Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Investigación en Ingeniería
De Procesos y Sistemas Industriales

MASTER EN INVESTIGACIÓN EN INGENIERÍA DE PROCESOS Y SISTEMAS INDUSTRIALES

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Implementación de controladores predictivos en dispositivos programables industriales

Autor: Dª. Laura de Frutos Bolzoni
Tutor: D. César de Prada Moraga
Tutor 2: D. Rubén Martí Martínez

Valladolid, Septiembre, 2017

Resumen:

El objetivo del proyecto es la implementación de algunos algoritmos de control predictivo en dispositivos programables industriales (PLCs) para, una vez validados, pasar a su integración dentro del entorno de desarrollo de aplicaciones de control industriales desarrollado y utilizado por el CERN (UNICOS). Los controladores implementados deben satisfacer dos necesidades concretas. En primer lugar, se requiere un controlador para sistemas con grandes retardos de una entrada y una salida y, en segundo lugar, un controlador para pequeños sistemas multivariable, de dos o tres entradas y salidas. Los algoritmos que se han seleccionado e implementado en el presente proyecto son el Predictive Functional Control, para el primer caso, y el Dynamic Matrix Control para el segundo.

Palabras clave:

- Automatización industrial
- Autómatas programables
- Control predictivo basado en modelos
- Predictive Functional Control
- Dynamic Matrix Control

ÍNDICE DE CONTENIDOS

Capítulo 1 . Introducción y objetivos 1
1. Motivación 1
2. Conseil Européen pour la Recherche Nucléaire (CERN) 1
2.1. Unified Industrial Control System (UNICOS) 1
2.1.1. UNICOS *Continuous Process Control* (UCPC) 3
2.1.2. UNICOS *Application Builder* (UAB) 4
3. Ampliación de la librería de control de UNICOS 6
4. Caso de aplicación del MPC 6
Capítulo 2 . Marco Teórico 9
1. Control Predictivo basado en Modelos (MPC) 9
2. Formulación del control predictivo basado en modelos 10
3. Elementos del control predictivo basado en modelos 12
3.1. Modelo de predicción 12
3.1.1. Respuesta impulso 13
3.1.2. Respuesta salto 14
3.1.3. Función de transferencia 14
3.1.4. Modelo en espacio de estados 15
3.1.5. Otros modelos 15
3.2. Predicción de la salida del proceso: Respuestas libre y forzada 16
3.3. Función de coste 16
3.3.1. Parámetros de configuración 17
3.3.2. Trayectoria de referencia 17
3.3.3. Restricciones 18
3.4. Obtención de la ley de control 18
4. Revisión de algunos algoritmos MPC 19
4.1. *Predictive Functional Control* 19
4.1.1. PFC aplicado a un sistema de primer orden 19
4.1.2. Incorporación de perturbaciones al modelo 21
4.1.1. Incorporación de retardo al modelo 22

4.1.2. Conclusiones 23
4.2. Dynamic Matrix Control 23
4.2.1. DMC aplicado a un sistema SISO (*Single Input Single Output*) 23
4.2.1.1. Acción de control 25
4.2.1.2. Trayectoria de referencia 26
4.2.1.3. Predicción de la salida 26
4.2.2. Sistema MIMO (*Multiple Inputs Multiple Ouputs*) 27
4.2.3. Optimizador 28
4.2.3.1. Programación cuadrática 29
4.2.3.2. QP con restricciones lineales de igualdad 29
4.2.3.3. QP con restricciones lineales de desigualdad 29
4.2.3.4. Conjuntos activos 30
4.2.3.5. Métodos primal-dual 30
4.2.3.6. Método QP de Hildreth 31
4.2.3.7. Conclusión 32
Capítulo 3 . Implementación 33
1. Introducción 33
2. Predictive Functional Control 33
2.1. Algoritmo para sistemas de primer orden con retardo 33
2.2. Integración del algoritmo en UNICOS 34
2.2.1. Características del objeto *Controller* 35
2.2.2. Modos de operación 36
2.2.3. Estados de trabajo 37
2.2.4. UNICOS Aplication Builder 38
2.2.5. WinCC-OA. Interfaz con el usuario 39
3. Dynamic Matrix Control 42
3.1. Preparación del problema QP 43
3.1.1. Acción de control 44
3.1.2. Matriz G 44
3.1.3. Matriz Hessiana 45
3.1.4. Trayectoria de referencia 45
3.1.5. Respuesta libre 46

3.1.6. Vector gradiente 47
3.1.7. Restricciones 48
3.2. Relajación de las restricciones 51
3.2.1. Variables de decisión 52
3.2.2. Matriz H 52
3.2.3. Gradiente 53
3.2.4. Restricciones 53
3.3. Optimizador: QP Hildreth 54
3.4. Ejecución del controlador DMC 55
3.4.1. Matrices constantes 56
3.4.2. Matrices variables 56
3.5. Implementación del DMC en el PLC 57
Capítulo 4 . Validación de resultados 61
1. Introducción 61
2. Predictive Functional Control 61
2.1. Identificación del sistema 62
2.2. Validación del controlador en EcosimPro 63
2.2.1. Sintonía del controlador 63
2.2.2. Validación del controlador sobre sistemas con grandes retardo ... 67
2.3. Validación del controlador en una planta real 68
3. Dynamic Matrix Control 69
3.1. Herramienta Integrada para Total Optimización (HITO) 69
3.2. Validación del controlador en Matlab 70
3.2.1. Identificación del sistema 71
3.2.2. Sintonía de parámetros 72
3.2.2.1. Pesos en la función de costes 72
3.2.2.2. Horizontes del controlador 72
3.2.2.3. Límites de las variables 73
3.2.2.4. Trayectoria de referencia 73
3.2.2.5. Número máximo de iteraciones del optimizador 74
3.2.3. Comportamiento del controlador 74
3.2.3.1. Seguimiento de referencia 74

3.2.3.2. Rechazo de perturbaciones 76
3.3. Comparación de los resultados obtenidos con HITO 77
3.4. Validación del controlador en el PLC 79
3.4.1. Ocupación de la memoria 80
3.4.2. Tiempo de ciclo 81
3.4.3. Comparación de los resultados obtenidos con el PLC y Matlab 83
Capítulo 5 . Conclusiones y trabajo futuro 87
1. Conclusiones 87
2. Trabajo futuro 89
Bibliografía 91
Anexo 1: 93
Código fuente del PFC: EcosimPro 93
Anexo 2: 99
Código fuente del DMC: Matlab 99
Anexo 3: 131
Código fuente del DMC: Step7 131

ÍNDICE DE FIGURAS

Figura 1.1 – Mapa de los aceleradores del CERN 2
Figura 1.2 – Arquitectura de un sistema de control aplicada a UNICOS 2
Figura 1.3 – Paquetes que conforman UNICOS 3
Figura 1.4 – Diseño de un lazo de control aplicando UNICOS 4
Figura 1.5 – Funcionamiento de la herramienta UAB 5
Figura 1.6 – Esquema de control de un sistema de climatización 6
Figura 2.1 – Estrategia del Control Predictivo Basado en Modelos 11
Figura 2.2 – Estructura de un lazo de regulación con MPC 11
Figura 2.3 – Respuesta impulso y respuesta salto 14
Figura 2.4 – Respuesta libre y forzada 16
Figura 2.5 – Trayectoria de referencia 18
Figura 2.6 – Estrategia del PFC aplicada a un sistema de primer orden 20
Figura 2.7 – Acciones y perturbaciones sobre el sistema que se consideran en el DMC 23
Figura 3.1 – Estructura genérica de un objeto UNICOS 35
Figura 3.2 – Modos de operación y estados de trabajo en el objeto Controller . 36
Figura 3.3 – Hoja de especificaciones diseñada para el PFC 38
Figura 3.4 – Panel principal del PFC 39
Figura 3.5 – Identificación del sistema 40
Figura 3.6 – Configuración del tiempo de muestreo y el tipo de acción 40
Figura 3.7 – Configuración del valor de la referencia y la variable manipulada ... 41
Figura 3.8 – Definición de límites 41
Figura 3.9 – Diagrama de flujo del optimizador 54
Figura 3.10 – Diagrama de flujo de funcionamiento del DMC 55
Figura 4.1 – Lazo de control de presión 61
Figura 4.2 – Identificación del sistema 62
Figura 4.3 – Obtención de los parámetros del PID 63
Figura 4.4 – Lazo de control en EcosimPro 63

Figura 4.5 – Comparativa PFC y PID: Caso ideal sin retardo en el sistema 64
Figura 4.6 –Comparativa PFC y PID: Modelo del sistema con ruido y retardo 65
Figura 4.7 – Comparativa PFC y PID: $\lambda = 0.5$ y $N = 10$ 66
Figura 4.8 – Comparativa PFC y PID: $\lambda = 0.95$ y $N = 30$ 66
Figura 4.9 – Comparativa PFC y PID: $\tau = 50$ y $k = 0.0255$ 67
Figura 4.10 – Ensayo PFC: $d = 30s,\ \lambda = 0.80$ y $N = 30$ 67
Figura 4.11 – Ensayo sobre planta real 68
Figura 4.12 – Sistema del reactor 70
Figura 4.13 – Identificación de los modelos entrada/salida 71
Figura 4.14 – Identificación de los modelos perturbación/salida 71
Figura 4.15 – Ensayo DMC. Seguimiento de la referencia (Salidas) 75
Figura 4.16 – Ensayo DMC. Seguimiento de la referencia (Acciones de control) . 75
Figura 4.17 – Eayo DMC. Rechazo de perturbaciones (Salidas) 76
Figura 4.18 – Ensayo DMC. Rechazo de perturbaciones (Acción de control) 77
Figura 4.19 – Ensayo DMC. Rechazo de perturbaciones (Perturbaciones) 77
Figura 4.20 – Comparativa entre controlador en Matlab e HITO (Salidas) 78
Figura 4.21 – Comparativa entre controlador en Matlab e HITO (Acciones de control) 78
Figura 4.22 – Comparativa entre controlador en Matlab e HITO (Perturbaciones) 79
Figura 4.23 – Ocupación del DMC en la memoria de una CPU 315-2 PN/DP 81
Figura 4.24 – Tiempo mínimo y máximo de ejecución del controlador 82

Capítulo 1 . INTRODUCCIÓN Y OBJETIVOS

1. Motivación

El uso del control predictivo es una realidad dentro del mundo de la industria del proceso. Su utilización se ha ido extendiendo en el entorno industrial debido a la sencillez de sus conceptos y a un funcionamiento intuitivo y robusto. Desde hace más de diez años, algunas de las marcas más conocidas de controladores industriales incluyen implementaciones de esta técnica, como son Honeywell, DeltaV de Emerson y PCS7 de Siemens, mientras que muchas otras aún mantienen solamente el PID como única alternativa de control en sus librerías. Este último caso es en el que se incluyen los entornos de desarrollo clásico de STEP7 y TIA Portal de Siemens.

Haciendo un análisis de las alternativas de controladores que han introducido el control predictivo en el mercado, se puede apreciar que son las gamas más altas las que lo ofrecen. Debido al coste de estas aplicaciones, el uso del control predictivo no está más extendido fuera de la industria del proceso y para pequeños sistemas multivariable.

2. Conseil Européen pour la Recherche Nucléaire (CERN)

El CERN, cuyas siglas se corresponden con su nombre en francés "*Conseil Européen pour la Recherche Nucléaire*" es la Organización Europea para la Investigación Nuclear. Se trata del mayor laboratorio de investigación en física de partículas del mundo, situado en la frontera entre Francia y Suiza. En sus instalaciones cuenta con un complejo de aceleradores de partículas entre los que destaca el LHC (*Large Hadron Collider*), un acelerador y colisionador de partículas.

2.1. Unified Industrial Control System (UNICOS)

UNICOS es un *framework* desarrollado por el CERN (*Conseil Européen pour la Recherche Nucléaire*) para el desarrollo de aplicaciones de control industriales. Este sistema comprende las dos capas superiores de un sistema de control clásico: supervisión y control. La propuesta de UNICOS es un método para diseñar y desarrollar aplicaciones de control basadas en el uso de sistemas de control de Siemens y Scheneider principalmente.

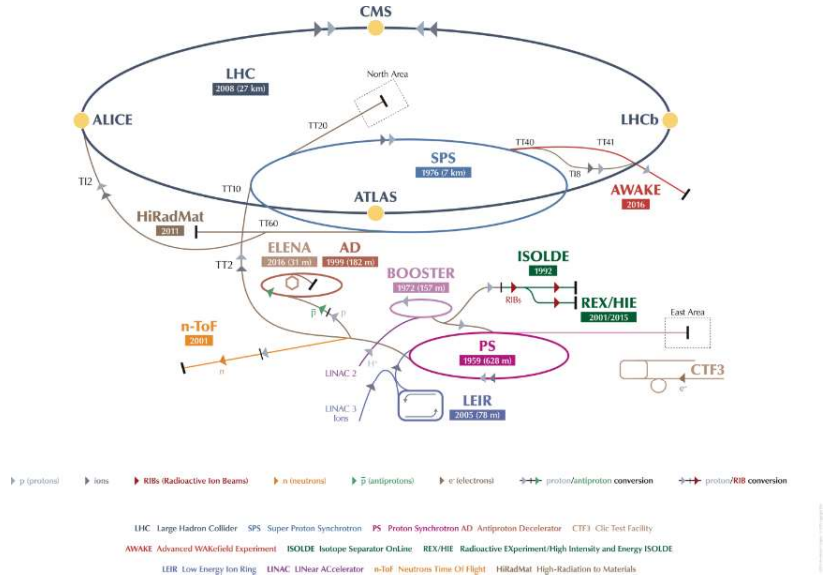


Figura 1.1 – Mapa de los aceleradores del CERN

Como se muestra en la Figura 1.2, UNICOS abarca las capas de supervisión, control y campo, así como las estructuras de comunicaciones necesarias para la interacción entre ellas. El objetivo de UNICOS es estandarizar el desarrollo de aplicaciones de control en el CERN para:

- Hacer hincapié en las buenas prácticas de diseño y operación de las aplicaciones de control de procesos.
- Reducir el coste de la automatización de procesos continuos (por ejemplo, refrigeración, climatización...).
- Optimizar los esfuerzos de ingeniería mediante, por ejemplo, el uso de herramientas de generación automática de código.

La aplicación del sistema UNICOS aporta una gran cantidad de ventajas, tanto a nivel de desarrollado como de operación.

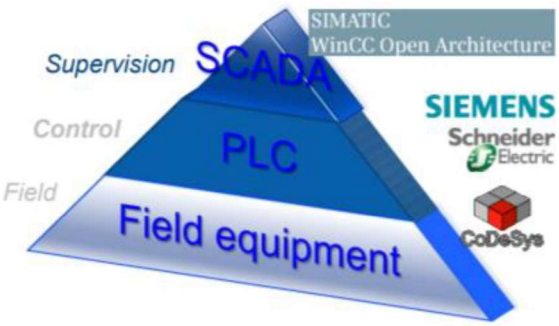


Figura 1.2 – Arquitectura de un sistema de control aplicada a UNICOS

2.1.1. UNICOS *Continuous Process Control* (UCPC)

UCPC proporciona una metodología, una librería de objetos y una serie de herramientas para el diseño y la implementación de aplicaciones de control industrial, tanto en las capas control como de supervisión. Entre las plataformas de desarrollo disponibles en UCPC se encuentran PLCs (*Programmable Logic Controller*) de Siemens y Schneider a nivel de control y en la capa de supervisión WinCC OA (antes PVSS) y WINCC Flexible.

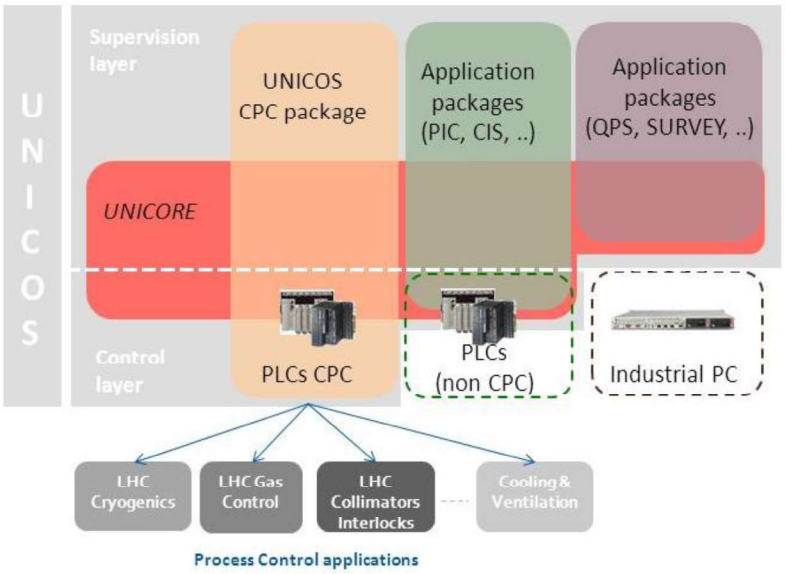


Figura 1.3 – Paquetes que conforman UNICOS

El paquete UNICOS-CPC contiene un conjunto de objetos clasificados por su funcionalidad. Hay cuatro categorías principales de objetos: objetos de E/S, de interfaz, de campo y de control:

- Objetos de E/S: Entrada/Salida de información. Estos pueden referirse a los PLC de la periferia, los buses de campo o simplemente la memoria interna.

  - *Digital Input*
  - *Analog Input*
  - *Analog Input Real*
  - *Digital Output*
  - *Analog Output*
  - *Analog Output Real*

- Objetos interfaz: parametrización y el estado (también pueden incluir información de la periferia).

  - *Analog Parameter*
  - *Digital Parameter*
  - *Word Parameter*
  - *Analog Status*
  - *Word Status*

- Objetos de campo: Modelo de los equipos físicos de campo (por ejemplo: válvulas, motores,...). Los objetos de campo están siempre conectados a los objetos de E/S.

  - *Local*
  - *OnOff*
  - *Analog*
  - *Anadig*
  - *AnaDO*
  - *Stepping Motor*

- Objetos de control: las acciones de control lógico proceso de generación de alarmas y enclavamientos y control de retroalimentación. Siempre actúan sobre objetos de campo

  - *Process Control Object*
  - *Controller*
  - *Digital Alarm*
  - *Analog Alarm*

La estructura de un sistema de control en UNICOS está fundamentada en la jerarquización del sistema en base a estos objetos elementales, en los cuales quedan identificados cada uno de los elementos que componen el proceso a controlar. Estos objetos incluyen la forma de interrelacionarse entre sí y, además, los datos relativos a la parte del proceso que controlan.

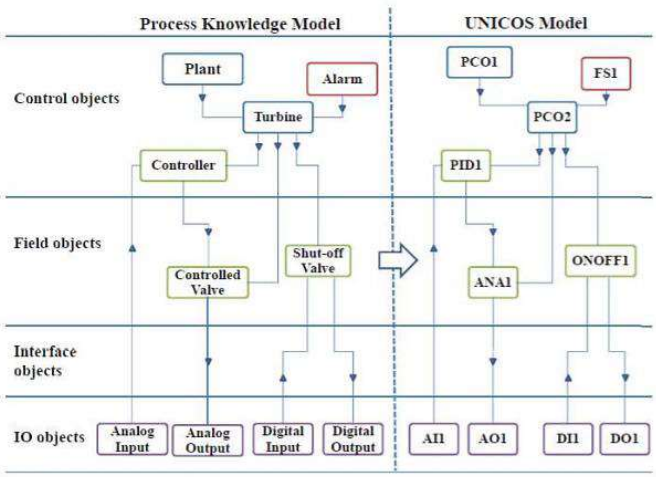


Figura 1.4 – Diseño de un lazo de control aplicando UNICOS

2.1.2. UNICOS *Application Builder* (UAB)

La base de la filosofía UNICOS de unificación y simplificación en el diseño y desarrollo de aplicaciones CPC se encuentra en el UAB. Esta herramienta es un generador de código que, a partir de una hoja de especificaciones introducida por el usuario, genera la aplicación de control para el PLC y el SCADA (*Supervisory Control And Data Acquisition*).

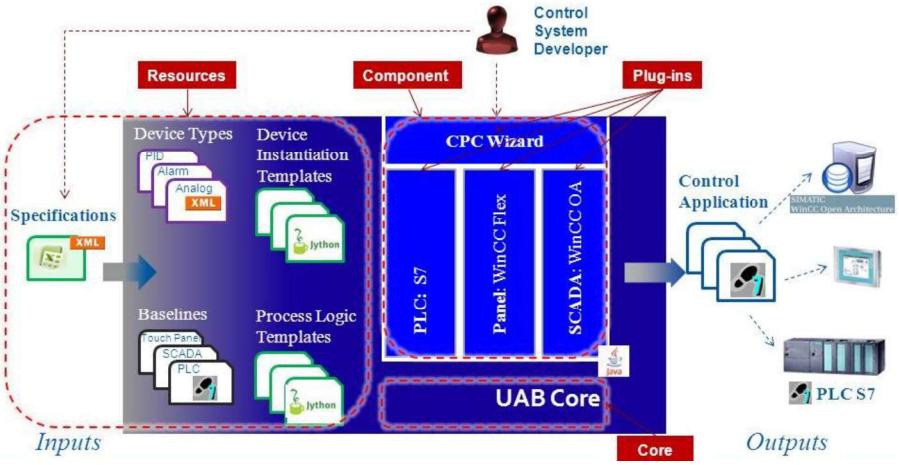


Figura 1.5 – Funcionamiento de la herramienta UAB

La hoja de especificaciones consta de una pestaña dedicada a cada tipo de objeto de la lista anterior. De este modo, una vez identificada la estructura de la planta en objetos elementales de UNICOS, sólo resta parametrizar cada una de dichas variables completando los campos requeridos por cada elemento. Al documento de especificaciones se puede añadir, para cada objeto, un archivo en Python que contiene la lógica específica del objeto en cuestión, como pueden ser por ejemplo los *interlocks.*

Una vez completado este paso, sólo resta ejecutar el asistente UAB. El generador UAB emplea una serie de *plugins* para crear los archivos que después se importarán en el PLC y en el SCADA.

- Generador de instancias S7 (*S7 Instance Code Generator Plugin*): encargado de generar código que contiene las instancias de los objetos para el PLC Siemens, así como las estructuras de comunicación con WinCC-OA.

- Generador de lógica S7 (*S7 Logic Generator Plugin*): encargado de generar el código de lógica para el PLC Siemens, es decir, todas las interconexiones entre los PCOs y los objetos de campo del proyecto a controlar

- Generador de instancias PVSS (*PVSS Instance Generator*): encargado de crear el fichero de importación para el SCADA.

Si después de generar el proyecto se introducen cambios en la lógica de los objetos en el PLC, se debe hacer una ingeniería inversa e incorporar dichos cambios en la hoja de especificaciones, de manera que si se vuelve a generar dicho proyecto no se pierdan las modificaciones realizadas.

3. Ampliación de la librería de control de UNICOS

El objeto *Controller* de la librería de UNICOS, anteriormente mencionado, sólo contiene a día de hoy un único tipo de controlador, el PID. Si bien es cierto que éste puede cubrir satisfactoriamente la mayoría de las necesidades que se encuentran en la industria, algunos lazos de control requieren consideraciones especiales.

El control predictivo es una de las técnicas más populares de control avanzado a la hora de afrontar sistemas con dinámicas complejas en sus respuestas o lazos multivariables con fuertes acoplamientos. Es por ello que se ha elegido este tipo de control a la hora de ampliar la librería de control de UNICOS.

4. Caso de aplicación del MPC

Un claro ejemplo de aplicación para la aplicación del control predictivo dentro y fuera del ámbito industrial son los sistemas de climatización, también llamados HVAC (*Heating Cooling and Ventilation*). El control de estos sistemas suele estar implementado en PLCs pequeños, con poca memoria y capacidad de cálculo, pero que están muy presentes en cualquier entorno y por ello conviene tenerlos en cuenta.

La Figura 1.6 muestra el esquema de control de un sistema real de HVAC en el cual se regula la temperatura y la humedad de una sala actuando sobre la apertura de dos válvulas, una que regula la circulación de agua caliente y otra de agua fría en sendos intercambiadores.

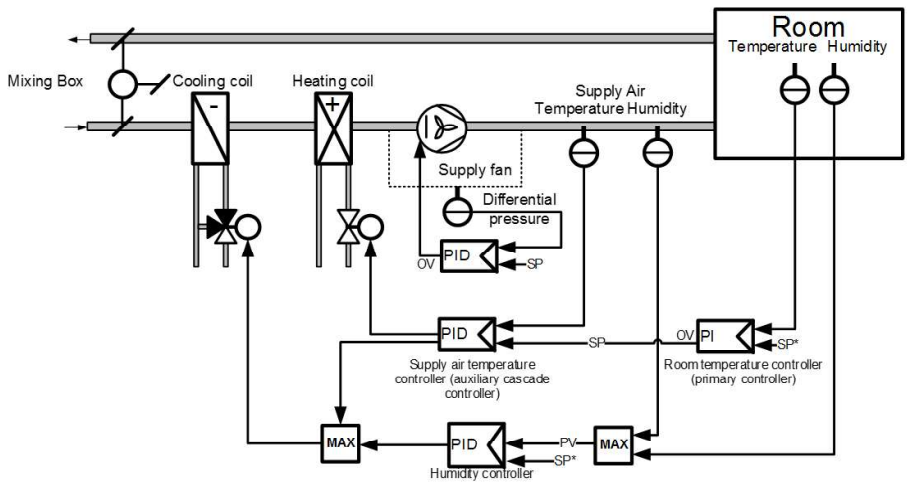


Figura 1.6 – Esquema de control de un sistema de climatización

Se trata, por lo tanto, de un sistema de dos entradas, temperatura y humedad en la sala, dos salidas, las dos válvulas, y dos perturbaciones, temperatura y humedad del aire suministrado en el conducto de aire.

La estructura de control que se aplica se compone de dos reguladores PID en cascada con un *split range* al final para el control de la temperatura que actúa sobre ambas válvulas y un PID para el control de humedad que sólo actúa sobre la válvula de agua fría en el caso de que el controlador de temperatura no esté solicitando un nivel de apertura superior.

Se trata por tanto de un pequeño sistema multivariable (2 entradas x 2 salidas) en el que hay un cierto grado de acoplamiento, puesto que la temperatura hace variar la humedad relativa del aire y para modificar la humedad se actúa sobre la temperatura. Además, una de las variables controladas, la temperatura, tiene prioridad sobre la otra, la humedad. Y por último, existen una serie de restricciones físicas sobre el valor que pueden tomar las variables manipuladas puesto que el rango de apertura y cierre de una válvula debe estar entre el 0 y el 100%.

Como se verá en capítulos posteriores, algunas de las grandes fortalezas del MPC residen en la gestión de este tipo de problemáticas.

Capítulo 2 . MARCO TEÓRICO

1. Control Predictivo basado en Modelos (MPC)

El control predictivo es una estrategia que, idealmente, trata de reflejar el razonamiento humano en tanto en cuanto una persona seleccionará la mejor acción a aplicar, teniendo en cuenta el plazo en que ha de ejecutarla y las limitaciones a las que está sujeto, y modifica estas decisiones en función de la nueva información que recibe.

Bajo la denominación de control predictivo basado en modelos (*Model Based Predictive Control* MBPC o MPC) se incluyen un amplio rango de estrategias de control basadas en la utilización de un modelo explícito del proceso para generar una señal de control mediante la optimización de una función objetivo. Están enmarcados, por tanto, dentro de los controladores óptimos.

Los diferentes métodos de control predictivo comparten una misma estructura en la que se pueden apreciar los siguientes elementos comunes:

- Un *modelo explícito del sistema* que será utilizado para predecir el comportamiento futuro del sistema en determinados instantes de tiempo.
- Una *ley de control* que junto con las restricciones del problema formulará el problema de optimización a resolver.
- Una estrategia de *horizonte deslizante* que implica la resolución del problema de optimización en cada instante de tiempo, siempre teniendo en cuenta el comportamiento futuro del sistema durante un cierto intervalo que se desplaza según avanza el tiempo actual.

El control predictivo presenta una serie de ventajas sobre otros métodos, entre las cuales se pueden enumerar las siguientes (Camacho & Bordons, 2007).

- Es una técnica particularmente atractiva para los operadores que requiere de pocos conocimientos de control porque los conceptos son muy intuitivos y la sintonización relativamente simple.
- Se puede utilizar para controlar una gran cantidad de procesos, desde muy simples hasta con dinámicas complejas, como son los sistemas con grandes tiempos de retardo, inestables, con fase no mínima o multivariables.
- Su carácter predictivo hace que compense intrínsecamente los tiempos muertos.

- Introduce un control anticipativo (*feedforward*) y de forma natural se compensan las perturbaciones medibles.
- La ley de control resultante es, en general, fácilmente implementable.
- Es muy útil cuando se conocen las referencias futuras, como ocurre en el caso de la robótica o los procesos por lotes.
- Permite tratar las restricciones de una forma sistemática y conceptualmente muy simple durante la fase de diseño.

Como es lógico, también presenta ciertos inconvenientes relacionados con el aumento en los cálculos que se requiere en comparación, por ejemplo, con un PID tradicional. En el caso de un proceso rápido, en el que la acción de control a aplicar se debe calcular en cada instante de muestreo, o si se quieren introducir restricciones en la ley de control, la carga computacional aumenta significativamente.

Aunque en el entorno de los ordenadores este problema no es trascendental debido a la gran capacidad de cálculo disponible en este tipo de dispositivos, dentro del mundo industrial la potencialidad de los computadores es más limitada. El principal problema es, sin embargo, la necesidad de un modelo adecuado del proceso a tratar. El diseño del algoritmo está basado en un conocimiento previo del modelo y, obviamente, el resultado obtenido por el controlador estará afectado en gran medida por la exactitud del modelo utilizado.

2. Formulación del control predictivo basado en modelos

La estrategia que sigue el control predictivo basado en modelos se puede expresar de forma genérica como se define en la Figura 2.1. La figura muestra la evolución del sistema hasta el instante actual t y la predicción del comportamiento futuro, así como algunos de los parámetros más relevantes de un MPC, como pueden ser:

- Horizonte de predicción (N): Indica la ventana de tiempo para la cual se calculan las salidas futuras $\hat{y}(t+k|t)$ para cada instante k.
- Horizonte de coincidencia (N_1, N_2): Indica la ventana de tiempo que se considera en el cálculo de la optimización. Se suele fijar un $N_1 \neq 0$, por ejemplo, en aquellos sistemas en que la dinámica incluye un retardo o una respuesta de fase no mínima.
- Horizonte de control (N_u): Indica el número cambios en la acción de control permitidos en un ciclo del controlador.

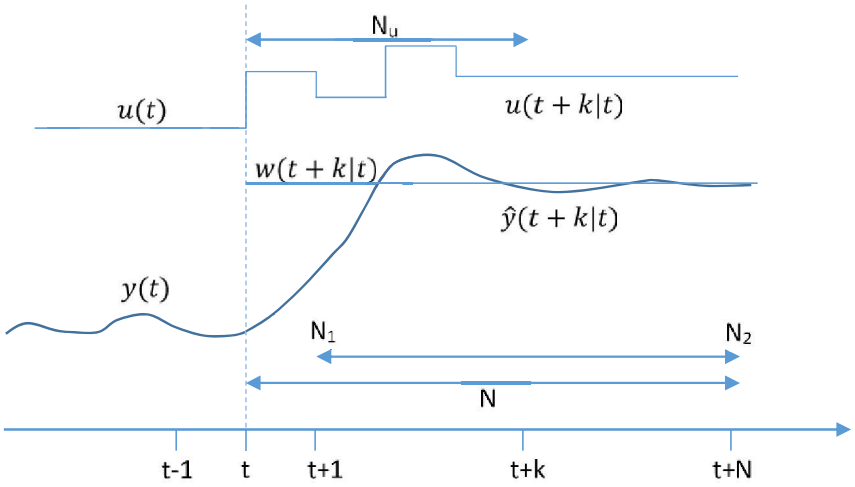


Figura 2.1 – Estrategia del Control Predictivo Basado en Modelos

La predicción de la evolución de cada variable controlada del sistema se realiza empleando un modelo matemático de éste y, basándose en el concepto de horizonte móvil (*receding horizon*), el MPC realiza, en primer lugar, las predicciones requeridas en una ventana temporal finita previamente definida (horizonte de predicción) para, a continuación, llevar a cabo la minimización de la función de coste. Finalmente, se aplica al sistema el primer valor calculado para cada variable controlada y se repite el proceso [Figura 2.2].

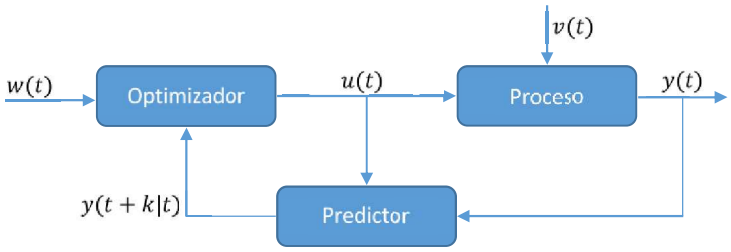


Figura 2.2 – Estructura de un lazo de regulación con MPC

El conjunto de acciones de control obtenidas de la resolución de dicho proceso responden habitualmente al criterio de la minimización del cuadrado de la función de error, es decir, la diferencia entre la salida predicha $\hat{y}(t+k|t)$ respecto de la trayectoria de referencia $w(t+t|k)$. También es habitual incluir en el problema de optimización el esfuerzo de control Δu ponderado por un factor β, resultando la siguiente función de coste [2.1]:

$$min\, J = \sum_{j=N_1}^{N_2} [\hat{y}(t+j) - w(t+j)]^2 + \sum_{j=0}^{N_{u-1}} [\beta \Delta u(t+j)]^2 \qquad [2.1]$$

Además a la función de costes se le pueden añadir una serie de restricciones sobre las variables controlada y manipulada [2.2] que delimitan la región factible de la solución del problema de optimización.

$$\hat{y}_{max} \geq \hat{y}(t+j) \geq \hat{y}_{min}$$

$$u_{max} \geq u(t+j) \geq u_{min} \qquad [2.2]$$

$$\Delta u_{max} \geq \Delta u(t+j) \geq \Delta u_{min}$$

A un nivel más alto que el meramente operativo, el uso de un controlador predictivo permite una optimización económica. Un sistema, normalmente, trabajará en su óptimo económico en un punto cercano a los límites de funcionamiento. La utilización del control predictivo reduce la varianza del sistema, haciendo posible pueda fijar en el proceso un mejor punto de funcionamiento desde este punto de vista.

3. Elementos del control predictivo basado en modelos

Para comprender correctamente el MPC es necesario tener en cuenta una serie de conceptos clave que se analizarán a continuación.

3.1. Modelo de predicción

El modelo de predicción es el resultado de la caracterización de un proceso real, que debe recoger toda la información relevante del comportamiento de un sistema, de forma que permita realizar predicciones que se asemejen lo más posible a la realidad.

Además de caracterizar la relación entre las entradas y las salidas del proceso, el modelo de predicción puede incluir el efecto de las perturbaciones medibles en el sistema. Esto permite que el controlador compense de manera anticipada su efecto sobre las salidas (control *feedforward*).

El modelo seleccionado puede ser lineal o no lineal, continuo o discreto y la elección del modelo será determinante a posteriori para decidir el algoritmo MPC a utilizar (*Predictive Functional Control*, *Dynamic Matrix Control*,…).

3.1.1. Respuesta impulso

Se denomina respuesta a impulso a la salida del sistema cuando la entrada es la función impulso unitario $\delta(n)$ [2.3]. Su uso práctico está limitado a sistemas asintóticamente estables, pero permite definir dinámicas complejas y no requiere un conocimiento previo del modelo. Si el proceso no cumpliera esa condición de estabilidad el modelo podría definirse pero requeriría de un número infinito de coeficientes.

La relación entre la salida y la entrada se expresa como:

$$y(t) = \sum_{i=1}^{\infty} h_i u(t-i) \qquad [2.3]$$

Donde $u(t)$ e $y(t)$ son variaciones sobre un punto de operación.

$$\begin{aligned} y(t) &= Y - Y_0 \\ u(t) &= U - U_0 \end{aligned} \qquad [2.4]$$

Teóricamente, la respuesta impulso contiene un número infinito de coeficientes pero, en la práctica, se trunca este valor y se consideran sólo N [2.5], debido a que a partir de un cierto número su aportación a la respuesta del sistema deja de ser significativa [2.5]. Este N es, normalmente, un número entre 30 y 50.

$$y(t) = \sum_{i=1}^{N} h_i u(t-i) \qquad [2.5]$$

Si el sistema a tratar es multivariable, se puede reflejar el efecto de las m diferentes entradas en la salida simplemente sumando la contribución de cada una de ellas sobre ésta [2.6].

$$y_i(t) = \sum_{k=1}^{m} \sum_{i=1}^{N} h_i^{kj} u^k(t-i) \qquad [2.6]$$

De este modo, la predicción de la salida en el instante $t+k$ resulta como sigue:

$$\hat{y}(t+k|t) = \sum_{i=1}^{N} h_i u(t+k-i|t) \qquad [2.7]$$

3.1.2. Respuesta salto

Esta respuesta se obtiene a partir de la aplicación de un salto de amplitud definida Δu en la entrada del sistema [2.8].

$$y(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-i) \qquad [2.8]$$

Si el sistema es asintóticamente estable, tras n periodos de muestreo los coeficientes g_i serán constantes y se puede deducir que:

$$y(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-\mathrm{i}) \approx \sum_{i=1}^{\infty} g_i \Delta u(t-i) + g_{n+1} \Delta u\big(t-(n+1)\big) \qquad [2.9]$$

El modelo en respuesta salto tiene las mismas ventajas y desventajas que la respuesta impulso puesto que, de hecho, un impulso puede considerarse como la diferencia entre dos saltos del mismo valor con un periodo de muestreo de diferencia [2.10].

$$h_i = g_i - g_{i-1} \qquad [2.10]$$

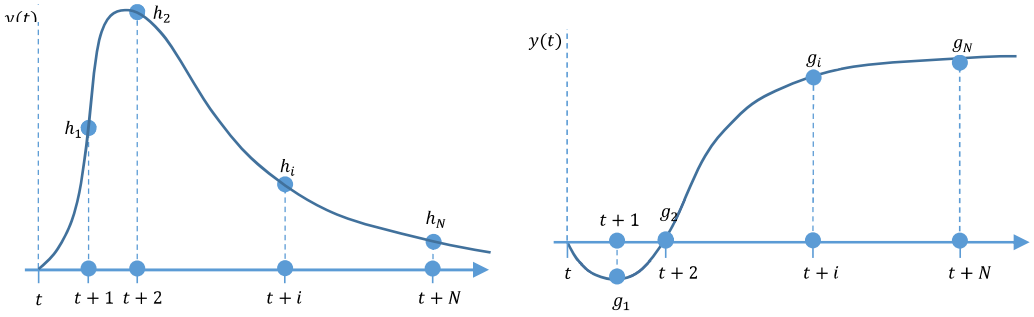


Figura 2.3 – Respuesta impulso y respuesta salto

3.1.3. Función de transferencia

Utilizando el concepto de función de transferencia donde $G = B/A$, la salida viene dada por:

$$y(t) = \frac{A(z^{-1})}{B(z^{-1})} u(t) \qquad [2.11]$$

Donde:

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}$$
$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n} \qquad [2.12]$$

De manera que la predicción vendrá dada por:

$$\hat{y}(t+k|t) = \frac{A(z^{-1})}{B(z^{-1})} u(t+k|t) \qquad [2.13]$$

Como se puede ver, este tipo de representación requiere de un número mucho menor de parámetros que el modelo en respuesta salto o impulso y además es válido para sistemas inestables, pero es importante tener una idea previa del orden del polinomio que caracterizará al sistema.

3.1.4. Modelo en espacio de estados

Una forma general de representación de espacios de estado de un sistema lineal se escribe de la siguiente forma:

$$x(t) = Ax(t-1) + Bu(t-1)$$
$$y(t) = Cx(t) \qquad [2.14]$$

Donde x son las variables de estado y A, B y C las matrices de estado del sistema. Para este modelo la predicción viene dada por:

$$\hat{y}(t+k|t) = C\hat{x}(t+k|t) = C[A^k x(t) + \sum_{i=1}^{k} A^{i-1} Bu(t+k-|t)] \qquad [2.15]$$

La representación en espacio de estados es válida para sistemas multivariables y permiten, si fuera necesario, analizar la estructura interna del proceso, aunque puede requerir de un observador de estados si éstos no son accesibles. Además, no es posible tener en cuenta los retardos de manera directa.

3.1.5. Otros modelos

En algunas aplicaciones, por ejemplo en casos de sistemas no lineales, se pueden utilizar modelos de redes neuronales o *fuzzy logic* (Zamarreño & Vega, 1999).

3.2. Predicción de la salida del proceso: Respuestas libre y forzada

La salida predicha del sistema se puede expresar como la suma de dos señales, la debida a las acciones de control aplicadas hasta el instante actual, respuesta libre, y la generada a partir de las acciones futuras calculadas por el controlador, respuesta forzada.

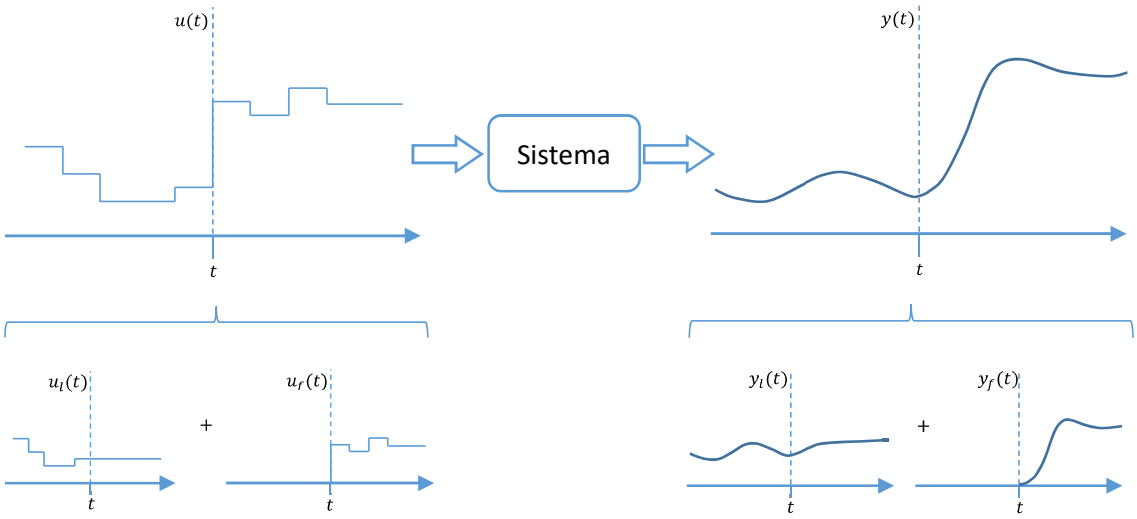


Figura 2.4 – Respuesta libre y forzada

La respuesta predicha se genera con la suma de las acciones de control pasadas, $u_l(t)$, y futuras, $u_f(t)$, como se muestra en la Figura 2.4. La acción de control $u(t)$ se corresponde con:

$$u(t) = u_l(t) + u_f(t) \qquad [2.16]$$

Donde $u_l(t)$ y $u_f(t)$ adquieren los siguientes valores en el tiempo:

$$u_l(t)\begin{cases} u_l(t-j) = u_l(t-j)\ para\ j = 1,2,\dots \\ u_l(t+j) = u_l(t-1)\ para\ j = 0,1,2,\dots \end{cases}$$
$$u_f(t)\begin{cases} u_f(t-j) = 0\ para\ j = 1,2,\dots \\ u(t+j) = u(t+j) - u(t-1)\ para\ j = 0,1,2,\dots \end{cases} \qquad [2.17]$$

3.3. Función de coste

La función objetivo que se pretende minimizar normalmente responde al criterio minimizar el error de la salida respecto de una determinada referencia, al mismo tiempo que se puede penalizar el esfuerzo de control requerido para ello y respetar el cumplimiento de unas ciertas restricciones, como se indicó al inicio del apartado [2].

La ecuación [2.18] muestra una expresión genérica de la función de coste en la que aparecen el error y el incremento en el valor de la acción de control ponderados con sendos coeficientes $\gamma(j)$ y $\beta(j)$, siempre considerados dentro de sus respectivos horizontes de coincidencia y de control.

En otras variantes, se añade a la expresión directamente el valor de la señal de control o se sustituye el incremento por éste último.

$$min\, J = \sum_{j=N_1}^{N_2} \gamma(j)[\hat{y}(t+j) - w(t+j)]^2 + \sum_{j=0}^{N_{u-1}} [\beta(j)\Delta u(t+j)]^2 \qquad [2.18]$$

3.3.1. Parámetros de configuración

Los parámetros N_1 y N_2 definen el comienzo y el final del horizonte de coincidencia y N_u es el horizonte de control, que no tiene por qué coincidir con el horizonte máximo y, de hecho, suele ser significativamente menor. El valor de N_1 representa a partir de cuándo se quiere que la salida siga la referencia, de manera que se eliminan de la función objetivo los errores presentes en los primeros instantes, lo cual es de gran interés en sistemas con retardo o de fase no mínima puesto que suavizará la respuesta del proceso. El parámetro N_2 debe tener un valor suficientemente alto como para asegurar que el sistema alcanzará la referencia en el intervalo.

Los coeficientes $\gamma(j)$ y $\beta(j)$ son, usualmente, valores constantes, rampas o secuencias exponenciales, en las que se la penalización varía con el paso del tiempo. Si la función de penalización es creciente, se consigue un control más suave y con menor esfuerzo. Si es, en cambio, decreciente los primeros errores tienen un mayor peso y el control será más brusco.

3.3.2. Trayectoria de referencia

La función de referencia puede ser, en el caso más sencillo, una constante aunque si se conoce de antemano la forma de la respuesta del sistema se pueden conseguir mejores resultados incluyendo esta dinámica en la función de coste.

Lo más habitual es aplicar un filtro de primer orden al valor de consigna desde el valor actual de la salida, para así lograr una aproximación más suave.

$$\begin{aligned} w(t) &= y(t) \\ w(t+k) &= \frac{1-\alpha}{1-\alpha z^{-1}} r(t+k) \end{aligned} \qquad [2.19]$$

El parámetro α que se muestra en la ecuación [2.19] puede tomar un valor entre 0 y 1 de tal modo que, la respuesta será más suave si es cercano a 1 ($w_2(t)$ Figura 2.5) y más brusca si es más cercano a 0 ($w_1(t)$ Figura 2.5).

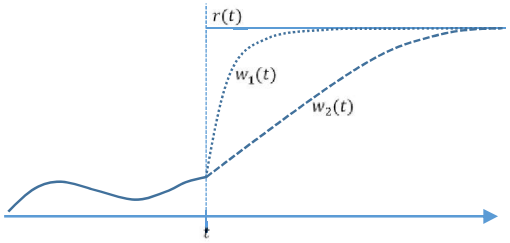


Figura 2.5 – Trayectoria de referencia

3.3.3. Restricciones

Una de las fortalezas del MPC reside en la posibilidad de introducir en el problema de optimización distintos tipos de restricciones. En la práctica, todos los procesos están sujetos a restricciones. Los actuadores tienen un rango de acción limitado y una velocidad de cambio determinada, ya sea por razones constructivas, de seguridad o ambientales, o incluso los propios alcances del sensor pueden causar límites en las variables del proceso tales como niveles en tanques, caudales en tuberías o temperaturas y presiones máximas.

Todo esto hace necesaria la introducción de restricciones en la función a minimizar. Normalmente, se considerarán límites en la amplitud y la velocidad de cambio permitida en la acción de control, así como en el valor de la salida [2.20]. Esto otorga al control predictivo una gran ventaja sobre el control convencional.

$$\begin{aligned} \hat{y}_{max} &\geq \hat{y}(t+j) \geq \hat{y}_{min} \\ u_{max} &\geq u(t+j) \geq u_{min} \\ \Delta u_{max} &\geq \Delta u(t+j) \geq \Delta u_{min} \end{aligned} \qquad [2.20]$$

3.4. Obtención de la ley de control

La obtención de los valores futuros de la acción de control responde al criterio de la minimización de la función de costes anteriormente descrita. En el caso de una función J cuadrática y si el modelo es lineal se puede obtener una solución analítica, pero en el resto de los casos será necesario utilizar un método iterativo.

El concepto de horizonte de control aparece con la idea de reducir el número de grados de libertad del problema, puesto que de otro modo el número de variables independientes sería de $N_2 - N_1 + 1$. Se considera, por tanto, que pasado un número $N_u < N_2$ de intervalos no hay variación en las señales de control o, dicho de otro modo:

$$\Delta u(t + j - 1) = 0 \qquad j > N_u \qquad [2.21]$$

4. Revisión de algunos algoritmos MPC

4.1. *Predictive Functional Control*

El PFC es un controlador desarrollado por Richalet (Richalet, 1992) cuyo objetivo reside en facilitar la implementación del algoritmo del controlador en un dispositivo programable industrial (PLC o DCS) mientras se mantienen las propiedades no lineales del sistema en su modelo interno.

En vez de buscar que la salida del sistema coincida con todos los puntos de la salida estimada en el horizonte de predicción, el PFC hace uso del concepto de puntos de coincidencia, es decir, algunos puntos en el futuro son seleccionados y sy no sobre todo el horizonte. De esta manera, en la práctica, se puede obtener un rendimiento del sistema similar al que se conseguiría con otros MPC con un algoritmo mucho más simple y válido para procesos rápidos. (Rossiter, 2003)

Aplicando esta estrategia a un sistema de primer orden, la solución al problema de control se puede calcular de forma analítica de forma sencilla utilizando como modelo una función de transferencia continua de primer orden. El método se puede extrapolar fácilmente a sistemas de primer orden con retardo e incluir en la formulación

4.1.1. PFC aplicado a un sistema de primer orden

El *Predictive Functional Control* aplicado a un sistema de primer orden de una entrada y una salida tiene como objetivo que la salida del sistema pasados N periodos de muestro sea la definida por un filtro de primer orden con un polo en λ aplicado sobre la referencia $w(t)$ en un solo cambio de la acción de control.

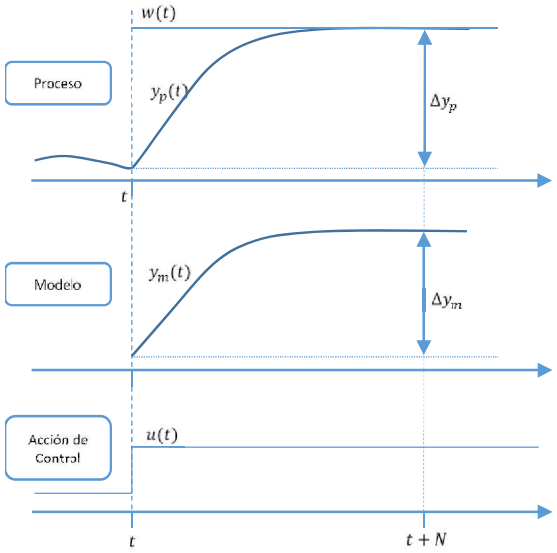


Figura 2.6 – Estrategia del PFC aplicada a un sistema de primer orden

Esto, expresado en los términos del PFC equivale a buscar un punto de coincidencia entre el modelo y la trayectoria de referencia en el instante N.

$$\frac{y_r(k)}{w(k)} = \frac{1-\lambda}{1-\lambda z^{-1}} \qquad [2.22]$$

Donde, asumiendo que la referencia se mantiene constante durante el horizonte de predicción $w(t) = w(t+1) = \cdots = w(t+N)$ y tomando $y_r(t) = y_p(t)$, se obtiene:

$$\Delta y_r(t + NT_s) = (1-\lambda^N)(w(t) - y_p(t)) \qquad [2.23]$$

Siendo la ecuación diferencial de primer orden que modela el proceso:

$$\tau \frac{dy(t)}{dt} + y(t) = ku(t) \qquad [2.24]$$

Su solución explícita es:

$$y_m(t + NT_s) = y_p(t) \cdot e^{\frac{-NT_s}{\tau}} + ku(t)\left(1 - e^{\frac{-NT_s}{\tau}}\right) \qquad [2.25]$$

Y por tanto:

$$\Delta y_m(t + NT_s) = \left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) y_p(t) - ku(t)\left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) \qquad [2.26]$$

Aplicando la condición de que el modelo y la referencia deben coincidir en el instante N:

$$\Delta y_r(t + NT_s) = \Delta y_m(t + NT_s) \qquad [2.27]$$

Se obtiene:

$$(1 - \lambda^N)\left(w(t) - y_p(t)\right) = \left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) \cdot y_p(t) - k \cdot u(t) \cdot \left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) \qquad [2.28]$$

De donde se deduce que el valor de la acción de control a aplicar en el siguiente periodo de muestreo:

$$u(t) = \frac{\left(e^{\frac{-NT_s}{\tau}} - 1\right) y_p(t) - (1 - \lambda^N)\left(w(t) - y_p(t)\right)}{k\left(e^{\frac{-NT_s}{\tau}} - 1\right)} \qquad [2.29]$$

Esto implica en última instancia que se requiere sólo de un cálculo para generar el valor del siguiente cambio en la variable manipulada, reduciendo de forma significativa la carga computacional requerida si se compara con los métodos que utilizan métodos iterativos.

4.1.2. Incorporación de perturbaciones al modelo

Con el fin de aumentar la robustez del algoritmo, se puede incluir en el modelo de predicción una perturbación que, además de integrar las posibles perturbaciones no medibles, haga posible tener en cuenta los errores de modelado del sistema. La salida predicha se calcularía en este caso del siguiente modo:

$$\tau \frac{dy(t)}{dt} + y(t) = ku(t) + v \qquad [2.30]$$

Donde:

$$\Delta y_m(t + NT_s) = \left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) y_p(t) - ku(t)\left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) - v\left(e^{\frac{-N \cdot T_s}{\tau}} - 1\right) \qquad [2.31]$$

Como se puede ver en la ecuación [2.30], se considera que la perturbación es constante durante todo el horizonte de predicción y su valor se calcula como la diferencia entre la salida medida del proceso y la estimación realizada a partir del modelo en el instante $t - T_s$.

$$\hat{v} = \frac{y_p(t) - y_p(t - T_s)e^{\frac{-T_s}{\tau}} - ku(t - T_s)\left(1 - e^{\frac{-T_s}{\tau}}\right)}{\left(1 - e^{\frac{-T_s}{\tau}}\right)} \quad [2.32]$$

Del mismo modo que en el caso anterior, igualando la salida del proceso y la del modelo en el punto de coincidencia se obtiene la acción de control a aplicar:

$$u(t) = \frac{\left(e^{\frac{-NT_s}{\tau}} - 1\right)(y_p(t) - \hat{v}) - (1 - \lambda^N)\left(w(t) - y_p(t)\right)}{k\left(e^{\frac{-NT_s}{\tau}} - 1\right)} \quad [2.33]$$

4.1.1. Incorporación de retardo al modelo

Considerando un modelo de primer orden con retardo de la forma:

$$\tau\frac{dy(t)}{dt} + y(t) = ku(t - d) + v \quad [2.34]$$

La función de referencia pasa a ser la siguiente:

$$d = D \cdot Ts$$
$$\Delta y_p(t + NT_s) = (1 - \lambda^{N-D})(w(t) - y_p(t)) \quad [2.35]$$

La solución explícita de la ecuación [2.36] para $N > D$ será:

$$y_m(t + NT_s) = y_p(t) \cdot e^{\frac{-NT_s}{\tau}} - ku(t)\left(e^{\frac{-(N-D)\cdot T_s}{\tau}} - 1\right) - v\left(e^{\frac{-N\cdot T_s}{\tau}} - 1\right)$$
$$+ k \cdot e^{\frac{-NT_s}{\tau}}\left(e^{\frac{T_s}{\tau}} - 1\right)\sum_{m=t}^{t-T_s(D-1)}\left[u(m - T_sD)e^{\frac{(m-t)T_s}{\tau}}\right] \quad [2.36]$$

Si, como en los casos anteriores, se iguala $\Delta y_p = \Delta y_m$ se obtiene el valor de la acción de control:

$$u(t) = \frac{\left(1 - e^{\frac{-NT_s}{\tau}}\right)(y_p(t) - \hat{v}) + (1 - \lambda^{N-D})\left(w(t) - y_p(t)\right)}{k\left(1 - e^{\frac{-(N-D)T_s}{\tau}}\right)}$$
$$+ \frac{k \cdot e^{\frac{-NT_s}{\tau}}\left(e^{\frac{T_s}{\tau}} - 1\right)\sum_{m=t}^{t-T_s(D-1)}\left[u(m - T_sD)e^{\frac{(m-t)T_s}{\tau}}\right]}{k\left(1 - e^{\frac{-(N-D)T_s}{\tau}}\right)} \quad [2.37]$$

Donde la estimación de la perturbación se calcula como:

$$\hat{v} = \frac{y_p(t) - y_p(t - T_s)e^{\frac{-T_s}{\tau}} - ku(t - (D+1)T_s)\left(1 - e^{\frac{-T_s}{\tau}}\right)}{\left(1 - e^{\frac{-T_s}{\tau}}\right)} \qquad [2.38]$$

Se puede ver que aunque la complejidad de la función que genera la acción de control va aumentando, continúa siendo un solo cálculo por periodo de muestreo.

4.1.2. Conclusiones

El PFC no es un algoritmo destinado al control multivariable, que es por otra parte una de las grandes fortalezas del MPC, aunque cuenta con un gran atractivo para la industria debido a su simplicidad (Rossiter, 2003).

4.2. Dynamic Matrix Control

El *Dynamic Matrix Control* o Control de Matriz Dinámica es un método desarrollado por Cutler y Ramaker en 1980. El DMC utiliza como modelo del proceso la respuesta escalón y es aplicable a procesos multivariables siempre que sean asintóticamente estables y sin integradores (Prada, Serrano, & Vega, A comparative study of GPC and DMC controllers, 1994).

Como se ha dicho anteriormente, este MPC es aplicable tanto para sistemas SISO como MIMO, de manera que en este apartado se tratará en primer lugar el método aplicado a un sistema de una entrada y una salida para, a continuación, extenderlo al caso multivariable.

4.2.1. DMC aplicado a un sistema SISO (*Single Input Single Output*)

El primer paso es definir la formulación del algoritmo del DMC para un sistema de una entrada $u(t)$ y una salida $y(t)$ incluyendo el efecto de una perturbación medible $v(t)$ y de otras no medibles $n(t)$.

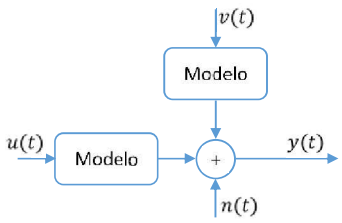


Figura 2.7 – Acciones y perturbaciones sobre el sistema que se consideran en el DMC

Donde la salida predicha del sistema se puede expresar en función de los modelos en respuesta salto de las entradas, definido por los coeficientes g_i, y de las perturbaciones medibles, determinado por los coeficientes d_i .

$$\hat{y}(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-i) + \sum_{i=1}^{\infty} d_i \Delta v(t-i) + n(t) \qquad [2.39]$$

En la ecuación de la predicción de la salida del sistema [2.40] se pueden diferenciar los valores pasados $[i = 1, t]$ y futuros $[i = k+1, N]$ que conforman la salida predicha.

$$\hat{y}(t+j) = \sum_{i=1}^{j} g_i \Delta u(t+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(t+j-i) + \sum_{i=1}^{j} d_i \Delta v(t+j-i) + \sum_{i=j+1}^{\infty} d_i \Delta v(t+j-i) + n(t+j) \qquad [2.40]$$

Donde $n(t)$ es el valor de la perturbación no medible, de valor igual a la diferencia entre el valor medido y el calculado mediante el modelo en el instante t. Para el cálculo de la salida predicha, se hace la hipótesis de que este valor se mantiene constante durante el horizonte de predicción [2.41].

$$n(t+j) = n(t) = y_p(t) - \sum_{i=1}^{\infty} g_i \Delta u(t-i) - \sum_{i=1}^{\infty} d_i \Delta v(t-i) \qquad [2.41]$$

Sustituyendo en la expresión [2.40] $n(t+j)$ por su definición [2.41] se obtiene:

$$\hat{y}(t+j) = \sum_{i=1}^{j} g_i \Delta u(t+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(t+j-i) + \sum_{i=1}^{j} d_i \Delta v(t+j-i) + \sum_{i=j+1}^{\infty} d_i \Delta v(t+j-i) + y_p - \sum_{i=1}^{\infty} g_i \Delta u(t-i) - \sum_{i=1}^{\infty} d_i \Delta v(t-i) \qquad [2.42]$$

Agrupando en la ecuación anterior los términos que sólo dependen de las acciones de control pasadas del sistema y aceptando la hipótesis de considerar el valor de la perturbación medible futura constante y de valor igual al último registrado $\Delta v(t+j) = \Delta v(t)$, se obtiene la expresión de la respuesta libre del sistema:

$$l(t+j) = y_p(t) + \sum_{i=j+1}^{\infty} g_i \Delta u(t+j-i) - \sum_{i=1}^{\infty} g_i \Delta u(t-i) + \sum_{i=1}^{j} d_i \Delta v(t+j-i) + \sum_{i=j+1}^{\infty} d_i \Delta v(t+j-i) - \sum_{i=1}^{\infty} d_i \Delta v(t-i) \quad [2.43]$$

Como se habló en el apartado 3.1.2 al tratar la respuesta salto, si el sistema es estable llega un momento en que $g_i - g_{i-1} = 0$, de manera, agrupando términos, resulta:

$$l(t+j) = y_p(t) + \sum_{i=1}^{N} (g_{j+i} - g_i) \Delta u(t-i) + \sum_{i=1}^{N} (d_{j+i} - d_i) \Delta v(t-i) + \sum_{i=1}^{j} d_i \Delta v(t+j-i) \quad [2.44]$$

La función de coste utilizada en el DMC es la estándar de los MPC que se mencionó en apartados anteriores. A continuación se explicará cómo se definirán en este método cada uno de los términos de la función objetivo a optimizar.

$$J = \sum_{j=N_1}^{N_2} [\hat{y}(t+j) - w(t+j)]^2 + \sum_{j=0}^{N_{u-1}} [\beta \Delta u(t+j)]^2$$

Sujeto a: [2.45]

$$\hat{y}_{max} \geq \hat{y}(t+j) \geq \hat{y}_{min}$$
$$u_{max} \geq u(t+j) \geq u_{min}$$
$$\Delta u_{max} \geq \Delta u(t+j) \geq \Delta u_{min}$$

4.2.1.1. Acción de control

El vector $\Delta \boldsymbol{u}(t)$ de tamaño $N_u - 1$ está formado por las acciones de control futuras que se obtendrán con la resolución del problema de optimización. El peso que se otorga a este término viene ponderado por el factor β.

4.2.1.2. Trayectoria de referencia

El vector $\boldsymbol{w}(t)$ de tamaño $N_2 - N_1 + 1$ contiene la trayectoria de referencia deseada para el sistema. Puede ser un valor constante o se puede aplicar un filtro de primer orden si se desea suavizar la respuesta.

4.2.1.3. Predicción de la salida

La salida predicha se define como la suma de la respuesta libre y la respuesta forzada. Expresando el cálculo de la respuesta forzada en forma matricial, la predicción de la salida resulta:

$$\hat{\boldsymbol{y}}(t+j) = \boldsymbol{G}\Delta\boldsymbol{u}(t+j) + \boldsymbol{l}(t+j) \qquad [2.46]$$

Donde G es una matriz que contiene los coeficientes de la respuesta salto del sistema:

$$\boldsymbol{G} = \begin{bmatrix} g_{N_1} & \cdots & g_2 & g_1 & 0 & \cdots & 0 \\ g_{N_1+1} & g_{N_1} & \cdots & g_2 & g_1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{N_2} & \cdots & \cdots & \cdots & \cdots & \cdots & g_{N_2-N_u+1} \end{bmatrix}$$

Y $\boldsymbol{l}(t)$ el vector de tamaño $N_2 - N_1 + 1$ que contiene las respuestas libres obtenidas hasta el momento actual.

Teniendo en cuenta estas definiciones, la función objetivo queda reformulada, como se muestra en la ecuación [2.48], con la forma de un problema de optimización cuadrática [2.47].

$$\min x = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \qquad [2.47]$$

$$min\, J = \Delta\boldsymbol{u}^T(t)[\boldsymbol{G}^T\boldsymbol{G} + \beta\boldsymbol{I}]\Delta\boldsymbol{u}(t) - 2\boldsymbol{e}_0^T\boldsymbol{G}\Delta\boldsymbol{u}(t) + \boldsymbol{e}_0^T\boldsymbol{e}_0 \qquad [2.48]$$

Donde $\boldsymbol{e}_0$ es un vector de tamaño $N_2 - N_1 + 1$ que contiene la diferencia entre la referencia y la respuesta libre.

$$\boldsymbol{e}_0^T = [\ w(t+N_1) - l(t+N_1) \quad \ldots \quad w(t+N_2) - l(t+N_2)]$$

En el caso de que no se impongan restricciones en las variables del proceso, los movimientos de control se pueden calcular derivando la función de coste y el problema tiene solución explícita:

$$\frac{\partial J}{\partial \Delta \boldsymbol{u}} = 0 \rightarrow 2\boldsymbol{G}^T\left[\boldsymbol{e}_0 \ - \boldsymbol{G}\Delta\boldsymbol{u}\right] + 2\beta\Delta\boldsymbol{u}$$
$$\Delta\boldsymbol{u}(t) = [\boldsymbol{G}^T\boldsymbol{G} + \beta\boldsymbol{I}]^{-1}\boldsymbol{G}^T(\boldsymbol{w} - \boldsymbol{l}) \qquad [2.49]$$

Si, en caso contrario, se quieren aplicar restricciones sobre el sistema [2.20] será necesario recurrir al cálculo numérico para obtener la solución del problema. Aunque computacionalmente la resolución es más costosa, permite obtener de manera exacta la solución buscada aunque es importante tener en cuenta los posibles problemas de factibilidad.

4.2.2. Sistema MIMO (*Multiple Inputs Multiple Ouputs*)

La solución del problema aplicado a un sistema multivariable se puede considerar una extensión del caso anterior. Si se considera, por ejemplo, un sistema de dos entradas, tres salidas y una perturbación medible, se obtendría el siguiente modelo del sistema:

$$\begin{aligned} y_1(t) &= \sum_{i=1}^{\infty} g_{11_i}\Delta u_1(t-i) + \sum_{i=1}^{\infty} g_{12_i}\Delta u_1(t-i) + \sum_{i=1}^{\infty} d_i\Delta v(t-i) + n_1(t) \\ y_2(t) &= \sum_{i=1}^{\infty} g_{21_i}\Delta u_2(t-i) + \sum_{i=1}^{\infty} g_{22_i}\Delta u_2(t-i) + \sum_{i=1}^{\infty} d_i\Delta v(t-i) + n_2(t) \\ y_3(t) &= \sum_{i=1}^{\infty} g_{31_i}\Delta u_3(t-i) + \sum_{i=1}^{\infty} g_{32_i}\Delta u_3(t-i) + \sum_{i=1}^{\infty} d_i\Delta v(t-i) + n_3(t) \end{aligned} \qquad [2.50]$$

Donde las predicciones de las salidas serían:

$$\begin{aligned} \hat{y}_1(t+j) &= \sum_{i=1}^{j} g_{11_i}\Delta u(t+j-i) + \sum_{i=1}^{j} d_{12_i}\Delta v(t+j-i) + l_1(t) \\ \hat{y}_2(t+j) &= \sum_{i=1}^{j} g_{21_i}\Delta u(t+j-i) + \sum_{i=1}^{j} d_{22_i}\Delta v(t+j-i) + l_2(t) \\ \hat{y}_3(t+j) &= \sum_{i=1}^{j} g_{31_i}\Delta u(t+j-i) + \sum_{i=1}^{j} d_{32_i}\Delta v(t+j-i) + l_3(t) \end{aligned} \qquad [2.51]$$

La función de coste en este caso incluye un nuevo parámetro de configuración γ cuya función es ponderar el peso de las salidas o, dicho de otro modo, para penalizar más fuertemente el error de unas salidas respecto a las demás.

$$J = \sum_{j=N_1}^{N_2} \gamma_1[\hat{y}_1(t+j) - w_1(t+j)]^2 + \sum_{j=N_1}^{N_2} \gamma_2[\hat{y}_2(t+j) - w_2(t+j)]^2 + \sum_{j=N_1}^{N_2} \gamma_3[\hat{y}_3(t+j) - w_3(t+j)]^2 + \sum_{j=0}^{N_{u-1}} [\beta_1 \Delta u_1(t+j)]^2 + \sum_{j=0}^{N_{u-1}} [\beta_2 \Delta u_2(t+j)]^2 \quad [2.52]$$

Extrapolando esta formulación para un caso genérico de N salidas y M entradas, se obtiene la función de costes que se muestra en la ecuación [2.53]

$$J = \sum_{k=1}^{N} \sum_{j=N_{1k}}^{N_{2k}} \gamma_k[\hat{y}_k(t+j) - w_k(t+j)]^2 + \sum_{k=1}^{M} \sum_{j=0}^{N_{uk}-1} \beta_k[\Delta u_k(t+j)]^2 \quad [2.53]$$

Que expresada en forma matricial resulta:

$$min\, J = \Delta \boldsymbol{u}^T(t) \overbrace{[\boldsymbol{G}^T \boldsymbol{\gamma} \boldsymbol{G} + \boldsymbol{\beta}]}^{\boldsymbol{H}} \Delta \boldsymbol{u}(t) - 2 \overbrace{(\boldsymbol{G}^T \boldsymbol{\gamma} \boldsymbol{e}_0)^T}^{\boldsymbol{c}^T} \Delta \boldsymbol{u}(t) + \overbrace{\boldsymbol{e}_0^T \boldsymbol{e}_0}^{cte} \quad [2.54]$$

4.2.3. Optimizador

El controlador DMC requiere de un algoritmo QP (*Quadratic Programming*) para minimizar la función de costes descrita y así obtener las acciones de control que se deben aplicar. El método de optimización seleccionado debe ser determinista y no heurístico, puesto que en el caso de que en un ciclo del controlador no se logre la convergencia a la solución óptima, el algoritmo debe asegurar que al menos la solución sea cercana a él y esto no es posible con métodos heurísticos.

El algoritmo seleccionado es el QP de Hildreth, puesto que es un método que cumple las condiciones descritas en el párrafo anterior y su implementación factible en un PLC (Wang, 2009).

4.2.3.1. Programación cuadrática

Si se expresa la función objetivo J y las restricciones de un problema de programación cuadrática como:

$$J = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + x^T\boldsymbol{c} \qquad [2.55]$$

$$\boldsymbol{A}x \leq \boldsymbol{b}$$

Donde $\boldsymbol{H}$, $\boldsymbol{c}$, $\boldsymbol{A}$ y $\boldsymbol{b}$ son matrices y vectores compatibles en el problema de programación cuadrática, se puede asumir sin pérdida de generalidad que $\boldsymbol{H}$ es simétrica definida positiva.

4.2.3.2. QP con restricciones lineales de igualdad

Para problemas de optimización con restricciones de igualdad, el método de los multiplicadores de Lagrange proporciona condiciones necesarias que deben cumplirse en el óptimo. Se trata convertir el problema con restricciones en otro sin ellas ampliado en m variables λ_j (los multiplicadores de Lagrange), tal que su solución coincida en las variables x con el primitivo y cumpla las restricciones.

$$J = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{c} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \qquad [2.56]$$

Para todos los x que cumplan las restricciones se verifica que:

$$\min_{\boldsymbol{x},\boldsymbol{\lambda}} J(\boldsymbol{x},\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} J(\boldsymbol{x}) \qquad [2.57]$$

4.2.3.3. QP con restricciones lineales de desigualdad

En la minimización con restricciones de desigualdad, el número de restricciones podría ser mayor que el número de variables de decisión. Las restricciones de desigualdad $\boldsymbol{A}x \leq \boldsymbol{b}$ pueden comprender restricciones activas e inactivas. Se dice que una desigualdad $\boldsymbol{A}_i x \leq b_i$ está activa si $\boldsymbol{A}_i x = b_i$ y inactiva si $\boldsymbol{A_i}x < b_i$, donde $\boldsymbol{A_i}$ junto con b_i forman la i-ésima restricción de desigualdad y son la i-ésima fila de la matriz $\boldsymbol{A}$ y el i-ésimo elemento del vector $\boldsymbol{b}$, respectivamente.

4.2.3.4. Conjuntos activos

La idea de los métodos de conjuntos activos es definir en cada paso de un algoritmo un conjunto de restricciones que debe ser tratado como el conjunto activo. El algoritmo entonces procede a moverse sobre la superficie definida por el conjunto activo de restricciones a un punto mejorado.

En cada paso del método de conjuntos activos, se resuelve un problema de restricción de igualdad. Si todos los multiplicadores de Lagrange $\lambda_i \geq 0$, entonces el punto es una solución local al problema original. Si, por otra parte, existe un $\lambda_i < 0$, entonces el valor objetivo de la función puede disminuirse relajando la restricción i (es decir, suprimiéndola de la ecuación de restricciones).

Durante el curso de la minimización, es necesario supervisar los valores de las otras restricciones para asegurarse de que no son violadas, ya que todos los puntos definidos por el algoritmo deben ser factibles. A menudo sucede que mientras se mueve sobre la superficie activa, se encuentra un nuevo límite de una restricción. Es necesario añadir esta restricción al conjunto activo para luego proceder a trabajar sobre el nuevo conjunto activo.

4.2.3.5. Métodos primal-dual

La familia de métodos activos pertenece al grupo de métodos primarios, donde las soluciones se basan en las variables de decisión (también llamadas variables primitivas en la literatura). En los métodos de conjuntos activos, las restricciones activas deben identificarse junto con las variables de decisión óptimas. Si hay muchas restricciones, la carga computacional es bastante grande Un método dual puede ser utilizado sistemáticamente para identificar las restricciones que no están activas, para luego poder eliminarlas del problema. Los multiplicadores de Lagrange se denominan variables duales en la literatura de optimización. Este método dará lugar a procedimientos de programación muy sencillos para encontrar soluciones óptimas de problemas de minimización restringidos.

El problema dual al problema primal original se deriva como sigue. Suponiendo viabilidad (es decir, hay una x tal que $\boldsymbol{A}x < \boldsymbol{b}$), el problema primario es equivalente a:

$$\max_{\lambda \geq 0} \min_{x} \left[\frac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{c} + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \right] \qquad [2.58]$$

Donde la solución de la minimización de la función sobre x sin restricciones viene dada por:

$$\boldsymbol{x} = -\boldsymbol{H}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) \quad [2.59]$$

4.2.3.6. Método QP de Hildreth

Un algoritmo sencillo, llamado procedimiento de programación cuadrática de Hildreth (Luenberger, 1969, Wismer y Chattergy, 1978), fue propuesto para resolver este problema dual. En este algoritmo, los vectores de dirección se seleccionaron para ser iguales a los vectores base, $e_i = [0\ 0\ \ldots 1\ \ldots\ 0\ 0]^T$, de manera que el vector λ varía sólo un componente cada vez. En un paso, habiendo obtenido un $\lambda \geq 0$, el algoritmo se centra en un solo componente λ_i. La función objetivo puede considerarse como una función cuadrática en este único componente. Se ajusta λ_i de manera que minimice la función objetivo. Si se requiere $\lambda_i < 0$, se establece $\lambda_i = 0$, por lo que, en cualquier caso, la función objetivo disminuye. Luego, se considera el siguiente componente λ_{i+1}. Considerando que un ciclo completo a través de los componentes es una iteración en la que el vector pasa desde λ^m a λ^{m+1}, el método puede expresarse explícitamente como:

$$\lambda^{m+1} = \max(0, w_i^{m+1}) \quad [2.60]$$

Con:

$$w_i^{m+1} = -\frac{1}{p_{ij}}[d_i + \sum_{j=1}^{i-1} p_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} p_{ij}\lambda_j^m] \quad [2.61]$$

Donde el escalar p_{ij} es el ij-ésimo elemento de la matriz $\boldsymbol{P}$ y d_{ij} es el i-ésimo elemento del vector $\boldsymbol{d}$.

$$\boldsymbol{P} = \boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T$$
$$\boldsymbol{d} = \boldsymbol{b} + \boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{c} \quad [2.62]$$

Nótese que en la ecuación anterior se incluyen dos conjuntos de valores λ en el cálculo, por un lado los $\boldsymbol{\lambda}^m$ y por otro los $\boldsymbol{\lambda}^{m+1}$ ya actualizados.

Como el vector $\boldsymbol{\lambda}^*$ al que se ha convergido contiene tanto ceros como valores positivos para los multiplicadores de Lagrange, se obtiene:

$$\boldsymbol{x} = -\boldsymbol{H}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}^*) \quad [2.63]$$

4.2.3.7. Conclusión

Hay algunos comentarios que se pueden hacer sobre el funcionamiento del método. En primer lugar, el algoritmo de programación cuadrática de Hildreth se basa en una búsqueda elemento por elemento, por lo tanto, no requiere ninguna inversión de matriz. Como resultado, si las restricciones activas son linealmente independientes y su número es menor o igual al número de variables de decisión, entonces las variables duales convergerán. Sin embargo, si uno o ambos de estos requisitos son violados, entonces las variables duales no convergerán a un conjunto de valores fijos. La búsqueda terminará cuando el contador de iteraciones alcance su valor máximo. Como no hay inversión de matrices, el cálculo continuará sin interrupción. En el caso de que se alcance el número máximo de iteraciones debido a una situación de conflicto con las restricciones, el algoritmo dará una solución comprometida y casi óptima.

Esta es una de las principales ventajas de utilizar este enfoque en aplicaciones en tiempo real, ya que la capacidad del algoritmo para recuperarse automáticamente de un problema restringido mal condicionado es primordial para la seguridad del funcionamiento de la planta.

Capítulo 3 . IMPLEMENTACIÓN

1. Introducción

Los algoritmos MPC que se ha decidido implementar tienen estructuras y características muy distintas, por lo que el recorrido que se ha seguido para la implementación de cada uno de ellos también lo es.

El *Predictive Functional Control*, en primer lugar, ha sido diseñado y validado en el entorno de EcosimPro para después ser implementado en el PLC incluyéndolo en la librería de UNICOS como alternativa al PID dentro del objeto *Controller*.

En el caso del *Dynamic Matrix Control* el primer paso será implementar el controlador en Matlab, puesto que en este entorno facilita el trabajar con matrices. Además, trabajando en Matlab es posible contrastar el funcionamiento del optimizador propuesto con otros resolvedores. El siguiente paso es la implementación del controlador en el PLC y realizar las verificaciones necesarias sobre su funcionamiento antes de plantear en incluirlo en UNICOS.

2. *Predictive Functional Control*

La implementación del PFC se realizará, en primer lugar, sobre EcosimPro y a continuación en el PLC. La ejecución del PFC no implica un gran coste computacional ni tampoco ocupa demasiado espacio en memoria, de manera que no es necesario tener consideraciones especiales a la hora implementarlo en el PLC.

2.1. Algoritmo para sistemas de primer orden con retardo

En el Capítulo 2 se demostró cómo, partiendo de la ecuación diferencial que define un sistema de primer orden con retardo, se puede obtener el valor de la acción de control necesaria para conseguir que la función coincida con la trayectoria de referencia en el instante N.

En la implementación del controlador en EcosimPro se incluirán las fórmulas descritas en las ecuaciones [3.1], que realiza el cálculo de la acción de control, y [3.2], que describe el valor de la perturbación en cada instante y que puede llegar a ser muy ruidosa en aplicación sobre un sistema real.

$$u(t) = \frac{\left(e^{\frac{-NT_S}{\tau}} - 1\right)\left(y_p(t) - \hat{v}\right) - (1 - \lambda^{N-D})\left(w(t) - y_p(t)\right)}{k\left(e^{\frac{-(N-D)T_S}{\tau}} - 1\right)} + \frac{k \cdot e^{\frac{-NT_S}{\tau}}\left(e^{\frac{T_S}{\tau}} - 1\right)\sum_{m=t}^{t-T_S(D-1)}\left[u(m - T_S D)e^{\frac{(m-t)T_S}{\tau}}\right]}{k\left(e^{\frac{-(N-D)T_S}{\tau}} - 1\right)} \quad [3.1]$$

$$\hat{v} = \frac{y_p(t) - y_p(t - T_S)e^{\frac{-T_S}{\tau}} - ku(t - (D+1)T_S)\left(1 - e^{\frac{-T_S}{\tau}}\right)}{\left(1 - e^{\frac{-T_S}{\tau}}\right)} \quad [3.2]$$

Para atenuar este efecto, se ha introducido un filtro de primer orden sobre el dato calculado, de manera que aparece un nuevo parámetro de sintonía en el sistema.

$$v_f(t) = \frac{(1-\alpha)z^{-1}}{1 - \alpha z^{-1}} v(t) \quad [3.3]$$

El método no tiene incluido el manejo de restricciones en su formulación, de manera que si por motivos operativos o de seguridad fuera necesario introducir límites al valor o la velocidad de cambio de la acción de control, la única posibilidad es truncar el resultado obtenido.

2.2. Integración del algoritmo en UNICOS

El objeto *Controller* de la librería de UNICOS contenía, hasta el momento, el controlador PID y varios métodos de auto sintonía para este tipo de controlador. Además, también incluye una herramienta de identificación válida para sistemas de primer orden, con o sin retardo, que sean estables en lazo abierto que puede resultar de gran utilidad para facilitar el uso del PFC.

Al tratarse de una modificación de un objeto ya existente, el nuevo *Controller* heredará todas las características de funcionamiento definidas para el PID, como son los modos de operación y estados de trabajo que explicarán más en detalle en el siguiente punto.

Además de las modificaciones que se realicen a nivel de PLC para la integración del nuevo algoritmo, también será necesario diseñar una interfaz con el usuario en WinCC-OA.

2.2.1. Características del objeto *Controller*

La estructura básica de un objeto cualquiera de UNICOS en el PLC (Figura 3.1) consta de una lógica propia del objeto (UNICOS *Object*) y una serie de interfaces de comunicación con otros objetos, con el SCADA o con otras interfaces con el usuario.

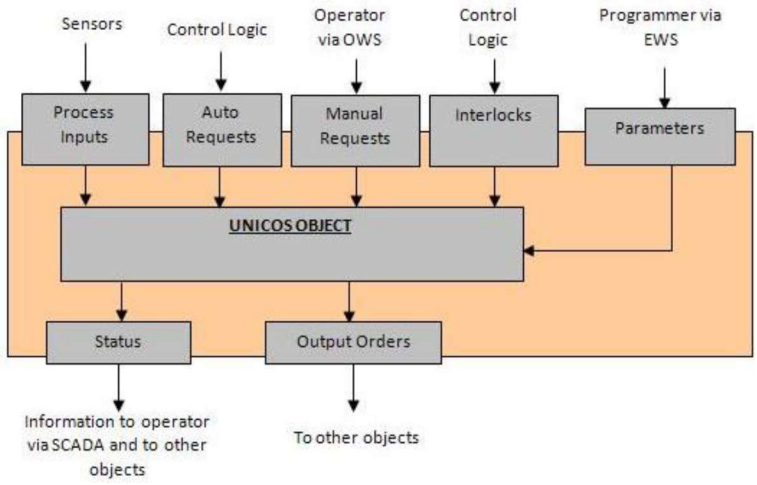


Figura 3.1 – Estructura genérica de un objeto UNICOS

Los objetos Controlador son objetos que encierran un algoritmo de regulación, en este caso un PID o un PFC. Este objeto permite:

- Controlar la salida:
  - Por medio de la regulación y aplicando el algoritmo de control correspondiente
  - Por solicitud de la lógica o el operador, independientemente del regulador
- Tener en cuenta los límites de velocidad con los que las instrucciones pueden cambiarse
- Tener en cuenta los límites que se ocupan de las salidas y las instrucciones
  - Definir los parámetros del regulador

El funcionamiento del controlador está definido por el modo de operación y el estado de trabajo activo en ese momento que se definen a continuación.

2.2.2. Modos de operación

El modo de operación es una característica aplicable a cualquier objeto UNICOS. Si nos referimos en particular al objeto *Controlador* hay algunas consideraciones particulares que se deben tener también en cuenta.

El modo de operación del objeto identifica si bien el objeto es manejado por el operador, por la lógica de control o por el proceso. Solo un modo puede estar activo al mismo tiempo. La activación de uno modo desactiva automáticamente los demás. En la **¡Error! No se encuentra el origen de la referencia.** se muestra la manera en que se interrelacionan los modos de funcionamiento y los estados de trabajo para el caso del controlador.

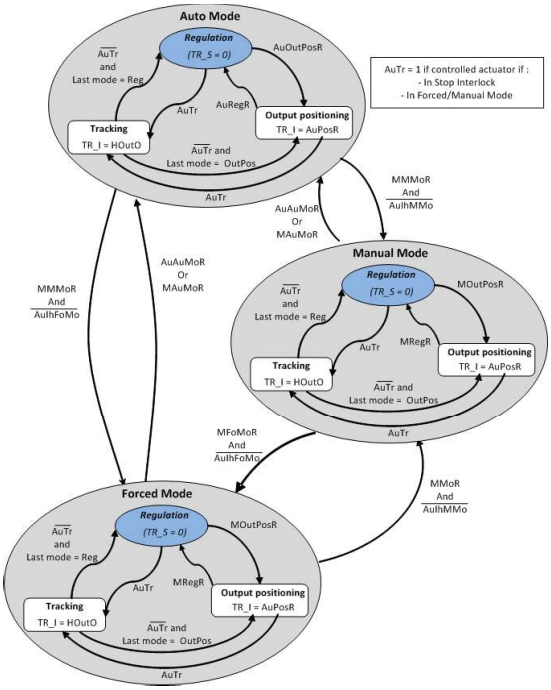


Figura 3.2 – Modos de operación y estados de trabajo en el objeto Controller

Los modos de operación que define UNICOS-CPC son los siguientes:

- ***Modo Automático (Auto Mode):*** el objeto es manejado por la lógica de control por un objeto más alto en la jerarquía.
  - Puede activarse mediante solicitud automática (*Auto Auto Mode Request*).
  - Puede activarse mediante solicitud manual (*Manual Auto Mode Request*).

- ***Modo Manual (Manual Mode):*** el operador maneja el objeto a través de la capa de supervisión.
  - Puede activarse solo mediante una solicitud del operador (*Manual Manual Mode Request*).
  - Si una solicitud automática Auto Inhibición en Modo Manual (*Auto Inhibit Manual Mode*) está activa mientras el objeto está en modo automático, el modo manual no puede ser activado con solicitud manual.
  - El retorno al modo automático es posible mediante la lógica de control (*Auto Auto Mode Request*).
  - Se aplican los *interlocks*.

- ***Modo forzado (Forced Mode)***: el operador maneja el objeto a través de la capa de supervisión.
  - Puede activarse solo mediante una solicitud del operador (*Manual Forced Mode Request*) y no es posible mediante la lógica de control.
  - Si una solicitud especial automática Auto Inhibición de Modo Forzado (Auto Inhibit Forced Mode) está activa mientras el objeto está en modo automático, el modo forzado no puede ser activado mediante solicitud manual
  - Se aplican los interlocks.

2.2.3. Estados de trabajo

El *Controller* tiene tres estados de trabajo habilitados desde cualquier modo de operación:

- ***Regulación (Regulation):*** El funcionamiento se corresponde al de un lazo cerrado de control, en el que la señal de control será emitida por el algoritmo de control que se encuentre configurado.

  Se fija un valor de consigna para la variable controlada y el regulador determina la acción de control necesaria para que la salida del sistema coincida con la referencia.

- ***Posicionamiento de salida (Output Positioning):*** El funcionamiento se corresponde con un bucle de control en lazo abierto, en el que el regulador aplica al sistema un determinado valor de señal de control fijado.

  El valor de la señal de control se asigna en la variable correspondiente con el modo de operación que se encuentre activo en ese instante, automático o manual.

- ***Seguimiento (Tracking***): Igualmente el funcionamiento es el de un bucle en lazo abierto, aunque ahora el valor de fijado para la variable de control será el propio valor de salida en el instante de activación de este estado de trabajo. Se trata de un modo de funcionamiento muy útil para las transferencias *bumpless* automático/manual y viceversa.

El funcionamiento del Controlador dependerá del modo de operación y del estado de trabajo activo:

- Cuando se activa una regulación, el *set-point* varía desde el valor actual del proceso hasta la referencia deseada con una rampa para evitar discontinuidades en el *set-point*.
- Si se para la regulación (modo manual), la salida del controlador se sitúa en una posición predefinida o permanece en su último valor y la referencia es igual al valor del proceso.
- Cuando el actuador del controlador está en estado manual o forzado, la salida está rastreando la posición del actuador y el *set-point* es igual al valor del proceso. Esto permite evitar saltos en el orden del actuador controlado.

2.2.4. UNICOS Aplication Builder

El primer paso en la integración de un lazo de control en UNICOS pasa por la herramienta de autogeneración de código UAB. El objetivo es que, al diseñar un lazo de regulación se pueda seleccionar si el controlador deseado es un PID o el PFC.

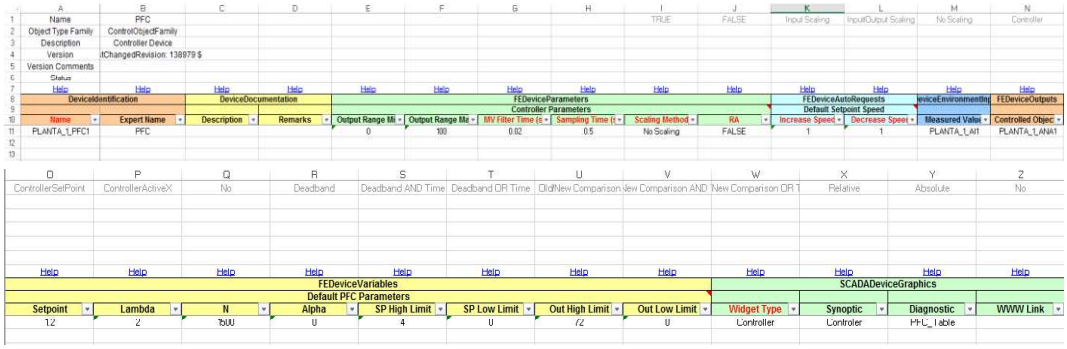


Figura 3.3 – Hoja de especificaciones diseñada para el PFC

Para realizar dicha integración es necesario incluir el nuevo objeto en los *plugins* de los que se habló en el Capítulo 1, lo cual está fuera del alcance del presente proyecto, de modo que formará parte del trabajo futro que queda pendiente de realizar.

2.2.5. WinCC-OA. Interfaz con el usuario

Uno de los requisitos que requiere la integración de un objeto en UNICOS es el diseño de la interfaz del objeto con el usuario de modo que el funcionamiento del controlador pueda ser seguido y sintonizado desde el SCADA.

El panel diseñado para la parametrización del PFC se ha embebido dentro de la ventana del bloque *Controller*, manteniendo las pestañas que muestran el estado de las variables controlada, manipulada y la referencia, las gráficas, el *autotunning* que contiene la herramienta de identificación, y la gestión de alarmas y eventos del objeto. Además, se ha añadido una pestaña más que contiene los parámetros necesarios para la configuración del PFC.

La pantalla diseñada para la gestión de los controladores predictivos de sistemas SISO [Figura 3.4] se puede descomponer, desde arriba hacia abajo, en tres zonas.

En primer lugar, se muestra un menú desplegable que cargará en función del algoritmo seleccionado, los contenidos correspondientes en la zona central de la ventana. Este diseño se ha ideado con la perspectiva de aumentar el número de MPCs integrados en UNICOS.

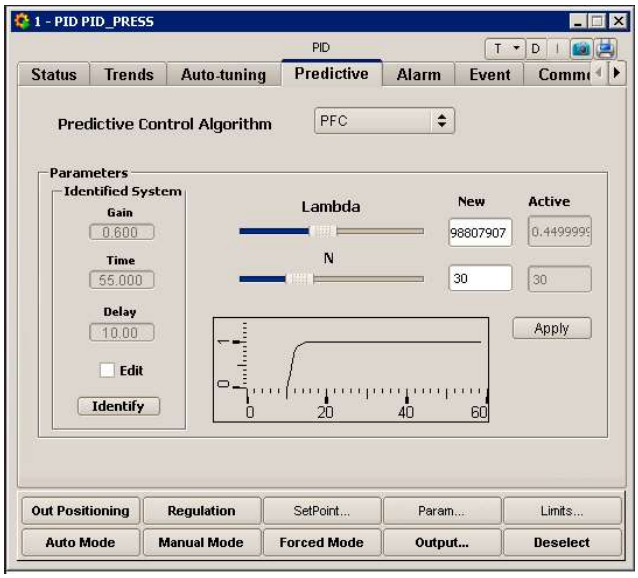


Figura 3.4 – Panel principal del PFC

La zona central del panel muestra dos zonas diferenciadas, que se corresponden con la parametrización del modelo del sistema y la configuración de los parámetros del algoritmo.

En la parte izquierda se pueden ver los parámetros del modelo de primer orden con retardo del sistema. Dicho valores pueden ser definidos por el usuario u obtenidos mediante un procedimiento de identificación al que se puede acceder mediante el botón *Identify*.

El algoritmo de identificación se basa en la obtención de un modelo en función de transferencia de primer orden con retardo mediante la aplicación de un escalón de valor determinado a la entrada (Frutos, 2015). Una vez finalizada la identificación se puede volver al panel del controlador pulsando el botón *PFC*.

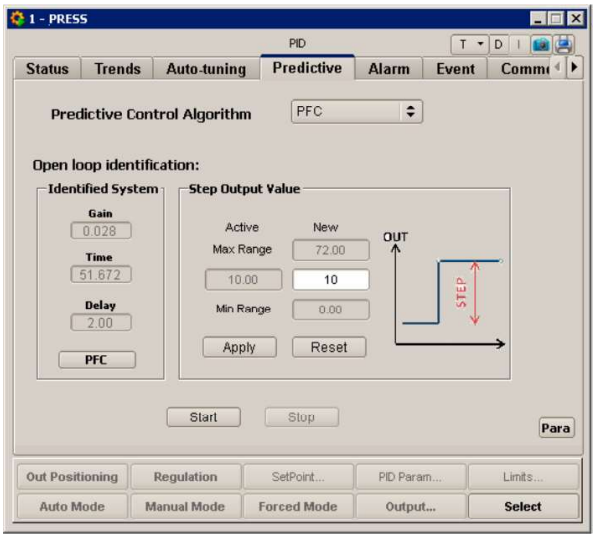


Figura 3.5 – Identificación del sistema

Por último, el botón *Param…* abre un panel con las configuraciones del modelo que se cambian con menos frecuencia del PFC, el periodo de muestreo y el tipo de acción, directa o inversa.

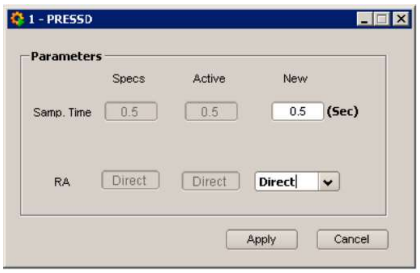


Figura 3.6 – Configuración del tiempo de muestreo y el tipo de acción

El panel del PFC muestra en la parte derecha los parámetros de configuración de éste, N y λ. La gráfica que aparece debajo muestra la forma de la respuesta del sistema que se ha definido con dichos parámetros para así facilitar al usuario la sintonía del controlador.

Por último, en la zona inferior se muestra una botonera que permite controlar los estados de trabajo *Output Positioning* y *Regulation* y los modos de operación *Auto*, *Manual* y *Forced.*

También desde la botonera se abren distintas ventanas que permiten la configuración de los parámetros de referencia y el valor de la salida, que se utilizarán dependiendo del modo de operación activo, y los límites de éstas variables.

El botón *SetPoint* abre una nueva ventana en la que se puede modificar la referencia del sistema. El nuevo valor debe estar dentro del rango permitido para poder ser aplicado. Éste será el dato utilizado en el algoritmo de control cuando el modo *Regulation* esté activo.

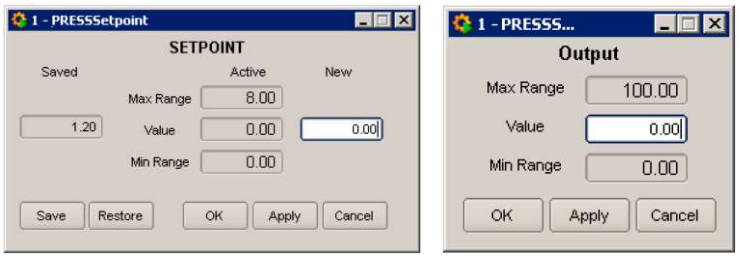


Figura 3.7 – Configuración del valor de la referencia y la variable manipulada

El botón *Output…,* de la misma forma que en el caso anterior, abre una ventana que permite determinar el valor de la variable manipulada cuando el modo *Output Positioning* esté activo.

En *Limits* se determinan los límites en las variables que se acaban de definir. Aunque el PFC no tenga incluidas las restricciones en el cálculo de la respuesta, se debe truncar el valor de salida del controlador si éste excede el rango permitido.

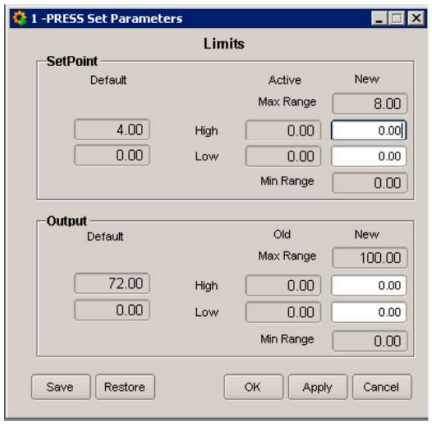


Figura 3.8 – Definición de límites

3. Dynamic Matrix Control

La implementación del *Dynamic Matrix Control* se realiza inicialmente en Matlab, pero sin perder de vista el objetivo final, la implementación del algoritmo en un PLC, por lo que siempre se deben tener en mente una serie de consideraciones.

En primer lugar, en un PLCs no cuentan con librerías específicas para trabajar con matrices, de modo que todas las operaciones de suma o multiplicación están programadas con bucles *For* en vez de utilizar las funciones propias de Matlab para ello. Esto es, en algunos casos, ventajoso desde el punto de vista computacional, puesto que al tener el control sobre la operación que se está ejecutando sobre cada elemento de la matriz es posible, en algunas ocasiones, agrupar operaciones o prescindir de algunas matrices auxiliares.

Por otro lado Matlab cuenta con diversos revolvedores de problemas de optimización QP, mientras que en el PLC será necesario implementarlos. De cualquier modo los algoritmos propios de Matlab servirán para contrastar los resultados que se han obtenido por otros medios. Obtención del modelo

El primer elemento, indispensable para la utilización de cualquier algoritmo MPC, es el modelo del sistema. Como se ha indicado en capítulos anteriores, el DMC basa sus cálculos en un modelo de respuesta salto, que se puede obtener de varias maneras.

Una opción es aplicar en cada una de las entradas y perturbaciones medibles del proceso una variación de valor conocido, Δu o Δv, que puede ser positivo o negativo y registrar los coeficientes obtenidos. Este método aplicado a un sistema real requiere normalmente de varios experimentos, puesto que el ruido de la medida afecta al modelo obtenido.

Una vez aplicado el escalón en la entrada seleccionada en el instante t, se almacenarán los valores obtenidos en cada periodo de muestreo para, a continuación, determinar el número N de coeficientes que son significativos en la respuesta, es decir, cuando el sistema haya alcanzado un nuevo estado estacionario e $y(t+i) - y(t+i-1) \cong 0$. Este valor suele estar comprendido entre 30 y 50. Si el número de coeficientes final estuviera fuera de este rango sería conveniente revisar el tiempo de muestreo que se ha fijado.

Es importante remarcar que, para la identificación del modelo, el sistema debe partir de un punto de funcionamiento estable $[\Delta u_0, \Delta v_o, \Delta y_o]$.

El último paso para obtener el vector de los coeficientes será normalizar los resultados obtenidos restando el valor inicial de la salida y dividiendo por el incremento aplicado.

$$\boldsymbol{g}_{ij} = \begin{bmatrix} g_{ij_1} \\ g_{ij_2} \\ \vdots \\ g_{ij_{N_i}} \end{bmatrix} = \left(\begin{bmatrix} y_i(t+1) \\ y_i(t+2) \\ \vdots \\ y_i(t+N_i) \end{bmatrix} - \begin{bmatrix} y_i(t) \\ y_i(t) \\ \vdots \\ y_i(t) \end{bmatrix} \right) \div \Delta u_j \qquad [3.4]$$

Una alternativa para evitar la problemática del ruido es la identificación del modelo en función de transferencia en el punto de trabajo, para la obtención posterior de la respuesta salto a partir de éste.

Una vez definidos los modelos de todas las salidas se definirán dos matrices, una para los modelos entre salidas y entradas y otra para las perturbaciones, estructuradas de la siguiente manera:

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_{11} & \cdots & \boldsymbol{g}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{g}_{m1} & \cdots & \boldsymbol{g}_{mn} \end{bmatrix}_{[\sum N_i, n]} \qquad \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_{11} & \cdots & \boldsymbol{d}_{1l} \\ \vdots & \ddots & \vdots \\ \boldsymbol{d}_{m1} & \cdots & \boldsymbol{d}_{ml} \end{bmatrix}_{[\sum N_i, l]} \qquad [3.5]$$

Donde:

- m → Número de salidas, que se identificarán por el índice i
- n → Número de entradas, que se identificarán por el índice j
- l → Número de perturbaciones medibles, que se identificarán por el índice k

A la hora de generar estas matrices es necesario tener en cuenta que el número N_i de coeficientes de las respuestas de cada salida debe ser el mismo para todas sus entradas y perturbaciones.

3.1. Preparación del problema QP

El objetivo del DMC es, en última instancia, la resolución de un problema de programación cuadrática, de modo que es necesario estructurar los datos de forma adecuada para después hacer la llamada al optimizador.

$$\min x = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x} \qquad [3.6]$$
$$Sujeto\ a: \quad \boldsymbol{A}x - \boldsymbol{b} \leq 0$$

En el Capítulo 2 ya se explicó cómo a partir de la función de costes se deduce la expresión de la ecuación [2.48]. A continuación se analizarán en detalle cada uno de los términos que aparecen en ella.

$$min\, J = \Delta \boldsymbol{u}^T(t)\, \overbrace{[\boldsymbol{G}^T \boldsymbol{\gamma} \boldsymbol{G} + \boldsymbol{\beta}]}^{\boldsymbol{H}}\, \Delta \boldsymbol{u}(t) - 2\, \overbrace{(\boldsymbol{G}^T\, \boldsymbol{\gamma} \boldsymbol{D})^T}^{-\boldsymbol{c}^T}\, \Delta \boldsymbol{u}(t) + \overbrace{\boldsymbol{D}^T \boldsymbol{D}}^{cte} \qquad [3.7]$$

3.1.1. Acción de control

El vector $\Delta \boldsymbol{u}$ contiene las acciones de control resultantes del proceso de optimización. Como para cada $\Delta \boldsymbol{u}_j$ se obtienen N_{u_j} valores, se obtendrá un vector columna de tamaño $\sum_{j=1}^{n} N_{u_j}$.

$$\Delta \boldsymbol{u} = \begin{bmatrix} \Delta \boldsymbol{u}_1 \\ \vdots \\ \Delta \boldsymbol{u}_n \end{bmatrix}_{[\sum Nu, 1]} \qquad \Delta \boldsymbol{u}_j = \begin{bmatrix} \Delta u_{j1} \\ \vdots \\ \Delta u_{jN_{u_j}} \end{bmatrix} \qquad [3.8]$$

3.1.2. Matriz G

La matriz $\boldsymbol{G}$ extendida al caso multivariable se compone de las matrices $\boldsymbol{G}_{ij}$ generadas a partir de cada uno de los modelos entrada-salida del sistema. Los elementos de esta matriz no cambian durante la ejecución del MPC, de modo que sólo es necesario calcularla al inicio o si alguno de los modelos sufre una modificación.

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_{11} & \cdots & \boldsymbol{G}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{G}_{m1} & \cdots & \boldsymbol{G}_{mn} \end{bmatrix}_{[\sum (N_{2_i} - N_{1_i} + 1), \sum N_{u_j}]} \qquad [3.9]$$

$$\boldsymbol{G_{ij}} = \begin{bmatrix} g_{ij_{N_{1_i}}} & \cdots & g_{ij_2} & g_{ij_1} & 0 & \cdots & 0 \\ g_{ij_{N_{1_i}+1}} & g_{ij_{N_1}} & \cdots & g_{ij_2} & g_{ij_1} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{ij_{N_{2_i}}} & \cdots & \cdots & \cdots & \cdots & \cdots & g_{ij_{N_{2_i} - N_{u_j} + 1}} \end{bmatrix}_{[N_{2_i} - N_{1_i} + 1, N_{u_j}]}$$

3.1.3. Matriz Hessiana

El cálculo de la matriz Hessiana requiere, además de la matriz $\boldsymbol{G}$, los pesos asignados a cada una de las salidas (Ecuación [3.10]) y de las acciones de control (Ecuación [3.11]). Para evitar que el orden de magnitud de las variables afecte a los pesos, el primer paso es normalizarlos dividiendo por el valor inicial de la salida/entrada correspondiente.

$$\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\gamma}_2 & \cdots & \cdots \\ \cdots & \cdots & \ddots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\gamma}_m \end{bmatrix}_{[\sum(N_{2_i}-N_{1_i}+1),\sum(N_{2_i}-N_{1_i}+1)]} \qquad [3.10]$$

$$\boldsymbol{\gamma}_i = \frac{100 \cdot \gamma_i}{y_{0_i}} \boldsymbol{I}_{[N_{2_i}-N_{1_i}+1,N_{2_i}-N_{1_i}+1]}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\beta}_2 & \cdots & \cdots \\ \cdots & \cdots & \ddots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\beta}_m \end{bmatrix}_{[\sum N_{u_j},\sum N_{u_j}]} \qquad \boldsymbol{\beta}_j = \frac{100 \cdot \beta_i}{u_{0_j}} \boldsymbol{I}_{[N_{u_j},N_{u_j}]} \qquad [3.11]$$

La matriz Hessiana de la función objetivo es una matriz cuadrada y simétrica. En teoría, por ser el producto de $\boldsymbol{G}^T\boldsymbol{G}$ debería ser simétrica siempre, aunque en la práctica por cuestiones de precisión puede llegar a no serlo. Una forma de asegurar el cumplimiento de esta condición es, una vez calculada la matriz calcular la media de la suma de ésta y su traspuesta.

$$\boldsymbol{H}_{\left[\sum N_{u_j},\sum N_{u_j}\right]} = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{H} + \boldsymbol{H}^T) \qquad [3.12]$$

3.1.4. Trayectoria de referencia

La trayectoria de referencia elegida es un sistema de primer orden para así conseguir acciones de control más robustas en el caso de una variación en la consigna del sistema. Es por ello que por cada salida existe un parámetro α_i con el que regular la velocidad de la respuesta.

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_m \end{bmatrix}_{[\sum(N_{2_i}-N_{1_i}+1),1]} \qquad [3.13]$$

3.1.5. Respuesta libre

La respuesta libre se calcula en tres etapas para así poder hacer los cálculos de forma recursiva y evitar la carga computacional que supone obtener las expresiones de los sumatorios en cada ciclo de ejecución.

El primer paso consiste en desplazar una posición hacia arriba los elementos del vector $\boldsymbol{L}_{ij}$ y rellenar la última posición con el valor anterior. Después, se suma el producto de las matrices $\boldsymbol{g}_{ij}$ y $\boldsymbol{d}_{il}$ multiplicado por el último incremento de las variables de control Δu_j y las perturbaciones Δv_l.

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 \\ \vdots \\ \boldsymbol{L}_m \end{bmatrix}_{[\sum N_i, 1]} \quad \boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_m \end{bmatrix}_{[\sum N_i n]} \quad \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_m \end{bmatrix}_{[\sum N_i, l]} \tag{3.14}$$

$$\boldsymbol{L}_i(k) = \begin{bmatrix} l_{i_1}(k|k) \\ l_{i_2}(k+1|k) \\ \vdots \\ l_{i_{N_{ij}}}(k+n|k) \end{bmatrix}_{[N_i,1]} = \begin{bmatrix} l_{i_2}(k+1|k) \\ \vdots \\ l_{i_{N_{ij}}}(k+n|k) \\ l_{i_{N_{ij}}}(k+n|k) \end{bmatrix}_{[N_i,1]} + \boldsymbol{g}_i \Delta u + \boldsymbol{d}_i \Delta v \tag{3.15}$$

El resultado que se obtiene realizando el cálculo de esta manera es equivalente a:

$$l(t+j) = \sum_{i=1}^{N} g_i \Delta u(t-i) + \sum_{i=1}^{N} d_i \Delta v(t-i) \tag{3.16}$$

El siguiente paso es añadir a la respuesta libre el efecto causado por las perturbaciones no medibles. Como se explicó anteriormente, este efecto se considera como la diferencia entre la salida del proceso y la del modelo en el instante k, la cual coincide con el primer elemento del vector calculado en la primera etapa, de modo que resulta la siguiente expresión:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_1 \\ \vdots \\ \boldsymbol{T}_m \end{bmatrix}_{[\sum(N_{2_i}), \mathbf{1}]} \tag{3.17}$$

Se puede apreciar que el número de filas de la matriz $\boldsymbol{T}$ no coincide con el de $\boldsymbol{L}$, aunque después se sumen. Para poder realizar esta operación, si $N_{2_i} > N_i$ se repetirá el elemento N_i tantas veces como sea necesario. Si por el contrario, $N_i > N_{2_i}$ sólo se considerarán los N_{2_i} primeros elementos.

$$\boldsymbol{T_i}(k) = \begin{bmatrix} T_{i_1}(k|k) \\ T_{i_2}(k+1|k) \\ \vdots \\ T_{i_N}(k+n|k) \end{bmatrix}_{[N_{2i},1]} = \begin{bmatrix} l_{i_1}(k|k) \\ l_{i_2}(k+1|k) \\ \vdots \\ l_{i_N}(k+n|k) \end{bmatrix}_{[N_{2i},1]} + (y_p - l_{i_1}(k|k)) \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{[N_{2i},1]} \quad [3.18]$$

La ecuación [3.19] muestra el resultado equivalente de realizar esta operación:

$$t(t+j) = y_p(t) + \sum_{i=1}^{N} (g_{j+i} - g_i)\Delta u(t-i) + \sum_{i=1}^{N} (d_{j+i} - d_i)\Delta v(t-i) \quad [3.19]$$

Por último resta incluir el efecto sobre la respuesta libre de las perturbaciones medibles en el futuro. Se considera como hipótesis para ello que el valor de éstas se mantiene constante durante la predicción y de valor $\Delta v(k+n) = \Delta v(k)$.

$$\boldsymbol{d}_{sum} = \begin{bmatrix} \boldsymbol{d}_{sum_1} \\ \vdots \\ \boldsymbol{d}_{sum_m} \end{bmatrix}_{[\sum \boldsymbol{N_i}, \boldsymbol{1}]} \quad \boldsymbol{d}_{sum_i} = \begin{bmatrix} \boldsymbol{d_{i_1}} \\ \boldsymbol{d_{i_1}} + \boldsymbol{d_{i_2}} \\ \vdots \\ \sum \boldsymbol{d_i} \end{bmatrix}_{[\boldsymbol{N},\boldsymbol{1}]} \quad [3.20]$$

$$\boldsymbol{T}_i(k) = \begin{bmatrix} t_1(k|k) \\ t_2(k+1|k) \\ \vdots \\ t_N(k+n|k) \end{bmatrix}_{[\boldsymbol{N_i},1]} = \begin{bmatrix} t_1(k|k) \\ t_2(k+1|k) \\ \vdots \\ t_N(k+n|k) \end{bmatrix}_{[\boldsymbol{N_i},1]} + \boldsymbol{d_{sum}}\Delta v \quad [3.21]$$

De modo que con este último cálculo se obtiene la respuesta predicha tal y como fue definida en el Capítulo 2.

$$t(t+j) = y_p(t) + \sum_{i=1}^{N} (g_{j+i} - g_i)\Delta u(t-i) + \sum_{i=1}^{N} (d_{j+i} - d_i)\Delta v(t-i) + \sum_{i=1}^{j} d_i \Delta v(t+j-i) \quad [3.22]$$

3.1.6. Vector gradiente

Una vez calculada la respuesta libre y definida la trayectoria de referencia, se puede calcular el vector del gradiente, que se definió en la ecuación [3.7] como:

$$\mathbf{c}^{\mathrm{T}} = -(\boldsymbol{G}^T \boldsymbol{\gamma} \boldsymbol{D})^T \quad [3.23]$$

La expresión anterior se puede descomponer en dos partes. En primer lugar, el producto $-\boldsymbol{G}^T\,\boldsymbol{\gamma}$ que no varía a no ser que lo hagan los parámetros de configuración del problema y el vector $\boldsymbol{D}$ que contiene el error de predicción y por tanto varía en cada ciclo del controlador.

$$\boldsymbol{c_{Aux}} = -\boldsymbol{G}^T\,\boldsymbol{\gamma} \qquad [3.24]$$

$$\boldsymbol{D}(k) = \boldsymbol{R}(k) - \boldsymbol{T}(k) \qquad [3.25]$$

Con lo que resulta:

$$\boldsymbol{c}^T = (\boldsymbol{c_{Aux}}\boldsymbol{D})^T \qquad [3.26]$$

3.1.7. Restricciones

La implementación propuesta contiene las siguientes restricciones sobre las variables controladas y los incrementos en las variables manipuladas:

$$\begin{aligned} \Delta u_{max} \geq \Delta u(t+j) \geq \Delta u_{min} \\ u_{max} \geq u(t+j) \geq u_{min} \\ \hat{y}_{max} \geq \hat{y}(t+j) \geq \hat{y}_{min} \end{aligned} \qquad [3.27]$$

Que expresado en la notación del optimizador QP es:

$$\begin{aligned} \Delta u(t+j) \leq \Delta u_{max} \\ -\Delta u(t+j) \leq -\Delta u_{min} \\ u(t+j) \leq u_{max} \\ -u(t+j) \leq -u_{min} \\ \hat{y}(t+j) \leq \hat{y}_{max} \\ -\hat{y}(t+j) \leq -\hat{y}_{min} \end{aligned} \qquad [3.28]$$

El término que está a la izquierda de la desigualdad será el que conforme la matriz $\boldsymbol{A}$ y el de la derecha el vector $\boldsymbol{b}$ de la siguiente forma.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{\Delta u_{max}} \\ \boldsymbol{A}_{\Delta u_{min}} \\ \boldsymbol{A}_{u_{max}} \\ \boldsymbol{A}_{u_{min}} \\ \boldsymbol{A}_{\hat{y}_{max}} \\ \boldsymbol{A}_{\hat{y}_{min}} \end{bmatrix}_{[4\sum N_{u_j} + 2\sum N_{2_i}, \sum N_{u_j}]} \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_{\Delta u_{max}} \\ \boldsymbol{b}_{\Delta u_{min}} \\ \boldsymbol{b}_{u_{max}} \\ \boldsymbol{b}_{u_{min}} \\ \boldsymbol{b}_{\hat{y}_{max}} \\ \boldsymbol{b}_{\hat{y}_{min}} \end{bmatrix}_{[4\sum N_{u_j} + 2\sum N_{2_i}, 1]} \qquad [3.29]$$

Las matrices $\boldsymbol{A}_{\Delta u}$ verifican que las acciones de control resultantes de la optimización están dentro del rango establecido, de manera que cada una de ellas es una matriz identidad de dimensiones $[\sum N_{u_j}, \sum N_{u_j}]$ y los vectores $\boldsymbol{b}_{\Delta u}$ correspondientes contienen directamente el valor máximo o mínimo correspondiente.

$$\boldsymbol{A}_{\Delta u_{max}} = \begin{bmatrix} \boldsymbol{A}_{u_{max_1}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{A}_{u_{max_n}} \end{bmatrix}_{[\sum N_{u_j}, \sum N_{u_j}]} \quad \boldsymbol{b}_{\Delta u_{max}} = \begin{bmatrix} \boldsymbol{b}_{\Delta u_{max_1}} \\ \vdots \\ \boldsymbol{b}_{\Delta u_{max_n}} \end{bmatrix}_{[\sum N_{u_j}, 1]}$$

$$\boldsymbol{A}_{\Delta u_{min}} = \begin{bmatrix} \boldsymbol{A}_{u_{min_{11}}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{A}_{u_{min_n}} \end{bmatrix}_{[\sum N_{u_j}, \sum N_{u_j}]} \quad \boldsymbol{b}_{\Delta u_{min}} = \begin{bmatrix} \boldsymbol{b}_{\Delta u_{min_1}} \\ \vdots \\ \boldsymbol{b}_{\Delta u_{min_j}} \end{bmatrix}_{[\sum N_{u_j}, 1]} \qquad [3.30]$$

Donde:

$$\boldsymbol{A}_{\Delta u_{max_j}} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{[N_{u_j}, N_{u_j}]} \quad \boldsymbol{b}_{\Delta u_{max_j}} = \begin{bmatrix} \Delta u_{max_j} \\ \vdots \\ \Delta u_{max_j} \end{bmatrix}_{[N_{u_j}, 1]}$$

$$\boldsymbol{A}_{\Delta u_{min_j}} = -\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{[N_{u_j}, N_{u_j}]} \quad \boldsymbol{b}_{\Delta u_{min_j}} = -\begin{bmatrix} \Delta u_{min_j} \\ \vdots \\ \Delta u_{min_j} \end{bmatrix}_{[N_{u_j}, 1]} \qquad [3.31]$$

Las matrices $\boldsymbol{A}_u$ deben verificar, en cambio, que la suma de las acciones de control que se van obteniendo durante el horizonte de control no exceden los límites del rango, de manera que las submatrices $\boldsymbol{A}_{u_{ij}}$ serán matrices diagonales inferiores. Los vectores $\boldsymbol{b}_{u_{ij}}$ por su parte deben tener en cuenta, además del valor del límite impuesto, el valor inicial de la acción de control en comienzo de la predicción.

$$u(k+j) = u(k) + \sum_{i=1}^{j} \Delta u(k+i) \qquad [3.32]$$

$$\boldsymbol{A}_{u_{max}} = \begin{bmatrix} \boldsymbol{A}_{u_{max_1}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{A}_{u_{max_j}} \end{bmatrix}_{[\sum N_{u_j}, \sum N_{u_j}]} \quad \boldsymbol{b}_{u_{max}} = \begin{bmatrix} \boldsymbol{b}_{u_{max_1}} \\ \vdots \\ \boldsymbol{b}_{u_{max_j}} \end{bmatrix}_{[\sum N_{u_j}, 1]}$$

$$\boldsymbol{A}_{u_{min}} = \begin{bmatrix} \boldsymbol{A}_{u_{min_1}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{A}_{u_{min_j}} \end{bmatrix}_{[\sum N_{u_j}, \sum N_{u_j}]} \quad \boldsymbol{b}_{u_{min}} = \begin{bmatrix} \boldsymbol{b}_{u_{min_1}} \\ \vdots \\ \boldsymbol{b}_{u_{min_j}} \end{bmatrix}_{[\sum N_{u_j}, 1]} \qquad [3.33]$$

Donde:

$$\boldsymbol{A}_{u_{max_j}} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{[N_{u_j}, N_{u_j}]} \quad \boldsymbol{b}_{u_{max_j}} = \begin{bmatrix} u_{max_j} - u_1(k) \\ \vdots \\ u_{max_j} - u_j(k) \end{bmatrix}_{[N_{u_j}, 1]}$$

$$\boldsymbol{A}_{u_{min_j}} = -\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{[N_{u_j}, N_{u_j}]} \quad \boldsymbol{b}_{u_{min_j}} = -\begin{bmatrix} u_{min_j} - u_1(k) \\ \vdots \\ u_{min_j} - u_j(k) \end{bmatrix}_{[N_{u_j}, 1]} \qquad [3.34]$$

Las matrices $\boldsymbol{A}_{\hat{y}}$ y $\boldsymbol{b}_{\hat{y}}$ deben contemplar que el valor de la salida predicha del modelo no sobrepasa el rango permitido durante todo el horizonte de predicción, es decir, entre $[1, N_2]$ y no solamente en $[N_1, N_2]$. La predicción de la salida está compuesta por la respuesta libre del sistema más la respuesta forzada, de forma que las desigualdades se pueden reescribir como:

$$\begin{aligned} \boldsymbol{T} + \boldsymbol{G}_g \Delta \boldsymbol{u} \leq \hat{\boldsymbol{y}}_{max} \quad &\rightarrow \quad \boldsymbol{G}_g \Delta \boldsymbol{u} \leq \hat{\boldsymbol{y}}_{max} - \boldsymbol{T} \\ -\left(\boldsymbol{T} + \boldsymbol{G}_g \Delta \boldsymbol{u}\right) \leq -\hat{\boldsymbol{y}}_{min} \rightarrow \; & -\boldsymbol{G}_g \Delta \boldsymbol{u} \leq -\hat{\boldsymbol{y}}_{min} + \boldsymbol{T} \end{aligned} \qquad [3.35]$$

La matriz $\boldsymbol{G}_g$, que aparece por primera vez en la ecuación [3.35], tiene la misma estructura diagonal que la matriz $\boldsymbol{G}$ que se había utilizado hasta el momento, con la diferencia de que no está truncada desde N_1.

$$\boldsymbol{G}_g = \begin{bmatrix} \boldsymbol{G}_{g_{11}} & \cdots & \boldsymbol{G}_{g_{1n}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{G}_{g_{m1}} & \cdots & \boldsymbol{G}_{g_{mn}} \end{bmatrix}_{[\sum(N_{2i}), \sum N_{u_j}]} \qquad [3.36]$$

$$\boldsymbol{G}_{g_{ij}} = \begin{bmatrix} g_{ij_1} & \cdots & 0 & 0 & 0 & \cdots & 0 \\ g_{ij_2} & g_{ij_1} & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{ij_{N_{2_i}}} & \cdots & \cdots & \cdots & \cdots & \cdots & g_{ij_{N_{2_i}-N_{u_j}+1}} \end{bmatrix}_{[N_{2_i},N_{u_j}]}$$

Por tanto las restricciones sobre el valor de la salida se expresarán como:

$$\boldsymbol{A}_{y_{max}} = \begin{bmatrix} \boldsymbol{A}_{y_{max_{11}}} & \cdots & \boldsymbol{A}_{y_{max_{1n}}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{y_{max_{m1}}} & \cdots & \boldsymbol{A}_{y_{max_{mn}}} \end{bmatrix}_{[\sum N_{2_i},\sum N_{u_j}]} \quad \boldsymbol{b}_{y_{max}} = \begin{bmatrix} \boldsymbol{b}_{y_{max_i}} \\ \vdots \\ \boldsymbol{b}_{y_{max_i}} \end{bmatrix}_{[\sum N_{2_i},1]}$$

$$\boldsymbol{A}_{y_{min}} = \begin{bmatrix} \boldsymbol{A}_{y_{min_{11}}} & \cdots & \boldsymbol{A}_{y_{min_{1n}}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{y_{min_{m1}}} & \cdots & \boldsymbol{A}_{y_{min_{mn}}} \end{bmatrix}_{[\sum N_{2_i},\sum N_{u_j}]} \quad \boldsymbol{b}_{y_{min}} = \begin{bmatrix} \boldsymbol{b}_{y_{min_i}} \\ \vdots \\ \boldsymbol{b}_{y_{min_i}} \end{bmatrix}_{[\sum N_{2_i},1]} \qquad [3.37]$$

Donde:

$$\boldsymbol{A}_{y_{max_{ij}}} = \boldsymbol{G}_{g_{ij}[N_{2_i},N_{u_j}]} \quad \boldsymbol{b}_{y_{max_{ij}}} = \begin{bmatrix} y_{max_i} \\ \vdots \\ y_{max_i} \end{bmatrix}_{[N_{2_i},1]}$$

$$\boldsymbol{A}_{y_{min_{ij}}} = -\boldsymbol{G}_{g_{ij}\left[N_{2_i},N_{u_j}\right]} \quad \boldsymbol{b}_{y_{min_{ij}}} = -\begin{bmatrix} y_{min_i} \\ \vdots \\ y_{min_i} \end{bmatrix}_{[N_{2_i},1]} \qquad [3.38]$$

3.2. Relajación de las restricciones

Las restricciones, tal y como han sido definidas hasta el momento, han de verificarse necesariamente durante la resolución del problema de optimización, lo cual puede acarrear problemas de factibilidad.

Una forma de tratar este inconveniente es transformar las restricciones duras en blandas, es decir, en vez de obligar a que las condiciones se satisfagan se opta por penalizar fuertemente su incumplimiento. Esto se consigue introduciendo lo que se conoce como variables de holgura.

Las variables de holgura ε cuantifican el grado de incumplimiento de la restricción y son un nuevo grado de libertad que se introduce en el problema de optimización y que, por lo tanto, debe ser minimizado.

La restricción que se ha decidido relajar es la concerniente al valor de la salida, mientras que las relativas a las acciones de control se mantienen como restricciones duras.

$$\begin{gathered} \Delta u_{max} \geq \Delta u(t+j) \geq \Delta u_{min} \\ u_{max} \geq u(t+j) \geq u_{min} \\ \varepsilon + \hat{y}_{max} \geq \hat{y}(t+j) \geq \hat{y}_{min} - \varepsilon \\ \varepsilon \geq 0 \end{gathered} \qquad [3.39]$$

La función de costes debe ser reformulada para incluir estas nuevas variables de decisión. Con el fin de ponderar el peso de las variables de holgura en la función objetivo se introducen las variables ρ.

$$J = [\Delta \boldsymbol{u}^T(t) \quad \boldsymbol{\varepsilon}(t)] \overbrace{\begin{bmatrix} \boldsymbol{G}^T\boldsymbol{\gamma}\boldsymbol{G} + \boldsymbol{\beta} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\rho} \end{bmatrix}}^{\boldsymbol{H}} \begin{bmatrix} \Delta \boldsymbol{u}(t) \\ \boldsymbol{\varepsilon}(t) \end{bmatrix} - 2 \overbrace{[(\boldsymbol{G}^T\boldsymbol{\gamma}\boldsymbol{e}_0)^T \quad \boldsymbol{0}]}^{\boldsymbol{c}^T} \begin{bmatrix} \Delta \boldsymbol{u}(t) \\ \boldsymbol{\varepsilon}(t) \end{bmatrix} \qquad [3.40]$$

3.2.1. Variables de decisión

Antes de introducir las variables de holgura, las únicas variables de decisión del problema eran las acciones de control mientras que, ahora, también es necesario introducir las variables de holgura correspondientes al número de salidas que tenga el problema.

$$\Delta \boldsymbol{u} = \begin{bmatrix} \Delta \boldsymbol{u}_1 \\ \vdots \\ \Delta \boldsymbol{u}_n \end{bmatrix}_{[\sum Nu,1]} = \begin{bmatrix} \begin{bmatrix} \Delta u_{11} & \cdots & \Delta u_{1N_{u_1}} \end{bmatrix}^T \\ \vdots \\ \begin{bmatrix} \Delta u_{n1} & \cdots & \Delta u_{nN_{u_1}} \end{bmatrix}^T \end{bmatrix} \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_{\mathrm{i}} \end{bmatrix} \qquad [3.41]$$

3.2.2. Matriz H

La matriz Hessiana es ahora una extensión de lo que se definió en el apartado anterior. En línea con la diagonal principal se ha añadido la matriz $\boldsymbol{\rho}$ que contiene los pesos de las variables de holgura y todas las demás posiciones se completan con ceros.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{G}^T\boldsymbol{\gamma}\boldsymbol{G} + \boldsymbol{\beta} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\rho} \end{bmatrix}_{\left[(\sum N_{u_j})+m,(\sum N_{u_j})+m\right]} \quad \boldsymbol{\rho} = \begin{bmatrix} \rho_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \rho_m \end{bmatrix} \qquad [3.42]$$

3.2.3. Gradiente

De igual modo, es necesario añadir al vector del gradiente tantos ceros como salidas tenga el sistema.

$$\boldsymbol{c}^T = [(\boldsymbol{G}^T \boldsymbol{\gamma} \boldsymbol{e}_0)^T \quad \boldsymbol{0}]_{\left[1,(\sum N_{u_j})+m\right]} \tag{3.43}$$

3.2.4. Restricciones

Las restricciones quedan reformuladas ahora del siguiente modo:

$$\begin{aligned} \Delta u(t+j) &\leq \Delta u_{max} \\ -\Delta u(t+j) &\leq -\Delta u_{min} \\ u(t+j) &\leq u_{max} \\ -u(t+j) &\leq -u_{min} \\ \hat{y}(t+j) &\leq \hat{y}_{max} - \varepsilon \\ -\hat{y}(t+j) &\leq -\hat{y}_{min} - \varepsilon \end{aligned} \tag{3.44}$$

Y por tanto las matrices $\boldsymbol{A}$ y $\boldsymbol{b}$ resultan:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{\Delta u_{max}} & \boldsymbol{0} \\ \boldsymbol{A}_{\Delta u_{min}} & \boldsymbol{0} \\ \boldsymbol{A}_{u_{max}} & \boldsymbol{0} \\ \boldsymbol{A}_{u_{min}} & \boldsymbol{0} \\ \boldsymbol{A}_{\hat{y}_{max}} & \boldsymbol{A}_{\varepsilon} \\ \boldsymbol{A}_{\hat{y}_{min}} & \boldsymbol{A}_{\varepsilon} \\ \boldsymbol{0} & \boldsymbol{A}_{\varepsilon_{min}} \end{bmatrix}_{[4\sum N_{u_j}+2\sum N_{2_i}, \sum N_{u_j}+m]} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_{\Delta u_{max}} \\ \boldsymbol{b}_{\Delta u_{min}} \\ \boldsymbol{b}_{u_{max}} \\ \boldsymbol{b}_{u_{min}} \\ \boldsymbol{b}_{\hat{y}_{max}} \\ \boldsymbol{b}_{\hat{y}_{min}} \\ \boldsymbol{b}_{\varepsilon_{min}} \end{bmatrix}_{[4\sum N_{u_j}+2\sum N_{2_i}+m} \tag{3.45}$$

Las submatrices que fueron definidas para caso anterior (restricciones duras) no sufren ninguna modificación. Las matrices nuevas que aparecen pasan a definirse a continuación. Todas las posiciones de la matriz $\boldsymbol{A}$ que no se han definido se completan con ceros.

$$\boldsymbol{A}_{\varepsilon_{min}} = -\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{[m,m]} \quad \boldsymbol{b}_{\varepsilon_{min}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{[m,1]} \tag{3.46}$$

$$\boldsymbol{A}_{\varepsilon} = \begin{bmatrix} \boldsymbol{A}_{\varepsilon_1} \\ \vdots \\ \boldsymbol{A}_{\varepsilon_m} \end{bmatrix}_{[\sum N_{2_i}, m]} \qquad \boldsymbol{A}_{\varepsilon_i} = \begin{bmatrix} \boldsymbol{A}_{\varepsilon_{i_1}} & \cdots & \boldsymbol{A}_{\varepsilon_{i_m}} \end{bmatrix}_{[N_{2_i}, m]}$$

$$\boldsymbol{A}_{\varepsilon_{i_x}} = -\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{[N_{2_i}, 1]} \quad Si\ i = x \qquad \boldsymbol{A}_{\varepsilon_{i_x}} = -\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{[N_{2_i}, 1]} \quad Si\ i \neq x$$

3.3. Optimizador: QP Hildreth

El método de optimización que se implementará es el algoritmo para QP de Hildreth, un método clásico pero sencillo de implementar (Wang, 2009).

El algoritmo calcula la solución en dos pasos. En primer lugar, obtiene la solución analítica del problema sin restricciones. Si esta solución no viola ninguna de las restricciones, se adopta. En caso contrario, pasa a tratar de resolver el problema con restricciones.

El algoritmo permite definir el número de iteraciones máximas permitidas en cada llamada al optimizador, de manera que, si en el intervalo definido no se ha obtenido una solución que verifique todas las restricciones, la salida de la función será la última que haya generado, es decir, la mejor hasta ese instante.

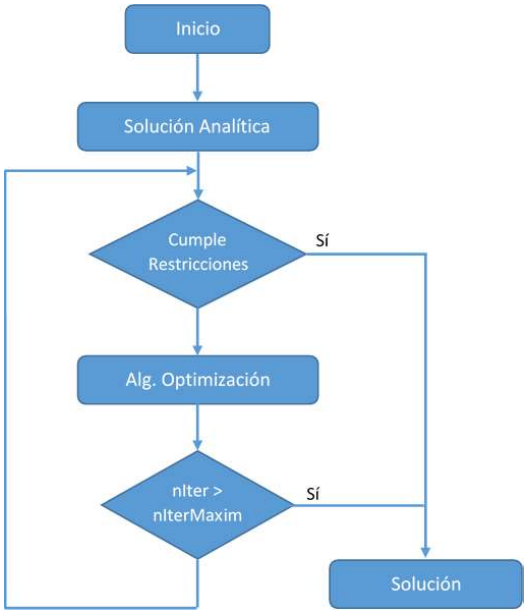


Figura 3.9 – Diagrama de flujo del optimizador

Uno de los inconvenientes de este algoritmo reside en el tamaño de las matrices que utiliza en el algoritmo. En el caso en que la solución no puede ser obtenida de forma analítica, el optimizador requiere de la utilización de las matrices $\boldsymbol{D}_{aux}$ y $\boldsymbol{P}$, cuyo tamaño es indudablemente el mayor del problema.

$$[\boldsymbol{D}_{aux}]_{\left[(\sum N_{u_j})+m,(4\sum N_{u_j}+2\sum N_{2_i},\sum N_{u_j})+m\right]} = \boldsymbol{H}^{-1}\boldsymbol{A}^{T}$$

$$[\boldsymbol{P}]_{\left[,(4\sum N_{u_j}+2\sum N_{2_i},\sum N_{u_j})+m,(4\sum N_{u_j}+2\sum N_{2_i},\sum N_{u_j})+m\right]} = \boldsymbol{A}\,\boldsymbol{P}_{aux} \qquad [3.47]$$

3.4. Ejecución del controlador DMC

La ejecución del algoritmo del DMC se compone de varias etapas, entre las cuales se puede diferenciar claramente aquellas que se pueden realizar solamente en la inicialización del controlador o si se ha producido algún cambio en los parámetros de éste (Matrices constantes) y las que deben ejecutarse cada periodo de muestreo (Matrices variables) [Figura 3.10].

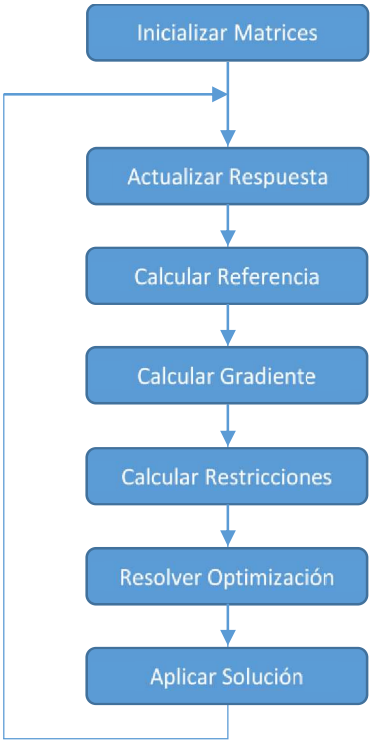


Figura 3.10 – Diagrama de flujo de funcionamiento del DMC

Con el fin de reducir lo máximo posible la carga computacional que implica la ejecución del algoritmo en cada iteración, se pre-calcularán todos aquellos valores que sea posible. A continuación se listan ambos tipos de datos.

3.4.1. Matrices constantes

Son fundamentalmente todas aquellas cuyos datos sólo dependen del modelo y los parámetros de configuración del sistema.

Matriz	Descripción
$\boldsymbol{g}$	Modelos salida/entrada
$\boldsymbol{d}$	Modelos salida/perturbación
$\boldsymbol{G_g}$	Matriz de coeficientes sin truncar
$\boldsymbol{G}$	Matriz de coeficientes truncada desde N_1
$\boldsymbol{H}$	Matriz Hessiana
$\boldsymbol{H^{-1}}$	Inversa de la matriz Hessiana
$\boldsymbol{A}$	Coeficientes de las restricciones
$\boldsymbol{C_{aux}}$	Matriz auxiliar para el cálculo del gradiente
$\boldsymbol{b_{\Delta u_{max}}}$	Término independiente de las restricciones sobre Δu_{max}
$\boldsymbol{b_{\Delta u_{min}}}$	Término independiente de las restricciones sobre Δu_{min}
$\boldsymbol{d_{aux}}$	Matriz utilizada por el optimizador
$\boldsymbol{P}$	Matriz utilizada por el optimizador

Tabla 1 – Matrices de valor constante en la ejecución del DMC

3.4.2. Matrices variables

Aquellas matrices que requieren los últimos datos obtenidos del sistema o aplicados sobre éste deben ser recalculadas en cada ciclo.

Matriz	Descripción
$\boldsymbol{\Delta u}$	Acciones de control calculadas
$\boldsymbol{L}$	Matriz auxiliar para el cálculo de la respuesta libre
$\boldsymbol{T}$	Respuesta libre
$\boldsymbol{R}$	Trayectorias de referencia del sistema
$\boldsymbol{D}$	Gradiente de la función de costes
$\boldsymbol{b_{u_{max}}}$	Término independiente de las restricciones sobre u_{max}
$\boldsymbol{b_{u_{min}}}$	Término independiente de las restricciones sobre u_{min}
$\boldsymbol{b_{\hat{y}_{max}}}$	Término independiente de las restricciones sobre $\hat{y}_{max}$
$\boldsymbol{b_{\hat{y}_{min}}}$	Término independiente de las restricciones sobre $\hat{y}_{min}$

Tabla 2 – Matrices de valor variable en la ejecución del DMC

3.5. Implementación del DMC en el PLC

Los PLCs son dispositivos que no están ideados con el fin de realizar cálculos matemáticos complejos o computacionalmente costosos y por otra parte cuentan con una capacidad de memoria de almacenamiento de programa y de datos muy limitada.

Una vez que los tamaños de las matrices involucradas en el algoritmo han sido definidos, es posible realizar un análisis del espacio de almacenamiento que se requiere.

Nombre		Tipo	Tamaño	
			Filas	Columnas
$nSal$	m	Int	1	
$nEnt$	n	Int	1	
$nPer$	l	Int	1	
$nCoef$	N_i	Int	1	m
Sum_nCoef	$\sum N_i$	Int	1	
g		Float	$\sum N_i$	n
d		Float	$\sum N_i$	l
Ts		Float	1	

Tabla 3 – Definición del modelo

Nombre		Tipo	Tamaño	
			Filas	Columnas
$N1$		Int	1	m
$N2$		Int	1	m
Nu		Int	1	n
$gamma$		Float	1	m
$beta$		Float	1	n
rho		Float	1	m
Sum_N2	$\sum N_{2_i}$	Int	1	
Sum_N2_N1	$\sum(N_{2_i} - N_{1_i} + 1)$	Int	1	
Sum_Nu	$\sum N_{u_j}$	Int	1	
Du_{Min}		Float	1	n
Du_{Max}		Float	1	n
u_{Min}		Float	1	n
u_{Max}		Float	1	n
y_{Min}		Float	1	m
y_{Max}		Float	1	m

Tabla 4 – Sintonización del DMC

Nombre	Tipo	Tamaño	
		Filas	Columnas
$\boldsymbol{u}$	Float	n	1
$\boldsymbol{u_{Old}}$	Float	n	1
$\boldsymbol{Du}$	Float	n	1
$\boldsymbol{Du_p}$	Float	$\sum N_{u_j}$	1
$\boldsymbol{v}$	Float	l	1
$\boldsymbol{v_{Old}}$	Float	l	1
$\boldsymbol{Dv}$	Float	l	1
$\boldsymbol{y_m}$	Float	1	m

Tabla 5 – Vectores de datos del proceso

Nombre	Tipo	Tamaño	
		Filas	Columnas
$\boldsymbol{G_g}$	Float	$\sum N_{u_j}$	$\sum N_{2_i}$
$\boldsymbol{G}$	Float	$\sum N_{u_j}$	$\sum(N_{2_i} - N_{1_i} + 1)$
$\boldsymbol{D_{Sum}}$	Float	m	$\sum N_{2_i}$
$\boldsymbol{L}$	Float	$\sum N_i$	1
$\boldsymbol{T}$	Float	$\sum N_{2_i}$	1
$\boldsymbol{R}$	Float	$\sum N_{2_i}$	1
$\boldsymbol{D}$	Float	$\sum(N_{2_i} - N_{1_i} + 1)$	1
$\boldsymbol{G_{Aux}}$	Float	$\sum N_{u_j}$	$\sum(N_{2_i} - N_{1_i} + 1)$
$\boldsymbol{C_{Aux}}$	Float	$\sum N_{u_j}$	$\sum(N_{2_i} - N_{1_i} + 1)$
$\boldsymbol{C}$	Float	$\sum N_{u_j} + m$	1
$\boldsymbol{H}$	Float	$\sum N_{u_j} + m$	$\sum N_{u_j} + m$
$\boldsymbol{A}$	Float	$4\sum N_{u_j} + 2\sum N_{2_i} + m$	$\sum N_{u_j} + m$
$\boldsymbol{B}$	Float	$4\sum N_{u_j} + 2\sum N_{2_i} + m$	1

Tabla 6 – Cálculo de datos del problema de optimización

Nombre	Tipo	Tamaño	
		Filas	Columnas
$\boldsymbol{H_{Aux}}$	Float	$\sum N_{u_j} + m$	$\sum N_{u_j} + m$
$\boldsymbol{H_{Inv}}$	Float	$\sum N_{u_j} + m$	$\sum N_{u_j} + m$

Tabla 7 – Inversión de la matriz Hessiana

Nombre	Tipo	Tamaño	
		Filas	Columnas
$\boldsymbol{H_{Aux}}$	Float	$\sum N_{u_j} + m$	$\sum N_{u_j} + m$
$\boldsymbol{H_{Inv}}$	Float	$\sum N_{u_j} + m$	$\sum N_{u_j} + m$
$\boldsymbol{Aux_2}$ $(\boldsymbol{P_{Aux}})$	Float	$\sum N_{u_j} + m$	$4\sum N_{u_j} + 2\sum N_{2_i} + m$
$\boldsymbol{P}$	Float	$4\sum N_{u_j} + 2\sum N_{2_i} + m$	$4\sum N_{u_j} + 2\sum N_{2_i} + m$
$\boldsymbol{eta}$	Float	$\sum N_{u_j} + m$	1
$\boldsymbol{lambda}$	Float	$4\sum N_{u_j} + 2\sum N_{2_i} + m$	1
$\boldsymbol{lambda\_p}$	Float	$4\sum N_{u_j} + 2\sum N_{2_i} + m$	1
$\boldsymbol{w}$	Float	1	
$\boldsymbol{la}$	Float	1	
$\boldsymbol{al}$	Float	1	

Tabla 8 – Cálculos del optimizador

El objetivo de la implementación del DMC en un PLC es controlar sistemas multivariables con restricciones pequeños, con pocas entradas y salidas. Si se calcula el espacio requerido para un sistema con la configuración y las dimensiones que se especifican en la Tabla 9, se obtiene que el tamaño requerido para las matrices del controlador es de unos 315 KBytes, de los cuales 236000 se consumen en la matriz $\boldsymbol{P}$ del controlador.

Tamaño del problema		Tamaño de los datos
nSal_Max	3	313188 Bytes
nEnt_Max	3	
nPer_Max	3	
nCoef_Max	100	
N2_Max	30	
Nu_Max	5	

Tabla 9 – Tamaño requerido por las matrices

Los PLCs de Siemens no permiten dimensionar las matrices en función de los parámetros del problema, por lo es necesario predefinir el tamaño máximo de todas las matrices, aunque hay que recalcar que las operaciones que se realizan tienen siempre en cuenta la dimensión real del problema.

Otra de las particularidades de la implementación en el PLC es el hecho de que el tamaño máximo de una estructura de datos está limitado al de un *Data Block*, de manera que si una matriz excede los 64KB permitidos en una sola DB es necesario partirla en varias.

Capítulo 4 . VALIDACIÓN DE RESULTADOS

1. Introducción

La verificación del funcionamiento de los algoritmos MPC implementados se realiza, al igual que su implementación, en dos fases. En primer lugar, se comprobarán los resultados obtenidos en Matlab o EcosimPro para, a continuación, contrastar sobre ellos la implementación en el PLC.

2. Predictive Functional Control

El caso de estudio del que se dispone para testear el PFC es un lazo de regulación de presión. Este proceso no es el modelo más representativo para estudiar el buen comportamiento del controlador puesto que se trata de un sistema con un retardo prácticamente nulo, pero sirve para verificar su comportamiento en un proceso real.

El sistema está compuesto por los siguientes elementos:

- Un autómata Siemens 315-2 PN/DP provisto de entradas y salidas analógicas y digitales.
- Un variador de frecuencia Omron VS mini J7.
- Un compresor JUN-AIR Quiet Air 6-25.
- Un transductor de presión Design Instruments TPR-14/N.

Figura 4.1 – Lazo de control de presión

2.1. Identificación del sistema

El primer paso a realizar es la obtención del modelo del sistema. Para ello se utilizará la herramienta de identificación de sistemas de primer orden con retardo que integra UNICOS. La aplicación utiliza un método de identificación en lazo abierto estándar, en el que aplicando un escalón en la entrada y realizando un análisis de la salida obtenida es capaz de calcular los parámetros del modelo de primer orden con retardo del sistema, asumiendo siempre que se debe partir de un estado estacionario del sistema y que, al final del experimento, se alcanzará otro.

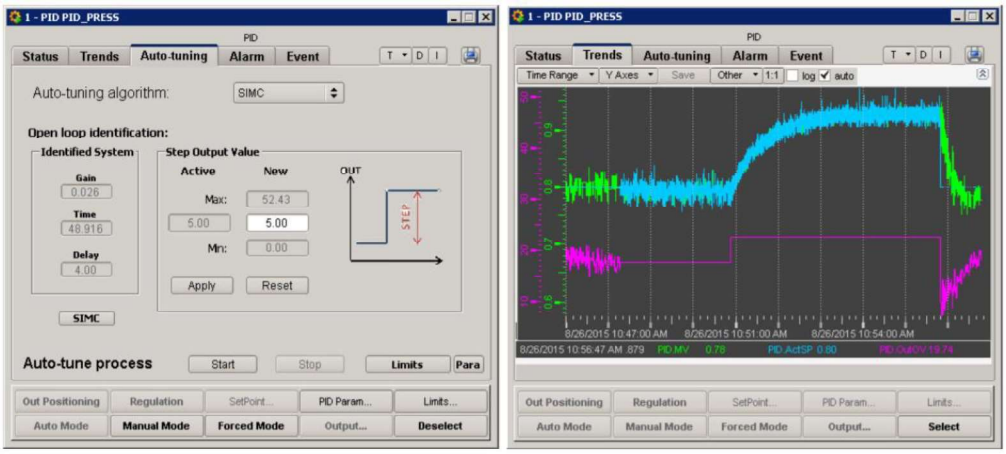


Figura 4.2 – Identificación del sistema

El resultado de la identificación que se obtiene es el siguiente:

$$y_0 = 0.8\,Bar \;\; u_0 = 20Hz \;\; \Delta u = 5Hz$$

$$\frac{\Delta y}{\Delta u} = \frac{0.026}{49.916s + 1} e^{-4s} \qquad [4.1]$$

Para comprobar los resultados obtenidos con el PFC se van a comparar los resultados que se obtengan con los de un PID común. Para obtener los parámetros de sintonía del PID en cuestión también se recurrirá a la herramienta de autosintonía que incluye UNICOS. Sin entrar en más detalles sobre la utilización de dicha herramienta, simplemente es necesario indicar que se ha seleccionado una configuración PI para el controlador, de tal modo que la dinámica final de la respuesta del sistema sea robusta, sin sobrepaso.

La configuración resultante, como se muestra en la Figura 4.3, es $P = 84.4$ y $T_i = 52.52$.

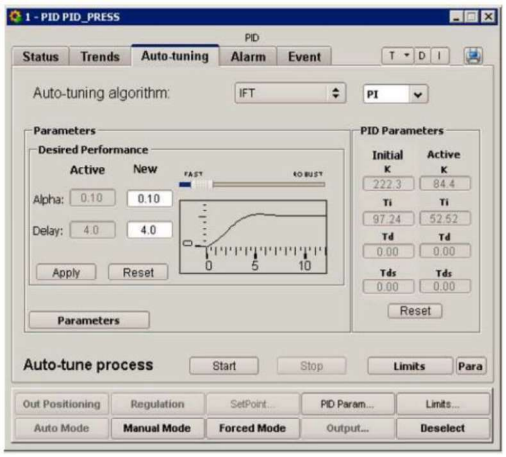


Figura 4.3 – Obtención de los parámetros del PID

2.2. Validación del controlador en EcosimPro

El algoritmo de control del PFC se ha encapsulado dentro de un bloque cuyas entradas son el valor *set point* y la medida de la variable controlada del proceso y la salida la acción de control a aplicar sobre la variable manipulada. Como datos de configuración del componente aparecen el parámetro del filtro aplicado a la estimación de la perturbación α y los valores λ y N que determinan el punto de coincidencia del sistema con la referencia.

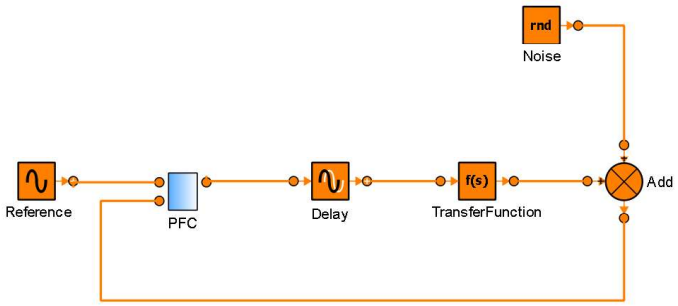


Figura 4.4 – Lazo de control en EcosimPro

2.2.1. Sintonía del controlador

El primer paso es determinar el tiempo de muestreo. En la Figura 4.2 se realizó un ensayo en lazo abierto sobre el sistema, donde se ve que el tiempo de asentamiento es de aproximadamente 2 minutos. En base a esto, se puede determinar que el periodo del controlador deberá estar en torno a los $4s$ y la acción del sistema es directa.

$$T_s = \frac{t_s}{S} = \frac{120s}{30} = 4s \qquad [4.2]$$

En primer lugar, se han definido unos parámetros de sintonía para el PID y para el PFC que generaran la misma respuesta en el sistema si éste no tuviera retardo y considerando que no hay errores de modelado ni ruido en las señales, para así poder comparar después los resultados que se obtienen en distintos escenarios con este resultado inicial. Para ello se ha reducido la ganancia del PID a $K_p = 40$ y se ha mantenido el tiempo integral $T_i = 50$ y se ha sintonizado el PFC, para que también tenga una respuesta robusta, con unos parámetros $\lambda = 0.8$ y $N = 30$. Es importante señalar que el parámetro N seleccionado debe ser un punto en el que el sistema se haya podido estabilizar, por lo que se debe haber tenido en consideración el tiempo de asentamiento del sistema.

Para verificar que se cumple que las respuestas de ambos controladores son, si no iguales, muy similares partiendo de esa sintonía inicial se realizará un ensayo en el que se aplica, partiendo de un estado inicial $y_0 = 0.8\,Bar$ estable, un incremento de 0.1Bar en el *set point* en $t = 500s$.

La Figura 4.5 muestra los resultados obtenidos con la configuración indicada. Las gráfica 1 muestra que, tanto la salida del sistema que se genera con el PFC (y_PFC) como la que se obtiene con el PID (y_PID) alcanzan la referencia (w) en unos 200s. Ambas salidas tienen prácticamente la misma dinámica, siendo ligeramente más rápida la controlada por el PFC. Además, como no hay ruido en el sistema ni errores de modelado, la salida del modelo (y_Model) y la del proceso (y_PFC) se solapan completamente. La gráfica 2 muestran las acciones de control o salidas de los controladores. La acción de control generada por el PFC (u_PFC) es algo más agresiva que la obtenida del PID (u_PID).

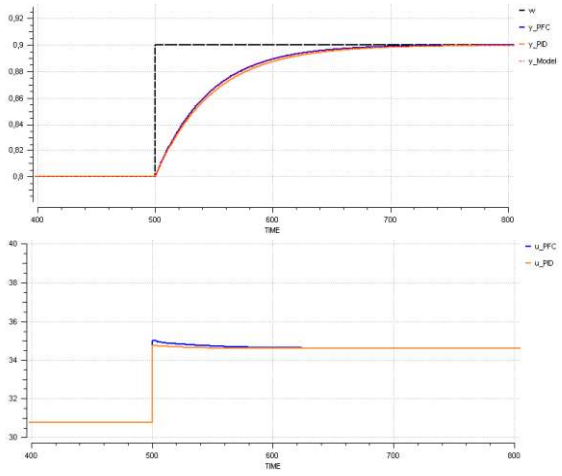


Figura 4.5 – Comparativa PFC y PID: Caso ideal sin retardo en el sistema

Una vez que se ha verificado que, con esos parámetros de sintonía ambos controladores parten de un estado inicial en el que presentaban un comportamiento similar, se han introducido el retardo del sistema y un ruido blanco en la medida.

El experimento que se realiza parte, igual que en el caso anterior, de un estado inicial estable y se aplica un incremento de 0.1Bar en el *set point* en $t = 500s$.

En la gráfica 1 de la Figura 4.6 se puede ver que las tres salidas se superponen y alcanzan el estacionario al mismo tiempo. La gráfica 3 muestra la salida de ambos controladores. Mientras que en el caso anterior, Figura 4.5, el valor máximo lo alcanzaba la salida del PFC (u_PFC), en este caso el efecto del retardo ha hecho que el valor máximo de la salida del PID (u_PID) aumente. Como el retardo en el sistema es muy pequeño, el funcionamiento del PID es tan adecuado como el del PFC en este caso. También se puede apreciar que como el valor de la perturbación medible estimada no está filtrado (Gráfica 2) la acción de control del PFC varía más que la del PID.

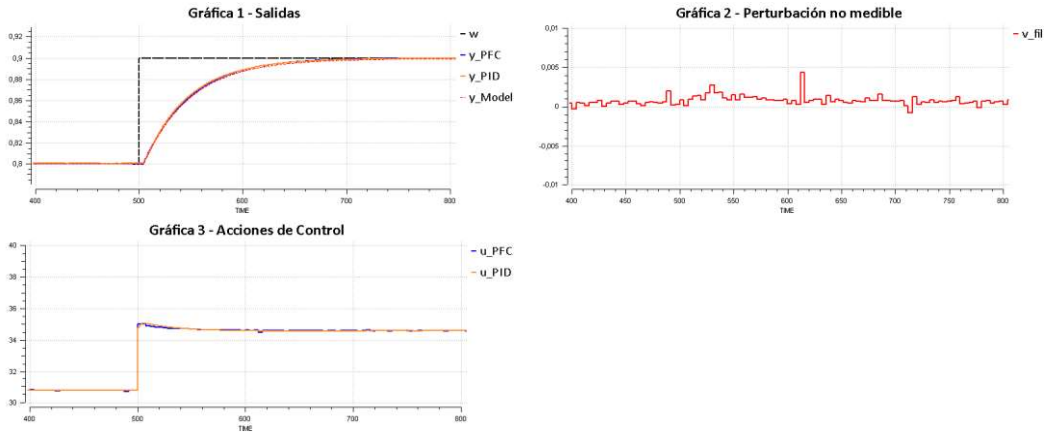


Figura 4.6 –Comparativa PFC y PID: Modelo del sistema con ruido y retardo

El siguiente paso será modificar los parámetros del controlador para obtener una respuesta más agresiva o más robusta. Para lograr una dinámica más rápida en el sistema, el punto de coincidencia fijado deberá estar más cerca, utilizando un N menor, y el parámetro λ deberá ser más cercano a 0. Si por el contrario se pretende obtener una respuesta más robusta el punto de coincidencia deberá estar más alejado y la trayectoria de referencia determinada por λ será más cercana a 1.

A continuación se han realizado dos test con las mismas características que el caso anterior, modificando solamente los parámetros N y λ del PFC. En el primero, se han fijado $\lambda = 0.5$ y $\mathrm{N} = 10$ y en el segundo $\lambda = 0.95$ y $\mathrm{N} = 35$.

La Figura 4.7 muestra en la gráfica 2 que la acción de control generada por el PFC (u_PFC) es ahora más agresiva que la del PID (u_PID), lo que implica que en la gráfica 1 la dinámica de la salida obtenida con el PFC (y_PFC) más rápida que la del PID (y_PID).

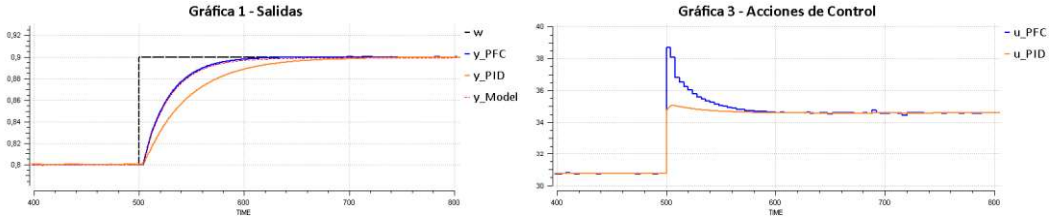


Figura 4.7 – Comparativa PFC y PID: $\lambda = 0.5$ y $N = 10$

En la Figura 4.8 se observa el efecto contrario. La gráfica 1 muestra una respuesta de la variable controlada por el PFC (y_PFC) más robusta debido a que la acción de control (u_PFC) es menos agresiva.

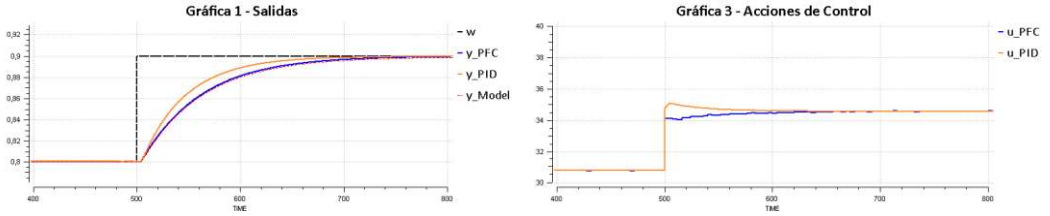


Figura 4.8 – Comparativa PFC y PID: $\lambda = 0.95$ y $N = 30$

El PFC también es robusto ante errores de modelado. Para la realización de los siguientes tests, se han definido las funciones de transferencia que se muestran en la ecuación [4.3] para el comportamiento del proceso y el del modelo. Dichas funciones contienen errores en los tres parámetros, la ganancia, la constante de tiempo y el retardo.

$$G_p = \frac{0.026}{49.916s+1}e^{-4s} \qquad G_m = \frac{0.0255}{50s+1}e^{-5s} \qquad [4.3]$$

Los parámetros de sintonía del controlador PFC serán $\lambda = 0.8$ y $N = 30$ y se realizará un cambio en la referencia en las mismas condiciones que en los ensayos anteriores.

La gráfica 2 de la Figura 4.9 muestra que la perturbación no medible absorbe la diferencia entre la salida del modelo (y_Model) y la del proceso (y_PFC) que aparecen en la gráfica 1. La salida del sistema (y_PFC) alcanza el estacionario con la misma dinámica que en la Figura 4.6 y generando una acción de control (u_PFC) también igual que en los casos en los que no había error en el modelo.

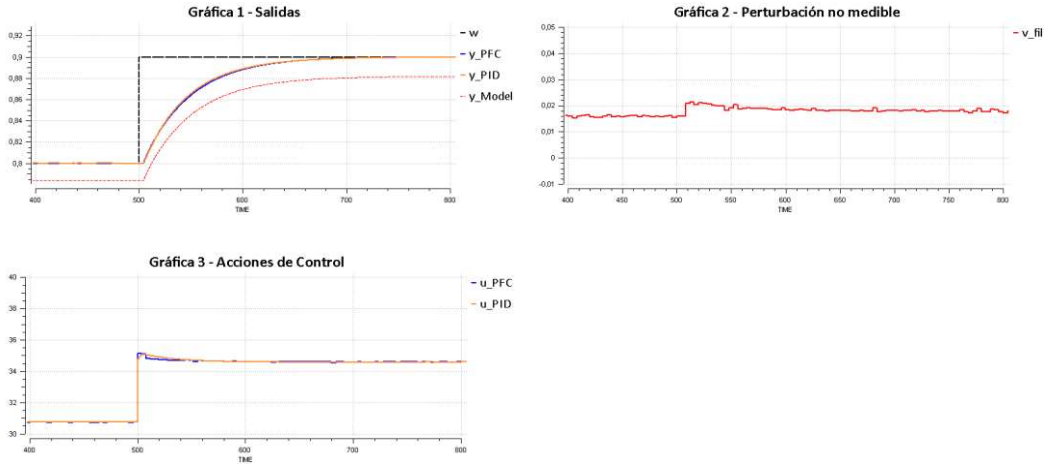


Figura 4.9 – Comparativa PFC y PID: $\tau = 50$ y $k = 0.0255$

2.2.2. Validación del controlador sobre sistemas con grandes retardo

El sistema que se ha seleccionado no es el más adecuado considerando que una de las mayores fortalezas de este método se hace patente en sistemas con grandes retardos.

A continuación, de modificará el modelo del proceso a uno con un retardo mayor con $d = 40s$, manteniendo los parámetros de sintonía del controlador en $\lambda = 0.8$ y $\mathrm{N} = 30$.

La Figura 4.10 muestra en la gráfica 2 que la acción de control generada por el PID (u_PID) es mucho más agresiva que la del PFC (u_PFC), que mantiene su valor máximo en torno a los 35Hz que se obtenían en los casos con un retardo menor. Las salidas del sistema que se muestran en la gráfica 1 hacen ver que si el retardo fuera mayor, la salida controlada por el PID (y_PID) se acabaría volviendo inestable, mientras que sobre la regulada por el PFC no se observa esta problemática.

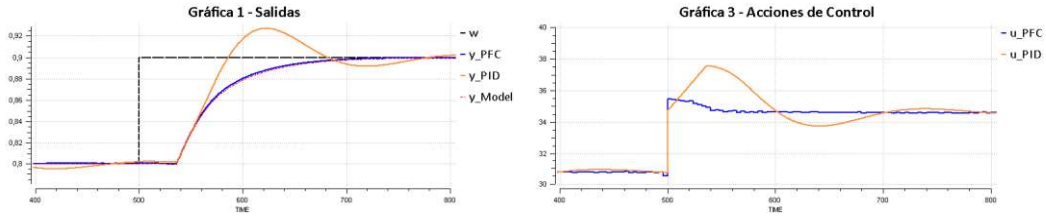


Figura 4.10 – Ensayo PFC: $d = 30s,\ \lambda = 0.80$ y $N = 30$

2.3. Validación del controlador en una planta real

La implementación del algoritmo del PFC se ha testeado directamente sobre el sistema del compresor, utilizando como modelo la identificación obtenida en el apartado anterior [4.1] y el tiempo de muestreo definido en la ecuación [4.2].

La validación del controlador se realizará aplicando una serie de cambios en la referencia del sistema y verificando que la salida es capaz de seguirla con la dinámica apropiada.

Se realizarán dos ensayos consecutivos con diferentes sintonías en el controlador, en los que se aplica el mismo incremento sobre la referencia del sistema para comparar los resultados. En el primer ensayo los valores de los parámetros del PFC serán $\lambda = 0.60$ y $N = 20$ y en el segundo $\lambda = 0.20$ y $N = 10$.

La Figura 4.11 muestra los resultados obtenidos de la prueba realizada. En azul aparece la referencia, en verde la salida del sistema (presión [Bar]) y en magenta la acción de control (frecuencia [Hz]).

En los dos primeros experimentos realizados se ha aplicado una variación en la referencia de 0.2Bar, primero negativa y después positiva. En ambos casos se puede ver que el sistema alcanza el *set point* requerido con una dinámica lenta. La acción de control que se aplica para ello alcanza un valor máximo $\Delta u = 20Hz$ aproximadamente respecto del valor inicial.

A continuación se han aplicado dos nuevos cambios en la referencia iguales a los anteriores, ahora habiendo modificado los parámetros de sintonía del controlador para que tenga un comportamiento más agresivo.

Figura 4.11 – Ensayo sobre planta real

Se puede ver en este caso que, efectivamente, la acción de control generada es mucho más agresiva ante el mismo cambio en el *set point,* llegando incluso a saturar el actuador, lo que implica un cambio máximo sobre el valor inicial de $\Delta u = 30Hz$, aunque podría haber sido mayor si no hubiera saturado. Por otra parte, se puede apreciar que la respuesta del sistema presenta un sobrepico. Esto es debido a que el valor N que fija la posición x del punto de coincidencia que se ha seleccionado no era suficientemente grande para asegurar que el sistema estaba en estacionario.

3. *Dynamic Matrix Control*

El controlador DMC ha sido implementado tanto en Matlab, como sobre el PLC. La verificación del funcionamiento del controlador en Matlab se realizará comparando los resultados obtenidos en dicho entorno con los resultados de HITO, un controlador predictivo comercial que contiene, entre otros muchos, el algoritmo DMC.

Una vez validada la implementación de Matlab, se comprobará que en el PLC se consiguen los mismos resultados.

3.1. Herramienta Integrada para Total Optimización (HITO)

HITO es un paquete de software comercial orientado a la industria de procesos para la implementación de un control jerárquico desarrollado por el departamento de Ingeniería de Sistemas y Automática de la Universidad de Valladolid (Prada, Cristea, Zamarreño, & Álvarez, 1999).

Esta aplicación utiliza técnicas de control que permiten el control de sistemas multivariables con restricciones, así como la de optimización de consignas aplicando los conceptos del control predictivo a la optimización económica de procesos. HITO ofrece un variado repertorio de algoritmos de control predictivo lineal y no-lineal que permiten seleccionar la técnica más adecuada para el tipo de proceso y los objetivos de control de cada aplicación.

Además de las herramientas de control, HITO incluye un amplio rango de herramientas de identificación de procesos, simulación, sistemas expertos en supervisión y diagnóstico.

3.2. Validación del controlador en Matlab

El sistema elegido para hacer las pruebas pertinentes es un reactor químico exotérmico, en el que tiene lugar la reacción de conversión de un producto A en otro B. Las variables manipuladas o entradas del proceso son:

- Caudal de refrigerante: $F_r[m^3/h]$
- Caudal de entrada del producto A: $F_l[m^3/h]$

Las variables controladas o salidas son:

- Concentración de salida del producto A: $C_a[Kg-mol/m^3]$
- Concentración de salida del producto B: $C_a[Kg-mol/m^3]$
- Temperatura en el reactor: $T_l[^{\circ}C]$

Y las perturbaciones medibles son:

- Concentración de entrada del producto A: $C_a[Kg-mol/m^3]$
- Temperatura del flujo de entrada: $T_{l_0}[^{\circ}C]$
- Temperatura de entrada del refrigerante: $T_{r_0}[^{\circ}C]$

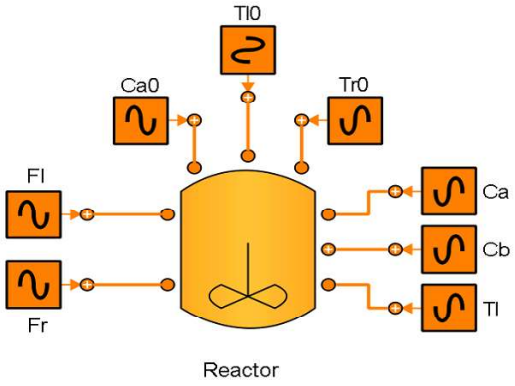


Figura 4.12 – Sistema del reactor

Se trata, por tanto, de un sistema multivariable con pocas entradas y salidas, en el que el grado de interacción entre las variables de este proceso es demasiado alto como para plantear lazos de control independientes entre las entradas y las salidas, a lo que además hay que añadir el efecto de las diferentes perturbaciones del sistema, de modo que es un caso apropiado de estudio para el tipo de problemática que se quiere abordar.

3.2.1. Identificación del sistema

Considerando que todos los parámetros que definen el proceso son conocidos y se puede definir su comportamiento planteando los balances de materia y energía del sistema, una forma de obtener el modelo es linealizar las ecuaciones diferenciales en torno al punto en que se vaya a trabajar y obtener las funciones de transferencia del sistema. Una vez conocidas, a partir de ellas, se pueden obtener los coeficientes de la respuesta salto.

La Figura 4.13 que se muestra a continuación contiene los coeficientes de la respuesta salto entre las entradas y las salidas del sistema y la Figura 4.14 entre las perturbaciones medibles y las salidas.

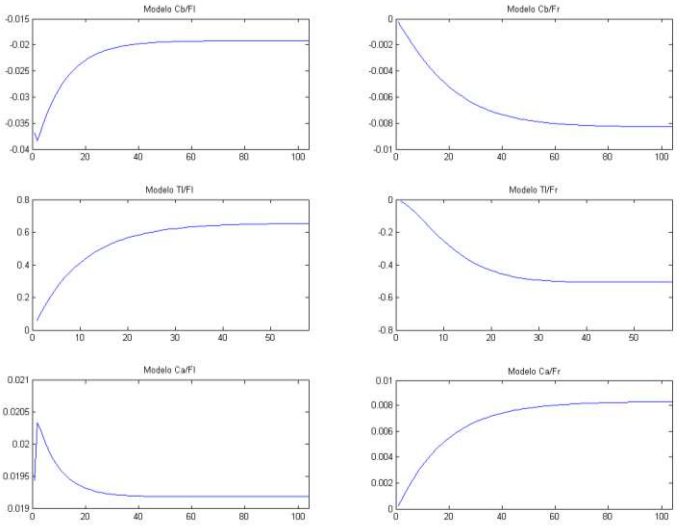


Figura 4.13 – Identificación de los modelos entrada/salida

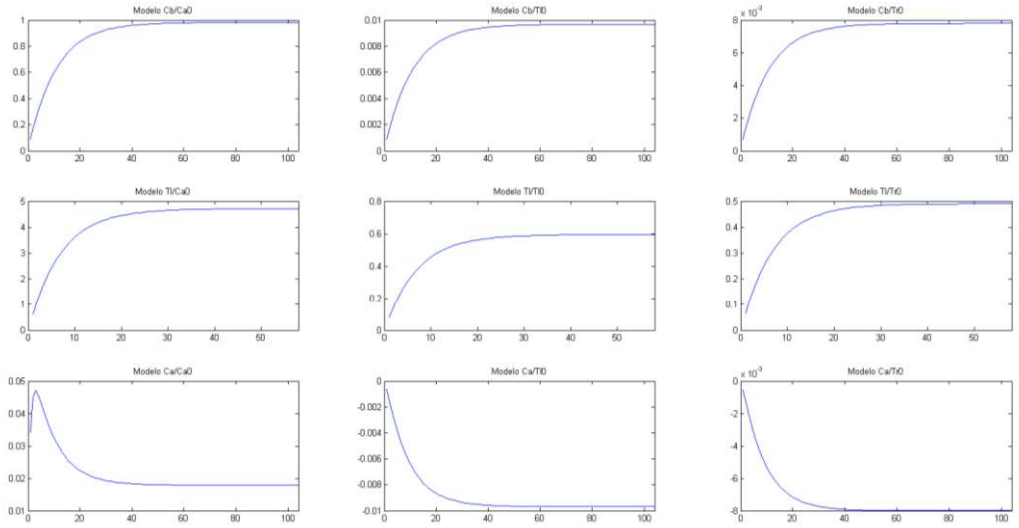


Figura 4.14 – Identificación de los modelos perturbación/salida

Si se analizan las respuestas obtenidas, se puede encontrar que algunos contienen respuestas de fase no mínima, como en el caso del modelo de la Figura 4.13 entre la concentración de reactivo A en la salida el caudal de reactivo de entrada o en la Figura 4.14 entre la concentración de reactivo A en la salida y en la entrada.

El control de sistemas con dinámicas complejas como los ejemplos que se acaban de mencionar es otra de las fortalezas de los controladores MPC.

3.2.2. Sintonía de parámetros

3.2.2.1. Pesos en la función de costes

Se debe asignar un peso al error de cada salida y así como al esfuerzo de control generado por cada una de las entradas.

Dado que el sistema tiene tres variables controladas y sólo dos manipuladas, no es posible configurar las tres salidas para que sigan una referencia. La práctica habitual en estos casos es configurar dos de las salidas para seguimiento de referencia y en la tercera solamente se comprueba que esté dentro de los límites permitidos, definiendo para ella un peso mucho menor. De este modo, la salida que mayor prioridad tiene es la C_b, puesto que es el objetivo del proceso y la C_a sólo deberá mantenerse dentro del rango permitido. De esta forma se obtiene:

$$\gamma = [3, 1, 0]$$

Las acciones de control, en este caso, serán penalizadas por igual, de manera que tengan una cierta importancia en la función de costes, pero no demasiada.

$$\beta = [0.5, 0.5]$$

Por último, el incumplimiento de los límites de la salida del sistema ha sido implementado como una restricción blanda, de manera que hay que determinar el peso que penalizará dicho incumplimiento. El valor debe ser lo suficientemente alto como para que su valor sea significativo en el total del coste.

$$\rho = [1000, 1000, 1000]$$

3.2.2.2. Horizontes del controlador

Los horizontes de predicción, de coincidencia y de control deben fijarse en base al conocimiento que se tiene del proceso.

La Figura 4.13 y la Figura 4.14 muestran los modelos obtenidos para el sistema. En ellas, se puede ver que las salidas 1 (C_b) y 2, (T_l) tienen una respuesta de primer orden, mientras que la salida 3 (C_a) presenta una dinámica de fase no mínima en alguno de los casos, aunque no en todos. De cualquier modo, hay que tener en cuenta que el peso de esta variable en el total del coste va a hacer que su error no sea demasiado significativo. El comienzo del horizonte de coincidencia podría ser, por tanto:

$$N_1 = [1,1,5]$$

Siguiendo un razonamiento similar se puede estimar el final del horizonte de coincidencia.

$$N_2 = [60, 40, 40]$$

El horizonte de control determina el número de acciones de control permitidas en cada optimización para alcanzar la referencia, aunque por otra parte aumentará o reducirá el número de grados de libertad del problema, afectando por tanto al tiempo que tarda el optimizador en hallar la solución. Fijar un $N_u = 2$ suele ser suficiente para lograr que las acciones de control generadas ante un cambio en la referencia no sean demasiado agresivas.

$$N_u = [2, 2]$$

3.2.2.3. Límites de las variables

Los límites en las variables controladas y manipuladas vienen dados por los elementos físicos (sensores y actuadores) y por el propio proceso. En este caso, sus valores son:

$$\Delta u_{Min} = [\,-5.0, -5.0\,]$$
$$\Delta u_{Max} = [\;5.0, 5.0\,]$$
$$u_{Min} = [\,10.0, 8.0\,]$$
$$u_{Max} = [\,56.0, 57.0\,]$$
$$y_{Min} = [\,6.2, 55.0, 0.0\,]$$
$$y_{Max} = [\,8.0, 74.0, 1.5\,]$$

3.2.2.4. Trayectoria de referencia

La dinámica deseada, por el momento, se quiere que sea robusta, de manera que se fija un α cercano a 1.

$$\alpha = [0.9, 0.9, 0.9]$$

3.2.2.5. Número máximo de iteraciones del optimizador

En el caso de que la solución analítica del problema de optimización no verifique las restricciones, el optimizador comenzará con un método iterativo de búsqueda de la solución.

Sobre todo de cara a la implementación en el PLC es necesario determinar un número máximo de ciclos de búsqueda para asegurar los cumplimientos de tiempos que requiere.

3.2.3. Comportamiento del controlador

La validación de la respuesta del controlador se realizará aplicando diferentes cambios en la referencia y en las perturbaciones. En estos casos se analizará el comportamiento del optimizador, el número de iteraciones que necesita para obtener el resultado y si verifica todas las restricciones, y el comportamiento del sistema.

3.2.3.1. Seguimiento de referencia

El primer test se realizará introduciendo solamente cambios en el *set point* de las variables controladas del sistema.

En la Figura 4.15 se puede observar que las salidas del sistema 1 y 2 siguen la referencia indicada, mientras que la 3 sólo verifica el cumplimiento de los límites que se han impuesto. El fuerte acoplamiento que hay entre las variables del sistema hace que un cambio en una de las referencias consiga que las demás salidas se vean afectadas pero, como se puede ver, estas perturbaciones se rechazan rápidamente.

Respecto al cumplimiento de las restricciones, en la Figura 4.15 se puede ver que en el instante 350 el límite inferior de la salida c_B ha sido violado. Esto se debe a que las restricciones impuestas sobre las variables controladas son blandas y solamente suponen un peso añadido a la función de costes. En la Figura 4.16 se observa en cambio que, mientras el valor de la acción de control está dentro de los límites en todo momento, los incrementos rebasan el rango permitido en el instante 450. La razón de este incumplimiento es que el optimizador alcanzó en ese momento el número máximo de iteraciones permitidas, 15. El aumentar este parámetro del resolvedor incrementa las posibilidades de encontrar la solución óptima pero hay que perder de vista que esto afectará al tiempo de ciclo del controlador.

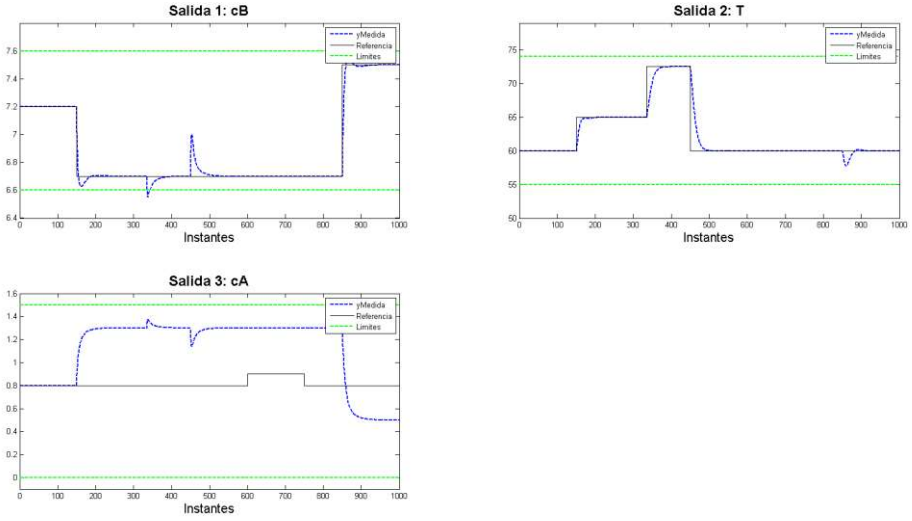


Figura 4.15 – Ensayo DMC. Seguimiento de la referencia (Salidas)

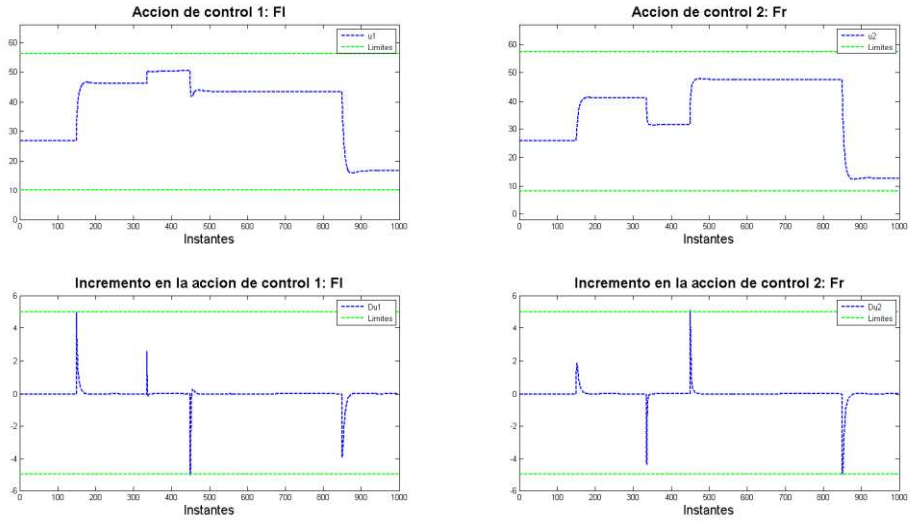


Figura 4.16 – Ensayo DMC. Seguimiento de la referencia (Acciones de control)

En este caso, el optimizador sólo ha recurrido al cálculo iterativo en una ocasión, en el punto en el que, en la Figura 4.16, se puede ver que el incremento en la acción de control superaba el valor permitido. El número máximo de iteraciones observadas en este test ha sido de cuatro.

3.2.3.2. Rechazo de perturbaciones

El siguiente caso muestra el comportamiento del controlador ante la presencia de perturbaciones. Para ello, se han añadido a los cambios en la referencia del apartado anterior una serie de perturbaciones (Figura 4.19) en las diferentes variables existentes. Además, el número de iteraciones máximo del optimizador se ha aumentado a 25.

Como muestra la Figura 4.18, el controlador es capaz de rechazar todas ellas y recuperar de nuevo el valor de referencia sobrepasando ligeramente los límites de las salidas en los instantes 225 y 350. Se puede ver también en la Figura 4.18 que el valor de las acciones de control se ha llevado a los límites en los dos casos, pero respetándolos en todo momento. Al haber incrementado el número de iteraciones del optimizador a 25 el punto 450, donde en el apartado anterior el incremento en las acciones de control superaba el valor permitido, ahora no lo hace, aunque sí se supera el límite en el punto 500, cuando apareció una perturbación sin que el sistema se hubiera recuperado del último cambio en la referencia.

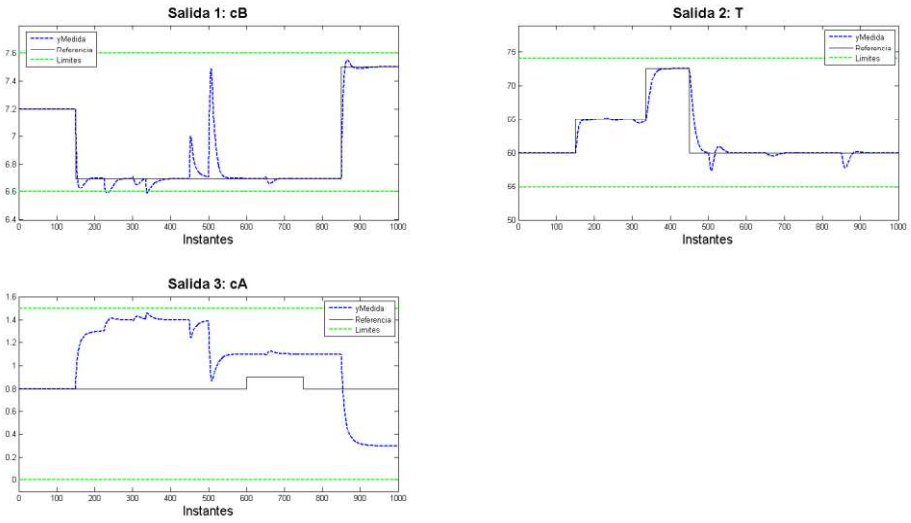


Figura 4.17 – Eayo DMC. Rechazo de perturbaciones (Salidas)

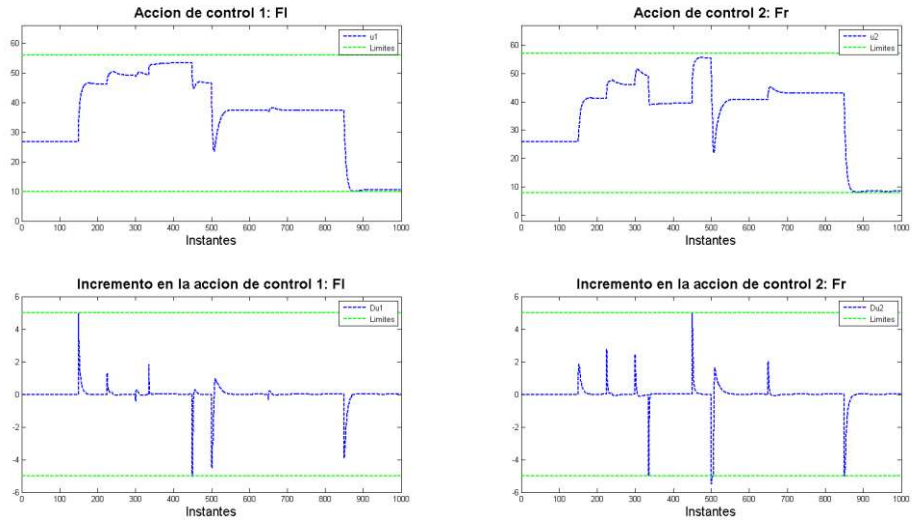


Figura 4.18 – Ensayo DMC. Rechazo de perturbaciones (Acción de control)

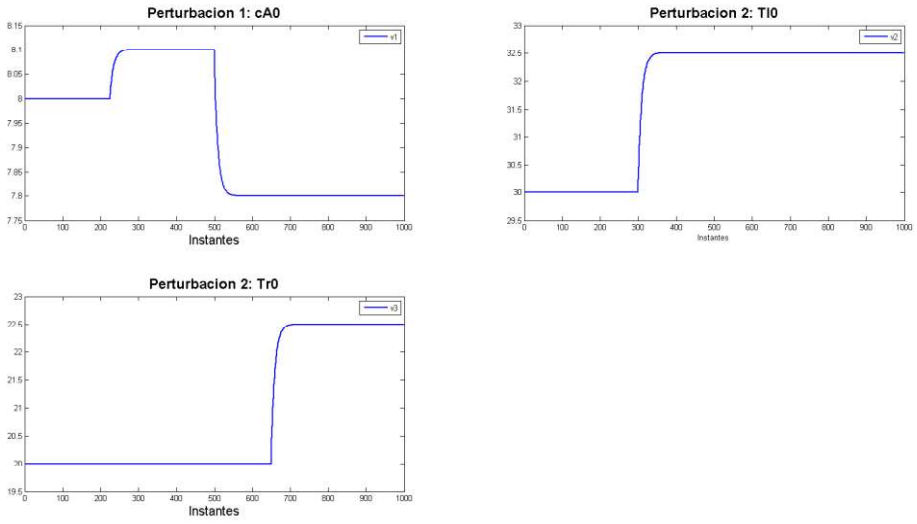


Figura 4.19 – Ensayo DMC. Rechazo de perturbaciones (Perturbaciones)

Analizando los resultados del optimizador, sólo ha necesitado en dos casos recurrir al cálculo numérico, con un máximo de 6 iteraciones.

3.3. Comparación de los resultados obtenidos con HITO

El último paso de la validación consiste en realizar el mismo experimento con una simulación en EcosimPro conectada al controlador HITO. Para ello, se modelado el proceso en base a las ecuaciones físicas que lo caracterizan y se ha sintonizado el controlador con los mismos parámetros que en el modelo de Matlab.

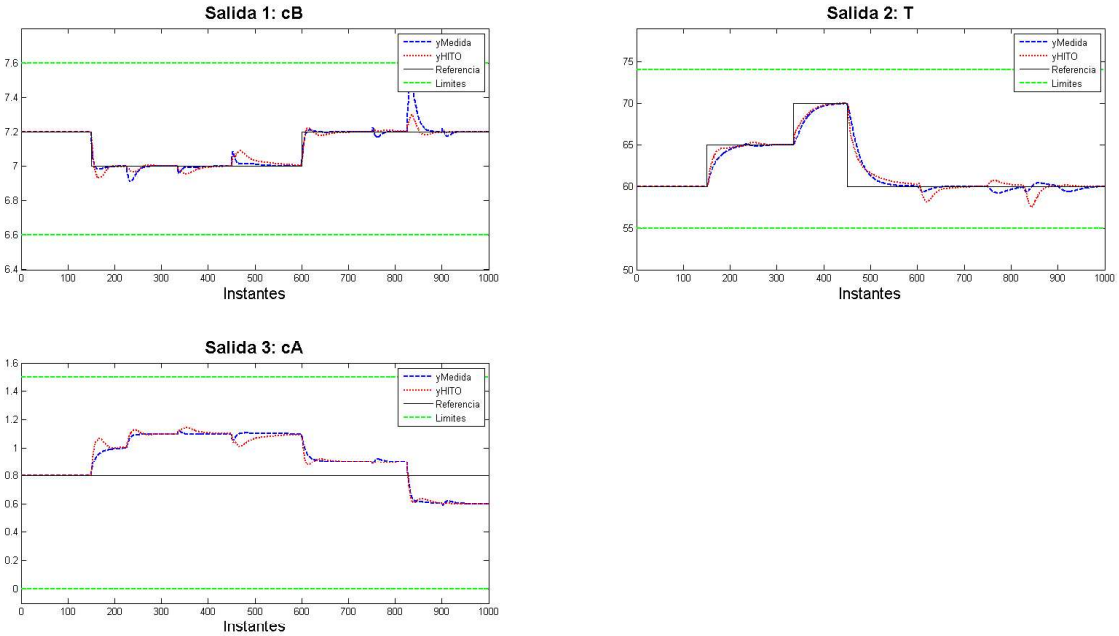


Figura 4.20 – Comparativa entre controlador en Matlab e HITO (Salidas)

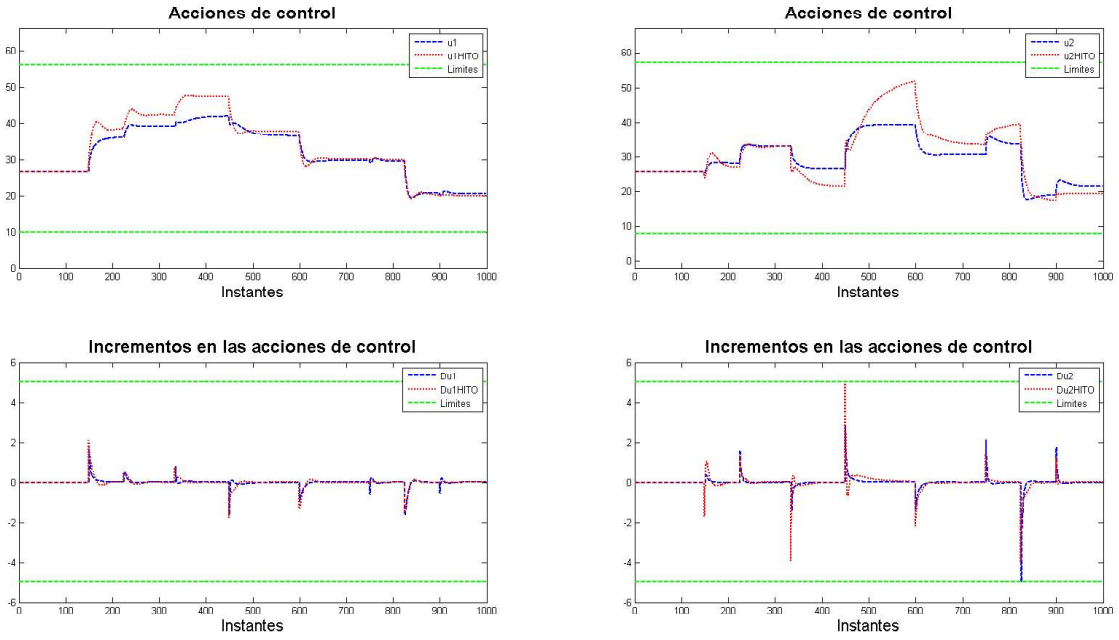


Figura 4.21 – Comparativa entre controlador en Matlab e HITO (Acciones de control)

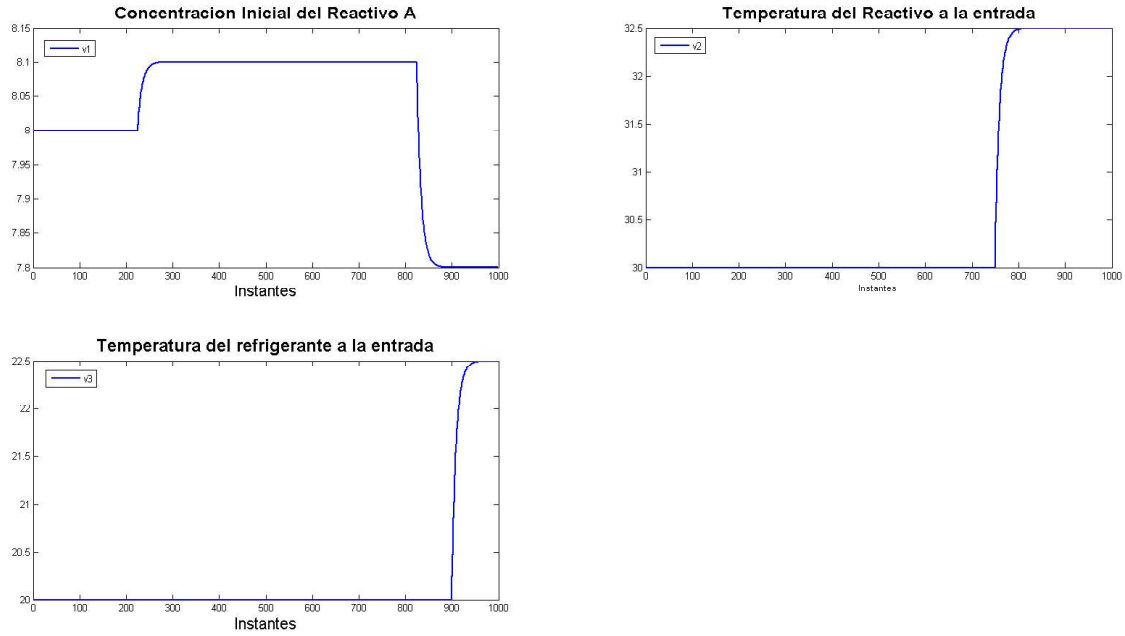


Figura 4.22 – Comparativa entre controlador en Matlab e HITO (Perturbaciones)

La Figura 4.20 muestra la comparativa entre las salidas obtenidas en Matlab y en HITO. En ella se puede ver que, en términos generales, los resultados obtenidos en las variables controladas del sistema son los mismos.

En la Figura 4.21 se pueden ver las acciones de control aplicadas en cada caso. En primer lugar se observa que los valores de los incrementos aplicados son más altos, en la mayoría de los casos, en el resultado obtenido en HITO. Esto puede ser debido a que en Matlab, los pesos β y γ se normalizan teniendo en cuenta el valor inicial de las variables y es posible que HITO aplique además otros criterios.

Teniendo en cuenta los resultados obtenidos en la comparativa, se puede considerar adecuado el funcionamiento de la implementación realizada en Matlab.

3.4. Validación del controlador en el PLC

Las pruebas del funcionamiento del DMC en un PLC se han realizado sobre una CPU (*Central Processing Unit*) de Siemens, modelo 315-2 PN/DP. Se ha seleccionado este equipo por ser uno de los más pequeños, en términos de memoria y capacidad de procesamiento, de Siemens y el menor de los utilizados en el CERN. De este modo, si se valida su funcionamiento en este terminal, se asegura el buen funcionamiento en modelos de gama superior.

Las pruebas que se quieren realizar en esta validación consisten en ejecutar el algoritmo del controlador en el PLC tomando como valores de entrada una serie de puntos de trabajo del sistema obtenidos de la simulación en Matlab.

A continuación, se evaluarán los siguientes aspectos:

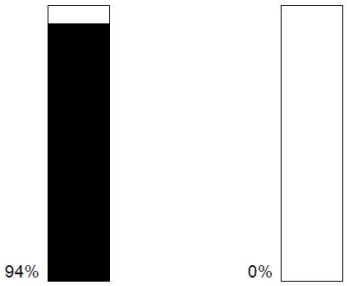
- Dimensiones reales del controlador en la memoria del PLC.
- Validación de la función de inversión de matrices
- Comprobación del cálculo de la respuesta libre
- Comprobación los parámetros de entrada al optimizador
- Validación del funcionamiento del optimizador
  - Número de iteraciones requeridas para converger
  - Solución obtenida

3.4.1. Ocupación de la memoria

Como se mencionó en ocasiones anteriores, el tamaño máximo de las matrices del controlador debe ser definido de antemano. En el Capítulo 3 se calculó que el tamaño que requiere un lazo de control de tres entradas, tres salidas y tres perturbaciones medibles es de 313 KB, con unos horizontes de predicción y de control máximos $N_2 = 30$ y $N_u = 5$. La CPU 315-2 PN/DP tiene un espacio de memoria de trabajo de 384KB y de memoria retentiva de 128KB, haciendo un total de 412KB. Al espacio requerido por las matrices hay que añadir el ocupado por el código, que supone unos 55KB. Teniendo en cuenta todo lo anterior se puede afirmar que el modelo de CPU seleccionado tiene memoria suficiente para el controlador DMC en las condiciones que se han especificado.

Una vez descargado el código en el autómata, como se puede observar en la Figura 4.23, la cantidad de memoria consumida es de 393KB, una cantidad algo superior a los 368KB que se habían calculado pero bastante cercana, de manera que se puede decir que se verifican los cálculos estimados.

El problema que se plantea en el reactor tiene unas dimensiones inferiores a las máximas definidas en este caso, de manera que podría ser implementado en el modelo de CPU elegido. Ajustar las dimensiones del problema al caso real, dos entradas, tres salidas y tres perturbaciones, reduciría unos 50KB el tamaño del problema, lo que sumado al espacio libre actual, sería más que suficiente para incluir la estructura de objetos y las comunicaciones con la capa superior de control de UNICOS.



	Memoria de trabajo	Memoria remanente, Datos
Libres:	25,504	131,072
Ocup.:	367,712	0
Total:	393,216	131,072
Mayor bloque libre:	- - -	- - -
Total enchufables:	- - -	- - -

Figura 4.23 – Ocupación del DMC en la memoria de una CPU 315-2 PN/DP

3.4.2. Tiempo de ciclo

El tiempo que tarda el controlador DMC en obtener una solución viene determinado en gran medida por el tiempo que tarda el optimizador en resolver el problema QP con restricciones.

El método de Hildreth que se ha implementado obtiene la solución del problema en dos pasos. En primer lugar, calcula la solución analítica del problema sin restricciones. Si dicha solución verifica las restricciones impuestas, no realiza ninguna iteración en el optimizador. En el caso de que las restricciones no se cumplan, el optimizador realizará el número de iteraciones necesarias para converger a la solución óptima o hasta alcanzar el número máximo de iteraciones permitidas. Por tanto, el tiempo mínimo entre llamadas al controlador vendrá determinado por el número de iteraciones máximo que se defina.

El objetivo de los test que se van a realizar es determinar el tiempo que necesita la CPU para cada ciclo del controlador que será, en el mejor de los casos, cuando la solución analítica al problema de optimización verifica las restricciones, y en el más desfavorable, cuando el optimizador alcanza el número máximo de iteraciones en la búsqueda de la solución.

En un modo normal de funcionamiento, el controlador se llama de forma cíclica desde un bloque de organización de interrupción en un PLC, pero para poder registrar los tiempos de ejecución del algoritmo, máximos y mínimos, el controlador se ejecutará en primer lugar en bloque de organización principal del autómata, que se ejecuta continuamente y se puede monitorizar su duración.

Para la obtención de estos resultados se han seleccionado dos puntos de trabajo en Matlab de manera que, en el primero la solución óptima se obtiene de forma analítica y, en el segundo, se requiere de 18 iteraciones para alcanzarlo. El número máximo de iteraciones se ha fijado en 15 para que salga de la rutina del optimizador cuando se haya alcanzado el número máximo.

Los parámetros de sintonía del controlador que afectan al tiempo de ciclo son aquellos que determinan en el tamaño del problema. Si se considera un número fijo de entradas, salidas y perturbaciones, la dimensión del problema de optimización que se debe resolver quedará establecida por los horizontes de predicción y control. Para realizar las pruebas se han fijado $N_{2_i} = 30$ y $N_{u_j} = 2$.

La Figura 4.24 muestra, en primer lugar, el tiempo de ciclo del controlador en un punto de trabajo en el que no se requieren iteraciones del optimizador, obteniendo un resultado que oscila alrededor de los 60ms.

También en la Figura 4.24 en la imagen de la derecha, se muestra el tiempo de ciclo del controlador en un punto de trabajo en el que la optimización termina tras 15 iteraciones por haber superado el número máximo permitido. En este caso el tiempo de ciclo del PLC muestra un resultado de aproximadamente 5500ms.

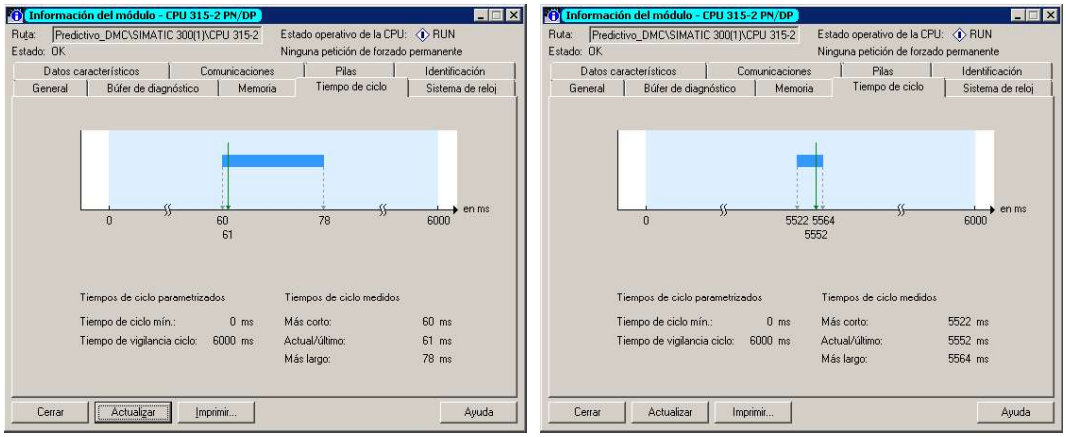


Figura 4.24 – Tiempo mínimo y máximo de ejecución del controlador

Los resultados obtenidos son, por tanto, 60ms en el caso de la solución analítica y unos 5.5s en el de 15 iteraciones, de modo que se puede estimar que cada iteración que realiza el optimizador supone un aumento de 360ms en el ciclo de ejecución del controlador DMC. Si, por ejemplo, se fijara un número máximo de iteraciones igual a 100, el tiempo de ciclo del controlador no podría ser inferior a 36 segundos.

Aumentar el número máximo de iteraciones del algoritmo de Hildreth incrementa las posibilidades de obtener la solución óptima del problema, pero el usuario debe tener en consideración a la hora de configurar este parámetro cómo afecta al tiempo de ciclo del controlador

En el caso de estudio que se ha elegido, el tiempo de ciclo requerido para el control del reactor es de 60s, de manera que el uso del DMC sería adecuado.

3.4.3. Comparación de los resultados obtenidos con el PLC y Matlab

Una vez verificado que el algoritmo es implementable en el PLC, el último paso es comprobar que los resultados numéricos son los mismos que los que se obtenidos en Matlab. Este es el último punto crítico que se debe comprobar, puesto que Matlab trabaja con datos reales de 64 bits cuando lo considera necesario, mientras que en el entorno de programación de SIMATIC hay un máximo de 32 bits por dato, lo que puede afectar a la velocidad de convergencia del optimizador.

La forma de realizar esta validación será introduciendo como datos de entrada al controlador los datos que se han utilizado en un experimento en Matlab de referencias y perturbaciones, así como la medida de las entradas y salidas del proceso en dichos puntos. Esta prueba tiene como finalidad verificar que el valor de la salida predicha, los incrementos en las acciones de control y el número de iteraciones en que se obtienen coinciden con los resultados de Matlab.

Partiendo del siguiente estado inicial del sistema:

Salidas			**Entradas**		**Perturbaciones**		
c_B	T_l	c_A	F_l	F_r	c_{A_0}	T_{l_0}	T_{r_0}
7.2	60.0	0.8	26.67	25.82	8.0	30.0	20.0

Tabla 10 – Condiciones iniciales del sistema

Los parámetros de sintonía del controlador serán los que se definieron en el apartado 3.2.2 con la excepción del horizonte de predicción que se ha modificado a $N_{2_j} = 30$.

En primer lugar, se aplicarán cambios en los valores de referencia de cada una de las entradas, una por una.

- Referencia $c_A = 7.0$
- Referencia $T_l = 70.0$
- Referencia $c_A = 0.7$

Los resultados obtenidos de la realización de estos test se muestran en las tres siguientes tablas. En primer lugar se ha aplicado un cambio $\Delta c_A = 1.0$ en la referencia de la primera salida. En la Tabla 11 se puede ver que tanto en Matlab como en el PLC la solución del problema de optimización se ha obtenido por el cálculo analítico, sin iterar. Los valores obtenidos en los incrementos en la acción de control son iguales hasta el segundo decimal, debido a que Matlab trabaja con datos de 64 bits y en Step7 se consideran tipos de dato de, como máximo, 32 bits.

Cambio en la Referencia c_B				
	Nº Iter.	T. de Ciclo	ΔF_l	ΔF_r
Matlab	0		0.653238484740019	0.887424349403297
PLC	0	60 ms	0.6558963	0.8814850

Tabla 11 – Comparativa de resultados obtenidos en el PLC y Matlab ante un cambio en la referencia c_B

En el siguiente ensayo, considerando de nuevo que se parte del estado inicial definido en la Tabla 10, se ha aplicado un cambio en la referencia de la segunda salida de valor $\Delta T_l = 10.0$. De nuevo, como se puede ver en la Tabla 12, la solución se obtiene sin necesidad de iteraciones, por lo que el tiempo en que se ha ejecutado el ciclo del controlador es el mínimo requerido, 60ms. La solución obtenida en Matlab y en el PLC es muy similar, cambiando a partir del segundo decimal.

Cambio en la Referencia T_l				
	Nº Iter.	T. de Ciclo	ΔF_l	ΔF_r
Matlab	0		4.148905009566276	-4.869987495496719
PLC	0	60 ms	4.151568	-4.875950

Tabla 12 – Comparativa de resultados obtenidos en el PLC y Matlab ante un cambio en la referenciaT_l

Por último se ha aplicado un cambio en el *set point* de la tercera salida, pero como se trata de una variable que no está regulada, sólo limitada puesto que se ha definido un peso 0 a su error en la función de costes, y en el instante inicial ya se verificaba que todas las señales estuvieran dentro del rango permitido, los resultados de Matlab y del PLC son iguales, no aplicando ningún cambio sobre la acción de control.

Cambio en la Referencia c_A				
	Nº Iter.	T. de Ciclo	ΔF_l	ΔF_r
Matlab	0		0.0	0.0
PLC	0	60 ms	0.0	0.0

Tabla 13 – Comparativa de resultados obtenidos en el PLC y Matlab ante un cambio en la referencia c_A

A continuación, se aplicarán cambios en los valores de cada una de las perturbaciones, una por una.

- Perturbaciones $c_{A_0} = 8.05$
- Perturbaciones $T_{l_0} = 31$
- Perturbaciones $T_{r_0} = 21$

En primer lugar, se aplica un incremento sobre la concentración inicial del reactivo $\Delta c_{A_0} = 0.05$. Los resultados que se obtienen en la Tabla 14 muestran que en los dos casos, en Matlab y en el PLC, el optimizador ha alcanzado el número máximo de iteraciones, fijado en 15, sin conseguir verificar completamente el cumplimiento de las restricciones. En ambos casos el Δu obtenido para el caudal de refrigerante es superior a cinco, aunque ya muy próximo a él. Se demuestra, por tanto que, en el caso de no lograr convergencia, el optimizador sale de la rutina con un valor casi-óptimo.

$$\Delta u_{Min} = [-5.0, -5.0]$$
$$\Delta u_{Max} = [5.0, 5.0]$$

Cambio en la Perturbación c_{A_0}				
	Nº Iter.	T. de Ciclo	ΔF_l	ΔF_r
Matlab	15		3.37043624336674	5.03227130713847
PLC	15	5560 ms	3.37162	5.032965

Tabla 14 – Comparativa de resultados obtenidos en el PLC y Matlab ante un cambio en la perturbación c_{A_0}

En el último caso se ha incrementado el valor inicial de la perturbación $\Delta T_{l_0} = 1.0$. En esta ocasión tanto el PLC como Matlab consiguen verificar el cumplimiento de las restricciones tras 5 iteraciones del optimizador (Tabla 15). El tiempo que ha tardado el controlador en obtener una nueva acción de control ha sido de aproximadamente, 2s, lo que supone un tiempo consumido en cada iteración del optimizador de 400ms.

Cambio en la Perturbación T_{l_0}				
	Nº Iter.	T. de Ciclo	ΔF_l	ΔF_r
Matlab	5		-3.450939031819857	5.000236646484587
PLC	5	1983 ms	-3.448324	5.000195

Tabla 15 – Comparativa de resultados obtenidos en el PLC y Matlab ante un cambio en la perturbación T_{l_o}

Capítulo 5 . CONCLUSIONES Y TRABAJO FUTURO

1. Conclusiones

El control predictivo basado en modelos está ampliamente aceptado dentro del entorno industrial, pero debido a la capacidad limitada en términos de memoria y capacidad de cálculo de los PLCs que se sitúan en la capa más baja de control, su uso no está extendido a este nivel.

En los capítulos anteriores, se han mostrado algunos ejemplos de implementaciones de controladores predictivos en autómatas programables y, a continuación, se discutirán los resultados obtenidos.

El *Predictive Functional Control* es un controlador muy sencillo de implementar y con un coste muy bajo a nivel de recursos en un PLC. La casuística de sistemas que cubre está limitada a aquellos que tienen un modelo de primer orden, con o sin retardo, pero resuelven de manera muy satisfactoria la problemática existente en los sistemas con tiempos muertos altos. Es por ello que se considera interesante incluir este algoritmo como alternativa al PID tradicional en el *framework* de UNICOS, así como en otras librerías de control en PLCs.

El *Dynamic Matrix Control* se trata de un sistema de control mucho más robusto. Los tipos de proceso aplicables a este tipo de controlador son mucho más amplios que en el caso anterior, siendo especialmente recomendable para sistemas multivariable con fuertes acoplamientos entre las distintas entradas y salidas, sistemas con dinámicas de dinámica compleja, fase no mínima o grandes retardos, o siempre que se quiera poder tener un control sobre el valor que pueden tomar las variables del sistema.

En la validación del controlador en el modelo de PLC elegido se han encontrado dos grandes limitaciones que son la memoria y la capacidad de cálculo de los autómatas programables. Las conclusiones que se pueden extraer de las pruebas realizadas sobre la CPU 315-2 PN/DP de Siemens es que este modelo puede ser adecuado para un sistema multivariable de, como máximo, tres entradas y tres salidas, siempre y cuando el sistema no necesite un periodo de muestreo para el controlador inferior a seis segundos ($T_s > 6s$).

El modelo de CPU que se seleccionó se incluye dentro de la gama baja de los PLCs de la gama antigua de Siemens. Esta elección era interesante porque es uno de los modelos más utilizados en la primera capa de control en industria, mientras que la gama de los 400 suele ser utilizada en el nivel superior y, por otro lado, aunque los nuevos 1500 sustituirán progresivamente a estos modelos, en la actualidad su uso sigue estando muy extendido.

La Tabla 16 muestra una comparativa con las características del 315-2 con dos modelos de CPU de la serie nueva a nivel de memoria y de capacidad computacional. Los modelos de 1500 que aparecen se corresponden con CPUs de la gama baja y media de la nueva serie, que son los que reemplazarán a la serie 300.

Propiedades	**315-2 PN/DP**	**1511-1 PN**	**1516-3 PN/DP**
Work memory	512 KByte	1.23 MByte	6.5 Mbyte
CPU processing times			
Bit operations	50 ns	60 ns	10 ns
Word operations	90 ns	72 ns	12 ns
Fixed point arithmetic.	120 ns	96 ns	16 ns
Floating point arithmetic	450 ns	384 ns	64 ns
1 000 binary instructions	50 µs	60 µs	10 µs

Tabla 16 – Características de las CPUs Siemens – Comparativa

El modelo 1511-1 PN sólo presenta ligeras mejoras en términos de capacidad computacional, por lo que un DMC implementado en este dispositivo continuaría siendo aplicable únicamente a procesos lentos. La memoria de datos presenta un aumento mucho más significativo, pero hay que tener en cuenta que un aumento del tamaño del problema implica también un aumento de la carga computacional y se observaría un aumento en el periodo de muestreo mínimo factible en el controlador.

La CPU 1516-3 PN/DP pertenece a un nivel superior de dispositivos de control. Se puede apreciar en la comparativa un aumento significativo en sus cualidades, tanto a nivel de memoria como de procesamiento de datos. En la implementación del DMC el mayor consumo de recursos se encuentra en operaciones con datos en formato de coma flotante o *float*. Teniendo en consideración que el coste computacional de una operación con este tipo de datos en un 315-2 es de 450ns y en un 1516-3 64 ns, se puede estimar el periodo mínimo de un controlador DMC, para un problema con las características descritas en el caso de estudio, implementado en esta CPU se podría reducir a un 20%. De esta forma, resultaría un tiempo de ciclo del controlador de 1200ms, en vez de los 6000ms obtenidos en los tests realizados.

Estos resultados se pueden comparar con un caso de estudio presentado por Siemens (PCS7, 2014) en el que se utiliza un reactor químico con 5 variables controladas, 6 variables manipuladas y 3 perturbaciones. El documento explica que el controlador puede alcanzar un tamaño superior a 850KB, con un tiempo de ciclo inferior a 5s. Extrapolando los resultados obtenidos con la implementación realizada, en términos de almacenamiento de datos un problema de las mismas dimensiones necesitaría de 970KB. El tiempo de ciclo es más difícil de estimar, pero suponiendo que el aumento en tiempo es proporcional al del aumento en las dimensiones del problema, se puede estimar que sería de unos 18s.

Con estos resultados se quiere demostrar que la implementación de algoritmos de control avanzado, como es el caso de los controladores predictivos, en el nivel más bajo de una arquitectura de control es una realidad factible gracias al salto cualitativo que están dando los PLCs en la actualidad.

2. Trabajo futuro

Durante el desarrollo de la herramienta de control predictivo para el entorno de UNICOS se han ido observando ciertas mejoras que podrían ser aplicadas sobre el controlador DMC en el futuro.

Una forma de reducir de forma significativa la dimensión del problema de optimización es analizar una serie de puntos concretos del horizonte de predicción en vez del vector completo. Los puntos seleccionados pueden estar equiespaciados o distribuidos de la manera más conveniente según las necesidades del caso a tratar.

La aplicación de esta técnica implica que las restricciones aplicadas tampoco serían comprobadas en todos los puntos, lo que podría llevar a la violación de los límites en los tramos intermedios.

La implementación de esta mejora consiste en una mera simplificación de la solución aportada, de manera en las que las matrices se reducen consecuentemente con los puntos de evaluación seleccionados, de manera que incluir esta modificación no supone un gran problema.

Otro gran punto a estudiar son otros optimizadores implementables en el PLC, puesto que son uno de los puntos más críticos de un MPC. En la bibliografía analizada se sugerían diferentes algoritmos, como el qpOASES, pero el que mejores resultados obtenía era el QP de Hildreth (Huyck, y otros, 2012).

Por último, una vez desarrollada una versión completamente validada del DMC, el último paso sería incluirla en la librería de objetos de UNICOS.

BIBLIOGRAFÍA

Bordóns, C. (2000). *Control Predictivo: metodología, tecnología y nuevas perspectivas.* Universidad de Sevilla.

Camacho, E., & Bordons, C. (2007). *Model Predictive Control.*

Frutos, L. d. (2015). *Desarrollo de algoritmos de control avanzado para dispositivos programables industriales.*

Hildreth, C. (1957). *A quadratic programming procedure.* Naval Research.

Huyck, B., Ferrau, H., Diehl, M., De Brabanter, J., Van Impe, J., De Moor, B., & Logist, F. (2012). *Towards Online Model Predictive Control on a Programmable Logic Controller: Practical Considerations.*

PCS7, S. (2014). *Configuration of the MPC10x10 for Tennessee Eastman Benchmark Process.*

Prada, C. (1997). *Fundamentos de Control Predictivo de Procesos. Los Manuales de Ingeniería Química, Instrumentación y Control de Procesos.* Universidad de Valladolid.

Prada, C., Cristea, S., Zamarreño, J., & Álvarez, T. (1999). HITO A tool for Hierarchical Control. *7th IEEE Emerging Technologies and Factory Automation.*

Prada, C., Serrano, J., & Vega, P. (1994). A comparative study of GPC and DMC controllers. Oxford University Press.

Prett., D., Ramaker, B., & Cutler, C. (1979). *Dynamic Matrix Control Method.* United States Patend.

Richalet, J. (1992). *Practique de la Commande Predictive.*

Rossiter, J. (2003). *Model Based Predictive Control: a practical approach.* CRC Press.

Wang, L. (2009). *Model Predictive Control System Design and Implementation.* Springer-Verlag.

Zamarreño, J., & Vega, P. (1999). *Neural predictive control. Application to a highly non-linear system.*

ANEXO 1:

CÓDIGO FUENTE DEL PFC: ECOSIMPRO

```
/*-------------------------------------------------------------------------------
 LIBRARY: PFC
 FILE: Compresor_v1
 CREATION DATE: 25/02/2016
-------------------------------------------------------------------------------*/

USE MATH


COMPONENT Compresor_v1 (INTEGER TAM = 500)

   DATA ------------------------------------------------------------------------*/

      REAL Ts = 4
      REAL StepTime = 500

      -- Filtro de la perturbacion
      REAL alpha = 0.0

      -- Parámetros del PFC
      REAL lambda  = 0.8
      REAL N       = 30
      REAL uMax     = 80
      REAL DuMax    = 25

      -- Sistema Real
      REAL k_p      = 0.026
      REAL tau_p    = 48.916
      INTEGER d_p   = 20

      -- Modelo del Sistema
      REAL k_m      = 0.026
      REAL tau_m    = 48.916
      INTEGER d_m   = 20

      -- Parametros PID
      REAL PID_k  = -40
      REAL PID_Ti = 50

   DECLS -----------------------------------------------------------------------*/

      -- Condiciones de contorno
      REAL w

      -- Variables del sistema
      REAL          y_Model
      REAL          y_PFC
      REAL          y_PID
      DISCR REAL    y_ant

      REAL       noiseR[97]
      INTEGER    noiseI[5]
      REAL       noise

      -- Variables del controlador
      DISCR REAL u[TAM]
      DISCR REAL u_Model
```

```
   DISCR REAL u_PFC
   DISCR REAL u_dp_PFC
   DISCR REAL v, v_fil, v_fil_1

   -- Variables del PID
   REAL          u_PID
   DISCR REAL    u_PID_d[TAM]
   DISCR REAL    u_dp_PID

   REAL ui
   REAL vi

   REAL e

   -- LLamada al controlador
   BOOLEAN sample = TRUE

   DISCR REAL aux
   DISCR REAL aux1
   DISCR REAL aux2
   DISCR REAL aux3
   DISCR REAL aux4
   DISCR REAL aux5
   DISCR REAL aux6

   INTEGER i

INIT ----------------------------------------------------------------------*/

   w = 30

   y_Model = 0.8
   y_PFC = 0.8
   y_PID = 0.8
   y_ant = 0.8

   v      = 0
   u_Model = 30.7731
   u_PFC = 30.7731
   u_PID = 30.7731
   u_dp_PID = 30.7731
   ui = 30.7731
   vi = 30.734


   noiseR[1] = 1
   noiseI[1] = 1

   FOR (i=1; i<=TAM; i=i+1)
         u[i]=u_PFC
   END FOR

DISCRETE ------------------------------------------------------------------*/

   WHEN (sample == TRUE) THEN

      -- Nuevos calculos
```

```
                -- Puedo dejar u_PFC porque es la u(t-(D+1)Ts) la última que realmente
he aplicado
            aux6 = (exp(-Ts/tau_m))
            v = (y_PFC - aux6*y_ant - k_m*u[d_m]*(1-aux6)) / (1-aux6)

            v_fil = (alpha*v_fil_1) + ((1 - alpha)*v)

            aux = 0
            FOR (i=1; i<=d_m; i=i+1)
               aux = aux + u[i]*exp((i-1)*Ts/tau_m)
            END FOR

            -- Actualizar vector de acciones de control
               -- PFC: Necesario para los cálculos y para simular el retardo
               -- PID: Necesario para simular el retardo
            FOR ( i=TAM; i>=2; i=i-1 )
               u[i] = u[i-1]
               u_PID_d[i] = u_PID_d[i-1]
            END FOR

            aux1 = (1 - (lambda**(N-d_m))) * (w - y_PFC)
            aux2 = ( (exp(-N*Ts/tau_m)) - 1 ) * y_PFC
            aux3 = ( exp(-N*Ts/tau_m)) * k_m * ((exp(Ts/tau_m))-1)*aux
            aux4 = v_fil * ( 1 - exp(-N*Ts/tau_m) )
            aux5 = k_m * (1 - exp( - ((N-d_m)*Ts) / tau_m ) )

            u[1] = ( aux1 - aux2 - aux3 - aux4 ) / aux5

            IF (abs(u[2]-u[1])>DuMax) THEN
               u[1]=u[2]+DuMax
            END IF

            IF (u[1]>uMax) THEN
               u[1]=uMax
            END IF

            u_PID_d[1] = u_PID

            -- Update
            u_PFC = u[1]
            u_dp_PFC = u[d_p]
            u_Model = u[d_m]
            y_ant = y_PFC

            u_dp_PID = u_PID_d[d_p]

            sample = FALSE
            sample = TRUE AFTER Ts

         END WHEN


      CONTINUOUS ------------------------------------------------------------------*/

         -- Modelo del Sistema
         tau_m * y_Model' + y_Model = k_m*u_Model --+ v
```

```
        -- Ruido
        noise = ran1(noiseI, noiseR)

        -- Proceso Real - Sistema de primer orden con retado - PFC
        tau_p * y_PFC' + y_PFC = k_p*u_dp_PFC + 1*(10**-3)*noise

        -- Proceso Real - Sistema de primer orden con retado - PID
        tau_p * y_PID' + y_PID = k_p*u_dp_PID + 1*(10**-3)*noise


        -- Controlador PID
        e = y_PID - w
        vi' = (PID_k / PID_Ti) * e
        ui = PID_k * e + vi
        u_PID = ui
        --u_PID = ZONE (ui > uMax) uMax
        --        OTHERS           ui

END COMPONENT
```

ANEXO 2:

CÓDIGO FUENTE DEL DMC: MATLAB

```
% ************************************************************************ %
%
%   - N1 y N2
%   - Perturbaciones medibles
%   - Perturbaciones no medibles
%   - Variables de holgura
%       · Du Restricciones duras
%       · u  Restricciones duras
%       · y  Restricciones blandas
%
%   - Respuesta libre calculada de forma recursiva
%
%   - Pesos beta y gamma normalizados.
%
%   - Con alpha para modificar la dinámica de la referencia
%
%   - Reducir carga computacional
%
% ************************************************************************ %

%clear all; close all; clc;
warning off;


% ************************************************************************ %
% ** Modelo en respuesta salto del sistema ******************************** %
% ************************************************************************ %

[ nEnt nPer nSal Ts g d nCoef sis dis y0 u0 v0] = RespuestaSalto( );

% ************************************************************************ %
% ** Simulacion *********************************************************** %
% ************************************************************************ %

% Número de veces que se ejecutará el algoritmo DMC
instantes = 300;

% Estado inicial para la función filter
z = max(length(sis(1,1).num{:,:}),length(sis(1,1).den{:,:}))-1;
z_11 = zeros(1,z);
z = max(length(sis(1,2).num{:,:}),length(sis(1,2).den{:,:}))-1;
z_12 = zeros(1,z);

z = max(length(sis(2,1).num{:,:}),length(sis(2,1).den{:,:}))-1;
z_21 = zeros(1,z);
z = max(length(sis(2,2).num{:,:}),length(sis(2,2).den{:,:}))-1;
z_22 = zeros(1,z);

z = max(length(sis(3,1).num{:,:}),length(sis(3,1).den{:,:}))-1;
z_31 = zeros(1,z);
z = max(length(sis(3,2).num{:,:}),length(sis(3,2).den{:,:}))-1;
z_32 = zeros(1,z);

% Estado inicial para la función filter
z = max(length(dis(1,1).num{:,:}),length(dis(1,1).den{:,:}))-1;
```

```
zd_11 = zeros(1,z);
z = max(length(dis(1,2).num{:,:}),length(dis(1,2).den{:,:}))-1;
zd_12 = zeros(1,z);
z = max(length(dis(1,3).num{:,:}),length(dis(1,3).den{:,:}))-1;
zd_13 = zeros(1,z);

z = max(length(dis(2,1).num{:,:}),length(dis(2,1).den{:,:}))-1;
zd_21 = zeros(1,z);
z = max(length(dis(2,2).num{:,:}),length(dis(2,2).den{:,:}))-1;
zd_22 = zeros(1,z);
z = max(length(dis(2,3).num{:,:}),length(dis(2,3).den{:,:}))-1;
zd_23 = zeros(1,z);

z = max(length(dis(3,1).num{:,:}),length(dis(3,1).den{:,:}))-1;
zd_31 = zeros(1,z);
z = max(length(dis(3,2).num{:,:}),length(dis(3,2).den{:,:}))-1;
zd_32 = zeros(1,z);
z = max(length(dis(3,3).num{:,:}),length(dis(3,3).den{:,:}))-1;
zd_33 = zeros(1,z);

% ********************************************************************** %
% ** Parámetros de configuración del MPC ******************************** %
% ********************************************************************** %

% Horizontes de control - [1, nEnt]
Nu = [2, 2];

% Horizontes de predicción - [1, nSal]
N2_1 = nCoef(1) + max([Nu(1) Nu(2)]);
N2_2 = nCoef(2) + max([Nu(1) Nu(2)]);
N2_3 = nCoef(3) + max([Nu(1) Nu(2)]);

N1 = [1 1 1];           % [1, nSal]
% N1 = [5 5 5];           % [1, nSal]
% N2 = [N2_1 N2_2 N2_3];  % [1, nSal]
N2 = [30 30 30];  % [1, nSal]

% Dinámica de la referencia (1-alpha)*z^-1/(1-alpha*z^-1) - [1, nSal]
alpha = [0.9, 0.9, 0.9];

% Factor de peso asignado a cada accion de control - [1, nEnt]
beta = [0.5 0.5];

% Factor de peso asignado a cada salida del sistema - [1, nSal]
gamma = [3 1 0];

% Factor de peso asignado a la variable de holgura de cada salida - [1, nSal]
rho = [1e4 1e4 1e4];

% Limites de las variables
DuMin = [ -5.0, -5.0 ];  % [1, nEnt]
DuMax = [  5.0,  5.0 ];  % [1, nEnt]

uMin = [ 10.0, 8.0 ];      % [1, nEnt]
uMax = [ 56.0, 57.0 ];     % [1, nEnt]
```

```
yMin = [ 6.6, 55.0, 0.0 ];    % [1, nSal]
yMax = [ 7.6, 74.0, 1.5 ];    % [1, nSal]


v1_old = v0(1);
v2_old = v0(2);
v3_old = v0(3);

v1_new = v0(1);
v2_new = v0(2);
v3_new = v0(3);

% ************************************************************************* %

% Variables para tamaños de matrices
Sum_Nu = 0;
for i=1:nEnt
    Sum_Nu = Sum_Nu + Nu(i);
end

Sum_N2 = 0;
for i=1:nSal
    Sum_N2 = Sum_N2 + N2(i);
end

Sum_N2_N1 = 0;
for i=1:nSal
    Sum_N2_N1 = Sum_N2_N1 + N2(i) - N1(i) + 1;
end

Sum_Coef = 0;
for i=1:nSal
    Sum_Coef = Sum_Coef + nCoef(i);
end


% ************************************************************************* %
% ** Condiciones Iniciales ************************************************ %
% ************************************************************************* %

% Vector de tiempo total
t=(0:1:instantes);

% Vector con el último valor aplicado de la acción de control
fil_u_old = nEnt;
col_u_old = 1;
u_old = u0;

% Vector con el valor de la acción de control que se va a aplicar
fil_u = nEnt;
col_u = 1;
u = u0;

v_u = [];
```

```
% Vector con los incrementos de la acción de control que se va a aplicar
fil_Du = nEnt;
col_Du = 1;
Du = zeros(fil_Du, col_Du);

v_Du = [];

% Vector con los incrementos de la acción de control predichos (para todo Nu)
fil_Du_p = Sum_Nu;
col_Du_p = 1;
Du_p = zeros(fil_Du_p, col_Du_p);

% Vector con el valor medido de las salidas del sistema
fil_ym = 1;
col_ym = nSal;
ym = y0;

ym1vec = y0(1); ym2vec = y0(2); ym3vec = y0(3);

% Vector con el valor predicho de las salidas del sistema
fil_yp = 1;
col_yp = Sum_Nu + nSal;
yp = y0;

v_yp = yp;

% Vector con la medida de las perturbaciones en (t-1)
fil_v_old = nPer;
col_v_old = 1;
v_old = v0;

% Vector con la medida de las perturbaciones en (t)
fil_v = nPer;
col_v = 1;
v = v0;

v_v = [];

% Vector con los incrementos de la perturbación v(t)-v(t-1)
fil_Dv = nPer;
col_Dv = 1;
Dv = zeros(fil_Dv, col_Dv);

v_Dv = [];

% Inicialización del vector que contendrá la referencia (solo para dibujar)
ref = zeros(instantes+1, nSal);

% ********************************************************************* %
% ** Inicializacion de las matrices ********************************** %
% ********************************************************************* %

[ G, L, A, B, H, H_1, Caux, G_g, D_sum, auxy, P ] = ...
RellenarMatrices( nEnt, nPer, nSal, nCoef, g, d, N1, N2, Nu,...
```

```
beta, gamma, rho, Sum_N2, Sum_N2_N1, Sum_Nu, Sum_Coef,...
ym, DuMin, DuMax, y0, u0);


% ********************************************************************** %
% ** Ejecución del algoritmo QDMC ************************************** %
% ********************************************************************** %

for i = 1:instantes;

% ** Variaciones en la referencia y en las perturbaciones ************** %

if      i<=10, ref(i,:) = y0;
elseif  i<=100, ref(i,:) = [6.7 65 0.8];
elseif  i<=200, ref(i,:) = [6.7 60 0.8];
else            ref(i,:) = [7.5 60 0.8];
end

if      i<=10,  v1_new = 8.0;
elseif  i<=50,  v1_new = 8.1;
elseif  i<=150, v2_new = 32.5;
else            v3_new = 22.5;
end

v1_new = 0.9*v1_old + 0.1*v1_new;
v2_new = 0.9*v2_old + 0.1*v2_new;
v3_new = 0.9*v3_old + 0.1*v3_new;

if      i<=10,  v = v0;
elseif  i<=50,  v = [v1_new; v2_new; v3_new];
elseif  i<=150, v = [v1_new; v2_new; v3_new];
else            v = [v1_new; v2_new; v3_new];
end

v1_old = v1_new;
v2_old = v2_new;
v3_old = v3_new;

% ********************************************************************** %

Dv = v - v_old;     v_old = v;
Du = u - u_old;     u_old = u;

[ Du_p, Du, T, L ] = Regulador_DMC( nEnt, nPer, nSal, nCoef, g, d,...
                                    N1, N2, Nu, Sum_N2,...
                                    Sum_N2_N1, Sum_Nu, ref(i,:),...
                                    alpha, ym, Du, u, Dv,...
                                    uMin, uMax, yMin, yMax,...
                                    G, L, A, B, H,...
                                    H_1, Caux, D_sum, auxy, P );

% ********************************************************************** %

% Cálculo de la salida predicha
yp = T + G_g*Du_p;
```

```
% Cálculo de la nueva acción de control a aplicar
u = u_old + Du;

% Acumular en vectores para después graficar datos
v_yp = [ v_yp; [yp(1), yp(N2(1)+1), yp(N2(1)+N2(2)+1)]];
v_Du = [ v_Du, Du ];
v_u  = [ v_u,  u ];
v_Dv = [ v_Dv, Dv ];
v_v  = [ v_v,  v ];


%=========================================================================
% Aplicacion de los incrementos y medición de la salida
%=========================================================================

[y1_, z_11] = filter(sis(1,1).num{:,:}, sis(1,1).den{:,:}, Du(1,1), z_11);
[y2_, z_12] = filter(sis(1,2).num{:,:}, sis(1,2).den{:,:}, Du(2,1), z_12);

[yd1_, zd_11] = filter(dis(1,1).num{:,:}, dis(1,1).den{:,:}, Dv(1,1), zd_11);
[yd2_, zd_12] = filter(dis(1,2).num{:,:}, dis(1,2).den{:,:}, Dv(2,1), zd_12);
[yd3_, zd_13] = filter(dis(1,3).num{:,:}, dis(1,3).den{:,:}, Dv(3,1), zd_13);

ym1vec(i+1) = ym1vec(i) + y1_ + y2_ + yd1_ + yd2_ + yd3_;

[y1_,z_21] = filter(sis(2,1).num{:,:}, sis(2,1).den{:,:}, Du(1,1), z_21);
[y2_,z_22] = filter(sis(2,2).num{:,:}, sis(2,2).den{:,:}, Du(2,1), z_22);

[yd1_, zd_21] = filter(dis(2,1).num{:,:}, dis(2,1).den{:,:}, Dv(1,1), zd_21);
[yd2_, zd_22] = filter(dis(2,2).num{:,:}, dis(2,2).den{:,:}, Dv(2,1), zd_22);
[yd3_, zd_23] = filter(dis(2,3).num{:,:}, dis(2,3).den{:,:}, Dv(3,1), zd_23);

ym2vec(i+1) = ym2vec(i) + y1_ + y2_ + yd1_ + yd2_ + yd3_;

[y1_,z_31] = filter(sis(3,1).num{:,:}, sis(3,1).den{:,:}, Du(1,1), z_31);
[y2_,z_32] = filter(sis(3,2).num{:,:}, sis(3,2).den{:,:}, Du(2,1), z_32);

[yd1_, zd_31] = filter(dis(3,1).num{:,:}, dis(3,1).den{:,:}, Dv(1,1), zd_31);
[yd2_, zd_32] = filter(dis(3,2).num{:,:}, dis(3,2).den{:,:}, Dv(2,1), zd_32);
[yd3_, zd_33] = filter(dis(3,3).num{:,:}, dis(3,3).den{:,:}, Dv(3,1), zd_33);


ym3vec(i+1) = ym3vec(i) + y1_ + y2_ + yd1_ + yd2_ + yd3_;

ym = [ym1vec(i+1), ym2vec(i+1), ym3vec(i+1)];

end

%* Sólo para gráficar ***************************************************%

y1_Min = yMin(1)*ones(length(t),1);
y1_Max = yMax(1)*ones(length(t),1);
y2_Min = yMin(2)*ones(length(t),1);
y2_Max = yMax(2)*ones(length(t),1);
y3_Min = yMin(3)*ones(length(t),1);
```

```
y3_Max = yMax(3)*ones(length(t),1);

u1_Min = uMin(1)*ones(length(t)-1,1);
u1_Max = uMax(1)*ones(length(t)-1,1);
u2_Min = uMin(2)*ones(length(t)-1,1);
u2_Max = uMax(2)*ones(length(t)-1,1);

Du1_Min = DuMin(1)*ones(length(t)-1,1);
Du1_Max = DuMax(1)*ones(length(t)-1,1);
Du2_Min = DuMin(2)*ones(length(t)-1,1);
Du2_Max = DuMax(2)*ones(length(t)-1,1);

ref(i+1,:) = ref(i,:);

%=========================================================================
% Gráficas de las variables
%=========================================================================
figure(1), set(gca,'FontSize',16)
subplot(2,2,1), plot(t,ym1vec,'--b','LineWidth',2);
                ylim([yMin(1)-0.2 yMax(1)+0.2])
                hold on
                %stairs(t,v_yp(:,1),':r','LineWidth',2);
                %stairs(t,HITO_cB,':r','LineWidth',2);
                stairs(t,ref(:,1),'k','LineWidth',1);
                plot(t,y1_Min,'--g','LineWidth',2);
                plot(t,y1_Max,'--g','LineWidth',2);
                title('Salida 1: cB','FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('yMedida','Referencia','Limites')
                %legend('yMedida','yHITO','Referencia','Limites')
subplot(2,2,2), plot(t,ym2vec,'--b','LineWidth',2);
                ylim([yMin(2)-5 yMax(2)+5])
                hold on
                %stairs(t,v_yp(:,2),':r','LineWidth',2);
                %stairs(t,HITO_Tl,':r','LineWidth',2);
                stairs(t,ref(:,2),'k','LineWidth',1);
                plot(t,y2_Min,'--g','LineWidth',2);
                plot(t,y2_Max,'--g','LineWidth',2);
                title('Salida 2: T','FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('yMedida','Referencia','Limites')
                %legend('yMedida','yHITO','Referencia','Limites')
subplot(2,2,3), plot(t,ym3vec,'--b','LineWidth',2);
                ylim([yMin(3)-0.1 yMax(3)+0.1])
                hold on
                %stairs(t,v_yp(:,3),':r','LineWidth',2);
                %stairs(t,HITO_cA,':r','LineWidth',2);
                stairs(t,ref(:,3),'k','LineWidth',1);
                plot(t,y3_Min,'--g','LineWidth',2);
                plot(t,y3_Max,'--g','LineWidth',2);
                hold off
                title('Salida 3: cA','FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('yMedida','Referencia','Limites')
                %legend('yMedida','yHITO','Referencia','Limites')
```

```
figure(2), set(gca,'FontSize',16)
t = t(1:end-1);
subplot(2,2,1), stairs(t,v_u(1,:),'--b','LineWidth',2);
                ylim([uMin(1)-10 uMax(1)+10])
                hold on
                plot(t,u1_Min,'--g','LineWidth',2);
                plot(t,u1_Max,'--g','LineWidth',2);
                hold off
                title('Accion de control 1: Fl',...
                    'FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('u1','Limites')
subplot(2,2,2), stairs(t,v_u(2,:),'--b','LineWidth',2);
                ylim([uMin(2)-10 uMax(2)+10])
                hold on
                plot(t,u2_Min,'--g','LineWidth',2);
                plot(t,u2_Max,'--g','LineWidth',2);
                hold off
                title('Accion de control 2: Fr',...
                    'FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('u2','Limites')
subplot(2,2,3), stairs(t,v_Du(1,:),'--b','LineWidth',2);
                ylim([DuMin(1)-1 DuMax(1)+1])
                hold on
                plot(t,Du1_Min,'--g','LineWidth',2);
                plot(t,Du1_Max,'--g','LineWidth',2);
                hold off
                title('Incremento en la accion de control 1: Fl',...
                    'FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                %legend('Du1','Du1HITO','Limites')
                legend('Du1','Limites')
subplot(2,2,4), stairs(t,v_Du(2,:),'--b','LineWidth',2);
                ylim([DuMin(2)-1 DuMax(2)+1])
                hold on
                plot(t,Du2_Min,'--g','LineWidth',2);
                plot(t,Du2_Max,'--g','LineWidth',2);
                hold off
                title('Incremento en la accion de control 2: Fr',...
                    'FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('Du2','Limites')

figure(3), set(gca,'FontSize',16)
subplot(2,2,1), stairs(t,v_v(1,:),'b','LineWidth',2);
                ylim([7.75 8.15])
                title('Perturbacion 1: cA0','FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('v1')
subplot(2,2,2), stairs(t,v_v(2,:),'b','LineWidth',2);
                ylim([29.5 33])
                title('Perturbacion 2: Tl0','FontSize',18,'FontWeight','bold')
                xlabel('Instantes')
```

```
                legend('v2')
subplot(2,2,3), stairs(t,v_v(3,:),'b','LineWidth',2);
                ylim([19.5 23])
                title('Perturbacion 2: Tr0','FontSize',18,'FontWeight','bold')
                xlabel('Instantes','FontSize',16)
                legend('v3')
```

```
function [ nEnt nPer nSal Ts g d nCoef sis dis y0 u0 v0] = RespuestaSalto( )

    clear all
    close all
    clc

    % Numero de Entradas, Perturbaciones Medibles y Salidas
    nEnt = 2;
    nPer = 3;
    nSal = 3;
    Ts = 60; %Segundos

    y0 = [7.2, 60, 0.8];
    u0 = [26.6665978; 25.822517];
    v0 = [8.0; 30.0; 20.0];

    % pol_a = zeros(3,7);
    % pol_b = zeros(2,6,3);
    % pol_d = zeros(3,6,3);

    pol_a = [   1.0,    -3.0534,    3.369994,   -1.579317,....
                0.27737852, -0.01427608,    0.0;
                % sal1: Cb
                1.0,    -3.5111,    4.6254905,  -2.70980711,...
                0.59567014,  0.0,           0.0;
                % sal2: Tl
                1.0,    -3.6213,    5.1413589,  -3.5869176,...
                1.2508025,  -0.19090276,    0.00717885
                % sal3: Ca
            ];


    pol_b(:,:,1) =  [   -0.03683910056770,   0.11094954714975,...
                        -0.11804852778705,   0.05023085961584,...
                        -0.00630005873810,  0.0;
                        % sal1: Cb / ent1: Fl \
                        -0.00024846607241,   0.00044005826084,...
                        -0.00019474186860,   0.0,...
                        0.0,                0.0
                        % sal1: Cb / ent2: Fr \
                    ];

    pol_b(:,:,2) =  [   0.06163700215760,   -0.16056439062055,...
                        0.13961397358719,  -0.04052016521841,...
                        0.0,                0.0;
                        % sal2: Tl / ent1: Fl \
                        -0.01012674720064,   0.01793548196706,...
                        -0.00793710647730,   0.0,...
                        0.0,                0.0
                        % sal2: Tl / ent2: Fr \
                    ];

    pol_b(:,:,3) =  [   0.01943539877748,   -0.06948155062950,...
                        0.09658297035918,  -0.06490763165537,...
                        0.02098415672885,  -0.00260912454428;
```

```
                    % sal3: Ca / ent1: Fl \
                    0.00024902141630,   -0.00055315127203,...
                    0.00039881748514,  -0.00009701976478,...
                    0.00000415741255,   0.0
                    % sal3: Ca / ent1: Fr \
                ];


pol_d(:,:,1) =  [   0.08839053190410, -0.18945626608324,...
                    0.12547036003786, -0.02541864204072,...
                    0.00138667066451,   0.0;
                    % sal1: Cb / per1: Ca0 \
                    0.00087098764213, -0.00186687491213,...
                    0.00123636695800, -0.00025047165822,...
                    0.00001366405413,   0.0;
                    % sal1: Cb / per2: Tl0 \
                    0.00070195444495, -0.00150456915731,...
                    0.00099642433461, -0.00020186244364,...
                    0.00001101226133,   0.0
                    % sal1: Cb / per3: Tr0 \
                ];

pol_d(:,:,2) =  [   0.63567441332302, -1.68205806509406,...
                    1.48532575411799, -0.43774828529311,...
                    0.0, 0.0;
                    % sal2: Tl / per1: Ca0 \
                    0.08023601070725, -0.21231250793246,...
                    0.18748058694483, -0.05525340546940,...
                    0.0, 0.0;
                    % sal2: Tl / per2: Tl0 \
                    0.06620433974585, -0.17518330340151,...
                    0.15469398309854, -0.04559069170523,...
                    0.0, 0.0
                    % sal2: Tl / per3: Tr0 \
                ];

pol_d(:,:,3) =  [   0.03412872910806,  -0.11273443711448,...
                    0.13813856826643, -0.07503936808018,...
                    0.01618589256366, -0.00067545245527;
                    % sal3: Ca / per1: Ca0 \
                   -0.00064839011670,   0.001516130609890,...
                   -0.00116147418385,  0.00030490480399,...
                   -0.00001329912968,  0.0;
                   % sal3: Ca / per2: Tl0 \
                   -0.00053565112321,   0.001252513021390,...
                   -0.00095952257002,  0.00025188940504,...
                   -0.00001098674019,  0.0
                   % sal3: Ca / per2: Tr0 \
                ];

sis(1,1) = filt(pol_b(1,:,1), pol_a(1,:), Ts);
sis(1,2) = filt(pol_b(2,:,1), pol_a(1,:), Ts);
sis(2,1) = filt(pol_b(1,:,2), pol_a(2,:), Ts);
sis(2,2) = filt(pol_b(2,:,2), pol_a(2,:), Ts);
sis(3,1) = filt(pol_b(1,:,3), pol_a(3,:), Ts);
```

```
sis(3,2) = filt(pol_b(2,:,3), pol_a(3,:), Ts);

g11 = step(sis(1,1));
g12 = step(sis(1,2));
g21 = step(sis(2,1));
g22 = step(sis(2,2));
g31 = step(sis(3,1));
g32 = step(sis(3,2));

coef_y1 = max([length(g11), length(g12)]);
coef_y2 = max([length(g21), length(g22)]);
coef_y3 = max([length(g31), length(g32)]);
nCoef = [coef_y1, coef_y2, coef_y3];

g11 = [ g11; g11(end)*ones( coef_y1 - length(g11), 1)];
g12 = [ g12; g12(end)*ones( coef_y1 - length(g12), 1)];
g21 = [ g21; g21(end)*ones( coef_y2 - length(g21), 1)];
g22 = [ g22; g22(end)*ones( coef_y2 - length(g22), 1)];
g31 = [ g31; g31(end)*ones( coef_y3 - length(g31), 1)];
g32 = [ g32; g32(end)*ones( coef_y3 - length(g32), 1)];

g = [   g11     g12
        g21     g22
        g31     g32
    ];

% figure
% subplot(3,2,1),plot(g11)
% subplot(3,2,2),plot(g12)
% subplot(3,2,3),plot(g21)
% subplot(3,2,4),plot(g22)
% subplot(3,2,5),plot(g31)
% subplot(3,2,6),plot(g32)

dis(1,1) = filt(pol_d(1,:,1), pol_a(1,:), Ts);
dis(1,2) = filt(pol_d(2,:,1), pol_a(1,:), Ts);
dis(1,3) = filt(pol_d(3,:,1), pol_a(1,:), Ts);
dis(2,1) = filt(pol_d(1,:,2), pol_a(2,:), Ts);
dis(2,2) = filt(pol_d(2,:,2), pol_a(2,:), Ts);
dis(2,3) = filt(pol_d(3,:,2), pol_a(2,:), Ts);
dis(3,1) = filt(pol_d(1,:,3), pol_a(3,:), Ts);
dis(3,2) = filt(pol_d(2,:,3), pol_a(3,:), Ts);
dis(3,3) = filt(pol_d(3,:,3), pol_a(3,:), Ts);


d11 = step(dis(1,1), (coef_y1-1)*Ts);
d12 = step(dis(1,2), (coef_y1-1)*Ts);
d13 = step(dis(1,3), (coef_y1-1)*Ts);
d21 = step(dis(2,1), (coef_y2-1)*Ts);
d22 = step(dis(2,2), (coef_y2-1)*Ts);
d23 = step(dis(2,3), (coef_y2-1)*Ts);
d31 = step(dis(3,1), (coef_y3-1)*Ts);
d32 = step(dis(3,2), (coef_y3-1)*Ts);
d33 = step(dis(3,3), (coef_y3-1)*Ts);
```

```
        % figure
        % subplot(3,3,1),plot(d11)
        % subplot(3,3,2),plot(d12)
        % subplot(3,3,3),plot(d13)
        % subplot(3,3,4),plot(d21)
        % subplot(3,3,5),plot(d22)
        % subplot(3,3,6),plot(d23)
        % subplot(3,3,7),plot(d31)
        % subplot(3,3,8),plot(d32)
        % subplot(3,3,9),plot(d33)


        d = [   d11     d12   d13
                d21     d22   d23
                d31     d32   d33
            ];

    end
```

```
function [ G, L, A, B, H, H_1, Caux, G_g, D_sum, auxy, P ] = ...
RellenarMatrices( nEnt, nPer, nSal, nCoef, g, d, N1, N2, Nu, beta,...
gamma, rho, Sum_N2, Sum_N2_N1, Sum_Nu, Sum_Coef, ym, DuMin, DuMax, y0, u0)

    beta = 100*beta./u0';

    gamma = 100*gamma./y0;

    % ********************************************************************* %
    % * Matriz D                                                          %
    %   Va acumulando en cada posición vertical la suma de los elementos    %
    %   anteriores más él mismo                                           %
    % ********************************************************************* %

    ini = 0;
    ini_d = 0;
    for i = 1:nSal
        D_sum(ini + 1, :) = d(ini_d + 1, :);
        for j = 2:N2(i)
            if j<nCoef(i)
                for k = 1:nPer
                    D_sum(ini + j, k) = D_sum(ini + j - 1, k) + d(ini_d + j, k);
                end
            else
                for k = 1:nPer
                    D_sum(ini + j, k) = D_sum(ini + j - 1, k) + d(ini_d + nCoef↙
(i), k);
                end
            end
        end
        ini = ini + j;
        ini_d = ini_d + nCoef(i);
    end

    % ********************************************************************* %
    % * Matriz G grande (Contiene todos los gij entre 1 y N2)             %
    %       Aunque en la función de costes sólo quiero que se tenga en cuenta %
    %       entre N1 y N2, para los límites quiero poder considerar todo N2   %
    % ********************************************************************* %

    ini_g = 0;      % Columna donde comienza la siguiente gij
    fil_Gij = 0;    % Indice para las filas de la matriz G
    fil_aux = 0;    % Indice para las filas de la submatriz Gij

    for i=1:nSal

        fil_Gij = fil_Gij + fil_aux;
        fil_aux = N2(i);    % Indice para las filas de la submatriz Gij

        col_Gij = 0; % Indice para las columnas de la matriz G
        col_aux = 0; % Indice para las columnas de la submatriz Gij

        for j=1:nEnt
```

```
            col_Gij = col_Gij + col_aux;
            col_aux = Nu(j);

            for k = 1:fil_aux         % Hacia abajo aumenta g
                for l = 1:col_aux     % Hacia la derecha diminuye g
                    col = (k-1) - (l-1) + 1;
                    if (col >= 1 && col <= nCoef(i))
                        G_g(fil_Gij+k, col_Gij+l) = g(ini_g + col, j);
                    elseif (col < 1 )
                        G_g(fil_Gij+k, col_Gij+l) = 0;
                    elseif (col > nCoef(i) )
                        G_g(fil_Gij+k, col_Gij+l) = g(ini_g + nCoef(i), j);
                    end
                end
            end
        end

        ini_g = ini_g + nCoef(i);

    end


    % ************************************************************************ %
    % * Matriz G (Matriz G_g truncada desde N1)                                  %
    % ************************************************************************ %

    fil_G = Sum_N2_N1;
    col_G = Sum_Nu;
    G = zeros(fil_G, col_G);

    ini_g = 0;      % Columna donde comienza la siguiente gij
    fil_Gij = 0;    % Indice para las filas de la matriz G
    fil_aux = 0;    % Indice para las filas de la submatriz Gij

    for i=1:nSal

        fil_Gij = fil_Gij + fil_aux;
        fil_aux = N2(i) - N1(i) + 1;    % Indice para las filas de la submatriz↙
Gij

        col_Gij = 0; % Indice para las columnas de la matriz G
        col_aux = 0; % Indice para las columnas de la submatriz Gij

        for j=1:nEnt

            col_Gij = col_Gij + col_aux;
            col_aux = Nu(j);

            for k = 1:fil_aux         % Hacia abajo aumenta g
                for l = 1:col_aux     % Hacia la derecha diminuye g
                    col = (k-1) + N1(i) - (l-1);
                    if (col >= 1 && col <= nCoef(i))
                        G(fil_Gij+k, col_Gij+l) = g(ini_g + col, j);
                    elseif (col < 1 )
                        G(fil_Gij+k, col_Gij+l) = 0;
```

```
                elseif (col > nCoef(i) )
                    G(fil_Gij+k, col_Gij+l) = g(ini_g + nCoef(i), j);
                end
            end
        end
    end

    ini_g = ini_g + nCoef(i);

end


% ************************************************************************ %
% * Matrices de pesos Beta (peso cada Du) y Gamma (peso cada y)            %
% ************************************************************************ %

% ** Matriz Beta: M. diagonal con el peso asignado a cada Du ******** %

fil_Beta = Sum_Nu;
col_Beta = Sum_Nu;
Beta = zeros(fil_Beta, col_Beta);

Ini = 0;
for i=1:nEnt
    nFil = Ini + Nu(i);
    nCol = Ini + Nu(i);
    for j=Ini+1:nFil
        for k=Ini+1:nCol
            if j==k
                Beta(j,k) = beta(i);
            else
                Beta(j,k) = 0;
            end
        end
    end
    Ini = nFil;
end

% ** Matriz Gamma: Matriz diagonal con el peso asignado a cada y **** %

fil_Gamma = Sum_N2_N1;
col_Gamma = Sum_N2_N1;
Gamma = zeros(fil_Gamma, col_Gamma);

Ini = 0;
for i=1:nSal
    nFil = Ini + N2(i) -  N1(i) + 1;
    nCol = Ini + N2(i) -  N1(i) + 1;
    for j=Ini+1:nFil
        for k=Ini+1:nCol
            if j==k
                Gamma(j,k) = gamma(i);
            else
                Gamma(j,k) = 0;
            end
```

```
        end
    end
    Ini = nFil;
end

% ************************************************************************ %
% * Matriz hessiana                                                      * %
% ************************************************************************ %

fil_Ceros = nSal;
col_Ceros = Sum_Nu;
Ceros = zeros(fil_Ceros, col_Ceros);

fil_H = Sum_Nu + nSal;
col_H = Sum_Nu + nSal;

H_aux = G'*Gamma*G + Beta;

k = 1;
for i=1:fil_H
    for j=1:col_H
        if (i<=Sum_Nu && j<=Sum_Nu )
            H(i,j) = H_aux(i,j);
        elseif i==j
            H(i,j) = rho(k);
            k = k+1;
        else
            H(i,j) = 0;
        end
    end
end

H = (H+H')/2;   % Hessiano simétrico
% H = H + 0.1*eye(size(H));


% ************************************************************************ %
% * Matrices para las restricciones lineales                             * %
% ************************************************************************ %

% Matriz A - Coefs restricciones lineales
%  - DuMax      [Sum_Nu, (Sum_Nu + nSal)]
%  - DuMin      [Sum_Nu, (Sum_Nu + nSal)]
%  - uMax       [Sum_Nu, (Sum_Nu + nSal)]
%  - uMin       [Sum_Nu, (Sum_Nu + nSal)]
%  - yMax       [Sum_N2, (Sum_Nu + nSal)]
%  - yMin       [Sum_N2, (Sum_Nu + nSal)]
%  - Epsilon    [nSal,   (Sum_Nu + nSal)]

fil_A = 4*Sum_Nu + 2*Sum_N2 + nSal;
col_A = Sum_Nu + nSal;

for i = 1:Sum_Nu            % Matriz Identidad I
    for j = 1:col_A
        if i==j
```

```
            A(i,j) = 1;
        else
            A(i,j) = 0;
        end
    end
end

Ini = Sum_Nu;

for i = 1:Sum_Nu            % Matriz Identidad -I
    for j = 1:col_A
        if i==j
            A(Ini+i,j) = -1;
        else
            A(Ini+i,j) = 0;
        end
    end
end

Ini = Ini + Sum_Nu;

Ini_col = 0;                          % Matriz de Diagonales Inferiores I
for i = 1:nEnt      % Filas
    for j = 1:nEnt  % Columnas
        for k = 1:Nu(i)               % Filas
            for l = 1:Nu(j)+nSal      % Columnas
                if (i==j && k>=l)
                    A(Ini+k, Ini_col+l) = 1;
                else
                    A(Ini+k, Ini_col+l) = 0;
                end
            end
        end
        Ini_col = Ini_col + Nu(j);
    end
    Ini_col = 0;
    Ini = Ini + Nu(i);
end


Ini_col = 0;                          % Matriz de Diagonales Inferiores -I
for i = 1:nEnt      % Filas
    for j = 1:nEnt  % Columnas
        for k = 1:Nu(i)               % Filas
            for l = 1:Nu(j)+nSal      % Columnas
                if (i==j && k>=l)
                    A(Ini+k, Ini_col+l) = -1;
                else
                    A(Ini+k, Ini_col+l) = 0;
                end
            end
        end
        Ini_col = Ini_col + Nu(j);
    end
    Ini_col = 0;
```

```
    Ini = Ini + Nu(i);
end


for i = 1:Sum_N2            % Matriz G
    for j = 1:Sum_Nu
        A(Ini+i,j) = G_g(i,j);
    end
end

for i = 1:nSal              % A la dcha de G
    for j = 1:nSal
        for k = 1:N2(j)
            if i==j
                A(Ini+k, Sum_Nu+i) = -1;
            else
                A(Ini+k, Sum_Nu+i) =  0;
            end
        end
        Ini = Ini + k;
    end
    Ini = Ini - Sum_N2;
end

Ini = Ini + Sum_N2;

for i = 1:Sum_N2            % Matriz -G
    for j = 1:Sum_Nu
        A(Ini+i,j) = -G_g(i,j);
    end
end

for i = 1:nSal                  % A la dcha de -G
    for j = 1:nSal
        for k = 1:N2(j)
            if i==j
                A(Ini+k, Sum_Nu+i) = -1;
            else
                A(Ini+k, Sum_Nu+i) =  0;
            end
        end
        Ini = Ini + k;
    end
    Ini = Ini - Sum_N2;
end

Ini = Ini + Sum_N2;

for i = 1:nSal
    for j = 1:Sum_Nu
        A(Ini+i,j) = 0;
    end
end

for i = 1:nSal
```

```
    for j = 1:nSal
        if i==j
            A(Ini+i,Sum_Nu+j) = -1;
        end
    end
end

% Matriz b - Termino indpte de las restricciones lineales
%  - DuMax [Sum_Nu, 1]  - No cambia
%  - DuMin [Sum_Nu, 1]  - No cambia
%  - uMax  [Sum_Nu, 1]
%  - uMin  [Sum_Nu, 1]
%  - yMax  [Sum_N2_N1, 1]
%  - yMin  [Sum_N2_N1, 1]
%  - Epsilon [nEnt, 1]

fil_B = 4*Sum_Nu + 2*Sum_N2_N1 + nSal;
col_B = 1;

% Limites en los incrementos

Ini = 0;                            % Du máximo
for i = 1:nEnt
   for j = 1:Nu(i)
       B(Ini+j, 1) = DuMax(i);
   end
   Ini = Ini + j;
end
for i = 1:nEnt
   for j = 1:Nu(i)                  % Du mínimo
       B(Ini+j, 1) = -DuMin(i);
   end
   Ini = Ini + j;
end


% ************************************************************************ %
% * Inicializacion de la resuesta libre                                    %
% ************************************************************************ %

fil_L = Sum_Coef;
col_L = 1;
L = zeros(fil_L, col_L);

k = 1;
for i = 1:nSal
    for j = 1:nCoef(i)
%         L(k, 1) = ym(i);
        L(k, 1) = 0;
        k = k+1;
    end
end

% ************************************************************************ %
% * Matriz auxiliar para el gradiente                                      %
```

```
    % ************************************************************************ %

    Caux = -G'*Gamma;


    H_1 = inv(H);
    auxy = H_1*A';
    P = A*auxy;

end
```

```
function [ Du_p, Du, T, L ] = ...
    Regulador_DMC( nEnt, nPer, nSal, nCoef, g, d, N1, N2, Nu,...
    Sum_N2, Sum_N2_N1, Sum_Nu, ref, alpha,...
    ym, Du, u, Dv, uMin, uMax, yMin, yMax,...
    G, L, A, B, H, H_1, Caux, D_sum, auxy, P )

    % ********************************************************************** %
    % * Actualizar la respuesta libre                                      * %
    % ********************************************************************** %

    ini = 0;
    for j = 1:nSal
        for k = 2:nCoef(j)
            L(ini+k-1, 1) = L(ini+k, 1);
        end
        ini = ini + nCoef(j);
    end

    if nPer>0
        L = L + g*Du + d*Dv;
    else
        L = L + g*Du;
    end


    ini   = 0;
    ini_T = 0;
    for j = 1:nSal
        for k = 2:N2(j)+1
            if k <= nCoef(j)
                T(ini_T+k-1, 1) = L(ini+k, 1)        + ym(j) - L(ini + 1);
            else
                T(ini_T+k-1, 1) = L(ini+nCoef(j), 1) + ym(j) - L(ini + 1);
            end
        end
        ini   = ini   + nCoef(j) ;
        ini_T = ini_T + N2(j);
    end

    if nPer>0
        T = T + D_sum*Dv;
    end

    % Referencia con filtro de primer orden

    ini   = 0;
    for j = 1:nSal
        y_ant = ym(j);
        for k = 1:N2(j)
            R(ini + k, 1) = (1-alpha(j))*ref(j) + alpha(j)*y_ant;
            y_ant = R(ini + k, 1);
        end
        ini = ini + N2(j) ;
    end
```

```
% No se coge el error de toda la respuesta libre, sólo entre N1 y N2

k   = 0;
ini = 0;
for i = 1:nSal
    for j = N1(i):N2(i)
        k = k+1;
        D(k,1) = R(ini+j, 1) - T(ini+j, 1) ;
    end
    ini = ini + N2(i) ;
end


% ************************************************************************* %
% * Restricciones                                                         * %
% ************************************************************************* %


Ini = 2*Sum_Nu;
for j = 1:nEnt
   for k = 1:Nu(j)
       B(Ini+k, 1) =  uMax(j) - u(j);
   end
   Ini = Ini + k;
end

for j = 1:nEnt
   for k = 1:Nu(j)
       B(Ini+k, 1) = -uMin(j) + u(j);
   end
   Ini = Ini + k;
end

Ini_T = 0;
for j = 1:nSal
   for k = 1:N2(j)
       B(Ini+k, 1) = yMax(j) - T(Ini_T + k, 1);
   end
   Ini = Ini + k;
   Ini_T = Ini_T + k;
end

Ini_T = 0;
for j = 1:nSal
   for k = 1:N2(j)
       B(Ini+k, 1) = -yMin(j) + T(Ini_T + k, 1);
   end
   Ini = Ini + k;
   Ini_T = Ini_T + k;
end

for j = 1:nSal
   B(Ini+j, 1) = 0;
end
```

```
    C = [Caux*D;0;0;0];


    % ********************************************************************** %
    % * LLamada al optimizador                                             * %
    % ********************************************************************** %

    % options = optimset('Algorithm', 'interior-point-convex');
    % [x,fval,exitflag,output] = quadprog(H,C,A,B,[],[],[],[],[],options);
    % [x,fval,exitflag,output] = quadprog(H,C,A,B);

    [x, nIter]  = QPhild (H_1, C, A, B, auxy, P, 1000);
    if nIter~=1
        disp(nIter);
    end
    % Du predicha (Los tres ultimos valores son las variables de holgura)
    Du_p = x(1:end-3);

    % Sólo se aplica la primera acción de control a cada entrada
    ini = 0;
    for i=1:nEnt
        Du(i,1) = x(ini+1);
        ini = ini + Nu(1,i);
    end

end
```

```
function [eta, km] = QPhild ( H_1, f, A_cons, b, auxy, P, n_iteration )

    tol = 10e-5;

    [n1,~] = size(A_cons);
    eta = -H_1*f;
    kk = 0;

    for i=1:n1
        if (A_cons(i,:)*eta>b(i))
            kk = kk + 1;
        else
            kk = kk + 0;
        end
    end

    if (kk==0)
        km=1;
        return;
    end

    d = auxy'*f + b;
    [n,m] = size(d);
    x_ini = zeros(n,m);
    lambda = x_ini;

    for km = 1:n_iteration
        lambda_p = lambda;
        for i = 1:n
            w = P(i,:)*lambda - P(i,i)*lambda(i,1);
            w = w + d(i,1);
            la = -w/P(i,i);
            lambda(i,1) = max(0,la);
        end
        al =(lambda-lambda_p)'*(lambda-lambda_p);
        if (al<tol)
            break;
        end
    end

    eta = -H_1*f - auxy*lambda;

end
```

ANEXO 3:

CÓDIGO FUENTE DEL DMC: STEP7

```
 1 DATA_BLOCK DB_QP_P_1
 2
 3     STRUCT
 4
 5     P: ARRAY [1..65, 1..243] OF REAL;
 6
 7     END_STRUCT
 8
 9 BEGIN
10
11 END_DATA_BLOCK
12
```

```
13 DATA_BLOCK DB_QP_P_2
14
15     STRUCT
16
17     P: ARRAY [1..65, 1..243] OF REAL;
18
19     END_STRUCT
20
21 BEGIN
22
23 END_DATA_BLOCK
24
```

```
25 DATA_BLOCK DB_QP_P_3
26
27     STRUCT
28
29     P: ARRAY [1..65, 1..243] OF REAL;
30
31     END_STRUCT
32
33 BEGIN
34
35 END_DATA_BLOCK
36
```

```
37 DATA_BLOCK DB_QP_P_4
38
39     STRUCT
40
41     P: ARRAY [1..48, 1..243] OF REAL;
42
43     END_STRUCT
44
45 BEGIN
46
47 END_DATA_BLOCK
48
```

```
49 DATA_BLOCK DB_aux
50
51     STRUCT
52
53     eta:         ARRAY [1..18]  OF REAL;
54     p:           ARRAY [1..243] OF REAL;
55     lambda:      ARRAY [1..243] OF REAL;
56     lambda_p:    ARRAY [1..243] OF REAL;
57
58     w:  REAL;
59     la: REAL;
60     al: REAL;
61
62     aux_1:  ARRAY [1..243] OF REAL;
63     aux_2:  ARRAY [1..18, 1..243] OF REAL;
64     aux_3:  ARRAY [1..18] OF REAL;
65
66     END_STRUCT
67
68 BEGIN
69
70 END_DATA_BLOCK
71
72
```

```
1  FUNCTION FC_Optimizador_QP: VOID
2  // ********************************************************************************************** //
3
4  CONST
5      // constantes
6      nEnt_max := 3;
7      nPer_max := 3;
8      nSal_max := 3;
9
10     nCoef_max := 100;
11     Sum_nCoef_max := nCoef_max*nSal_max;
12
13     N2_max := 30;
14     Sum_N2_max := N2_max*nSal_max;
15
16     Nu_max := 5;
17     Sum_Nu_max := Nu_max*nEnt_max;
18
19     Dim_A :=  Sum_Nu_max + nSal_max;
20     Dim_B :=  4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;
21
22 END_CONST
23
24 // ********************************************************************************************** //
25     VAR_INPUT
26
27         MaxIter:    INT;
28
29         Dim_1:      INT;    // Numero de columnas de la matriz A
30         Dim_2:      INT;    // Numero de filas de la matriz A
31
32     END_VAR
33 // ********************************************************************************************** //
34     VAR_OUTPUT
35
36         nIter:  INT;
37
38     END_VAR
39 // ********************************************************************************************** //
40     VAR_IN_OUT
41
42         A:      ARRAY [1..Dim_B, 1..Dim_A]  OF REAL;
43         B:      ARRAY [1..Dim_B]            OF REAL;
44         C:      ARRAY [1..Dim_A]            OF REAL;
45         invH:   ARRAY [1..Dim_A, 1..Dim_A]  OF REAL;
46
47         Sol:    ARRAY [1..Dim_A]  OF REAL;
48
49     END_VAR
50 // ********************************************************************************************** //
51     VAR_TEMP
52
53         km, i, j, k: INT;  // Indices
54         aux :INT;
55
56         nFil_A, nFil_B, nCol_A, nCol_B: INT;
57
58         auxy: REAL;
59
60     END_VAR
61 // ********************************************************************************************** //
62     VAR
63
64         //eta:          ARRAY [1..Dim_A]  OF REAL;
65         //p:            ARRAY [1..Dim_B] OF REAL;
66         //lambda:       ARRAY [1..Dim_B] OF REAL;
67         //lambda_p:     ARRAY [1..Dim_B] OF REAL;
68
69         //w:            REAL;
70         //la:           REAL;
71         //al:           REAL;
72
73         //aux_1:        ARRAY [1..Dim_B] OF REAL;
74         //aux_2:        ARRAY [1..Dim_A, 1..Dim_B] OF REAL;
```

```
75          //aux_3:      ARRAY [1..Dim_A] OF REAL;
76
77      END_VAR
78  // ********************************************************************************
    ********** //
79
80  BEGIN
81
82      (*************************************************************************)
83      (* Calcular eta (Solucion analitica del problema sin restricciones)      *)
84      (*     eta = -inv_H*C                                                    *)
85      (*************************************************************************)
86
87      nFil_A := Dim_1;
88      nCol_A := Dim_1;
89
90      nFil_B := Dim_1;
91      nCol_B := 1;
92
93      FOR i := 1 TO nFil_A DO
94          DB_aux.eta[i] := 0.0;
95          FOR j :=1 TO nCol_A DO
96              // CUIDADO ESTA CAMBIADA DE SIGNO (-)
97              DB_aux.eta[i] := DB_aux.eta[i] - (invH[i,j]*C[j]);
98          END_FOR;
99      END_FOR;
100
101
102     (*************************************************************************)
103     (* Comprobar si la solucion del problema sin restricciones las cumple    *)
104     (*     A_cons*eta > b                                                    *)
105     (*************************************************************************)
106
107     nFil_A := Dim_2;
108     nCol_A := Dim_1;
109
110     nFil_B := Dim_1;
111     nCol_B := 1;
112
113
114     FOR i := 1 TO nFil_A DO
115         DB_aux.aux_1[i] := 0.0;
116         FOR j :=1 TO nCol_A DO
117             DB_aux.aux_1[i] := DB_aux.aux_1[i] + (A[i,j]*DB_aux.eta[j]);
118         END_FOR;
119     END_FOR;
120
121     aux := 0;
122
123     FOR i:=1 TO  nFil_A DO
124         IF (DB_aux.aux_1[i]>B[i]) THEN
125             aux := aux + 1;
126         ELSE
127             aux := aux + 0;
128         END_IF;
129     END_FOR;
130
131     IF(aux = 0)THEN
132         //Sale sin iterar
133         nIter := 1;
134         Sol   := DB_aux.eta;
135         RETURN;
136     END_IF;
137
138
139
140     (*************************************************************************)
141     (* Calcular d                                                            *)
142     (*     p =  A*inv_H*C + B                                                *)
143     (*     p = (aux_2')*C + B                                                *)
144     (*************************************************************************)
145
146     nFil_A := Dim_2;
147     nCol_A := Dim_1;
148
149     nFil_B := Dim_1;
150     nCol_B := 1;
151
152     FOR i := 1 TO nFil_A DO
153         DB_aux.p[i] := B[i];
```

```
154         FOR j :=1 TO nCol_A DO
155             DB_aux.p[i] := DB_aux.p[i] + (DB_aux.aux_2[j,i]*C[j]);
156         END_FOR;
157     END_FOR;
158
159
160     (*************************************************************************)
161     (* Inicializar lambda                                                    *)
162     (*************************************************************************)
163
164     FOR i:=1 TO (Dim_2) DO
165         DB_aux.lambda[i] := 0.0;
166     END_FOR;
167
168
169     (*************************************************************************)
170     (* Iteraciones                                                           *)
171     (*************************************************************************)
172
173     DB_aux.al := 0.0;
174     FOR km:=1 TO MaxIter DO
175
176         DB_aux.lambda_P := DB_aux.lambda;
177
178         FOR i:=1 TO (Dim_2) DO
179
180             // w = P(i,:)*lambda - P(i,i)*lambda(i,1) + p(i,1);
181             // la = -w/P(i,i);
182
183             IF i<=65 THEN
184
185                 DB_aux.w := -DB_QP_P_1.P[i,i]*DB_aux.lambda[i] + DB_aux.p[i];
186                 FOR j:=1 TO (Dim_2) DO
187                     DB_aux.w := DB_aux.w + DB_QP_P_1.P[i,j]*DB_aux.lambda[j];
188                 END_FOR;
189
190                 DB_aux.la := -DB_aux.w / DB_QP_P_1.P[i,i];
191
192             ELSIF i<=130 THEN
193
194                 DB_aux.w := -DB_QP_P_2.P[i-65,i]*DB_aux.lambda[i] + DB_aux.p[i];
195                 FOR j:=1 TO (Dim_2) DO
196                     DB_aux.w := DB_aux.w + DB_QP_P_2.P[i-65,j]*DB_aux.lambda[j];
197                 END_FOR;
198
199                 DB_aux.la := -DB_aux.w / DB_QP_P_2.P[i-65,i];
200
201             ELSIF i<=195 THEN
202
203                 DB_aux.w := -DB_QP_P_3.P[i-130,i]*DB_aux.lambda[i] + DB_aux.p[i];
204                 FOR j:=1 TO (Dim_2) DO
205                     DB_aux.w := DB_aux.w + DB_QP_P_3.P[i-130,j]*DB_aux.lambda[j];
206                 END_FOR;
207
208                 DB_aux.la := -DB_aux.w / DB_QP_P_3.P[i-130,i];
209
210             ELSIF i<=243 THEN
211
212                 DB_aux.w := -DB_QP_P_4.P[i-195,i]*DB_aux.lambda[i] + DB_aux.p[i];
213                 FOR j:=1 TO (Dim_2) DO
214                     DB_aux.w := DB_aux.w + DB_QP_P_4.P[i-195,j]*DB_aux.lambda[j];
215                 END_FOR;
216
217                 DB_aux.la := -DB_aux.w / DB_QP_P_4.P[i-195,i];
218
219             END_IF;
220
221             IF(DB_aux.la>0) THEN
222                 DB_aux.lambda[i]:= DB_aux.la;
223             ELSE
224                 DB_aux.lambda[i]:= 0.0;
225             END_IF;
226
227
228             DB_aux.al := DB_aux.al + ((DB_aux.lambda[i] - DB_aux.lambda_P[i])*(DB_aux.
    lambda[i] - DB_aux.lambda_P[i]));
229
230         END_FOR;
231
232
```

```
233         IF(DB_aux.al<1E-5) THEN
234             EXIT;
235         END_IF;
236
237     END_FOR;
238
239
240     (*************************************************************************)
241     (* Solucion del problema con restricciones                               *)
242     (*  eta = -H_1*f - H_1*A_cons'*lambda                                    *)
243     (*  eta = eta - aux_2*lambda                                             *)
244     (*************************************************************************)
245
246     nFil_A := Dim_1;
247     nCol_A := Dim_2;
248
249     nFil_B := Dim_2;
250     nCol_B := 1;
251
252     FOR i := 1 TO nFil_A DO
253         FOR j :=1 TO nCol_A DO
254             DB_aux.eta[i] := DB_aux.eta[i] - (DB_aux.aux_2[i,j]*DB_aux.lambda[j]);
255         END_FOR;
256     END_FOR;
257
258
259     nIter := km;
260     Sol   := DB_aux.eta;
261
262
263 END_FUNCTION
264
265
266
267
```

```
FUNCTION FC_Inicializar_Ctes : VOID

// ********************************************************************* //
CONST
    // constantes
    nEnt_max := 3;
    nPer_max := 3;
    nSal_max := 3;

    nCoef_max := 100;
    Sum_nCoef_max := nCoef_max*nSal_max;

    N2_max := 30;
    Sum_N2_max := N2_max*nSal_max;

    Nu_max := 5;
    Sum_Nu_max := Nu_max*nEnt_max;

    Dim_1 := Sum_Nu_max + nSal_max;
    Dim_2 := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;

END_CONST
// ********************************************************************* //
VAR_INPUT
    nEnt:   INT;
    nSal:   INT;
END_VAR
// ********************************************************************* //
VAR_OUTPUT

    Sum_N2:     INT;
    Sum_N2_N1:  INT;
    Sum_Nu:     INT;
    Sum_nCoef:  INT;

END_VAR
// ********************************************************************* //
VAR_IN_OUT

    nCoef:  ARRAY [1..nSal_max] OF INT;

    u0:     ARRAY [1..nEnt_max] OF REAL; // Punto en el que se obtuvo el modelo
    y0:     ARRAY [1..nSal_max] OF REAL; // Punto en el que se obtuvo el model

    N1:     ARRAY [1..nSal_max] OF INT;  // Inicio del horizonte de coincidencia
    N2:     ARRAY [1..nSal_max] OF INT;  // Horizonte de prediccion
    Nu:     ARRAY [1..nEnt_max] OF INT;  // Horizonte de control

    beta:   ARRAY [1..nEnt_max] OF REAL; // Peso de las acciones de control
    gamma:  ARRAY [1..nSal_max] OF REAL; // Peso del error de cada salida del sistema


END_VAR
// ********************************************************************* //
VAR_TEMP
    // variables temporales
    i,j:    INT;
END_VAR

// ********************************************************************* //

BEGIN

    (*************************************************************************)
    (* Normalizar gamma y beta                                               *)
    (*  Dividir por el valor inicial de las entradas/salidas                 *)
    (*************************************************************************)

    FOR i := 1 TO nEnt DO
        beta[i] := 100*beta[i]/u0[i];
    END_FOR;

    FOR i := 1 TO nSal DO
        gamma[i] := 100*gamma[i]/y0[i];
    END_FOR;

    (*************************************************************************)
    (* Sumatorio de Nu, N2 y N2-N1-1                                         *)
    (*************************************************************************)
```

```
80
81      Sum_Nu := 0;
82      FOR i := 1 TO nEnt DO
83          Sum_Nu := Sum_Nu + Nu[i];
84      END_FOR;
85
86      Sum_N2     := 0;
87      Sum_N2_N1 := 0;
88      Sum_nCoef := 0;
89      FOR i := 1 TO nSal DO
90          Sum_N2     := Sum_N2     + N2[i];
91          Sum_N2_N1 := Sum_N2_N1 + N2[i] - N1[i] + 1;
92          Sum_nCoef := Sum_nCoef + nCoef[i];
93      END_FOR;
94
95  END_FUNCTION
96
```

```
FUNCTION FC_Vaciar_Matrices : VOID

// ********************************************************************** //
CONST
    // constantes
    nEnt_max := 3;
    nPer_max := 3;
    nSal_max := 3;

    nCoef_max := 100;
    Sum_nCoef_max := nCoef_max*nSal_max;

    N2_max := 30;
    Sum_N2_max := N2_max*nSal_max;

    Nu_max := 5;
    Sum_Nu_max := Nu_max*nEnt_max;

    Dim_1 := Sum_Nu_max + nSal_max;
    Dim_2 := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;

END_CONST
// ********************************************************************** //
VAR_IN_OUT

    D_sum:  ARRAY [1..Sum_N2_max,   1..nEnt_max]    OF REAL;

    G:      ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;
    G_g:    ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;

    A:      ARRAY [1..Dim_2,        1..Dim_1]       OF REAL;
    B:      ARRAY [1..Dim_2]                        OF REAL;

    H:      ARRAY [1..Dim_1,        1..Dim_1]       OF REAL;
    inv_H:  ARRAY [1..Dim_1,        1..Dim_1]       OF REAL;

    C:      ARRAY [1..Dim_1]                        OF REAL;
    C_aux:  ARRAY [1..Sum_Nu_max,   1..Sum_N2_max]  OF REAL;

    L:      ARRAY [1..Sum_nCoef_max]                OF REAL;

END_VAR
// ********************************************************************** //
VAR_TEMP
    // variables temporales
    i,j:    INT;
END_VAR
// ********************************************************************** //
BEGIN

    FOR i := 1 TO Sum_N2_max DO
        FOR j := 1 TO nPer_max DO
            D_sum[i,j] := 0.0;
        END_FOR;
    END_FOR;

    FOR i := 1 TO Sum_N2_max DO
        FOR j := 1 TO Sum_Nu_max DO
            G[i,j] := 0.0;
        END_FOR;
    END_FOR;

    FOR i := 1 TO Sum_N2_max DO
        FOR j := 1 TO Sum_Nu_max DO
            G_g [i,j] := 0.0;
        END_FOR;
    END_FOR;

    FOR i := 1 TO Dim_2 DO
        FOR j := 1 TO Dim_1 DO
            A[i,j] := 0.0;
        END_FOR;
    END_FOR;

    FOR i := 1 TO Dim_2 DO
        B[i] := 0.0;
    END_FOR;

    FOR i := 1 TO Dim_1 DO
        FOR j := 1 TO Dim_1 DO
```

```
81              H[i,j] := 0.0;
82          END_FOR;
83      END_FOR;
84
85      FOR i := 1 TO Sum_Nu_max DO
86          FOR j := 1 TO Sum_N2_max DO
87              C_aux[i,j] := 0.0;
88          END_FOR;
89      END_FOR;
90
91      FOR i := 1 TO Sum_nCoef_max DO
92          L[i] := 0.0;
93      END_FOR;
94
95
96  END_FUNCTION
```

```
1  FUNCTION FC_Rellenar_Matrices_1 : VOID
2
3  // ********************************************************************* //
4
5  CONST
6      // constantes
7      nEnt_max := 3;
8      nPer_max := 3;
9      nSal_max := 3;
10
11     nCoef_max := 100;
12     Sum_nCoef_max := nCoef_max*nSal_max;
13
14     N2_max := 30;
15     Sum_N2_max := N2_max*nSal_max;
16
17     Nu_max := 5;
18     Sum_Nu_max := Nu_max*nEnt_max;
19
20     Dim_1 :=  Sum_Nu_max + nSal_max;
21     Dim_2 := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;
22
23 END_CONST
24
25 // ********************************************************************* //
26
27 VAR_INPUT
28     nEnt:    INT;
29     nPer:    INT;
30     nSal:    INT;
31 END_VAR
32
33 // ********************************************************************* //
34
35 VAR_IN_OUT
36
37     nCoef:  ARRAY [1..nSal_max] OF INT;
38
39     g_1:    ARRAY [1..Sum_nCoef_max, 1..nSal_max] OF REAL;
40     d_1:    ARRAY [1..Sum_nCoef_max, 1..nSal_max] OF REAL;
41
42     u0:     ARRAY [1..nEnt_max] OF REAL; // Punto en el que se obtuvo el modelo
43     y0:     ARRAY [1..nSal_max] OF REAL; // Punto en el que se obtuvo el modelo
44
45     N1:     ARRAY [1..nSal_max] OF INT;  // Inicio del horizonte de coincidencia
46     N2:     ARRAY [1..nSal_max] OF INT;  // Horizonte de prediccion
47     Nu:     ARRAY [1..nEnt_max] OF INT;  // Horizonte de control
48
49     D_sum:  ARRAY [1..Sum_N2_max,   1..nEnt_max]    OF REAL;
50
51     G:      ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;
52     G_g:    ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;
53
54
55 END_VAR
56
57 // ********************************************************************* //
58
59 VAR_TEMP
60
61     i,j,k,l:     INT;
62     ini, ini_d:  INT;
63
64     fil_Gij, fil_aux: INT;
65     col, col_Gij, col_aux: INT;
66
67     nFil_A, nFil_B, nCol_A, nCol_B: INT;
68
69     Sum_Nu, Sum_N2, Sum_N2_N1, Sum_nCoef: INT;
70
71 END_VAR
72
73 // ********************************************************************* //
74
75
76     // Área de instrucciones
77 BEGIN
78
79
80     (*************************************************************************)
```

```
81      (* Sumatorio de Nu, N2 y N2-N1-1                                          *)
82      (*************************************************************************)
83
84      Sum_Nu := 0;
85      FOR i := 1 TO nEnt DO
86          Sum_Nu := Sum_Nu + Nu[i];
87      END_FOR;
88
89      Sum_N2    := 0;
90      Sum_N2_N1 := 0;
91      Sum_nCoef := 0;
92      FOR i := 1 TO nSal DO
93          Sum_N2    := Sum_N2    + N2[i];
94          Sum_N2_N1 := Sum_N2_N1 + N2[i] - N1[i] + 1;
95          Sum_nCoef := Sum_nCoef + nCoef[i];
96      END_FOR;
97
98
99      (*************************************************************************)
100     (** Matriz D                                                             *)
101     (*   Va acumulando en cada posición vertical la suma de los elementos    *)
102     (*   anteriores más él mismo                                             *)
103     (*************************************************************************)
104
105     ini   := 0;
106     ini_d := 0;
107
108     FOR i := 1 TO nSal DO
109
110         FOR j := 1 TO nPer DO
111             D_sum[ini + 1, j] := d_1[ini_d + 1, j];
112         END_FOR;
113
114         FOR j := 2 TO N2[i] DO
115             IF j<nCoef[i] THEN
116                 FOR k := 1 TO nPer DO
117                     D_sum[ini + j, k] := D_sum[ini + j - 1, k] + d_1[ini_d + j, k];
118                 END_FOR;
119             ELSE
120                 FOR k := 1 TO nPer DO
121                     D_sum[ini + j, k] := D_sum[ini + j - 1, k] + d_1[ini_d + nCoef[i],
    k];
122                 END_FOR;
123             END_IF;
124         END_FOR;
125
126             ini   := ini + j - 1;
127             ini_d := ini_d + nCoef[i];
128
129     END_FOR;
130
131     (*************************************************************************)
132     (** Matriz G grande (Contiene todos los gij entre 1 y N2)                *)
133     (*      Aunque EN la función de costes sólo quiero que se tenga EN cuenta *)
134     (*      entre N1 y N2, para los límites quiero poder considerar todo N2   *)
135     (*************************************************************************)
136
137     ini := 0;         // Columna donde comienza la siguiente gij
138     fil_Gij := 0;     // Indice para las filas de la matriz G
139     fil_aux := 0;     // Indice para las filas de la submatriz Gij
140
141     FOR i := 1 TO nSal DO
142
143         fil_Gij := fil_Gij + fil_aux;     // Indice para las filas de la matriz G
144         fil_aux := N2[i];    // Indice para las filas de la submatriz Gij
145
146         col_Gij := 0;  // Indice para las columnas de la matriz G
147         col_aux := 0;  // Indice para las columnas de la submatriz Gij
148
149         FOR j := 1 TO nEnt DO
150
151             col_Gij := col_Gij + col_aux;
152             col_aux := Nu[j];
153
154             FOR k := 1 TO fil_aux DO
155                 FOR l := 1 TO col_aux DO
156
157                     col := (k-1) - (l-1) + 1;
158
159                     IF (col >= 1 AND col <= nCoef[i]) THEN
```

```
160                         G_g[fil_Gij+k, col_Gij+l] := g_1[ini + col, j];

161                     ELSIF (col < 1 ) THEN
162                         G_g[fil_Gij+k, col_Gij+l] := 0;
163                     ELSIF (col > nCoef[i] )THEN
164                         G_g[fil_Gij+k, col_Gij+l] := g_1[ini + nCoef[i], j];
165                     END_IF;
166
167                 END_FOR;
168             END_FOR;
169
170         END_FOR;
171
172         ini := ini + nCoef[i];
173
174     END_FOR;
175
176
177     (*************************************************************************)
178     (** Matriz G (Matriz G_g truncada desde N1)                              *)
179     (*************************************************************************)
180
181     ini := 0;         // Columna donde comienza la siguiente gij
182     fil_Gij := 0;     // Indice para las filas de la matriz G
183     fil_aux := 0;     // Indice para las filas de la submatriz Gij
184
185     FOR i := 1 TO nSal DO
186
187         fil_Gij := fil_Gij + fil_aux;
188         fil_aux := N2[i] - N1[i] + 1;     // Indice para las filas de la submatriz Gij
189
190         col_Gij := 0; // Indice para las columnas de la matriz G
191         col_aux := 0; // Indice para las columnas de la submatriz Gij
192
193         FOR j := 1 TO nEnt DO
194
195             col_Gij := col_Gij + col_aux;
196             col_aux := Nu[i];
197
198             FOR k := 1 TO fil_aux DO
199                 FOR l := 1 TO col_aux DO
200                     col := (k-1) + N1[i] - (l-1);
201                     IF (col >= 1 AND col <= nCoef[i]) THEN
202                         G[fil_Gij+k, col_Gij+l] := g_1[ini + col, j];

203                     ELSIF  (col < 1 ) THEN
204                         G[fil_Gij+k, col_Gij+l] := 0;
205                     ELSIF  (col > nCoef[i] ) THEN
206                         G[fil_Gij+k, col_Gij+l] := g_1[ini + nCoef[i], j];
207                     END_IF;
208                 END_FOR;
209             END_FOR;
210         END_FOR;
211
212         ini := ini + nCoef[i];
213
214     END_FOR;
215
216
217 END_FUNCTION
218
```

```
FUNCTION FC_Rellenar_Matrices_2 : VOID

// ********************************************************************* //
CONST
    // constantes
    nEnt_max := 3;
    nPer_max := 3;
    nSal_max := 3;

    nCoef_max := 100;
    Sum_nCoef_max := nCoef_max*nSal_max;

    N2_max := 30;
    Sum_N2_max := N2_max*nSal_max;

    Nu_max := 5;
    Sum_Nu_max := Nu_max*nEnt_max;

    Dim_1 := Sum_Nu_max + nSal_max;
    Dim_2 := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;

END_CONST
// ********************************************************************* //
VAR_INPUT
    nEnt:   INT;
    nSal:   INT;
END_VAR

// ********************************************************************* //

VAR_IN_OUT

    nCoef:  ARRAY [1..nSal_max] OF INT;

    N1:     ARRAY [1..nSal_max] OF INT;  // Inicio del horizonte de coincidencia
    N2:     ARRAY [1..nSal_max] OF INT;  // Horizonte de prediccion
    Nu:     ARRAY [1..nEnt_max] OF INT;  // Horizonte de control

    beta:   ARRAY [1..nEnt_max] OF REAL; // Peso de las acciones de control
    gamma:  ARRAY [1..nSal_max] OF REAL; // Peso del error de cada salida del sistema

    rho:    ARRAY [1..nSal_max] OF REAL; // Penalización de las variables de holgura

    G:      ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;

    H:      ARRAY [1..Dim_1,        1..Dim_1]       OF REAL;
    inv_H:  ARRAY [1..Dim_1,        1..Dim_1]       OF REAL;

    C:      ARRAY [1..Dim_1]                        OF REAL;
    C_aux:  ARRAY [1..Sum_Nu_max,   1..Sum_N2_max]  OF REAL;

    G_aux:  ARRAY [1..Sum_Nu_max, 1..Sum_N2_max]   OF REAL;
    H_aux:  ARRAY [1..Dim_1, 1..Dim_1]    OF REAL;


END_VAR

// ********************************************************************* //

VAR_TEMP

    i,j,k,l:     INT;
    ini, ini_d:  INT;

    fil_Gij, fil_aux: INT;
    col, col_Gij, col_aux: INT;

    nFil_A, nFil_B, nCol_A, nCol_B: INT;

    Sum_Nu, Sum_N2, Sum_N2_N1, Sum_nCoef: INT;

END_VAR

    // Área de instrucciones
BEGIN

    (*********************************************************************)
    (* Sumatorio de Nu, N2 y N2-N1-1                                     *)
    (*********************************************************************)

```

```
81      Sum_Nu := 0;
82      FOR i := 1 TO nEnt DO
83          Sum_Nu := Sum_Nu + Nu[i];
84      END_FOR;
85
86      Sum_N2    := 0;
87      Sum_N2_N1 := 0;
88      Sum_nCoef := 0;
89      FOR i := 1 TO nSal DO
90          Sum_N2    := Sum_N2    + N2[i];
91          Sum_N2_N1 := Sum_N2_N1 + N2[i] - N1[i] + 1;
92          Sum_nCoef := Sum_nCoef + nCoef[i];
93      END_FOR;
94
95      (*************************************************************************)
96      (** Matriz hessiana                                                     *)
97      (**      H_aux = G'*Gamma*G + Beta;                                     *)
98      (*************************************************************************)
99
100     // G'*Gamma - No es una multiplicacion normal de matrices
101
102     FOR i := 1 TO Sum_Nu DO
103        l := 0;
104        FOR j := 1 TO nSal DO
105           FOR k := 1 TO (N2[j]-N1[j]+1) DO
106               l := l+1;
107               G_aux[i,l] := G[l,i] * gamma[j];
108           END_FOR;
109        END_FOR;
110     END_FOR;
111
112     // [G'*Gamma]*G - Multiplicacion de matrices
113
114     nFil_A := Sum_Nu;
115     nCol_A := Sum_N2_N1;
116
117     nFil_B := Sum_N2_N1;
118     nCol_B := Sum_Nu;
119
120
121     FOR i := 1 TO nFil_A DO
122         FOR j := 1 TO nCol_B DO
123             FOR k :=1 TO nCol_A DO
124                 H_aux[i,j] := H_aux[i,j] + (G_aux[i,k]*G[k,j]);
125             END_FOR;
126         END_FOR;
127     END_FOR;
128
129     // Forzar que sea simetrico (H+H')/2 y anyadir variables de holgura
130
131     k := 1;
132     FOR i := 1 TO (Sum_Nu + nSal) DO
133         FOR j := 1 TO (Sum_Nu + nSal) DO
134             IF (i<=Sum_Nu AND j<=Sum_Nu ) THEN
135                 H[i,j] := (H_aux[i,j] + H_aux[j,i])/2;
136             ELSIF ( i=j ) THEN
137                 H[i,j] := rho[k];
138             ELSE
139                 H[i,j] := 0;
140             END_IF;
141         END_FOR;
142     END_FOR;
143
144     // G'*Gamma*G + Beta - Sumar a la matriz
145
146     ini := 0;
147     FOR i := 1 TO nEnt DO
148         FOR j := 1 TO Nu[i] DO
149             H[ini+j,ini+j] := H[ini+j,ini+j] + beta[i];
150         END_FOR;
151         ini := ini + Nu[i];
152     END_FOR;
153
154
155     (*************************************************************************)
156     (** Inversa de la Matriz Hessiana                                       *)
157     (*************************************************************************)
158
159     inv_H := H;
160
```

```
161     FOR i := 1 TO (Sum_Nu + nSal) DO
162
163         inv_H[i,i]:= -1/inv_H[i,i];
164
165         FOR fil_aux := 1 TO (Sum_Nu + nSal) DO
166             IF fil_aux <> i THEN
167                 inv_H[fil_aux,i]:= inv_H[fil_aux,i]*inv_H[i,i];
168             END_IF;
169         END_FOR;
170
171         FOR fil_aux := 1 TO (Sum_Nu + nSal) DO
172
173             IF fil_aux <> i THEN
174                 FOR col_aux := 1 TO (Sum_Nu + nSal) DO
175                     IF col_aux <> i THEN
176                         inv_H[fil_aux,col_aux]:=inv_H[fil_aux,col_aux]+inv_H[fil_aux,i
    ]*inv_H[i,col_aux];
177                     END_IF;
178                 END_FOR;
179             END_IF;
180         END_FOR;
181
182         FOR col_aux := 1 TO (Sum_Nu + nSal) DO            // Calculation of elements i
    n the Pivot-line.
183             IF col_aux <> i THEN
184                 inv_H[i,col_aux] := inv_H[i,col_aux]*inv_H[i,i];
185             END_IF;
186         END_FOR;
187     END_FOR;
188
189     FOR fil_aux := 1 TO (Sum_Nu + nSal) DO                // Convert all algebraic sig
    n of all Matrix elements.
190         FOR col_aux := 1 TO (Sum_Nu + nSal)DO
191             inv_H[fil_aux,col_aux] := -inv_H[fil_aux,col_aux];
192         END_FOR;
193     END_FOR;
194
195
196     (*************************************************************************)
197     (** Inicializacion de la resuesta libre                                  *)
198     (*************************************************************************)
199
200     // Como se inicializa a 0 no hace falta
201
202
203     (*************************************************************************)
204     (** Matriz auxiliar para el gradiente                                    *)
205     (*************************************************************************)
206
207     FOR i := 1 TO Sum_Nu DO
208         FOR j := 1 TO Sum_N2_N1 do
209             C_aux[i,j] := -G_aux[i,j];
210         END_FOR;
211     END_FOR;
212
213
214 END_FUNCTION
215
```

```
FUNCTION FC_Rellenar_Matrices_3 : VOID

// ********************************************************************** //

CONST
    // constantes
    nEnt_max := 3;
    nPer_max := 3;
    nSal_max := 3;

    nCoef_max := 100;
    Sum_nCoef_max := nCoef_max*nSal_max;

    N2_max := 30;
    Sum_N2_max := N2_max*nSal_max;

    Nu_max := 5;
    Sum_Nu_max := Nu_max*nEnt_max;

    Dim_1 := Sum_Nu_max + nSal_max;
    Dim_2 := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;

END_CONST

// ********************************************************************** //

VAR_INPUT
    nEnt:   INT;
    nSal:   INT;
END_VAR

// ********************************************************************** //

VAR_IN_OUT

    nCoef:  ARRAY [1..nSal_max] OF INT;

    N1:     ARRAY [1..nSal_max] OF INT;  // Inicio del horizonte de coincidencia
    N2:     ARRAY [1..nSal_max] OF INT;  // Horizonte de prediccion
    Nu:     ARRAY [1..nEnt_max] OF INT;  // Horizonte de control

    Du_Min: ARRAY [1..nEnt_max] OF REAL; // Limite superior en los incrementos
    Du_Max: ARRAY [1..nEnt_max] OF REAL; // Limite inferior en los incrementos

    u_Min:  ARRAY [1..nEnt_max] OF REAL; // Limite superior en las acciones de control
    u_Max:  ARRAY [1..nEnt_max] OF REAL; // Limite inferior en las acciones de control

    y_Min:  ARRAY [1..nSal_max] OF REAL; // Limite superior en las salidas del sistema
    y_Max:  ARRAY [1..nSal_max] OF REAL; // Limite inferior en las salidas del sistema

    G_g:    ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;

    A:      ARRAY [1..Dim_2,        1..Dim_1]       OF REAL;
    B:      ARRAY [1..Dim_2]                        OF REAL;


END_VAR

// ********************************************************************** //

VAR_TEMP

    i,j,k,l:    INT;
    ini:        INT;

    fil_Gij, fil_aux: INT;
    col, col_Gij, col_aux: INT;

    nFil_A, nFil_B, nCol_A, nCol_B: INT;

    Sum_Nu, Sum_N2, Sum_N2_N1, Sum_nCoef: INT;

END_VAR

    // Área de instrucciones
BEGIN


    (*********************************************************************)
    (* Sumatorio de Nu, N2 y N2-N1-1                                     *)
```

```
81      (*****************************************************************************)
82
83      Sum_Nu := 0;
84      FOR i := 1 TO nEnt DO
85          Sum_Nu := Sum_Nu + Nu[i];
86      END_FOR;
87
88      Sum_N2    := 0;
89      Sum_N2_N1 := 0;
90      Sum_nCoef := 0;
91      FOR i := 1 TO nSal DO
92          Sum_N2    := Sum_N2    + N2[i];
93          Sum_N2_N1 := Sum_N2_N1 + N2[i] - N1[i] + 1;
94          Sum_nCoef := Sum_nCoef + nCoef[i];
95      END_FOR;
96
97
98      (*****************************************************************************)
99      (** Matrices para las restricciones lineales                                 *)
100     (*****************************************************************************)
101
102     // Matriz A - Coefs restricciones lineales
103     //  - DuMax      [Sum_Nu, (Sum_Nu + nSal)]
104     //  - DuMin      [Sum_Nu, (Sum_Nu + nSal)]
105     //  - uMax       [Sum_Nu, (Sum_Nu + nSal)]
106     //  - uMin       [Sum_Nu, (Sum_Nu + nSal)]
107     //  - yMax       [Sum_N2, (Sum_Nu + nSal)]
108     //  - yMin       [Sum_N2, (Sum_Nu + nSal)]
109     //  - Epsilon    [nSal,   (Sum_Nu + nSal)]
110
111     col := Sum_Nu + nSal;
112
113     // ******************** Matriz Identidad I ******************** //
114
115     FOR i := 1 TO Sum_Nu DO
116         FOR j := 1 TO col DO
117             IF i=j THEN
118                 A[i,j] := 1;
119             ELSE
120                 A[i,j] := 0;
121             END_IF;
122         END_FOR;
123     END_FOR;
124
125     // ******************** Matriz Identidad -I ******************** //
126
127     ini := Sum_Nu;
128
129     FOR i := 1 TO Sum_Nu DO
130         FOR j := 1 TO col DO
131             IF i=j THEN
132                 A[ini+i,j] := -1;
133             ELSE
134                 A[ini+i,j] := 0;
135             END_IF;
136         END_FOR;
137     END_FOR;
138
139     // ************* Matriz de Diagonales Inferiores I ************ //
140
141     ini := ini + Sum_Nu;
142     col_aux := 0;
143
144     FOR i := 1 TO nEnt DO
145         FOR j := 1 TO nEnt DO
146             FOR k := 1 TO Nu[i] DO
147                 FOR l := 1 TO Nu[j]+nSal DO
148                     IF (i=j AND k>=l) THEN
149                         A[ini+k, col_aux+l] := 1;
150                     ELSE
151                         A[ini+k, col_aux+l] := 0;
152                     END_IF;
153                 END_FOR;
154             END_FOR;
155             col_aux := col_aux + Nu[j];
156         END_FOR;
157         col_aux := 0;
158         ini := ini + Nu[i];
159     END_FOR;
160
```

```
161     // ************ Matriz de Diagonales Inferiores -I ************ //
162
163     col_aux := 0;
164
165     FOR i := 1 TO nEnt DO
166         FOR j := 1 TO nEnt DO
167             FOR k := 1 TO Nu[i] DO
168                 FOR l := 1 TO Nu[j]+nSal DO
169                     IF (i=j AND k>=l) THEN
170                         A[ini+k, col_aux+l] := -1;
171                     ELSE
172                         A[ini+k, col_aux+l] := 0;
173                     END_IF;
174                 END_FOR;
175             END_FOR;
176             col_aux := col_aux + Nu[j];
177         END_FOR;
178         col_aux := 0;
179         ini := ini + Nu[i];
180     END_FOR;
181
182
183     // ************ Matriz G_g para restricciones en y ************ //
184
185     FOR i := 1 TO Sum_N2 DO             // Matriz G
186         FOR j := 1 TO Sum_Nu DO
187             A[ini+i,j] := G_g[i,j];
188         END_FOR;
189     END_FOR;
190
191     FOR i := 1 TO nSal DO               // A la dcha de G
192         FOR j := 1 TO nSal DO
193             FOR k := 1 TO N2[j] DO
194                 IF i=j THEN
195                     A[ini+k, Sum_Nu+i] := -1;
196                 ELSE
197                     A[ini+k, Sum_Nu+i] :=  0;
198                 END_IF;
199             END_FOR;
200             ini := ini + k -1;
201         END_FOR;
202         ini := ini - Sum_N2;
203     END_FOR;
204
205
206     ini := ini + Sum_N2;
207
208     // ************ Matriz -G_g para restricciones en y ************ //
209
210     FOR i := 1 TO Sum_N2 DO            // Matriz -G
211         FOR j := 1 TO Sum_Nu DO
212             A[ini+i,j] := -G_g[i,j];
213         END_FOR;
214     END_FOR;
215
216     FOR i := 1 TO nSal DO               // A la dcha de -G
217         FOR j := 1 TO nSal DO
218             FOR k := 1 TO N2[j] DO
219                 IF i=j THEN
220                     A[ini+k, Sum_Nu+i] := -1;
221                 ELSE
222                     A[ini+k, Sum_Nu+i] :=  0;
223                 END_IF;
224             END_FOR;
225             ini := ini + k - 1;
226         END_FOR;
227         ini := ini - Sum_N2;
228     END_FOR;
229
230     ini := ini + Sum_N2;
231
232
233     // **** Variables de holgura para restricciones blandas **** //
234
235     FOR i := 1 TO nSal DO
236         FOR j := 1 TO Sum_Nu DO
237             A[ini+i,j] := 0;
238         END_FOR;
239     END_FOR;
240
```

```
241     FOR i := 1 TO nSal DO
242         FOR j := 1 TO nSal DO
243             IF i=j THEN
244                 A[ini+i,Sum_Nu+j] := -1;
245             END_IF;
246         END_FOR;
247     END_FOR;
248
249
250     // Matriz b - Termino indpte de las restricciones lineales
251     //  - DuMax [Sum_Nu, 1]  - No cambia
252     //  - DuMin [Sum_Nu, 1]  - No cambia
253     //  - uMax  [Sum_Nu, 1]
254     //  - uMin  [Sum_Nu, 1]
255     //  - yMax  [Sum_N2_N1, 1]
256     //  - yMin  [Sum_N2_N1, 1]
257     //  - Epsilon [nSal, 1]
258
259
260     // Limites en los incrementos
261
262     ini := 0;                                                    // Du máximo
263     FOR i := 1 TO nEnt DO
264        FOR j := 1 TO Nu[i] DO
265            B[ini+j] := Du_Max[i];
266        END_FOR;
267        ini := ini + j -1;
268     END_FOR;
269
270     FOR i := 1 TO nEnt DO
271        FOR j := 1 TO Nu[i] DO                              // Du mínimo
272            B[ini+j] := -Du_Min[i];
273        END_FOR;
274        ini := ini + j -1;
275     END_FOR;
276
277
278
279 END_FUNCTION
280
```

```
FUNCTION FC_Rellenar_Matrices_4 : VOID

// ********************************************************************** //

CONST
    // constantes
    nEnt_max := 3;
    nPer_max := 3;
    nSal_max := 3;

    nCoef_max := 100;
    Sum_nCoef_max := nCoef_max*nSal_max;

    N2_max := 30;
    Sum_N2_max := N2_max*nSal_max;

    Nu_max := 5;
    Sum_Nu_max := Nu_max*nEnt_max;

    Dim_A := Sum_Nu_max + nSal_max;
    Dim_B := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;

END_CONST

// ********************************************************************** //

VAR_INPUT

    Dim_1:      INT;    // Numero de columnas de la matriz A
    Dim_2:      INT;    // Numero de filas de la matriz A

END_VAR

// ********************************************************************** //

VAR_IN_OUT

    A:      ARRAY [1..Dim_B,        1..Dim_A]       OF REAL;
    invH:   ARRAY [1..Dim_A, 1..Dim_A]  OF REAL;

END_VAR

// ********************************************************************** //

VAR_TEMP

    i,j,k,l:    INT;
    ini:        INT;

    fil_Gij, fil_aux: INT;
    col, col_Gij, col_aux: INT;

    nFil_A, nFil_B, nCol_A, nCol_B: INT;

    Sum_Nu, Sum_N2, Sum_N2_N1, Sum_nCoef: INT;

END_VAR

    // Área de instrucciones
BEGIN


    (*************************************************************************)
    (* Calcular P                                                            *)
    (*     P = A_cons*(H_1*A_cons')                                          *)
    (*************************************************************************)

    (*************************************************************************)
    (*      aux_2 = H_1*A'                                                   *)
    (*************************************************************************)

    nFil_A := Dim_1;
    nCol_A := Dim_1;

    nFil_B := Dim_1;
    nCol_B := Dim_2;


    FOR i := 1 TO nFil_A DO
        FOR j := 1 TO nCol_B DO
```

```
81              DB_aux.aux_2[i,j] := 0;
82              FOR k :=1 TO nCol_A DO
83                  // Se han cambiado los indices de la matriz A para que sea la traspues
    ta
84                  DB_aux.aux_2[i,j] := DB_aux.aux_2[i,j] + (invH[i,k]*A[j,k]);
85              END_FOR;
86          END_FOR;
87      END_FOR;
88
89
90      (*************************************************************************)
91      (*      P := A*aux_2                                                     *)
92      (*************************************************************************)
93
94      nFil_A := Dim_2;
95      nCol_A := Dim_1;
96
97      nFil_B := Dim_1;
98      nCol_B := Dim_2;
99
100
101     FOR i := 1 TO nFil_A DO
102         IF i<=65 THEN
103             FOR j := 1 TO nCol_B DO
104                 DB_QP_P_1.P[i,j] := 0;
105                 FOR k :=1 TO nCol_A DO
106                     DB_QP_P_1.P[i,j] := DB_QP_P_1.P[i,j] + (A[i,k]*DB_aux.aux_2[k,j]);
107                 END_FOR;
108             END_FOR;
109         ELSIF i<=130 THEN
110             FOR j := 1 TO nCol_B DO
111                 DB_QP_P_2.P[i-65,j] := 0;
112                 FOR k :=1 TO nCol_A DO
113                     DB_QP_P_2.P[i-65,j] := DB_QP_P_2.P[i-65,j] + (A[i,k]*DB_aux.aux_2[
    k,j]);
114                 END_FOR;
115             END_FOR;
116         ELSIF i<=195 THEN
117             FOR j := 1 TO nCol_B DO
118                 DB_QP_P_3.P[i-130,j] := 0;
119                 FOR k :=1 TO nCol_A DO
120                     DB_QP_P_3.P[i-130,j] := DB_QP_P_3.P[i-130,j] + (A[i,k]*DB_aux.aux_
    2[k,j]);
121                 END_FOR;
122             END_FOR;
123         ELSIF i<=243 THEN
124             FOR j := 1 TO nCol_B DO
125                 DB_QP_P_4.P[i-195,j] := 0;
126                 FOR k :=1 TO nCol_A DO
127                     DB_QP_P_4.P[i-195,j] := DB_QP_P_4.P[i-195,j] + (A[i,k]*DB_aux.aux_
    2[k,j]);
128                 END_FOR;
129             END_FOR;
130         END_IF;
131     END_FOR;
132
133
134 END_FUNCTION
135
136
```

```
FUNCTION_BLOCK FB_DMC

// ********************************************************************** //
CONST
    // constantes
    nEnt_max := 3;
    nPer_max := 3;
    nSal_max := 3;

    nCoef_max := 100;
    Sum_nCoef_max := nCoef_max*nSal_max;

    N2_max := 30;
    Sum_N2_max := N2_max*nSal_max;

    Nu_max := 5;
    Sum_Nu_max := Nu_max*nEnt_max;

    Dim_1 := Sum_Nu_max + nSal_max;
    Dim_2 := 4*Sum_Nu_max + 2*Sum_N2_max + nSal_max;

END_CONST

// ********************************************************************** //
VAR_INPUT
    // variables de entrada
    flag:   BOOL;
END_VAR
// ********************************************************************** //
VAR_OUTPUT
    // variables de salida
END_VAR

// ********************************************************************** //
VAR_IN_OUT
    // variables de entrada/salida

END_VAR
// ********************************************************************** //
VAR_TEMP
    // variables temporales
    i,j,k: INT;

    nSample: INT;

    ini, ini_T: INT;

    nFil_A, nCol_A, nFil_B, nCol_B: INT;

END_VAR
// ********************************************************************** //
VAR
    // variables estáticas

    // ********************************************************** //
    // ** Identificacion del sistema ***************************** //
    // ********************************************************** //
    nEnt:   INT;
    nPer:   INT;
    nSal:   INT;

    nCoef:  ARRAY [1..nSal_max] OF INT;

    g_1:    ARRAY [1..Sum_nCoef_max, 1..nSal_max] OF REAL;
    d_1:    ARRAY [1..Sum_nCoef_max, 1..nSal_max] OF REAL;

    u0:     ARRAY [1..nEnt_max] OF REAL;
    v0:     ARRAY [1..nPer_max] OF REAL;
    y0:     ARRAY [1..nSal_max] OF REAL;

    // ********************************************************** //
    // ** Parámetros propios del DMC ***************************** //
    // ********************************************************** //

    N1:     ARRAY [1..nSal_max] OF INT; // Inicio del horizonte de coincidencia
    N2:     ARRAY [1..nSal_max] OF INT; // Horizonte de prediccion
    Nu:     ARRAY [1..nEnt_max] OF INT; // Horizonte de control

    alpha:  ARRAY [1..nSal_max] OF REAL; // Polo discreto para trayectoria de referenc
ia
```

```
80
81      beta:   ARRAY [1..nEnt_max] OF REAL; // Peso de las acciones de control
82      gamma:  ARRAY [1..nSal_max] OF REAL; // Peso del error de cada salida del sistema
83
84
85      rho:    ARRAY [1..nSal_max] OF REAL; // Penalización de las variables de holgura
86
87      Du_Min: ARRAY [1..nEnt_max] OF REAL; // Limite superior en los incrementos
88      Du_Max: ARRAY [1..nEnt_max] OF REAL; // Limite inferior en los incrementos
89
90      u_Min:  ARRAY [1..nEnt_max] OF REAL; // Limite superior en las acciones de control
91      u_Max:  ARRAY [1..nEnt_max] OF REAL; // Limite inferior en las acciones de control
92
93      y_Min:  ARRAY [1..nSal_max] OF REAL; // Limite superior en las salidas del sistema
94      y_Max:  ARRAY [1..nSal_max] OF REAL; // Limite inferior en las salidas del sistema
95
96      // ********************************************************** //
97      // ** Matrices del DMC ************************************** //
98      // ********************************************************** //
99
100     Sum_N2:     INT;
101     Sum_N2_N1:  INT;
102     Sum_Nu:     INT;
103     Sum_nCoef:  INT;
104
105     D_sum:  ARRAY [1..Sum_N2_max,   1..nEnt_max]    OF REAL;
106
107     G:      ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;
108     G_g:    ARRAY [1..Sum_N2_max,   1..Sum_Nu_max]  OF REAL;
109
110     A:      ARRAY [1..Dim_2,        1..Dim_1]       OF REAL;
111     B:      ARRAY [1..Dim_2]                        OF REAL;
112
113     H:      ARRAY [1..Dim_1,        1..Dim_1]       OF REAL;
114     inv_H:  ARRAY [1..Dim_1,        1..Dim_1]       OF REAL;
115
116     C:      ARRAY [1..Dim_1]                        OF REAL;
117     C_aux:  ARRAY [1..Sum_Nu_max,   1..Sum_N2_max]  OF REAL;
118
119     L:      ARRAY [1..Sum_nCoef_max]                OF REAL;
120
121     u_old: ARRAY [1..nEnt_max] OF REAL;
122     v_old: ARRAY [1..nPer_max] OF REAL;
123
124
125     // ********************************************************** //
126     // ** Variables auxiliares ********************************** //
127     // ********************************************************** //
128
129     flag_ini:   BOOL := TRUE;
130
131     G_aux:  ARRAY [1..Sum_Nu_max, 1..Sum_N2_max]    OF REAL;
132     H_aux:  ARRAY [1..Dim_1, 1..Dim_1]              OF REAL;
133
134
135     // Estas despues son temporales
136
137     Du: ARRAY [1..nEnt_max] OF REAL;
138     Dv: ARRAY [1..nPer_max] OF REAL;
139
140     T:          ARRAY [1..Sum_N2_max] OF REAL; // [Sum_N2]
141     R:          ARRAY [1..Sum_N2_max] OF REAL; // [Sum_N2]
142     D:          ARRAY [1..Sum_N2_max] OF REAL; // [Sum_N2_N1]
143
144     y_ant:      REAL;
145
146     // Estas despues son in/out
147     u:      ARRAY [1..nEnt_max] OF REAL;
148     v:      ARRAY [1..nPer_max] OF REAL;
149     y:      ARRAY [1..nSal_max] OF REAL;
150     ref:    ARRAY [1..nSal_max] OF REAL;
151
152     // ********************************************************** //
153     // ** Salida del optimizador ******************************** //
154     // ********************************************************** //
155
156     nIter:  INT;
157     Sol:    ARRAY [1..Dim_1]  OF REAL;
158
159     Du_new: ARRAY [1..nEnt_max] OF REAL;
```

```
160
161 END_VAR
162
163 // ********************************************************************* //
164
165 BEGIN
166
167     IF (flag=TRUE AND flag_ini=TRUE) THEN
168
169         FC_Inicializar_Ctes ( nEnt:=nEnt, nSal:=nSal, Sum_N2:=Sum_N2, Sum_N2_N1:=Sum_N
    2_N1, Sum_Nu:=Sum_Nu, Sum_nCoef:=Sum_nCoef, nCoef:=nCoef, u0:=u0, y0:=y0, N1:=N1, N2:=
    N2, Nu:=Nu, beta:=beta, gamma:=gamma );
170         FC_Vaciar_Matrices ( D_sum:=D_sum, G:=G, G_g:=G_g, A:=A, B:=B, H:=H, inv_H:=in
    v_H, C:=C, C_aux:=C_aux, L:=L );
171         FC_Rellenar_Matrices_1 ( nEnt:=nEnt, nPer:=nPer, nSal:=nSal, nCoef:=nCoef, g_1
    :=g_1, d_1:=d_1, u0:=u0, y0:=y0, N1:=N1, N2:=N2, Nu:=Nu, D_sum:=D_sum, G:=G, G_g:=G_g)
    ;
172         FC_Rellenar_Matrices_2 ( nEnt:=nEnt, nSal:=nSal, nCoef:=nCoef, N1:=N1, N2:=N2,
     Nu:=Nu, beta:=beta, gamma:=gamma, rho:=rho, G:=G, H:=H, inv_H:=inv_H, C:=C, C_aux:=C_
    aux, G_aux:=G_aux, H_aux:=H_aux);
173         FC_Rellenar_Matrices_3 ( nEnt:=nEnt, nSal:=nSal, nCoef:=nCoef, N1:=N1, N2:=N2,
     Nu:=Nu, Du_Min:=Du_Min, Du_Max:=Du_Max, u_Min:=u_Min, u_Max:=u_Max, y_Min:=y_Min, y_M
    ax:=y_Max, G_g:=G_g, A:=A, B:=B );
174         FC_Rellenar_Matrices_4 ( Dim_1:=(Sum_Nu + nSal), Dim_2:=(4*Sum_Nu + 2*Sum_N2 +
     nSal), A:=A, invH:=inv_H );
175
176         flag_ini := FALSE;
177
178     ELSE
179
180     (*************************************************************************)
181     (**                          Regulador DMC                              **)
182     (*************************************************************************)
183
184     (*****************************)
185
186     FOR i:=1 TO nEnt DO
187         Du[i] := u[i]-u_old[i];
188         //u_old[i] := u[i];
189     END_FOR;
190
191     FOR i:=1 TO nPer DO
192         Dv[i] := v[i]-v_old[i];
193         //v_old[i] := v[i];
194     END_FOR;
195
196
197     (*************************************************************************)
198     (* Actualizar la respuesta libre                                         *)
199     (*      IF nPer>0 THEN                                                   *)
200     (*          L = L + g*Du + d*Dv;                                         *)
201     (*      ELSE                                                             *)
202     (*          L = L + g*Du;                                                *)
203     (*      END_IF;                                                          *)
204     (*************************************************************************)
205
206     // Desplazar hacia abajo una posicion
207
208     ini := 0;
209     FOR j := 1 TO nSal DO
210         FOR k := 2 TO nCoef[j] DO
211             L[ini+k-1] := L[ini+k];
212         END_FOR;
213         ini := ini + nCoef[j];
214     END_FOR;
215
216     // g*Du - Multiplicacion de matrices
217
218     nFil_A := Sum_nCoef;
219     nCol_A := nEnt;
220
221     nFil_B := nEnt;
222     nCol_B := 1;
223
224
225     FOR i := 1 TO nFil_A DO
226         // No inicializamos de nuevo L y sumamos el efecto de las nuevas entradas
227         FOR j :=1 TO nCol_A DO
228             L[i] := L[i] + (g_1[i,j]*Du[j]);
229         END_FOR;
```

```
230     END_FOR;
231
232     // L + d*Dv - Multiplicacion de matrices - Solo si hay perturbaciones medibles
233
234     IF nPer>0 THEN
235
236         nFil_A := Sum_nCoef;
237         nCol_A := nPer;
238
239         nFil_B := nPer;
240         nCol_B := 1;
241
242         FOR i := 1 TO nFil_A DO
243             // No inicializamos de nuevo L y sumamos el efecto de las nuevas pertubaci
    ones
244             FOR j :=1 TO nCol_A DO
245                 L[i] := L[i] + (d_1[i,j]*Dv[j]);
246             END_FOR;
247         END_FOR;
248
249     END_IF;
250
251     // Calcular la salida predicha
252
253     ini   := 0;
254     ini_T := 0;
255     FOR j := 1 TO nSal DO
256         FOR k := 2 TO N2[j]+1 DO
257             IF k <= nCoef[j] THEN
258                 T[ini_T+k-1] := L[ini + k]        + y[j] - L[ini + 1];
259             ELSE
260                 T[ini_T+k-1] := L[ini + nCoef[j]] + y[j] - L[ini + 1];
261             END_IF;
262         END_FOR;
263         ini   := ini   + nCoef[j] ;
264         ini_T := ini_T + N2[j];
265     END_FOR;
266
267     // Si hay perturbaciones medibles se suma su efecto considerando que Dv(t+1)=Dv(t)
    =cte
268
269     IF nPer>0 THEN
270
271         nFil_A := Sum_N2;
272         nCol_A := nPer;
273
274         nFil_B := nPer;
275         nCol_B := 1;
276
277         FOR i := 1 TO nFil_A DO
278             FOR k :=1 TO nCol_A DO
279                 T[i] := T[i] + (D_sum[i,k]*Dv[k]);
280             END_FOR;
281         END_FOR;
282
283     END_IF;
284
285     // Filtrar la referencia para que tenga dinamica
286
287     ini   := 0;
288     FOR j := 1 TO nSal DO
289         y_ant := y[j];
290         FOR k := 1 TO N2[j] DO
291             R[ini + k] := (1-alpha[j])*ref[j] + alpha[j]*y_ant;
292             y_ant := R[ini + k];
293         END_FOR;
294         ini := ini + N2[j] ;
295     END_FOR;
296
297
298    // No se coge el error de toda la respuesta libre, sólo entre N1 y N2
299
300     k   := 0;
301     ini := 0;
302     FOR i := 1 TO nSal DO
303         FOR j := N1[i] TO N2[i] DO
304             k := k+1;
305             D[k] := R[ini+j] - T[ini+j] ;
306         END_FOR;
307         ini := ini + N2[i];
```

```
308     END_FOR;
309
310
311     (*************************************************************************)
312     (** Gradiente                                                            *)
313     (*      - C := [Caux*D;0;0;0];                                           *)
314     (*************************************************************************)
315
316     nFil_A := Sum_Nu;
317     nCol_A := Sum_N2_N1;
318
319     nFil_B := Sum_N2_N1;
320     nCol_B := 1;
321
322     FOR i := 1 TO nFil_A DO
323         C[i] := 0.0;
324         FOR k :=1 TO nCol_A DO
325             C[i] := C[i] + (C_aux[i,k]*D[k]);
326         END_FOR;
327     END_FOR;
328
329     FOR i := 1 TO nSal DO
330         C[Sum_Nu+i] := 0.0;
331     END_FOR;
332
333
334     (*************************************************************************)
335     (** Restricciones                                                        *)
336     (**      - Solo varia el termino independiente B                       *)
337     (*************************************************************************)
338
339     ini := 2*Sum_Nu;
340     FOR j := 1 TO nEnt DO
341        FOR k := 1 TO Nu[j] DO
342            B[ini+k] :=  u_Max[j] - u[j];
343        END_FOR;
344        ini := ini + k - 1;
345     END_FOR;
346
347     FOR j := 1 TO nEnt DO
348        FOR k := 1 TO Nu[j] DO
349            B[ini+k] := -u_Min[j] + u[j];
350        END_FOR;
351        ini := ini + k - 1;
352     END_FOR;
353
354     ini_T := 0;
355     FOR j := 1 TO nSal DO
356        FOR k := 1 TO N2[j] DO
357            B[ini+k] := y_Max[j] - T[ini_T + k];
358        END_FOR;
359        ini := ini + k - 1;
360        ini_T := ini_T + k - 1;
361     END_FOR;
362
363     ini_T := 0;
364     FOR j := 1 TO nSal DO
365        FOR k := 1 TO N2[j] DO
366            B[ini+k] := -y_Min[j] + T[ini_T + k];
367        END_FOR;
368        ini := ini + k - 1;
369        ini_T := ini_T + k - 1;
370     END_FOR;
371
372     FOR j := 1 TO nSal DO
373        B[ini+j] := 0.0;
374     END_FOR;
375
376
377
378     (*************************************************************************)
379     (** Llamada al optimizador                                               *)
380     (*************************************************************************)
381
382     FC_Optimizador_QP( MaxIter:=15, Dim_1:=(Sum_Nu + nSal), Dim_2:=(4*Sum_Nu + 2*Sum_N
    2 + nSal), nIter:=nIter, A:=A, B:=B, C:=C, invH:=inv_H, Sol:=Sol );
383
384     ini := 0;
385     FOR i:=1 TO nEnt DO
386         Du_new[i] := Sol[ini+1];
```

```
387          ini := ini+Nu[i];
388      END_FOR;
389
390      END_IF;
391
392  END_FUNCTION_BLOCK
393
```

```
DATA_BLOCK DB_Reactor_DMC FB_DMC

BEGIN

    (* Esto es solo para pruebas *)
    ref[1]:= 7.2;           ref[2]:= 60.0;          ref[3]:=0.8;
    u[1] := 26.665115472;   u[2] := 25.822517;      u[3] := 0.0;
    v[1] := 8.0;            v[2] := 31.0;           v[3] := 20.0;
    y[1] := 7.2;            y[2] := 60.0;           y[3] := 0.8;

    (* ******************************************************* *)
    (* Modelo de Respuesta Salto del Sistema                   *)
    (* ******************************************************* *)

    (* Número de parámetros del sistema *)
    nEnt := 2;
    nPer := 3;
    nSal := 3;

    (* Número de coeficientes de la respuesta salto *)
    nCoef[1] := 104;
    nCoef[2] := 58;
    nCoef[3] := 104;

    (* Valores iniciales de las variables *)

    // Entradas
    u0[1] := 26.6665978;    u_old[1] := 26.6665978;
    u0[2] := 25.822517;     u_old[2] := 25.822517;
    u0[3] := 0.0;           u_old[3] := 0.0;

    // Perturbaciones
    v0[1] := 8.0;           v_old[1] := 8.0;
    v0[2] := 30.0;          v_old[2] := 30.0;
    v0[3] := 20.0;          v_old[3] := 20.0;

    // Salidas
    y0[1] := 7.2;
    y0[2] := 60.0;
    y0[3] := 0.8;

   (* Coeficientes de la respuesta salto - Entrada/Salida - g_1[nCoef(i)*nSal, nEnt]*
)
    g_1[1,1] := -0.036839;
    g_1[1,2] := -0.00024847;
    g_1[2,1] := -0.038374;
    g_1[2,2] := -0.00056707;
    g_1[3,1] := -0.036962;
    g_1[3,2] := -0.00089732;
    g_1[4,1] := -0.035427;
    g_1[4,2] := -0.0012244;
    g_1[5,1] := -0.034005;
    g_1[5,2] := -0.0015445;
    g_1[6,1] := -0.032706;
    g_1[6,2] := -0.0018562;
    g_1[7,1] := -0.03152;
    g_1[7,2] := -0.0021588;
    g_1[8,1] := -0.030439;
    g_1[8,2] := -0.0024521;
    g_1[9,1] := -0.029452;
    g_1[9,2] := -0.0027357;
    g_1[10,1] := -0.028552;
    g_1[10,2] := -0.0030093;
    g_1[11,1] := -0.027731;
    g_1[11,2] := -0.003273;
    g_1[12,1] := -0.026982;
    g_1[12,2] := -0.0035267;
    g_1[13,1] := -0.026298;
    g_1[13,2] := -0.0037703;
    g_1[14,1] := -0.025675;
    g_1[14,2] := -0.004004;
    g_1[15,1] := -0.025106;
    g_1[15,2] := -0.004228;
    g_1[16,1] := -0.024587;
    g_1[16,2] := -0.0044423;
    g_1[17,1] := -0.024113;
    g_1[17,2] := -0.0046472;
    g_1[18,1] := -0.023681;
    g_1[18,2] := -0.0048429;
    g_1[19,1] := -0.023287;
```

```
 80      g_1[19,2] := -0.0050296;
 81      g_1[20,1] := -0.022928;
 82      g_1[20,2] := -0.0052075;
 83      g_1[21,1] := -0.0226;
 84      g_1[21,2] := -0.005377;
 85      g_1[22,1] := -0.0223;
 86      g_1[22,2] := -0.0055384;
 87      g_1[23,1] := -0.022027;
 88      g_1[23,2] := -0.0056918;
 89      g_1[24,1] := -0.021778;
 90      g_1[24,2] := -0.0058376;
 91      g_1[25,1] := -0.021551;
 92      g_1[25,2] := -0.0059761;
 93      g_1[26,1] := -0.021344;
 94      g_1[26,2] := -0.0061075;
 95      g_1[27,1] := -0.021155;
 96      g_1[27,2] := -0.0062322;
 97      g_1[28,1] := -0.020982;
 98      g_1[28,2] := -0.0063504;
 99      g_1[29,1] := -0.020825;
100      g_1[29,2] := -0.0064624;
101      g_1[30,1] := -0.020681;
102      g_1[30,2] := -0.0065685;
103      g_1[31,1] := -0.02055;
104      g_1[31,2] := -0.0066689;
105      g_1[32,1] := -0.020431;
106      g_1[32,2] := -0.0067639;
107      g_1[33,1] := -0.020321;
108      g_1[33,2] := -0.0068537;
109      g_1[34,1] := -0.020222;
110      g_1[34,2] := -0.0069386;
111      g_1[35,1] := -0.020131;
112      g_1[35,2] := -0.0070189;
113      g_1[36,1] := -0.020048;
114      g_1[36,2] := -0.0070947;
115      g_1[37,1] := -0.019973;
116      g_1[37,2] := -0.0071662;
117      g_1[38,1] := -0.019904;
118      g_1[38,2] := -0.0072338;
119      g_1[39,1] := -0.019841;
120      g_1[39,2] := -0.0072975;
121      g_1[40,1] := -0.019784;
122      g_1[40,2] := -0.0073576;
123      g_1[41,1] := -0.019731;
124      g_1[41,2] := -0.0074143;
125      g_1[42,1] := -0.019684;
126      g_1[42,2] := -0.0074678;
127      g_1[43,1] := -0.01964;
128      g_1[43,2] := -0.0075181;
129      g_1[44,1] := -0.0196;
130      g_1[44,2] := -0.0075656;
131      g_1[45,1] := -0.019564;
132      g_1[45,2] := -0.0076102;
133      g_1[46,1] := -0.019531;
134      g_1[46,2] := -0.0076523;
135      g_1[47,1] := -0.019501;
136      g_1[47,2] := -0.0076919;
137      g_1[48,1] := -0.019473;
138      g_1[48,2] := -0.0077292;
139      g_1[49,1] := -0.019448;
140      g_1[49,2] := -0.0077643;
141      g_1[50,1] := -0.019425;
142      g_1[50,2] := -0.0077973;
143      g_1[51,1] := -0.019404;
144      g_1[51,2] := -0.0078283;
145      g_1[52,1] := -0.019385;
146      g_1[52,2] := -0.0078575;
147      g_1[53,1] := -0.019368;
148      g_1[53,2] := -0.0078849;
149      g_1[54,1] := -0.019352;
150      g_1[54,2] := -0.0079107;
151      g_1[55,1] := -0.019338;
152      g_1[55,2] := -0.007935;
153      g_1[56,1] := -0.019324;
154      g_1[56,2] := -0.0079577;
155      g_1[57,1] := -0.019312;
156      g_1[57,2] := -0.0079791;
157      g_1[58,1] := -0.019301;
158      g_1[58,2] := -0.0079992;
159      g_1[59,1] := -0.019291;
```

```
160     g_1[59,2] := -0.0080181;
161     g_1[60,1] := -0.019282;
162     g_1[60,2] := -0.0080358;
163     g_1[61,1] := -0.019274;
164     g_1[61,2] := -0.0080524;
165     g_1[62,1] := -0.019266;
166     g_1[62,2] := -0.0080681;
167     g_1[63,1] := -0.019259;
168     g_1[63,2] := -0.0080827;
169     g_1[64,1] := -0.019253;
170     g_1[64,2] := -0.0080965;
171     g_1[65,1] := -0.019247;
172     g_1[65,2] := -0.0081094;
173     g_1[66,1] := -0.019242;
174     g_1[66,2] := -0.0081215;
175     g_1[67,1] := -0.019237;
176     g_1[67,2] := -0.0081328;
177     g_1[68,1] := -0.019233;
178     g_1[68,2] := -0.0081435;
179     g_1[69,1] := -0.019229;
180     g_1[69,2] := -0.0081535;
181     g_1[70,1] := -0.019225;
182     g_1[70,2] := -0.0081628;
183     g_1[71,1] := -0.019222;
184     g_1[71,2] := -0.0081716;
185     g_1[72,1] := -0.019219;
186     g_1[72,2] := -0.0081798;
187     g_1[73,1] := -0.019216;
188     g_1[73,2] := -0.0081875;
189     g_1[74,1] := -0.019213;
190     g_1[74,2] := -0.0081948;
191     g_1[75,1] := -0.019211;
192     g_1[75,2] := -0.0082016;
193     g_1[76,1] := -0.019209;
194     g_1[76,2] := -0.0082079;
195     g_1[77,1] := -0.019207;
196     g_1[77,2] := -0.0082139;
197     g_1[78,1] := -0.019205;
198     g_1[78,2] := -0.0082194;
199     g_1[79,1] := -0.019204;
200     g_1[79,2] := -0.0082247;
201     g_1[80,1] := -0.019202;
202     g_1[80,2] := -0.0082295;
203     g_1[81,1] := -0.019201;
204     g_1[81,2] := -0.0082341;
205     g_1[82,1] := -0.019201;
206     g_1[82,2] := -0.0082384;
207     g_1[83,1] := -0.019201;
208     g_1[83,2] := -0.0082424;
209     g_1[84,1] := -0.019201;
210     g_1[84,2] := -0.0082462;
211     g_1[85,1] := -0.019201;
212     g_1[85,2] := -0.0082497;
213     g_1[86,1] := -0.019201;
214     g_1[86,2] := -0.008253;
215     g_1[87,1] := -0.019201;
216     g_1[87,2] := -0.0082561;
217     g_1[88,1] := -0.019201;
218     g_1[88,2] := -0.008259;
219     g_1[89,1] := -0.019201;
220     g_1[89,2] := -0.0082617;
221     g_1[90,1] := -0.019201;
222     g_1[90,2] := -0.0082642;
223     g_1[91,1] := -0.019201;
224     g_1[91,2] := -0.0082666;
225     g_1[92,1] := -0.019201;
226     g_1[92,2] := -0.0082688;
227     g_1[93,1] := -0.019201;
228     g_1[93,2] := -0.0082709;
229     g_1[94,1] := -0.019201;
230     g_1[94,2] := -0.0082729;
231     g_1[95,1] := -0.019201;
232     g_1[95,2] := -0.0082747;
233     g_1[96,1] := -0.019201;
234     g_1[96,2] := -0.0082764;
235     g_1[97,1] := -0.019201;
236     g_1[97,2] := -0.008278;
237     g_1[98,1] := -0.019201;
238     g_1[98,2] := -0.0082795;
239     g_1[99,1] := -0.019201;
```

```
240     g_1[99,2] := -0.0082809;
241     g_1[100,1] := -0.019201;
242     g_1[100,2] := -0.0082822;
243     g_1[101,1] := -0.019201;
244     g_1[101,2] := -0.0082834;
245     g_1[102,1] := -0.019201;
246     g_1[102,2] := -0.0082845;
247     g_1[103,1] := -0.019201;
248     g_1[103,2] := -0.0082856;
249     g_1[104,1] := -0.019201;
250     g_1[104,2] := -0.0082866;
251     g_1[105,1] := 0.061637;
252     g_1[105,2] := -0.010127;
253     g_1[106,1] := 0.11749;
254     g_1[106,2] := -0.027747;
255     g_1[107,1] := 0.16809;
256     g_1[107,2] := -0.050711;
257     g_1[108,1] := 0.21394;
258     g_1[108,2] := -0.077275;
259     g_1[109,1] := 0.25549;
260     g_1[109,2] := -0.10605;
261     g_1[110,1] := 0.29314;
262     g_1[110,2] := -0.13592;
263     g_1[111,1] := 0.32725;
264     g_1[111,2] := -0.16603;
265     g_1[112,1] := 0.35816;
266     g_1[112,2] := -0.19572;
267     g_1[113,1] := 0.38616;
268     g_1[113,2] := -0.2245;
269     g_1[114,1] := 0.41154;
270     g_1[114,2] := -0.252;
271     g_1[115,1] := 0.43453;
272     g_1[115,2] := -0.27799;
273     g_1[116,1] := 0.45537;
274     g_1[116,2] := -0.30232;
275     g_1[117,1] := 0.47425;
276     g_1[117,2] := -0.32488;
277     g_1[118,1] := 0.49135;
278     g_1[118,2] := -0.34566;
279     g_1[119,1] := 0.50685;
280     g_1[119,2] := -0.36466;
281     g_1[120,1] := 0.52089;
282     g_1[120,2] := -0.38194;
283     g_1[121,1] := 0.53362;
284     g_1[121,2] := -0.39756;
285     g_1[122,1] := 0.54515;
286     g_1[122,2] := -0.4116;
287     g_1[123,1] := 0.5556;
288     g_1[123,2] := -0.42417;
289     g_1[124,1] := 0.56506;
290     g_1[124,2] := -0.43537;
291     g_1[125,1] := 0.57364;
292     g_1[125,2] := -0.4453;
293     g_1[126,1] := 0.58141;
294     g_1[126,2] := -0.45406;
295     g_1[127,1] := 0.58846;
296     g_1[127,2] := -0.46177;
297     g_1[128,1] := 0.59484;
298     g_1[128,2] := -0.46852;
299     g_1[129,1] := 0.60062;
300     g_1[129,2] := -0.47441;
301     g_1[130,1] := 0.60586;
302     g_1[130,2] := -0.47952;
303     g_1[131,1] := 0.6106;
304     g_1[131,2] := -0.48394;
305     g_1[132,1] := 0.61491;
306     g_1[132,2] := -0.48775;
307     g_1[133,1] := 0.6188;
308     g_1[133,2] := -0.49101;
309     g_1[134,1] := 0.62233;
310     g_1[134,2] := -0.4938;
311     g_1[135,1] := 0.62553;
312     g_1[135,2] := -0.49617;
313     g_1[136,1] := 0.62843;
314     g_1[136,2] := -0.49818;
315     g_1[137,1] := 0.63106;
316     g_1[137,2] := -0.49986;
317     g_1[138,1] := 0.63344;
318     g_1[138,2] := -0.50128;
319     g_1[139,1] := 0.6356;
```

```
320     g_1[139,2] := -0.50245;
321     g_1[140,1] := 0.63755;
322     g_1[140,2] := -0.50342;
323     g_1[141,1] := 0.63932;
324     g_1[141,2] := -0.50421;
325     g_1[142,1] := 0.64093;
326     g_1[142,2] := -0.50486;
327     g_1[143,1] := 0.64238;
328     g_1[143,2] := -0.50538;
329     g_1[144,1] := 0.6437;
330     g_1[144,2] := -0.5058;
331     g_1[145,1] := 0.64489;
332     g_1[145,2] := -0.50612;
333     g_1[146,1] := 0.64597;
334     g_1[146,2] := -0.50638;
335     g_1[147,1] := 0.64695;
336     g_1[147,2] := -0.50657;
337     g_1[148,1] := 0.64784;
338     g_1[148,2] := -0.50671;
339     g_1[149,1] := 0.64865;
340     g_1[149,2] := -0.50681;
341     g_1[150,1] := 0.64938;
342     g_1[150,2] := -0.50687;
343     g_1[151,1] := 0.65004;
344     g_1[151,2] := -0.50691;
345     g_1[152,1] := 0.65064;
346     g_1[152,2] := -0.50691;
347     g_1[153,1] := 0.65118;
348     g_1[153,2] := -0.50691;
349     g_1[154,1] := 0.65167;
350     g_1[154,2] := -0.50691;
351     g_1[155,1] := 0.65211;
352     g_1[155,2] := -0.50691;
353     g_1[156,1] := 0.65252;
354     g_1[156,2] := -0.50691;
355     g_1[157,1] := 0.65288;
356     g_1[157,2] := -0.50691;
357     g_1[158,1] := 0.65321;
358     g_1[158,2] := -0.50691;
359     g_1[159,1] := 0.65351;
360     g_1[159,2] := -0.50691;
361     g_1[160,1] := 0.65379;
362     g_1[160,2] := -0.50691;
363     g_1[161,1] := 0.65403;
364     g_1[161,2] := -0.50691;
365     g_1[162,1] := 0.65426;
366     g_1[162,2] := -0.50691;
367     g_1[163,1] := 0.019435;
368     g_1[163,2] := 0.00024902;
369     g_1[164,1] := 0.020335;
370     g_1[164,2] := 0.00059765;
371     g_1[165,1] := 0.020253;
372     g_1[165,2] := 0.00097865;
373     g_1[166,1] := 0.020132;
374     g_1[166,2] := 0.0013621;
375     g_1[167,1] := 0.020023;
376     g_1[167,2] := 0.0017352;
377     g_1[168,1] := 0.019926;
378     g_1[168,2] := 0.0020926;
379     g_1[169,1] := 0.019841;
380     g_1[169,2] := 0.0024325;
381     g_1[170,1] := 0.019765;
382     g_1[170,2] := 0.0027547;
383     g_1[171,1] := 0.019698;
384     g_1[171,2] := 0.0030596;
385     g_1[172,1] := 0.019639;
386     g_1[172,2] := 0.003348;
387     g_1[173,1] := 0.019587;
388     g_1[173,2] := 0.0036206;
389     g_1[174,1] := 0.01954;
390     g_1[174,2] := 0.0038782;
391     g_1[175,1] := 0.0195;
392     g_1[175,2] := 0.0041216;
393     g_1[176,1] := 0.019463;
394     g_1[176,2] := 0.0043517;
395     g_1[177,1] := 0.019431;
396     g_1[177,2] := 0.0045691;
397     g_1[178,1] := 0.019403;
398     g_1[178,2] := 0.0047745;
399     g_1[179,1] := 0.019378;
```

```
400     g_1[179,2] := 0.0049686;
401     g_1[180,1] := 0.019356;
402     g_1[180,2] := 0.0051521;
403     g_1[181,1] := 0.019336;
404     g_1[181,2] := 0.0053254;
405     g_1[182,1] := 0.019319;
406     g_1[182,2] := 0.0054892;
407     g_1[183,1] := 0.019304;
408     g_1[183,2] := 0.005644;
409     g_1[184,1] := 0.01929;
410     g_1[184,2] := 0.0057903;
411     g_1[185,1] := 0.019278;
412     g_1[185,2] := 0.0059285;
413     g_1[186,1] := 0.019268;
414     g_1[186,2] := 0.0060591;
415     g_1[187,1] := 0.019258;
416     g_1[187,2] := 0.0061825;
417     g_1[188,1] := 0.01925;
418     g_1[188,2] := 0.0062991;
419     g_1[189,1] := 0.019243;
420     g_1[189,2] := 0.0064093;
421     g_1[190,1] := 0.019236;
422     g_1[190,2] := 0.0065134;
423     g_1[191,1] := 0.019231;
424     g_1[191,2] := 0.0066118;
425     g_1[192,1] := 0.019226;
426     g_1[192,2] := 0.0067048;
427     g_1[193,1] := 0.019221;
428     g_1[193,2] := 0.0067927;
429     g_1[194,1] := 0.019217;
430     g_1[194,2] := 0.0068757;
431     g_1[195,1] := 0.019214;
432     g_1[195,2] := 0.0069541;
433     g_1[196,1] := 0.019211;
434     g_1[196,2] := 0.0070283;
435     g_1[197,1] := 0.019208;
436     g_1[197,2] := 0.0070983;
437     g_1[198,1] := 0.019205;
438     g_1[198,2] := 0.0071645;
439     g_1[199,1] := 0.019203;
440     g_1[199,2] := 0.0072271;
441     g_1[200,1] := 0.019201;
442     g_1[200,2] := 0.0072862;
443     g_1[201,1] := 0.0192;
444     g_1[201,2] := 0.007342;
445     g_1[202,1] := 0.019198;
446     g_1[202,2] := 0.0073948;
447     g_1[203,1] := 0.019197;
448     g_1[203,2] := 0.0074447;
449     g_1[204,1] := 0.019196;
450     g_1[204,2] := 0.0074918;
451     g_1[205,1] := 0.019195;
452     g_1[205,2] := 0.0075364;
453     g_1[206,1] := 0.019194;
454     g_1[206,2] := 0.0075785;
455     g_1[207,1] := 0.019193;
456     g_1[207,2] := 0.0076182;
457     g_1[208,1] := 0.019192;
458     g_1[208,2] := 0.0076558;
459     g_1[209,1] := 0.019192;
460     g_1[209,2] := 0.0076913;
461     g_1[210,1] := 0.019192;
462     g_1[210,2] := 0.0077249;
463     g_1[211,1] := 0.019192;
464     g_1[211,2] := 0.0077566;
465     g_1[212,1] := 0.019192;
466     g_1[212,2] := 0.0077865;
467     g_1[213,1] := 0.019192;
468     g_1[213,2] := 0.0078149;
469     g_1[214,1] := 0.019192;
470     g_1[214,2] := 0.0078416;
471     g_1[215,1] := 0.019192;
472     g_1[215,2] := 0.0078669;
473     g_1[216,1] := 0.019192;
474     g_1[216,2] := 0.0078908;
475     g_1[217,1] := 0.019192;
476     g_1[217,2] := 0.0079134;
477     g_1[218,1] := 0.019192;
478     g_1[218,2] := 0.0079347;
479     g_1[219,1] := 0.019192;
```

```
480     g_1[219,2] := 0.0079548;
481     g_1[220,1] := 0.019192;
482     g_1[220,2] := 0.0079739;
483     g_1[221,1] := 0.019192;
484     g_1[221,2] := 0.0079919;
485     g_1[222,1] := 0.019192;
486     g_1[222,2] := 0.0080089;
487     g_1[223,1] := 0.019192;
488     g_1[223,2] := 0.008025;
489     g_1[224,1] := 0.019192;
490     g_1[224,2] := 0.0080402;
491     g_1[225,1] := 0.019192;
492     g_1[225,2] := 0.0080545;
493     g_1[226,1] := 0.019192;
494     g_1[226,2] := 0.0080681;
495     g_1[227,1] := 0.019192;
496     g_1[227,2] := 0.0080809;
497     g_1[228,1] := 0.019192;
498     g_1[228,2] := 0.008093;
499     g_1[229,1] := 0.019192;
500     g_1[229,2] := 0.0081044;
501     g_1[230,1] := 0.019192;
502     g_1[230,2] := 0.0081153;
503     g_1[231,1] := 0.019192;
504     g_1[231,2] := 0.0081255;
505     g_1[232,1] := 0.019192;
506     g_1[232,2] := 0.0081351;
507     g_1[233,1] := 0.019192;
508     g_1[233,2] := 0.0081443;
509     g_1[234,1] := 0.019192;
510     g_1[234,2] := 0.0081529;
511     g_1[235,1] := 0.019192;
512     g_1[235,2] := 0.008161;
513     g_1[236,1] := 0.019192;
514     g_1[236,2] := 0.0081687;
515     g_1[237,1] := 0.019192;
516     g_1[237,2] := 0.008176;
517     g_1[238,1] := 0.019192;
518     g_1[238,2] := 0.0081829;
519     g_1[239,1] := 0.019192;
520     g_1[239,2] := 0.0081894;
521     g_1[240,1] := 0.019192;
522     g_1[240,2] := 0.0081955;
523     g_1[241,1] := 0.019192;
524     g_1[241,2] := 0.0082013;
525     g_1[242,1] := 0.019192;
526     g_1[242,2] := 0.0082068;
527     g_1[243,1] := 0.019192;
528     g_1[243,2] := 0.008212;
529     g_1[244,1] := 0.019192;
530     g_1[244,2] := 0.0082169;
531     g_1[245,1] := 0.019192;
532     g_1[245,2] := 0.0082215;
533     g_1[246,1] := 0.019192;
534     g_1[246,2] := 0.0082258;
535     g_1[247,1] := 0.019192;
536     g_1[247,2] := 0.00823;
537     g_1[248,1] := 0.019192;
538     g_1[248,2] := 0.0082339;
539     g_1[249,1] := 0.019192;
540     g_1[249,2] := 0.0082376;
541     g_1[250,1] := 0.019192;
542     g_1[250,2] := 0.0082411;
543     g_1[251,1] := 0.019192;
544     g_1[251,2] := 0.0082443;
545     g_1[252,1] := 0.019192;
546     g_1[252,2] := 0.0082475;
547     g_1[253,1] := 0.019192;
548     g_1[253,2] := 0.0082504;
549     g_1[254,1] := 0.019192;
550     g_1[254,2] := 0.0082532;
551     g_1[255,1] := 0.019192;
552     g_1[255,2] := 0.0082558;
553     g_1[256,1] := 0.019192;
554     g_1[256,2] := 0.0082583;
555     g_1[257,1] := 0.019192;
556     g_1[257,2] := 0.0082606;
557     g_1[258,1] := 0.019192;
558     g_1[258,2] := 0.0082628;
559     g_1[259,1] := 0.019192;
```

```
560     g_1[259,2] := 0.0082649;
561     g_1[260,1] := 0.019192;
562     g_1[260,2] := 0.0082669;
563     g_1[261,1] := 0.019192;
564     g_1[261,2] := 0.0082688;
565     g_1[262,1] := 0.019192;
566     g_1[262,2] := 0.0082705;
567     g_1[263,1] := 0.019192;
568     g_1[263,2] := 0.0082722;
569     g_1[264,1] := 0.019192;
570     g_1[264,2] := 0.0082738;
571     g_1[265,1] := 0.019192;
572     g_1[265,2] := 0.0082753;
573     g_1[266,1] := 0.019192;
574     g_1[266,2] := 0.0082767;
575
576     (* Coeficientes de la respuesta salto - Perturbación/Salida - d_1[nCoef(i)*nSal,
    nPer]*)
577
578     d_1[1,1] := 0.088391;
579     d_1[1,2] := 0.00087099;
580     d_1[1,3] := 0.00070195;
581     d_1[2,1] := 0.16883;
582     d_1[2,2] := 0.0016636;
583     d_1[2,3] := 0.0013407;
584     d_1[3,1] := 0.24202;
585     d_1[3,2] := 0.0023849;
586     d_1[3,3] := 0.001922;
587     d_1[4,1] := 0.30863;
588     d_1[4,2] := 0.0030412;
589     d_1[4,3] := 0.002451;
590     d_1[5,1] := 0.36924;
591     d_1[5,2] := 0.0036385;
592     d_1[5,3] := 0.0029324;
593     d_1[6,1] := 0.4244;
594     d_1[6,2] := 0.004182;
595     d_1[6,3] := 0.0033704;
596     d_1[7,1] := 0.4746;
597     d_1[7,2] := 0.0046766;
598     d_1[7,3] := 0.003769;
599     d_1[8,1] := 0.52027;
600     d_1[8,2] := 0.0051267;
601     d_1[8,3] := 0.0041318;
602     d_1[9,1] := 0.56184;
603     d_1[9,2] := 0.0055363;
604     d_1[9,3] := 0.0044619;
605     d_1[10,1] := 0.59966;
606     d_1[10,2] := 0.005909;
607     d_1[10,3] := 0.0047622;
608     d_1[11,1] := 0.63409;
609     d_1[11,2] := 0.0062482;
610     d_1[11,3] := 0.0050356;
611     d_1[12,1] := 0.66541;
612     d_1[12,2] := 0.0065568;
613     d_1[12,3] := 0.0052843;
614     d_1[13,1] := 0.69391;
615     d_1[13,2] := 0.0068377;
616     d_1[13,3] := 0.0055107;
617     d_1[14,1] := 0.71985;
618     d_1[14,2] := 0.0070933;
619     d_1[14,3] := 0.0057167;
620     d_1[15,1] := 0.74345;
621     d_1[15,2] := 0.0073259;
622     d_1[15,3] := 0.0059042;
623     d_1[16,1] := 0.76493;
624     d_1[16,2] := 0.0075376;
625     d_1[16,3] := 0.0060747;
626     d_1[17,1] := 0.78448;
627     d_1[17,2] := 0.0077302;
628     d_1[17,3] := 0.00623;
629     d_1[18,1] := 0.80227;
630     d_1[18,2] := 0.0079054;
631     d_1[18,3] := 0.0063712;
632     d_1[19,1] := 0.81845;
633     d_1[19,2] := 0.0080649;
634     d_1[19,3] := 0.0064998;
635     d_1[20,1] := 0.83318;
636     d_1[20,2] := 0.0082101;
637     d_1[20,3] := 0.0066167;
638     d_1[21,1] := 0.84659;
```

```
639     d_1[21,2] := 0.0083422;
640     d_1[21,3] := 0.0067232;
641     d_1[22,1] := 0.85879;
642     d_1[22,2] := 0.0084623;
643     d_1[22,3] := 0.0068201;
644     d_1[23,1] := 0.86989;
645     d_1[23,2] := 0.0085717;
646     d_1[23,3] := 0.0069082;
647     d_1[24,1] := 0.87999;
648     d_1[24,2] := 0.0086713;
649     d_1[24,3] := 0.0069884;
650     d_1[25,1] := 0.88918;
651     d_1[25,2] := 0.0087618;
652     d_1[25,3] := 0.0070614;
653     d_1[26,1] := 0.89754;
654     d_1[26,2] := 0.0088443;
655     d_1[26,3] := 0.0071278;
656     d_1[27,1] := 0.90515;
657     d_1[27,2] := 0.0089193;
658     d_1[27,3] := 0.0071883;
659     d_1[28,1] := 0.91208;
660     d_1[28,2] := 0.0089875;
661     d_1[28,3] := 0.0072433;
662     d_1[29,1] := 0.91838;
663     d_1[29,2] := 0.0090496;
664     d_1[29,3] := 0.0072934;
665     d_1[30,1] := 0.92412;
666     d_1[30,2] := 0.0091061;
667     d_1[30,3] := 0.0073389;
668     d_1[31,1] := 0.92934;
669     d_1[31,2] := 0.0091576;
670     d_1[31,3] := 0.0073804;
671     d_1[32,1] := 0.93409;
672     d_1[32,2] := 0.0092044;
673     d_1[32,3] := 0.0074181;
674     d_1[33,1] := 0.93841;
675     d_1[33,2] := 0.009247;
676     d_1[33,3] := 0.0074524;
677     d_1[34,1] := 0.94235;
678     d_1[34,2] := 0.0092857;
679     d_1[34,3] := 0.0074836;
680     d_1[35,1] := 0.94593;
681     d_1[35,2] := 0.009321;
682     d_1[35,3] := 0.0075121;
683     d_1[36,1] := 0.94918;
684     d_1[36,2] := 0.0093531;
685     d_1[36,3] := 0.0075379;
686     d_1[37,1] := 0.95215;
687     d_1[37,2] := 0.0093823;
688     d_1[37,3] := 0.0075615;
689     d_1[38,1] := 0.95484;
690     d_1[38,2] := 0.0094089;
691     d_1[38,3] := 0.0075829;
692     d_1[39,1] := 0.9573;
693     d_1[39,2] := 0.0094331;
694     d_1[39,3] := 0.0076024;
695     d_1[40,1] := 0.95953;
696     d_1[40,2] := 0.0094551;
697     d_1[40,3] := 0.0076201;
698     d_1[41,1] := 0.96156;
699     d_1[41,2] := 0.0094751;
700     d_1[41,3] := 0.0076363;
701     d_1[42,1] := 0.96341;
702     d_1[42,2] := 0.0094933;
703     d_1[42,3] := 0.007651;
704     d_1[43,1] := 0.9651;
705     d_1[43,2] := 0.0095099;
706     d_1[43,3] := 0.0076643;
707     d_1[44,1] := 0.96663;
708     d_1[44,2] := 0.009525;
709     d_1[44,3] := 0.0076765;
710     d_1[45,1] := 0.96802;
711     d_1[45,2] := 0.0095388;
712     d_1[45,3] := 0.0076876;
713     d_1[46,1] := 0.96929;
714     d_1[46,2] := 0.0095513;
715     d_1[46,3] := 0.0076976;
716     d_1[47,1] := 0.97045;
717     d_1[47,2] := 0.0095626;
718     d_1[47,3] := 0.0077068;
```

```
719     d_1[48,1] := 0.9715;
720     d_1[48,2] := 0.009573;
721     d_1[48,3] := 0.0077152;
722     d_1[49,1] := 0.97245;
723     d_1[49,2] := 0.0095824;
724     d_1[49,3] := 0.0077227;
725     d_1[50,1] := 0.97332;
726     d_1[50,2] := 0.009591;
727     d_1[50,3] := 0.0077296;
728     d_1[51,1] := 0.97411;
729     d_1[51,2] := 0.0095988;
730     d_1[51,3] := 0.0077359;
731     d_1[52,1] := 0.97483;
732     d_1[52,2] := 0.0096059;
733     d_1[52,3] := 0.0077417;
734     d_1[53,1] := 0.97549;
735     d_1[53,2] := 0.0096123;
736     d_1[53,3] := 0.0077469;
737     d_1[54,1] := 0.97609;
738     d_1[54,2] := 0.0096182;
739     d_1[54,3] := 0.0077516;
740     d_1[55,1] := 0.97663;
741     d_1[55,2] := 0.0096236;
742     d_1[55,3] := 0.0077559;
743     d_1[56,1] := 0.97712;
744     d_1[56,2] := 0.0096284;
745     d_1[56,3] := 0.0077598;
746     d_1[57,1] := 0.97757;
747     d_1[57,2] := 0.0096329;
748     d_1[57,3] := 0.0077634;
749     d_1[58,1] := 0.97798;
750     d_1[58,2] := 0.0096369;
751     d_1[58,3] := 0.0077666;
752     d_1[59,1] := 0.97835;
753     d_1[59,2] := 0.0096406;
754     d_1[59,3] := 0.0077696;
755     d_1[60,1] := 0.97869;
756     d_1[60,2] := 0.0096439;
757     d_1[60,3] := 0.0077723;
758     d_1[61,1] := 0.979;
759     d_1[61,2] := 0.0096469;
760     d_1[61,3] := 0.0077747;
761     d_1[62,1] := 0.97928;
762     d_1[62,2] := 0.0096497;
763     d_1[62,3] := 0.007777;
764     d_1[63,1] := 0.97954;
765     d_1[63,2] := 0.0096522;
766     d_1[63,3] := 0.007779;
767     d_1[64,1] := 0.97977;
768     d_1[64,2] := 0.0096545;
769     d_1[64,3] := 0.0077808;
770     d_1[65,1] := 0.97998;
771     d_1[65,2] := 0.0096566;
772     d_1[65,3] := 0.0077825;
773     d_1[66,1] := 0.98017;
774     d_1[66,2] := 0.0096585;
775     d_1[66,3] := 0.007784;
776     d_1[67,1] := 0.98035;
777     d_1[67,2] := 0.0096602;
778     d_1[67,3] := 0.0077854;
779     d_1[68,1] := 0.98051;
780     d_1[68,2] := 0.0096618;
781     d_1[68,3] := 0.0077867;
782     d_1[69,1] := 0.98065;
783     d_1[69,2] := 0.0096632;
784     d_1[69,3] := 0.0077879;
785     d_1[70,1] := 0.98078;
786     d_1[70,2] := 0.0096645;
787     d_1[70,3] := 0.0077889;
788     d_1[71,1] := 0.9809;
789     d_1[71,2] := 0.0096657;
790     d_1[71,3] := 0.0077899;
791     d_1[72,1] := 0.98101;
792     d_1[72,2] := 0.0096668;
793     d_1[72,3] := 0.0077907;
794     d_1[73,1] := 0.98111;
795     d_1[73,2] := 0.0096677;
796     d_1[73,3] := 0.0077915;
797     d_1[74,1] := 0.9812;
798     d_1[74,2] := 0.0096686;
```

```
799     d_1[74,3] := 0.0077922;
800     d_1[75,1] := 0.98128;
801     d_1[75,2] := 0.0096694;
802     d_1[75,3] := 0.0077929;
803     d_1[76,1] := 0.98136;
804     d_1[76,2] := 0.0096702;
805     d_1[76,3] := 0.0077935;
806     d_1[77,1] := 0.98143;
807     d_1[77,2] := 0.0096708;
808     d_1[77,3] := 0.007794;
809     d_1[78,1] := 0.98149;
810     d_1[78,2] := 0.0096715;
811     d_1[78,3] := 0.0077945;
812     d_1[79,1] := 0.98155;
813     d_1[79,2] := 0.009672;
814     d_1[79,3] := 0.007795;
815     d_1[80,1] := 0.9816;
816     d_1[80,2] := 0.0096725;
817     d_1[80,3] := 0.0077954;
818     d_1[81,1] := 0.98164;
819     d_1[81,2] := 0.009673;
820     d_1[81,3] := 0.0077957;
821     d_1[82,1] := 0.98169;
822     d_1[82,2] := 0.0096734;
823     d_1[82,3] := 0.0077961;
824     d_1[83,1] := 0.98173;
825     d_1[83,2] := 0.0096738;
826     d_1[83,3] := 0.0077964;
827     d_1[84,1] := 0.98176;
828     d_1[84,2] := 0.0096741;
829     d_1[84,3] := 0.0077967;
830     d_1[85,1] := 0.98179;
831     d_1[85,2] := 0.0096744;
832     d_1[85,3] := 0.0077969;
833     d_1[86,1] := 0.98182;
834     d_1[86,2] := 0.0096747;
835     d_1[86,3] := 0.0077972;
836     d_1[87,1] := 0.98185;
837     d_1[87,2] := 0.009675;
838     d_1[87,3] := 0.0077974;
839     d_1[88,1] := 0.98187;
840     d_1[88,2] := 0.0096752;
841     d_1[88,3] := 0.0077976;
842     d_1[89,1] := 0.98189;
843     d_1[89,2] := 0.0096755;
844     d_1[89,3] := 0.0077977;
845     d_1[90,1] := 0.98191;
846     d_1[90,2] := 0.0096756;
847     d_1[90,3] := 0.0077979;
848     d_1[91,1] := 0.98193;
849     d_1[91,2] := 0.0096758;
850     d_1[91,3] := 0.007798;
851     d_1[92,1] := 0.98195;
852     d_1[92,2] := 0.009676;
853     d_1[92,3] := 0.0077982;
854     d_1[93,1] := 0.98196;
855     d_1[93,2] := 0.0096761;
856     d_1[93,3] := 0.0077983;
857     d_1[94,1] := 0.98198;
858     d_1[94,2] := 0.0096763;
859     d_1[94,3] := 0.0077984;
860     d_1[95,1] := 0.98199;
861     d_1[95,2] := 0.0096764;
862     d_1[95,3] := 0.0077985;
863     d_1[96,1] := 0.982;
864     d_1[96,2] := 0.0096765;
865     d_1[96,3] := 0.0077986;
866     d_1[97,1] := 0.98201;
867     d_1[97,2] := 0.0096766;
868     d_1[97,3] := 0.0077987;
869     d_1[98,1] := 0.98202;
870     d_1[98,2] := 0.0096767;
871     d_1[98,3] := 0.0077987;
872     d_1[99,1] := 0.98203;
873     d_1[99,2] := 0.0096768;
874     d_1[99,3] := 0.0077988;
875     d_1[100,1] := 0.98204;
876     d_1[100,2] := 0.0096769;
877     d_1[100,3] := 0.0077989;
878     d_1[101,1] := 0.98205;
```

```
879     d_1[101,2] := 0.0096769;
880     d_1[101,3] := 0.0077989;
881     d_1[102,1] := 0.98205;
882     d_1[102,2] := 0.009677;
883     d_1[102,3] := 0.007799;
884     d_1[103,1] := 0.98206;
885     d_1[103,2] := 0.0096771;
886     d_1[103,3] := 0.007799;
887     d_1[104,1] := 0.98206;
888     d_1[104,2] := 0.0096771;
889     d_1[104,3] := 0.0077991;
890     d_1[105,1] := 0.63567;
891     d_1[105,2] := 0.080236;
892     d_1[105,3] := 0.066204;
893     d_1[106,1] := 1.1855;
894     d_1[106,2] := 0.14964;
895     d_1[106,3] := 0.12347;
896     d_1[107,1] := 1.6612;
897     d_1[107,2] := 0.20967;
898     d_1[107,3] := 0.17301;
899     d_1[108,1] := 2.0726;
900     d_1[108,2] := 0.2616;
901     d_1[108,3] := 0.21586;
902     d_1[109,1] := 2.4285;
903     d_1[109,2] := 0.30652;
904     d_1[109,3] := 0.25292;
905     d_1[110,1] := 2.7363;
906     d_1[110,2] := 0.34538;
907     d_1[110,3] := 0.28498;
908     d_1[111,1] := 3.0026;
909     d_1[111,2] := 0.37899;
910     d_1[111,3] := 0.31271;
911     d_1[112,1] := 3.2329;
912     d_1[112,2] := 0.40806;
913     d_1[112,3] := 0.3367;
914     d_1[113,1] := 3.4321;
915     d_1[113,2] := 0.43321;
916     d_1[113,3] := 0.35745;
917     d_1[114,1] := 3.6045;
918     d_1[114,2] := 0.45496;
919     d_1[114,3] := 0.3754;
920     d_1[115,1] := 3.7535;
921     d_1[115,2] := 0.47378;
922     d_1[115,3] := 0.39092;
923     d_1[116,1] := 3.8825;
924     d_1[116,2] := 0.49005;
925     d_1[116,3] := 0.40435;
926     d_1[117,1] := 3.994;
927     d_1[117,2] := 0.50413;
928     d_1[117,3] := 0.41597;
929     d_1[118,1] := 4.0905;
930     d_1[118,2] := 0.51631;
931     d_1[118,3] := 0.42602;
932     d_1[119,1] := 4.174;
933     d_1[119,2] := 0.52685;
934     d_1[119,3] := 0.43471;
935     d_1[120,1] := 4.2462;
936     d_1[120,2] := 0.53596;
937     d_1[120,3] := 0.44223;
938     d_1[121,1] := 4.3086;
939     d_1[121,2] := 0.54384;
940     d_1[121,3] := 0.44873;
941     d_1[122,1] := 4.3626;
942     d_1[122,2] := 0.55066;
943     d_1[122,3] := 0.45435;
944     d_1[123,1] := 4.4094;
945     d_1[123,2] := 0.55655;
946     d_1[123,3] := 0.45922;
947     d_1[124,1] := 4.4498;
948     d_1[124,2] := 0.56166;
949     d_1[124,3] := 0.46343;
950     d_1[125,1] := 4.4847;
951     d_1[125,2] := 0.56607;
952     d_1[125,3] := 0.46707;
953     d_1[126,1] := 4.515;
954     d_1[126,2] := 0.56989;
955     d_1[126,3] := 0.47022;
956     d_1[127,1] := 4.5411;
957     d_1[127,2] := 0.57319;
958     d_1[127,3] := 0.47294;
```

```
 959     d_1[128,1] := 4.5638;
 960     d_1[128,2] := 0.57604;
 961     d_1[128,3] := 0.4753;
 962     d_1[129,1] := 4.5833;
 963     d_1[129,2] := 0.57851;
 964     d_1[129,3] := 0.47734;
 965     d_1[130,1] := 4.6003;
 966     d_1[130,2] := 0.58065;
 967     d_1[130,3] := 0.4791;
 968     d_1[131,1] := 4.6149;
 969     d_1[131,2] := 0.5825;
 970     d_1[131,3] := 0.48062;
 971     d_1[132,1] := 4.6276;
 972     d_1[132,2] := 0.5841;
 973     d_1[132,3] := 0.48194;
 974     d_1[133,1] := 4.6385;
 975     d_1[133,2] := 0.58548;
 976     d_1[133,3] := 0.48308;
 977     d_1[134,1] := 4.648;
 978     d_1[134,2] := 0.58668;
 979     d_1[134,3] := 0.48407;
 980     d_1[135,1] := 4.6562;
 981     d_1[135,2] := 0.58771;
 982     d_1[135,3] := 0.48492;
 983     d_1[136,1] := 4.6633;
 984     d_1[136,2] := 0.58861;
 985     d_1[136,3] := 0.48566;
 986     d_1[137,1] := 4.6694;
 987     d_1[137,2] := 0.58938;
 988     d_1[137,3] := 0.4863;
 989     d_1[138,1] := 4.6748;
 990     d_1[138,2] := 0.59005;
 991     d_1[138,3] := 0.48685;
 992     d_1[139,1] := 4.6793;
 993     d_1[139,2] := 0.59063;
 994     d_1[139,3] := 0.48733;
 995     d_1[140,1] := 4.6833;
 996     d_1[140,2] := 0.59113;
 997     d_1[140,3] := 0.48774;
 998     d_1[141,1] := 4.6868;
 999     d_1[141,2] := 0.59157;
1000     d_1[141,3] := 0.4881;
1001     d_1[142,1] := 4.6897;
1002     d_1[142,2] := 0.59194;
1003     d_1[142,3] := 0.48841;
1004     d_1[143,1] := 4.6923;
1005     d_1[143,2] := 0.59227;
1006     d_1[143,3] := 0.48868;
1007     d_1[144,1] := 4.6945;
1008     d_1[144,2] := 0.59255;
1009     d_1[144,3] := 0.48891;
1010     d_1[145,1] := 4.6964;
1011     d_1[145,2] := 0.59279;
1012     d_1[145,3] := 0.48911;
1013     d_1[146,1] := 4.6981;
1014     d_1[146,2] := 0.593;
1015     d_1[146,3] := 0.48928;
1016     d_1[147,1] := 4.6996;
1017     d_1[147,2] := 0.59318;
1018     d_1[147,3] := 0.48943;
1019     d_1[148,1] := 4.7008;
1020     d_1[148,2] := 0.59334;
1021     d_1[148,3] := 0.48956;
1022     d_1[149,1] := 4.7019;
1023     d_1[149,2] := 0.59347;
1024     d_1[149,3] := 0.48967;
1025     d_1[150,1] := 4.7028;
1026     d_1[150,2] := 0.59359;
1027     d_1[150,3] := 0.48977;
1028     d_1[151,1] := 4.7036;
1029     d_1[151,2] := 0.59369;
1030     d_1[151,3] := 0.48985;
1031     d_1[152,1] := 4.7043;
1032     d_1[152,2] := 0.59378;
1033     d_1[152,3] := 0.48992;
1034     d_1[153,1] := 4.7049;
1035     d_1[153,2] := 0.59386;
1036     d_1[153,3] := 0.48999;
1037     d_1[154,1] := 4.7054;
1038     d_1[154,2] := 0.59392;
```

```
1039     d_1[154,3] := 0.49004;
1040     d_1[155,1] := 4.7059;
1041     d_1[155,2] := 0.59398;
1042     d_1[155,3] := 0.49009;
1043     d_1[156,1] := 4.7063;
1044     d_1[156,2] := 0.59403;
1045     d_1[156,3] := 0.49013;
1046     d_1[157,1] := 4.7066;
1047     d_1[157,2] := 0.59407;
1048     d_1[157,3] := 0.49016;
1049     d_1[158,1] := 4.7069;
1050     d_1[158,2] := 0.59411;
1051     d_1[158,3] := 0.49019;
1052     d_1[159,1] := 4.7072;
1053     d_1[159,2] := 0.59414;
1054     d_1[159,3] := 0.49022;
1055     d_1[160,1] := 4.7074;
1056     d_1[160,2] := 0.59417;
1057     d_1[160,3] := 0.49024;
1058     d_1[161,1] := 4.7076;
1059     d_1[161,2] := 0.59419;
1060     d_1[161,3] := 0.49026;
1061     d_1[162,1] := 4.7077;
1062     d_1[162,2] := 0.59421;
1063     d_1[162,3] := 0.49028;
1064     d_1[163,1] := 0.034129;
1065     d_1[163,2] := -0.00064839;
1066     d_1[163,3] := -0.00053565;
1067     d_1[164,1] := 0.044985;
1068     d_1[164,2] := -0.0014803;
1069     d_1[164,3] := -0.0012229;
1070     d_1[165,1] := 0.046968;
1071     d_1[165,2] := -0.0023206;
1072     d_1[165,3] := -0.0019171;
1073     d_1[166,1] := 0.045713;
1074     d_1[166,2] := -0.0031077;
1075     d_1[166,3] := -0.0025673;
1076     d_1[167,1] := 0.043408;
1077     d_1[167,2] := -0.0038233;
1078     d_1[167,3] := -0.0031585;
1079     d_1[168,1] := 0.040891;
1080     d_1[168,2] := -0.004466;
1081     d_1[168,3] := -0.0036895;
1082     d_1[169,1] := 0.038467;
1083     d_1[169,2] := -0.0050401;
1084     d_1[169,3] := -0.0041638;
1085     d_1[170,1] := 0.036239;
1086     d_1[170,2] := -0.0055518;
1087     d_1[170,3] := -0.0045865;
1088     d_1[171,1] := 0.034229;
1089     d_1[171,2] := -0.0060073;
1090     d_1[171,3] := -0.0049628;
1091     d_1[172,1] := 0.03243;
1092     d_1[172,2] := -0.0064126;
1093     d_1[172,3] := -0.0052976;
1094     d_1[173,1] := 0.030825;
1095     d_1[173,2] := -0.0067732;
1096     d_1[173,3] := -0.0055955;
1097     d_1[174,1] := 0.029395;
1098     d_1[174,2] := -0.007094;
1099     d_1[174,3] := -0.0058605;
1100     d_1[175,1] := 0.028123;
1101     d_1[175,2] := -0.0073794;
1102     d_1[175,3] := -0.0060963;
1103     d_1[176,1] := 0.026991;
1104     d_1[176,2] := -0.0076332;
1105     d_1[176,3] := -0.006306;
1106     d_1[177,1] := 0.025984;
1107     d_1[177,2] := -0.0078591;
1108     d_1[177,3] := -0.0064926;
1109     d_1[178,1] := 0.025089;
1110     d_1[178,2] := -0.0080599;
1111     d_1[178,3] := -0.0066585;
1112     d_1[179,1] := 0.024292;
1113     d_1[179,2] := -0.0082386;
1114     d_1[179,3] := -0.0068061;
1115     d_1[180,1] := 0.023583;
1116     d_1[180,2] := -0.0083976;
1117     d_1[180,3] := -0.0069374;
1118     d_1[181,1] := 0.022953;
```

```
1119     d_1[181,2] := -0.0085389;
1120     d_1[181,3] := -0.0070542;
1121     d_1[182,1] := 0.022392;
1122     d_1[182,2] := -0.0086647;
1123     d_1[182,3] := -0.0071581;
1124     d_1[183,1] := 0.021893;
1125     d_1[183,2] := -0.0087766;
1126     d_1[183,3] := -0.0072506;
1127     d_1[184,1] := 0.02145;
1128     d_1[184,2] := -0.0088761;
1129     d_1[184,3] := -0.0073328;
1130     d_1[185,1] := 0.021055;
1131     d_1[185,2] := -0.0089646;
1132     d_1[185,3] := -0.0074059;
1133     d_1[186,1] := 0.020704;
1134     d_1[186,2] := -0.0090434;
1135     d_1[186,3] := -0.007471;
1136     d_1[187,1] := 0.020392;
1137     d_1[187,2] := -0.0091134;
1138     d_1[187,3] := -0.0075288;
1139     d_1[188,1] := 0.020114;
1140     d_1[188,2] := -0.0091758;
1141     d_1[188,3] := -0.0075803;
1142     d_1[189,1] := 0.019867;
1143     d_1[189,2] := -0.0092312;
1144     d_1[189,3] := -0.0076261;
1145     d_1[190,1] := 0.019648;
1146     d_1[190,2] := -0.0092805;
1147     d_1[190,3] := -0.0076669;
1148     d_1[191,1] := 0.019452;
1149     d_1[191,2] := -0.0093244;
1150     d_1[191,3] := -0.0077031;
1151     d_1[192,1] := 0.019279;
1152     d_1[192,2] := -0.0093634;
1153     d_1[192,3] := -0.0077353;
1154     d_1[193,1] := 0.019124;
1155     d_1[193,2] := -0.0093981;
1156     d_1[193,3] := -0.007764;
1157     d_1[194,1] := 0.018987;
1158     d_1[194,2] := -0.009429;
1159     d_1[194,3] := -0.0077895;
1160     d_1[195,1] := 0.018864;
1161     d_1[195,2] := -0.0094564;
1162     d_1[195,3] := -0.0078122;
1163     d_1[196,1] := 0.018756;
1164     d_1[196,2] := -0.0094809;
1165     d_1[196,3] := -0.0078324;
1166     d_1[197,1] := 0.018659;
1167     d_1[197,2] := -0.0095026;
1168     d_1[197,3] := -0.0078503;
1169     d_1[198,1] := 0.018573;
1170     d_1[198,2] := -0.0095219;
1171     d_1[198,3] := -0.0078663;
1172     d_1[199,1] := 0.018496;
1173     d_1[199,2] := -0.0095391;
1174     d_1[199,3] := -0.0078805;
1175     d_1[200,1] := 0.018428;
1176     d_1[200,2] := -0.0095544;
1177     d_1[200,3] := -0.0078931;
1178     d_1[201,1] := 0.018368;
1179     d_1[201,2] := -0.009568;
1180     d_1[201,3] := -0.0079044;
1181     d_1[202,1] := 0.018314;
1182     d_1[202,2] := -0.0095801;
1183     d_1[202,3] := -0.0079144;
1184     d_1[203,1] := 0.018266;
1185     d_1[203,2] := -0.0095909;
1186     d_1[203,3] := -0.0079233;
1187     d_1[204,1] := 0.018224;
1188     d_1[204,2] := -0.0096005;
1189     d_1[204,3] := -0.0079312;
1190     d_1[205,1] := 0.018186;
1191     d_1[205,2] := -0.009609;
1192     d_1[205,3] := -0.0079382;
1193     d_1[206,1] := 0.018152;
1194     d_1[206,2] := -0.0096166;
1195     d_1[206,3] := -0.0079445;
1196     d_1[207,1] := 0.018122;
1197     d_1[207,2] := -0.0096233;
1198     d_1[207,3] := -0.0079501;
```

```
1199     d_1[208,1] := 0.018096;
1200     d_1[208,2] := -0.0096293;
1201     d_1[208,3] := -0.007955;
1202     d_1[209,1] := 0.018072;
1203     d_1[209,2] := -0.0096346;
1204     d_1[209,3] := -0.0079594;
1205     d_1[210,1] := 0.018051;
1206     d_1[210,2] := -0.0096394;
1207     d_1[210,3] := -0.0079633;
1208     d_1[211,1] := 0.018032;
1209     d_1[211,2] := -0.0096436;
1210     d_1[211,3] := -0.0079668;
1211     d_1[212,1] := 0.018016;
1212     d_1[212,2] := -0.0096474;
1213     d_1[212,3] := -0.0079699;
1214     d_1[213,1] := 0.018001;
1215     d_1[213,2] := -0.0096507;
1216     d_1[213,3] := -0.0079727;
1217     d_1[214,1] := 0.017988;
1218     d_1[214,2] := -0.0096537;
1219     d_1[214,3] := -0.0079751;
1220     d_1[215,1] := 0.017976;
1221     d_1[215,2] := -0.0096563;
1222     d_1[215,3] := -0.0079773;
1223     d_1[216,1] := 0.017966;
1224     d_1[216,2] := -0.0096587;
1225     d_1[216,3] := -0.0079793;
1226     d_1[217,1] := 0.017956;
1227     d_1[217,2] := -0.0096608;
1228     d_1[217,3] := -0.007981;
1229     d_1[218,1] := 0.017948;
1230     d_1[218,2] := -0.0096626;
1231     d_1[218,3] := -0.0079825;
1232     d_1[219,1] := 0.017941;
1233     d_1[219,2] := -0.0096643;
1234     d_1[219,3] := -0.0079839;
1235     d_1[220,1] := 0.017934;
1236     d_1[220,2] := -0.0096658;
1237     d_1[220,3] := -0.0079851;
1238     d_1[221,1] := 0.017929;
1239     d_1[221,2] := -0.0096671;
1240     d_1[221,3] := -0.0079862;
1241     d_1[222,1] := 0.017923;
1242     d_1[222,2] := -0.0096682;
1243     d_1[222,3] := -0.0079872;
1244     d_1[223,1] := 0.017919;
1245     d_1[223,2] := -0.0096693;
1246     d_1[223,3] := -0.007988;
1247     d_1[224,1] := 0.017915;
1248     d_1[224,2] := -0.0096702;
1249     d_1[224,3] := -0.0079888;
1250     d_1[225,1] := 0.017911;
1251     d_1[225,2] := -0.009671;
1252     d_1[225,3] := -0.0079895;
1253     d_1[226,1] := 0.017908;
1254     d_1[226,2] := -0.0096718;
1255     d_1[226,3] := -0.0079901;
1256     d_1[227,1] := 0.017905;
1257     d_1[227,2] := -0.0096724;
1258     d_1[227,3] := -0.0079906;
1259     d_1[228,1] := 0.017903;
1260     d_1[228,2] := -0.009673;
1261     d_1[228,3] := -0.0079911;
1262     d_1[229,1] := 0.0179;
1263     d_1[229,2] := -0.0096735;
1264     d_1[229,3] := -0.0079915;
1265     d_1[230,1] := 0.017898;
1266     d_1[230,2] := -0.009674;
1267     d_1[230,3] := -0.0079919;
1268     d_1[231,1] := 0.017897;
1269     d_1[231,2] := -0.0096744;
1270     d_1[231,3] := -0.0079922;
1271     d_1[232,1] := 0.017895;
1272     d_1[232,2] := -0.0096747;
1273     d_1[232,3] := -0.0079925;
1274     d_1[233,1] := 0.017894;
1275     d_1[233,2] := -0.009675;
1276     d_1[233,3] := -0.0079928;
1277     d_1[234,1] := 0.017893;
1278     d_1[234,2] := -0.0096753;
```

```
1279      d_1[234,3] := -0.007993;
1280      d_1[235,1] := 0.017891;
1281      d_1[235,2] := -0.0096756;
1282      d_1[235,3] := -0.0079932;
1283      d_1[236,1] := 0.01789;
1284      d_1[236,2] := -0.0096758;
1285      d_1[236,3] := -0.0079934;
1286      d_1[237,1] := 0.01789;
1287      d_1[237,2] := -0.009676;
1288      d_1[237,3] := -0.0079936;
1289      d_1[238,1] := 0.017889;
1290      d_1[238,2] := -0.0096762;
1291      d_1[238,3] := -0.0079937;
1292      d_1[239,1] := 0.017888;
1293      d_1[239,2] := -0.0096764;
1294      d_1[239,3] := -0.0079939;
1295      d_1[240,1] := 0.017888;
1296      d_1[240,2] := -0.0096765;
1297      d_1[240,3] := -0.007994;
1298      d_1[241,1] := 0.017887;
1299      d_1[241,2] := -0.0096766;
1300      d_1[241,3] := -0.0079941;
1301      d_1[242,1] := 0.017887;
1302      d_1[242,2] := -0.0096767;
1303      d_1[242,3] := -0.0079942;
1304      d_1[243,1] := 0.017886;
1305      d_1[243,2] := -0.0096768;
1306      d_1[243,3] := -0.0079943;
1307      d_1[244,1] := 0.017886;
1308      d_1[244,2] := -0.0096769;
1309      d_1[244,3] := -0.0079943;
1310      d_1[245,1] := 0.017885;
1311      d_1[245,2] := -0.009677;
1312      d_1[245,3] := -0.0079944;
1313      d_1[246,1] := 0.017885;
1314      d_1[246,2] := -0.0096771;
1315      d_1[246,3] := -0.0079945;
1316      d_1[247,1] := 0.017885;
1317      d_1[247,2] := -0.0096771;
1318      d_1[247,3] := -0.0079945;
1319      d_1[248,1] := 0.017885;
1320      d_1[248,2] := -0.0096772;
1321      d_1[248,3] := -0.0079946;
1322      d_1[249,1] := 0.017884;
1323      d_1[249,2] := -0.0096772;
1324      d_1[249,3] := -0.0079946;
1325      d_1[250,1] := 0.017884;
1326      d_1[250,2] := -0.0096773;
1327      d_1[250,3] := -0.0079946;
1328      d_1[251,1] := 0.017884;
1329      d_1[251,2] := -0.0096773;
1330      d_1[251,3] := -0.0079947;
1331      d_1[252,1] := 0.017884;
1332      d_1[252,2] := -0.0096774;
1333      d_1[252,3] := -0.0079947;
1334      d_1[253,1] := 0.017884;
1335      d_1[253,2] := -0.0096774;
1336      d_1[253,3] := -0.0079947;
1337      d_1[254,1] := 0.017884;
1338      d_1[254,2] := -0.0096774;
1339      d_1[254,3] := -0.0079948;
1340      d_1[255,1] := 0.017884;
1341      d_1[255,2] := -0.0096774;
1342      d_1[255,3] := -0.0079948;
1343      d_1[256,1] := 0.017884;
1344      d_1[256,2] := -0.0096775;
1345      d_1[256,3] := -0.0079948;
1346      d_1[257,1] := 0.017883;
1347      d_1[257,2] := -0.0096775;
1348      d_1[257,3] := -0.0079948;
1349      d_1[258,1] := 0.017883;
1350      d_1[258,2] := -0.0096775;
1351      d_1[258,3] := -0.0079948;
1352      d_1[259,1] := 0.017883;
1353      d_1[259,2] := -0.0096775;
1354      d_1[259,3] := -0.0079948;
1355      d_1[260,1] := 0.017883;
1356      d_1[260,2] := -0.0096775;
1357      d_1[260,3] := -0.0079948;
1358      d_1[261,1] := 0.017883;
```

```
1359      d_1[261,2] := -0.0096775;
1360      d_1[261,3] := -0.0079949;
1361      d_1[262,1] := 0.017883;
1362      d_1[262,2] := -0.0096776;
1363      d_1[262,3] := -0.0079949;
1364      d_1[263,1] := 0.017883;
1365      d_1[263,2] := -0.0096776;
1366      d_1[263,3] := -0.0079949;
1367      d_1[264,1] := 0.017883;
1368      d_1[264,2] := -0.0096776;
1369      d_1[264,3] := -0.0079949;
1370      d_1[265,1] := 0.017883;
1371      d_1[265,2] := -0.0096776;
1372      d_1[265,3] := -0.0079949;
1373      d_1[266,1] := 0.017883;
1374      d_1[266,2] := -0.0096776;
1375      d_1[266,3] := -0.0079949;
1376
1377
1378      (* ********************************************************* *)
1379      (* Parametros de Configuracion del DMC                       *)
1380      (* ********************************************************* *)
1381
1382      (* Inicio del horizonte de coincidencia *)
1383      N1[1] := 1;      N1[2] := 1;      N1[3] := 1;
1384      (* Final del horizonte de coincidencia *)
1385      N2[1] := 30;     N2[2] := 30;     N2[3] := 30;
1386      (* Horizonte de control *)
1387      Nu[1] := 2;      Nu[2] := 2;      Nu[3] := 2;
1388
1389      (* Dinamica para la referencia *)
1390      alpha[1] := 0.9;   alpha[2] := 0.9;   alpha[3] := 0.9;
1391
1392      beta[1] := 0.5;      beta[2] := 0.5;      beta[3] := 0.0;
1393
1394      gamma[1] := 3.0;   gamma[2] := 1.0;   gamma[3] := 0.0;
1395
1396      rho[1] := 1000.0;   rho[2] := 1000.0;   rho[3] := 1000.0;
1397
1398      Du_Min[1] := -5.0;  Du_Min[2] := -5.0;  Du_Min[3] := -5.0;
1399
1400      Du_Max[1] := 5.0;   Du_Max[2] := 5.0;   Du_Max[3] := 5.0;
1401
1402      u_Min[1] := 10.0;   u_Min[2] := 8.0;    u_Min[3] := 0.0;
1403
1404      u_Max[1] := 56.0;   u_Max[2] := 57.0;   u_Max[3] := 0.0;
1405
1406      y_Min[1] := 6.2;    y_Min[2] := 55.0;   y_Min[3] := 0.0;
1407
1408      y_Max[1] := 8.0;    y_Max[2] := 74.0;   y_Max[3] := 1.5;
1409
1410  END_DATA_BLOCK
1411
```

```
ORGANIZATION_BLOCK OB1

VAR_TEMP
    // reservado
    informacion : ARRAY[0..19] OF BYTE;
    // variables temporales
END_VAR

BEGIN

    FB_DMC.DB_Reactor_DMC(flag:=FALSE);
    ;

END_ORGANIZATION_BLOCK
```

```
17 ORGANIZATION_BLOCK OB100
18
19 VAR_TEMP
20     // reservado
21     informacion : ARRAY[0..19] OF BYTE;
22     // variables temporales
23
24 END_VAR
25
26 BEGIN
27
28     FB_DMC.DB_Reactor_DMC(flag:=TRUE);
29     FB_DMC.DB_Reactor_DMC(flag:=FALSE);
30     //
31
32 END_ORGANIZATION_BLOCK
```