respuesta más rápida que la del PI, $\lambda = 0.1$ y $\lambda_2 = 50$.
- La constante de filtro se ha fijado en 0.8 como se ha explicado antes, filtrando así el ruido pero sin perturbar demasiado la realidad.

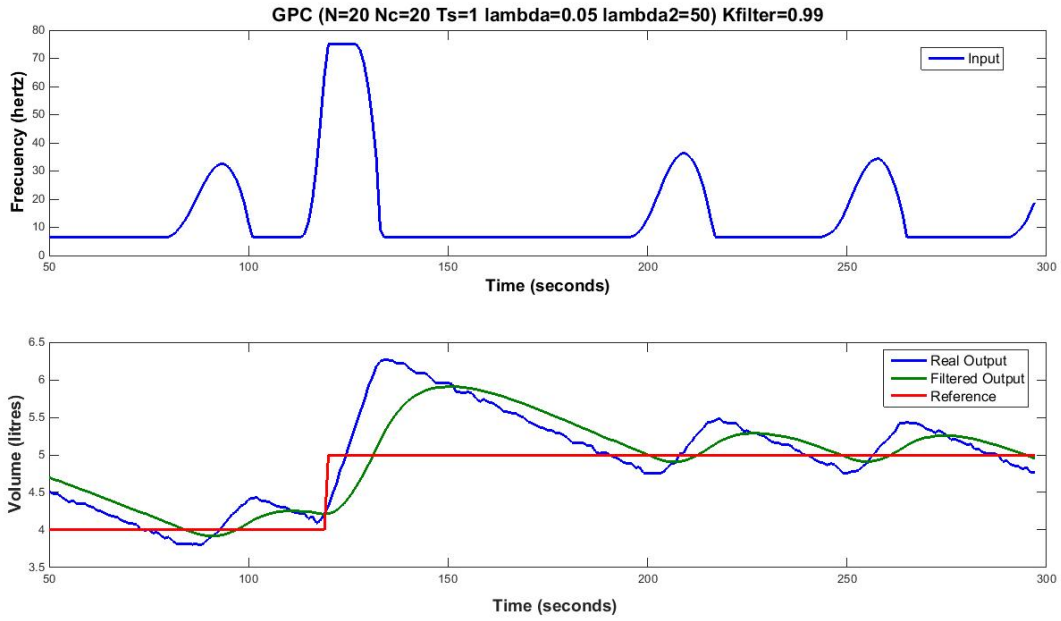

Figura 3.27 Ensayo GPC con Filtro k=0.99 .

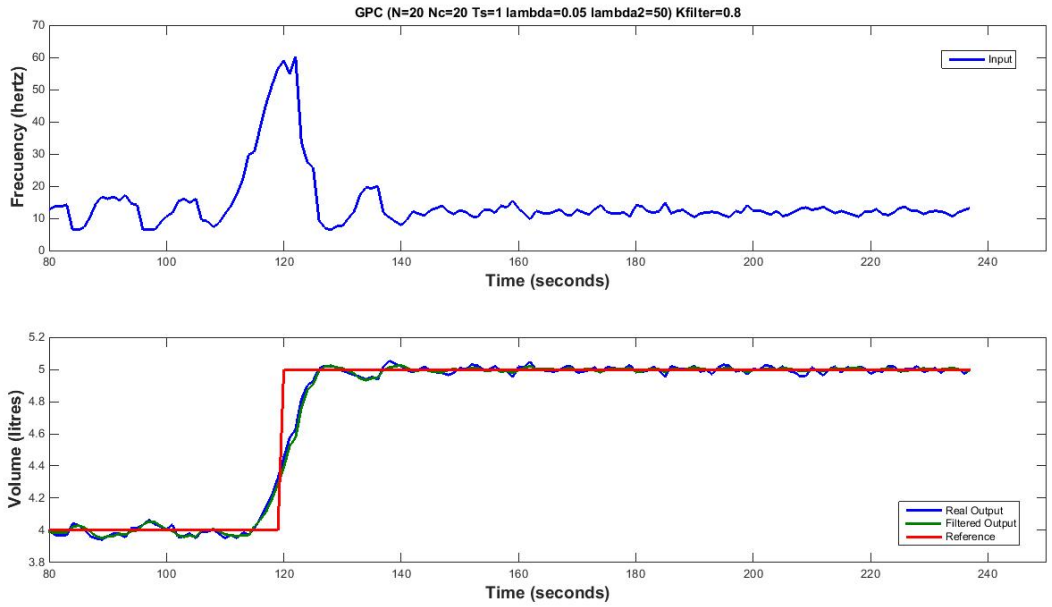

Figura 3.28 Ensayo GPC con Filtro k=0.8 .

Estos son los parámetros elegidos en el proyecto para el resto de ensayos, pero es importante decir que el GPC con restricciones creado tiene todos estos parámetros libres para que sean tuneados según se requiera.

Con los parámetros fijados como se han dicho previamente se ha realizado el ensayo de la Figura 3.29 donde queda demostrado la mejora del controlador frente al PI.

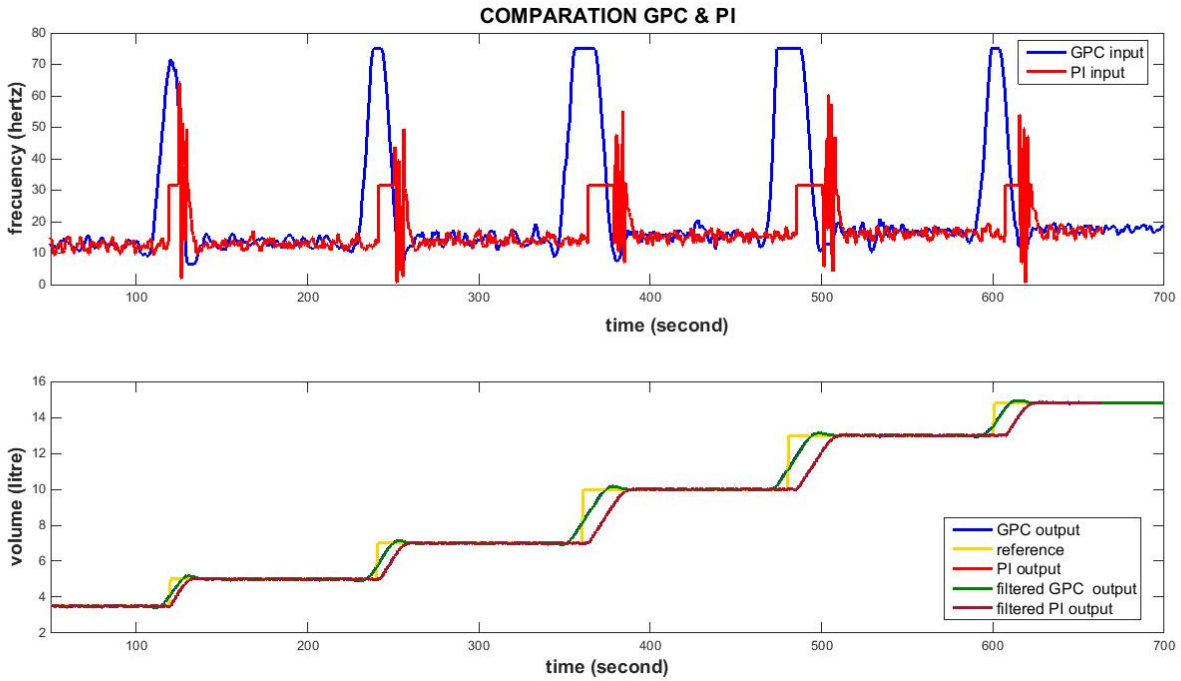

Figura 3.29 Ensayo comparación GPC y PI.

3.6 Control GPC + Gain-Schudeling

La introducción de la estrategia Gain-Scheduling es debida a la presencia de no linealidades en el sistema. Cuando se encuentra un sistema con ganancias y constantes de tiempo variables esta estrategia facilita variar el valor de las constantes del modelo del proceso, acoplándose así mejor a la realidad y dando mejores resultados.
La no linealidad del proceso viene dada por dos causas, una de ellas despreciable como se ha analizado el apartado 3.4 , y otra a tener en cuenta. De modo que se han identificado un conjunto de modelos del sistema en función del punto de funcionamiento de la característica de su no linealidad: el volumen del tanque.
Haciendo uso de la herramienta de MATLAB® Identify Toolbox la cual se ha explicado su funcionamiento en el apartado de modelado de sistemas 3.4.1 se han definido tres modelos del sistema, cada uno en un punto de funcionamiento diferente.

1. Modelo definido con el segundo escalón del ensayo en bucle abierto, Figura 3.15. Éste será usado para cualquier punto de trabajo cuyo volumen sea inferior a seis litros.

$$G_1(s) = \frac{0.0088704}{s^2 + 3.498s + 0.004679}$$

2. Modelo definido con el cuarto escalón del ensayo en bucle abierto, Figura 3.15. Éste será usado para cualquier punto de trabajo cuyo volumen sea superior a seis litros e inferior a ocho litros.

$$G_2(s) = \frac{0.01152}{s^2 + 3.507s + 0.004685}$$

3. Modelo definido con el sexto escalón del ensayo en bucle abierto, Figura 3.15. Éste será usado para cualquier punto de trabajo cuyo volumen sea superior a ocho litros.

$$G_3(s) = \frac{0.002274}{s^2 + 0.8613s + 0.0007163}$$

Las características del modelo no varían mucho por lo que tan solo con tres modelos, va a ser suficiente y a la vez se verá mejoría en los resultados del controlador ya que cada modelo se acopla mejor al proceso en cada punto de trabajo.

3.6.1 Implementación

La implementación de la estrategia Gain-Scheduling al controlador GPC se ha desarrollado como se muestra en Código 3.4. El código forma parte del bucle que se realiza cada tiempo de muestreo, cada segundo, de forma que en primer lugar se comprueba en que punto de trabajo se ha quedado el sistema en el instante anterior y(na) y, en función de éste, fija el modelo definiendo la función de transferencia correspondiente y creando el controlador. Para crearlo se activa la función "Values" en la que se encuentra programado la construcción del GPC como se describió en el Código 3.2.

Código 3.4 Código MATLAB® : Bucle del controlador GPC con Gain Scheduling.

```
if y(na)<6
num=0.0088704; den=[1 3.498 0.004679];
Values
modelo(time)=1;
end

if y(na)>=6 && y(na)<8
num=0.01152; den=[1 3.507 0.004685];
Values
modelo(time)=2;
end

if y(na)>=8
num=0.002274; den=[1 0.8613 0.0007163];
Values
modelo(time)=3;
end

% Free response;
x=[du(nb+1:-1:2);y(na+1:-1:1)];
Fx=[Gprima';F']';
f=Fx*x;

b=2*lambda2*(f-ref(time:time+N-1))'*G;
c=[dumax;-dumin;umax-c1*u(nb+1);-umin+c1*u(nb+1);ymax-f;-ymin+f];
[duk,min,flag]=quadprog(H,b,R,c);
```

Los resultados de la implantación de la estrategia Gain- Scheduling son los mostrados en la Figura 3.30, Figura 3.31 y Figura 3.32.

Analizando el resultado del control GPC con Gain-Scheduling en la Figura 3.31 y la Figura 3.32, se puede ver al comparar con los resultados del GPC sin Gain-Scheduling que para puntos de trabajo en los que el volumen es menor que 6 litros o en los que es mayor que 8 litros el control con G-S se acopla mejor y da resultados mejores.

Ya que los resultados han sido los esperados, queda demostrado que utilizar una estrategia Gain-Scheduling ayuda al controlador GPC a dar mejores resultados.

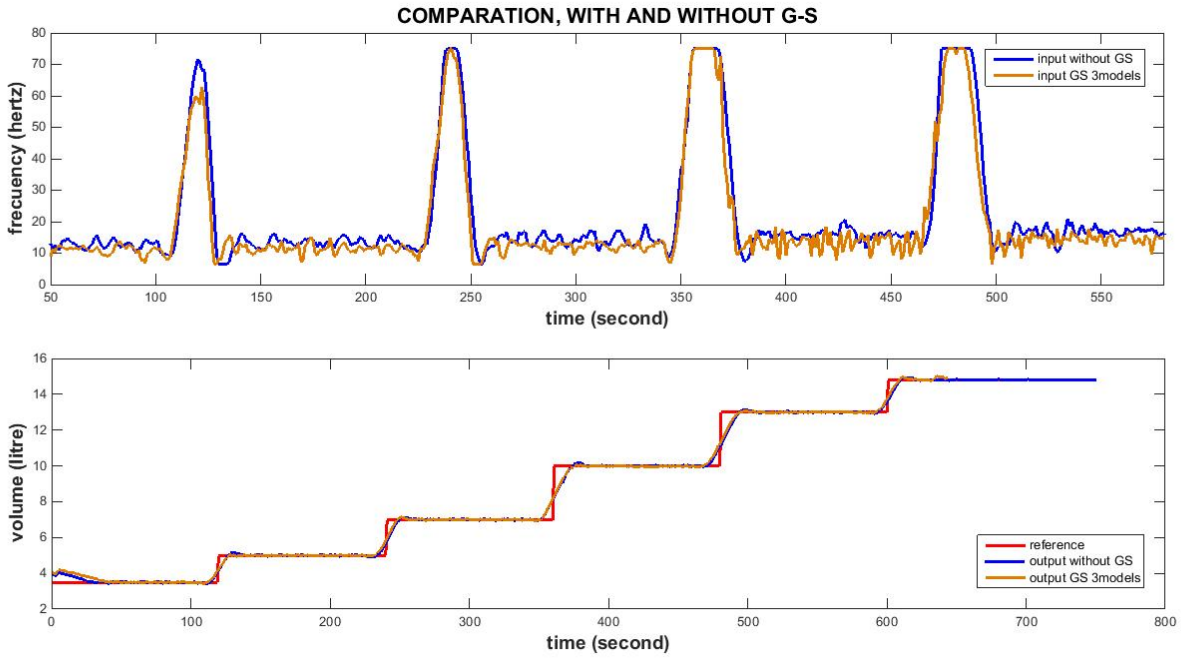

Figura 3.30 Ensayo GPC con Gain-Scheduling .

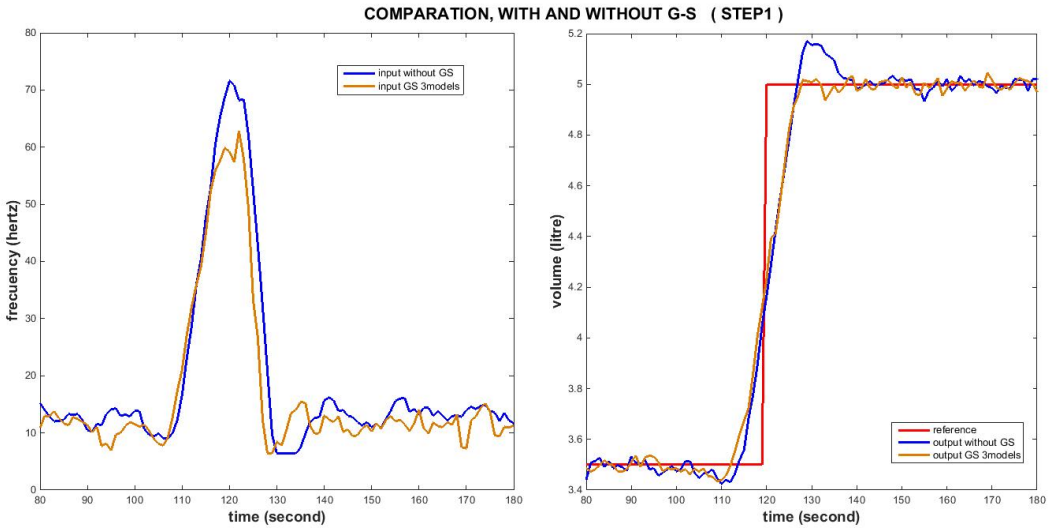

Figura 3.31 Ampliación Escalón 1 Figura 3.30 .

3.7 Implementación OFFLINE del GPC

Para finalizar este proyecto se ha querido plantear una solución de control offline. La idea es la de implementar el controlador GPC con restricciones dentro del autómata y así tener una solución autónoma.

Debido a la carga computacional que conlleva el algoritmo de resolución del problema QP, el GPC no puede programarse en el PLC ya que éste no dispone de las funciones necesarias para hacer los cálculos, por lo que se necesita un programa complementario, en este caso MATLAB® , para el cálculo de la señal de control. Ésta es la razón por la que se hace necesario un ordenador externo donde implementar

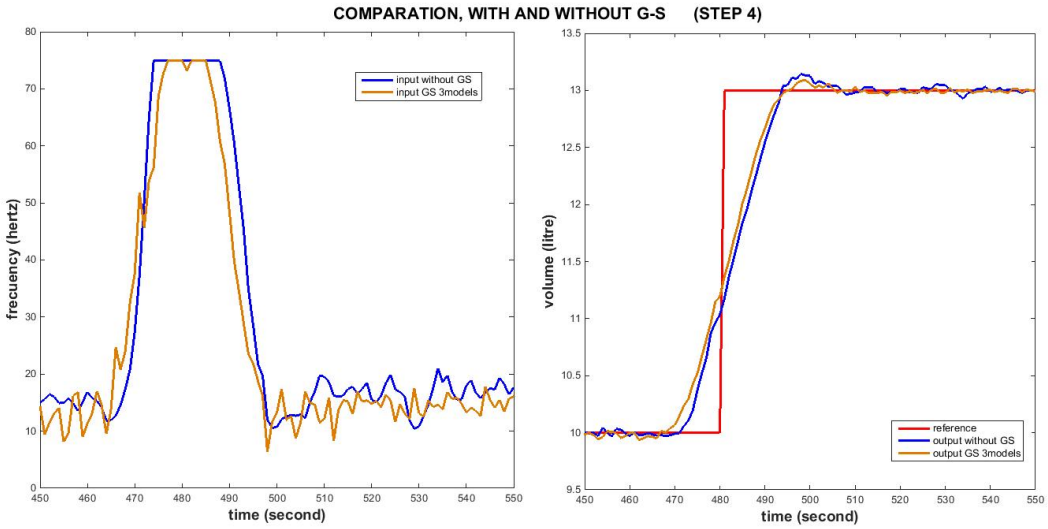

Figura 3.32 Ampliación Escalón 4 Figura 3.30 .

dicho programa.

El algoritmo del controlador tal y como está definido ahora no puede ser implementado en el autómata sin el uso de una computadora externa.

Una solución multiparamétrica del controlador es la solución al problema previamente citado. La Programación cuadrática multiparamétrica, en adelante MPQP, es un medio alternativo para la implementación de algoritmos de control predictivo por la cual se reduce en gran parte la carga computacional haciendo así realizable el control implementado en el PLC.

En los últimos años la estrategia de plantear el problema como un MPQP [5] se ha convertido en una herramienta útil para la caracterización explícita del algoritmo de control predictivo GPC. La eficacia de este enfoque es debido a dos principales factores: la aplicación offline se reduce a una tabla de consulta la cual hace un código sencillo y, a su vez, la estructura de la estrategia de control es transparente, a diferencia que cuando se utiliza una optimización online.

3.7.1 Formulación MPQP

Un problema MPQP general se plantea como un problema de minimización de la ecuación siguiente:

$$\begin{aligned} minJ &= \frac{1}{2}\bar{u}^T H\bar{u} + \bar{\theta}^T C_\theta^T \bar{u} \\ \text{sujeto a} \quad & A\bar{u} \leq \bar{b} + S_\theta \bar{\theta} \end{aligned} \tag{3.14}$$

Donde $\bar{\theta}$ es el vector de parámetros, y $\bar{u}$ es la solución óptima de la ecuación.

El problema MPQP puede ser reescrito como un conjunto de funciones que describen un control óptimo afín a trozos, PWA, donde se divide el espacio de estados del parámetro $\bar{\theta}$ en varias regiones convexas, poliedros, donde cada una tiene una única ley de control representadas como:

$$\Delta\bar{u}(t) = f(\bar{\theta}(t)) = \begin{cases} F_1\bar{\theta}(t) + g_1 & \text{if } \bar{\theta}(t) \in \bar{\Theta}_1 \\ F_2\bar{\theta}(t) + g_2 & \text{if } \bar{\theta}(t) \in \bar{\Theta}_2 \\ \vdots & \\ F_{N_p}\bar{\theta}(t) + g_{N_p} & \text{if } \bar{\theta}(t) \in \bar{\Theta}_{N_p} \end{cases} \tag{3.15}$$

La división en poliedros se define como $P = \bar{\Theta}_1, \bar{\Theta}_2, \ldots, \bar{\Theta}_{N_p}$, donde el conjunto de poliedros están definidos según la siguiente inecuación lineal (hiperplanos):

$$\bar{\Theta}_i = \left\{ \bar{\theta}(t) \in R^{dim(\bar{x})+1} H_i \bar{\theta}(t) \leq k_i \right\}, i = 1, \ldots, N_p \tag{3.16}$$

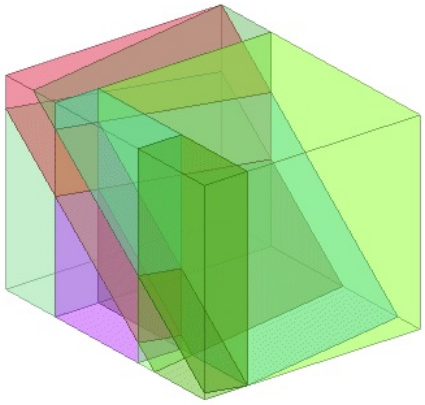

Figura 3.33 Ejemplo del espacio de $\bar{\theta}$ dividido en regiones.

De este modo se tiene definido el problema MPQP como un conjunto de sencillas funciones, creando así un controlador explícito el cual es implementable en un autómata.

Para el cálculo de los parámetros de la solución PWA se ha hecho uso de la herramienta de MATLAB® , Hybrid Toolbox desarrollada por A. Bemporad [1], que definiendo cualquier problema como en la ecuación (3.14) ofrece los valores de los parámetros del conjunto de regiones y funciones de control de las ecuaciones (3.15) (3.16).

En el caso del controlador del proyecto se ha seguido los siguientes pasos desarrollados a continuación.

3.7.2 Transformación del problema QP en MPQP

El problema QP a resolver planteado en apartados anteriores es el siguiente:

$$minJ = \frac{1}{2}\bar{u}^T H \bar{u} + b\bar{u} \quad \text{s. t.} \quad R\bar{u} \leq c \tag{3.17}$$

donde:

$$H = 2(G^T(\lambda_1 I^N)G + \lambda_2 I^{N_c}) \qquad b = 2\lambda_1(f-w)^T G$$

$$R = \begin{bmatrix} +G \\ -G \\ +T \\ -T \end{bmatrix} \qquad c = \begin{bmatrix} \bar{y}_{max} - f \\ -\bar{y}_{min} + f \\ \bar{u}_{max} - 1^{N_c} u(t-1) \\ -\bar{u}_{min} + 1^{N_c} u(t-1) \end{bmatrix}$$

$$f = F_x \bar{x} \quad F_x = [G'(z^{-1})F(z^{-1})] \quad \bar{x} = [\Delta u(t-1) y(t)]^T$$

El problema planteando en la ecuación (3.17) se ha transformado a la forma de un problema MPQP, ecuación (3.14), de la siguiente manera:

$$minJ = \frac{1}{2}\bar{u}^T H \bar{u} + \bar{\theta}^T C_\theta^T \bar{u} \quad \text{s. t.} \quad R\bar{u} \leq \bar{a} + S_\theta \bar{\theta} \tag{3.18}$$

donde:

$$\bar{u} = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \cdots \\ \Delta u(t+N_c-1) \end{bmatrix} \quad H = 2(G^T(\lambda_1 I^N)G + \lambda_2 I^{N_c}) \quad \bar{C}_\theta = 2G^T \begin{bmatrix} F_x & 0^N & -1^N \end{bmatrix}$$

$$R = \begin{bmatrix} +G \\ -G \\ +T \\ -T \end{bmatrix} \bar{a} = \begin{bmatrix} \bar{y}_{max} \\ \bar{y}_{min} \\ \bar{u}_{max} \\ \bar{u}_{min} \end{bmatrix} S_\theta = \begin{bmatrix} -F_x & 0^N & 0^N \\ F_x & 0^N & 0^N \\ 0^{N_c \times dim(\bar{x})} & -1^{N_c} & 0^{N_c} \\ 0^{N_c \times dim(\bar{x})} & 1^{N_c} & 0^{N_c} \end{bmatrix} \bar{\theta} = \begin{bmatrix} \bar{x} \\ u(t-1) \\ ref(t+N) \end{bmatrix}$$

Una vez transformado el problema de optimización QP como uno del tipo MPQP, el siguiente paso es la obtención de la solución explícita planteada en las ecuaciones (3.15) y (3.16).

3.7.3 Transformación del problema MPQP en PWA mediante Hybrid Toolbox

Como previamente se ha definido, el problema MPQP puede ser reescrito como un problema de control óptimo afín a trozos. Para obtener esta solución se ha hecho uso de la herramienta de MATLAB® Hybrid Toolbox [5], la cual definiendo el problema MPQP y haciendo uso de la función *mpqp* ofrece la solución afín a trozos perfectamente definida por el conjunto de vectores y matrices necesarios:

mpqpsol=mpqp(H,C,A,b,S,thmin,thmax).

El Código 3.5 ha sido usado para la obtención de la solución PWA del controlador. La solución está guardada en una estructura tipo mpqpsol en la cual:

H,K,i1,i2: Region #i is stored in $H(i1(i):i2(i),:)$,$K(i1(i):i2(i),:)$
F,G: Control law #i is stored in $F((i-1)*n+1:i*n,:)$,$G((i-1)*n+1:i*n)$
nr = Number of regions = length(i1)

Código 3.5 Código MATLAB® : Obtención del controlador explícito PWA.

```
num=0.01152;
den=[1 3.507 0.004685];
sys=tf(num,den);

%% Parameters:
lambda=0.1; % DU COST CONSTANT
lambda2=50; % ERROR COST CONSTANT
N=7;       % PREDICTION HORIZONT
Nc=1;      % CONTROL HORIZONT
Ts=1;      % Sample time: IMPORTANT!

umin0=6.4;
umax0=75;
ymin0=0;
ymax0=15;
dumin0=-umax0;dumax0=umax0;refmin0=ymin0;refmax0=ymax0;
%Constraints:
umin(1:Nc,1)=umin0;umax(1:Nc,1)=umax0;ymin(1:N,1)=ymin0;ymax(1:N,1)=ymax0;

% Construcción del problema QP segun el Código 3.2
```

```
% ...
% Se obtiene: G,Gprima,F

% Parámetros MPQP
T=tril(ones(Nc),0);
vec1(1:N,1)=1; vec0(1:N,1)=0; vec11(1:Nc,1)=1; vec00(1:Nc,1)=0;

Fx=[Gprima';F']';
H=2*(G'*(lambda2*eye(N))*G+lambda*eye(Nc));
Cth=2*lambda2*G'*[Fx,vec0,-vec1];
R=[G;-G;T;-T];
a=[ymax;-ymin;umax;-umin];
Sth= [ -Fx,vec0,vec0; Fx,vec0,vec0;vec00,vec00,vec00,vec00,-vec11,vec00;
    vec00,vec00,vec00,vec00, vec11,vec00];

thetamin=[dumin0;ymin0;ymin0;ymin0;umin0;refmin0];
thetamax=[dumax0;ymax0;ymax0;ymax0;umax0;refmax0];

mpqpsol=mpqp(H,Cth,R,a,Sth,thetamin,thetamax,1,'quadprog');
```

La solución que se obtiene es como la que se muestra a continuación, ecuación (3.19), en ella están los resultados de solo tres regiones como ejemplo, pero la solución al controlador propuesto ha dado 13

regiones.

$$\Delta\bar{u}(t) = \begin{cases} \text{Región 1:} \begin{bmatrix} -0.1072 & -158.5289 & 136.9569 & -3.9695 & 0 & 25.5415 \end{bmatrix} \bar{\theta}(t) + 0 \\ \text{if} \begin{bmatrix} 0.0000 & 0.1101 & -0.0489 & 0.0014 & 0 & 0.0040 \\ 0.0001 & 0.1443 & -0.0898 & 0.0026 & 0 & 0.0096 \\ 0.0001 & 0.1781 & -0.1304 & 0.0038 & 0 & 0.0152 \\ 0.0001 & 0.2119 & -0.1709 & 0.0049 & 0 & 0.0207 \\ 0.0002 & 0.3130 & -0.2922 & 0.0085 & 0 & 0.0374 \\ -0.0006 & -1.6521 & 0.7333 & 0.0206 & 0 & -0.0607 \\ -0.0010 & -2.1645 & 1.3468 & -0.0385 & 0 & -0.1438 \\ -0.0015 & -2.6721 & 1.9559 & -0.0563 & 0 & -0.2274 \\ -0.0020 & -3.1789 & 2.5641 & -0.0741 & 0 & -0.3110 \\ -0.0034 & -4.6953 & 4.3837 & -0.1274 & 0 & -0.5611 \\ -0.0014 & -2.1137 & 1.8261 & -0.0529 & 0.0133 & 0.3406 \\ 0.0167 & 24.7701 & -21.3995 & 0.6202 & -0.1563 & -3.9909 \end{bmatrix} \bar{\theta} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \\ \text{Región 2:} \begin{bmatrix} -0.340 & -853.9715 & 445.6473 & -12.6228 & 0 & 0.0000 \end{bmatrix} \bar{\theta}(t) + 6.3142e^{03} \\ \text{if} \begin{bmatrix} -0.0000 & -0.0852 & 0.0190 & -0.0005 & 0 & 0.0000 \\ -0.0000 & -0.0854 & 0.0192 & -0.0005 & 0 & 0.0000 \\ -0.0001 & -0.1369 & 0.0714 & -0.0020 & 0.0002 & 0.0000 \\ 0.0001 & 0.1354 & -0.0707 & 0.0020 & -0.0002 & -0.0000 \\ -0.0000 & -0.1101 & 0.0489 & -0.0014 & 0 & -0.0040 \end{bmatrix} \bar{\theta} \leq \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \\ \text{Región 3:} \begin{bmatrix} -0.2918 & -543.0582 & 376.2159 & -10.8071 & 0 & 0.0000 \end{bmatrix} \bar{\theta}(t) + 2.6647e^{03} \\ \text{if} \begin{bmatrix} 0.0000 & 0.0852 & -0.0190 & 0.0005 & 0 & 0.0000 \\ -0.0000 & -0.0861 & 0.0200 & -0.0005 & 0 & 0.0000 \\ -0.0001 & -0.2097 & 0.1453 & -0.0042 & 0.0004 & 0.0000 \\ 0.0001 & 0.2043 & -0.1415 & 0.0041 & -0.0004 & -0.0000 \\ -0.0001 & -0.1443 & 0.0898 & -0.0026 & 0 & -0.0096 \end{bmatrix} \bar{\theta} \leq \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \\ \dots \end{cases} \tag{3.19}$$

3.7.4 Limitaciones

1. El problema que se ha presentado al hacer uso de esta herramienta MATLAB® es que el controlador explícito GPC se ve limitado a la posibilidad de la resolución del problema MPQP. Los parámetros N y N_c se ven limitados, ya que cuanto más grandes sean, el número de regiones crece significativamente, y no siempre la herramienta nos da una solución. Definido nuestro modelo, el problema ha quedado limitado a

$$N = 7 \qquad N_c = 1 \tag{3.20}$$

en adelante estarán estos valores fijados.

2. En el controlador GPC planteado en el apartado anterior la referencia se define como un vector de N componentes, que en el instante t conocía la trayectoria de la referencia desde t a t+N pudiendo éste ser variante a lo largo del vector, lo que hacía al GPC poder adelantarse a los cambios de referencia.
   Para la solución explícita la referencia se define constante para t a t+N debido a que si se quisiera definir como una trayectoria variable el vector de parámetros θ crecería bastante ya que habría que introducir el vector de referencia de N componentes en él. Esto provocaría un aumento muy grande del número de regiones y la herramienta usada tampoco da una solución a este problema debido a su dimensión. Aun así en la vida real la referencia no es siempre conocida, por lo que en muchas ocasiones el problema se ve limitado tal y como se plantea ahora.

3.7.5 GPC Explícito con Gain-Scheduling

Ya que el controlador desarrollado en el apartado 3.6 se ha definido como un GPC con Gain-Scheduling, no solo se ha obtenido la solución explícita para en control GPC basado en un solo modelo, si no que en este proyecto se ha querido desarrollar la solución explícita del controlador GPC con Gain-Scheduling. Para ello se ha desarrollado la idea planteada en este capítulo para cada uno de los modelos, haciendo uso del Código 3.5, cambiando tan solo la función de transferencia hasta en tres ocasiones, resolviendo así tres problemas MPQP, uno por cada modelo del controlador y obteniendo la solución explícita de estos.

El resultado ha sido:

Primer modelo $\Rightarrow$ 13regiones.
Segundo modelo $\Rightarrow$ 13regiones.
Tercer modelo $\Rightarrow$ 17regiones.

3.7.6 Implementación

Control explícito desde MATLAB®

Antes de implementar el controlador en el autómata, se ha querido comprobar el buen funcionamiento implementando el nuevo controlador en MATLAB® y usando la conexión OPC hacer las pruebas al sistema.
La implementación del GPC explícito se muestra en el Código 3.6. Previamente se han guardado en las variables 'mpqpsol1', 'mpqpsol2','mpqpsol3'las estructuras del controlador PWA de cada modelo.

Código 3.6 Código MATLAB® : Bucle de control explícito.

```
if y(na)<6
mpqpsol=mpqpsol1;
memory(k,1)=1;
end
if y(na)>=6 && y(na)<8
mpqpsol=mpqpsol2;
memory(k,1)=2;
end
if y(na)>=8
mpqpsol=mpqpsol3;
memory(k,1)=3;
end

y0=read(y0_1); y(na+1)=(double(y0.Value))/1000;yy(time)=y(na+1);
y0real=read(y0_real); yreal(time)=(double(y0real.Value))/1000;
```

```
rr(time)=ref(time);

theta(time,:)=[du(nb+1) y(na+1) y(na) y(na-1) u(nb+1) rr(time)];
    for kk=1:length(mpqpsol.i1)
        H0=mpqpsol.H(mpqpsol.i1(kk):mpqpsol.i2(kk),:);
        K0=mpqpsol.K(mpqpsol.i1(kk):mpqpsol.i2(kk));
         if H0*theta(time,:)'<=K0
         ii=kk;
         memory(time,2)=kk;
         end
        clear H0 K0
    end
    F0=mpqpsol.F((ii-1)*Nc+1:ii*Nc,:);
    G0=mpqpsol.G((ii-1)*Nc+1:ii*Nc);

duk=F0*theta(time,:)'+G0;

du(nb+2)=duk(1);duu(time)=du(nb+2);
u(nb+2)=u(nb+1)+du(nb+2);uu(time)=u(nb+2);

write(u0_1,(u(nb+2)*10));
```

Varios ensayos se han realizado una vez que se ha tenido programado el controlador. Tras realizar un ensayo como el de la Figura 3.34 se puede analizar que:

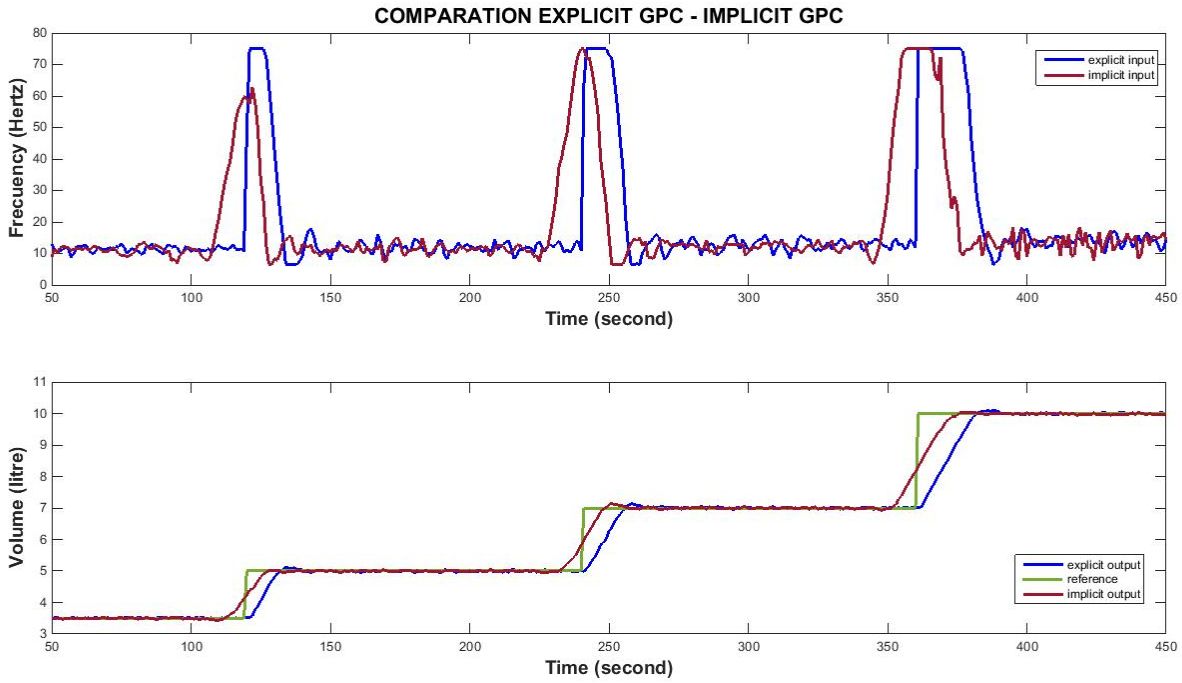

Figura 3.34 Ensayo GPC Explícito ($N = 7\ N_c = 1$) frente GPC Implícito ($N = 20\ N_c = 20$) .

- El controlador explícito GPC con restricciones funciona bien y es bueno.
- Debido a las limitaciones antes nombradas la solución explícita no es tan buena como el GPC implícito calculado en el apartado anterior del proyecto.
- Un ensayo en simulación del GPC implícito con las mismas condiciones que el nuevo controlador tiene, $N = 7\ N_c = 1$, se ha realizado para compararlo con los resultados del GPC explícito de

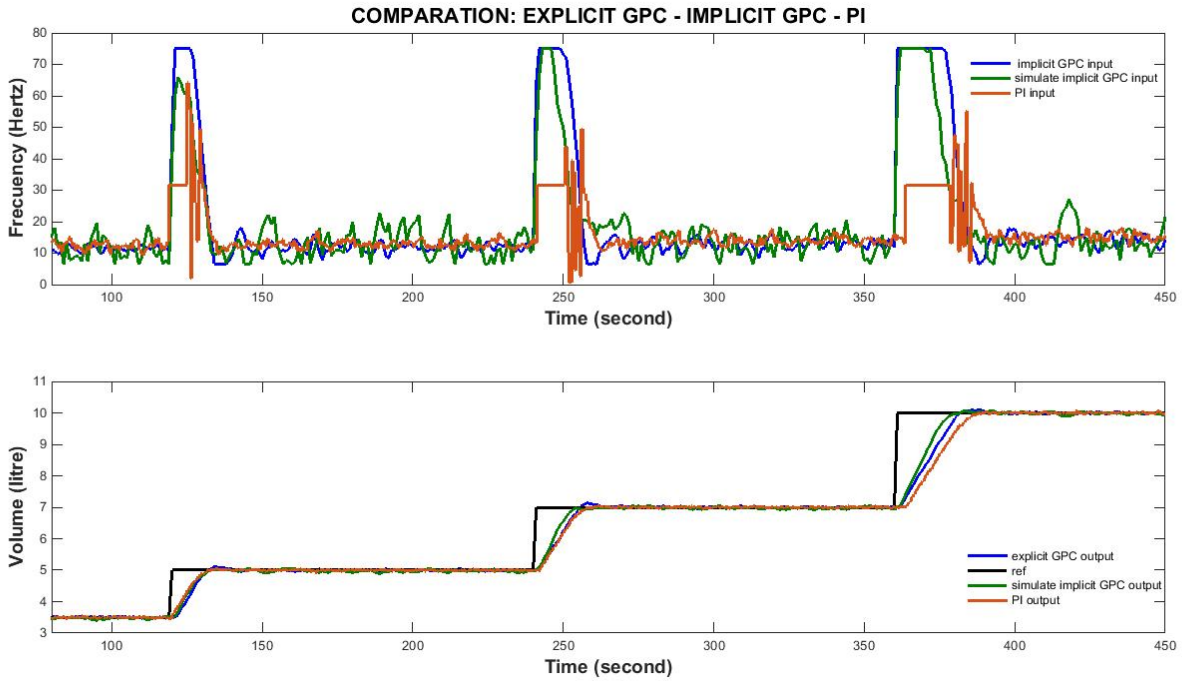

Figura 3.35 Ensayo GPC Explícito ($N = 7\ N_c = 1$) frente GPC Implícito Simulado ($N = 7\ N_c = 1$) y PI .

la Figura 3.34. Esta comparación se puede ver en la Figura 3.35 donde queda demostrado que se puede obtener un controlador GPC explícito igual de bueno que el GPC implícito ya que los resultados son iguales de buenos.

– También se ha querido destacar que aunque las condiciones del controlador sean distintas el controlador GPC explícito sigue dando mejores resultados que el controlador PI como puede verse en la Figura 3.35.

Control explícito desde PLC. OFFLINE

Una vez comprobado que los resultados del controlador explícito son buenos, se ha querido implementar en el autómata y así tener un controlador offline, completamente autónomo de otras computadoras o conexiones OPC.

La implementación del controlador se ha hecho en lenguaje SCL, uno de los lenguajes de programación que dispone el software del PLC el cual está basado en en el lenguaje ST de la norma IEC-G11317.

Se ha creado una función, Figura 4.1, en la cual está programado el controlador. Esta función trabaja con un bloque de datos propio "CONTROL GPC DB 1". Es necesario copiar todos los valores de las matrices y vectores que describen el controlador PWA , ecuaciones 3.15 y 3.16, en variables estáticas tipo array dentro del bloque de datos. Esta función se ejecuta cada tiempo de muestreo , 1 segundo, al igual que lo hace el bloque PI.

El código SCL, Código 3.6, que tiene programado la función "CONTROL GPC" tiene un algoritmo parecido al usado en MATLAB® Código 3.7.

Código 3.7 Código SCL: Bucle de control explícito. TIA PORTAL.

```
% PROGRAMA QUE SE EJECUTA CADA 1 SEGUNDO= SAMPLE TIME.
% Actualización de las variables.
%Pasadas:
#DU1:= #DU0;
```

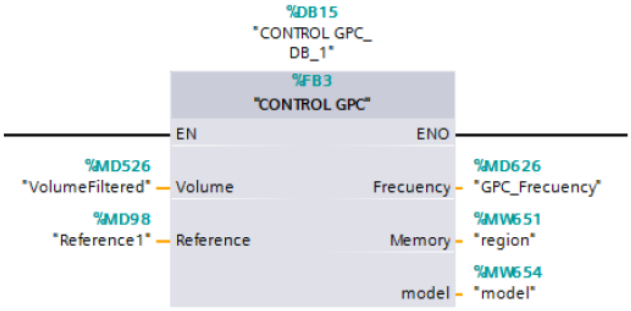

Figura 3.36 GPC. TIA PORTAL.

```
#Frecuency1:=#Frecuency0;
#Volume2:=#Volume1;
#Volume1:=#Volume0;
%Actuales:
#Volume0:=#Volume;
#Reference0:=#Reference;
% Actualización del vector de estado:
#Theta[0]:=#DU1;
#Theta[1]:=#Volume0;
#Theta[2]:=#Volume1;
#Theta[3]:=#Volume2;
#Theta[4]:=#Frecuency1;
#Theta[5]:=#Reference0;


IF #Volume0 < 6 THEN

% Comprobación región tras región para saber en cual nos encontramos
FOR #count := 0 TO 12 DO % 13 regiones
#dimension := #I1[1,#count]- #I1[0,#count]; % dimensión de las matrices
#flag:=#dimension+1;
#min :=#I1[0,#count]-1;

FOR #ii := 0 TO #dimension DO % Comprobamos la inecuación fila por fila,
    en función de la dimensión.
#N[#ii] := #H1[0,(#ii+#min)]*#Theta[0]+ #H1[1,(#ii+#min)]*#Theta[1]+#H1
    [2,(#ii+#min)]*#Theta[2]+#H1[3,(#ii+#min)]*#Theta[3]+#H1[4,(#ii+#min)
    ]*#Theta[4]+#H1[5,(#ii+#min)]*#Theta[5];
   IF #N[#ii] <= #K1[0,(#ii+#min)] THEN
    #flag:=#flag-1; % Si una fila se cumple restamos un flag.
   END_IF;
   IF #flag=0 THEN
    % Si todas las filas se han cumplido flag=0. Región correcta.
    #region:= #count;
   END_IF;

END_FOR;
END_FOR;

% Cálculo de la acción de control: du= F(region)*thetha + G(region)
```

```
#M:= #F1[0,#region]*#Theta[0]+ #F1[1,#region]*#Theta[1] + #F1[2,#region]*#
    Theta[2] + #F1[3,#region]*#Theta[3] +#F1[4,#region]*#Theta[4] +#F1[5,#
    region]*#Theta[5];
#DUK:= #M + #G1[0,#region];
#DU0:=#DUK;
#Frecuency0 := #Frecuency1 + #DU0;
% Salidas del bloque:
#Frecuency := #Frecuency0;
#Memory := #region;
#model:=1;
END_IF;


IF (#Volume0 >= 6 & #Volume0 < 8) THEN
FOR #count := 0 TO 12 DO % 13 Regiones
...
%Mismas operaciones que en el bucle if anterior pero utilizando H2,K2,I2
END_FOR;
...
%Mismas operaciones que en el bucle if anterior pero utilizando F2,G2
#model:=2;
END_IF;


IF #Volume0 >= 8 THEN
FOR #count := 0 TO 16 DO % 17 Regiones
...%Mismas operaciones que en el bucle if anterior pero utilizando H3,K3,
    I3
END_FOR;
...%Mismas operaciones que en el bucle if anterior pero utilizando F3,G3
#model:=3;
END_IF;
```

– Resultados:
Debido a la necesidad de modificar el diseño de la planta para el proyecto que en el centro Nimbus se estaba desarrollando, la planta ha sufrido ciertos cambios que han cambiado significativamente la dinámica de la planta.

Un caudalímetro a la salida de la bomba y otro caudalímetro junto a la válvula de salida de agua han modificado tanto la dinámica de entrada de agua como la dinámica de salida.

Como se explicó en el apartado 2.2.1 el control predictivo está basado en tres ideas principales, una de ellas es el uso de un modelo para la predicción de la salida en un intervalo de tiempo futuro. Esto hace que el modelo sea una de las claves del MPC. La fidelidad entre el modelo y la planta caracteriza a un buen controlador.

Las modificaciones en el diseño de la planta, han producido grandes cambios en la dinámica lo que hace que los modelos con los que se han trabajado no sean validos para la nueva planta. De modo que el control en este proyecto desarrollado ya no es el adecuado para la planta.

La solución a este problema presentado sería empezar desde el principio todos los pasos seguidos en este proyecto. Empezando por volver a modelar el sistema, definir el problema GPC, obtener el controlador y una vez definido crear la solución explícita de este.

4 Conclusiones y Líneas Futuras

4.1 Conclusiones

En este proyecto se ha querido plantear un problema completo, que envuelve técnicas de control avanzadas junto a las técnicas necesarias para implementar un control en un sistema real, buscando solución a todos los problemas que se han presentado como la comunicación entre el controlador y el sistema, o la eliminación de ruido existente en la planta.

GPC con restricciones

El primer objetivo del proyecto era el de crear un controlador GPC para controlar un sistema SISO y así ver como funciona un controlador predictivo en la realidad y no tan solo en simulación como se ha visto a lo largo de la carrera. El resultado ha sido la creación de un controlador valido para un sistema SISO definido por un modelo en función de transferencia y sus restricciones que además tiene un conjunto de parámetros a escoger según requisitos.

Los resultados obtenidos han sido los esperados y también se ha querido comparar con un controlador PI para ver las ventajas que tiene el control predictivo frente al control tradicional.

GPC con restricciones + Gain- Scheduling

Uno de los pilares del control predictivo es el modelo que utiliza para la predicción. En muchas ocasiones debido a las no linealidades existentes un solo modelo no representa bien un sistema completo haciendo así complicado obtener un buen controlador predictivo basado en un modelo.

Como solución a este problema se plantea como segundo objetivo de este proyecto la implantación de la estrategia Gain-Scheduling. Esta técnica basada en definir el sistema con un conjunto de modelos y utilizar el modelo adecuado en cada momento en el controlador, ofrece al GPC la oportunidad de dar tan buenos resultados como los que teóricamente se esperan.

Tal y como se esperaba, los resultados de esta estrategia aplicada en la planta real han sido buenos, y el controlador GPC utilizando Gain-Scheduling ofrece un mejor resultado.

GPC con restricciones OFFLINE

Como se ha dicho previamente, la idea del proyecto era la de implementar un controlador en un sistema real. La opción de controlar una planta mediante un controlador implementado en MATLAB® hace necesario una conexión OPC que comunica el autómata con el controlador. Se puede decir que esta solución no es completa del todo, porque hace al sistema dependiente de otra computadora. Una solución real sería si el controlador GPC estuviera implementado en el PLC y eso es lo que se ha conseguido en este último objetivo del proyecto.

El controlador GPC tiene una solución explícita cuando no se tienen en cuenta ningún tipo de restricciones, pero al tenerlas en cuenta, la resolución del GPC envuelve un problema de optimización QP que el autómata no puede resolver.

Como se ha desarrollado en el último apartado del proyecto, aplicando la técnica de resolución de problemas de optimización como problemas multiparamétricos, se ha reescrito el problema de control QP como un problema multiparamétrico el cual queda definido como un conjunto de funciones explícitas, las cuales sí son implementables en el autómata.

La capacidad de control utilizando esta técnica ha sido probada en la planta dando lo resultados esperados.

Debido a la necesidad de modificar el diseño de la planta para el proyecto que en el centro Nimbus se estaba desarrollando, la planta ha sufrido ciertos cambios que han cambiado significativamente la dinámica de la planta y no se ha podido probar el controlador desde el autómata. Aun así, en el ensayo realizado, Figura 3.35, ha quedado demostrado la posible implementación explícita del GPC aunque probado desde MATLAB® .

4.2 Líneas Futuras

1. En primer lugar, en cuanto el nuevo sistema quedase definido, la primera idea sería la de volver a modelar la planta y crear el nuevo controlador a partir de los nuevos modelos. Implementando después la solución explícita obtenida en el autómata. De este modo se deja el controlador GPC implementado en el PLC controlando la planta.

2. Una de las desventajas que presenta el controlador explícito es que no acepta ningún tipo de cambio una vez implementado. Es decir previamente se han debido fijar todos los parámetros del controlador. Para poder cambiar las constantes de ponderación, el tiempo de muestreo o los horizontes habría que plantear el problema de nuevo, obtener la solución explícita e implementarla en el autómata. Una solución a este problema y una futura línea de trabajo sería la de obtener soluciones explícitas de controladores variando sus parámetros, implementar todos esos controladores y así poder escoger los parámetros una vez implementado todo en el PLC.

3. Debido a los cambios de diseño de la planta, unos caudalímetros van a ser introducidos a la entrada y a la salida del tanque. La dinámica de un tanque de agua puede ser estudiada basándose en la física de éste siempre que la entrada sea en caudal. En este caso el modelo podría obtenerse a partir de la linealización de la ecuación física correspondiente. Podría resultar más útil implementar el controlador bajo estas circunstancias.

   Ya que el caudal de entrada sigue dependiendo de la frecuencia de la bomba, que es la única variable a la que tenemos acceso, se podría aplicar la técnica de control en cascada al sistema. Implementando un controlador PID para controlar el caudal de entrada en función de la frecuencia de la bomba. El nuevo control seguiría el siguiente esquema:

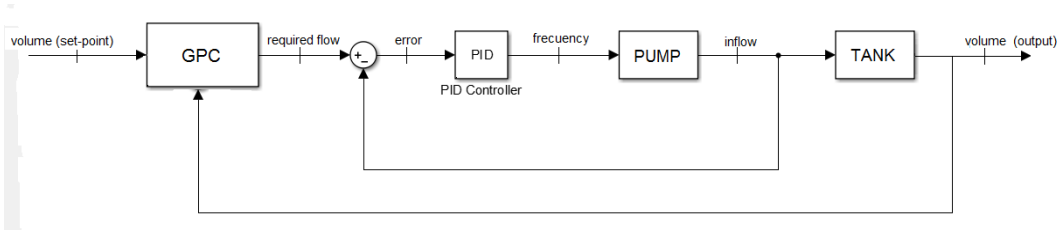

Figura 4.1 Esquema Control en Cascada.

Apéndice A
Códigos MATLAB®

A.1 Controlador GPC con y sin restricciones para Simulación

Código A.1 Código MATLAB® : Controlador GPC para Simulación.

```
%% GPC with or without CONSTRAINTS.
% SIMULATE SYSTEM

clear
clc

lambda=0.5; % DU COST CONSTANT
lambda2=1; %  ERROR COST CONSTANT
N=20;      % PREDICTION HORIZONT
Nc=5;     % CONTROL HORIZONT
Ts=10;     % Sample time: IMPORTANT! DEFINE THE DISCRET TIME AND CONTROL TIME

%G(s)
num=0.01152;
den=[1 3.507 0.004685];

Initial_values

%Constrains:
Constrains=1;
dumin(1:Nc,1)=-75;
dumax(1:Nc,1)=75;
umin(1:Nc,1)=6.4;
umax(1:Nc,1)=75;
ymin(1:N,1)=0;
ymax(1:N,1)=15;

tsim=700;
ref=Rfunction;
u=zeros((nb+2),1);
du=zeros((nb+2),1);
y=zeros((na+1),1);
uuu=zeros((nb+2),1);
yy=zeros(tsim,1);
uu=zeros(tsim,1);
duu=zeros(tsim,1);
```

```
%% Simulation:
for  k=(1+na):tsim

for i=1:na
    y(i)=y(i+1);
end
for i=1:na
    du(i)=du(i+1);
end
for i=1:na
    u(i)=u(i+1);
end
for i=1:na
    uuu(i)=uuu(i+1);
end


%System
yk=-A(+2:na+1)*y(na:-1:1)+B*uuu(nb+1:-1:1)+(rand(1)-0.5)*0.084;
y(na+1)=yk;
yy(k)=y(na+1);



% Free response;
x=[du(nb+1:-1:2);y(na+1:-1:1)];
Fx=[Gprima';F']';
f=Fx*x;


b=2*lambda2*(f-ref(k:k+N-1))'*G;
if (Constrains==1) % Solution with Constraints
    c=[dumax;-dumin;umax-c1*u(nb+1);-umin+c1*u(nb+1);ymax-f;-ymin+f];
    [duk,min,flag]=quadprog(H,b,R,c);
    if (flag~=1)
        flag
    end
else               % Solution without Constraints
    [duk]=-Hinv*b';
end

duk(1)=round(duk(1)*10)/10;

du(nb+2)=duk(1);
duu(k)=du(nb+2);

u(nb+2)=u(nb+1)+du(nb+2);
uu(k)=u(nb+2);
uuu(nb+2)=u(nb+2)-10.4;

end

figure
subplot(2,1,1)
plot(Ts:Ts:Ts*tsim,uu,Ts:Ts:Ts*tsim,duu)
```

```
subplot(2,1,2)
plot(Ts:Ts:Ts*tsim,yy,Ts:Ts:Ts*tsim,ref(1:tsim))
```

Código A.2 Código MATLAB® : "Initial_Values" Creación del controlador GPC.

```
sys=tf(num,den);


sysd = c2d(sys,Ts,'zoh'); % Discrete system
[fil,col]=size(sysd.num{1});
B=sysd.num{1}(2:col);
A=sysd.den{1};

nb=length(B)-1;
na=length(A)-1;


%Calculation of Fj
Avir=conv(A,[1 -1]);
F=zeros(N,na+1);
poluno=[1 zeros(1,na+1)];
F1aux=poluno-Avir;
F(1,:)=F1aux(1,2:na+2);
for j=1:(N-1)
    for i=0:na-1
        F(j+1,i+1)=F(j,i+2)-F(j,1)*Avir(1,i+2);
    end
    F(j+1,na+1)=-F(j,1)*Avir(1,na+2);
end

%Calculation of Ej
E=zeros(N,N);
E(1,1)=1;
for j=1:(N-1)
    E(j+1,:)=E(j,:);
    E(j+1,j+1)=F(j,1);
end

%Calculation of Gj=Ej*Denj
Gpols=zeros(N,N+nb);
for j=1:N
    Gpols(j,:)=conv(E(j,:),B);
end

%Calculation of G Matrix
G2=zeros(N,N);
for j=1:N
    kk=1;
    for i=j:-1:1
        G2(j,kk)=Gpols(j,i);
        kk=kk+1;
    end
end
G=G2(:,1:Nc);
```

```
%Calculation of G' Matrix
Gprima=[];
if nb>0
    Gprima=zeros(N,nb);
    for j=1:N
        Gprima(j,:)=Gpols(j,1+j:(j+nb));
    end
end

H=2*(G'*(lambda2*eye(N))*G+lambda*eye(Nc));
Hinv=inv(H);
c1(1:Nc,1)=1;
I=eye(Nc);
T=tril(ones(Nc),0);
R=[I;-I;T;-T;G;-G];
```

Código A.3 Código MATLAB® : "Rfunction" : Ejemplo de Función para la Referencia.

```
function [ref] =Rfunction
    ref(1:119,1)=3.5;
    ref(120:240,1)=5;
    ref(241:360,1)=7;
    ref(361:430,1)=5;
    ref(431:600,1)=6;
    ref(601:800,1)=13;
end
```

A.2 Controlador GPC con restricciones para Sistema real

Código A.4 Código MATLAB® : Controlador GPC para sistema real mediante OPC.

```
%% GPC with CONSTRAINTS
% Real System Control.
% With OPC
clear
clc

OPC_Matlab

lambda=0.1; % DU COST CONSTANT
lambda2=50; % ERROR COST CONSTANT
N=20;      % PREDICTION HORIZONT
Nc=20;     % CONTROL HORIZONT
Ts=1;     % Sample time: IMPORTANT! DEFINE THE DISCRET TIME AND CONTROL TIME

%G(s)
num=0.01152;
den=[1 3.507 0.004685];

Initial_values

%Constrains:
dumin(1:Nc,1)=-75;
```

```
dumax(1:Nc,1)=75;
umin(1:Nc,1)=6.4;
umax(1:Nc,1)=75;
ymin(1:N,1)=0;
ymax(1:N,1)=15;

u=zeros((nb+2),1);
du=zeros((nb+2),1);
y=zeros((na+1),1);

ref=Rfunction;

GPC=read(GPC0);
Value=GPC.Value

while(Value==0)
tic
time=time+1;

for i=1:na
    y(i)=y(i+1);
end
for i=1:nb+1
    du(i)=du(i+1);
end
for i=1:nb+1
    u(i)=u(i+1);
end

% Las lecturas del OPC llegan como enteros en mililitros
y0=read(y0_1);
y(na+1)=(double(y0.Value))/1000;
y0real=read(y0_real);
yreal(time)=(double(y0real.Value))/1000;

yy(time)=y(na+1);

rr(time)=ref(time);

% Free response;
x=[du(nb+1:-1:2);y(na+1:-1:1)];
Fx=[Gprima';F']';
f=Fx*x;

b=2*lambda2*(f-ref(time:time+N-1))'*G;
c=[dumax;-dumin;umax-c1*u(nb+1);-umin+c1*u(nb+1);ymax-f;-ymin+f];
[duk,min,flag]=quadprog(H,b,R,c);
if (flag~=1)
 flag
end
duk(1)=round(duk(1)*10)/10;

du(nb+2)=duk(1);
duu(time)=du(nb+2);

u(nb+2)=u(nb+1)+du(nb+2);
uu(time)=u(nb+2);
```

```
write(u0_1,(u(nb+2)*10));

GPC=read(GPC0);
Value=GPC.Value;
interval=toc;
pause(Ts-interval);
end

figure
subplot(2,1,1)
plot(Ts:Ts:Ts*(time),uu,Ts:Ts:Ts*(time),duu);
subplot(2,1,2)
plot(Ts:Ts:Ts*(time),yreal,Ts:Ts:Ts*(time),yy,Ts:Ts:Ts*(time),rr);
```

Código A.5 Código MATLAB® : "OPC_Matlab" Creación Cliente OPC.

```
%Create the OPC Client:
client=opcserverinfo('localhost');

da=opcda('127.0.0.1','Kepware.KEPServerEX.V5');

%Connect with the server
connect(da);

%Create a group:
group1=addgroup(da);

y0_1 = additem(group1,'OPC1.PLC1.Volume1000');
y0_real = additem(group1,'OPC1.PLC1.Volume1000Real');
y0_2 = additem(group1,'OPC1.PLC1.Volume2000');

u0_1 = additem(group1,'OPC1.PLC1.Frecuency1');
u0_0 = additem(group1,'OPC1.PLC1.PIFREC');
u0_2 = additem(group1,'OPC1.PLC1.Frecuency2');

r0_1 = additem(group1,'OPC1.PLC1.Reference1000');
r0_2 = additem(group1,'OPC1.PLC1.Reference2000');

GPC0 = additem(group1,'OPC1.PLC1.GPC');
start = additem(group1,'OPC1.PLC1.StarCopy');
```

A.3 Controlador GPC con restricciones Gain-Scheduling para Simulación

Código A.6 Código MATLAB® : Controlador GPC con Gain-Scheduling. Simulación.

```
%% GPC with CONSTRAINTS with Gain- Scheduling
% Simulation
clear
clc

lambda=0.1; % DU COST CONSTANT
lambda2=50; % ERROR COST CONSTANT
```

```
N=7;      % PREDICTION HORIZONT
Nc=1;     % CONTROL HORIZONT
Ts=1;     % Sample time: IMPORTANT! DEFINE THE DISCRET TIME AND CONTROL TIME


%Constrains:
dumin(1:Nc,1)=-75;
dumax(1:Nc,1)=75;
umin(1:Nc,1)=6.4;
umax(1:Nc,1)=75;
ymin(1:N,1)=0;
ymax(1:N,1)=15;

tsim=1400;
ref=Rfunction;
nb=1;
na=2;
u=zeros((nb+2),1);
du=zeros((nb+2),1);
y=zeros((na+1),1);
uuu=zeros((nb+2),1);
yy=zeros(tsim,1);
uu=zeros(tsim,1);
duu=zeros(tsim,1);


%% Simulation
for  k=(1+na):tsim

for i=1:na
    y(i)=y(i+1);
end
for i=1:na
    du(i)=du(i+1);
end
for i=1:na
    u(i)=u(i+1);
end
for i=1:na
    uuu(i)=uuu(i+1);
end

if y(na)<6
 num=0.0088704;
 den=[1 3.498 0.004679];
 offset=9.7;
 Initial_values
end

if y(na)>=6 && y(na)<8
 num=0.01152;
 den=[1 3.507 0.004685];
 offset=10.4;
 Initial_values
end
if y(na)>=8
  num=0.002274;
```

```
  den=[1 0.8613 0.0007163];
  offset=11.1;
  Initial_values
end


yk=-A(+2:na+1)*y(na:-1:1)+B*uuu(nb+1:-1:1)+(rand(1)-0.5)*0.05;
y(na+1)=yk;
yy(k)=y(na+1);

% Free response;
x=[du(nb+1:-1:2);y(na+1:-1:1)];
Fx=[Gprima';F']';
f=Fx*x;


b=2*lambda2*(f-ref(k:k+N-1))'*G;
c=[dumax;-dumin;umax-c1*u(nb+1);-umin+c1*u(nb+1);ymax-f;-ymin+f];
[duk,min,flag]=quadprog(H,b,R,c);
if (flag~=1)
  flag
end

duk(1)=round(duk(1)*10)/10;

du(nb+2)=duk(1);
duu(k)=du(nb+2);

u(nb+2)=u(nb+1)+du(nb+2);
uu(k)=u(nb+2);
uuu(nb+2)=u(nb+2)-offset;

end

figure
subplot(2,1,1)
plot(Ts:Ts:Ts*tsim,uu,Ts:Ts:Ts*tsim,duu)
subplot(2,1,2)
plot(Ts:Ts:Ts*tsim,yy,Ts:Ts:Ts*tsim,ref(1:tsim))
```

A.4 Controlador GPC con restricciones Gain-Scheduling para Sistema real

Código A.7 Código MATLAB® : Controlador GPC con Gain-Scheduling para Sistema Real mediante OPC.

```
%% GPC with CONSTRAINTS
clear
clc

OPC_Matlab

lambda=0.1; % DU COST CONSTANT
lambda2=50; % ERROR COST CONSTANT
N=20;       % PREDICTION HORIZONT
Nc=20;      % CONTROL HORIZONT
```

```
Ts=1;     % Sample time: IMPORTANT! DEFINE THE DISCRET TIME AND CONTROL TIME


%Constrains:
dumin(1:Nc,1)=-75;
dumax(1:Nc,1)=75;
umin(1:Nc,1)=6.4;
umax(1:Nc,1)=75;
ymin(1:N,1)=0;
ymax(1:N,1)=15;
nb=1;
na=2;
u=zeros((nb+2),1);
du=zeros((nb+2),1);
y=zeros((na+1),1);

ref=Rfunction;

GPC=read(GPC0);
Value=GPC.Value

while(Value==0)
tic
time=time+1;

for i=1:na
    y(i)=y(i+1);
end
for i=1:nb+1
    du(i)=du(i+1);
end
for i=1:nb+1
    u(i)=u(i+1);
end

if y(na)<6
 num=0.0088704;
 den=[1 3.498 0.004679];
 modelo(time)=1;
 Initial_Values
end

if y(na)>=6 && y(na)<8
 num=0.01152;
 den=[1 3.507 0.004685];
 modelo(time)=2;
 Initial_Values
end
if y(na)>=8
  num=0.002274;
  den=[1 0.8613 0.0007163];
  modelo(time)=3;
  Initial_Values
end

y0=read(y0_1);
y(na+1)=(double(y0.Value))/1000;
```

```
y0real=read(y0_real);
yreal(time)=(double(y0real.Value))/1000;

yy(time)=y(na+1);

rr(time)=ref(time);

% Free response;
x=[du(nb+1:-1:2);y(na+1:-1:1)];
Fx=[Gprima';F']';
f=Fx*x;

b=2*lambda2*(f-ref(time:time+N-1))'*G;
c=[dumax;-dumin;umax-c1*u(nb+1);-umin+c1*u(nb+1);ymax-f;-ymin+f];
[duk,min,flag]=quadprog(H,b,R,c);
if (flag~=1)
 flag
end
duk(1)=round(duk(1)*10)/10;

du(nb+2)=duk(1);
duu(time)=du(nb+2);

u(nb+2)=u(nb+1)+du(nb+2);
uu(time)=u(nb+2);
write(u0_1,(u(nb+2)*10));

GPC=read(GPC0);
Value=GPC.Value;
interval=toc;
pause(Ts-interval);
end

figure
subplot(2,1,1)
plot(Ts:Ts:Ts*(time),uu,Ts:Ts:Ts*(time),duu);
subplot(2,1,2)
plot(Ts:Ts:Ts*(time),yy,Ts:Ts:Ts*(time),yreal,Ts:Ts:Ts*(time),rr);
```

A.5 Controlador explícito GPC con restricciones Gain-Scheduling para Simulación

Código A.8 Código MATLAB® : Control PWA, GPC Explícito para Simulación.

```
%% GPC_GS_OFFLINE

%% GPC-OFFLINE-SIMULATION
clear
clc


%% Parameters:
lambda=0.1; % DU COST CONSTANT
lambda2=50; % ERROR COST CONSTANT
```

```
N=7;      % PREDICTION HORIZONT
Nc=1;     % CONTROL HORIZONT
Ts=1;     % Sample time: IMPORTANT! DEFINE THE DISCRET TIME AND CONTROL TIME

umin0=6.4;
umax0=75;
ymin0=0;
ymax0=15;
dumin0=-umax0;
dumax0=umax0;
refmin0=ymin0;
refmax0=ymax0;

%Constraints:
umin(1:Nc,1)=umin0;
umax(1:Nc,1)=umax0;
ymin(1:N,1)=ymin0;
ymax(1:N,1)=ymax0;


% MP 3 Models
num=0.0088704;
den=[1 3.498 0.004679];
sys=tf(num,den);

InitialValues2
mpqpsol1=mpqp(H,C,AA,b,Sth,thetamin,thetamax,1,'quadprog');

num=0.01152;
den=[1 3.507 0.004685];
sys=tf(num,den);

InitialValues2
mpqpsol2=mpqp(H,C,AA,b,Sth,thetamin,thetamax,1,'quadprog');

num=0.002274;
den=[1 0.8613 0.0007163];
sys=tf(num,den);

InitialValues2
mpqpsol3=mpqp(H,C,AA,b,Sth,thetamin,thetamax,1,'quadprog');


%% SIMULATION :

tsim=1400;
ref=trial1;
u=zeros((nb+2),1);
du=zeros((nb+2),1);
y=zeros((na+1),1);
uuu=zeros((nb+2),1);
yy=zeros(tsim,1);
uu=zeros(tsim,1);
duu=zeros(tsim,1);


for  k=(1+na):tsim
```

```
for i=1:na
    y(i)=y(i+1);
end
for i=1:na
    du(i)=du(i+1);
end
for i=1:na
    u(i)=u(i+1);
end
for i=1:na
    uuu(i)=uuu(i+1);
end

if y(na)<6
mpqpsol=mpqpsol1;
memory(k,3)=1;
%Model to simulate:
num=0.0088704;den=[1 3.498 0.004679];offset=9.7;sys=tf(num,den);
sysd = c2d(sys,Ts,'zoh'); % Discrete system
[fil,col]=size(sysd.num{1});B=sysd.num{1}(2:col);A=sysd.den{1};
end
if y(na)>=6 && y(na)<8
mpqpsol=mpqpsol2;
memory(k,3)=2;
%Model to simulate:
num=0.01152;den=[1 3.507 0.004685];offset=10.4;sys=tf(num,den);
sysd = c2d(sys,Ts,'zoh'); % Discrete system
[fil,col]=size(sysd.num{1});B=sysd.num{1}(2:col);A=sysd.den{1};
end
if y(na)>=8
mpqpsol=mpqpsol3;
memory(k,3)=3;
%Model to simulate:
num=0.002274;den=[1 0.8613 0.0007163];offset=11.1;sys=tf(num,den);
sysd = c2d(sys,Ts,'zoh'); % Discrete system
[fil,col]=size(sysd.num{1});B=sysd.num{1}(2:col);A=sysd.den{1};
end

yk=-A(+2:na+1)*y(na:-1:1)+B*uuu(nb+1:-1:1)+(rand(1)-0.5)*0.05;
y(na+1)=yk;
yy(k)=y(na+1);

% Free response;
rr(k)=ref(k);
theta(k,:)=[du(nb+1) y(na+1) y(na) y(na-1) u(nb+1) rr(k) ];
count0=0;

    for kk=1:length(mpqpsol.i1)
        H0=mpqpsol.H(mpqpsol.i1(kk):mpqpsol.i2(kk),:);
        K0=mpqpsol.K(mpqpsol.i1(kk):mpqpsol.i2(kk));
         if H0*theta(k,:)'<=K0
         ii=kk; memory(k,1)=kk;count0=count0+1;
         end
        memory(k,2)=count0;
    end
    F0=mpqpsol.F((ii-1)*Nc+1:ii*Nc,:);
```

```
    G0=mpqpsol.G((ii-1)*Nc+1:ii*Nc);

duk=F0*theta(k,:)'+G0;
du(nb+2)=duk(1);
u(nb+2)=u(nb+1)+du(nb+2);

duu(k)=du(nb+2);
uu(k)=u(nb+2);

uuu(nb+2)=u(nb+2)-offset;
end


figure
subplot(2,1,1)
plot(Ts:Ts:Ts*tsim,uu,Ts:Ts:Ts*tsim,duu)
subplot(2,1,2)
plot(Ts:Ts:Ts*tsim,yy,Ts:Ts:Ts*tsim,ref(1:tsim))
```

Código A.9 Código MATLAB® : "InitialValues2". Crea el problema MPQP.

```
sysd = c2d(sys,Ts,'zoh'); % Discrete system
[fil,col]=size(sysd.num{1});
B=sysd.num{1}(2:col);
A=sysd.den{1};

nb=length(B)-1;
na=length(A)-1;


%Calculation of Fj
Avir=conv(A,[1 -1]);
F=zeros(N,na+1);
poluno=[1 zeros(1,na+1)];
F1aux=poluno-Avir;
F(1,:)=F1aux(1,2:na+2);
for j=1:(N-1)
    for i=0:na-1
        F(j+1,i+1)=F(j,i+2)-F(j,1)*Avir(1,i+2);
    end
    F(j+1,na+1)=-F(j,1)*Avir(1,na+2);
end

%Calculation of Ej
E=zeros(N,N);
E(1,1)=1;
for j=1:(N-1)
    E(j+1,:)=E(j,:);
    E(j+1,j+1)=F(j,1);
end

%Calculation of Gj=Ej*Denj
Gpols=zeros(N,N+nb);
for j=1:N
    Gpols(j,:)=conv(E(j,:),B);
end
```

```
%Calculation of G Matrix
G2=zeros(N,N);
for j=1:N
    kk=1;
    for i=j:-1:1
        G2(j,kk)=Gpols(j,i);
        kk=kk+1;
    end
end
G=G2(:,1:Nc);

%Calculation of G' Matrix
Gprima=[];
if nb>0
    Gprima=zeros(N,nb);
    for j=1:N
        Gprima(j,:)=Gpols(j,1+j:(j+nb));
    end
end
%% mpqp Parametres
Fx=[Gprima';F']';
T=tril(ones(Nc),0);
vec1(1:N,1)=1;
vec0(1:N,1)=0;
vec11(1:Nc,1)=1;
vec00(1:Nc,1)=0;

H=2*(G'*(lambda2*eye(N))*G+lambda*eye(Nc));
C=2*lambda2*G'*[Fx,vec0,-vec1];
AA=[G;-G;T;-T];
b=[ymax;-ymin;umax;-umin];
Sth= [ -Fx,              vec0,vec0;
        Fx,              vec0,vec0;
       vec00,vec00,vec00,vec00,-vec11,vec00;
       vec00,vec00,vec00,vec00, vec11,vec00];

thetamin=[dumin0;ymin0;ymin0;ymin0;umin0;refmin0];
thetamax=[dumax0;ymax0;ymax0;ymax0;umax0;refmax0];
```

A.6 Controlador explícito GPC con restricciones Gain-Scheduling para Sistema Real

Código A.10 Código MATLAB® : Control PWA, GPC Explícito para Sistema Real mediante OPC.

```
%% GPC_GS_OFFLINE
% GPC-REAL SYSTEM
% OPC Connection
clear
clc


%% Parameters:
lambda=0.1; % DU COST CONSTANT
lambda2=50; % ERROR COST CONSTANT
```

```
N=7;      % PREDICTION HORIZONT
Nc=1;     % CONTROL HORIZONT
Ts=1;     % Sample time: IMPORTANT! DEFINE THE DISCRET TIME AND CONTROL TIME

umin0=6.4;
umax0=75;
ymin0=0;
ymax0=15;
dumin0=-umax0;
dumax0=umax0;
refmin0=ymin0;
refmax0=ymax0;

%Constraints:
umin(1:Nc,1)=umin0;
umax(1:Nc,1)=umax0;
ymin(1:N,1)=ymin0;
ymax(1:N,1)=ymax0;


% MP 3 Models
num=0.01152*.77;
den=[1 3.507 0.004685];
sys=tf(num,den);

InitialValues2
mpqpsol1=mpqp(H,C,AA,b,Sth,thetamin,thetamax,1,'quadprog');

num=0.01152;
den=[1 3.507 0.004685];
sys=tf(num,den);

InitialValues2
mpqpsol2=mpqp(H,C,AA,b,Sth,thetamin,thetamax,1,'quadprog');

num=0.002274;
den=[1 0.8613 0.0007163];
sys=tf(num,den);

InitialValues2
mpqpsol3=mpqp(H,C,AA,b,Sth,thetamin,thetamax,1,'quadprog');

OPC_Matlab
%% REAL TRIAL:
ii=1;
ref=trial1;
u=ones((nb+2),1);
du=zeros((nb+2),1);
y=zeros((na+1),1);
time=0;

GPC=read(GPC0);
Value=GPC.Value

while(Value==0)
tic
time=time+1;
```

```
for i=1:na
    y(i)=y(i+1);
end
for i=1:nb+1
    du(i)=du(i+1);
end
for i=1:nb+1
    u(i)=u(i+1);
end

if y(na)<6
mpqpsol=mpqpsol1;
memory(k,3)=1;
end
if y(na)>=6 && y(na)<8
mpqpsol=mpqpsol2;
memory(k,3)=2;
end
if y(na)>=8
mpqpsol=mpqpsol3;
memory(k,3)=3;
end

y0=read(y0_1);
y(na+1)=(double(y0.Value))/1000;
yy(time)=y(na+1);

y0real=read(y0_real);
yreal(time)=(double(y0real.Value))/1000;

rr(time)=ref(time);

theta(time,:)=[du(nb+1) y(na+1) y(na) y(na-1) u(nb+1) rr(time)];
count0=0;
    for kk=1:length(mpqpsol.i1)
        H0=mpqpsol.H(mpqpsol.i1(kk):mpqpsol.i2(kk),:);
        K0=mpqpsol.K(mpqpsol.i1(kk):mpqpsol.i2(kk),:);
         if H0*theta(time,:)'<=K0
         ii=kk
         memory(time,1)=kk; count0=count0+1;
         end
        clear H0 K0
         memory(time,2)=count0;
    end
    F0=mpqpsol.F((ii-1)*Nc+1:ii*Nc,:);
    G0=mpqpsol.G((ii-1)*Nc+1:ii*Nc);

duk=F0*theta(time,:)'+G0;
du(nb+2)=duk(1);
duu(time)=du(nb+2);

u(nb+2)=u(nb+1)+du(nb+2);
if u(nb+2)<0
    u(nb+2)=0;
end
uu(time)=u(nb+2);
```

```
write(u0_1,(u(nb+2)*10));

GPC=read(GPC0);
Value=GPC.Value;
interval=toc;
pause(Ts-interval);
end

figure
subplot(2,1,1)
plot(Ts:Ts:Ts*(time),uu,Ts:Ts:Ts*(time),duu);
subplot(2,1,2)
plot(Ts:Ts:Ts*(time),yreal,Ts:Ts:Ts*(time),yy,Ts:Ts:Ts*(time),rr);
```

Apéndice B
Código SCL, TIA PORTAL®

B.1 Controlador explícito GPC con restricciones Gain-Scheduling

Código B.1 Código SCL: Control GPC Implementado en PLC.

```
% PROGRAMA QUE SE EJECUTA CADA 1 SEGUNDO= SAMPLE TIME.
% Actualización de las variables. Guardando las del instante anterior como las
    pasadas y guardando las nuevan medidas de los sensors como las actuales:
%Pasadas:
#DU1:= #DU0;
#Frecuency1:=#Frecuency0;
#Volume2:=#Volume1;
#Volume1:=#Volume0;
%Actuales:
#Volume0:=#Volume;
#Reference0:=#Reference;
% Actualización del vector de estado:
#Theta[0]:=#DU1;
#Theta[1]:=#Volume0;
#Theta[2]:=#Volume1;
#Theta[3]:=#Volume2;
#Theta[4]:=#Frecuency1;
#Theta[5]:=#Reference0;


IF #Volume0 < 6 THEN
% Comprobación región tras región para saber en cual nos encontramos
FOR #count := 0 TO 12 DO % 13 regiones
#dimension := #I1[1,#count]- #I1[0,#count]; % dimensión de las matrices en la
    región #count
#flag:=#dimension+1;
#min :=#I1[0,#count]-1;

FOR #ii := 0 TO #dimension DO % Comprobamos la inecuación fila por fila, en
    función de la dimensión.
#N[#ii] := #H1[0,(#ii+#min)]*#Theta[0]+ #H1[1,(#ii+#min)]*#Theta[1]+#H1[2,(#ii
    +#min)]*#Theta[2]+#H1[3,(#ii+#min)]*#Theta[3]+#H1[4,(#ii+#min)]*#Theta[4]+#
    H1[5,(#ii+#min)]*#Theta[5];
IF #N[#ii] <= #K1[0,(#ii+#min)] THEN % Si una fila se cumple restamos un flag:
#flag:=#flag-1;
END_IF;
```

```
IF #flag=0 THEN % Si todas las filas se han cumplido flag =0. Estamos en la
    región correcta.
#region:= #count;
END_IF;
END_FOR;
END_FOR;

% Calculo de la acción de control: du= F(region)*thetha + G(region)
#M:= #F1[0,#region]*#Theta[0]+ #F1[1,#region]*#Theta[1] + #F1[2,#region]*#Theta
    [2] + #F1[3,#region]*#Theta[3] +#F1[4,#region]*#Theta[4] +#F1[5,#region]*#
    Theta[5];
#DUK:= #M + #G1[0,#region];
#DU0:=#DUK;
#Frecuency0 := #Frecuency1 + #DU0;
% Salidas del bloque:
#Frecuency := #Frecuency0;
#Memory := #region;
#model:=1;
END_IF;


IF (#Volume0 >= 6 & #Volume0 <= 8) THEN
FOR #count := 0 TO 12 DO
#dimension := #I2[1,#count]- #I2[0,#count];
#flag:=#dimension+1;
#min :=#I2[0,#count]-1;

FOR #ii := 0 TO #dimension DO
#N[#ii] := #H2[0,(#ii+#min)]*#Theta[0]+ #H2[1,(#ii+#min)]*#Theta[1]+#H2[2,(#ii
    +#min)]*#Theta[2]+#H2[3,(#ii+#min)]*#Theta[3]+#H2[4,(#ii+#min)]*#Theta[4]+#
    H2[5,(#ii+#min)]*#Theta[5];
IF #N[#ii] < #K2[0,(#ii+#min)] THEN
#flag:=#flag-1;
END_IF;
IF #flag=0 THEN
#region:= #count;
END_IF;
END_FOR;
END_FOR;

#M:= #F2[0,#region]*#Theta[0]+ #F2[1,#region]*#Theta[1] + #F2[2,#region]*#Theta
    [2] + #F2[3,#region]*#Theta[3] +#F2[4,#region]*#Theta[4] +#F2[5,#region]*#
    Theta[5];
#DUK:= #M + #G2[0,#region];
#DU0:=#DUK;
#Frecuency0 := #Frecuency1 + #DU0;
#Frecuency := #Frecuency0;
#Memory := #region;
#model:=2;
END_IF;


IF #Volume0 > 8 THEN
FOR #count := 0 TO 16 DO
#dimension := #I3[1,#count]- #I3[0,#count];
#flag:=#dimension+1;
#min :=#I3[0,#count]-1;
```

```
FOR #ii := 0 TO #dimension DO
#N0[#ii] := #H3[0,(#ii+#min)]**Theta[0]+ #H3[1,(#ii+#min)]**Theta[1]+#H3[2,(#ii
    +#min)]**Theta[2]+#H3[3,(#ii+#min)]**Theta[3]+#H3[4,(#ii+#min)]**Theta[4]+#
    H3[5,(#ii+#min)]**Theta[5];
IF #N0[#ii] < #K3[0,(#ii+#min)] THEN
#flag:=#flag-1;
END_IF;
IF #flag=0 THEN
#region:= #count;
END_IF;
END_FOR;
END_FOR;

#M:= #F3[0,#region]**Theta[0]+ #F3[1,#region]**Theta[1] + #F3[2,#region]**Theta
    [2] + #F3[3,#region]**Theta[3] +#F3[4,#region]**Theta[4] +#F3[5,#region]**#
    Theta[5];
#DUK:= #M + #G3[0,#region];
#DU0:=#DUK;
#Frecuency0 := #Frecuency1 + #DU0;
#Frecuency := #Frecuency0;
#Memory := #region;
#model:=3;
END_IF;
```

Índice de Figuras

Índice de Códigos

Bibliografía

[1] A. Bemporad, *Hybrid toolbox - user's guide*, 2004.

[2] C. Bordons, *Control predictivo: metodología,tecnología y nuevas perspectivas*, 2000.

[3] E.F. Camacho and C. Bordons, *Model predictive control*, Springer-Verlag, 2004.

[4] G. Cavalcanti, *Design and analysis of multivariable predictive control applied to an oil-water-gas separator: a polynomial approach*, 2001.

[5] A. Bemporad F. Borrelli and Morari M, *Constrained optimal control for linear and hybrid system*, Springer, 2010.

[6] S. Joe Qin and Thomas A. Badgwell, *An overview of industrial model predictive control technology*, 1997.

[7] D.R. Ramirez and C. Bordons, *Apuntes de ingeniería de control*, 2005.

[8] A.A. Cornelio S. Roshany-Yamchi, R.R Negenborn, *Nash-based distributed mpc for multi-rate systems in distributed mpc made easy*, Springer, 2015.

[9] Siemens, *Manual de sistema, controlador programable simatic s7-1200*, 2009.