MÁSTER EN AUTOMÁTICA E INFORMÁTICA INDUSTRIAL

TESINA FIN DE MÁSTER

ALGORITMOS DE CONTROL PREDICTIVO MULTIVARIABLE PARA PROCESOS CON DINÁMICA RÁPIDA. APLICACIÓN AL CONTROL DE UN SISTEMA DE MOTORES ACOPLADOS

Presentado por:

Edwin Alonso González Querubín

Bajo la dirección de:

Dr. Javier Sanchis Sáez

Valencia, 2011

ÍNDICE GENERAL

I MEMORIA

II ANEXOS

DOCUMENTO I

MEMORIA

CAPÍTULO 1

INTRODUCCIÓN

pág.

1.1. INTRODUCCIÓN.

El objetivo del control predictivo es el de resolver de forma eficiente, problemas de control y automatización de una amplia gama de procesos industriales que pueden presentar comportamientos dinámicos complicados como acoplamientos entre variables, retardos de transporte, inestabilidades o incluso restricciones en algunas de sus variables. La estrategia empleada por este tipo de control, consiste en utilizar un modelo matemático del proceso, para obtener una serie de predicciones del comportamiento futuro de dicho proceso. Con base en estas predicciones y a una serie de referencias posiciones deseadas para cada variable controlada, se calculan las señales de control futuras que hacen que estas variables converjan hacia sus respectivos valores de referencia, respetando las restricciones en las diferentes variables del sistema en caso de que existan.

El algoritmo de control predictivo utilizado en este proyecto es el de Dynamic Matrix Control (DMC) o Control por Matriz Dinámica, el cual se basa en el uso de un modelo de respuesta al escalón de cada una de variables controladas con respecto a leves incrementos en las variables manipuladas, para predecir la evolución o comportamiento futuro de cada variable controlada del proceso.

Las acciones de control, en caso de que no exista ningún tipo de restricción en las variables, se calculan empleando el método de mínimos cuadrados ordinarios, obteniéndose así una ley de control lineal que se puede expresar analíticamente. Si alguna variable está sujeta a restricciones, éstas se obtienen resolviendo un problema de optimización de un índice cuadrático sujeto a restricciones, usualmente haciendo uso de la programación cuadrática.

1.2. JUSTIFICACIÓN.

Hoy en día los diversos sistemas de control empleados en procesos industriales deben satisfacer una serie de necesidades que pueden ser de tipo medioambiental, de seguridad, de calidad en la producción, económicas todas ellas relacionadas con las variables del proceso de tal manera que se minimice el coste de operación.

Una de las principales razones que hacen del control predictivo una de las estrategias de control avanzado más atractivas y que ha causado un gran impacto en el ámbito industrial, es que éste puede ser usado para controlar una amplia gama procesos desde aquellos con una dinámica relativamente simple hasta otros más complejos (sistemas con retardos temporales, multivariables, inestable, dinámica no lineal etc.), siendo capaz de ofrecer soluciones aún si se consideran restricciones en las variables.

Este es un método que resulta atractivo para personas que no poseen conocimientos profundos en materia de control, ya que los conceptos resultan muy intuitivos y su sintonización es relativamente sencilla. Por otra parte su potencia y robustez tienen un precio asociado al coste computacional debido a la gran cantidad de operaciones matemáticas que éste realiza a la hora de buscar soluciones, cobrando una gran relevancia si el control se realiza en procesos con periodos de lazo o constantes de tiempo muy pequeñas.

Este inconveniente hace necesaria la búsqueda de nuevas herramientas que complementen este tipo de estrategia para hacerla más veloz en términos computacionales y por ende apta para usarse en procesos con dinámicas rápidas.

1.3. OBJETIVOS.

1.3.1. Objetivo general.

Analizar el comportamiento de distintos algoritmos de control predictivo cuando son aplicados a procesos de dinámica rápida.

1.3.2. Objetivos específicos.

- Diseñar y simular un control predictivo usando el método de programación cuadrática aplicado a un sistema multivariable.
- Diseñar y simular un control predictivo usando el método de mínimos cuadrados iterativos aplicado a un sistema multivariable.
- Comparar los dos algoritmos de control con uno de control clásico implementándolos en un proceso real de motores acoplados.

1.4. DESARROLLO HISTÓRICO.

El Control Predictivo Basado en Modelos ([2] y [5]) (CPBM) desarrollado en los años sesenta ha generado un alto impacto en el ámbito industrial en el control de procesos. Ésta es una estrategia de control la cual se basa en la utilización de un modelo dinámico del proceso, con el fin de realizar predicciones de la evolución de sus diferentes variables controladas respecto a los valores futuros de sus variables manipuladas, todo esto a lo largo de un horizonte temporal finito. Con base en estas predicciones se calculan las acciones de control que hacen que las variables controladas del proceso converjan hacia sus valores predichos.

Las primeras implementaciones del control predictivo en el entorno industrial se realizaron a finales de los setenta con los métodos DMC y MPHC. El algoritmo llamado Dynamic Matrix Control [4] (DMC) emplea modelos expresados en forma matricial que contienen los coeficientes de respuesta al escalón de las variables controladas del proceso respecto a variaciones en cada una de sus variables manipuladas. Esta estrategia permite tener un

modelo más preciso del proceso ya que estos coeficientes se adquieren midiendo directamente las variables en el proceso real. Este modelo de respuesta al escalón es de tipo lineal y, en ausencia de restricciones las acciones de control se generan resolviendo un problema de mínimos cuadrados ordinarios. Además permite la inclusión de un modelo de perturbaciones y restricciones en las diferentes variables.

Por otra parte el método MPHC [12] Model Predictive Heuristic Control hace uso de un modelo de respuesta al impulso del proceso e incluye la generación de trayectorias de las referencias expresadas como sistemas de primer orden y, al igual que el DMC permite el uso de modelos de perturbaciones así como inclusión de restricciones en las variables del proceso. El cálculo de las acciones de control se realiza de manera iterativa.

El algoritmo denominado Generalized Predictive Control ([1], [3] y [5]) (GPC) desarrollado a finales de los ochenta es tal vez uno de los métodos de control predictivo que más éxito han tenido en el entorno académico. Éste está inspirado en el Controlador de Mínima Varianza Generalizado (GMV, Generalized Minimum Variance) que da una mayor robustez a los MBPC, permitiendo su aplicación en procesos con retardos variables o desconocidos, de fase no mínima, inestables o con modelos sobreparametrizados o no muy fieles al proceso real.

La filosofía de todos estos métodos es común. Consiste en calcular las acciones de control futuras de tal forma que se minimice una función de coste definida sobre un horizonte de predicción. Ésta función contiene un término cuadrático que pondera los errores entre las predicciones y sus referencias y, el esfuerzo de control que hace que las variables del proceso converjan hacia sus respectivas predicciones.

Gracias a la gran demanda de control predictivo en el sector industrial, los investigadores siguen desarrollado nuevos algoritmos que buscan mejorar

cada vez más las prestaciones de esta herramienta, para hacerla una de las opciones de control avanzado más ideales a la hora de automatizar.

1.5. ESTRUCTURA Y DESARROLLO DEL PROYECTO.

El proyecto consta de dos partes, una es la *memoria* y la otra son los *anexos*:

- La memoria se divide en cuatro capítulos detallados de la siguiente manera:

  El *primer capítulo* contiene la introducción, la justificación, los objetivos y el desarrollo histórico. En el *segundo capítulo* se hace una descripción de la metodología del control predictivo basado en modelos MBPC así como el método de la Matriz de Control Dinámica DMC. Posteriormente se aborda el manejo de restricciones en las variables del proceso mediante los algoritmos de programación cuadrática (QP) y mínimos cuadrados iterativos. Finalmente se explican las estrategias Blocking y Revisión de Horizontes, las cuales buscan disminuir la carga computacional en los algoritmos anteriormente mencionados. En el *tercer capítulo* se comparan ambos algoritmos a través de simulaciones en Matlab sobre un sistema multivariable. También se realizan comparaciones de los dos algoritmos con uno de control clásico implementados en un sistema real de motores acoplados. El *cuarto capítulo* contiene las conclusiones generales del proyecto.

- Los anexos:

  Éstos contienen datos tales como códigos en Matlab, gráficas tanto de simulaciones como de implementaciones e imágenes varias del proceso.

CAPÍTULO 2

MARCO TEÓRICO

Pág.

2.1. CONTROL PREDICTIVO BASADO EN MODELOS MBPC.

La estrategia del MBPC es la de resolver en cada periodo de muestreo un problema de optimización de control óptimo en lazo abierto para un horizonte finito, donde posteriormente se aplican las acciones de control que resultan de esta optimización, manteniéndolas hasta el siguiente intervalo de muestreo. Aunque las predicciones se realizan en lazo abierto, la estrategia las corrige midiendo el estado de las variables del proceso previamente a la etapa de optimización en la que se busca minimizar una función de coste, dando así un efecto de control realimentado.

Las predicciones de la evolución de cada una de las variables controladas del sistema, se realizan empleando un modelo matemático de éste y, basándose en la idea de horizonte móvil (receding Horizon); el MBPC realiza las predicciones en una ventana temporal de longitud previamente definida, donde posteriormente se lleva a cabo la minimización de una función de coste, se toma el primer elemento de cada vector de movimientos calculados para cada entrada, se aplica a su respectiva entrada y se desecha el resto. Ambas etapas se repiten en cada instante de muestreo.

La optimización se efectúa minimizando una función de coste en la que participan el error de predicción futuro (resultado de la diferencia entre las trayectorias de referencias de las variables controladas y sus respectivas predicciones) y las restricciones (en caso de que existan) en las distintas variables del proceso. La variable independiente de esta función de coste son las acciones de control futuras, de modo que los valores de éstas que minimizan dicha función son las acciones que se aplicaran a las entradas del proceso.

2.1.1. Metodología.

En la figura 2.1 se puede apreciar la configuración de un control predictivo, donde se pueden observar los pasos explicados anteriormente.

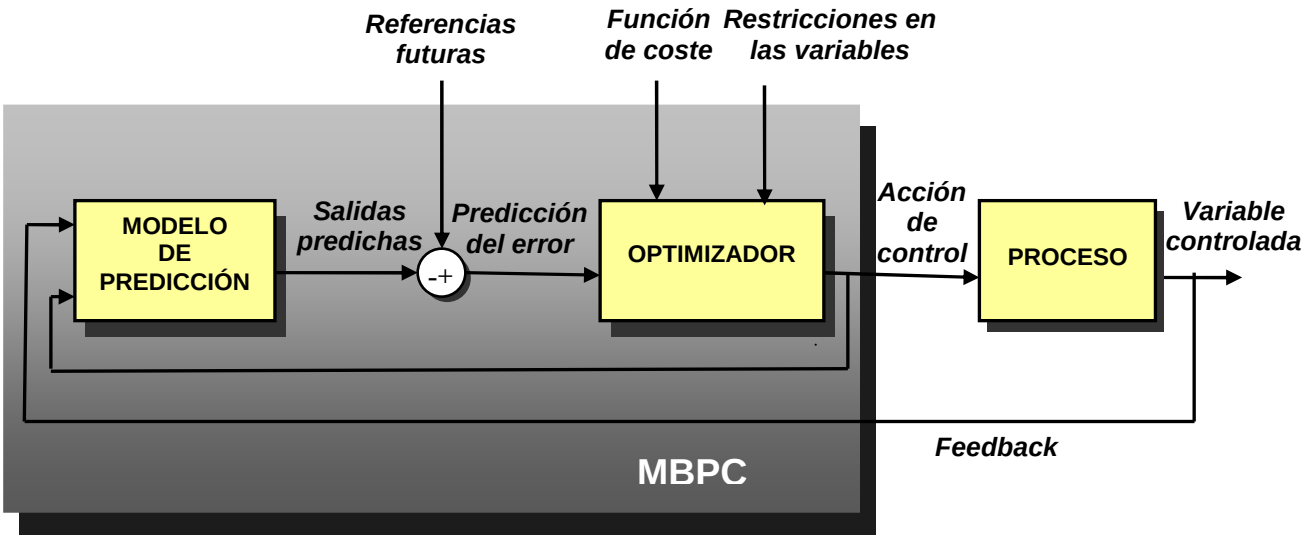


Fig. 2.1: Configuración del MBPC.

1. Se hace uso de un modelo del proceso para predecir su comportamiento futuro a lo largo de un horizonte de predicción de longitud N intervalos de muestreo (véase la figura 2.2). Los valores **y(k+i|k)** [1], para i=1,..,N, corresponden a las predicciones de las salidas para los instantes k+i (futuro), calculadas en el instante k (presente). Estas predicciones dependen tanto de las entradas y salidas anteriores como de las señales de control futuras de las cuales se desconocen sus valores. Estas señales de control **u(k+i|k)**, para i=1,…,N, están proyectadas a lo largo de un horizonte, las cuales permanecen constantes a partir de k+Nc.
2. Se define una trayectoria de referencias futuras **w(k+i|k)**, para i=1,…,N , la cual describe el comportamiento deseado del proceso o a qué estado se desea llevar. Ésta puede estar representada mediante una función de
3. cualquier orden o puede ser un valor fijo en el tiempo.

[1] Predicción de la salida en el instante k, para el instante k+i, i=1,..,N

4. Restar de los valores de la trayectoria de referencias futuras **w(k+i|k)** las predicciones **y(k+i|k)**, teniendo como resultado una ecuación del error de predicción la cual está en función de las acciones de control futuras **u(k+i|k)**.

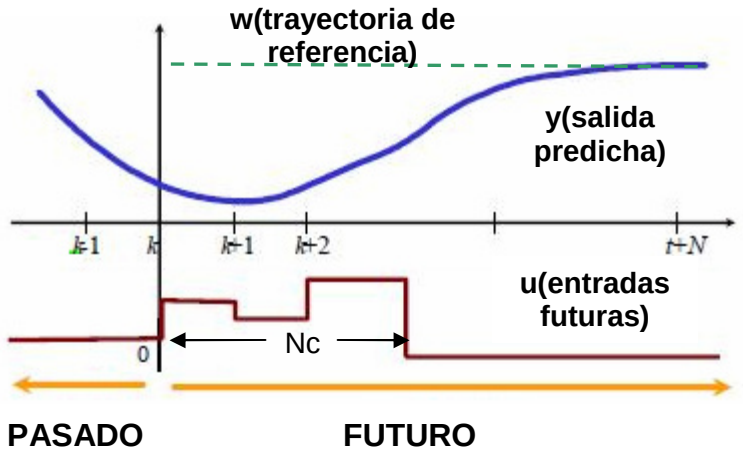


Fig. 2.2: Funcionamiento del MBPC.

5. Se dispone de una función de coste la cual incluye la suma de los cuadrados de los errores de predicción, en donde éstos son penalizados fuertemente; es decir que ésta obliga a mantener la predicción de la salida lo más cercana posible a la referencia previamente establecida. Dicha función adicionalmente suele incluir la suma de los cuadrados de las acciones de control, a fin de tomar en cuenta el esfuerzo de control.
6. La función de coste se introduce en un optimizador que puede incorporar restricciones para las distintas variables. El resultado de esta optimización es el vector de valores de acciones de control futuras. En caso de que existan restricciones de cualquier tipo, la optimización se efectúa empleando técnicas iterativas; y en caso de que éstas no existieran, la solución se obtiene analíticamente.
7. Se toma el primer elemento del vector de acciones de control, el cual corresponde a u(k|k) y se aplica a la entrada del proceso. Los primeros elementos de los vectores de predicciones y de acción de control son almacenados y los demás valores son desechados. Puesto que esta

estrategia hace uso de la idea de horizonte móvil, los pasos anteriores se repiten en el siguiente intervalo de muestreo, calculando de esta forma nuevas predicciones **y(k+1+i|k+1)** y el control u(k+1|k+1) que será diferente de u(k+1|k).

2.1.2. Modelo de predicción.

En el MPBC, este es tal vez el componente más crucial para realizar un control óptimo. Este es el resultado de la caracterización del proceso real y debe ser capaz de recoger toda la información posible de su dinámica, de tal manera que permita realizar predicciones lo más parecidas a la realidad.

Este componente puede incluir los efectos de las perturbaciones no medibles y errores de modelado, así como un modelo de perturbaciones medibles, de tal forma que permita de manera anticipada a sus acontecimientos efectuar los correctivos necesarios (feedforward). En caso de que ambos sean modelos lineales, se aplicaría el principio de superposición, sumando las salidas de ambos submodelos para llegar a una salida general (ver figura 2.3).

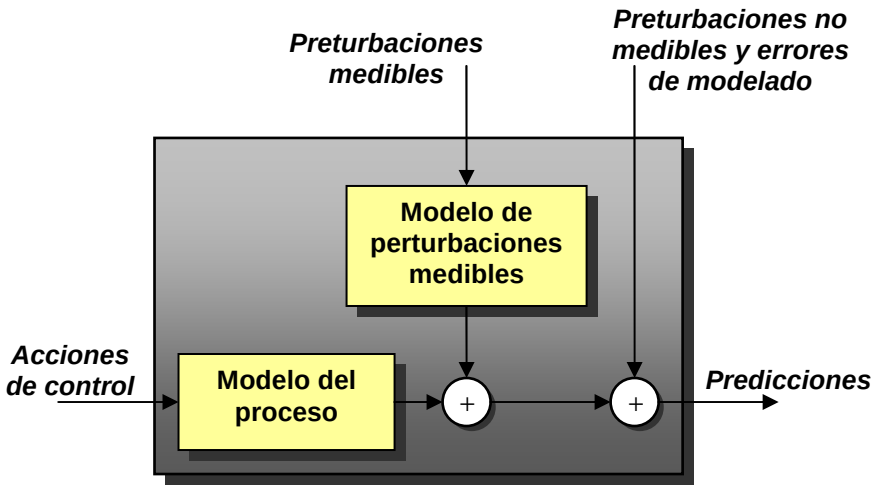


Fig. 2.3: Modelo de predicción.

2.1.2.1. Modelo del proceso: Las representaciones más comunes [5] de éste son las de respuesta al impulso, respuesta al escalón, función de transferencia y de espacio de estados.

- Modelo de respuesta al impulso: la relación entrada-salida de este modelo está dada por la ecuación 2.1.

$$y(t) = \sum_{k=1}^{\infty} h_k u(t-k) \qquad (2.1)$$

Los coeficientes h_k son valores muestreados del proceso luego de aplicársele un impulso unitario de ancho un intervalo de muestreo donde sólo se consideran N valores (ver figura 2.4).

$$y(t) = \sum_{k=1}^{N} h_k u(t-k) = H(z^{-1})u(t)$$

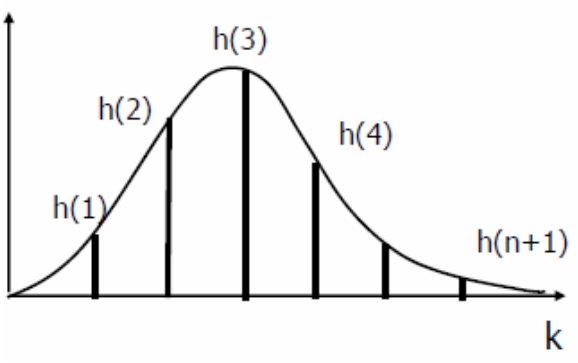


Fig. 2.4: Respuesta al impulso.

Donde $H(z^{-1}) = h_1 z^{-1} + h_2 z^{-2} + ... + h_N z^{-N}$ y z^{-1} representa el retardo unitario. Finalmente la predicción de la salida en el instante t, está dada por la ecuación:

$$y(t+j\,|\,t) = \sum_{k=1}^{N} h_k u(t+j-k\,|\,t) = H(z^{-1})u(t-k\,|\,t) \qquad (2.2)$$

Las ventajas de este modelo son su sencillez para la descripción de la dinámica de procesos ya sean de fase no mínima o con retardos y, el hecho de que no se requiera información previa. En cuanto a desventajas, éste no puede representar procesos inestables y su gran número de coeficientes resultados del muestreo que aparecen en el modelo.

- **Modelo de respuesta al escalón**: Tiene las mismas ventajas e inconvenientes que el modelo anterior. Su respuesta se observa en la figura 2.5 y la salida se representa mediante la siguiente ecuación:

$$y(t) = y_0 + \sum_{k=1}^{N} s_k \Delta u(t-k \mid t) = y_0 + S(z^{-1})(1-z^{-1})u(t)$$

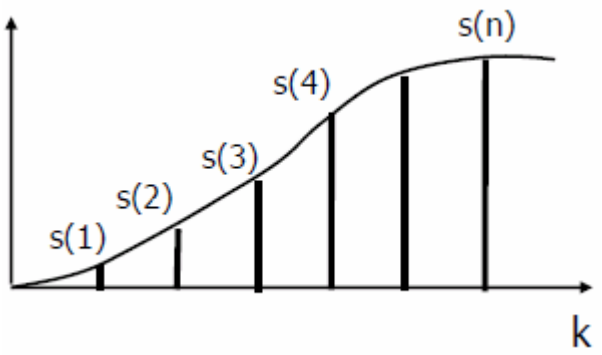


Fig. 2.5: Respuesta al Escalón.

Los coeficientes S_k son los valores medidos de la respuesta al escalón en cada intervalo de muestreo. La variable $\Delta u(t) = u(t) - u(t-1)$ indica el incremento en la acción de control en el instante t, e y_0 es el valor inicial de la salida. Normalmente se busca encontrar una expresión del incremento en la salida al realizar los movimientos $\Delta u(t)$ en la acción de control, por tanto se prescinde de y_0 para representar dicho efecto en la ecuación 2.3:

$$\Delta y(t+j \mid t) = \sum_{k=1}^{N} s_k \Delta u(t+j-k \mid t) \qquad (2.3)$$

- **Modelo de función de transferencia**: Se hace uso de un modelo discreto del proceso para obtener la salida dada por la ecuación:

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

Los coeficientes del numerador B y denominador A son:

$$A(z^{-1}) = 1 + a_1 z^{-2} + a_2 z^{-2} + \ldots + a_{na} z^{-na}$$
$$B(z^{-1}) = b_1 z^{-2} + b_2 z^{-2} + \ldots + b_{nb} z^{-nb} \qquad (2.4)$$

Finalmente mediante la ecuación 2.5 se calcula la predicción de la salida:

$$y(t+j\,|\,t)=\frac{B(z^{-1})}{A(z^{-1})}u(t+j\,|\,t) \tag{2.5}$$

El uso de este tipo de representación requiere de un buen de conocimiento del proceso en cuanto a su orden para la obtención de los polinomios A y B, puesto que se debe disponer de un modelo lo más preciso de éste. Como ventajas tiene que es útil para representar procesos inestables y el número de parámetros de los que dispone es mínimo.

- **Modelo de espacio de estados**: se representa de la siguiente forma:

$$x(t)=Ax(t-1)+Bu(t-1)$$
$$y(t)=Cx(t)$$

La variable x representa el estado y A, B y C son las matrices de estado del proceso. Por último las predicciones de la salida están dadas por:

$$y(t+k\,|\,t)=C\hat{x}(t+k\,|\,t)=C[A^{k}x(t)+\sum_{i=1}^{k}A^{i-1}Bu(t+k-i\,|\,t)] \tag{2.6}$$

Esta representación posee como ventaja la extensión al caso multivariable así como la de analizar la representación interna. Su principal desventaja es la necesidad de precisar en ocasiones de un observador del estado.

2.1.2.2. Modelo de perturbaciones.

Este componente hace parte del modelo de predicción general, y tiene una gran importancia dado que éste permite modelar perturbaciones ambientales, errores de modelado, etc. Uno de los modelos más empleados es el ARIMA (Autorregresivo Integrado de Media Móvil) [2]. En éste la representación de la diferencia entre la salida medida y la predicha se da por medio de la ecuación:

$$d(t)=\frac{C(z^{-1})}{D(z^{-1})}e(t)$$

Normalmente el polinomio del numerador es igual a uno, el polinomio del denominador incluye únicamente un integrador y el modelo está en función de $e(t)$, que es un ruido de media cero. Este modelo es apropiado cuando se presentan cambios aleatorios ocurridos en cualquier instante y, con la

inclusión del integrador se consigue un error nulo en régimen permanente. Finalmente el modelo de perturbación es el expresado en la ecuación:

$$d(t) = \frac{1}{1 - z^{-1}} e(t)$$

Y la predicción óptima será:

$$d(t+k \mid t) = d(t) \qquad (2.7)$$

2.1.3. Predicción de las salidas del proceso.

La predicción de la salida ([1] y [6]) suele estar formada por la suma de dos señales: la respuesta libre (depende de las entradas pasadas) y respuesta forzada (depende de las entradas futuras).

- Respuesta libre: Ésta depende de las acciones de control pasadas y representa la evolución futura del proceso si la entrada aplicada en el intervalo de muestreo anterior se mantiene constante (véase la figura 2.6).

Las entradas futuras son iguales a la entrada en el instante k-1:

$$ulibre(k-j) = u(k-j) \quad \text{para} \quad j = 0,1,2,...$$
$$ulibre(k+j) = u(k-1) \quad \text{para} \quad j = 0,1,2,...$$

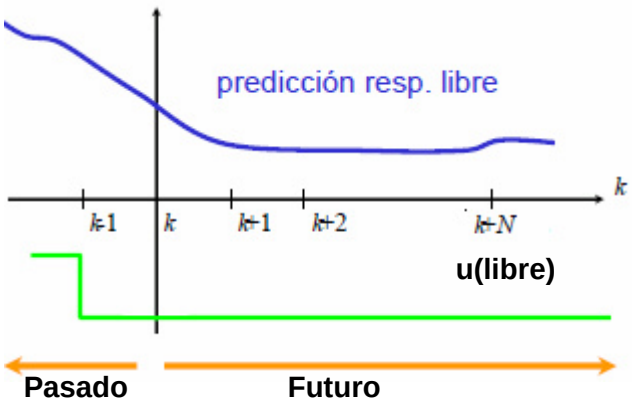


Fig. 2.6: Respuesta libre.

- Respuesta forzada: Ésta depende de los movimientos de control futuros (valores desconocidos) y mide la evolución del proceso debida a dichos movimientos (véase la figura 2.7).

$$uforzada(k-j)=0 \text{ para } j=0,1,2,...$$

$$uforzada(k+j)=u(k+j)-u(k-1) \text{ Para } j=0,1,2,...$$

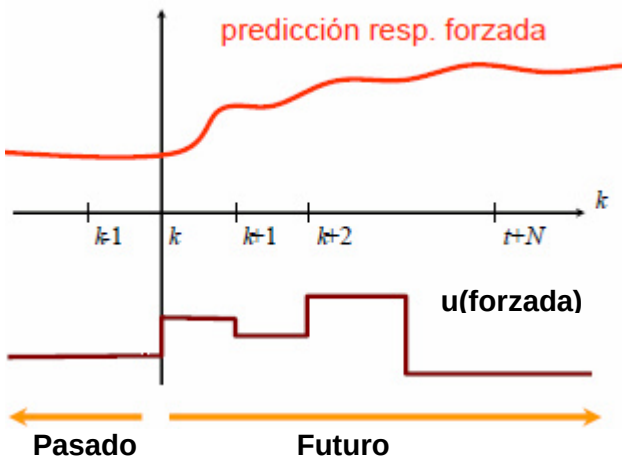


Fig. 2.7: Respuesta forzada.

Finalmente en la figura 2.8 se observa predicción de la salida (resultado de la suma de la respuesta libre y forzada) del proceso:

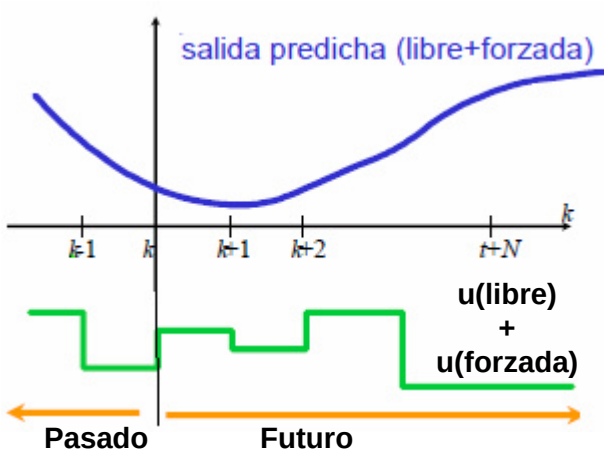


Fig. 2.8: Predicción de la salida.

2.1.4. Trayectoria de referencia.

La trayectoria de referencia [2] **R(k+i)** define la forma en la que se desea llevar la variable controlada del proceso desde una posición y(k) hasta una posición deseada **r(k+i)** (ver figura 2.9).

Ésta puede tener una aproximación de primer orden o bien puede ser constante en el tiempo tomando directamente el valor de r(k+i):

$$R(k) = y(k)$$
$$R(k+i) = \rho R(k+i-1) + (1-\rho)R(k+i) \quad \text{para } i = 1,2,... \qquad (2.8)$$

El parámetro ρ (varía entre 0 y 1) determina la aproximación de **R(k+i)**. En la figura 2.9 se puede contemplar que para valores grandes de ρ se consigue una transición suave (**R2(k+i)**), mientras que para valores pequeños ésta se torna más brusca (**R1(k+i)**).

Una de las ventajas de los MBPC es que al tener conocimiento de los valores futuros de la referencia, el sistema comienza a reaccionar antes de que los cambios en ésta se hayan realizado.

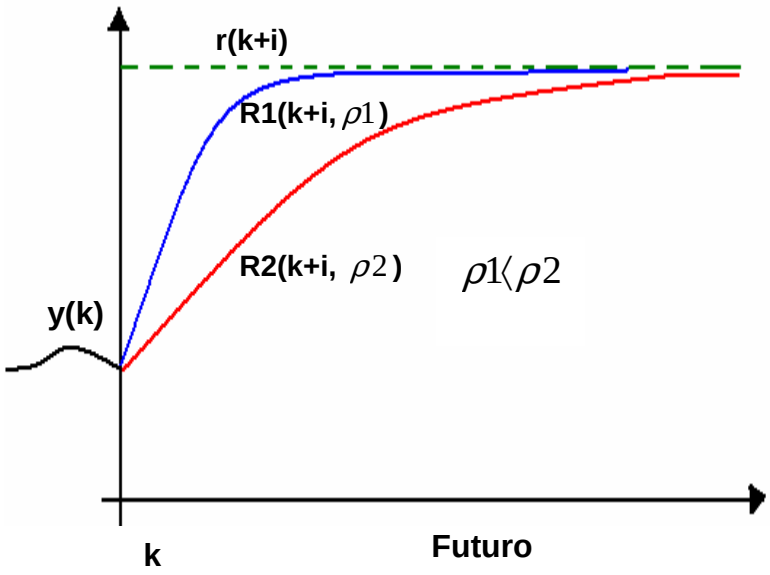


Fig. 2.9: Trayectoria de referencia.

2.1.5. Función de coste.

El propósito de dicha función [3], es el de obtener la ley de control a partir de su minimización. La mayoría de los MBPC emplean diferentes tipos de funciones de coste, siendo la expresión más extendida de ésta es la siguiente:

$$J(N_0, N_P, N_C) = \sum_{j=N_0}^{N_P} \alpha(j)[R(k+j) - y(k+j \mid k)]^2 + \sum_{j=1}^{N_C} \lambda(j)[\Delta u(k+j-1)]^2 \quad (2.9)$$

Se observa que el índice J, es una expresión cuadrática compuesta por la suma de dos términos:

- El primer término está relacionado con el error de predicción, resultado de restar las predicciones de la salida, a la trayectoria de referencia. N_P es el horizonte de predicción (instantes de muestreo) y N_0 determina a partir de qué instante se desea que la salida sea igual a la referencia.

- El segundo término es el asociado a los incrementos futuros $\Delta u(k+j-1)$ en las acciones de control. Estos incrementos se realizan a lo largo de un horizonte de control N_C (instantes de muestreo) en donde a partir de dicho horizonte, éstos serán nulos ($\Delta u(k+j-1) = 0 \quad j > N_C$); es decir que una vez superado el instante $(k+Nc)$ la acción de control permanece constante.

Los coeficientes $\alpha(j)$ y $\lambda(j)$, son secuencias que penalizan los errores de predicción y el esfuerzo de control respectivamente. Normalmente éstas son constantes o pueden tener un comportamiento exponencial $\beta(j) = \delta^{\text{N-j}}$. Para $0 < \delta < 1$ se castigará fuertemente los errores más lejanos al instante k, obteniendo de esta forma un control más suave y con un esfuerzo de control menor. Si $\delta > 1$ se penalizan fuertemente los errores al inicio y en consecuencia el control sería más brusco.

2.1.6. Ley de control.

La secuencia de movimientos futuros se calcula mediante un proceso de optimización. En éste se minimiza la función de coste y puede incorporar una serie de restricciones. Éstas suelen estar motivadas debido a limitaciones

físicas del proceso (velocidad de apertura de una válvula, temperatura máxima de un horno, velocidad de llenado de un tanque, etc.), por cuestiones de seguridad, económicas, etc. Las más comunes son las de límites mínimos y máximos para las variables del proceso y de velocidades de cambio en las variables controladas.

En caso de que no existan restricciones, la solución se obtiene analíticamente, y en caso contrario la optimización se efectúa empleando técnicas iterativas.

2.1.7. Ventajas y desventajas.

Entre las ventajas del MBPC sobresalen las siguientes:

- No maneja cálculos complejos, los conceptos resultan intuitivos y su sintonización es sencilla.
- Su aplicación a una amplia gama de procesos industriales que pueden presentar comportamientos dinámicos complicados como acoplamientos entre variables, retardos de transporte, inestabilidades o incluso restricciones en algunas de sus variables
- La facilidad de su extensión del caso SISO para controlar procesos multivariables.
- Permite incluir en el problema de control las restricciones a las que está sometido el proceso.
- Uso práctico en procesos en los que las referencias futuras son conocidas (robótica).
- Su robustez ante errores de modelado y presencia de perturbaciones no medibles.

Unos de sus principales inconvenientes son:

- Carga computacional elevada debido a la cantidad de operaciones que realiza, condicionando así su uso en procesos con dinámicas rápidas.
- Requiere de un esfuerzo adicional en ingeniería para obtener un buen modelo del proceso.

2.2. DYNAMIC MATRIX CONTROL DMC.

Como se vio en la sección 2.1.2.1., la mayoría de algoritmos de control predictivo se pueden diferenciar por el tipo de modelo usado para realizar las predicciones. En el caso del DMC ([4] y [6]), éste puede disponer de un modelo de respuesta al impulso o al escalón. En esta sección se detalla el funcionamiento de un DMC para un sistema SISO.

2.2.1. Modelo del proceso.

De acuerdo a la sección 2.1.2.1., la forma de obtener los coeficientes de la respuesta al escalón de manera experimental es la siguiente:

1. El sistema debe estar en un punto de operación estable (u_0, y_0).
2. En el instante k, se aplica una variación $\pm\Delta u$ a la entrada del proceso.
3. Se almacenan los valores medidos de la salida y_m a partir del instante de muestreo $k+1$ hasta que el sistema se estabilice (ver figura 2.5):

$$\begin{bmatrix} y_m(k+1) \\ y_m(k+2) \\ y_m(k+3) \\ \vdots \\ y_m(k+N) \\ \vdots \\ y_m(k+m) \end{bmatrix}$$

4. El vector de salidas almacenadas se redimensiona de tal forma que contenga los primeros N valores, donde N es el instante en el que el sistema ya se ha estabilizado y por lo tanto, para instantes posteriores a éste la salida es la misma:

$$\begin{bmatrix} y_m(k+1) \\ y_m(k+2) \\ y_m(k+3) \\ \vdots \\ y_m(k+N) \end{bmatrix}$$

5. Los coeficientes de respuesta al escalón S^u se obtienen restando del vector de salidas almacenadas la posición inicial y_0 y dividiendo posteriormente el resultado por el valor de la variación $\pm\Delta u$:

$$S^u = \begin{bmatrix} s^u(1) \\ s^u(2) \\ s^u(3) \\ \vdots \\ s^u(N) \end{bmatrix} = \left(\begin{bmatrix} y_m(k+1) \\ y_m(k+2) \\ y_m(k+3) \\ \vdots \\ y_m(k+N) \end{bmatrix} - \begin{bmatrix} y_0 \\ y_0 \\ y_0 \\ \vdots \\ y_0 \end{bmatrix} \right) \div (\pm \Delta u)$$

Este paso se realiza con el fin de obtener un modelo que permita calcular la contribución Δy a la salida del proceso debido a una variación Δu en la entrada de éste.

$$S^u = \begin{bmatrix} s^u(1) \\ s^u(2) \\ s^u(3) \\ \vdots \\ s^u(N) \end{bmatrix} \tag{2.10}$$

2.2.2. Actualización de la respuesta libre: cálculo recursivo.

Cuando el algoritmo DMC se ejecuta por primera vez, la respuesta libre se inicializa con el valor medido de la salida $y_m(k)$ (ver ecuación 2.12). Como se vio en la sección 2.1.3, ésta depende únicamente de las entradas pasadas. Su cálculo para el instante k se hace de manera recursiva usando la respuesta libre del el instante $k-1$ desplazándola una fila y repitiendo el último término:

$$\underbrace{\begin{bmatrix} L(k|k) \\ L(k+1|k) \\ L(k+2|k) \\ \vdots \\ L(k+N-1|k) \end{bmatrix}}_{N\times 1} = Q1 \cdot \underbrace{\begin{bmatrix} L(k|k-1) \\ L(k+1|k-1) \\ L(k+2|k-1) \\ \vdots \\ L(k+N-1|k-1) \end{bmatrix}}_{N\times 1} + \underbrace{\begin{bmatrix} s^u(1) \\ s^u(2) \\ s^u(3) \\ \vdots \\ s^u(N) \end{bmatrix}}_{N\times 1} \cdot \underbrace{\Delta u(k|k-1)}_{1\times 1} \tag{2.11}$$

$Q1$ es una matriz de dimensiones $N \times N$, y se encarga de desplazar la respuesta libre del instante anterior, repetir su último elemento y es de la forma:

$$Q1=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ 0 & 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \vdots & \cdots & 1\\ 0 & 0 & 0 & \cdots & 1\end{bmatrix}$$

$$\begin{bmatrix}L(k\,|\,k)\\ L(k+1\,|\,k)\\ L(k+2\,|\,k)\\ \vdots\\ L(k+N-1\,|\,k)\end{bmatrix}=\begin{bmatrix}y_m(k)\\ y_m(k)\\ y_m(k)\\ \vdots\\ y_m(k)\end{bmatrix} \qquad (2.12)$$

2.2.3. Respuesta forzada.

Esta representa la contribución a la salida causada por los movimientos futuros Δu en la entrada. Estos movimientos están proyectados a lo largo de un horizonte de control N_C donde para instantes posteriores a éste las variaciones son nulas:

$$\underbrace{\begin{bmatrix}F(k+1\,|\,k)\\ F(k+2\,|\,k)\\ F(k+3\,|\,k)\\ \vdots\\ F(k+N_P\,|\,k)\end{bmatrix}}_{F_{N_P\times 1}}=\underbrace{\begin{bmatrix}s^u(1) & 0 & 0 & \cdots & 0\\ s^u(2) & s^u(1) & 0 & \cdots & 0\\ s^u(3) & s^u(2) & s^u(1) & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ s^u(N_P) & s^u(N_P-1) & s^u(N_P-2) & \cdots & s^u(N_P-N_C+1)\end{bmatrix}}_{G_{N_P\times N_C}}\cdot\underbrace{\begin{bmatrix}\Delta u(k\,|\,k)\\ \Delta u(k+1\,|\,k)\\ \vdots\\ \Delta u(k+N_C-1\,|\,k)\end{bmatrix}}_{\Delta u_{N_C\times 1}} \qquad (2.13)$$

La matriz G es la denominada matriz dinámica y se construye con los primeros N_P coeficientes de la respuesta al escalón. La multiplicación de cada columna de G por su correspondiente variación Δu, indica la contribución de dicha variación a la evolución del proceso.

2.2.4. Perturbaciones.

- **Perturbaciones medibles**: El algoritmo DMC considera que éstas se mantienen constantes en el futuro:

$$\Delta d(k+1\,|\,k)=\Delta d(k+2\,|\,k)=\cdots=\Delta d(k+p\,|\,k)=0$$

donde para el instante k la variación en ésta se puede calcular como:

$$\Delta d(k\,|\,k)=d(k)-d(k-1)$$

Finalmente con un modelo de respuesta al escalón para las perturbaciones medibles, se pueden realizar predicciones del efecto en la salida causado por dichas variaciones:

$$\begin{bmatrix} m(k+1|k) \\ m(k+2|k) \\ m(k+3|k) \\ \vdots \\ m(k+N_P|k) \end{bmatrix} = \begin{bmatrix} s^d(1) \\ s^d(2) \\ s^d(3) \\ \vdots \\ s^d(N_P) \end{bmatrix} \cdot \Delta d(k|k) = S^d \Delta d \qquad (2.14)$$

- **Perturbaciones no medibles**: Se consideran constantes en el futuro

$$w(k|k) = w(k+1|k) = w(k+2|k) \cdots = w(k+p|k)$$

para el instante k se pueden estimar restando el primer elemento de la respuesta libre al valor medido de ésta:

$$\begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ w(k+3|k) \\ \vdots \\ w(k+N_P|k) \end{bmatrix} = \begin{bmatrix} y_m(k) \\ y_m(k) \\ y_m(k) \\ \vdots \\ y_m(k) \end{bmatrix} - \begin{bmatrix} F(k|k) \\ F(k|k) \\ F(k|k) \\ \vdots \\ F(k|k) \end{bmatrix} \qquad (2.15)$$

2.2.5. Predicción de la salida.

La predicción de la evolución del proceso para un horizonte de predicción N_P, se realiza mediante la suma de cuatro efectos, los cuales corresponden a las ecuaciones (2.11), (2.13), (2.14) y (2.15):

$$\begin{matrix} \textit{Salida} \\ \textit{predicha} \end{matrix} = \begin{matrix} \textit{Respuesta} \\ \textit{libre} \end{matrix} + \begin{matrix} \textit{Respuesta} \\ \textit{forzada} \end{matrix} + \begin{matrix} \textit{Efecto de las} \\ \textit{perturbaciones} \\ \textit{no medibles} \end{matrix} + \begin{matrix} \textit{Efecto de las} \\ \textit{perturbaciones} \\ \textit{medibles} \end{matrix}$$

$$\begin{bmatrix} Y_P(k+1|k) \\ Y_P(k+2|k) \\ Y_P(k+3|k) \\ \vdots \\ Y_P(k+N_P|k) \end{bmatrix} = Q2 \cdot \begin{bmatrix} L(k|k) \\ L(k+1|k) \\ L(k+2|k) \\ \vdots \\ L(k+N-1|k) \end{bmatrix} + \begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ \vdots \\ F(k+N_P|k) \end{bmatrix} + \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ w(k+3|k) \\ \vdots \\ w(k+N_P|k) \end{bmatrix} + \begin{bmatrix} m(k+1|k) \\ m(k+2|k) \\ m(k+3|k) \\ \vdots \\ m(k+N_P|k) \end{bmatrix} \qquad (2.16)$$

La matriz $Q2$ de la ecuación 2.16, se encarga de extraer los primeros N_P elementos de la respuesta libre a partir de su segunda fila. Ésta es de dimensiones $N_P \times N$ y es de la forma:

a) si $N_P > N$, desplaza y repite el último elemento hasta llegar a N_P.

b) si $N_P = N$, desplaza y repite el último elemento.

c) si $N_P < N$, desplaza y toma los primeros N_P elementos.

$$Q2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad Q2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & \cdots & & \ddots & 0 & \vdots \\ 0 & 0 & \cdots & & & \ddots & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \quad Q2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \ddots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 1 & 0 & \cdots & 0 \end{bmatrix}$$

a) b) c)

2.2.6. Trayectoria de referencia.

La trayectoria de referencia puede ser un valor constante o bien puede adoptar la forma que se describe en la ecuación 2.8. Su vector final será de N_P coeficientes:

$$R = \begin{bmatrix} R(k+1) \\ R(k+2) \\ R(k+3) \\ \vdots \\ R(k+N_P) \end{bmatrix} \tag{2.17}$$

2.2.7. Función de coste.

Como primera medida se calcula el error de predicción restando las predicciones de la salida (ecuación 2.16) a la trayectoria de referencia (ecuación 2.17):

$$\begin{bmatrix} E_P(k+1|k) \\ E_P(k+2|k) \\ E_P(k+3|k) \\ \vdots \\ E_P(k+N_P|k) \end{bmatrix} = \begin{bmatrix} R(k+1) \\ R(k+2) \\ R(k+3) \\ \vdots \\ R(k+N_P) \end{bmatrix} - \begin{bmatrix} Y_P(k+1|k) \\ Y_P(k+2|k) \\ Y_P(k+3|k) \\ \vdots \\ Y_P(k+N_P|k) \end{bmatrix}$$

$$\begin{bmatrix} E_P(k+1|k) \\ E_P(k+2|k) \\ E_P(k+3|k) \\ \vdots \\ E_P(k+N_P|k) \end{bmatrix} = \begin{bmatrix} R(k+1) \\ R(k+2) \\ R(k+3) \\ \vdots \\ R(k+N_P) \end{bmatrix} - Q2 \cdot \begin{bmatrix} L(k|k) \\ L(k+1|k) \\ L(k+2|k) \\ \vdots \\ L(k+N-1|k) \end{bmatrix} - \begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ \vdots \\ F(k+N_P|k) \end{bmatrix} - \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ w(k+3|k) \\ \vdots \\ w(k+N_P|k) \end{bmatrix} - \begin{bmatrix} m(k+1|k) \\ m(k+2|k) \\ m(k+3|k) \\ \vdots \\ m(k+N_P|k) \end{bmatrix} \quad (2.18)$$

Reagrupando los sumandos de la ecuación 2.18 separando los términos conocidos de los desconocidos:

$$\underbrace{\begin{bmatrix} E_P(k+1|k) \\ E_P(k+2|k) \\ E_P(k+3|k) \\ \vdots \\ E_P(k+N_P|k) \end{bmatrix}}_{E_P} = \underbrace{\begin{bmatrix} R(k+1) \\ R(k+2) \\ R(k+3) \\ \vdots \\ R(k+N_P) \end{bmatrix} - Q2 \cdot \begin{bmatrix} L(k|k) \\ L(k+1|k) \\ L(k+2|k) \\ \vdots \\ L(k+N-1|k) \end{bmatrix} - \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ w(k+3|k) \\ \vdots \\ w(k+N_P|k) \end{bmatrix} - \begin{bmatrix} m(k+1|k) \\ m(k+2|k) \\ m(k+3|k) \\ \vdots \\ m(k+N_P|k) \end{bmatrix}}_{\text{Valores conocidos: } D} - \underbrace{\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ \vdots \\ F(k+N_P|k) \end{bmatrix}}_{G \cdot \Delta u} \quad (2.19)$$

Expresando la función de coste (ecuación 2.9) en forma maticial [1]:

$$\min_{\Delta u} J = \min_{\Delta u} [E_P{}^T \alpha E_P + \Delta u^T \lambda \Delta u]$$

$$\min_{\Delta u} J = \min_{\Delta u} [(D - G\Delta u)^T \alpha (D - G\Delta u) + \Delta u^T \lambda \Delta u] \quad (2.20)$$

Las matrices α y λ contienen los coeficientes que ponderan los errores de predicción y el esfuerzo de control respectivamente. Sus estructuraciones son las siguientes:

$$\alpha = \begin{bmatrix} \alpha_1 & 0 & 0 & 0 \\ 0 & \alpha_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \alpha_{N_P} \end{bmatrix} \lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_{N_C} \end{bmatrix} \quad (2.21)$$

2.2.8. Ley de control.

Si no existen restricciones de ningún tipo para las variables del proceso, los movimientos de control se pueden calcular derivando la función de coste 2.20 e igualándola a cero, obteniendo así una solución analítica [1]:

$$\frac{\partial J}{\partial \Delta u} = 2G^T \alpha (G\Delta u - D) + 2\lambda \Delta u = 0$$

$$\Delta u = (G^T \alpha G + \lambda)^{-1} G^T \alpha D$$

$$\Delta u = M \cdot D \tag{2.22}$$

$$M = (G^T \alpha G + \lambda)^{-1} G^T \alpha \qquad \text{Constante para todo k} \tag{2.23}$$

La solución es un mínimo único por ser J una función cuadrática y por tener el Hessiano definido positivo, siempre que λ sea positiva:

$$\text{Hessiano:} \quad \frac{\partial^2 J}{\partial \Delta u^2} = 2(G^T \alpha G + \lambda)$$

En caso de existencia de restricciones, la ecuación 2.20 se minimiza mediante el uso de técnicas iterativas como la programación cuadrática o lineal.

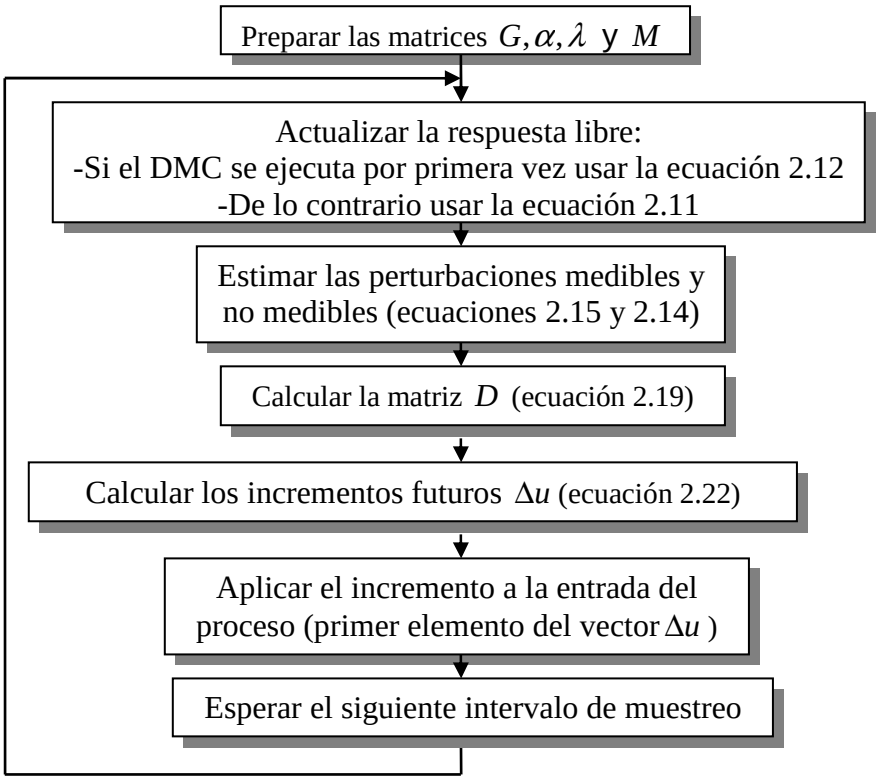


Fig. 2.10: Diagrama de flujo de la implementación de un DMC.

2.3. MANEJO DE RESTRICCIONES.

Éstas suelen estar motivadas debido a limitaciones físicas del proceso (velocidad de apertura de una electroválvula, temperatura máxima de un horno, velocidad de llenado de un tanque, etc.), por cuestiones de seguridad, económicas, etc.

Entre las más comunes y que se tratan en este apartado se tienen tres tipos:

- Limitaciones en la velocidad de variación de la acción de control:

$$\Delta u_{\min} \leq \Delta u \leq \Delta u_{\max}$$

- Limitaciones en las magnitudes máximas y mínimas de la acción de control:

$$u_{\min} \leq u \leq u_{\max}$$

- Limitaciones en las magnitudes máximas y mínimas permitidas para las salidas:

$$Y_{\min} \leq Y_P \leq Y_{\max}$$

Su inclusión en el modelo de predicción exige un replanteamiento del problema de optimización.

2.3.1. Quadratic Dynamic Matrix Control QDMC.

Es uno de los métodos más utilizados por la mayoría de los MBPC para la resolución de la ecuación de coste 2.20 sujeta a restricciones, usando *Programación Cuadrática* ([6] y [9]). Su uso hace necesario reconfigurar dicha ecuación para llevarla a la siguiente expresión:

$$J_{QP} = \frac{1}{2} x^T H x + C^T x \ ; \text{ Sujeto a las restricciones: } \ Ax \leq B$$

Es decir:

$$J_{QP} = \frac{1}{2} \Delta u^T H \Delta u + C^T \Delta u \qquad (2.24)$$

Sujeto a:

$$A\Delta u \leq B \qquad (2.25)$$

Reconfigurando la ecuación 2.20:

$$J = (D - G\Delta u)^T \alpha (D - G\Delta u) + \Delta u^T \lambda \Delta u$$
$$J = (D^T - \Delta u^T G^T) \alpha (D - G\Delta u) + \Delta u^T \lambda \Delta u$$
$$J = D^T \alpha D - D^T \alpha G \Delta u - \Delta u^T G^T \alpha D + \Delta u^T G^T \alpha G \Delta u + \Delta u^T \lambda \Delta u$$
$$J = \Delta u^T (G^T \alpha G + \lambda) \Delta u - 2 D^T \alpha G \Delta u + D^T \alpha D$$
$$J = \Delta u^T (G^T \alpha G + \lambda) \Delta u - 2 D^T \alpha G \Delta u + D^T \alpha D$$
$$J_{QP} = \frac{1}{2} J$$
$$J_{QP} = \frac{1}{2} \Delta u^T (G^T \alpha G + \lambda) \Delta u - D^T \alpha G \Delta u + \frac{1}{2} D^T \alpha D$$
$$J_{QP} = \frac{1}{2} \Delta u^T (G^T \alpha G + \lambda) \Delta u - D^T \alpha G \Delta u + cte$$

Se prescinde del término $\frac{1}{2} D^T \alpha D$ que no influye en el óptimo puesto que no depende de Δu , obteniendo finalmente:

$$J_{QP} = \frac{1}{2} \Delta u^T \underbrace{(G^T \alpha G + \lambda)}_{H} \Delta u + \underbrace{(-G^T \alpha D)^T}_{C^T} \Delta u \qquad (2.26)$$

Como se puede apreciar, la ecuación 2.26 es igual a la 2.24. La matriz H es constante y la matriz C^T varía en cada iteración. Las ecuaciones J y J_{QP} tienen el mismo mínimo y Hessiano, esto significa que el problema sigue siendo un problema de optimización convexa, lo cual garantiza un mínimo global único; por lo tanto el algoritmo converge (si existe solución Δu que satisfaga las restricciones).

La desigualdad matricial 2.25 se forma a partir de la inclusión de seis submatrices:

$$A\Delta u \leq B \quad = \quad \begin{bmatrix} A1 \\ A2 \\ A3 \end{bmatrix} \cdot \Delta u \leq \begin{bmatrix} B1 \\ B2 \\ B3 \end{bmatrix} \qquad (2.27)$$

2.3.1.1. Restricciones duras.

Este tipo de restricciones siempre deben cumplirse y están mayormente asociadas a las variables manipuladas. Ocasionalmente se aplican a las variables controladas por motivos de seguridad.

- **Restricciones para los incrementos en la acción de control.**

Para la existencia de restricciones en la velocidad de cambio de la acción de control para todo el horizonte N_C se debe cumplir:

$$\Delta u_{\min} \leq \Delta u \leq \Delta u_{\max}$$

También se puede escribir de la forma:

$$\begin{aligned} \Delta u &\leq \Delta u_{\max} \\ -\Delta u &\leq -\Delta u_{\min} \end{aligned} \tag{2.28}$$

Recordando que:

$$\Delta u = \begin{bmatrix} \Delta u(k \mid k) \\ \Delta u(k+1 \mid k) \\ \vdots \\ \Delta u(k+N_C-1 \mid k) \end{bmatrix}$$

Se pueden expresar las desigualdades 2.28 en forma matricial, llegando a una expresión en términos de las matrices $A1$ y $B1$:

$$\underbrace{\begin{bmatrix} I \\ -I \end{bmatrix}}_{A1} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix}}_{\Delta u} \leq \underbrace{\begin{bmatrix} \Delta u_{\max 1} \\ \Delta u_{\max 2} \\ \vdots \\ \Delta u_{\max N_C} \\ -\Delta u_{\min 1} \\ -\Delta u_{\min 2} \\ \vdots \\ -\Delta u_{\min N_C} \end{bmatrix}}_{B1} \tag{2.29}$$

Donde I es una matriz identidad de dimensiones $N_C \times N_C$.

- **Restricciones de magnitud de la acción de control.**

Cuando existen restricciones de magnitud de las acciones de control futuras u, se debe cumplir:

$$u_{\min} \leq u \leq u_{\max} \tag{2.30}$$

O escritas de la forma:

$$\begin{aligned} u &\le u_{\max} \\ -u &\le -u_{\min} \end{aligned} \tag{2.31}$$

Las acciones de control futuras u, para un horizonte de control N_C, dependen de los incrementos futuros Δu y de la entrada aplicada en el instante anterior $u(k-1)$:

$$u = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_C-1) \end{bmatrix} = \begin{bmatrix} \Delta u(k|k) + u(k-1) \\ \Delta u(k+1|k) + \Delta u(k|k) + u(k-1) \\ \vdots \\ \Delta u(k+N_C-1|k) + \ldots + \Delta u(k|k) + u(k-1) \end{bmatrix}$$

$$u = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_C-1) \end{bmatrix} = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) + \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_C-1|k) + \ldots + \Delta u(k|k) \end{bmatrix} + \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix}$$

De esta manera que reemplazando u en la inecuación 2.30 se tendría:

$$\begin{bmatrix} u_{\min 1} \\ u_{\min 2} \\ \vdots \\ u_{\min N_C} \end{bmatrix} \le \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) + \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_C-1|k) + \ldots + \Delta u(k|k) \end{bmatrix} + \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \le \begin{bmatrix} u_{\max 1} \\ u_{\max 2} \\ \vdots \\ u_{\max N_C} \end{bmatrix}$$

Llevando la expresión anterior a la forma de las inecuaciones 2.31:

$$\begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) + \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_C-1|k) + \ldots + \Delta u(k|k) \end{bmatrix} + \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \le \begin{bmatrix} u_{\max 1} \\ u_{\max 2} \\ \vdots \\ u_{\max N_C} \end{bmatrix}$$

$$-\begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) + \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_C-1|k) + \ldots + \Delta u(k|k) \end{bmatrix} - \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \le -\begin{bmatrix} u_{\min 1} \\ u_{\min 2} \\ \vdots \\ u_{\min N_C} \end{bmatrix}$$

Reordenando las desigualdades anteriores se llega a una expresión en términos de las matrices $A2$ y $B2$:

$$\underbrace{\begin{bmatrix} I_L \\ -I_L \end{bmatrix}}_{A2} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix}}_{\Delta u} \leq \underbrace{\begin{bmatrix} u_{\max 1} \\ u_{\max 2} \\ \vdots \\ u_{\max N_C} \\ -u_{\min 1} \\ -u_{\min 2} \\ \vdots \\ -u_{\min N_C} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \cdot u(k-1)}_{B2} \tag{2.32}$$

la matriz I_L es una matriz triangular inferior de unos y ceros y sus dimensiones son $N_C \times N_C$:

$$I_L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

- **Restricciones de magnitud para las predicciones de la salida.**

Retomando la ecuación 2.16 de predicciones de la salida para un horizonte N_P:

$$\underbrace{\begin{bmatrix} Y_P(k+1|k) \\ Y_P(k+2|k) \\ Y_P(k+3|k) \\ \vdots \\ Y_P(k+N_P|k) \end{bmatrix}}_{Y_P} = \underbrace{Q2 \cdot \begin{bmatrix} L(k|k) \\ L(k+1|k) \\ L(k+2|k) \\ \vdots \\ L(k+N-1|k) \end{bmatrix} + \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ w(k+3|k) \\ \vdots \\ w(k+N_P|k) \end{bmatrix} + \begin{bmatrix} m(k+1|k) \\ m(k+2|k) \\ m(k+3|k) \\ \vdots \\ m(k+N_P|k) \end{bmatrix}}_{T} + \underbrace{\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ \vdots \\ F(k+N_P|k) \end{bmatrix}}_{G\Delta u} \tag{2.33}$$

Las predicciones a lo largo de dicho horizonte no pueden infringir las restricciones de límites máximo y mínimo permitidos:

$$Y_{\min} \leq Y_P \leq Y_{\max}$$

$$Y_{\min} \le T + G\Delta u \le Y_{\max}$$

Escritas de la forma:

$$\begin{aligned} T + G\Delta u &\le Y_{\max} \\ -T - G\Delta u &\le -Y_{\min} \end{aligned}$$

$$\begin{aligned} G\Delta u &\le Y_{\max} - T \\ -G\Delta u &\le -Y_{\min} + T \end{aligned} \tag{2.34}$$

Se puede llegar a una expresión para matrices $A3$ y $B3$:

$$\underbrace{\begin{bmatrix} G \\ -G \end{bmatrix}}_{A3} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix}}_{\Delta u} \le \underbrace{\begin{bmatrix} y_{\max 1} \\ y_{\max 2} \\ \vdots \\ y_{\max N_P} \\ -y_{\min 1} \\ -y_{\min 2} \\ \vdots \\ -y_{\min N_P} \end{bmatrix} + \begin{bmatrix} -T(k+1\,|\,k) \\ -T(k+2\,|\,k) \\ \vdots \\ -T(k+N_P\,|\,k) \\ T(k+1\,|\,k) \\ T(k+2\,|\,k) \\ \vdots \\ T(k+N_P\,|\,k) \end{bmatrix}}_{B3} \tag{2.35}$$

Con las matrices $A1$, $A2$, $A3$, $B1$, $B2$ y $B3$ generadas, se construye la expresión general para las restricciones del algoritmo QP:

$$A\Delta u \le B \quad = \quad \begin{bmatrix} A1 \\ A2 \\ A3 \end{bmatrix} \cdot \Delta u \le \begin{bmatrix} B1 \\ B2 \\ B3 \end{bmatrix}$$

Donde:

Δu: es de dimensiones $N_C \times 1$

A: es de dimensiones $(4N_C + 4N_P) \times N_C$ y permanece constante en cada iteración.

B: es de dimensiones $(4N_C + 4N_P) \times 1$, $B1$ es constante y, $B2$ y $B3$ cambian en cada iteración.

$$\underbrace{\begin{bmatrix} I \\ -I \\ I_L \\ -I_L \\ G \\ -G \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix}}_{\Delta u} \leq \underbrace{\begin{bmatrix} \Delta u_{\max 1} \\ \Delta u_{\max 2} \\ \vdots \\ \Delta u_{\max N_C} \\ -\Delta u_{\min 1} \\ -\Delta u_{\min 2} \\ \vdots \\ -\Delta u_{\min N_C} \\ u_{\max 1} - u(k-1) \\ u_{\max 2} - u(k-1) \\ \vdots \\ u_{\max N_C} - u(k-1) \\ -u_{\min 1} + u(k-1) \\ -u_{\min 2} + u(k-1) \\ \vdots \\ -u_{\min N_C} + u(k-1) \\ y_{\max 1} - T(k+1|k) \\ y_{\max 2} - T(k+1|k) \\ \vdots \\ y_{\max N_P} - T(k+1|k) \\ -y_{\min 1} + T(k+1|k) \\ -y_{\min 2} + T(k+1|k) \\ \vdots \\ -y_{\min N_P} + T(k+1|k) \end{bmatrix}}_{B} \tag{2.36}$$

2.3.1.2. Restricciones blandas.

En ocasiones no existe la necesidad de definir restricciones duras para las variables controladas. Es decir, se permite violar ligeramente los límites de las variables controladas (restricciones blandas). Para esto se hace necesaria la definición de variables de holgura o de sobrepaso. El uso de este tipo de restricciones puede servir para evitar problemas de no factibilidad (cuando no existe solución alguna que satisfaga las restricciones duras).

Los límites para la variable controlada se definen de la siguiente manera:

$$Y_{\min} - \varepsilon \le Y_{N_P} \le Y_{\max} + \varepsilon$$
$$\varepsilon \ge 0$$

ε es una variable de holgura que debe incluirse en la ecuación de coste 2.20.

$$J = \min_{\varepsilon,\Delta u}[(D - G\Delta u)^T \alpha(D - G\Delta u) + \Delta u^T \lambda \Delta u + \varepsilon^T \rho \varepsilon] \quad (2.37)$$

La constante de ponderación ρ busca penalizar los valores grandes de ε. Es decir, es mantenido en cero mientras no exista violación de restricciones y, es fuertemente penalizado cuando existe una violación, obligando a ε a tomar un valor pequeño o suficiente para evitar el problema de no factibilidad. Obteniendo una nueva expresión de la ecuación (2.26) a partir de la (2.37) se llega a:

$$J_{QP} = \frac{1}{2}\Delta u^T (G^T \alpha G + \lambda)\Delta u + (-G^T \alpha D)^T \Delta u + \frac{1}{2}\varepsilon^T \rho \varepsilon$$

Reordenando términos se deduce la ecuación matricial para expresar la función de coste final:

$$J_{QP} = \frac{1}{2}\begin{bmatrix}\Delta u \\ \varepsilon\end{bmatrix}^T \begin{bmatrix}(G^T \alpha G + \lambda) & I1^T \\ I1 & \rho\end{bmatrix}\begin{bmatrix}\Delta u \\ \varepsilon\end{bmatrix} + \begin{bmatrix}(-G^T \alpha D) \\ 0\end{bmatrix}^T \begin{bmatrix}\Delta u \\ \varepsilon\end{bmatrix} \quad (2.38)$$
$$= \frac{1}{2} x^T H x + C^T x$$

Donde:

$x = \begin{bmatrix}\Delta u \\ \varepsilon\end{bmatrix}$; $H = \begin{bmatrix}(G^T \alpha G + \lambda) & I1^T \\ I1 & \rho\end{bmatrix}$; $C^T = \begin{bmatrix}(-G^T \alpha D) \\ 0\end{bmatrix}^T$; $I1$ es un vector de ceros de dimensiones $1 \times N_c$.

Las nuevas matrices $A3$ y $B3$ se calculan de la siguiente manera:

$$Y_{\min} - \varepsilon \le Y_{N_P} \le Y_{\max} + \varepsilon$$
$$Y_{\min} - \varepsilon \le T + G\Delta u \le Y_{\max} + \varepsilon$$

$$G\Delta u - \varepsilon \le Y_{\max} - T$$
$$-G\Delta u - \varepsilon \le -Y_{\min} + T \quad (2.39)$$

Finalmente con las desigualdades 2.39 se genera una expresión en términos de las matrices $A3$ y $B3$:

$$\begin{bmatrix} G & -Ib \\ -G & -Ib \end{bmatrix} \cdot \begin{bmatrix} \Delta u \\ \varepsilon \end{bmatrix} \leq \begin{bmatrix} Y_{\max} - T \\ -Y_{\min} + T \end{bmatrix}$$

Ib es un vector de unos, de dimensiones $N_P \times 1$ y las restricciones generales: $Ax \leq B = \begin{bmatrix} A1 \\ A2 \\ A3 \end{bmatrix} \cdot x \leq \begin{bmatrix} B1 \\ B2 \\ B3 \end{bmatrix}$

$$\underbrace{\begin{bmatrix} I & Ia \\ -I & Ia \\ I_L & Ia \\ -I_L & Ia \\ G & -Ib \\ -G & -Ib \\ Ia^T & -1 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \\ \varepsilon \end{bmatrix}}_{x} \leq \underbrace{\begin{bmatrix} \Delta u_{\max 1} \\ \Delta u_{\max 2} \\ \vdots \\ \Delta u_{\max N_C} \\ -\Delta u_{\min 1} \\ -\Delta u_{\min 2} \\ \vdots \\ -\Delta u_{\min N_C} \\ u_{\max 1} - u(k-1) \\ u_{\max 2} - u(k-1) \\ \vdots \\ u_{\max N_C} - u(k-1) \\ -u_{\min 1} + u(k-1) \\ -u_{\min 2} + u(k-1) \\ \vdots \\ -u_{\min N_C} + u(k-1) \\ y_{\max 1} - T(k+1|k) \\ y_{\max 2} - T(k+1|k) \\ \vdots \\ y_{\max N_P} - T(k+1|k) \\ -y_{\min 1} + T(k+1|k) \\ -y_{\min 2} + T(k+1|k) \\ \vdots \\ -y_{\min N_P} + T(k+1|k) \\ 0 \end{bmatrix}}_{B} \quad (2.40)$$

Donde:

Ia es un vector de ceros, de dimensiones $N_C \times 1$.

Ib es un vector de unos, de dimensiones $N_P \times 1$.

Como se puede observar, las matrices A y B son las mismas que las del caso con restricciones duras, sólo que varían un poco al contener los vectores Ia e Ib. Por otra parte ambas matrices adquieren una fila más, debido a la existencia de la nueva restricción $0 \leq -\varepsilon$.

2.3.1.3. Observaciones.

-Se resuelve el problema de optimización de forma iterativa.

- No existe expresión analítica para la solución.

- El algoritmo encuentra directamente el vector de acciones de control a aplicar.

- Si se emplean restricciones duras y no existe solución posible que satisfaga las restricciones el algoritmo se detiene.

- A mayor número de iteraciones, más iteraciones toma el algoritmo para encontrar una solución.

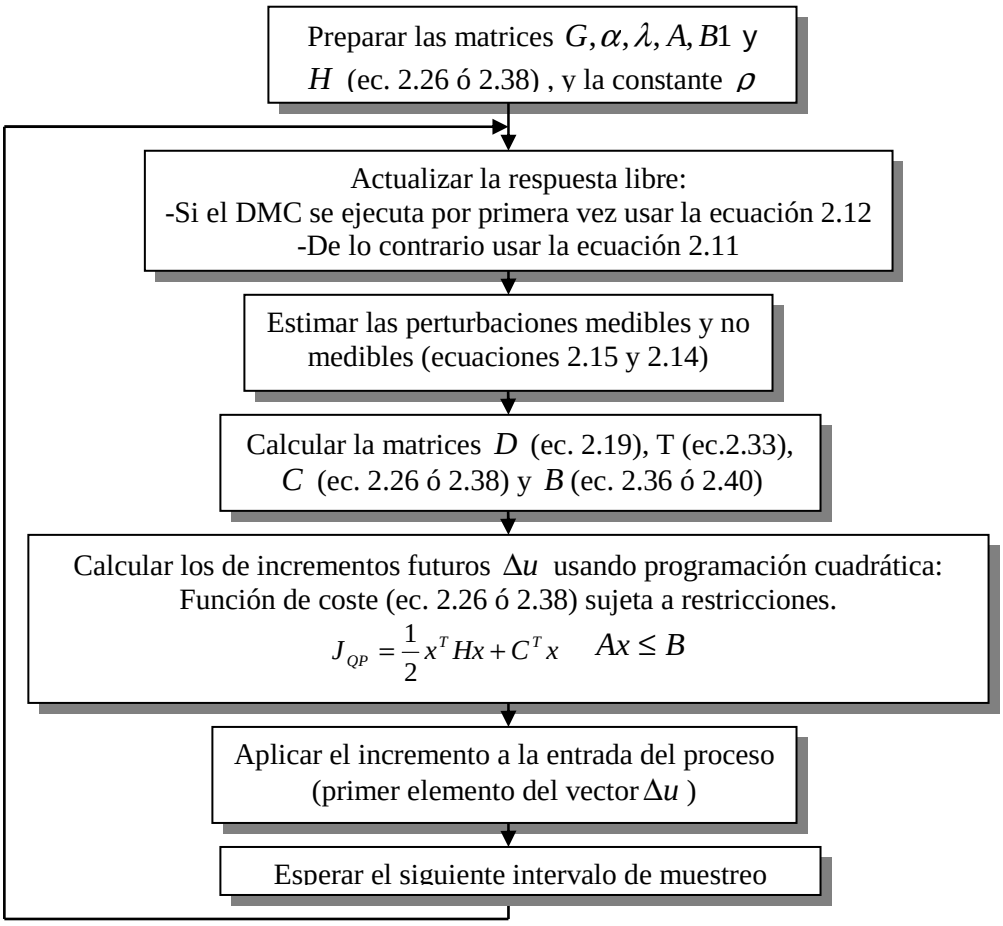


Fig. 2.11: Diagrama de flujo de la implementación de un QDMC.

2.4. DISMINUCIÓN DE LA CARGA COMPUTACIONAL.

El uso de técnicas que disminuyan el tiempo que el algoritmo DMC emplea para realizar la optimización del índice de coste sujeto a restricciones en cada periodo de control, cobra gran relevancia cuando se efectúa un control a periodos de control pequeños, se controlan procesos con dinámicas rápidas, se maneja un número elevado de variables, etc.

2.4.1. **Blocking**.

Es una técnica que se usa en los algoritmos DMC sin restricciones o QDMC y se puede usar para los siguientes propósitos:

1. Calcular los movimientos futuros sin necesidad de emplear el vector completo de errores de predicción E_P, sino a partir de unos cuantos elementos de éste. Es decir, se realizan predicciones para los instantes previamente definidos y no para todo el horizonte de predicción.

2. Los movimientos futuros no deben estar necesariamente equiespaciados en periodos de muestreo, sino que se puede elegir en qué instante a lo largo del horizonte de control se desea aplicar cada uno de éstos.

Para el **primer caso** se debe modificar el vector del error de predicción:

$$\underbrace{\begin{bmatrix} E_P(k+1|k) \\ E_P(k+2|k) \\ E_P(k+3|k) \\ \vdots \\ E_P(k+N_P|k) \end{bmatrix}}_{E_P} = \underbrace{\begin{bmatrix} R(k+1) \\ R(k+2) \\ R(k+3) \\ \vdots \\ R(k+N_P) \end{bmatrix} - Q2 \cdot \begin{bmatrix} L(k|k) \\ L(k+1|k) \\ L(k+2|k) \\ \vdots \\ L(k+N-1|k) \end{bmatrix} - \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ w(k+3|k) \\ \vdots \\ w(k+N_P|k) \end{bmatrix} - \begin{bmatrix} m(k+1|k) \\ m(k+2|k) \\ m(k+3|k) \\ \vdots \\ m(k+N_P|k) \end{bmatrix}}_{D} - \underbrace{\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ \vdots \\ F(k+N_P|k) \end{bmatrix}}_{G \cdot \Delta u}$$

$$\begin{bmatrix} E_P(k+1|k) \\ E_P(k+2|k) \\ E_P(k+3|k) \\ \vdots \\ E_P(k+N_P|k) \end{bmatrix} = \begin{bmatrix} D(k+1|k) \\ D(k+2|k) \\ D(k+3|k) \\ \vdots \\ D(k+N_P|k) \end{bmatrix} - \begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ \vdots \\ F(k+N_P|k) \end{bmatrix}$$

Suponiendo que se hacen predicciones para un horizonte de predicción $N_P = 10$ y un horizonte de control $N_C = 5$:

$$\begin{bmatrix} E_P(k+1|k) \\ E_P(k+2|k) \\ E_P(k+3|k) \\ E_P(k+4|k) \\ E_P(k+5|k) \\ E_P(k+6|k) \\ E_P(k+7|k) \\ E_P(k+8|k) \\ E_P(k+9|k) \\ E_P(k+10|k) \end{bmatrix} = \begin{bmatrix} D(k+1|k) \\ D(k+2|k) \\ D(k+3|k) \\ D(k+4|k) \\ D(k+5|k) \\ D(k+6|k) \\ D(k+7|k) \\ D(k+8|k) \\ D(k+9|k) \\ D(k+10|k) \end{bmatrix} - \begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ F(k+4|k) \\ F(k+5|k) \\ F(k+6|k) \\ F(k+7|k) \\ F(k+8|k) \\ F(k+9|k) \\ F(k+10|k) \end{bmatrix}$$

La figura 2.12, corresponde la grafica de salidas predichas para cada instante a lo largo del horizonte de predicción:

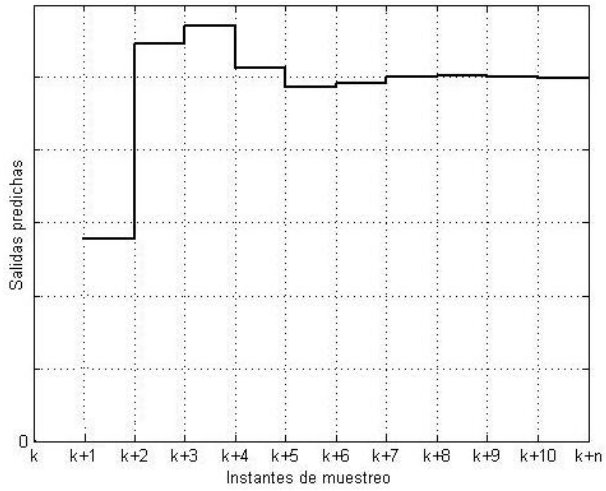


Fig. 2.12: Predicciones calculadas para cada instante de muestreo a lo largo de un horizonte de predicción.

Para la disminución de la carga computacional se seleccionan los instantes para los cuales se desean realizar predicciones o que los errores sean lo más pequeños posible de manera que la salida siga la referencia. (p.ej. instantes 1, 2, 3, 7 y 10 (ver figura 2.13)).

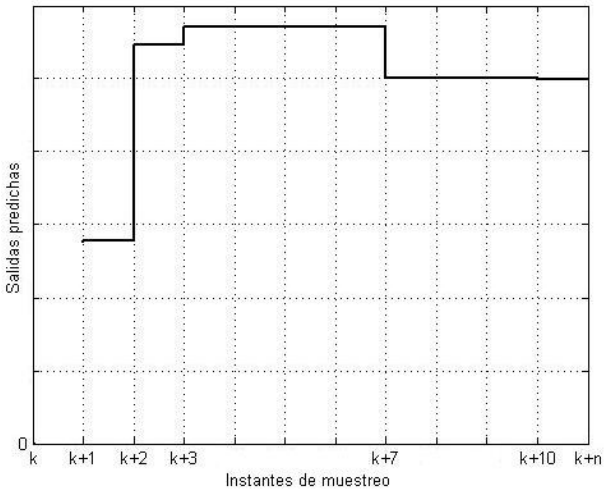


Fig. 2.13: Predicciones calculadas para los instantes deseados a lo largo de un horizonte de predicción.

Esto requiere redimensionar los vectores D y F. El vector D se modifica fácilmente incluyendo únicamente los instantes deseados (1, 2, 3, 7, 10):

$$D = \begin{bmatrix} D(k+1|k) \\ D(k+2|k) \\ D(k+3|k) \\ D(k+7|k) \\ D(k+10|k) \end{bmatrix}$$

su modificación se efectúa en cada iteración ya que el original contiene todos los instantes del horizonte de predicción N_P

En cuanto al vector F, éste se modifica, redimensionando la matriz G de manera que sólo contenga las filas correspondientes a las predicciones deseadas (1, 2, 3, 7, 10):

$$\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ F(k+4|k) \\ F(k+5|k) \\ F(k+6|k) \\ F(k+7|k) \\ F(k+8|k) \\ F(k+9|k) \\ F(k+10|k) \end{bmatrix} = \begin{bmatrix} s^u(1) & 0 & 0 & 0 & 0 \\ s^u(2) & s^u(1) & 0 & 0 & 0 \\ s^u(3) & s^u(2) & s^u(1) & 0 & 0 \\ s^u(4) & s^u(3) & s^u(2) & s^u(1) & 0 \\ s^u(5) & s^u(4) & s^u(3) & s^u(2) & s^u(1) \\ s^u(6) & s^u(5) & s^u(4) & s^u(3) & s^u(2) \\ s^u(7) & s^u(6) & s^u(5) & s^u(4) & s^u(3) \\ s^u(8) & s^u(7) & s^u(6) & s^u(5) & s^u(4) \\ s^u(9) & s^u(8) & s^u(7) & s^u(6) & s^u(5) \\ s^u(10) & s^u(9) & s^u(8) & s^u(7) & s^u(6) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \Delta u(k+2|k) \\ \Delta u(k+3|k) \\ \Delta u(k+4|k) \end{bmatrix}$$

$$\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ F(k+7|k) \\ F(k+10|k) \end{bmatrix} = \begin{bmatrix} s^u(1) & 0 & 0 & 0 & 0 \\ s^u(2) & s^u(1) & 0 & 0 & 0 \\ s^u(3) & s^u(2) & s^u(1) & 0 & 0 \\ s^u(7) & s^u(6) & s^u(5) & s^u(4) & s^u(3) \\ s^u(10) & s^u(9) & s^u(8) & s^u(7) & s^u(6) \end{bmatrix} \cdot \Delta u$$

Finalmente la matriz G será la siguiente:

$$G = \begin{bmatrix} s^u(1) & 0 & 0 & 0 & 0 \\ s^u(2) & s^u(1) & 0 & 0 & 0 \\ s^u(3) & s^u(2) & s^u(1) & 0 & 0 \\ s^u(7) & s^u(6) & s^u(5) & s^u(4) & 0 \\ s^u(10) & s^u(9) & s^u(8) & s^u(7) & s^u(6) \end{bmatrix}$$

Otra matriz que cambia es α, de manera que sólo contenga los coeficientes correspondientes a las predicciones deseadas:

$$\alpha = \begin{bmatrix} \alpha_1 & 0 & 0 & 0 \\ 0 & \alpha_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \alpha_{10} \end{bmatrix}$$

$$\alpha = \begin{bmatrix} \alpha 1 & 0 & 0 & 0 & 0 \\ 0 & \alpha 2 & 0 & 0 & 0 \\ 0 & 0 & \alpha 3 & 0 & 0 \\ 0 & 0 & 0 & \alpha 7 & 0 \\ 0 & 0 & 0 & 0 & \alpha 10 \end{bmatrix}$$

Si se están manejando restricciones, se debe modificar $A3$ (matríz de restricciones para la salida) incluyendo únicamente las filas correspondientes a las predicciones deseadas:

$$\begin{bmatrix} G \\ -G \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix} \leq \begin{bmatrix} y_{\max 1} \\ y_{\max 2} \\ \vdots \\ y_{\max 10} \\ -y_{\min 1} \\ -y_{\min 2} \\ \vdots \\ -y_{\min 10} \end{bmatrix} + \begin{bmatrix} -T(k+1|k) \\ -T(k+2|k) \\ \vdots \\ -T(k+10|k) \\ T(k+1|k) \\ T(k+2|k) \\ \vdots \\ T(k+10|k) \end{bmatrix} \qquad A3 = \begin{bmatrix} y_{\max 1} \\ y_{\max 2} \\ \vdots \\ y_{\max 10} \\ -y_{\min 1} \\ -y_{\min 2} \\ \vdots \\ -y_{\min 10} \end{bmatrix} + \begin{bmatrix} -T(k+1|k) \\ -T(k+2|k) \\ \vdots \\ -T(k+10|k) \\ T(k+1|k) \\ T(k+2|k) \\ \vdots \\ T(k+10|k) \end{bmatrix}$$

Finalmente:

$$A3 = \begin{bmatrix} y_{\max 1} \\ y_{\max 2} \\ y_{\max 3} \\ y_{\max 7} \\ y_{\max 10} \\ -y_{\max 1} \\ -y_{\max 2} \\ -y_{\max 3} \\ -y_{\max 7} \\ -y_{\max 10} \end{bmatrix} + \begin{bmatrix} -T(k+1|k) \\ -T(k+2|k) \\ -T(k+3|k) \\ -T(k+7|k) \\ -T(k+10|k) \\ T(k+1|k) \\ T(k+2|k) \\ T(k+3|k) \\ T(k+7|k) \\ T(k+10|k) \end{bmatrix}$$

Segundo caso: Los movimientos futuros normalmente se calculan para cada instante de muestreo a lo largo de un horizonte de control (ver figura 14):

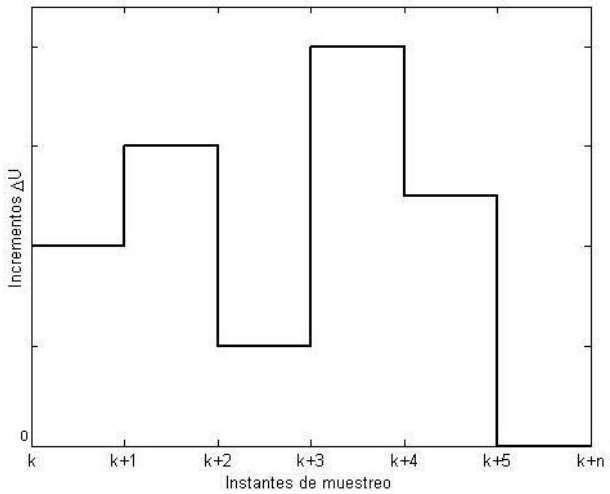


Fig. 2.14: Movimientos futuros calculados para todos los instantes a lo largo del horizonte de control.

pero en el **segundo caso,** suponiendo que sólo se desean realizar los movimientos $\Delta u(k|k)$, $\Delta u(k+2|k)$, $\Delta u(k+4|k)$ con $N_P = 5$:

$$\Delta u = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \Delta u(k+2|k) \\ \Delta u(k+3|k) \\ \Delta u(k+4|k) \end{bmatrix}$$

Lo cual quiere decir que para $\Delta u(k+1|k)$ y $\Delta u(k+3|k)$ la acción de control permanece constante, es decir que para estos instantes Δu es nula:

$$\Delta u = \begin{bmatrix} \Delta u(k|k) \\ 0 \\ \Delta u(k+2|k) \\ 0 \\ \Delta u(k+4|k) \end{bmatrix}$$

En la figura 15 se muestran los movimientos calculados par los instantes deseados mientras que para los demás instantes la variación es nula:

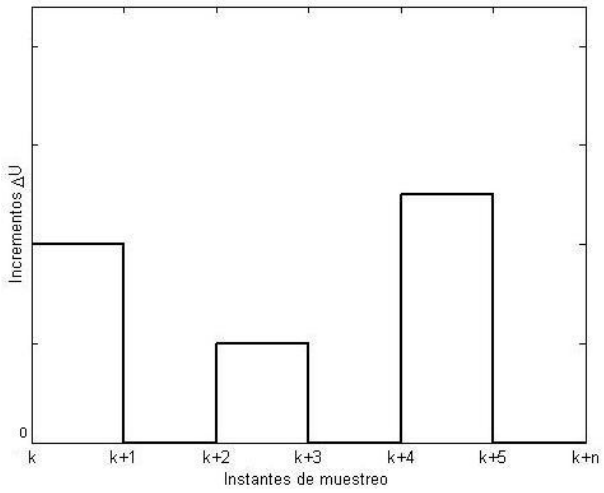


Fig. 2.15: Movimientos que se desean realizar a lo largo de un horizonte de control.

Por ende el vector calculado de incrementos Δu será:

$$\Delta u = \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+2|k) \\ \Delta u(k+4|k) \end{bmatrix}$$

Para esto es necesario reconfigurar la matriz G suprimiendo las columnas correspondientes a los incrementos que son nulos:

$$\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ F(k+4|k) \\ F(k+5|k) \end{bmatrix} = \begin{bmatrix} s^u(1) & 0 & 0 & 0 & 0 \\ s^u(2) & s^u(1) & 0 & 0 & 0 \\ s^u(3) & s^u(2) & s^u(1) & 0 & 0 \\ s^u(4) & s^u(3) & s^u(2) & s^u(1) & 0 \\ s^u(5) & s^u(4) & s^u(3) & s^u(2) & s^u(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ 0 \\ \Delta u(k+2|k) \\ 0 \\ \Delta u(k+4|k) \end{bmatrix}$$

$$\begin{bmatrix} F(k+1|k) \\ F(k+2|k) \\ F(k+3|k) \\ F(k+4|k) \\ F(k+5|k) \end{bmatrix} = \begin{bmatrix} s^u(1) & 0 & 0 \\ s^u(2) & 0 & 0 \\ s^u(3) & s^u(1) & 0 \\ s^u(4) & s^u(2) & 0 \\ s^u(5) & s^u(3) & s^u(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+2|k) \\ \Delta u(k+4|k) \end{bmatrix} \quad G = \begin{bmatrix} s^u(1) & 0 & 0 \\ s^u(2) & 0 & 0 \\ s^u(3) & s^u(1) & 0 \\ s^u(4) & s^u(2) & 0 \\ s^u(5) & s^u(3) & s^u(1) \end{bmatrix}$$

La matriz λ cambia conteniendo solamente los coeficientes correspondientes a los movimientos deseados:

$$\lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_{5_C} \end{bmatrix}$$

$$\lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_3 & 0 \\ 0 & 0 & \lambda_5 \end{bmatrix}$$

Si se están manejando restricciones, se cambian las matrices $A1, A2, B1, B2$ de manera que sólo contengan las filas correspondientes a los movimientos deseados:

$$\underbrace{\begin{bmatrix} I \\ -I \end{bmatrix}}_{A1} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ 0 \\ \Delta u(k+2) \\ 0 \\ \Delta u(k+4) \end{bmatrix}}_{\Delta u} \leq \underbrace{\begin{bmatrix} \Delta u_{\max 1} \\ \Delta u_{\max 2} \\ \vdots \\ \Delta u_{\max 5} \\ -\Delta u_{\min 1} \\ -\Delta u_{\min 2} \\ \vdots \\ -\Delta u_{\min 5} \end{bmatrix}}_{B1} \qquad \begin{bmatrix} I \\ -I \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k) \\ \Delta u(k+2) \\ \Delta u(k+4) \end{bmatrix} \leq \begin{bmatrix} \Delta u_{\max 1} \\ \Delta u_{\max 3} \\ \Delta u_{\max 5} \\ -\Delta u_{\max 1} \\ -\Delta u_{\max 3} \\ -\Delta u_{\max 5} \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} I_L \\ -I_L \end{bmatrix}}_{A2} \cdot \underbrace{\begin{bmatrix} \Delta u(k) \\ 0 \\ \Delta u(k+2) \\ 0 \\ \Delta u(k+4) \end{bmatrix}}_{\Delta u} \leq \underbrace{\begin{bmatrix} u_{\max 1} \\ u_{\max 2} \\ \vdots \\ u_{\max 5} \\ -u_{\min 1} \\ -u_{\min 2} \\ \vdots \\ -u_{\min 5} \end{bmatrix}}_{B2} - \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \cdot u(k-1) \qquad \begin{bmatrix} I_L \\ -I_L \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k) \\ \Delta u(k+2) \\ \Delta u(k+4) \end{bmatrix} \leq \begin{bmatrix} u_{\max 1} \\ u_{\max 3} \\ u_{\max 5} \\ -u_{\min 1} \\ -u_{\min 3} \\ -u_{\min 5} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \cdot u(k-1)$$

2.4.1.1. Observaciones.

-Calcular los incrementos empleando solamente una parte del vector de predicciones, contribuye fuertemente a la disminución del tiempo empleado en la optimización, especialmente si se están manejando restricciones, puesto que cada predicción es una ecuación por resolver.

- A menor número de predicciones usadas, más rápido será el algoritmo pero en consecuencia la calidad del control disminuye. Por ende el usuario debe fijar un compromiso entre calidad del control y rapidez de ejecución del algoritmo.

-Cuando se calculan incrementos para instantes deseados, siempre se debe calcular el primer elemento $\Delta u(k)$ puesto que este se debe aplicar a la entrada del proceso en cada ejecución del lazo de control.

2.4.2. Mínimos cuadrados iterativos DMC RLS.

Es un algoritmo [4] que se comporta como un DMC sin restricciones. Éste se basa en calcular los movimientos futuros Δu_m, usando el método de mínimos cuadrados ordinarios (empleando la ecuación 2.22). Posteriormente, se comprueba si con estos movimientos no se infringe alguna restricción. En caso de que no se produzca alguna violación, los movimientos Δu_m son factibles y por lo tanto se aplican al proceso.

Si se llega a producir alguna infracción, se calculan unos nuevos movimientos Δu_{m+1} a partir de los Δu_m actuales y de unas matrices adicionales E y h^T, las cuales se construyen a partir de las restricciones que se han violado. El algoritmo se ejecuta de forma recursiva haciendo uso de términos anteriores tales como restricciones E y h^T e incrementos Δu_m, hasta obtener un vector Δu_{m+1} que cumpla las restricciones generales.

2.4.2.1. Matrices de restricciones.

Son las llamadas matrices E y h^T y se construyen a partir de submatrices según el tipo de restricción que se haya violado.

$$\begin{bmatrix} E1 \\ E2 \\ E3 \end{bmatrix} = \begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix}$$

$$E = h^T \Delta u \qquad E = \begin{bmatrix} E1 \\ E2 \\ E3 \end{bmatrix} \qquad h^T = \begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix}$$

- **Restricciones para los incrementos en la acción de control.**

En esta parte se verifica si algún elemento del vector de incrementos infringe los límites máximo y mínimo:

$$\begin{bmatrix} \Delta u_{\min} \\ \Delta u_{\min} \\ \vdots \\ \Delta u_{\min} \end{bmatrix} \leq \begin{bmatrix} \Delta u(k \mid k) \\ \Delta u(k+1 \mid k) \\ \vdots \\ \Delta u(k+N_C-1 \mid k) \end{bmatrix} \leq \begin{bmatrix} \Delta u_{\max} \\ \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix}$$

Suponiendo que los incrementos $\Delta u(k \mid k)$ y $\Delta u(k+2 \mid k)$ violan los límites máximo y mínimo respectivamente:

$$\Delta u(k \mid k) \geq u_{\max}$$
$$\Delta u(k+2 \mid k) \leq u_{\min}$$

Se crean las matrices $E1$ y $h1$, las cuales buscan penalizar esos incrementos, haciéndolos estar dentro de los límites:

$$(\Delta u_{\max} - tol1)\varpi 1 = \varpi 1 \Delta u(k \mid k)$$
$$(\Delta u_{\min} + tol1)\varpi 1 = \varpi 1 \Delta u(k+2 \mid k)$$

$$\underbrace{\begin{bmatrix} (\Delta u_{\max} - tol1)\varpi 1 \\ (\Delta u_{\min} + tol1)\varpi 1 \end{bmatrix}}_{E1} = \underbrace{\begin{bmatrix} \varpi 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \varpi 1 & 0 & \cdots & 0 \end{bmatrix}}_{h1} \cdot \underbrace{\begin{bmatrix} \Delta u(k \mid k) \\ \Delta u(k+1 \mid k) \\ \Delta u(k+2 \mid k) \\ \Delta u(k+3 \mid k) \\ \Delta u(k+4 \mid k) \\ \vdots \\ \Delta u(k+N_C-1 \mid k) \end{bmatrix}}_{\Delta u} \qquad (2.41)$$

Las constantes $tol1$ y $\varpi 1$ son valores especificados off line. El primero corresponde a una pequeña tolerancia para los límites de la restricción y el segundo a un factor de peso o de importancia de la variable del proceso.

- **Restricciones de magnitud de la acción de control.**

En esta parte se verifica si algunas de las acciones de control futuras infringen sus límites:

$$u_{\min} \le u \le u_{\max}$$

Es decir:

$$\begin{bmatrix} u_{\min} \\ u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix} \le \begin{bmatrix} \Delta u(k \mid k) \\ \Delta u(k+1 \mid k) + \Delta u(k \mid k) \\ \vdots \\ \Delta u(k+N_C - 1 \mid k) + \ldots + \Delta u(k \mid k) \end{bmatrix} + \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \le \begin{bmatrix} u_{\max} \\ u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix}$$

Suponiendo que se presentan 2 casos:

-Primer caso: Las acciones $u(k+2 \mid k)$ y $u(k+4 \mid k)$ violan el límite inferior.

Para este caso, al ser $u(k+2 \mid k)$ más cercana en el tiempo que $u(k+4 \mid k)$, se busca que la suma de los tres primeros movimientos más la entrada actual no exceda el límite inferior:

$$u_{\min} + tol2 = \Delta u(k \mid k) + \Delta u(k+1 \mid k) + \Delta u(k+2 \mid k) + u(k-1)$$
$$u_{\min} + tol2 - u(k-1) = \Delta u(k \mid k) + \Delta u(k+1 \mid k) + \Delta u(k+2 \mid k)$$

-Segundo caso: La acción $u(k+1 \mid k)$ viola el límite superior.

Para este caso, se busca que la suma de los dos primeros movimientos más la entrada actual no exceda el límite superior:

$$u_{\max} - tol2 = \Delta u(k \mid k) + \Delta u(k+1 \mid k) + u(k-1)$$
$$u_{\max} - tol2 - u(k-1) = \Delta u(k \mid k) + \Delta u(k+1 \mid k)$$

Con base a las condiciones anteriores se generan las matrices $E2$ y $h2$:

$$\underbrace{\begin{bmatrix} (u_{\min} + tol2 - u(k-1))\varpi 2 \\ (u_{\max} - tol2 - u(k-1))\varpi 2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{E2} = \underbrace{\begin{bmatrix} \varpi 2 & \varpi 2 & \varpi 2 & 0 & 0 & \cdots & 0 \\ \varpi 2 & \varpi 2 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \varpi 2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \varpi 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & \varpi 2 \end{bmatrix}}_{h2} \cdot \underbrace{\begin{bmatrix} \Delta u(k \mid k) \\ \Delta u(k+1 \mid k) \\ \Delta u(k+2 \mid k) \\ \Delta u(k+3 \mid k) \\ \Delta u(k+4 \mid k) \\ \vdots \\ \Delta u(k+N_C - 1 \mid k) \end{bmatrix}}_{\Delta u} \quad (2.42)$$

Como se puede observar en la ecuación 2.42, las restricciones para ambos casos hacen parte de las dos primeras filas de cada una de las matrices $E2$ y $h2$, mientras que a los movimientos no intervienen en éstas, se busca hacerlos lo más cercanos a cero. Las constantes $tol2$ y $\varpi 2$ son la tolerancia de los limites de la variable y un factor de peso que mide la importancia que se le da a ésta respectivamente.

- **Restricciones de magnitud para las predicciones de la salida.**

Aquí se busca conocer si las predicciones de la salida a lo largo del horizonte de predicción se encuentran dentro de sus límites fijados:

$$Y_{\min} \le Y_{N_P} \le Y_{\max}$$
$$Y_{\min} \le T + G\Delta u \le Y_{\max}$$

Es decir:

$$\begin{bmatrix} Y_{\min} \\ Y_{\min} \\ \vdots \\ Y_{\min} \end{bmatrix} \le \begin{bmatrix} T(k+1|k) \\ T(k+2|k) \\ \vdots \\ T(k+N_P|k) \end{bmatrix} + \begin{bmatrix} G(1,:) \\ G(2,:) \\ \vdots \\ G(N_P,:) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+N_C-1|k) \end{bmatrix} \le \begin{bmatrix} Y_{\max} \\ Y_{\max} \\ \vdots \\ Y_{\max} \end{bmatrix}$$

Suponiendo que las predicciones para $(k+3|k)$ y $(k+5|k)$ violan el límite superior y la predicción para $(k+10|k)$ el límite inferior; se toman las filas de las matrices T y G correspondientes a cada predicción y se generan las matrices $E3$ y $h3$ de tal manera que estas predicciones se mantengan dentro de sus límites:

$$Y_{\max} - tol3 = T(k+3|k) + G(3,:)\Delta u$$
$$Y_{\max} - tol3 = T(k+5|k) + G(5,:)\Delta u$$
$$Y_{\min} + tol3 = T(k+10|k) + G(10,:)\Delta u$$

$$Y_{\max} - tol3 - T(k+3|k) = G(3,:)\Delta u$$
$$Y_{\max} - tol3 - T(k+5|k) = G(5,:)\Delta u$$
$$Y_{\min} + tol3 - T(k+10|k) = G(10,:)\Delta u$$

$$\underbrace{\begin{bmatrix} (Y_{\max} - tol3 - T(k+3|k))\varpi 3 \\ (Y_{\max} - tol3 - T(k+5|k))\varpi 3 \\ (Y_{\min} + tol3 - T(k+10|k))\varpi 3 \end{bmatrix}}_{E3} = \underbrace{\begin{bmatrix} G(3,:)\varpi 3 \\ G(5,:)\varpi 3 \\ G(10,:)\varpi 3 \end{bmatrix}}_{h3} \cdot \underbrace{\begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \Delta u(k+2|k) \\ \Delta u(k+3|k) \\ \Delta u(k+4|k) \\ \vdots \\ \Delta u(k+N_C-1|k) \end{bmatrix}}_{\Delta u} \quad (2.43)$$

Las constantes $tol3$ y $\varpi 3$ son valores especificados off line. El primero corresponde a una pequeña tolerancia de los límites de la variable y el segundo a un factor de peso o de importancia de la variable del proceso.

Evaluadas las restricciones y creadas las diferentes submatrices, se procede a crear las matrices generales E y h^T que contienen las restricciones que se han violado:

$$\begin{bmatrix} E1 \\ E2 \\ E3 \end{bmatrix} = \begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_C-1) \end{bmatrix}$$

$$E = h^T \Delta u \qquad E = \begin{bmatrix} E1 \\ E2 \\ E3 \end{bmatrix} \qquad h^T = \begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix}$$

Donde:

E es un vector columna cuya longitud varía a cada instante.

h^T es una matriz de N_C columnas y su número de filas varía en cada iteración.

2.4.2.2. Funcionamiento.

A continuación se describe de manera detallada el funcionamiento del algoritmo:

1. Se realizan las predicciones y se calculan los movimientos futuros:

$$\Delta u_m = M \cdot D$$

2. Se verifica si con estos movimientos se infringe alguna de las restricciones:

$$\Delta u_{\min} \le \Delta u_m \le \Delta u_{\max}$$
$$u_{\min} \le u \le u_{\max}$$
$$Y_{\min} \le Y_{N_P} \le Y_{\max}$$

3. En caso de que no se viole ninguna restricción, ir al paso 10.
4. Se crea la matriz inicial:

$$P_m = (G^T G)^{-1}$$

5. Se generan las matrices E y h^T a partir de las restricciones que se han violado.
6. Con base en las matrices E y h^T, se calculan los nuevos movimientos Δu_{m+1} que tienen en cuenta las restricciones:

$$\Delta u_{m+1} = \Delta u_m + P_m h(I + h^T P_m h)^{-1}(E - h^T \Delta u_m)$$

7. Se aplica el paso 2 para los nuevos movimientos Δu_{m+1}. En caso de que no se viole ninguna restricción, ir al paso 10.
8. Se actualiza recursivamente una matriz P_m, la cual contendrá las restricciones actuales (E y h^T): $P_{m+1} = P_m - P_m h(I + h^T P_m h)^{-1} h^T P_m$
9. Se regresa al paso 5 para crear las nuevas matrices E y h^T con respecto a los nuevos incrementos Δu_{m+1}.
10. Aplicar el nuevo incremento a la entrada del proceso (primer elemento del vector Δu_m).
11. Esperar el siguiente instante de muestreo.
12. vuelve al paso 1.

2.4.2.3. Revisión de horizontes.

Es una técnica exclusiva del algoritmo DMC empleando mínimos cuadrados Iterativos, y se emplea para disminuir aún más la carga computacional. El algoritmo se ejecuta normalmente realizando predicciones para los horizontes de predicción N_P y de control N_C. Cuando se ejecuta la parte del algoritmo correspondiente a verificar si se infringen los límites, solamente se revisarán los primeros N_{Uexam} elementos de los vectores de incrementos y

de entradas futuras y las primeras N_{Pexam} salidas predichas. El uso de está técnica requiere de la modificación de la matriz inicial $P_m = (G^T G)^{-1}$ y consiste en tomar únicamente las primeras N_{Uexam} filas y N_{Uexam} columnas, resultando en una matriz cuadrada de (N_{Uexam} x N_{Uexam}). De igual manera la matriz G_{aux} usada para verificar las predicciones de la salida se genera tomando las primeras N_{Pexam} filas y N_{Uexam} columnas de la matriz G, resultando en una matriz cuadrada de (N_{Pexam} x N_{Uexam}).

2.4.2.4. Observaciones.

-Cuando se generan infracciones en los límites, las diferentes submatrices de E y h^T, varían en cada instante ya que dependen de los elementos en especial que violan dichos límites.

-El algoritmo encuentra rápidamente una solución pero no la óptima.

-Cuando se trabaja muy cerca de los límites, o se presente una perturbación que aleje la señal de su rango de operación, el algoritmo se puede quedar en un bucle infinito buscando una solución factible. Para evitar esto, es recomendable fijar un número máximo de iteraciones.

Al emplear la revisión de horizontes:

- la rapidez del algoritmo para encontrar soluciones ante restricciones está estrechamente ligada a los valores de las tolerancias $tol1, tol2, tol3$.

- A menor número de predicciones y movimientos futuros considerados, más rápido será el algoritmo pero en consecuencia la calidad del control disminuye. Por ende el usuario debe fijar un compromiso entre calidad del control y rapidez.

2.5. CASO MULTIVARIABLE.

La extensión al caso multivariable donde se tienen m variables manipuladas y n variables controladas es un poco más complicada y más costosa en

términos computacionales. Los pasos para obtener el modelo de respuesta al escalón para estos tipos de sistemas son los siguientes:

1. Estando el sistema en estado estable, en el instante k se aplica un incremento a una de las entradas del proceso y, se miden todas las n salidas a partir del instante $k+1$ hasta que todas se hayan estabilizado. Este paso se realiza para cada una de las m entradas.
2. Al igual que en el caso SISO, a cada vector de salidas se le resta su respectivo valor al inicio y posteriormente se divide por la magnitud del incremento aplicado a la salida.
3. Los vectores $S_{i,j}^{u}$ de coeficientes son redimensionados de tal manera que sólo contengan los coeficientes hasta el punto en que cada salida se ha estabilizado.
4. Al final se tendrán m vectores de coeficientes por cada una de las n salidas del proceso.

La respuesta libre para la entrada uno se calcula de la misma forma que en el caso SISO, sólo que el incremento realizado en cada entrada en el instante anterior, se multiplica por su respectivo vector de coeficientes:

$$\begin{bmatrix} L1(k\mid k) \\ L1(k+1\mid k) \\ L1(k+2\mid k) \\ \vdots \\ L1(k+N1-1\mid k) \end{bmatrix} = Q1_1 \cdot \begin{bmatrix} L1(k\mid k-1) \\ L1(k+1\mid k-1) \\ L1(k+2\mid k-1) \\ \vdots \\ L1(k+N1-1\mid k-1) \end{bmatrix} + \begin{bmatrix} S_{11}^{u}(1) \\ S_{11}^{u}(2) \\ S_{11}^{u}(3) \\ \vdots \\ S_{11}^{u}(N1) \end{bmatrix} \cdot \Delta u1(k\mid k-1) + \cdots$$

$$\cdots + \begin{bmatrix} S_{12}^{u}(1) \\ S_{12}^{u}(2) \\ S_{12}^{u}(3) \\ \vdots \\ S_{12}^{u}(N1) \end{bmatrix} \cdot \Delta u2(k\mid k-1) + \cdots + \begin{bmatrix} S_{1n}^{u}(1) \\ S_{1n}^{u}(2) \\ S_{1n}^{u}(3) \\ \vdots \\ S_{1n}^{u}(N1) \end{bmatrix} \cdot \Delta un(k\mid k-1)$$

el vector de respuesta libre total para todas las n salidas se calcula de la siguiente manera:

$$\begin{bmatrix} L1 \\ L2 \\ \vdots \\ Ln \end{bmatrix} = \begin{bmatrix} Q1_1 & 0 & \cdots & 0 \\ 0 & Q1_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q1_m \end{bmatrix} \cdot \begin{bmatrix} L1_{ant} \\ L2_{ant} \\ \vdots \\ Ln_{ant} \end{bmatrix} + \begin{bmatrix} S_{11}^u & S_{12}^u & \cdots & S_{1m}^u \\ S_{21}^u & S_{22}^u & \cdots & S_{2m}^u \\ \vdots & \vdots & \ddots & \vdots \\ S_{n1}^u & S_{n2}^u & \cdots & S_{nm}^u \end{bmatrix} \cdot \begin{bmatrix} \Delta u1(k-1) \\ \Delta u2(k-1) \\ \vdots \\ \Delta um(k-1) \end{bmatrix}$$

La matriz G global se genera calculando las $n \times m$ submatrices $G_{i,j}$ con los $n \times m$ vectores de coeficientes:

$$G = \begin{bmatrix} G1 \\ G2 \\ \vdots \\ Gn \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nm} \end{bmatrix}$$

El vector de incrementos futuros para todas las entradas donde Ni_C es el horizonte de control de la $i-ésima$ entrada:

$$\Delta u = \begin{bmatrix} \Delta u1(k \mid k) \\ \Delta u1(k+1 \mid k) \\ \vdots \\ \Delta u1(k+N1_C-1 \mid k) \\ \Delta u2(k \mid k) \\ \Delta u2(k+1 \mid k) \\ \vdots \\ \Delta u2(k+N2_C-1 \mid k) \\ \vdots \\ \Delta um(k \mid k) \\ \Delta um(k+1 \mid k) \\ \vdots \\ \Delta um(k+Nm_C-1 \mid k) \end{bmatrix} \quad ; \quad \Delta u = \begin{bmatrix} \Delta u1 \\ \Delta u2 \\ \vdots \\ \Delta um \end{bmatrix}$$

Por consiguiente la respuesta forzada total se obtiene de la siguiente forma:

$$\begin{bmatrix} F1 \\ F2 \\ \vdots \\ Fn \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nm} \end{bmatrix} \cdot \begin{bmatrix} \Delta u1 \\ \Delta u2 \\ \vdots \\ \Delta um \end{bmatrix}$$

Donde el término Fi corresponde al vector de la respuesta forzada para la $i-ésima$ variable controlada.

El vector de predicciones de las salidas se calcula así:

$$\begin{bmatrix} Y_P1 \\ Y_P2 \\ \vdots \\ Y_Pn \end{bmatrix} = \begin{bmatrix} Q2_1 & 0 & \cdots & 0 \\ 0 & Q2_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q2_m \end{bmatrix} \cdot \begin{bmatrix} L1 \\ L2 \\ \vdots \\ Ln \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nm} \end{bmatrix} \cdot \begin{bmatrix} \Delta u1 \\ \Delta u2 \\ \vdots \\ \Delta um \end{bmatrix} + \begin{bmatrix} w1 \\ w2 \\ \vdots \\ wn \end{bmatrix} + \begin{bmatrix} m1 \\ m2 \\ \vdots \\ mn \end{bmatrix}$$

cada término Y_Pi, Li, wi y mi corresponde a vectores de: predicciones, respuesta libre, predicciones no medibles y predicciones medibles de la $i-ésima$ variable controlada respectivamente. Con el vector de referencias donde cada término Ri es la referencia para la $i-ésima$ variable controlada se genera el vector de errores de predicción general:

$$\begin{bmatrix} E_P1 \\ E_P2 \\ \vdots \\ E_Pn \end{bmatrix} = \begin{bmatrix} R1 \\ R2 \\ \vdots \\ Rn \end{bmatrix} - \begin{bmatrix} Q2_1 & 0 & \cdots & 0 \\ 0 & Q2_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q2_m \end{bmatrix} \cdot \begin{bmatrix} L1 \\ L2 \\ \vdots \\ Ln \end{bmatrix} - \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nm} \end{bmatrix} \cdot \begin{bmatrix} \Delta u1 \\ \Delta u2 \\ \vdots \\ \Delta um \end{bmatrix} - \begin{bmatrix} w1 \\ w2 \\ \vdots \\ wn \end{bmatrix} - \begin{bmatrix} m1 \\ m2 \\ \vdots \\ mn \end{bmatrix}$$

$$E_P = D - G\Delta u$$

Al igual que en el caso SISO, estos vectores se reemplazan en la ecuación de coste:

$$\min_{\Delta u} J = \min_{\Delta u} [(D - G\Delta u)^T \alpha (D - G\Delta u) + \Delta u^T \lambda \Delta u]$$

Donde las matrices α y λ se forman con las submatrices αi y λi correspondientes a cada variable manipulada del proceso:

$$\alpha = \begin{bmatrix} \alpha 1 & 0 & 0 & 0 \\ 0 & \alpha 2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \alpha n \end{bmatrix} \quad \lambda = \begin{bmatrix} \lambda 1 & 0 & 0 & 0 \\ 0 & \lambda 2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda m \end{bmatrix}$$

Finalmente se obtiene el vector de incrementos futuros para todas las variables manipuladas:

$$\Delta u = M \cdot D$$

$$M = (G^T \alpha G + \lambda)^{-1} G^T \alpha$$

$$\Delta u = \begin{bmatrix} \Delta u1(k \mid k) \\ \Delta u1(k+1 \mid k) \\ \vdots \\ \Delta u1(k + N1_C - 1 \mid k) \\ \Delta u2(k \mid k) \\ \Delta u2(k+1 \mid k) \\ \vdots \\ \Delta u2(k + N2_C - 1 \mid k) \\ \vdots \\ \Delta um(k \mid k) \\ \Delta um(k+1 \mid k) \\ \vdots \\ \Delta um(k + Nm_C - 1 \mid k) \end{bmatrix}$$

CAPÍTULO 3

SIMULACIÓN E IMPLEMENTACIÓN DE DOS ALGORITMOS DE CONTROL PREDICTIVO SOBRE SISTEMAS MULTIVARIABLES

pág.

3.1. SIMULACIÓN SOBRE UN SISTEMA MULTIVARIABLE.

En este apartado se exponen los resultados de varias simulaciones realizadas en Matlab de un controlador DMC multivariable, empleando los métodos de Programación Cuadrática (QDMC) y Mínimos Cuadrados Iterativos (DMC RLS) para el manejo de restricciones.

3.1.1. Modelo del proceso.

El proceso a controlar consiste en una columna de destilación de petróleo (ver figura 3.1), representado mediante un sistema LTI de tres entradas y tres salidas, en el cual las últimas presentan retardos. Este modelo aparece como Benchmark en diferentes publicaciones y es muy usado para evaluar diferentes estrategias de control.

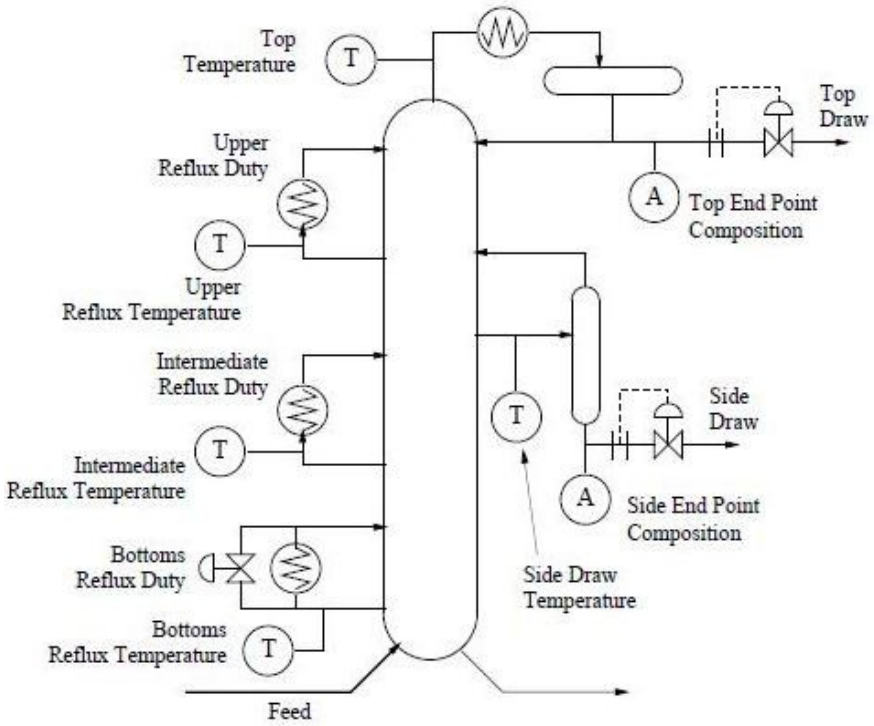


Fig. 3.1: Columna de destilación del petróleo (imagen tomada de [9]).

Las **variables manipuladas** del proceso son: la extracción de destilado superior (Top draw) $u1$, la extracción de destilado lateral (Side draw) $u2$ y el reflujo de fondo (Bottoms reflux duty) $u3$.

Las **variables controladas** son: la composición del producto de la parte superior (Top end point composition) $y1$, la composición del producto de la parte lateral (Side end point composition) $y2$ y la temperatura de la parte inferior (Bottoms reflux temperature) $y3$.

En el modelo matemático (extraído de [1] y [9]), cada salida está representada mediante la suma de las salidas de tres subsistemas donde cada uno de éstos depende de una entrada del proceso:

$$\begin{aligned} y1(s) &= \frac{4{,}05e^{-27s}}{50s+1}u1(s) + \frac{1{,}77e^{-28s}}{60s+1}u2(s) + \frac{2{,}88e^{-23s}}{50s+1}u3(s) \\ y2(s) &= \frac{5{,}39e^{-18s}}{50s+1}u1(s) + \frac{5{,}72e^{-14s}}{60s+1}u2(s) + \frac{6{,}90e^{-15s}}{40s+1}u3(s) \\ y3(s) &= \frac{4{,}38e^{-20s}}{33s+1}u1(s) + \frac{4{,}42e^{-22s}}{44s+1}u2(s) + \frac{7{,}20}{19s+1}u3(s) \end{aligned} \qquad (3.1)$$

En forma matricial:

$$\begin{bmatrix} y1(s) \\ y2(s) \\ y3(s) \end{bmatrix} = \begin{bmatrix} \frac{4{,}05e^{-27s}}{50s+1} & \frac{1{,}77e^{-28s}}{60s+1} & \frac{2{,}88e^{-23s}}{50s+1} \\ \frac{5{,}39e^{-18s}}{50s+1} & \frac{5{,}72e^{-14s}}{60s+1} & \frac{6{,}90e^{-15s}}{40s+1} \\ \frac{4{,}38e^{-20s}}{33s+1} & \frac{4{,}42e^{-22s}}{44s+1} & \frac{7{,}20}{19s+1} \end{bmatrix} \cdot \begin{bmatrix} u1(s) \\ u2(s) \\ u3(s) \end{bmatrix}$$

Como se observa en las ecuaciones 3.1, las salidas $y1$ e $y2$ presentan retardos de 23 y 14 minutos respectivamente.

Por otra parte, los tiempos de establecimiento del total de las salidas no son menores a los 200 minutos.

Los límites de las diferentes variables del proceso se muestran a continuación:

- Límites de velocidades de cambio máximas y mínimas de las acciones de control:

$$\Delta u1min = -0.05$$
$$\Delta u1max = 0.05$$
$$\Delta u2min = -0.05$$
$$\Delta u2max = 0.05$$
$$\Delta u3min = -0.05$$
$$\Delta u3max = 0.05$$

- Límites de magnitudes máximas y mínimas de las acciones de control:

$$u1min = -0.2$$
$$u1max = 0.2$$
$$u2min = -0.2$$
$$u2max = 0.2$$
$$u3min = -0.2$$
$$u3max = 0.2$$

- Límites de magnitudes máximas y mínimas de las variables controladas:

$$y1min = -0.4$$
$$y1max = 0.4$$
$$y2min = -0.3$$
$$y2max = 0.3$$
$$y3min = -0.2$$
$$y3max = 0.2$$

3.1.2. Resultados de simulaciones.

El modelo de respuesta al escalón se formó a partir de la medición de las salidas a un periodo de muestreo de 4 minutos, por consiguiente el control se realizó a este mismo periodo.

La resolución del problema de programación cuadrática 3.2 en Matlab se efectúa empleando la función de programación cuadrática *QUADPROG*, que resuelve el siguiente problema:

$$J_{QP} = \frac{1}{2}\Delta u^T H \Delta u + C^T \Delta u \qquad (3.2)$$

Sujeto a:

$$A\Delta u \leq B$$

$$\Delta u = quadprog(H, C, A, B)$$

Y donde H, C, A y B son las matrices referidas en la sección 2.3

- **Simulación 1**: Para este experimento se tomaron valores para las referencias de las salidas en posiciones seguras dentro de los límites de las variables. En total se realizaron cien iteraciones o ejecuciones de lazo de control y el tiempo total tomado por cada algoritmo en realizar las operaciones se muestra en la siguiente tabla:

Algoritmo	Número de iteraciones	Duración (segundos)
Programación Cuadrática	100	2,6263
Mínimos cuadrados Iterativos	100	0,28382

En la tabla anterior se observa que el tiempo tomado por el algoritmo de mínimos cuadrados iterativos en realizar el total de las iteraciones es mucho menor que el tomado por el de programación cuadrática, esto se debe a que el algoritmo sólo se ejecuta cuando se viola alguna restricción.

Las figuras 3.2 y 3.3 corresponden a las gráficas de los comportamientos de las diferentes variables del sistema usando los dos algoritmos de control. En las dos figuras se observa que las salidas se posicionan en sus correspondientes valores de referencias, con tiempos de establecimiento similares y sin violar los límites.

Por otra parte, en cuanto a las entradas del proceso y sus velocidades de variación, se observa que en ambos casos éstas al comienzo son altas pero siempre se mantienen dentro de sus correspondientes límites.

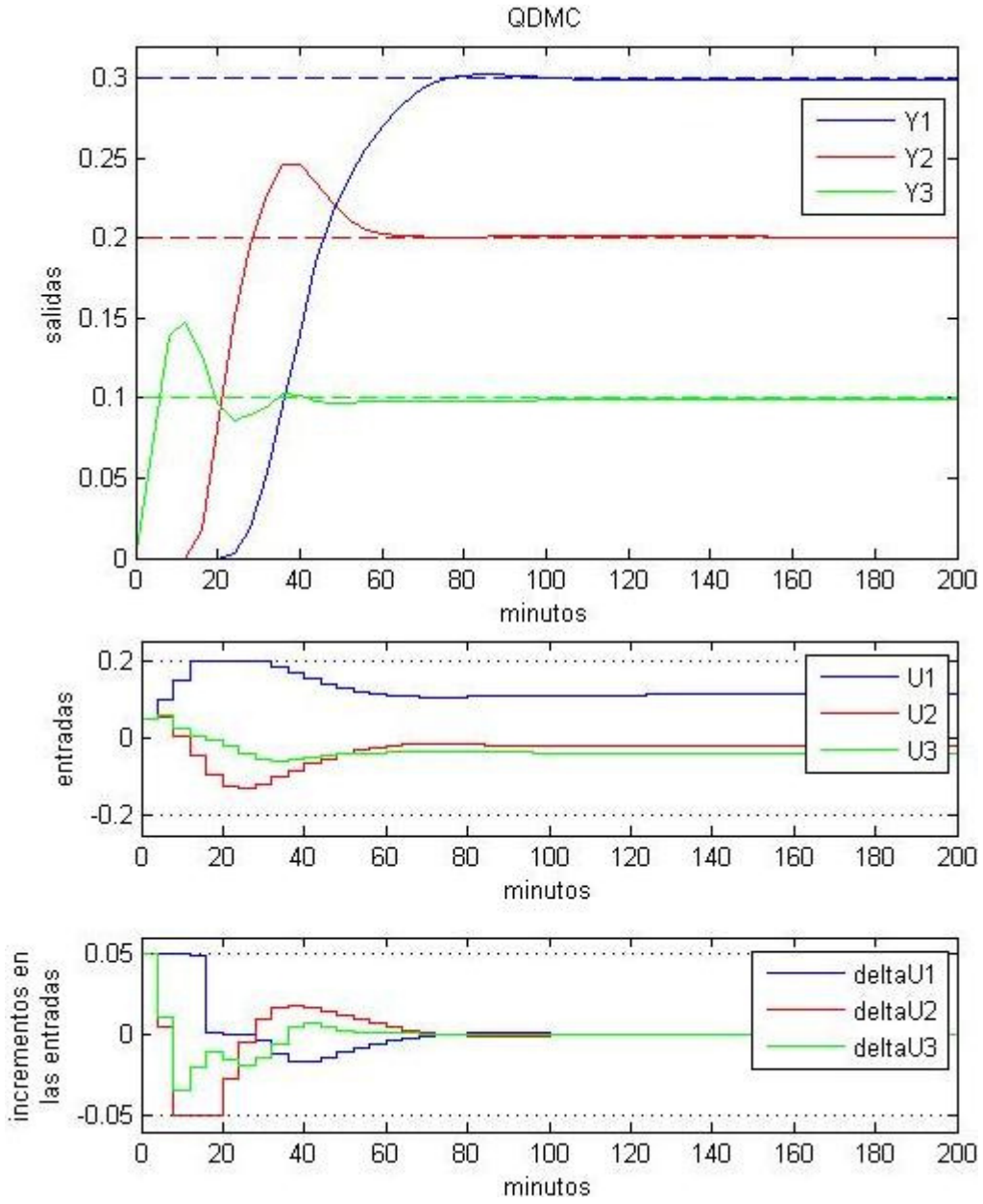


Fig. 3.2: Simulación 1, QDMC.

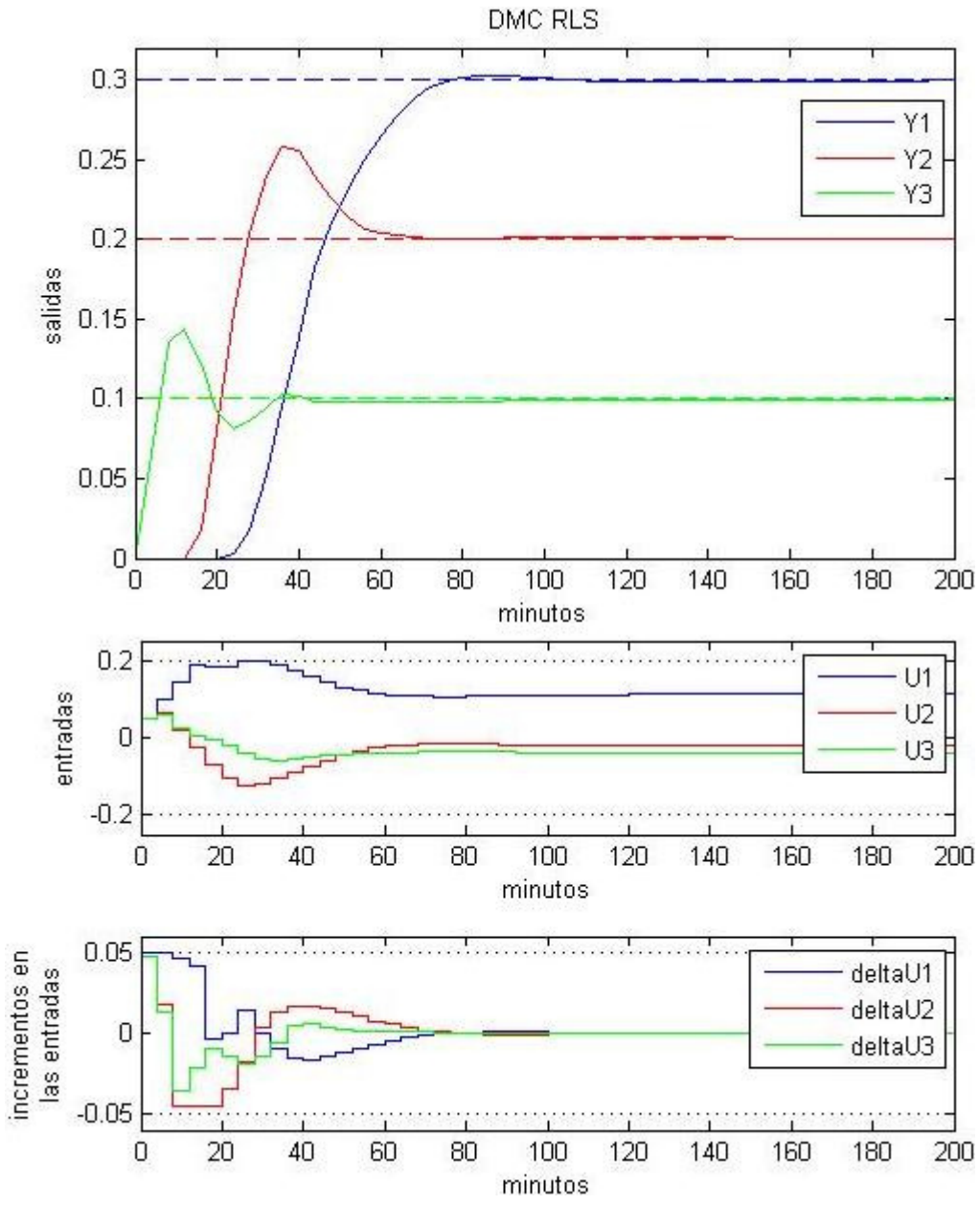


Fig. 3.3: Simulación 1, DMC RLS.

- **Simulación 2**: Para este experimento se tomaron valores de referencias iguales a los límites superiores de cada variable controlada, con el fin de comprobar si existe algún sobrepaso y ver el comportamiento de ambos algoritmos ante estos valores de referencias. En la siguiente tabla se pueden apreciar los tempos empleados por los dos algoritmos para realizar los cálculos para un total de cien ejecuciones de lazo de control:

Algoritmo	Número de iteraciones	Duración (segundos)
Programación Cuadrática	100	7,7929
Mínimos cuadrados Iterativos	100	3,0652

Nuevamente el algoritmo de mínimos cuadrados iterativos es el más rápido de los dos. Comparando estos tiempos con los del experimento anterior, se aprecia que los del presente experimento son mucho más altos, aún teniendo el mismo número de iteraciones para ambos casos.

Estos elevados costes computacionales se deben a que el sistema es llevado a sus límites, por lo que las predicciones realizadas son más propensas a violar los límites y por lo tanto, el proceso de optimización en cada algoritmo requiere de un mayor número de iteraciones para obtener un vector óptimo de incrementos futuros que respete todas las restricciones.

Las figuras 3.4 y 3.5 corresponden a las gráficas de los comportamientos de las diferentes variables del sistema usando los dos algoritmos de control. En las dos figuras se observa que las salidas se posicionan en sus correspondientes valores de referencias, con tiempos de establecimiento similares y en ningún caso se observa algún sobre paso (no se viola ningún límite).

Las salidas para el caso del algoritmo de mínimos cuadrados, no coinciden exactamente con sus respectivas referencias dado que este algoritmo hace uso de una tolerancia previamente definida, donde se busca que la salida máxima esté por debajo del límite esa tolerancia.

En lo concerniente a las entradas del proceso y a sus velocidades de variación, se advierte que en ambos casos éstas son altas al comienzo pero siempre se mantienen dentro de sus correspondientes límites.

Existen unas ligeras variaciones en las velocidades de cambio en las acciones de control para el caso de mínimos cuadrados, debido a que las salidas del proceso nunca alcanzan sus respectivas referencias (límites máximos) puesto que sus máximos valores permitidos serán sus límites superiores menos una tolerancia previamente definida.

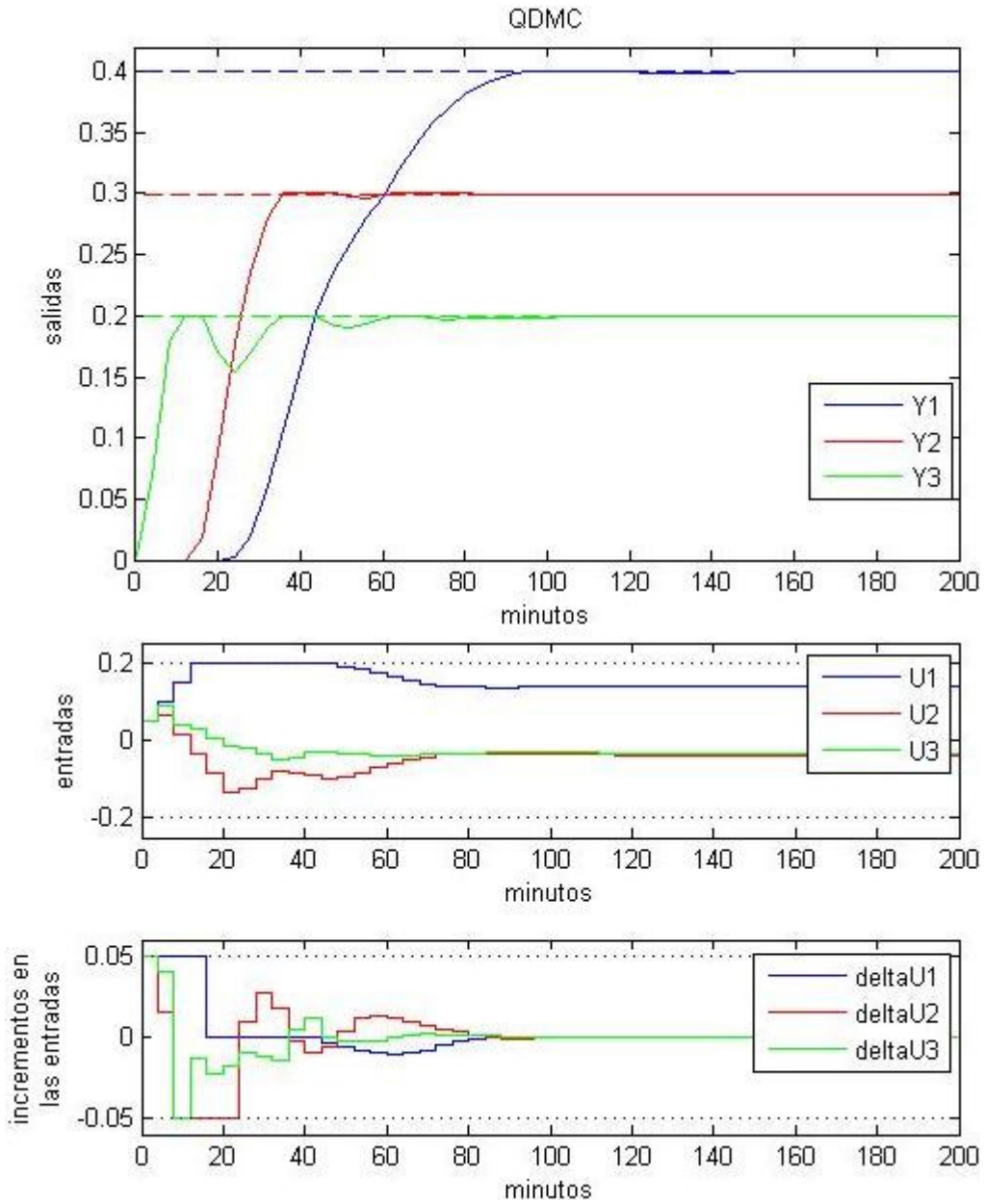


Fig. 3.4: Simulación 2, QDMC.

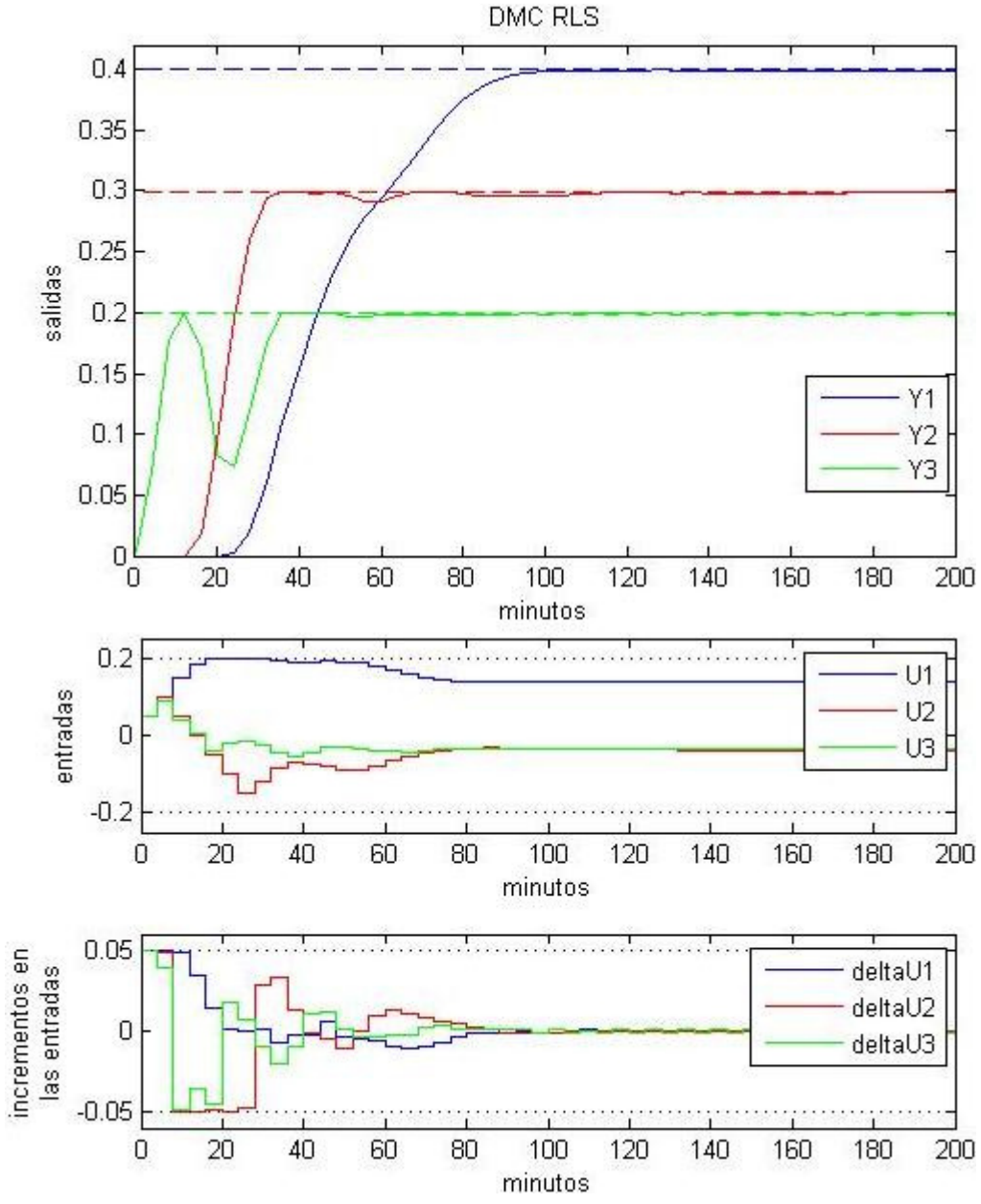


Fig. 3.5: Simulación 2, DMC RLS.

- **Simulación 3**: Para este caso se tomaron valores de referencias seguros dentro de los límites de las salidas, donde se busca analizar el comportamiento de los dos algoritmos DMC ante la presencia de una perturbación no medible en la primera salida del sistema una vez éste se ha estabilizado.

Al igual que en los experimentos anteriores, se realizaron cien ejecuciones de lazo de control y los tiempos empleados por los algoritmos para la realización de los cálculos se exponen a continuación:

Algoritmo	Número de iteraciones	Duración (segundos)
Programación Cuadrática	100	2,5402
Mínimos cuadrados Iterativos	100	0,28358

El algoritmo de mínimos cuadrados iterativos es más veloz que el de programación cuadrática (siete veces más rápido aproximadamente).

Las figuras 3.6 y 3.7 pertenecen a las gráficas de los comportamientos de las variables del sistema usando los algoritmos de programación cuadrática y mínimos cuadrados iterativos. Al comienzo de cada ejecución, las variables controladas se sitúan en sus correspondientes referencias con tiempos de establecimiento parecidos en ambos algoritmos.

Al momento de presentarse la perturbación en la variable controlada $y1$, los controladores responden inmediatamente corrigiendo el error y posicionando nuevamente la variable en su valor de referencia. También se observa que al presentarse la perturbación y durante la posterior corrección del error, las demás variables controladas $y2$ e $y3$ no varían significativamente.

Por otra parte, las variables manipuladas y sus velocidades de cambio al inicio alcanzan sus límites máximos pero en ningún momento los excede, manteniéndose siempre dentro de sus rangos de acción permitidos.

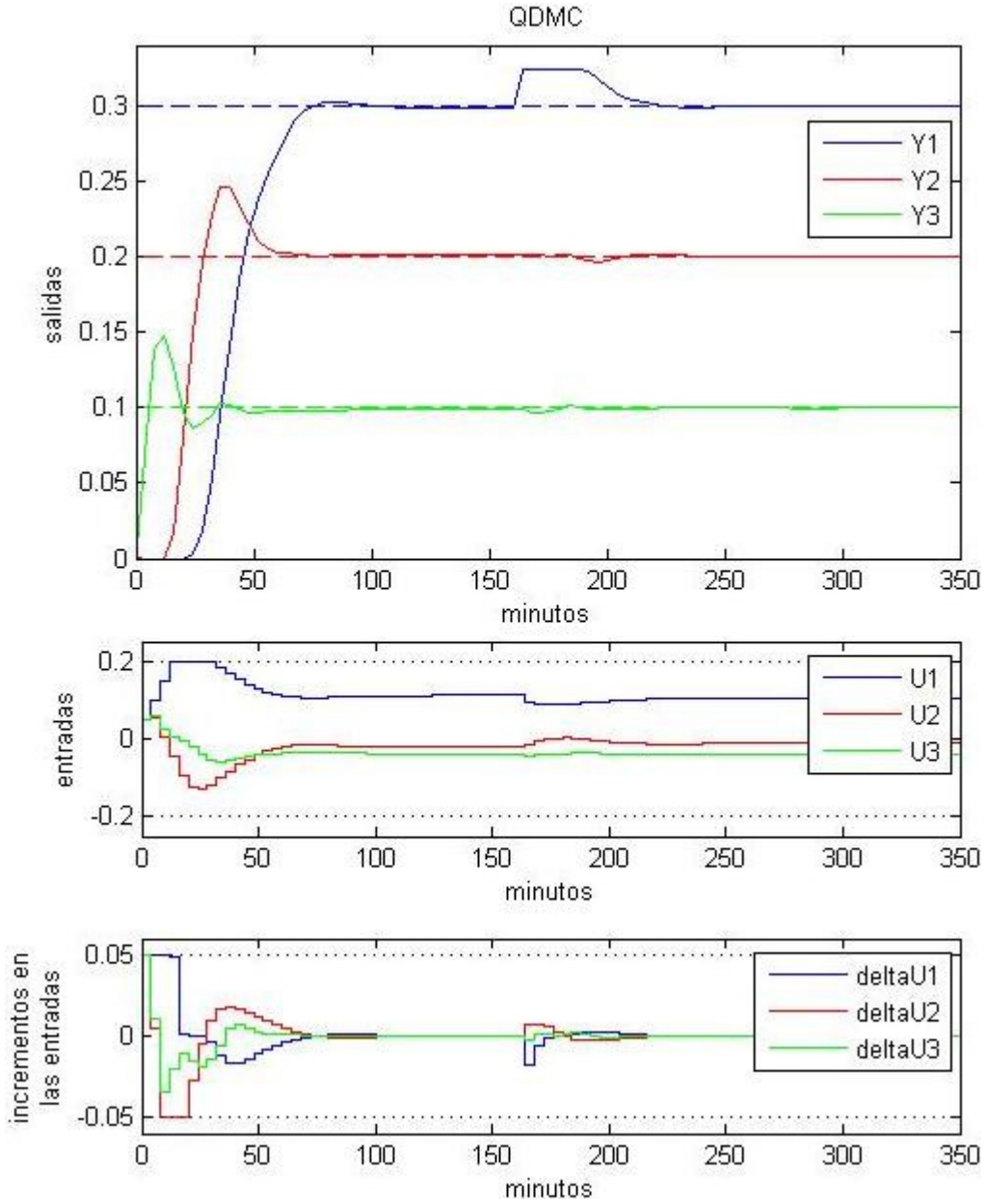


Fig. 3.6: Simulación 3, QDMC.

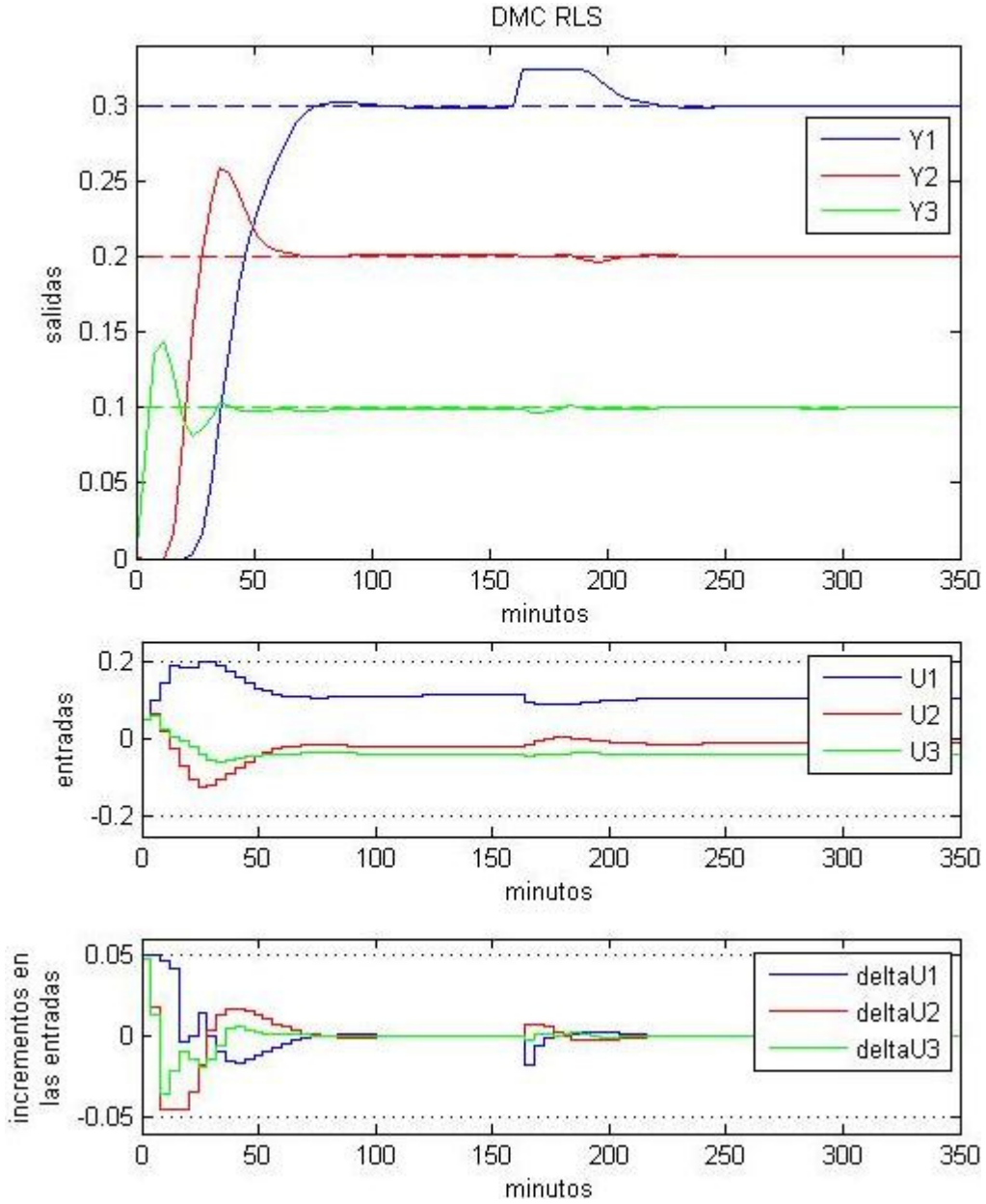


Fig. 3.7: Simulación 3, DMC RLS.

3.1.3. Observaciones.

- Los dos algoritmos DMC ejercen un buen control y respetan las restricciones mientras exista una solución posible.

- Las reacciones de ambos controladores ante la presencia de perturbaciones son muy buenas ya que corrigen rápidamente los errores,

intentando que la repercusión sobre las demás variables controladas sean lo menos perceptibles.

- El manejo de restricciones usando el algoritmo de Mínimos Cuadrados Iterativos, es mucho más rápido en términos computacionales que el de Programación cuadrática.
- La rapidez del algoritmo de mínimos cuadrados radica en que éste sólo se ejecuta cuando se infringe un límite y la optimización finaliza al encontrar el primer vector de incrementos que respete las restricciones.
- La lentitud del algoritmo de programación cuadrática se debe a que éste siempre busca el vector de incrementos óptimo que respeta las restricciones, utilizando un método iterativo.
- El uso del método de programación cuadrática es muy conveniente cuando se tienen en cuenta factores económicos o cuando los periodos de lazo permiten su uso; mientras que el método de mínimos cuadrados es de gran utilidad cuando se controlan procesos con dinámicas rápidas.

3.2. IMPLEMENTACIÓN SOBRE UN SISTEMA DE MOTORES ACOPLADOS.

En este apartado se exponen resultados del comportamiento de un sistema de motores acoplados real, controlado con dos controladores predictivos DMC. El primero usa el método de programación cuadrática y el segundo el método de mínimos cuadrados iterativos. Adicionalmente se utiliza un tercer controlador el cual consiste en un algoritmo de control clásico.

3.2.1. Descripción del proceso.

El proceso consiste en un sistema de dos motores que están acoplados por medio de una banda elástica que pasa alrededor de sus ejes y el de una polea, la cual está unida a un brazo móvil que se encuentra suspendido de un resorte (ver figura 3.8).

Fig. 3.8: Sistema de motores acoplados.

El sistema consta de dos entradas y dos salidas (ver figura 3.9). Las entradas $u1$ y $u2$ corresponden a los voltajes aplicados a los motores 1 y 2 respectivamente y determinan la velocidad y sentido de giro de cada uno de éstos. Las salidas ω y θ corresponden a la velocidad de giro de la polea y ángulo de inclinación del brazo respectivamente.

Fig. 3.9: Diagrama de bloques del sistema.

En la figura 3.10 se observa el esquema [10] del funcionamiento del proceso. La banda que acopla los motores es una banda elástica y continua; la polea gira libremente sobre su eje y está unida a un brazo móvil el cual se encuentra suspendido de un resorte y tiene un extremo unido a un eje con posición fija. Con los voltajes $u1$ y $u2$ se definen la magnitud de la velocidad y sentido de giro de los motores 1 y 2, y por lo tanto con la

variación de estas entradas se puede regular la velocidad de la polea y el ángulo de inclinación del brazo.

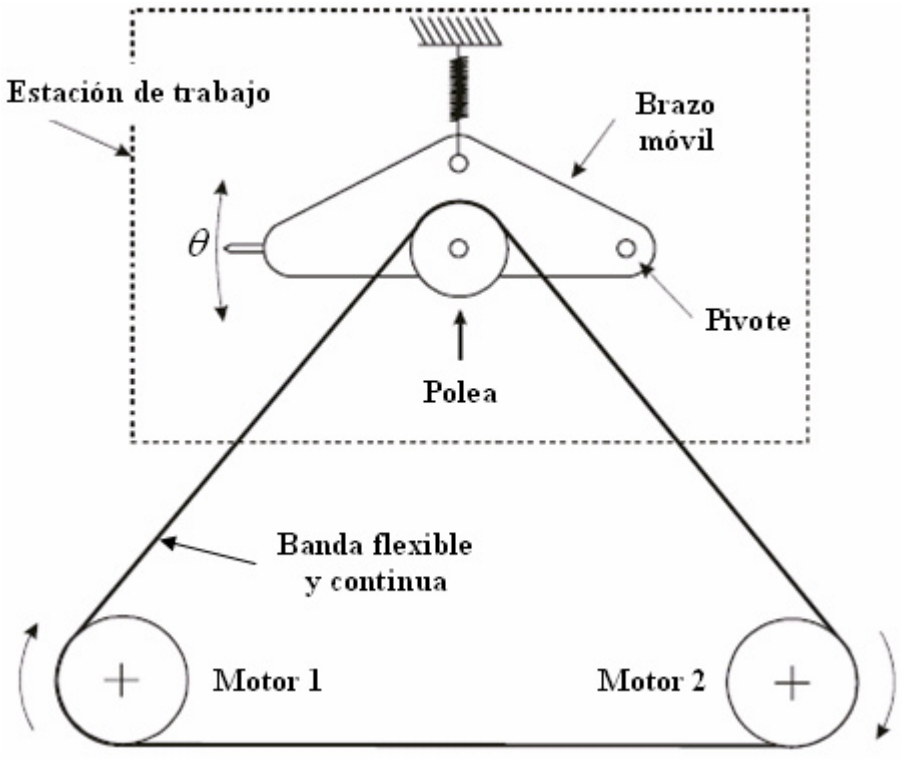


Fig. 3.10: Sistema de motores acoplados.

3.2.2. Control clásico.

En la figura 3.11 se muestra un modelo aproximado del sistema de motores acoplados [11] (figura 3.8). Éste está formado por los subsistemas $G\omega(s)$ y $G\theta(s)$ los cuales estiman la velocidad angular de la polea y el ángulo de inclinación del brazo respectivamente. La entrada del subsistema $G\omega(s)$ es la suma de las dos entradas $(u1+u2)$ y, la del subsistema $G\theta(s)$ es la resta de estas dos $(u2-u1)$.

En [11] se obtuvieron unos modelos aproximados para las funciones de transferencia $G\omega(s)$ y $G\theta(s)$ y son los siguientes:

$$G\omega(s)=\frac{1}{0.3s+1}$$

$$G\theta(s)=\frac{-185600}{(s^2+11s+150)(s^2+1.6s+800)}$$

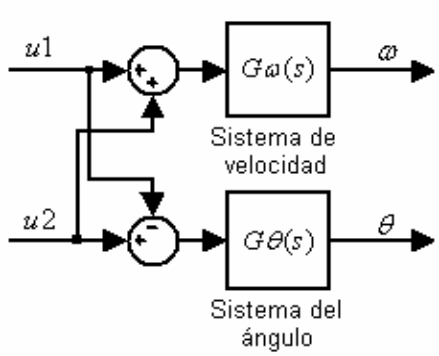


Fig. 3.11: Diagrama de bloques interno del sistema.

En la figura 3.12 se muestra el sistema controlado mediante los controladores $K\omega(s)$ y $K\theta(s)$ para la velocidad de la polea y el ángulo de inclinación del brazo respectivamente. El uso de un pre-compensador permite tratar al sistema MIMO como dos sistemas SISO independientes puesto que con éste se consigue separar las dinámicas de la velocidad angular ω y del ángulo del brazo θ. Por lo tanto la acción de control generada por cada controlador afecta solamente al sistema que está controlando.

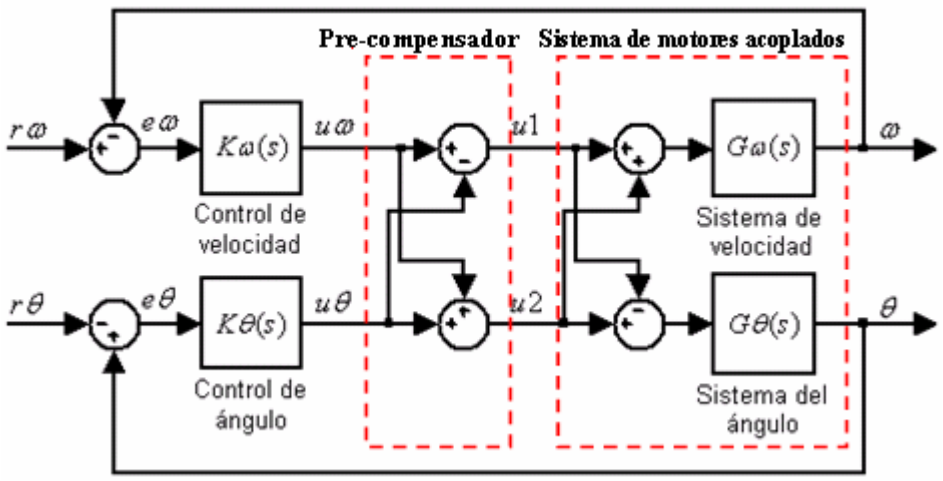


Fig. 3.12: Diagrama de bloques del sistema controlado.

El sistema MIMO controlado tendrá las mismas salidas ω y θ pero sus entradas ahora serán los valores de referencia para la velocidad angular $r\omega$ y ángulo del brazo $r\theta$:

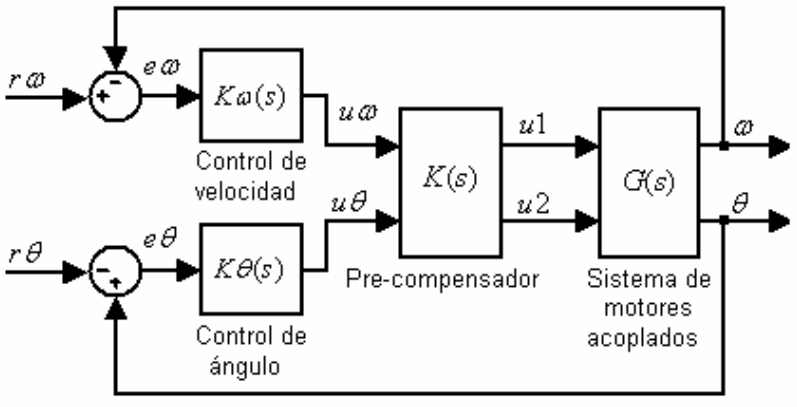


Fig. 3.13: Esquema del sistema de controlado.

Ambos controladores son de tipo proporcional integral (PI) y sus respectivos parámetros k_p y T_i son sintonizados en línea y varían dependiendo del periodo de muestreo T_s :

$$K(z) = k_p\left(1 + \frac{T_s}{T_i(1 - z^{-1})}\right)$$

3.2.3. Resultados de las implementaciones.

El software empleado para implementar los diferentes sistemas de control fue *Labview* y se hizo uso especialmente de las herramientas *Quadratic Programming.vi* (de la Paleta *optimization)* para calcular los movimientos mediante programación cuadrática y *PID.vi* (de la paleta *PID)* para realizar el control clásico.

El periodo de muestreo o de ejecución para cada controlador fue de $T_s = 60\,ms$, por ende los controladores PI se sintonizaron para ese periodo y el modelo de respuesta al escalón para el DMC se formó a partir de la medición de las salidas cada 60ms. Se tomó este periodo de muestreo debido a que éste era el periodo mínimo de trabajo del QDMC.

Los controles QDMC y DMC RLS emplearon los mismos parámetros de sintonización y los límites de las diferentes variables del proceso las cuales

son manejadas en porcentajes de los valores máximos de funcionamiento, se muestran a continuación:

- Límites de velocidades de cambio máximas y mínimas de las acciones de control:

$$\Delta u1min = -100\%$$
$$\Delta u1max = 100\%$$
$$\Delta u2min = -100\%$$
$$\Delta u2max = 100\%$$

- Límites de magnitudes máximas y mínimas de las acciones de control:

$$u1min = -100\%$$
$$u1max = 100\%$$
$$u2min = -100\%$$
$$u2max = 100\%$$

- Límites de magnitudes máximas y mínimas de las variables controladas:

$$\omega\,min = -50\%$$
$$\omega\,max = 50\%$$
$$\theta\,min = -20\%$$
$$\theta\,max = 20\%$$

Los límites definidos anteriormente sólo aplicaron para los controladores DMC mientras que el control clásico no estuvo sujeto a ningún tipo de restricción.

Se realizaron una serie de experimentos en donde el funcionamiento del sistema aplicando los diferentes controladores, se compara empleando dos indicadores:

- IAE: es la integral del valor absoluto del error en las variables controladas.

$$IAE = \sum | r\omega_i - \omega_i | + \sum | r\theta_i - \theta_i |$$

- IADu: es la integral del valor absoluto del incremento en las acciones de control.

$$IADu = \sum | \Delta u1_i | + \sum | \Delta u2_i |$$

Experimento 1: éste consiste en llevar las variables controladas al mismo tiempo desde unas posiciones iniciales hasta unas finales.

Las figuras 3.14, 3.15 y 3.16 representan las gráficas de evolución de las salidas del sistema y de las entradas aplicadas al proceso, correspondientes al control clásico, QDMC y DMC RLS respectivamente. En éstas se puede ver que en todos los casos las salidas se estabilizan en sus respectivos valores de referencias sin que existan infracciones de algún tipo. En los tres casos los tiempos de establecimiento de las señales son parecidos, pero aún así las señales se estabilizan más rápido con los dos controles DMC y con sobrepaso despreciables frente a los presentados por el control PI.

En la siguiente tabla se muestran los indicadores IAE e IADu de los tres controles, donde se observa que el menor IAE lo presenta el DMC RLS, seguido de cerca por el QDMC. El indicador IADu más bajo lo presenta el control DMC RLS, seguido por el QDMC y un poco más lejos el control PI.

Experimento 1	**Control PI**	**QDMC**	**DMC RLS**
IAE	578,2	575,5	574,9
IADu	153,8	138,7	133,7

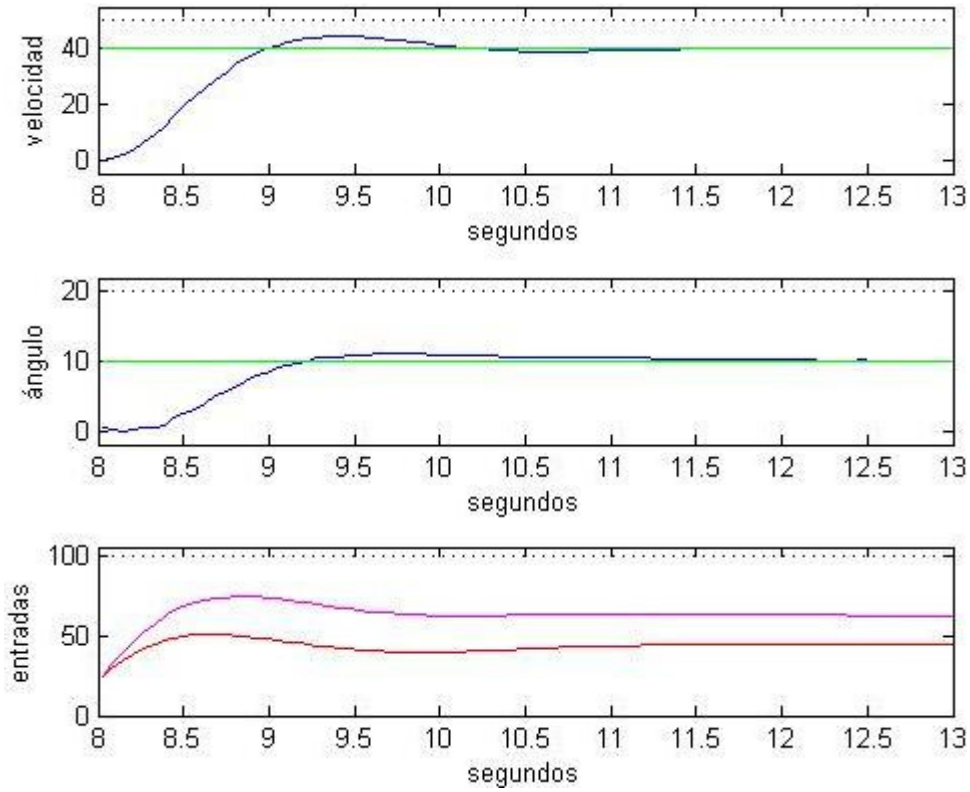


Fig. 3.14: Experimento 1, Control clásico.

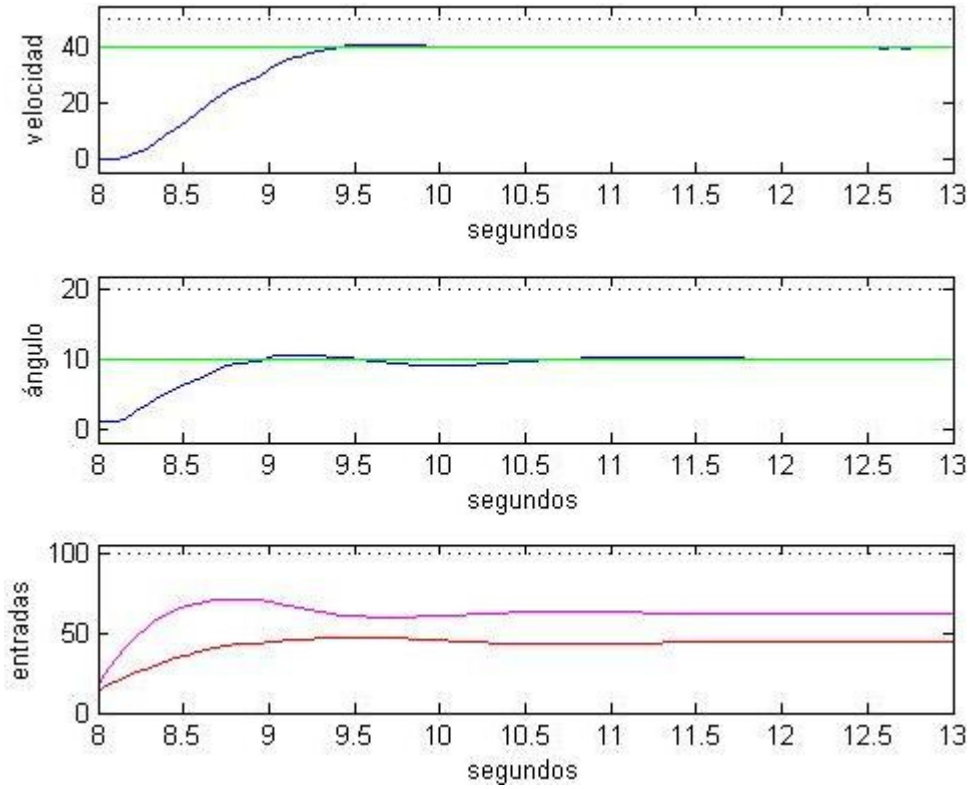


Fig. 3.15: Experimento 1, QDMC.

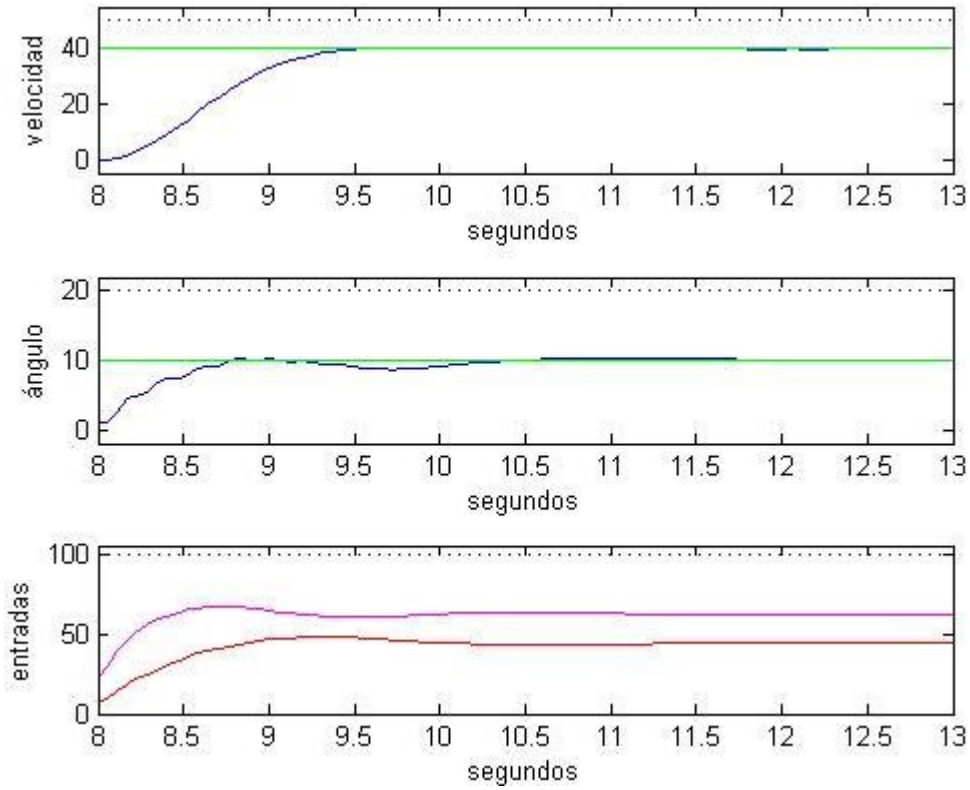


Fig. 3.16: Experimento 1, DMC RLS.

Experimento 2: éste consiste en cambiar solamente el valor de referencia de la velocidad de la polea, para de esta forma medir el efecto de dicho cambio sobre la variable del ángulo de inclinación del brazo cuyo valor de referencia permanece constante.

Aquí se observa que los tiempos de establecimiento de la velocidad angular son muy parecidos, pero la transición de ésta con los dos controles DMC es más suave. Por otra parte el efecto del cambio de referencia de la velocidad angular sobre el ángulo del brazo, no es considerable dado que los controles lo corrigen rápidamente. En cuanto a los indicadores IAE, es más pequeño el del control clásico, seguido por el DMC RLS y el QDMC. Los indicadores IADu más bajos los presentan los dos controles DMC siendo el más bajo el del DMC RLS.

Experimento 2	Control PI	QDMC	DMC RLS
IAE	344,4	377,5	360,9
IADu	110,2	95,5	89,7

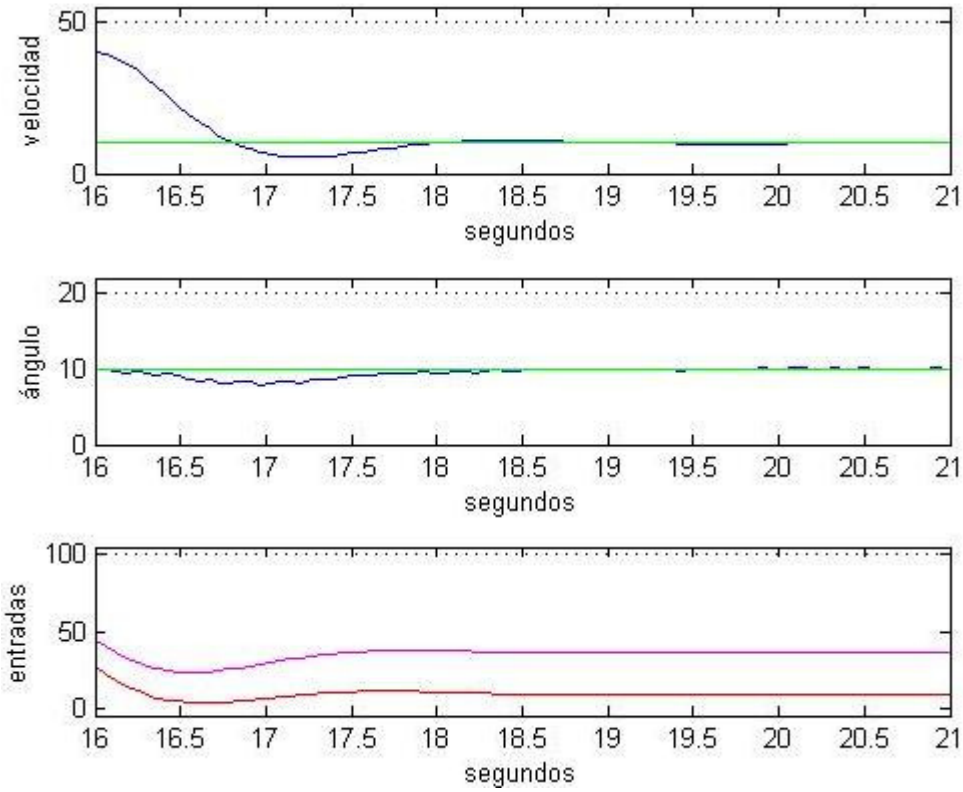


Fig. 3.17: Experimento 2, Control clásico.

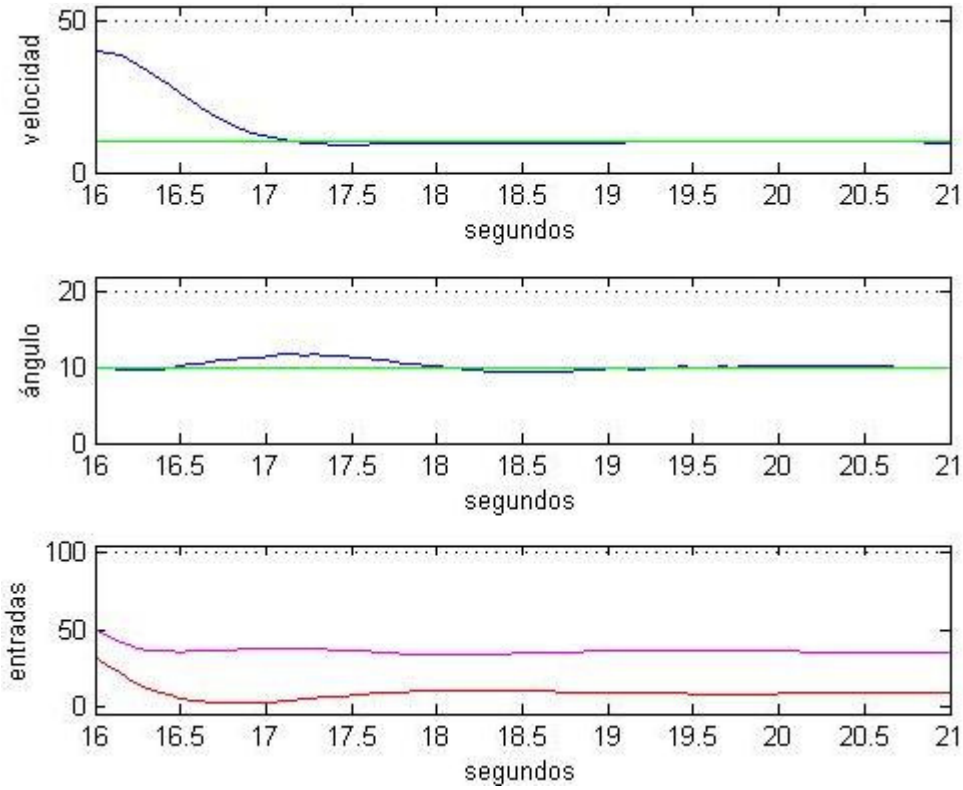


Fig. 3.18: Experimento 2, QDMC.

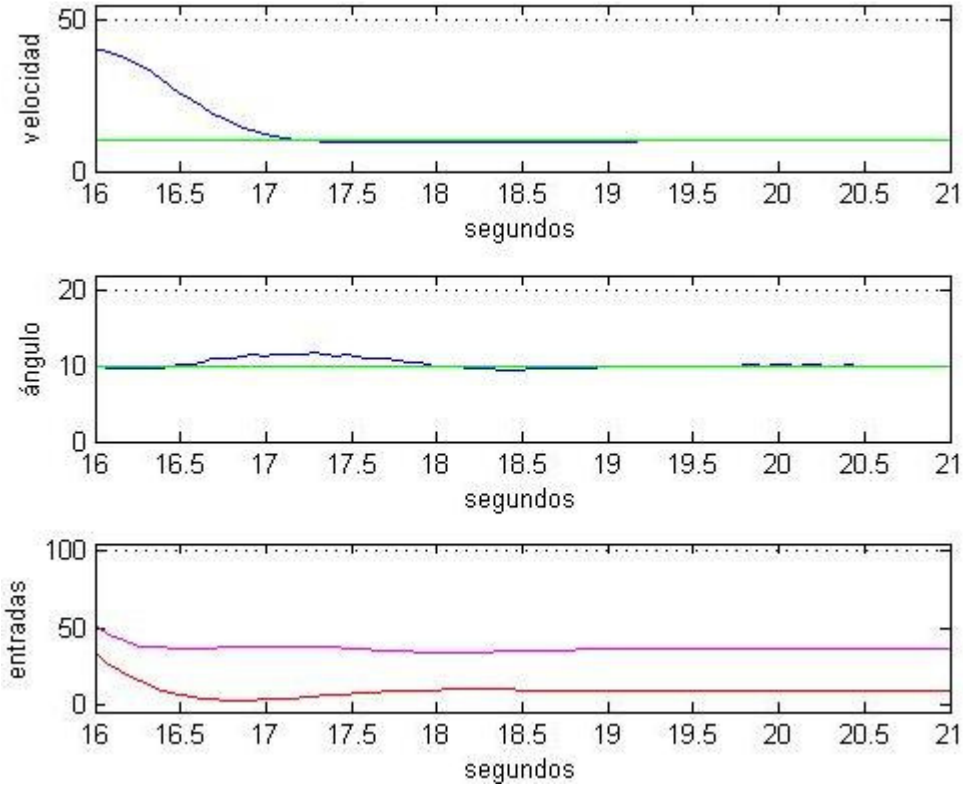


Fig. 3.19: Experimento 2, DMC RLS.

Experimento 3: éste es similar al experimento anterior sólo que el cambio en la referencia será sobre el ángulo de inclinación del brazo.

En los tres casos el ángulo del brazo se estabiliza en su respectivo valor de referencia con tiempos de establecimiento parecidos y, la transición más suave se presenta con el control PI. El efecto de este cambio de referencia sobre la velocidad angular la cual se mantiene en su valor de referencia es despreciable.

Los indicadores IAE son muy parecidos y el menor valor lo presenta el DMC RLS. En cuanto a los indicadores IADu, el más bajo lo presenta el control PI mientras que los de los DMC son considerablemente altos. Esto se debe a que se realizan más maniobras para corregir las sobreoscilaciones que se presentan en el ángulo del brazo. .

Experimento 3	Control PI	QDMC	DMC RLS
IAE	169,2	173,8	162,3
IADu	53,0	98,1	101,4

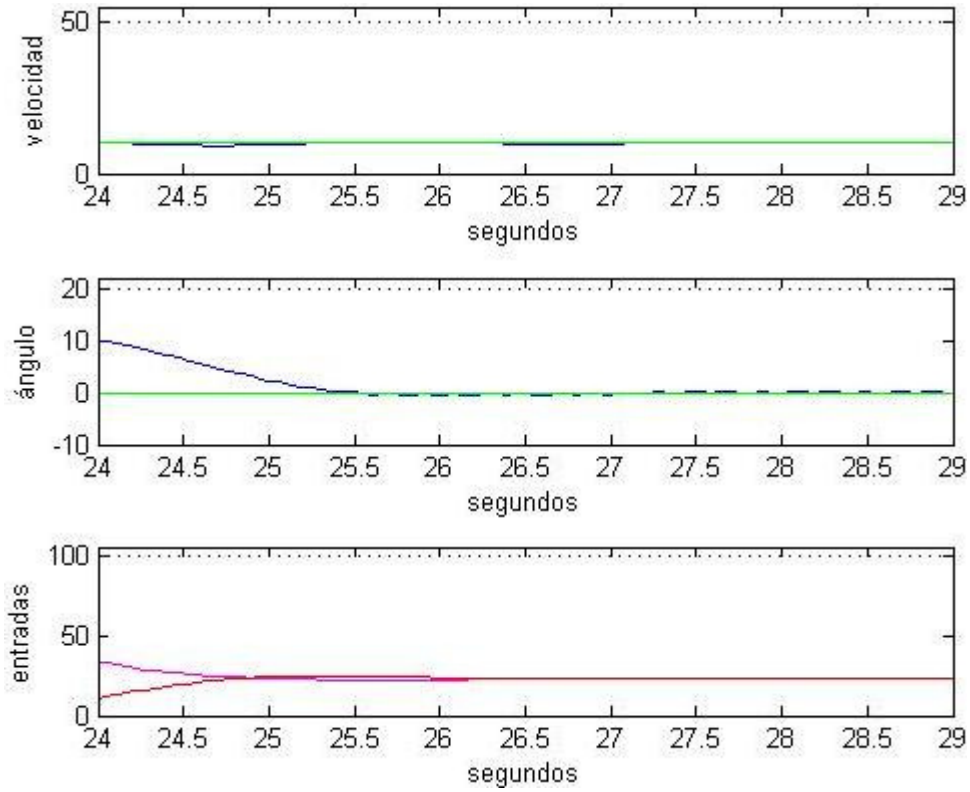


Fig. 3.20: Experimento 3, Control clásico.

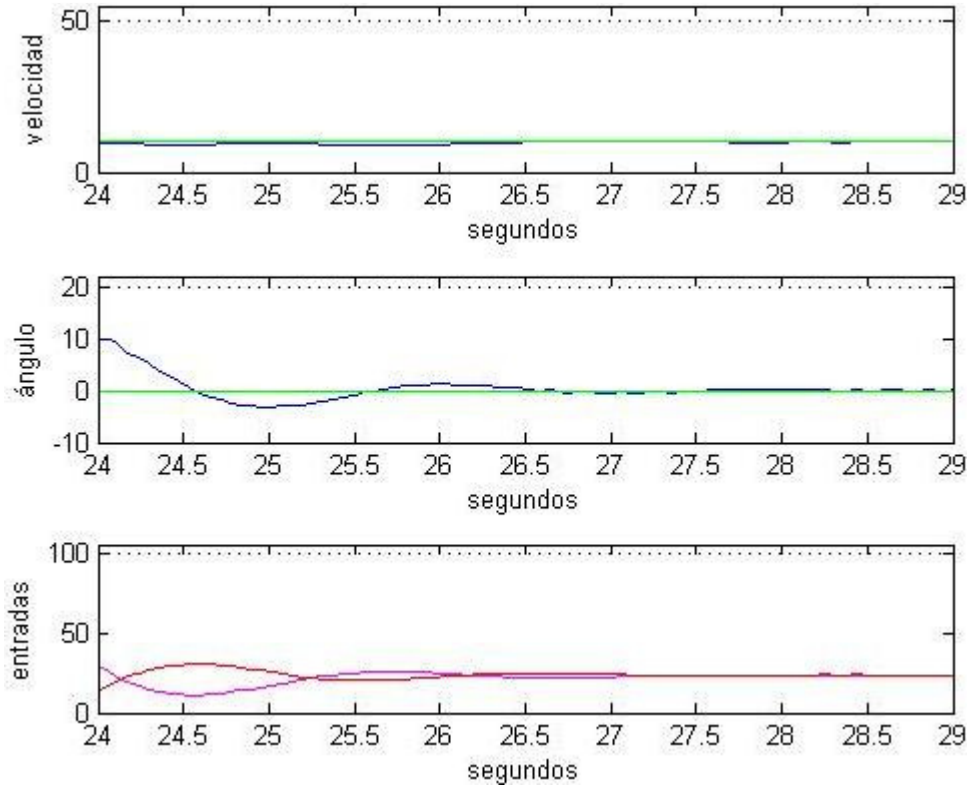


Fig. 3.21: Experimento 3, QDMC.

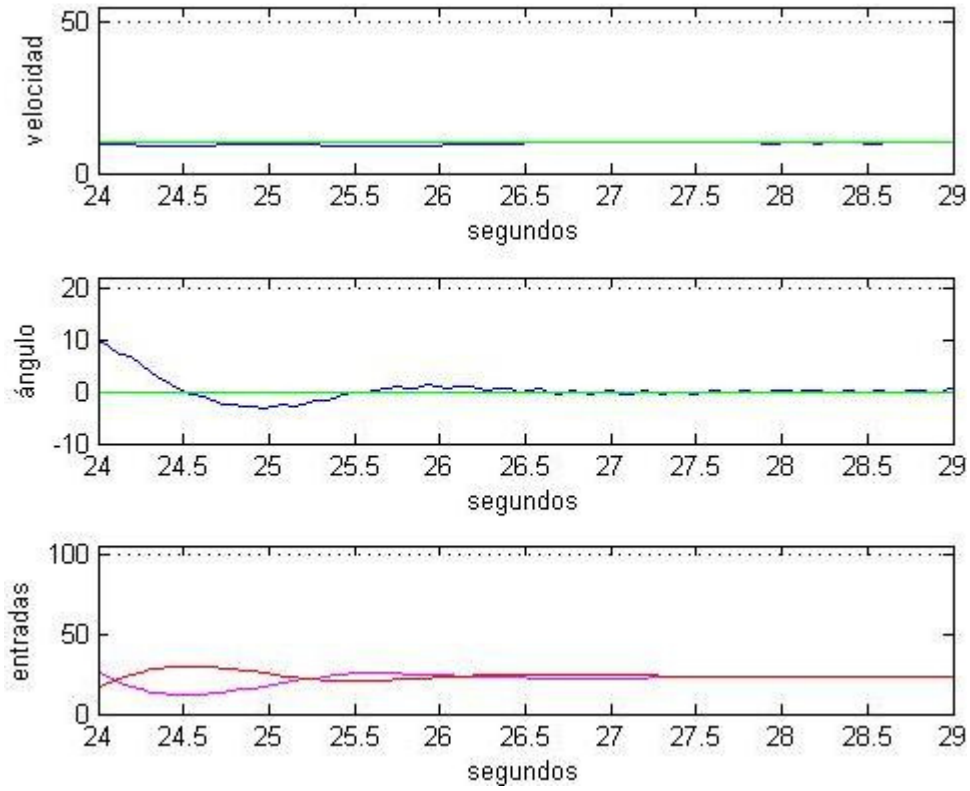


Fig. 3.22: Experimento 3, DMC RLS.

Experimento 4: aquí se busca observar el comportamiento del sistema, ante la reducción de la velocidad de variación de las variables controladas ($\Delta u \max = 1\%$ y $\Delta u \min = -1\%$) mientras que los límites máximos de estas se mantienen ($u \max = 100\%$, $u \min = -100\%$).

Como se puede observar en las figuras 3.24 y 3.25, empleando los dos DMC's las variables controladas se posicionan en sus respectivos valores de referencia, con tiempos de establecimiento parecidos y de manera más suave que en los casos anteriores debido a las nuevas restricciones las cuales se respetan en todo momento. Por otra parte, el sistema controlado con los controladores PI, presenta un comportamiento es muy pobre, donde se observan sobrepasos grandes y prolongados en ambas variables controladas con tiempos de establecimiento demasiado altos. Este deficiente control se debe a que al limitar la velocidad de variación de las entradas, el sistema no se puede posicionar de la manera agresiva como normalmente lo hace donde al comienzo las entradas aplicadas al proceso y generadas por los controladores son muy altas contrarias a este caso.

Como era de esperarse el indicador IAE más alto lo presenta el control clásico siendo casi el doble de los DMC's, mientras que el indicador más bajo lo presenta el DMC RLS. De igual manera el mayor IADu pertenece al control clásico debido a que las entradas aplicadas al proceso son elevadas y en un momento una de éstas presenta saturación, además de aplicar incrementos durante toda la transición. De los dos DMC's, el IADu más bajo es el del DMC RLS pero en ambos casos las entradas aplicadas al proceso son de bajas magnitudes además de generar incrementos sólo al comienzo y durante tiempos cortos.

Experimento 4	Control PI	QDMC	DMC RLS
IAE	3458,1	1912,2	1502,6
IADu	320,6	139,0	107,6

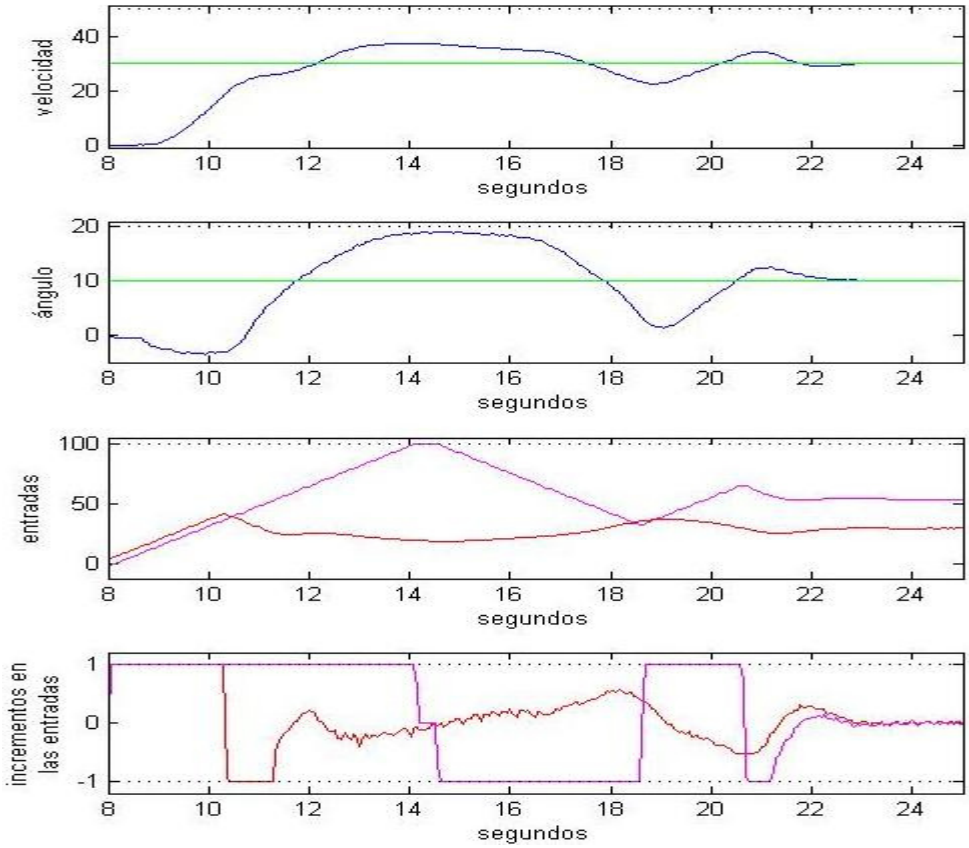


Fig. 3.23: Experimento 4, Control clásico.

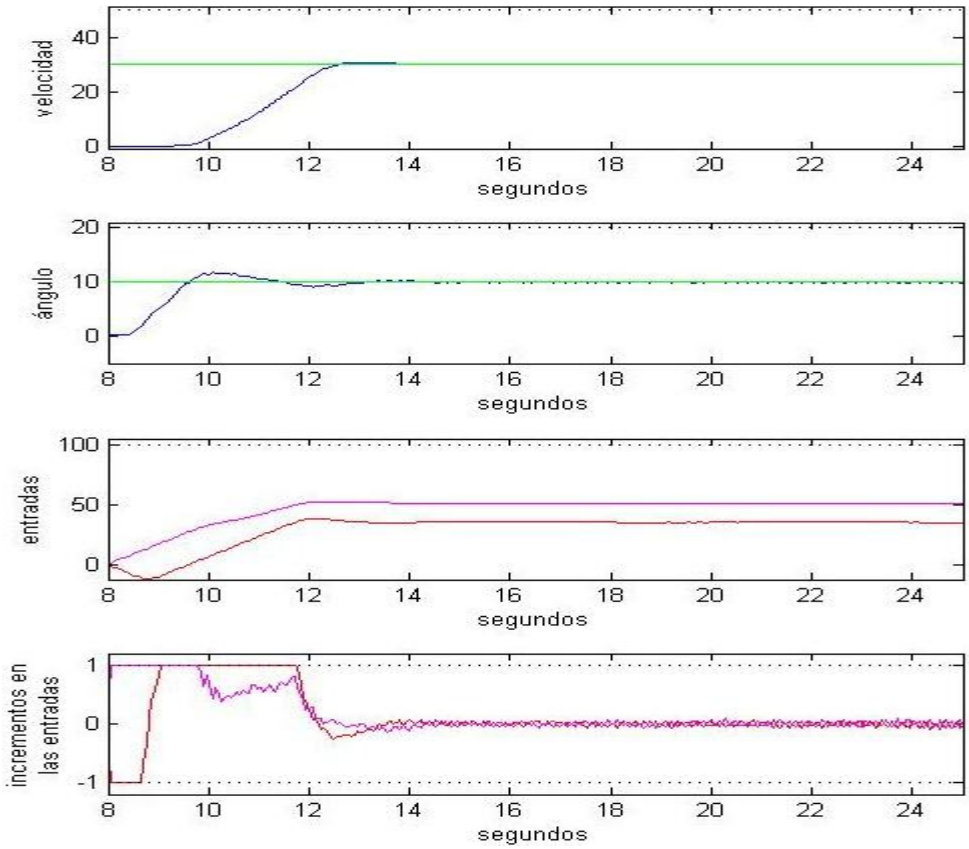


Fig. 3.24: Experimento 4, QDMC.

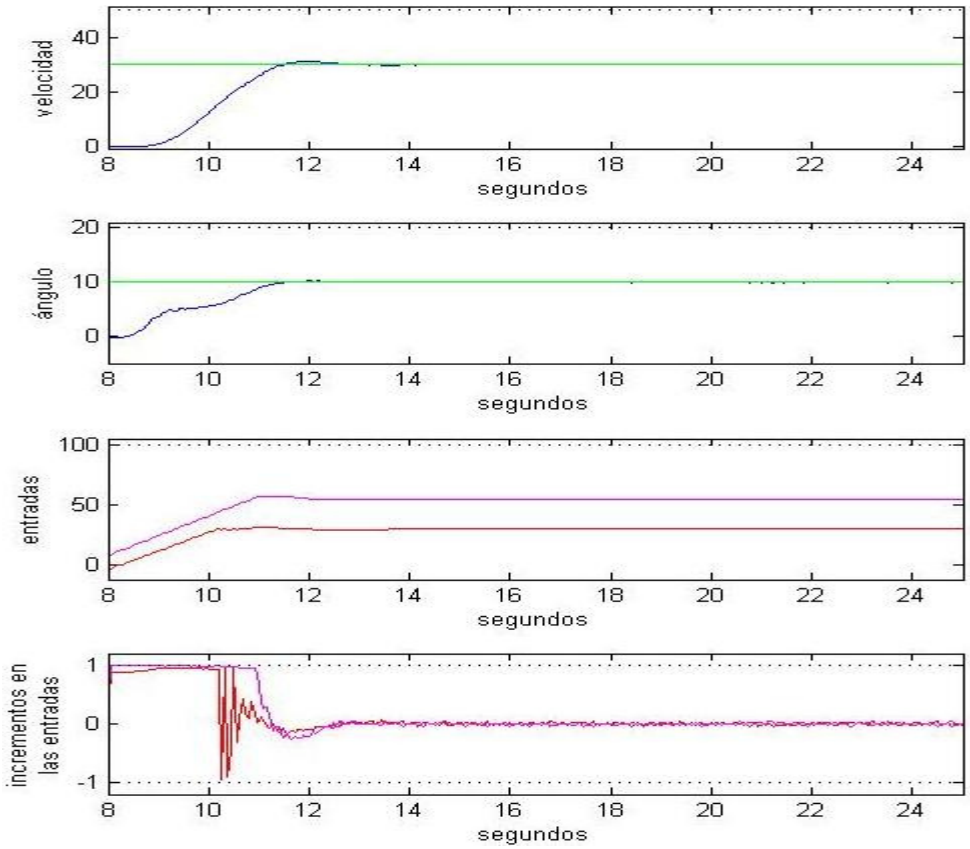


Fig. 3.25: Experimento 4, DMC RLS.

Experimento 5: éste es similar al del caso anterior, con las mismas referencias; sólo que se amplía un poco más el rango de la velocidad de variación de las variables controladas mientras que sus magnitudes máximas permitidas se mantienen:

$$\Delta u \max = 2\%$$
$$\Delta u \min = -2\%$$
$$u \max = 100\%$$
$$u \min = -100\%$$

Al igual que en el experimento anterior, el sistema controlado por PI's presenta sobrepasos altos pero no tan prolongados, consiguiendo posicionar las variables controladas en sus respectivos valores de referencia con tiempos de establecimiento mucho más cortos y por ende reduciendo el indicador IAE casi al 50% (1939,8).

En cuanto al indicador IADu, no se aprecia una mejora, por el contrario éste presenta un leve aumento ya que se siguen produciendo entradas altas y con variaciones sostenidas durante toda la experiencia.

Por otra parte, los indicadores más bajos de los dos DMC's los presenta el DMC RLS y con respecto a la experiencia anterior, ambos reducen considerablemente los indicadores IAE mientras que los IADu casi que se mantienen.

Experimento 5	Control PI	QDMC	DMC RLS
IAE	1939,8	1249,8	1027,6
IADu	353,2	140,0	123,6

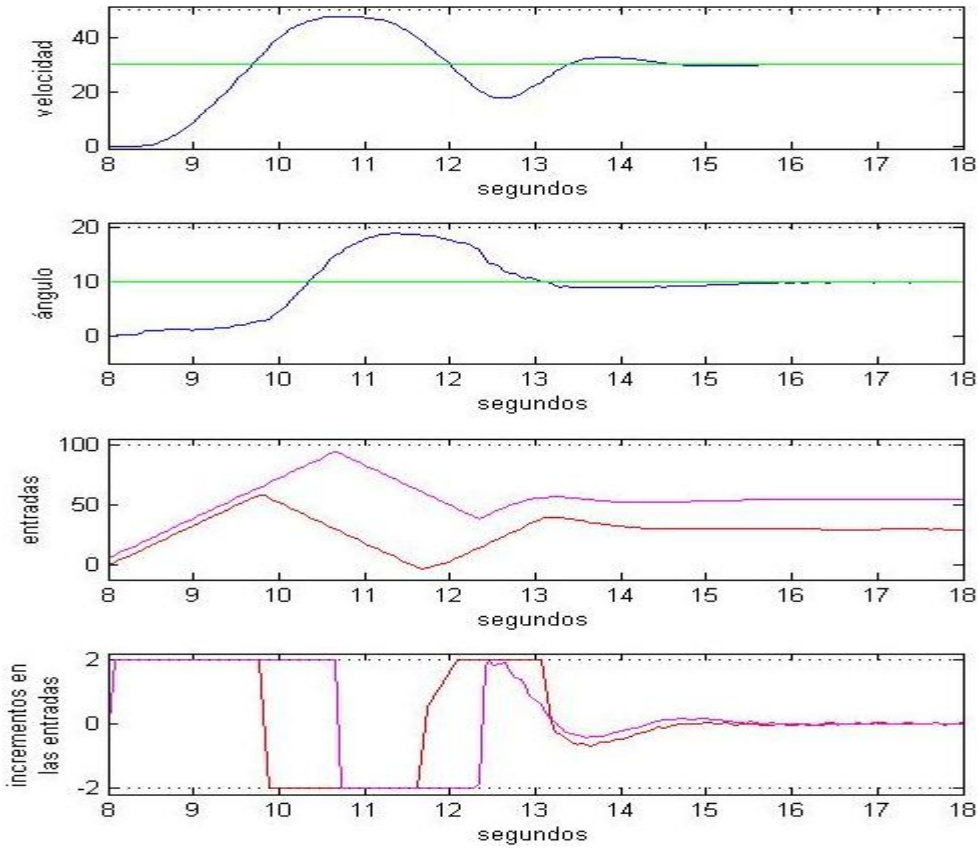


Fig. 3.26: Experimento 5, Control clásico.

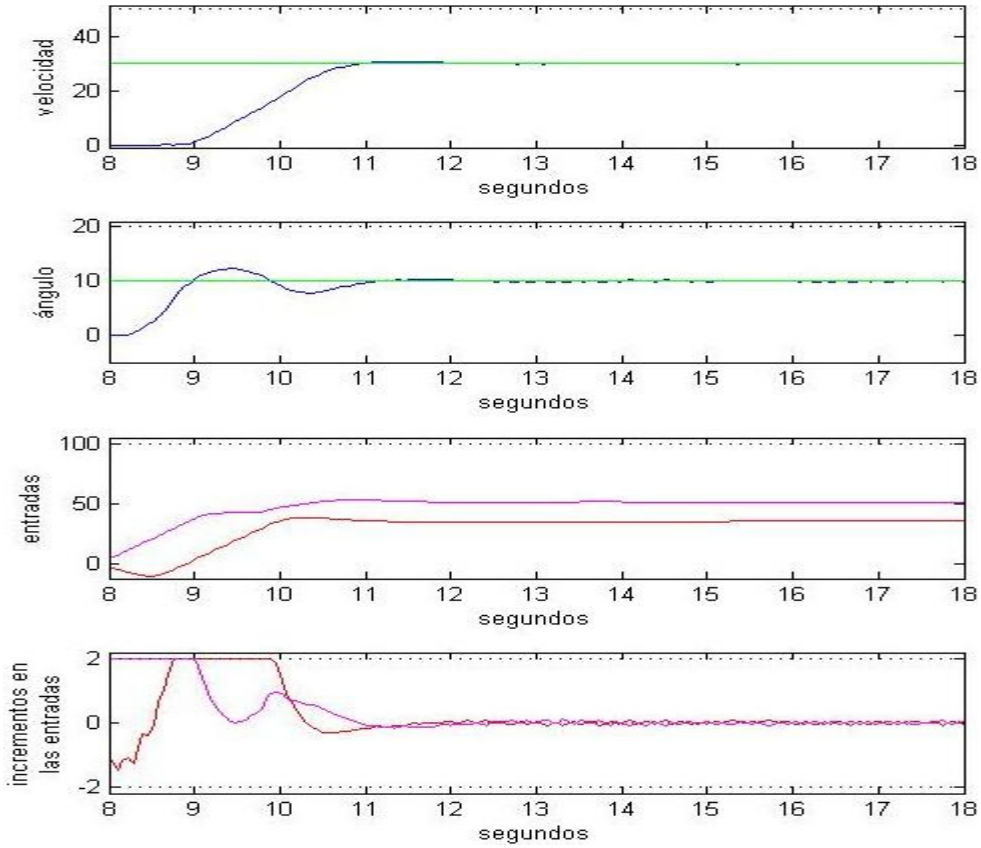


Fig. 3.27: Experimento 5, QDMC.

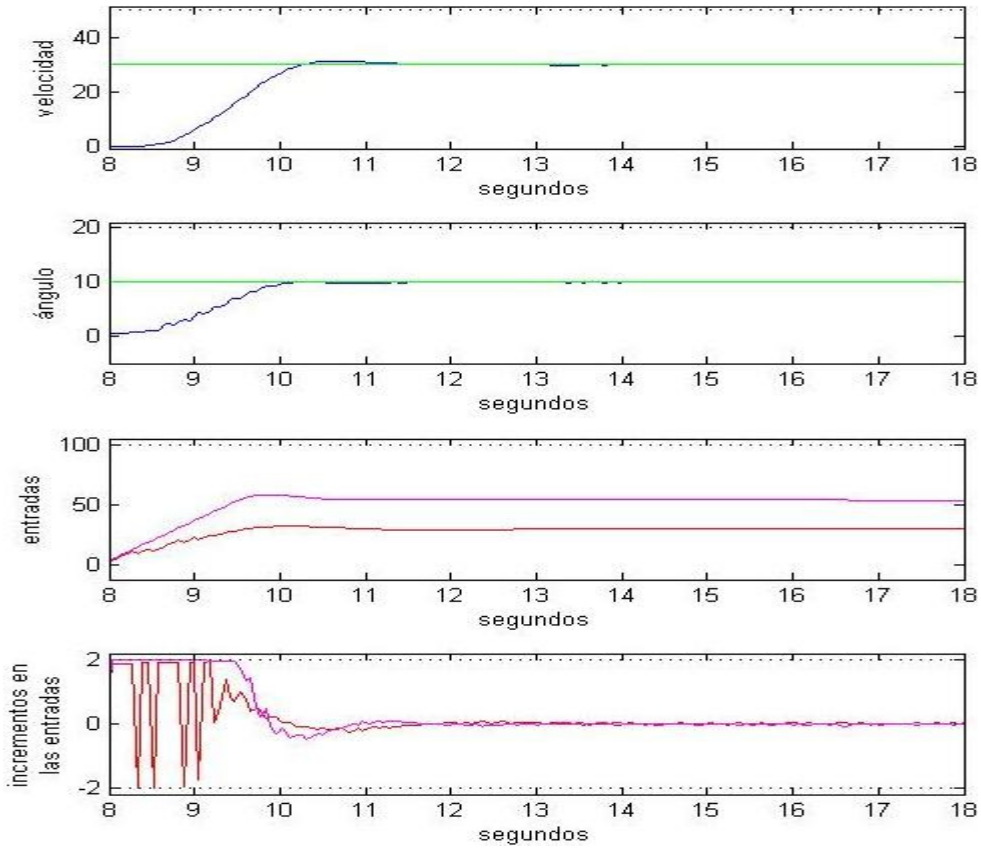


Fig. 3.28: Experimento 5, DMC RLS.

Experimento 6: En este experimento se mantienen los valores de referencias para las dos variables controladas pero, los límites máximos y mínimos de las variables manipuladas así como sus velocidades de variación son reducidos a los siguientes valores:

$$\Delta u \max = 1\%, \Delta u \min = -1\%, u \max = 70\% \text{ y } u \min = -70\%$$

Con estos nuevos límites no se consigue mejorar el comportamiento del sistema controlado con el control clásico, por lo contrario, éste presenta un comportamiento muy similar al del la experiencia cuatro con indicadores IAE e IADu similares. El sistema controlado con los dos controles DMC presenta comportamientos similares y con buenos desempeños. No obstante los indicadores más bajos los presenta el DMC RLS.

Experimento 6	Control PI	QDMC	DMC RLS
IAE	3129,2	2020,1	1464,1
IADu	237,1	147,6	104,6

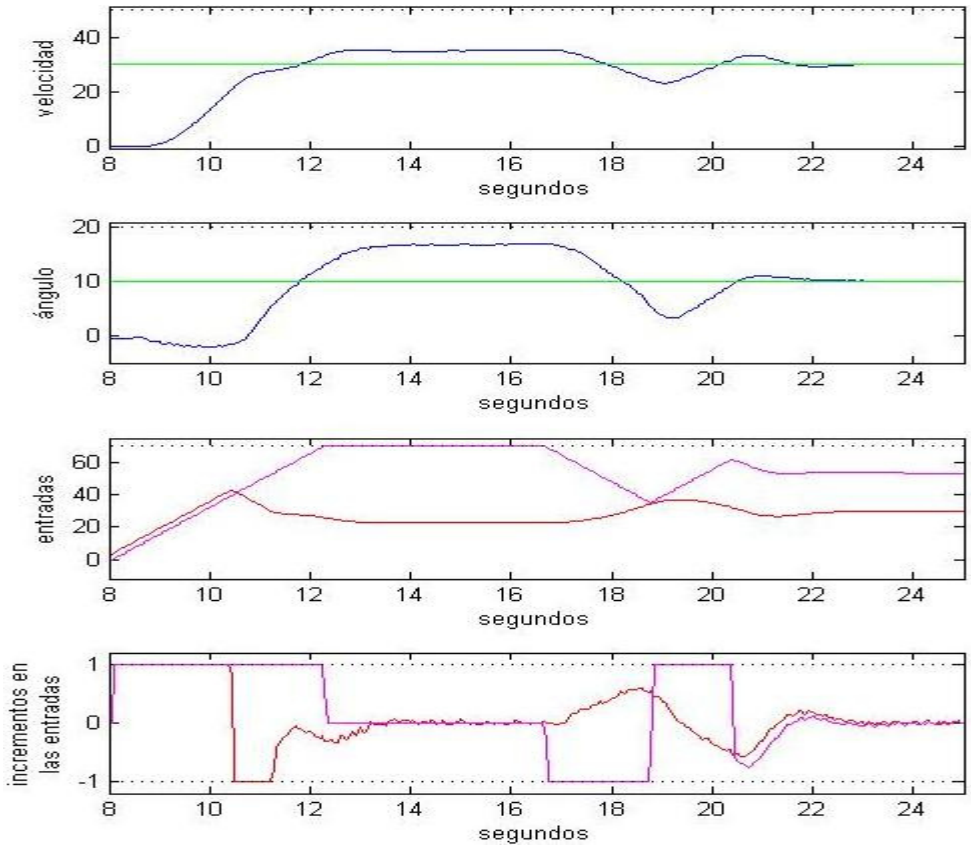


Fig. 3.29: Experimento 6, Control clásico.

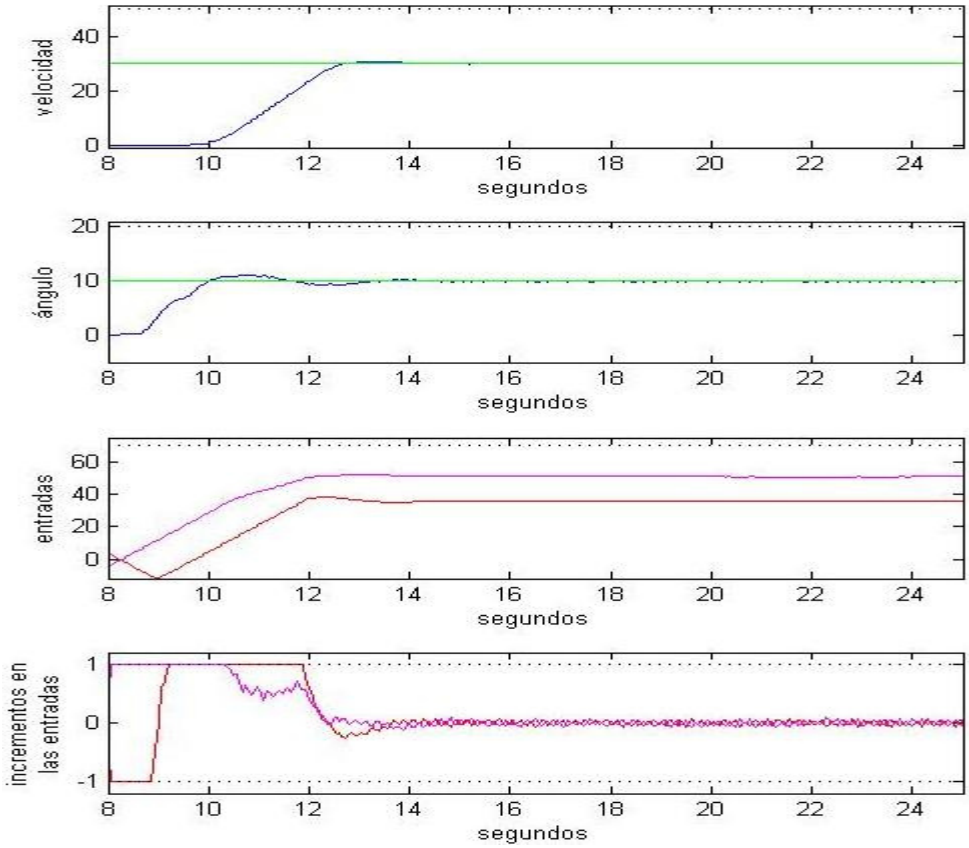


Fig. 3.30: Experimento 6, QDMC.

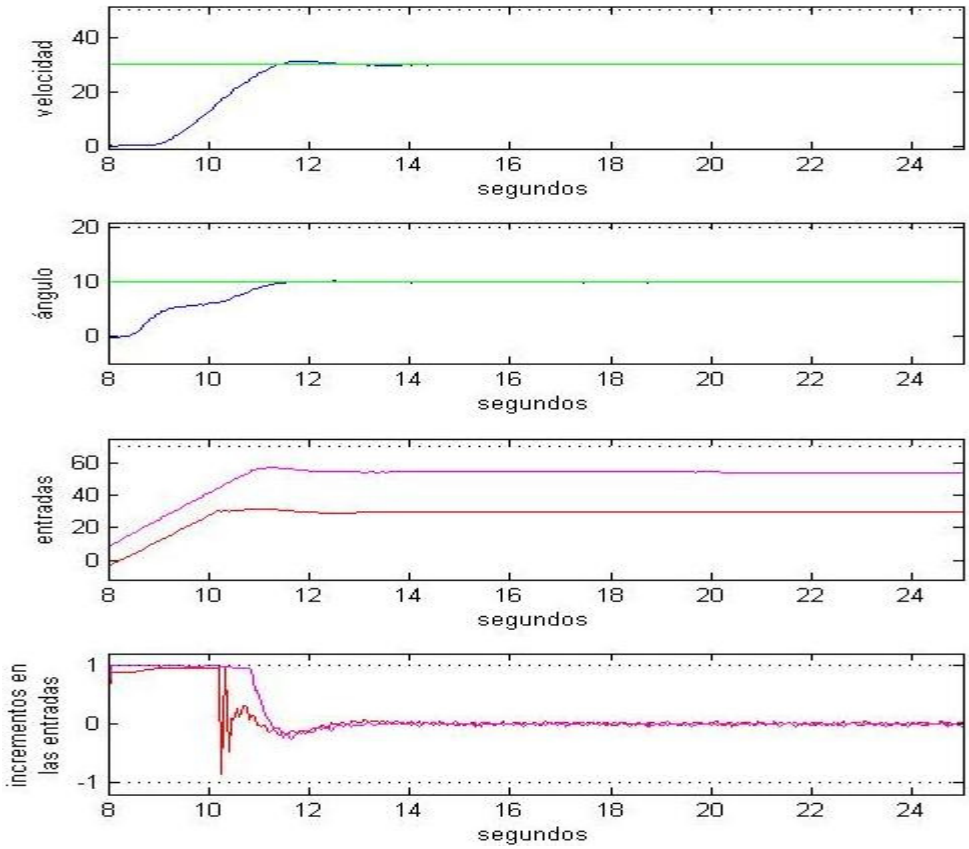


Fig. 3.31: Experimento 6, DMC RLS.

Experimento 7: experiencia similar a la del caso anterior, manteniendo las mismas referencias; sólo que se amplía un poco más el rango de la velocidad de variación de las variables controladas mientras que sus magnitudes máximas permitidas se mantienen:

$$\Delta u \max = 2\%$$
$$\Delta u \min = -2\%$$
$$u \max = 70\%$$
$$u \min = -70\%$$

Comparado con el caso anterior, el sistema controlado con los PI's consigue mejorar un poco su comportamiento, reduciendo los tiempos de establecimiento. No obstante se siguen presentando sobrepasos altos en ambas variables controladas y entradas altas con variaciones durante toda la experiencia.

Los tiempos de establecimiento con el control clásico comparados con los del sistema controlado con los dos DMC's, siguen siendo altos dado que con los DMC's las variables controladas se posicionan rápidamente en sus respectivos valores de referencia y con sobrepasos poco considerables.

Finalmente como era de esperarse los indicadores IAE e IADu más altos los presenta el sistema controlado con PI's, mientras que los más bajos los arroja el DMC RLS.

Experimento 7	**Control PI**	**QDMC**	**DMC RLS**
IAE	1728,3	1314,1	954,0
IADu	212,3	146,9	118,3

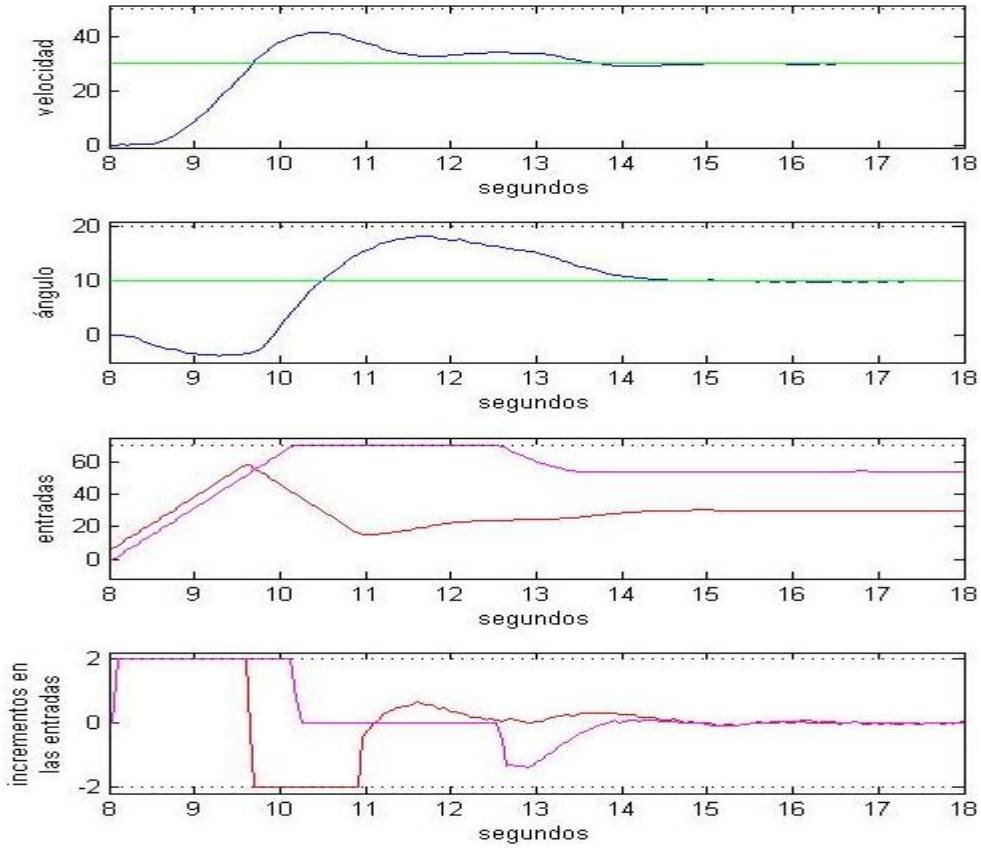


Fig. 3.32: Experimento 7, Control clásico.

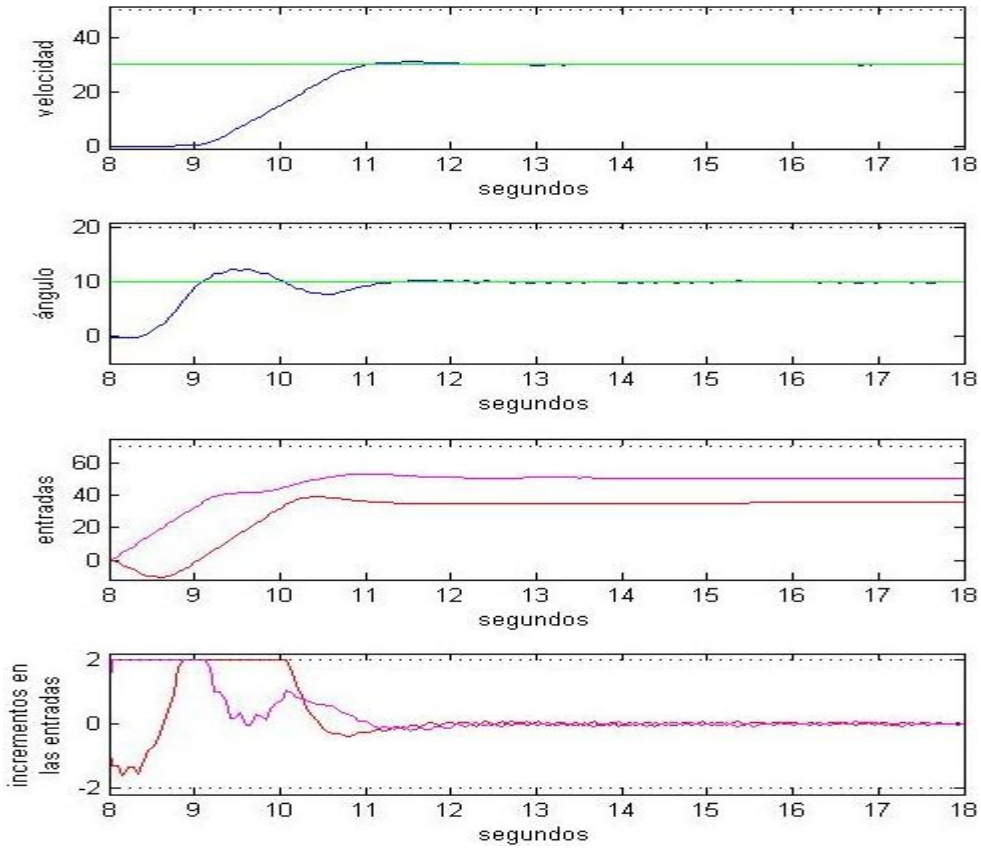


Fig. 3.33: Experimento 7, QDMC.

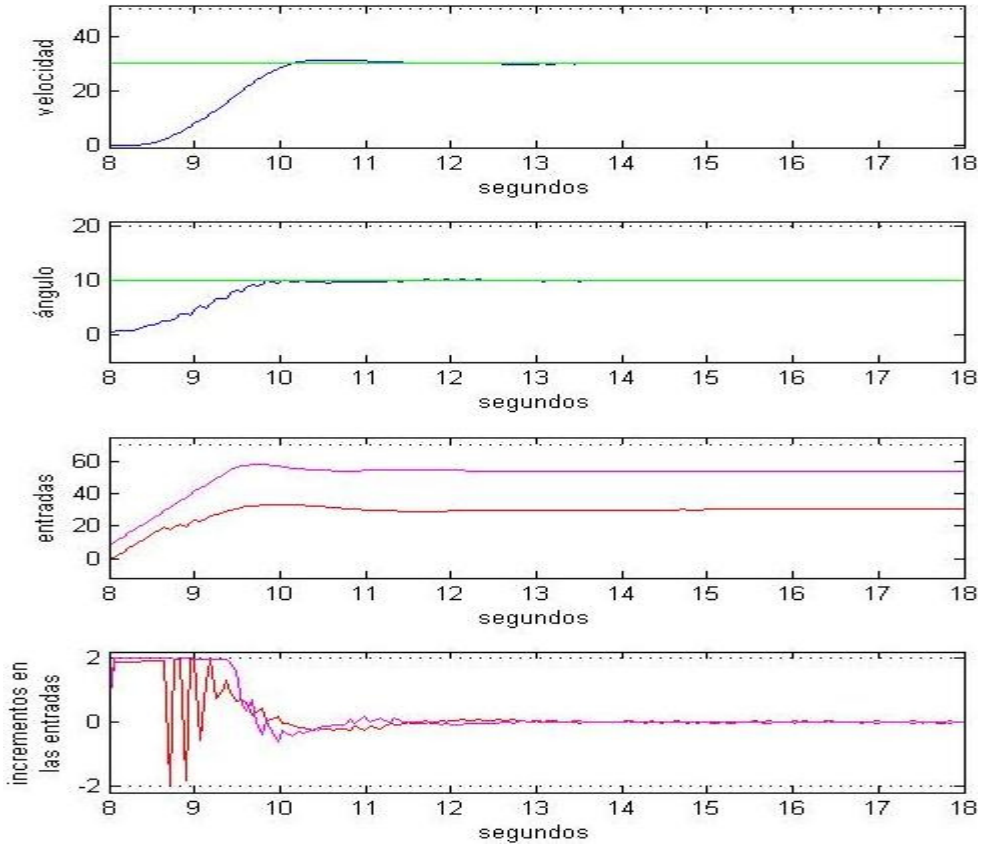


Fig. 3.34: Experimento 7, DMC RLS.

3.2.4. Observaciones.

- El sistema controlado con un DMC con restricciones y con un control clásico sin restricciones, presenta comportamientos similares donde en la mayoría de los casos el primero es el más eficiente.
- Cuando el sistema controlado con un control clásico con restricciones presenta saturaciones en sus entradas e incrementos de éstas, puede presentar comportamientos demasiado pobres y en ocasiones inestables; mientras que con un control con DMC éstos son muy buenos ya que no se presentan sobrepasos elevados, tiempos de establecimientos altos y siempre se respetan las diferentes restricciones.
- Los controles DMC en la mayoría de los casos presentan los indicadores IAE e IADu más bajos con relación al control clásico, debido a que éstos siempre buscan las entradas óptimas a aplicar en el proceso y que reducen los errores de predicción.

- Como se observó en la mayoría de las experiencias, el control con DMC RLS presenta un mejor desempeño que el conseguido con el QDMC puesto que con este algoritmo se consiguieron los indicadores IAE e IADu más bajos.
- Entre más pequeño sea el periodo de muestreo, más coeficientes tendrá el modelo de respuesta al escalón del control DMC y en consecuencia aumenta la cantidad de operaciones que se realiza en cada instante de muestreo aumentando así el coste computacional.
- Los dos algoritmos DMC ejercen un buen control y respetan las restricciones mientras exista una solución posible.
- El manejo de restricciones usando el algoritmo de Mínimos Cuadrados Iterativos, es mucho más rápido en términos computacionales que el de Programación cuadrática.
- La rapidez del algoritmo de mínimos cuadrados radica en que éste sólo se ejecuta cuando se infringe un límite y la optimización finaliza al encontrar el primer vector de incrementos que respete las restricciones.
- La lentitud del algoritmo de programación cuadrática se debe a que éste siempre busca el vector de incrementos óptimo que respeta las restricciones, mientras que el método de mínimos cuadrados iterativos no lo garantiza.
- El uso del método de programación cuadrática es muy conveniente cuando se tienen en cuenta factores económicos o cuando los periodos de lazo permiten su uso; mientras que el método de mínimos cuadrados es de gran utilidad cuando se controlan procesos con dinámicas rápidas.
- El uso del algoritmo de mínimos cuadrados iterativos es más viable que el de de programación cuadrática cuando se desea trabajar con periodos de muestreo muy pequeños.

CAPÍTULO 4

CONCLUSIONES

4.1. CONCLUSIONES.

1. Los controladores MBPC no manejan cálculos complejos, los conceptos resultan intuitivos y su sintonización es sencilla.
2. Tienen aplicación a una amplia gama de procesos industriales que pueden presentar comportamientos dinámicos complicados como acoplamientos entre variables, retardos de transporte, inestabilidades o incluso restricciones en algunas de sus variables, además de tener una facilidad para su extensión del caso SISO para controlar procesos multivariables.
3. El MBPC permite incluir en el problema de control las restricciones a las que está sometido el proceso, tiene un uso práctico en procesos en los que las referencias futuras son conocidas (robótica) y presenta gran robustez ante errores de modelado y presencia de perturbaciones no medibles.
4. El MBPC tiene una carga computacional elevada debido a la cantidad de operaciones que realiza, condicionando así su uso en procesos con dinámicas rápidas.
5. Con los algoritmos QDMC y DMC RLS se resuelve el problema de optimización de forma iterativa, donde no existe expresión analítica para la solución y donde los algoritmos encuentran directamente el vector de acciones de control a aplicar.
6. Los dos algoritmos DMC ejercen un buen control y respetan las restricciones mientras exista una solución posible.
7. Las reacciones de los controladores QDMC y DMC RLS ante la presencia de perturbaciones son muy buenas ya que corrigen rápidamente los errores, intentando que la repercusión sobre las demás variables controladas sean lo menos perceptibles.

8. El manejo de restricciones usando el algoritmo de Mínimos Cuadrados Iterativos (DMC RLS), es mucho más rápido en términos computacionales que el de Programación cuadrática y su rapidez radica en que éste sólo se ejecuta cuando se infringe un límite y la optimización finaliza al encontrar el primer vector de incrementos que respete las restricciones.
9. La lentitud del algoritmo de programación cuadrática (QDMC) se debe a que éste siempre busca el vector de incrementos óptimo que respeta las restricciones, mientras que el método de mínimos cuadrados iterativos no lo garantiza.
10. El uso del método de programación cuadrática es muy conveniente cuando se tienen en cuenta factores económicos o cuando los periodos de lazo permiten su uso; mientras que el método de mínimos cuadrados es de gran utilidad cuando se controlan procesos con dinámicas rápidas.
11. El control clásico es levemente mejor que los dos DMC debido a que no se manejan restricciones de ningún tipo y a su simplicidad.
12. Los controles DMC presentan IADu's más bajos con relación al control clásico, debido a que éstos encuentran las entradas óptimas a aplicar en el proceso y a sus parámetros de sintonización; los cuales con una mejor sintonización se puede conseguir un mejor desempeño.
13. Como se observó en los indicadores IAE e IADu de los diferentes experimentos, el control DMC RLS presenta un mejor desempeño que el QDMC.
14. Entre más pequeño sea el periodo de muestreo, más coeficientes tendrá el modelo de respuesta al escalón del control DMC y en consecuencia aumenta la cantidad de operaciones que se realiza en cada instante de muestreo aumentando así el coste computacional.
15. El uso del algoritmo de mínimos cuadrados iterativos es más viable que el de de programación cuadrática cuando se desea trabajar con periodos de muestreo muy pequeños.

BIBLIOGRAFÍA.

[1] J. Sanchis. *GPC mediante Descomposición en Valores Singulares (SVD). Análisis de Componentes Principales (PCA) y Criterios de Selección*. Tesis doctoral, Universidad Politécnica de Valencia, 2002.

[2] J.A. Méndez. *Desarrollo de Estrategias de Control Predictivas de Alto Rendimiento Mediante la Incorporación de Técnicas de Control Robusto y de Redes Neuronales*. Tesis doctoral, Universidad de la Laguna, 1998.

[3] J.V. Salcedo. *GPCs en Espacio de Estados Para el Control de Sistemas No Lineales*. Tesis Doctoral, Universidad Politécnica de Valencia, 2005.

[4] D.M. Prett., B.L. Ramaker., C.L. Cutler. *Dynamic Matrix Control Method*. United States Patend, 1979.

[5] C. Bordóns. *Control Predictivo: metodología, tecnología y nuevas perspectivas*. Universidad de Sevilla, 2000.

[6] Control Predictivo: *Metodología, tecnología y Perspectiva Histórica. Curso de Técnicas Avanzadas de Optimización y Control de Procesos*, Universidad Politécnica de Valencia, 2008.

[7] C.E. García., D.M. Prett., M. Morari. *Model predictive Control: Theory and Practice – a Survey*. Publicación, International Federation of Automatic Control, Gran Bretaña, 1989.

[8] C. De Prada. *Fundamentos de Control Predictivo de Procesos*. Los Manuales de Ingeniería Química, Instrumentación y Control de Procesos, Universidad de Valladolid, 1997.

[9] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[10] H. Hagadoorn., M. Readman. *Coupled Drives 1: Basics*. www.control-systems-principles.co.uk.

[11] M. Readman., H. Hagadoorn., *Coupled Drives 2: Control and Analysis*. www.control-systems-principles.co.uk.

[12] J. Richalet., A. Rault., J.L. Testud. and J. Papon. *Model Predictive Heuristic Control: Applications to Industrial Processes*. Automatica: A Journal of IFAC, the International Federation of Automatic Control. Volume 14, Issue 5, September 1978, Pages 413-428.

[13] K. Ogata. *Ingeniería de Control Moderna*. Prentice Hall, 1993.

[14] N. Kehtarnavaz. *Digital Signal Processing System Design: Labview Based Hybrid Programming*. Elsevier Inc., 2008.

[15] R. Bitter., T. Mohiuddin., M. Nawrocki. *Labview Advanced Programming Techniques*. CRC Press, 2007.

[16] *Quadratic Programming*. Help, Matlab r2009b.

DOCUMENTO II

ANEXOS

ANEXOS

pág.

ALGORITMOS PARA SIMULACIONES EN MATLAB

DMC SISO

```
clear all; clc
%=================================================================
%          Obtención de los coeficientes Su de la rta al escalón
%=================================================================
Ts=4;%tiempo de muestreo
sis=c2d(tf([0 4.05],[50 1]),Ts,'zoh');%sistema
[numsis densis]=tfdata(sis,'v');
Su=step(sis);
Su=Su(2:end);%vector de coeficientes de respuesta al escalón
N=length(Su);%longitud del vector de coeficientes Su
%=================================================================
%                          Parámetros
%=================================================================
instantes=50;%número de periodos de muestreo que se ejecutará el %algoritmo DMC
Nc=5; %horizonte de control
Np=N+Nc;%horizonte de predicción
alfa=1;%parámetros alfa lambda y W
lambda=1;
r=2;%referencia
%=================================================================
%              Cálculo de las matrices G, Q1, Q2, alfa, lambda y M
%=================================================================
if Np>N;
   Saux=[Su; ones(Np-N,1)*Su(end)];
else
   Saux=Su;
end
G=zeros(Np,Nc);
for i=1:Nc;
   G(i:end,i)=Saux(1:Np-i+1);
end
Q1=zeros(N,N);
Q1(1:end-1,2:end)=eye(N-1,N-1);
Q1(end,end)=1;
Q2=zeros(Np,N);
if Np==N;
   Q2=Q1;
elseif Np<N
   Q2(1:end,2:Np+1)=eye(Np,Np);
else
   Q2(1:N-1,2:end)=eye(N-1,N-1);
   Q2(N:end,end)=1;
end
alfa=alfa*eye(Np,Np);
lambda=lambda*eye(Nc,Nc);
M=inv(G'*alfa*G+lambda)*G'*alfa;
%=================================================================
%                  condiciones iniciales
%=================================================================
t=(0:1:instantes);%vector de tiempo total en instantes
R=r*ones(instantes+Np,1);%vector de trayectoria de referencia
ymvec=0;%Vector que contendra el historial de sallidas medidas
ypvec=ymvec;%vector que contendra el historial de predicciones para %cada instante
(k+1|k)
```

```
deltauvec=0;%vector que contendra el historial de incrementos deltau %para cada instante (k|k)
uvec=deltauvec;%vector que contendra el historial de la entrada para %cada instante (k|k)
zyu=0;%cond. iniciales para poder simular los sistemas mediante el %comando filter
L=ymvec*ones(N,1);%inicializacion de la resuesta libre
%=====================================================================
%                    Ejecución del algoritmo
%=====================================================================
for i=1:instantes;
   L=Q1*L+Su*deltauvec(end);%actualización de la rta libre
   w=(ymvec(end)-L(1))*ones(Np,1);%estimación de la perturbación no  %medible
   T=Q2*L+w;
   D=R(i:i+Np-1)-T;
   deltau=M*D;%cálculo de los movimientos futuros
   yp=T+G*deltau;%salida predicha
   ypvec(i+1)=yp(1);%salida predicha del instante Yp(k+1|k)
   deltauvec(i+1)=deltau(1);%deltau del instante deltau(k|k)
   uvec(i+1)=uvec(end)+deltau(1);%accion de control del instante %u(k|k)
%=====================================================================
%      aplicacion de los incrementos y medición de la salida

%=====================================================================
   [y,zyu]=filter(numsis(2:end),densis,uvec(end),zyu);%se aplica la %acción de control
   ymvec(i+1)=y;%se mide la salida del proceso
%=====================================================================
end
R=R(1:instantes+1);
deltauvec=deltauvec(2:end);
uvec=uvec(2:end);
%=====================================================================
%                    Gráficas de las variables
%=====================================================================
figure(1)
plot(t,ymvec,'b');
hold on
stairs(t,ypvec,'r');
stairs(t,R,'--g');
hold off
title('Salida')
xlabel('instantes')
legend('Yreal','Ypredicha','Ref')
figure(2)
subplot(211)
stairs(t(1:end-1),uvec,'b');
title('Acciones de control')
xlabel('instantes')
legend('u')
subplot(212)
stairs(t(1:end-1),deltauvec,'b');
title('Incrementos en las acciones de control')
xlabel('instantes')
legend('deltau')
```

QDMC RESTRICCIONES BLANDAS SISO

```
clear all; clc
%=====================================================================
%          Obtención de los coeficientes Su de la rta al escalón
```

```
%==========================================================================
Ts=4;%tiempo de muestreo
sis=c2d(tf([0 4.05],[50 1]),Ts,'zoh');%sistema
[numsis densis]=tfdata(sis,'v');
Su=step(sis);
Su=Su(2:end);%vector de coeficientes de respuesta al escalón
N=length(Su);%longitud del vector de coeficientes Su
%==========================================================================
%                          Parámetros
%==========================================================================
instantes=50;%número de periodos de muestreo que se ejecutará el %algoritmo DMC
Nc=5; %horizonte de control
Np=N+Nc;%horizonte de predicción
alfa=1;%parámetros alfa lambda y W
lambda=1;
rho=100000;
ymin=-0.51;%límites máximos y mínimos de las VC
ymax= 0.51;
umin=-0.1;%límites máximos y mínimos y de incrementos de las VM
umax= 0.1;
deltaumin=-0.01;
deltaumax= 0.01;
r=0.2;%referencia
%==========================================================================
%              Cálculo de las matrices G, Q1, Q2, alfa, lambda
%==========================================================================
if Np>N;
   Saux=[Su; ones(Np-N,1)*Su(end)];
else
   Saux=Su;
end
G=zeros(Np,Nc);
for i=1:Nc;
   G(i:end,i)=Saux(1:Np-i+1);
end
Q1=zeros(N,N);
Q1(1:end-1,2:end)=eye(N-1,N-1);
Q1(end,end)=1;
Q2=zeros(Np,N);
if Np==N;
   Q2=Q1;
elseif Np<N
   Q2(1:end,2:Np+1)=eye(Np,Np);
else
   Q2(1:N-1,2:end)=eye(N-1,N-1);
   Q2(N:end,end)=1;
end
alfa=alfa*eye(Np,Np);
lambda=lambda*eye(Nc,Nc);
%==========================================================================
%                    Matrices para el algoritmo QP
%==========================================================================
I1=zeros(1,Nc);
H=[[G'*alfa*G+lambda,I1'];[I1,rho]];
H=(H+H')/2;
Ia=zeros(Nc,1);
Ib(Np,1)=1;
I=eye(Nc,Nc);
```

```
IL=tril(ones(Nc,Nc));
A1=[[I; -I], [Ia; Ia]];
A2=[[IL; -IL], [Ia; Ia]];
A3=[[G; -G],[-Ib; -Ib]];
A=[A1; A2; A3; [Ia' -1]];
B1=[ones(Nc,1)*deltaumax; -ones(Nc,1)*deltaumin];
urest=[umax*ones(Nc,1); -umin*ones(Nc,1)];
yrest=[ymax*ones(Np,1); -ymin*ones(Np,1)];
Caux=-G'*alfa;
%=================================================================
%                  condiciones iniciales
%=================================================================
t=(0:1:instantes);%vector de tiempo total en instantes
R=r*ones(instantes+Np,1);%vector de trayectoria de referencia
ymvec=0;%Vector que contendra el historial de sallidas medidas
ypvec=ymvec;%vector que contendra el historial de predicciones para %cada instante
(k+1|k)
deltauvec=0;%vector que contendra el historial de incrementos deltau %para cada instante
(k|k)
uvec=deltauvec;%vector que contendra el historial de la entrada para %cada instante (k|k)
zyu=0;%cond. iniciales para poder simular los sistemas mediante el %comando filter
L=ymvec*ones(N,1);%inicializacion de la resuesta libre
%=================================================================
%                    Ejecución del algoritmo QDMC
%=================================================================
for i=1:instantes;
   L=Q1*L+Su*deltauvec(end);%actualización de la rta libre
   w=(ymvec(end)-L(1))*ones(Np,1);%estimación de la perturbación no %medible
   T=Q2*L+w;
   D=R(i:i+Np-1)-T;
   a=ones(Nc,1)*uvec(end);
   B2=urest+[-a;a];
   B3=yrest+[-T;T];
   B=[B1; B2; B3; 0];
   C=[Caux*D;0];
   x=QUADPROG(H,C,A,B);%cálculo del vector de incrementos en la %accion de control
   deltau=x(1:end-1);
   yp=T+G*deltau;%salida predicha
   ypvec(i+1)=yp(1);%salida predicha del instante Yp(k+1|k)
   deltauvec(i+1)=deltau(1);%deltau del instante deltau(k|k)
   uvec(i+1)=uvec(end)+deltau(1);%accion de control del instante %u(k|k)
%=================================================================
   %      aplicacion de los incrementos y medición de la salida
%=================================================================
   [y,zyu]=filter(numsis(2:end),densis,uvec(end),zyu);%se aplica la %acción de control
   ymvec(i+1)=y;%se mide la salida del proceso
%=================================================================
end
R=R(1:instantes+1);
deltauvec=deltauvec(2:end);
uvec=uvec(2:end);
%=================================================================
%                    Gráficas de las variables
%=================================================================
figure(1)
plot(t,ymvec,'b');
hold on
stairs(t,ypvec,'r');
```

```
stairs(t,R,'--g');
hold off
title('Salida')
xlabel('instantes')
legend('Yreal','Ypredicha','Ref')
figure(2)
subplot(211)
stairs(t(1:end-1),uvec,'b');
title('Acciones de control')
xlabel('instantes')
legend('u')
subplot(212)
stairs(t(1:end-1),deltauvec,'b');
title('Incrementos en las acciones de control')
xlabel('instantes')
legend('deltau')
```

DMC RLS SISO

```
function DMC_Iterativo_siso()
clear all; clc
%=====================================================================
%          Obtención de los coeficientes Su de la rta al escalón
%=====================================================================
Ts=4;%tiempo de muestreo
sis=c2d(tf([0 4.05],[50 1]),Ts,'zoh');%sistema
[numsis densis]=tfdata(sis,'v');
Su=step(sis);
Su=Su(2:end);%vector de coeficientes de respuesta al escalón
N=length(Su);%longitud del vector de coeficientes Su
%=====================================================================
%                          Parámetros
%=====================================================================
instantes=50;%número de periodos de muestreo que se ejecutará el %algoritmo DMC
Nc=5; %horizonte de control
Np=N+Nc;%horizonte de predicción
alfa=1;%parámetros alfa lambda y W
lambda=1;
ymin=-0.15;%límites máximos y mínimos de las VC
ymax= 0.15;
umin=-0.05;%límites máximos y mínimos y de incrementos de las VM
umax= 0.05;
deltaumin=-0.01;
deltaumax= 0.01;
r=-0.2;%referencia
%=====================================================================
%                 Parámetros del algoritmo recursivo
%=====================================================================
n=500;%número de veces que se ejecuta el algoritmo recursivo para %encontrar solución
tol1=0.01;%tolerancias de los límites de las entradas, incrementos y %salidas
tol2=0.009;
tol3=0.01;
w1=5;
w2=5;
w3=5;
%=====================================================================
%             Cálculo de las matrices G, Q1, Q2, alfa, lambda y M
%=====================================================================
if Np>N;
```

```
    Saux=[Su; ones(Np-N,1)*Su(end)];
else
    Saux=Su;
end
G=zeros(Np,Nc);
for i=1:Nc;
    G(i:end,i)=Saux(1:Np-i+1);
end
Q1=zeros(N,N);
Q1(1:end-1,2:end)=eye(N-1,N-1);
Q1(end,end)=1;
Q2=zeros(Np,N);
if Np==N;
    Q2=Q1;
elseif Np<N
    Q2(1:end,2:Np+1)=eye(Np,Np);
else
    Q2(1:N-1,2:end)=eye(N-1,N-1);
    Q2(N:end,end)=1;
end
alfa=alfa*eye(Np,Np);
lambda=lambda*eye(Nc,Nc);
M=inv(G'*alfa*G+lambda)*G'*alfa;
%=====================================================================
%                matrices del algoritmo recursivo
%=====================================================================
Pm0=inv(G'*alfa*G+lambda);
%=====================================================================
%                 condiciones iniciales
%=====================================================================
t=(0:1:instantes);%vector de tiempo total en instantes
R=r*ones(instantes+Np,1);%vector de trayectoria de referencia
ymvec=0;%Vector que contendra el historial de sallidas medidas
ypvec=ymvec;%vector que contendra el historial de predicciones para %cada instante
(k+1|k)
deltauvec=0;%vector que contendra el historial de incrementos deltau %para cada instante
(k|k)
uvec=deltauvec;%vector que contendra el historial de la entrada para %cada instante (k|k)
zyu=0;%cond. iniciales para poder simular los sistemas mediante el %comando filter
L=ymvec*ones(N,1);%inicializacion de la resuesta libre
%=====================================================================
%                   Ejecución del algoritmo
%=====================================================================
for i=1:instantes;
    L=Q1*L+Su*deltauvec(end);%actualización de la rta libre
    w=(ymvec(end)-L(1))*ones(Np,1);%estimación de la perturbación no %medible
    T=Q2*L+w;
    D=R(i:i+Np-1)-T;
    deltau=M*D;%cálculo de los movimientos futuros

%=====================================================================
    % aplicación del algoritmo recursivo
%=====================================================================
    cont=0;
    Pm=Pm0;
    while (cont<n)
        yp=T+G*deltau;%salida predicha
        ufut=uvec(end)+cumsum(deltau);
```

```
        flagu=verificar(ufut,umax,umin);
        flagdeltau=verificar(deltau,deltaumax,deltaumin);
        flagyp=verificar(yp,ymax,ymin);
        if  0==sum([flagu flagdeltau flagyp])
            break
        end
        cont=cont+1;
        E=0;
        h=zeros(1,Nc);
        if flagdeltau>0
            [E1,h1]=deltau_rest(flagdeltau,deltau,deltaumax,deltaumin,w1,tol1);
            E=[E;E1];
            h=[h;h1];
        end
        if flagu>0
            [E2,h2]=u_rest(flagu,ufut,uvec(end),umax,umin,w2,tol2);
            E=[E;E2];
            h=[h;h2];
        end
        if flagyp>0
            [E3,h3]=yp_rest(flagyp,yp,ymax,ymin,w3,tol3,G,T);
            E=[E;E3];
            h=[h;h3];
        end
        E=E(2:end);
        h=h(2:end,:)';
        m1=h'*Pm;
        m2=Pm*h*inv(m1*h+eye(size(h,2)));
        deltau=deltau+m2*(E-h'*deltau);
        Pm=Pm-m2*m1;
    end
    yp=T+G*deltau;%salida predicha
    ypvec(i+1)=yp(1);%salida predicha del instante Yp(k+1|k)
    deltauvec(i+1)=deltau(1);%deltau del instante deltau(k|k)
    uvec(i+1)=uvec(end)+deltau(1);%accion de control del instante %u(k|k)
%=========================================================================
    %       aplicacion de los incrementos y medición de la salida
%=========================================================================
    [y,zyu]=filter(numsis(2:end),densis,uvec(end),zyu);%se aplica la %acción de control
    ymvec(i+1)=y;%se mide la salida del proceso
%=========================================================================
%=========================================================================
end
R=R(1:instantes+1);
deltauvec=deltauvec(2:end);
uvec=uvec(2:end);
%=========================================================================
%                     Gráficas de las variables
%=========================================================================
figure(1)
plot(t,ymvec,'b');
hold on
stairs(t,ypvec,'r');
stairs(t,R,'--g');
hold off
title('Salida')
xlabel('instantes')
legend('Yreal','Ypredicha','Ref')
```

```
figure(2)
subplot(211)
stairs(t(1:end-1),uvec,'b');
title('Acciones de control')
xlabel('instantes')
legend('u')
subplot(212)
stairs(t(1:end-1),deltauvec,'b');
title('Incrementos en las acciones de control')
xlabel('instantes')
legend('deltau')
%================================================================
%verificar si los incrementos, entradas o predicciones violan %restricciones
%================================================================
function flag=verificar(vector,lim_max,lim_min)
flag=0;
if min(vector)<lim_min
  flag=flag+1;
end
if max(vector)>lim_max
  flag=flag+2;
end
%================================================================
% Cálculo de las matrices de restricciones h1 y E1 para los %incrementos
%================================================================
function  [E,h]=deltau_rest(flag,deltau,deltau_max,deltau_min,W,tol)
h=zeros(1,length(deltau));
E=0;
switch flag
   case 1
      for j=1:length(deltau);
         if  deltau(j)<deltau_min;
            h(end+1,j)=W;
            E(end+1,1)=(deltau_min+tol)*W;
         end
      end
   case 2
      for j=1:length(deltau);
         if  deltau(j)>deltau_max;
            h(end+1,j)=W;
            E(end+1,1)=(deltau_max-tol)*W;
         end
      end
   case 3
      for j=1:length(deltau);
         if  deltau(j)>deltau_max;
            h(end+1,j)=W;
            E(end+1,1)=(deltau_max-tol)*W;
         end
         if  deltau(j)<deltau_min;
            h(end+1,j)=W;
            E(end+1,1)=(deltau_min+tol)*W;
         end
      end
end
h=h(2:end,:);
E=E(2:end,1);
%================================================================
```

```
%     Cálculo de las matrices de restricciones h2 y E2 para las %entradas
%=================================================================
function  [E,h]=u_rest(flag,U_fut,U,U_max,U_min,W,tol)
a=length(U_fut);
switch flag
   case 1
      flag2=1;
      h=zeros(1,a);
      E=(U_min+tol-U)*W;
      for j=1:a;
         if U_fut(j)<U_min && flag2==1;
            h(1,1:j)=W;
            flag2=2;
         elseif flag2==2
            h(end+1,j)=W;
         end
      end
   case 2
      flag1=1;
      h=zeros(1,a);
      E=(U_max-tol-U)*W;
      for j=1:a;
         if U_fut(j)>U_max && flag1==1;
            h(1,1:j)=W;
            flag1=2;
         elseif flag1==2
            h(end+1,j)=W;
         end
      end
   case 3
      flag2=1;
      flag1=1;
      h=zeros(2,a);
      E=[U_max-tol-U;U_min+tol-U]*W;
      for j=1:a;
         if U_fut(j)>U_max && flag1==1;
            h(1,1:j)=W;
            flag1=2;
         elseif flag1==2
            h(end+1,j)=W;
         end
         if U_fut(j)<U_min && flag2==1;
            h(2,1:j)=W;
            flag2=2;
         elseif flag2==2 && h(end,j)==0
            h(end+1,j)=W;
         end
      end
end
E=[E;zeros(size(h,1)-length(E),1)];
%=================================================================
%Matrices de restricciones h3 y E3 para las predicciones en las %salidas
%=================================================================
function  [E,h]=yp_rest(flag,ypred,y_max,y_min,W,tol,G,L)
h=zeros(1,size(G,2));
E=0;
switch flag
   case 1
```

```
        for j=1:length(ypred);
            if ypred(j)<y_min;
                h(end+1,:)=G(j,:)*W;
                E(end+1,1)=(y_min+tol-L(j))*W;
            end
        end
    case 2
        for j=1:length(ypred);
            if ypred(j)>y_max;
                h(end+1,:)=G(j,:)*W;
                E(end+1,1)=(y_max-tol-L(j))*W;
            end
        end
    case 3
        for j=1:length(ypred);
            if ypred(j)>y_max;
                h(end+1,:)=G(j,:)*W;
                E(end+1,1)=(y_max-tol-L(j))*W;
            end
            if ypred(j)<y_min;
                h(end+1,:)=G(j,:)*W;
                E(end+1,1)=(y_min+tol-L(j))*W;
            end
        end
end
h=h(2:end,:);
E=E(2:end,1);
%=========================================================================
%=========================================================================
```

DMC MIMO

```
clear all; close all; clc
Ts=4;
del1=27;%retardos de los sistemas
del2=28;
del3=23;
del4=18;
del5=14;
del6=15;
del7=20;
del8=22;
sisy1u1=c2d(tf([0 4.05],[50 1],'inputdelay',del1),Ts,'zoh');%27
sisy1u2=c2d(tf([0 1.77],[60 1],'inputdelay',del2),Ts,'zoh');%28
sisy1u3=c2d(tf([0 2.88],[50 1],'inputdelay',del3),Ts,'zoh');%23
sisy2u1=c2d(tf([0 5.39],[50 1],'inputdelay',del4),Ts,'zoh');%18
sisy2u2=c2d(tf([0 5.72],[60 1],'inputdelay',del5),Ts,'zoh');%14
sisy2u3=c2d(tf([0 6.90],[40 1],'inputdelay',del6),Ts,'zoh');%15
sisy3u1=c2d(tf([0 4.38],[33 1],'inputdelay',del7),Ts,'zoh');%20
sisy3u2=c2d(tf([0 4.42],[44 1],'inputdelay',del8),Ts,'zoh');%22
sisy3u3=c2d(tf([0 7.20],[19 1]),Ts,'zoh');
[numy1u1 deny1u1]=tfdata(sisy1u1,'v');
[numy1u2 deny1u2]=tfdata(sisy1u2,'v');
[numy1u3 deny1u3]=tfdata(sisy1u3,'v');
[numy2u1 deny2u1]=tfdata(sisy2u1,'v');
[numy2u2 deny2u2]=tfdata(sisy2u2,'v');
[numy2u3 deny2u3]=tfdata(sisy2u3,'v');
[numy3u1 deny3u1]=tfdata(sisy3u1,'v');
[numy3u2 deny3u2]=tfdata(sisy3u2,'v');
```

```
[numy3u3 deny3u3]=tfdata(sisy3u3,'v');
Sy1u1=step(sisy1u1);
Sy1u2=step(sisy1u2);
Sy1u3=step(sisy1u3);
Sy2u1=step(sisy2u1);
Sy2u2=step(sisy2u2);
Sy2u3=step(sisy2u3);
Sy3u1=step(sisy3u1);
Sy3u2=step(sisy3u2);
Sy3u3=step(sisy3u3);
Sy1u1=Sy1u1(2:end);
Sy1u2=Sy1u2(2:end);
Sy1u3=Sy1u3(2:end);
Sy2u1=Sy2u1(2:end);
Sy2u2=Sy2u2(2:end);
Sy2u3=Sy2u3(2:end);
Sy3u1=Sy3u1(2:end);
Sy3u2=Sy3u2(2:end);
Sy3u3=Sy3u3(2:end);
Ny1=max([length(Sy1u1) length(Sy1u2) length(Sy1u3)]);
Ny2=max([length(Sy2u1) length(Sy2u2) length(Sy2u3)]);
Ny3=max([length(Sy3u1) length(Sy3u2) length(Sy3u3)]);
Sy1u1(end:Ny1,1)=Sy1u1(end);
Sy1u2(end:Ny1,1)=Sy1u2(end);
Sy1u3(end:Ny1,1)=Sy1u3(end);
Sy2u1(end:Ny2,1)=Sy2u1(end);
Sy2u2(end:Ny2,1)=Sy2u2(end);
Sy2u3(end:Ny2,1)=Sy2u3(end);
Sy3u1(end:Ny3,1)=Sy3u1(end);
Sy3u2(end:Ny3,1)=Sy3u2(end);
Sy3u3(end:Ny3,1)=Sy3u3(end);
Sy1=[Sy1u1 Sy1u2 Sy1u3];
Sy2=[Sy2u1 Sy2u2 Sy2u3];
Sy3=[Sy3u1 Sy3u2 Sy3u3];
Su=[Sy1; Sy2; Sy3];
%=====================================================================
%                        Parámetros
%=====================================================================
instantes=100;%número de veces que se ejecutará el algoritmo DMC
Ncu1=5; %horizontes de control
Ncu2=6;
Ncu3=7;
Npy1=Ny1+max([Ncu1 Ncu2 Ncu3]);%horizontes de predicción
Npy2=Ny2+max([Ncu1 Ncu2 Ncu3]);
Npy3=Ny3+max([Ncu1 Ncu2 Ncu3]);
alfay1=3;%parámetros alfa lambda y W
alfay2=5;
alfay3=15;
lambdau1=1;
lambdau2=1;
lambdau3=1;
ry1=0.49;%referencias
ry2=0.3;
ry3=0.1399;
%=====================================================================
%                   Cálculo de la matriz G
%=====================================================================
Nc=[Ncu1 Ncu2 Ncu3];
```

```
G=zeros(1,sum(Nc));
for i=1:3
   switch i;
      case 1
         Np=Npy1;
         N=Ny1;
         S=Sy1;
      case 2
         Np=Npy2;
         N=Ny2;
         S=Sy2;
      case 3
         Np=Npy3;
         N=Ny3;
         S=Sy3;
   end
   if Np<=N;
      Saux=S(1:Np,:);
   else
      Saux=[S; ones(Np-N,3)*[S(end,1) 0 0; 0 S(end,2) 0; 0 0 S(end,3)]];
   end
   Gaux=zeros(Np,sum(Nc));
   k=0;
   for j=1:3
      for i=1:Nc(j);
         Gaux(i:end,k+i)=Saux(1:Np-i+1,j);
      end
      k=sum(Nc(1:j));
   end
   G=[G;Gaux];
end
G=G(2:end,:);
%=====================================================================
%              Cálculo de las matrices alfa,lambda y M
%=====================================================================
alfa=blkdiag(alfay1*eye(Npy1,Npy1),alfay2*eye(Npy2,Npy2),alfay3*eye(Npy3,Npy3));
lambda=blkdiag(lambdau1*eye(Ncu1,Ncu1),lambdau2*eye(Ncu2,Ncu2),lambdau3*eye(Ncu3
,Ncu3));
M=inv(G'*alfa*G+lambda)*G'*alfa;
%=====================================================================
%              Cálculo de las matrices Q1 y Q2
%=====================================================================
Q1y1=zeros(Ny1,Ny1);
Q1y1(1:end-1,2:end)=eye(Ny1-1,Ny1-1);
Q1y1(end,end)=1;
Q1y2=zeros(Ny2,Ny2);
Q1y2(1:end-1,2:end)=eye(Ny2-1,Ny2-1);
Q1y2(end,end)=1;
Q1y3=zeros(Ny3,Ny3);
Q1y3(1:end-1,2:end)=eye(Ny3-1,Ny3-1);
Q1y3(end,end)=1;
Q1=blkdiag(Q1y1,Q1y2,Q1y3);
Q2=0;
for i=1:3;
   switch i
      case 1
         Np=Npy1;
         N=Ny1;
```

```
            Q2aux1=Q1y1;
        case 2
            Np=Npy2;
            N=Ny2;
            Q2aux1=Q1y2;
        case 3
            Np=Npy3;
            N=Ny3;
            Q2aux1=Q1y3;
    end
    Q2aux2=zeros(Np,N);
    if Np==N;
        Q2aux2=Q2aux1;
    elseif Np<N
        Q2aux2(1:end,2:Np+1)=eye(Np,Np);
    else
        Q2aux2(1:N-1,2:end)=eye(N-1,N-1);
        Q2aux2(N:end,end)=1;
    end
    Q2=blkdiag(Q2,Q2aux2);
end
Q2=Q2(2:end,2:end);
%=====================================================================
%                       condiciones iniciales
%=====================================================================
t=(0:1:instantes);%vector de tiempo total
deltau1vec=0;
deltau2vec=0;
deltau3vec=0;
u1vec=0;%vector que contendra las entadas aplicadas al sistema
u2vec=0;
u3vec=0;
ym1vec=0;%salida en el instante actual
ym2vec=0;
ym3vec=0;
yp1vec=0;%vector que contendra las salidas predicha de cada instante
yp2vec=0;
yp3vec=0;
zy1u1=0;%cond. iniciales para poder simular los sistemas mediante el comando filter
zy1u2=0;
zy1u3=0;
zy2u1=0;
zy2u2=0;
zy2u3=0;
zy3u1=0;
zy3u2=0;
zy3u3=0;
rety1u1=zeros(1,round(del1/Ts));%vectores para tener en cuenta los retardos de los
%sistemas para poder simular los sistemas mediante el comando filter
rety1u2=zeros(1,round(del2/Ts));
rety1u3=zeros(1,round(del3/Ts));
rety2u1=zeros(1,round(del4/Ts));
rety2u2=zeros(1,round(del5/Ts));
rety2u3=zeros(1,round(del6/Ts));
rety3u1=zeros(1,round(del7/Ts));
rety3u2=zeros(1,round(del8/Ts));
Ry1=ry1*ones(instantes+Npy1,1);%vectores de referencias
Ry2=ry2*ones(instantes+Npy2,1);
```

```
Ry3=ry3*ones(instantes+Npy3,1);
L=[ym1vec*ones(Ny1,1);ym2vec*ones(Ny2,1);ym3vec*ones(Ny3,1)];%inicializacion de la
%resuesta libre
%=========================================================================
%                    Ejecución del algoritmo DMC
%=========================================================================
for i=1:instantes;
   L=Q1*L+Su*[deltau1vec(end);deltau2vec(end);deltau3vec(end)];%actualización de la rta
%libre
   w=[(ym1vec(end)-L(1))*ones(Npy1,1);...
     (ym2vec(end)-L(Ny1+1))*ones(Npy2,1);...
     (ym3vec(end)-L(Ny1+Ny2+1))*ones(Npy3,1)];%estimación de la perturbación no
%medible
   T=Q2*L+w;
   R=[Ry1(i:i+Npy1-1,1); Ry2(i:i+Npy2-1,1); Ry3(i:i+Npy3-1,1)];
   D=R-T;
   deltau=M*D;
%=========================================================================
%=========================================================================
   yp=T+G*deltau;%salida predicha
   yp1vec(i+1)=yp(1);%salida predicha del instante Yp(k+1|k)
   yp2vec(i+1)=yp(Npy1+1);%salida predicha del instante Yp(k+1|k)
   yp3vec(i+1)=yp(Npy1+Npy2+1);%salida predicha del instante Yp(k+1|k)
   deltau1vec(i+1)=deltau(1);%deltau del instante deltau(k|k)
   deltau2vec(i+1)=deltau(Ncu1+1);%deltau del instante deltau(k|k)
   deltau3vec(i+1)=deltau(Ncu1+Ncu2+1);%deltau del instante deltau(k|k)
   u1vec(i+1)=u1vec(end)+deltau(1);%accion de control del instante u(k|k)
   u2vec(i+1)=u2vec(end)+deltau(Ncu1+1);%accion de control del instante u(k|k)
   u3vec(i+1)=u3vec(end)+deltau(Ncu1+Ncu2+1);%accion de control del instante u(k|k)
%=========================================================================
   %      aplicacion de los incrementos y medición de la salida
%=========================================================================
   [y1_,zy1u1]=filter(numy1u1(1:end),deny1u1,u1vec(end),zy1u1);
   [y2_,zy1u2]=filter(numy1u2(1:end),deny1u2,u2vec(end),zy1u2);
   [y3_,zy1u3]=filter(numy1u3(1:end),deny1u3,u3vec(end),zy1u3);
   rety1u1=[y1_ rety1u1(1:end-1)];
   rety1u2=[y2_ rety1u2(1:end-1)];
   rety1u3=[y3_ rety1u3(1:end-1)];
   ym1vec(i+1)=rety1u1(end)+rety1u2(end)+rety1u3(end);
   [y1_,zy2u1]=filter(numy2u1(1:end),deny2u1,u1vec(end),zy2u1);
   [y2_,zy2u2]=filter(numy2u2(1:end),deny2u2,u2vec(end),zy2u2);
   [y3_,zy2u3]=filter(numy2u3(1:end),deny2u3,u3vec(end),zy2u3);
   rety2u1=[y1_ rety2u1(1:end-1)];
   rety2u2=[y2_ rety2u2(1:end-1)];
   rety2u3=[y3_ rety2u3(1:end-1)];
   ym2vec(i+1)=rety2u1(end)+rety2u2(end)+rety2u3(end);
   [y1_,zy3u1]=filter(numy3u1(1:end),deny3u1,u1vec(end),zy3u1);
   [y2_,zy3u2]=filter(numy3u2(1:end),deny3u2,u2vec(end),zy3u2);
   [y3_,zy3u3]=filter(numy3u3(2:end),deny3u3,u3vec(end),zy3u3);
   rety3u1=[y1_ rety3u1(1:end-1)];
   rety3u2=[y2_ rety3u2(1:end-1)];
   rety3u3=y3_;
   ym3vec(i+1)=rety3u1(end)+rety3u2(end)+rety3u3(end);
end
Ry1=Ry1(1:instantes+1);
Ry2=Ry2(1:instantes+1);
Ry3=Ry3(1:instantes+1);
deltau1vec=deltau1vec(2:end);
```

```
deltau2vec=deltau2vec(2:end);
deltau3vec=deltau3vec(2:end);
u1vec=u1vec(2:end);
u2vec=u2vec(2:end);
u3vec=u3vec(2:end);
%=====================================================================
%                    Gráficas de las variables
%=====================================================================
figure()
plot(t,ym1vec,'b',t,ym2vec,'r',t,ym3vec,'g');
hold on
stairs(t,yp1vec,'b');
stairs(t,yp2vec,'r');
stairs(t,yp3vec,'g');
stairs(t,Ry1,'--b');
stairs(t,Ry2,'--r');
stairs(t,Ry3,'--g');
hold off
title('Salidas')
xlabel('Instantes')
legend('Y1','Y2','Y3')
figure()
t=t(1:end-1);
subplot(211)
stairs(t,u1vec,'b');
hold on
stairs(t,u2vec,'r');
stairs(t,u3vec,'g');
hold off
title('Acciones de control')
xlabel('Instantes')
legend('u1','u2','u3')
subplot(212)
stairs(t,deltau1vec,'b');
hold on
stairs(t,deltau2vec,'r');
stairs(t,deltau3vec,'g');
hold off
title('Incrementos en las acciones de control')
xlabel('Instantes')
legend('deltau1','deltau2','deltau3')
```

QDMC RESTRICCIONES BLANDAS MIMO

```
clear all; close all; clc
Ts=4;
del1=27;%retardos de los sistemas
del2=28;
del3=23;
del4=18;
del5=14;
del6=15;
del7=20;
del8=22;
sisy1u1=c2d(tf([0 4.05],[50 1],'inputdelay',del1),Ts,'zoh');%27
sisy1u2=c2d(tf([0 1.77],[60 1],'inputdelay',del2),Ts,'zoh');%28
sisy1u3=c2d(tf([0 2.88],[50 1],'inputdelay',del3),Ts,'zoh');%23
sisy2u1=c2d(tf([0 5.39],[50 1],'inputdelay',del4),Ts,'zoh');%18
sisy2u2=c2d(tf([0 5.72],[60 1],'inputdelay',del5),Ts,'zoh');%14
```

```
sisy2u3=c2d(tf([0 6.90],[40 1],'inputdelay',del6),Ts,'zoh');%15
sisy3u1=c2d(tf([0 4.38],[33 1],'inputdelay',del7),Ts,'zoh');%20
sisy3u2=c2d(tf([0 4.42],[44 1],'inputdelay',del8),Ts,'zoh');%22
sisy3u3=c2d(tf([0 7.20],[19 1]),Ts,'zoh');
[numy1u1 deny1u1]=tfdata(sisy1u1,'v');
[numy1u2 deny1u2]=tfdata(sisy1u2,'v');
[numy1u3 deny1u3]=tfdata(sisy1u3,'v');
[numy2u1 deny2u1]=tfdata(sisy2u1,'v');
[numy2u2 deny2u2]=tfdata(sisy2u2,'v');
[numy2u3 deny2u3]=tfdata(sisy2u3,'v');
[numy3u1 deny3u1]=tfdata(sisy3u1,'v');
[numy3u2 deny3u2]=tfdata(sisy3u2,'v');
[numy3u3 deny3u3]=tfdata(sisy3u3,'v');
Sy1u1=step(sisy1u1);
Sy1u2=step(sisy1u2);
Sy1u3=step(sisy1u3);
Sy2u1=step(sisy2u1);
Sy2u2=step(sisy2u2);
Sy2u3=step(sisy2u3);
Sy3u1=step(sisy3u1);
Sy3u2=step(sisy3u2);
Sy3u3=step(sisy3u3);
Sy1u1=Sy1u1(2:end);
Sy1u2=Sy1u2(2:end);
Sy1u3=Sy1u3(2:end);
Sy2u1=Sy2u1(2:end);
Sy2u2=Sy2u2(2:end);
Sy2u3=Sy2u3(2:end);
Sy3u1=Sy3u1(2:end);
Sy3u2=Sy3u2(2:end);
Sy3u3=Sy3u3(2:end);
Ny1=max([length(Sy1u1) length(Sy1u2) length(Sy1u3)]);
Ny2=max([length(Sy2u1) length(Sy2u2) length(Sy2u3)]);
Ny3=max([length(Sy3u1) length(Sy3u2) length(Sy3u3)]);
Sy1u1(end:Ny1,1)=Sy1u1(end);
Sy1u2(end:Ny1,1)=Sy1u2(end);
Sy1u3(end:Ny1,1)=Sy1u3(end);
Sy2u1(end:Ny2,1)=Sy2u1(end);
Sy2u2(end:Ny2,1)=Sy2u2(end);
Sy2u3(end:Ny2,1)=Sy2u3(end);
Sy3u1(end:Ny3,1)=Sy3u1(end);
Sy3u2(end:Ny3,1)=Sy3u2(end);
Sy3u3(end:Ny3,1)=Sy3u3(end);
Sy1=[Sy1u1 Sy1u2 Sy1u3];
Sy2=[Sy2u1 Sy2u2 Sy2u3];
Sy3=[Sy3u1 Sy3u2 Sy3u3];
Su=[Sy1; Sy2; Sy3];
%=====================================================================
%                       Parámetros
%=====================================================================
instantes=100;%número de veces que se ejecutará el algoritmo DMC
Ncu1=5; %horizontes de control
Ncu2=6;
Ncu3=7;
Npy1=Ny1+max([Ncu1 Ncu2 Ncu3]);%horizontes de predicción
Npy2=Ny2+max([Ncu1 Ncu2 Ncu3]);
Npy3=Ny3+max([Ncu1 Ncu2 Ncu3]);
alfay1=3;%parámetros alfa lambda y W
```

```
alfay2=5;
alfay3=15;
lambdau1=1;
lambdau2=1;
lambdau3=1;
y1min=-0.51;%límites máximos y mínimos de las VC
y1max= 0.51;
y2min=-0.31;
y2max= 0.31;
y3min=-0.15;
y3max= 0.15;
u1min=-0.2;%límites máximos y mínimos y de incrementos de las VM
u1max= 0.2;
u2min=-0.2;
u2max= 0.2;
u3min=-0.2;
u3max= 0.2;
deltau1min=-0.02;
deltau1max= 0.02;
deltau2min=-0.02;
deltau2max= 0.02;
deltau3min=-0.02;
deltau3max= 0.02;
ry1=0.49;%referencias
ry2=0.3;
ry3=0.1399;
%=====================================================================
%                  Parámetros del algoritmo QP
%=====================================================================
rhoy1=100000;
rhoy2=100000;
rhoy3=100000;
%=====================================================================
%                   Cálculo de la matriz G
%=====================================================================
Nc=[Ncu1 Ncu2 Ncu3];
G=zeros(1,sum(Nc));
for i=1:3
   switch i;
      case 1
         Np=Npy1;
         N=Ny1;
         S=Sy1;
      case 2
         Np=Npy2;
         N=Ny2;
         S=Sy2;
      case 3
         Np=Npy3;
         N=Ny3;
         S=Sy3;
   end
   if Np<=N;
      Saux=S(1:Np,:);
   else
      Saux=[S; ones(Np-N,3)*[S(end,1) 0 0; 0 S(end,2) 0; 0 0 S(end,3)]];
   end
   Gaux=zeros(Np,sum(Nc));
```

```
    k=0;
    for j=1:3
        for i=1:Nc(j);
            Gaux(i:end,k+i)=Saux(1:Np-i+1,j);
        end
        k=sum(Nc(1:j));
    end
    G=[G;Gaux];
end
G=G(2:end,:);
%=====================================================================
%              Cálculo de las matrices alfa y lambda
%=====================================================================
alfa=blkdiag(alfay1*eye(Npy1,Npy1),alfay2*eye(Npy2,Npy2),alfay3*eye(Npy3,Npy3));
lambda=blkdiag(lambdau1*eye(Ncu1,Ncu1),lambdau2*eye(Ncu2,Ncu2),lambdau3*eye(Ncu3
,Ncu3));
%=====================================================================
%              Cálculo de las matrices Q1 y Q2
%=====================================================================
Q1y1=zeros(Ny1,Ny1);
Q1y1(1:end-1,2:end)=eye(Ny1-1,Ny1-1);
Q1y1(end,end)=1;
Q1y2=zeros(Ny2,Ny2);
Q1y2(1:end-1,2:end)=eye(Ny2-1,Ny2-1);
Q1y2(end,end)=1;
Q1y3=zeros(Ny3,Ny3);
Q1y3(1:end-1,2:end)=eye(Ny3-1,Ny3-1);
Q1y3(end,end)=1;
Q1=blkdiag(Q1y1,Q1y2,Q1y3);
Q2=0;
for i=1:3;
    switch i
        case 1
            Np=Npy1;
            N=Ny1;
            Q2aux1=Q1y1;
        case 2
            Np=Npy2;
            N=Ny2;
            Q2aux1=Q1y2;
        case 3
            Np=Npy3;
            N=Ny3;
            Q2aux1=Q1y3;
    end
    Q2aux2=zeros(Np,N);
    if Np==N;
        Q2aux2=Q2aux1;
    elseif Np<N
        Q2aux2(1:end,2:Np+1)=eye(Np,Np);
    else
        Q2aux2(1:N-1,2:end)=eye(N-1,N-1);
        Q2aux2(N:end,end)=1;
    end
    Q2=blkdiag(Q2,Q2aux2);
end
Q2=Q2(2:end,2:end);
%=====================================================================
```

```
%                   Matrices para el algoritmo QP
%=================================================================
I1=zeros(3,Ncu1+Ncu2+Ncu3);
H=[[G'*alfa*G+lambda,I1'];[I1,[rhoy1 0 0;0 rhoy2 0;0 0 rhoy3]]];
H=(H+H')/2;
Ia=zeros(Ncu1+Ncu2+Ncu3,3);
Ib=blkdiag(-ones(Npy1,1),-ones(Npy2,1),-ones(Npy3,1));
I=eye(Ncu1+Ncu2+Ncu3);
IL=tril(ones(Ncu1+Ncu2+Ncu3));
A1=[[I; -I],[Ia; Ia]];
A2=[[IL; -IL], [Ia;Ia]];
A3=[[G; -G],[Ib;Ib]];
A=[A1; A2; A3; [Ia',-eye(3,3)]];
B1=[ones(Ncu1,1)*deltau1max;ones(Ncu2,1)*deltau2max;ones(Ncu3,1)*deltau3max;...
  -[ones(Ncu1,1)*deltau1min;ones(Ncu2,1)*deltau2min;ones(Ncu3,1)*deltau3min]];
urest=[u1max*ones(Ncu1,1);  u2max*ones(Ncu2,1);  u3max*ones(Ncu3,1);...
     -u1min*ones(Ncu1,1); -u2min*ones(Ncu2,1); -u3min*ones(Ncu3,1)];
yrest=[y1max*ones(Npy1,1);  y2max*ones(Npy2,1);  y3max*ones(Npy3,1);...
     -y1min*ones(Npy1,1); -y2min*ones(Npy2,1); -y3min*ones(Npy3,1)];
Caux=-G'*alfa;
%=================================================================
%                       condiciones iniciales
%=================================================================
t=(0:1:instantes);%vector de tiempo total
deltau1vec=0;
deltau2vec=0;
deltau3vec=0;
u1vec=0;%vector que contendra las entadas aplicadas al sistema
u2vec=0;
u3vec=0;
ym1vec=0;%salida en el instante actual
ym2vec=0;
ym3vec=0;
yp1vec=0;%vector que contendra las salidas predicha de cada instante
yp2vec=0;
yp3vec=0;
zy1u1=0;%cond. iniciales para poder simular los sistemas mediante el comando filter
zy1u2=0;
zy1u3=0;
zy2u1=0;
zy2u2=0;
zy2u3=0;
zy3u1=0;
zy3u2=0;
zy3u3=0;
rety1u1=zeros(1,round(del1/Ts));%vectores para tener en cuenta los retardos de los
%sistemas para poder simular los sistemas mediante el comando filter
rety1u2=zeros(1,round(del2/Ts));
rety1u3=zeros(1,round(del3/Ts));
rety2u1=zeros(1,round(del4/Ts));
rety2u2=zeros(1,round(del5/Ts));
rety2u3=zeros(1,round(del6/Ts));
rety3u1=zeros(1,round(del7/Ts));
rety3u2=zeros(1,round(del8/Ts));
Ry1=ry1*ones(instantes+Npy1,1);%vectores de referencias
Ry2=ry2*ones(instantes+Npy2,1);
Ry3=ry3*ones(instantes+Npy3,1);
```

```
L=[ym1vec*ones(Ny1,1);ym2vec*ones(Ny2,1);ym3vec*ones(Ny3,1)];%inicializacion de la
%resuesta libre
%=========================================================================
%                    Ejecución del algoritmo QDMC
%=========================================================================
for i=1:instantes;
   L=Q1*L+Su*[deltau1vec(end);deltau2vec(end);deltau3vec(end)];%actualización de la rta
%libre
   w=[(ym1vec(end)-L(1))*ones(Npy1,1);...
     (ym2vec(end)-L(Ny1+1))*ones(Npy2,1);...
     (ym3vec(end)-L(Ny1+Ny2+1))*ones(Npy3,1)];%estimación de la perturbación no
%medible
   T=Q2*L+w;
   R=[Ry1(i:i+Npy1-1,1); Ry2(i:i+Npy2-1,1); Ry3(i:i+Npy3-1,1)];
   D=R-T;
   a=[u1vec(end)*ones(Ncu1,1);  u2vec(end)*ones(Ncu2,1);  u3vec(end)*ones(Ncu3,1)];
   B2=urest+[-a;a];
   B3=yrest+[-T;T];
   B=[B1;B2;B3;0;0;0];
   C=[Caux*D;0;0;0];
   x=QUADPROG(H,C,A,B);%cálculo del vector de incrementos en la accion de control
   deltau=x(1:end-3);
%=========================================================================
   yp=T+G*deltau;%salida predicha
   yp1vec(i+1)=yp(1);%salida predicha del instante Yp(k+1|k)
   yp2vec(i+1)=yp(Npy1+1);%salida predicha del instante Yp(k+1|k)
   yp3vec(i+1)=yp(Npy1+Npy2+1);%salida predicha del instante Yp(k+1|k)
   deltau1vec(i+1)=deltau(1);%deltau del instante deltau(k|k)
   deltau2vec(i+1)=deltau(Ncu1+1);%deltau del instante deltau(k|k)
   deltau3vec(i+1)=deltau(Ncu1+Ncu2+1);%deltau del instante deltau(k|k)
   u1vec(i+1)=u1vec(end)+deltau(1);%accion de control del instante u(k|k)
   u2vec(i+1)=u2vec(end)+deltau(Ncu1+1);%accion de control del instante u(k|k)
   u3vec(i+1)=u3vec(end)+deltau(Ncu1+Ncu2+1);%accion de control del instante u(k|k)
%=========================================================================
   %      aplicacion de los incrementos y medición de la salida
%=========================================================================
   [y1_,zy1u1]=filter(numy1u1(1:end),deny1u1,u1vec(end),zy1u1);
   [y2_,zy1u2]=filter(numy1u2(1:end),deny1u2,u2vec(end),zy1u2);
   [y3_,zy1u3]=filter(numy1u3(1:end),deny1u3,u3vec(end),zy1u3);
   rety1u1=[y1_ rety1u1(1:end-1)];
   rety1u2=[y2_ rety1u2(1:end-1)];
   rety1u3=[y3_ rety1u3(1:end-1)];
   ym1vec(i+1)=rety1u1(end)+rety1u2(end)+rety1u3(end);
   [y1_,zy2u1]=filter(numy2u1(1:end),deny2u1,u1vec(end),zy2u1);
   [y2_,zy2u2]=filter(numy2u2(1:end),deny2u2,u2vec(end),zy2u2);
   [y3_,zy2u3]=filter(numy2u3(1:end),deny2u3,u3vec(end),zy2u3);
   rety2u1=[y1_ rety2u1(1:end-1)];
   rety2u2=[y2_ rety2u2(1:end-1)];
   rety2u3=[y3_ rety2u3(1:end-1)];
   ym2vec(i+1)=rety2u1(end)+rety2u2(end)+rety2u3(end);
   [y1_,zy3u1]=filter(numy3u1(1:end),deny3u1,u1vec(end),zy3u1);
   [y2_,zy3u2]=filter(numy3u2(1:end),deny3u2,u2vec(end),zy3u2);
   [y3_,zy3u3]=filter(numy3u3(2:end),deny3u3,u3vec(end),zy3u3);
   rety3u1=[y1_ rety3u1(1:end-1)];
   rety3u2=[y2_ rety3u2(1:end-1)];
   rety3u3=y3_;
   ym3vec(i+1)=rety3u1(end)+rety3u2(end)+rety3u3(end);
end
```

```
Ry1=Ry1(1:instantes+1);
Ry2=Ry2(1:instantes+1);
Ry3=Ry3(1:instantes+1);
deltau1vec=deltau1vec(2:end);
deltau2vec=deltau2vec(2:end);
deltau3vec=deltau3vec(2:end);
u1vec=u1vec(2:end);
u2vec=u2vec(2:end);
u3vec=u3vec(2:end);
%=====================================================================
%                    Gráficas de las variables
%=====================================================================
t2=[t(1) t(end)];
figure()
plot(t,ym1vec,'b',t,ym2vec,'r',t,ym3vec,'g');
hold on
stairs(t,yp1vec,'b');
stairs(t,yp2vec,'r');
stairs(t,yp3vec,'g');
stairs(t,Ry1,'--b');
stairs(t,Ry2,'--r');
stairs(t,Ry3,'--g');
%plot(t2,[1 1]*y1max,':b', t2, [1 1]*y2max,':r',t2,[1 1]*y3max,':g');
%plot(t2,[1 1]*y1min,':b', t2, [1 1]*y2min,':r',t2,[1 1]*y3min,':g');
hold off
title('Salidas')
xlabel('Instantes')
legend('Y1','Y2','Y3')
figure()
t=t(1:end-1);
subplot(211)
stairs(t,u1vec,'b');
hold on
stairs(t,u2vec,'r');
stairs(t,u3vec,'g');
%plot(t2,[1 1]*u1max,':b', t2, [1 1]*u2max,':r',t2,[1 1]*u3max,':g');
%plot(t2,[1 1]*u1min,':b', t2, [1 1]*u2min,':r',t2,[1 1]*u3min,':g');
hold off
title('Acciones de control')
xlabel('Instantes')
legend('u1','u2','u3')
subplot(212)
stairs(t,deltau1vec,'b');
hold on
stairs(t,deltau2vec,'r');
stairs(t,deltau3vec,'g');
%plot(t2,[1 1]*deltau1max,':b', t2, [1 1]*deltau2max,':r',t2,[1 1]*deltau3max,':g');
%plot(t2,[1 1]*deltau1min,':b', t2, [1 1]*deltau2min,':r',t2,[1 1]*deltau3min,':g');
hold off
title('Incrementos en las acciones de control')
xlabel('Instantes')
legend('deltau1','deltau2','deltau3')
```

DMC RLS MIMO

```
function PATENTE_mimo()
clear all; close all; clc
Ts=4;
del1=27;%retardos de los sistemas
```

```
del2=28;
del3=23;
del4=18;
del5=14;
del6=15;
del7=20;
del8=22;
sisy1u1=c2d(tf([0 4.05],[50 1],'inputdelay',del1),Ts,'zoh');%27
sisy1u2=c2d(tf([0 1.77],[60 1],'inputdelay',del2),Ts,'zoh');%28
sisy1u3=c2d(tf([0 2.88],[50 1],'inputdelay',del3),Ts,'zoh');%23
sisy2u1=c2d(tf([0 5.39],[50 1],'inputdelay',del4),Ts,'zoh');%18
sisy2u2=c2d(tf([0 5.72],[60 1],'inputdelay',del5),Ts,'zoh');%14
sisy2u3=c2d(tf([0 6.90],[40 1],'inputdelay',del6),Ts,'zoh');%15
sisy3u1=c2d(tf([0 4.38],[33 1],'inputdelay',del7),Ts,'zoh');%20
sisy3u2=c2d(tf([0 4.42],[44 1],'inputdelay',del8),Ts,'zoh');%22
sisy3u3=c2d(tf([0 7.20],[19 1]),Ts,'zoh');
[numy1u1 deny1u1]=tfdata(sisy1u1,'v');
[numy1u2 deny1u2]=tfdata(sisy1u2,'v');
[numy1u3 deny1u3]=tfdata(sisy1u3,'v');
[numy2u1 deny2u1]=tfdata(sisy2u1,'v');
[numy2u2 deny2u2]=tfdata(sisy2u2,'v');
[numy2u3 deny2u3]=tfdata(sisy2u3,'v');
[numy3u1 deny3u1]=tfdata(sisy3u1,'v');
[numy3u2 deny3u2]=tfdata(sisy3u2,'v');
[numy3u3 deny3u3]=tfdata(sisy3u3,'v');
Sy1u1=step(sisy1u1);
Sy1u2=step(sisy1u2);
Sy1u3=step(sisy1u3);
Sy2u1=step(sisy2u1);
Sy2u2=step(sisy2u2);
Sy2u3=step(sisy2u3);
Sy3u1=step(sisy3u1);
Sy3u2=step(sisy3u2);
Sy3u3=step(sisy3u3);
Sy1u1=Sy1u1(2:end);
Sy1u2=Sy1u2(2:end);
Sy1u3=Sy1u3(2:end);
Sy2u1=Sy2u1(2:end);
Sy2u2=Sy2u2(2:end);
Sy2u3=Sy2u3(2:end);
Sy3u1=Sy3u1(2:end);
Sy3u2=Sy3u2(2:end);
Sy3u3=Sy3u3(2:end);
Ny1=max([length(Sy1u1) length(Sy1u2) length(Sy1u3)]);
Ny2=max([length(Sy2u1) length(Sy2u2) length(Sy2u3)]);
Ny3=max([length(Sy3u1) length(Sy3u2) length(Sy3u3)]);
Sy1u1(end:Ny1,1)=Sy1u1(end);
Sy1u2(end:Ny1,1)=Sy1u2(end);
Sy1u3(end:Ny1,1)=Sy1u3(end);
Sy2u1(end:Ny2,1)=Sy2u1(end);
Sy2u2(end:Ny2,1)=Sy2u2(end);
Sy2u3(end:Ny2,1)=Sy2u3(end);
Sy3u1(end:Ny3,1)=Sy3u1(end);
Sy3u2(end:Ny3,1)=Sy3u2(end);
Sy3u3(end:Ny3,1)=Sy3u3(end);
Sy1=[Sy1u1 Sy1u2 Sy1u3];
Sy2=[Sy2u1 Sy2u2 Sy2u3];
Sy3=[Sy3u1 Sy3u2 Sy3u3];
```

```
Su=[Sy1; Sy2; Sy3];
%=================================================================
%                        Parámetros
%=================================================================
instantes=50;%número de veces que se ejecutará el algoritmo DMC
instantes=100;%número de veces que se ejecutará el algoritmo DMC
Ncu1=5; %horizontes de control
Ncu2=6;
Ncu3=7;
Npy1=Ny1+max([Ncu1 Ncu2 Ncu3]);%horizontes de predicción
Npy2=Ny2+max([Ncu1 Ncu2 Ncu3]);
Npy3=Ny3+max([Ncu1 Ncu2 Ncu3]);
alfay1=1;%parámetros alfa lambda y W
alfay2=1;
alfay3=1;
lambdau1=1;
lambdau2=1;
lambdau3=1;
y1min=-0.51;%límites máximos y mínimos de las VC
y1max= 0.51;
y2min=-0.31;
y2max= 0.31;
y3min=-0.15;
y3max= 0.15;
u1min=-0.15;%límites máximos y mínimos y de incrementos de las VM
u1max= 0.15;
u2min=-0.15;
u2max= 0.15;
u3min=-0.15;
u3max= 0.15;
deltau1min=-0.015;
deltau1max= 0.015;
deltau2min=-0.015;
deltau2max= 0.015;
deltau3min=-0.015;
deltau3max= 0.015;
ry1=0.29;%referencias
ry2=0.2;
ry3=0.1399;
%=================================================================
%            Parámetros del algoritmo recursivo
%=================================================================
nn=100;%número de veces que se ejecuta el algoritmo recursivo para encontrar solución
w1=50;
w2=50;
w3=10;
tol1=0.0015;%tolerancias de los límites de las entradas, incrementos y salidas
tol2=0.015;
tol3=0.03;
%=================================================================
%                  Cálculo de la matriz G
%=================================================================
Nc=[Ncu1 Ncu2 Ncu3];
G=zeros(1,sum(Nc));
for i=1:3
   switch i;
      case 1
         Np=Npy1;
```

```
            N=Ny1;
            S=Sy1;
        case 2
            Np=Npy2;
            N=Ny2;
            S=Sy2;
        case 3
            Np=Npy3;
            N=Ny3;
            S=Sy3;
    end
    if Np<=N;
        Saux=S(1:Np,:);
    else
        Saux=[S; ones(Np-N,3)*[S(end,1) 0 0; 0 S(end,2) 0; 0 0 S(end,3)]];
    end
    Gaux=zeros(Np,sum(Nc));
    k=0;
    for j=1:3
        for i=1:Nc(j);
            Gaux(i:end,k+i)=Saux(1:Np-i+1,j);
        end
        k=sum(Nc(1:j));
    end
    G=[G;Gaux];
end
G=G(2:end,:);
%=========================================================================
%              Cálculo de las matrices alfa,lambda y M
%=========================================================================
alfa=blkdiag(alfay1*eye(Npy1,Npy1),alfay2*eye(Npy2,Npy2),alfay3*eye(Npy3,Npy3));
lambda=blkdiag(lambdau1*eye(Ncu1,Ncu1),lambdau2*eye(Ncu2,Ncu2),lambdau3*eye(Ncu3
,Ncu3));
M=inv(G'*alfa*G+lambda)*G'*alfa;
%=========================================================================
%              Cálculo de las matrices Q1 y Q2
%=========================================================================
Q1y1=zeros(Ny1,Ny1);
Q1y1(1:end-1,2:end)=eye(Ny1-1,Ny1-1);
Q1y1(end,end)=1;
Q1y2=zeros(Ny2,Ny2);
Q1y2(1:end-1,2:end)=eye(Ny2-1,Ny2-1);
Q1y2(end,end)=1;
Q1y3=zeros(Ny3,Ny3);
Q1y3(1:end-1,2:end)=eye(Ny3-1,Ny3-1);
Q1y3(end,end)=1;
Q1=blkdiag(Q1y1,Q1y2,Q1y3);
Q2=0;
for i=1:3;
    switch i
        case 1
            Np=Npy1;
            N=Ny1;
            Q2aux1=Q1y1;
        case 2
            Np=Npy2;
            N=Ny2;
            Q2aux1=Q1y2;
```

```
        case 3
            Np=Npy3;
            N=Ny3;
            Q2aux1=Q1y3;
    end
    Q2aux2=zeros(Np,N);
    if Np==N;
        Q2aux2=Q2aux1;
    elseif Np<N
        Q2aux2(1:end,2:Np+1)=eye(Np,Np);
    else
        Q2aux2(1:N-1,2:end)=eye(N-1,N-1);
        Q2aux2(N:end,end)=1;
    end
    Q2=blkdiag(Q2,Q2aux2);
end
Q2=Q2(2:end,2:end);
%=========================================================================
%         matrices del algoritmo recursivo
%=========================================================================
Pm0=inv(G'*alfa*G+lambda);
%=========================================================================
%                        condiciones iniciales
%=========================================================================
t=(0:1:instantes);%vector de tiempo total
deltau1vec=0;
deltau2vec=0;
deltau3vec=0;
u1vec=0;%vector que contendra las entadas aplicadas al sistema
u2vec=0;
u3vec=0;
ym1vec=0;%salida en el instante actual
ym2vec=0;
ym3vec=0;
yp1vec=0;%vector que contendra las salidas predicha de cada instante
yp2vec=0;
yp3vec=0;
zy1u1=0;%cond. iniciales para poder simular los sistemas mediante el comando filter
zy1u2=0;
zy1u3=0;
zy2u1=0;
zy2u2=0;
zy2u3=0;
zy3u1=0;
zy3u2=0;
zy3u3=0;
rety1u1=zeros(1,round(del1/Ts));%vectores para tener en cuenta los retardos de los
%sistemas para poder simular los sistemas mediante el comando filter
rety1u2=zeros(1,round(del2/Ts));
rety1u3=zeros(1,round(del3/Ts));
rety2u1=zeros(1,round(del4/Ts));
rety2u2=zeros(1,round(del5/Ts));
rety2u3=zeros(1,round(del6/Ts));
rety3u1=zeros(1,round(del7/Ts));
rety3u2=zeros(1,round(del8/Ts));
Ry1=ry1*ones(instantes+Npy1,1);%vectores de referencias
Ry2=ry2*ones(instantes+Npy2,1);
Ry3=ry3*ones(instantes+Npy3,1);
```

```
L=[ym1vec*ones(Ny1,1);ym2vec*ones(Ny2,1);ym3vec*ones(Ny3,1)];%inicializacion de la
%resuesta libre
%=========================================================================
%                    Ejecución del algoritmo recursivo
%=========================================================================
for i=1:instantes;
   L=Q1*L+Su*[deltau1vec(end);deltau2vec(end);deltau3vec(end)];%actualización de la rta
%libre
   w=[(ym1vec(end)-L(1))*ones(Npy1,1);...
     (ym2vec(end)-L(Ny1+1))*ones(Npy2,1);...
     (ym3vec(end)-L(Ny1+Ny2+1))*ones(Npy3,1)];%estimación de la perturbación no
%medible
   T=Q2*L+w;
   R=[Ry1(i:i+Npy1-1,1); Ry2(i:i+Npy2-1,1); Ry3(i:i+Npy3-1,1)];
   D=R-T;
   deltau=M*D;
%=========================================================================
   % aplicación del algoritmo recursivo
%=========================================================================
   cont=0;
   Pm=Pm0;
   while (cont<nn)
      yp=T+G*deltau;%salida predicha
      u1fut=u1vec(end)+cumsum(deltau(1:Ncu1));
      u2fut=u2vec(end)+cumsum(deltau(Ncu1+1:Ncu1+1:Ncu1+Ncu2));
      u3fut=u3vec(end)+cumsum(deltau(Ncu1+Ncu2+1:end));
      flagu1=verificar(u1fut,u1max,u1min);
      flagu2=verificar(u2fut,u2max,u2min);
      flagu3=verificar(u3fut,u3max,u3min);
      flagdeltau1=verificar(deltau(1:Ncu1)          ,deltau1max,deltau1min);
      flagdeltau2=verificar(deltau(Ncu1+1:Ncu1+Ncu2),deltau2max,deltau2min);
      flagdeltau3=verificar(deltau(Ncu1+Ncu2+1:end) ,deltau3max,deltau3min);
      flagyp1=verificar(yp(1:Npy1)          ,y1max,y1min);
      flagyp2=verificar(yp(Npy1+1:Npy1+Npy2),y2max,y2min);
      flagyp3=verificar(yp(Npy1+Npy2+1:end) ,y3max,y3min);
      if  0==sum([flagu1 flagu2 flagu3 flagdeltau1 flagdeltau2 flagdeltau3...
            flagyp1 flagyp2 flagyp3])
         break
      end
      cont=cont+1;
      E=0;
      h=zeros(1,Ncu1+Ncu2+Ncu3);
      if flagu1>0
         [Eu1,hu1]=u_rest(flagu1,u1fut,u1vec(end),u1max,u1min,w2,tol2);
         E=[E;Eu1];
         h(2:size(hu1,1)+1,1:Ncu1)=hu1;
      end
      if flagu2>0
         [Eu2,hu2]=u_rest(flagu2,u2fut,u2vec(end),u2max,u2min,w2,tol2);
         E=[E;Eu2];
         h(end+1:end+size(hu2,1),Ncu1+1:Ncu1+Ncu2)=hu2;
      end
      if flagu3>0
         [Eu3,hu3]=u_rest(flagu3,u3fut,u3vec(end),u3max,u3min,w2,tol2);
         E=[E;Eu3];
         h(end+1:end+size(hu3,1),Ncu1+Ncu2+1:end)=hu3;
      end
      if flagdeltau1>0
```

```
[Edeltau1,hdeltau1]=deltau_rest(flagdeltau1,deltau(1:Ncu1),deltau1max,deltau1min,w1,tol1);
            E=[E;Edeltau1];
            h(end+1:end+size(hdeltau1,1),1:Ncu1)=hdeltau1;
        end
        if flagdeltau2>0

[Edeltau2,hdeltau2]=deltau_rest(flagdeltau2,deltau(Ncu1+1:Ncu1+Ncu2),deltau2max,deltau2
min,w1,tol1);
            E=[E;Edeltau2];
            h(end+1:end+size(hdeltau2,1),Ncu1+1:Ncu1+Ncu2)=hdeltau2;
        end
        if flagdeltau3>0

[Edeltau3,hdeltau3]=deltau_rest(flagdeltau3,deltau(Ncu1+Ncu2+1:end),deltau3max,deltau3
min,w1,tol1);
            E=[E;Edeltau3];
            h(end+1:end+size(hdeltau3,1),Ncu1+Ncu2+1:end)=hdeltau3;
        end
        if flagyp1>0
            [Ey1,hy1]=pred_rest(flagyp1,yp(1:Npy1),y1max,y1min,w3,tol3,G,T);
            E=[E;Ey1];
            h=[h;hy1];
        end
        if flagyp2>0

[Ey2,hy2]=pred_rest(flagyp2,yp(Npy1+1:Npy1+Npy2),y2max,y2min,w3,tol3,G(Npy1+1:end,:)
,T(Npy1+1:end));
            E=[E;Ey2];
            h=[h;hy2];
        end
        if flagyp3>0

[Ey3,hy3]=pred_rest(flagyp3,yp(Npy1+Npy2+1:end),y3max,y3min,w3,tol3,G(Npy1+Npy2+1:
end,:),T(Npy1+Npy2+1:end));
            E=[E;Ey3];
            h=[h;hy3];
        end
        E=E(2:end);
        h=h(2:end,:)';
        m1=h'*Pm;
        m2=Pm*h*inv(m1*h+eye(size(h,2)));
        deltau=deltau+m2*(E-h'*deltau);
        Pm=Pm-m2*m1;
    end
%=====================================================================
    %                  Actualización de vectores
%=====================================================================
    yp=T+G*deltau;%salida predicha
    yp1vec(i+1)=yp(1);%salida predicha del instante Yp(k+1|k)
    yp2vec(i+1)=yp(Npy1+1);%salida predicha del instante Yp(k+1|k)
    yp3vec(i+1)=yp(Npy1+Npy2+1);%salida predicha del instante Yp(k+1|k)
    deltau1vec(i+1)=deltau(1);%deltau del instante deltau(k|k)
    deltau2vec(i+1)=deltau(Ncu1+1);%deltau del instante deltau(k|k)
    deltau3vec(i+1)=deltau(Ncu1+Ncu2+1);%deltau del instante deltau(k|k)
    u1vec(i+1)=u1vec(end)+deltau(1);%accion de control del instante u(k|k)
    u2vec(i+1)=u2vec(end)+deltau(Ncu1+1);%accion de control del instante u(k|k)
    u3vec(i+1)=u3vec(end)+deltau(Ncu1+Ncu2+1);%accion de control del instante u(k|k)
```

```
%=================================================================
   %      aplicacion de los incrementos y medición de la salida
%=================================================================
   [y1_,zy1u1]=filter(numy1u1(1:end),deny1u1,u1vec(end),zy1u1);
   [y2_,zy1u2]=filter(numy1u2(1:end),deny1u2,u2vec(end),zy1u2);
   [y3_,zy1u3]=filter(numy1u3(1:end),deny1u3,u3vec(end),zy1u3);
   rety1u1=[y1_ rety1u1(1:end-1)];
   rety1u2=[y2_ rety1u2(1:end-1)];
   rety1u3=[y3_ rety1u3(1:end-1)];
   ym1vec(i+1)=rety1u1(end)+rety1u2(end)+rety1u3(end);
   [y1_,zy2u1]=filter(numy2u1(1:end),deny2u1,u1vec(end),zy2u1);
   [y2_,zy2u2]=filter(numy2u2(1:end),deny2u2,u2vec(end),zy2u2);
   [y3_,zy2u3]=filter(numy2u3(1:end),deny2u3,u3vec(end),zy2u3);
   rety2u1=[y1_ rety2u1(1:end-1)];
   rety2u2=[y2_ rety2u2(1:end-1)];
   rety2u3=[y3_ rety2u3(1:end-1)];
   ym2vec(i+1)=rety2u1(end)+rety2u2(end)+rety2u3(end);
   [y1_,zy3u1]=filter(numy3u1(1:end),deny3u1,u1vec(end),zy3u1);
   [y2_,zy3u2]=filter(numy3u2(1:end),deny3u2,u2vec(end),zy3u2);
   [y3_,zy3u3]=filter(numy3u3(2:end),deny3u3,u3vec(end),zy3u3);
   rety3u1=[y1_ rety3u1(1:end-1)];
   rety3u2=[y2_ rety2u2(1:end-1)];
   rety3u3=y3_;
   ym3vec(i+1)=rety3u1(end)+rety3u2(end)+rety3u3(end);
end
Ry1=Ry1(1:instantes+1);
Ry2=Ry2(1:instantes+1);
Ry3=Ry3(1:instantes+1);
deltau1vec=deltau1vec(2:end);
deltau2vec=deltau2vec(2:end);
deltau3vec=deltau3vec(2:end);
u1vec=u1vec(2:end);
u2vec=u2vec(2:end);
u3vec=u3vec(2:end);
%=================================================================
%                    Gráficas de las variables
%=================================================================
t2=[t(1) t(end)];
figure()
plot(t,ym1vec,'b',t,ym2vec,'r',t,ym3vec,'g');
hold on
stairs(t,yp1vec,'b');
stairs(t,yp2vec,'r');
stairs(t,yp3vec,'g');
stairs(t,Ry1,'--b');
stairs(t,Ry2,'--r');
stairs(t,Ry3,'--g');
%plot(t2,[1 1]*y1max,':b', t2, [1 1]*y2max,':r',t2,[1 1]*y3max,':g');
%plot(t2,[1 1]*y1min,':b', t2, [1 1]*y2min,':r',t2,[1 1]*y3min,':g');
hold off
title('Salidas')
xlabel('Instantes')
legend('Y1','Y2','Y3')
figure()
t=t(1:end-1);
subplot(211)
stairs(t,u1vec,'b');
hold on
```

```
stairs(t,u2vec,'r');
stairs(t,u3vec,'g');
plot(t2,[1 1]*u1max,':b', t2, [1 1]*u2max,':r',t2,[1 1]*u3max,':g');
plot(t2,[1 1]*u1min,':b', t2, [1 1]*u2min,':r',t2,[1 1]*u3min,':g');
hold off
title('Acciones de control')
xlabel('Instantes')
legend('u1','u2','u3')
subplot(212)
stairs(t,deltau1vec,'b');
hold on
stairs(t,deltau2vec,'r');
stairs(t,deltau3vec,'g');
plot(t2,[1 1]*deltau1max,':b', t2, [1 1]*deltau2max,':r',t2,[1 1]*deltau3max,':g');
plot(t2,[1 1]*deltau1min,':b', t2, [1 1]*deltau2min,':r',t2,[1 1]*deltau3min,':g');
hold off
title('Incrementos en las acciones de control')
xlabel('Instantes')
legend('deltau1','deltau2','deltau3')
%=========================================================================
%verificar si los incrementos, entradas o predicciones violan restricciones
%=========================================================================
function flag=verificar(vector,lim_max,lim_min)
flag=0;
if min(vector)<lim_min
  flag=flag+1;
end
if max(vector)>lim_max
  flag=flag+2;
end
%=========================================================================
%    Cálculo de las matrices de restricciones h y Em+1 para las entradas
%=========================================================================
function  [E,h]=u_rest(flag,U_fut,U,U_max,U_min,W,tol)
a=length(U_fut);
switch flag
   case 1
      flag2=1;
      h=zeros(1,a);
      E=(U_min+tol-U)*W;
      k=2;
      m=1;
      for j=1:a;
         if U_fut(j)<U_min && flag2==1;
            h(1,1:j)=W;
            flag2=2;
         elseif flag2==2
            h(k,j)=W;
            k=k+1;
         end
      end
   case 2
      flag1=1;
      h=zeros(1,a);
      E=(U_max-tol-U)*W;
      k=2;
      m=1;
      for j=1:a;
```

```
            if U_fut(j)>U_max && flag1==1;
                h(1,1:j)=W;
                flag1=2;
            elseif flag1==2
                h(k,j)=W;
                k=k+1;
            end
        end
    case 3
        flag2=1;
        flag1=1;
        h=zeros(2,a);
        E=[U_max-tol-U;U_min+tol-U]*W;
        k=3;
        m=2;
        for j=1:a;
            if U_fut(j)>U_max && flag1==1;
                h(1,1:j)=W;
                flag1=2;
            elseif flag1==2
                h(k,j)=W;
                k=k+1;
            end
            if U_fut(j)<U_min && flag2==1;
                h(m,1:j)=W;
                flag2=2;
            elseif flag2==2 && h(k-1,j)==0
                h(k,j)=W;
                k=k+1;
            end
        end
end
E=[E;zeros(size(h,1)-m,1)];
%=========================================================================
% Cálculo de las matrices de restricciones h y Em+1 para los incrementos
%=========================================================================
function  [E,h]=deltau_rest(flag,deltau,deltau_max,deltau_min,W,tol)
h=zeros(1,length(deltau));
E=0;
switch flag
    case 1
        for j=1:length(deltau);
            if  deltau(j)<deltau_min;
                h(end+1,j)=W;
                E(end+1,1)=(deltau_min+tol)*W;
            end
        end
    case 2
        for j=1:length(deltau);
            if  deltau(j)>deltau_max;
                h(end+1,j)=W;
                E(end+1,1)=(deltau_max-tol)*W;
            end
        end
    case 3
        for j=1:length(deltau);
            if  deltau(j)>deltau_max;
                h(end+1,j)=W;
```

```
            E(end+1,1)=(deltau_max-tol)*W;
        end
        if  deltau(j)<deltau_min;
            h(end+1,j)=W;
            E(end+1,1)=(deltau_min+tol)*W;
        end
    end
end
h=h(2:end,:);
E=E(2:end,1);
%=====================================================================
%Matrices de restricciones h y Em+1 para las predicciones en las salidas
%=====================================================================
function  [E,h]=pred_rest(flag,ypred,y_max,y_min,W,tol,G,L)
h=zeros(1,size(G,2));
E=0;
switch flag
   case 1
      for j=1:length(ypred);
         if ypred(j)<y_min;
            h(end+1,:)=G(j,:)*W;
            E(end+1,1)=(y_min+tol-L(j))*W;
         end
      end
   case 2
      for j=1:length(ypred);
         if ypred(j)>y_max;
            h(end+1,:)=G(j,:)*W;
            E(end+1,1)=(y_max-tol-L(j))*W;
         end
      end
   case 3
      for j=1:length(ypred);
         if ypred(j)>y_max;
            h(end+1,:)=G(j,:)*W;
            E(end+1,1)=(y_max-tol-L(j))*W;
         end
         if ypred(j)<y_min;
            h(end+1,:)=G(j,:)*W;
            E(end+1,1)=(y_min+tol-L(j))*W;
         end
      end
end
h=h(2:end,:);
E=E(2:end,1);
%=====================================================================
%=====================================================================
```

CÓDIGOS EN MATLAB PARA IMPLEMENTACIONES EN LABVIEW

CÓDIGO PARA EL MODELO QDMC

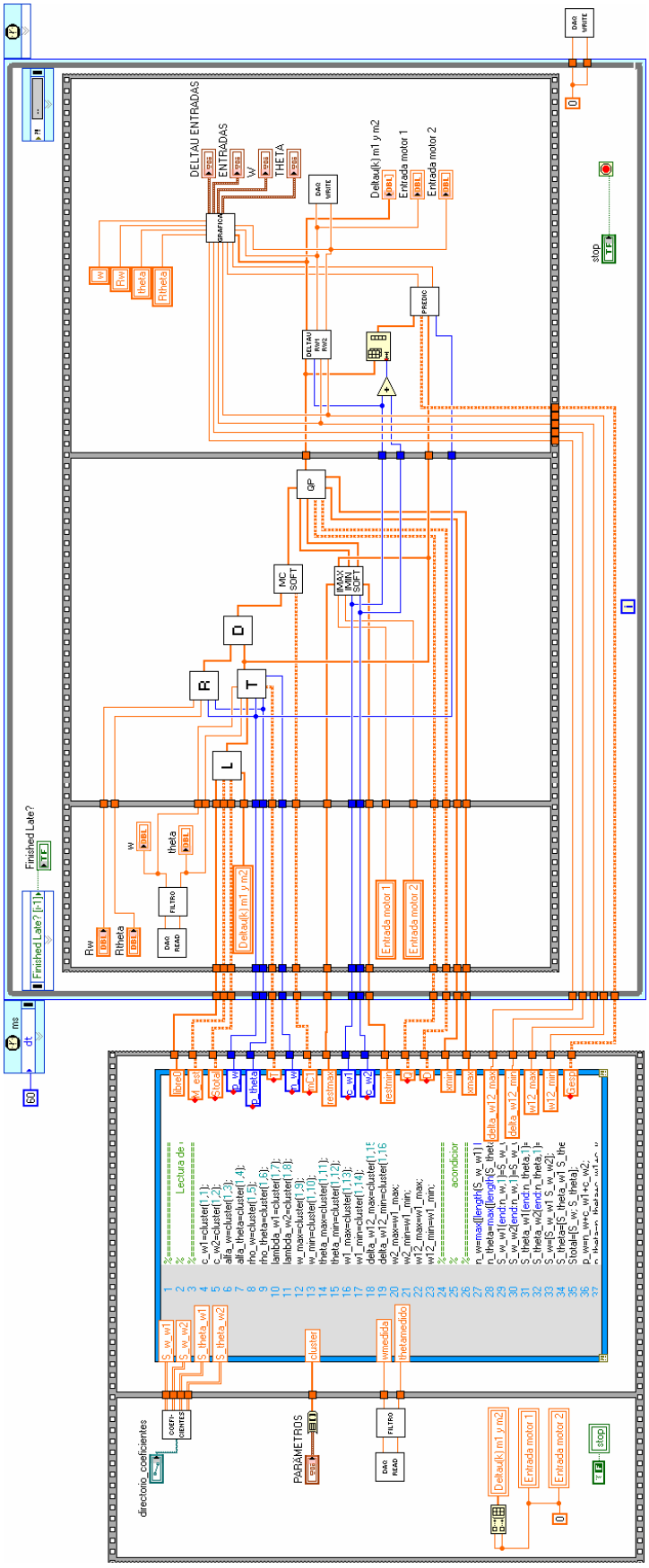
```
%=====================================================================
%                        Lectura de datos
%=====================================================================
c_w1=cluster(1,1);
```

```
c_w2=cluster(1,2);
alfa_w=cluster(1,3);
alfa_theta=cluster(1,4);
rho_w=cluster(1,5);
rho_theta=cluster(1,6);
lambda_w1=cluster(1,7);
lambda_w2=cluster(1,8);
w_max=cluster(1,9);
w_min=cluster(1,10);
theta_max=cluster(1,11);
theta_min=cluster(1,12);
w1_max=cluster(1,13);
w1_min=cluster(1,14);
delta_w12_max=cluster(1,15);
delta_w12_min=cluster(1,16);
w2_max=w1_max;
w2_min=w1_min;
w12_max=w1_max;
w12_min=w1_min;
%=========================================================================
%                          acondicionamiento de datos
%=========================================================================
n_w=max([length(S_w_w1) length(S_w_w2)]);
n_theta=max([length(S_theta_w1) length(S_theta_w2)]);
S_w_w1(end:n_w,1)=S_w_w1(end);
S_w_w2(end:n_w,1)=S_w_w2(end);
S_theta_w1(end:n_theta,1)=S_theta_w1(end);
S_theta_w2(end:n_theta,1)=S_theta_w2(end);
S_w=[S_w_w1 S_w_w2];
S_theta=[S_theta_w1 S_theta_w2];
Stotal=[S_w; S_theta];
p_w=n_w+c_w1+c_w2;
p_theta=n_theta+c_w1+c_w2;
%=========================================================================
%                        Cálculo de la matriz G
%=========================================================================
c=[c_w1 c_w2];
G=zeros(1,sum(c));
for i=1:2
    switch i;
        case 1
            p=p_w;
            n=n_w;
            S=S_w;
        case 2
            p=p_theta;
            n=n_theta;
            S=S_theta;
    end
    Saux=[S; ones(p-n,2)*[S(end,1) 0; 0 S(end,2)]];
    Gaux=zeros(p,sum(c));
    k=0;
    for j=1:2
        for i=1:c(j);
            Gaux(i:end,k+i)=Saux(1:p-i+1,j);
        end
        k=sum(c(1:j));
    end
    G=[G;Gaux];
```

```
end
G=G(2:end,:);
%=====================================================================
%                 Cálculo de las matrices alfa y lambda
%=====================================================================
alfa=alfa_w*eye(p_w,p_w);
alfa(end+1:end+p_theta,end+1:end+p_theta)=alfa_theta*eye(p_theta,p_t
heta);
lambda=lambda_w1*eye(c_w1,c_w1);
lambda(end+1:end+c_w2,end+1:end+c_w2)=lambda_w2*eye(c_w2,c_w2);
%=====================================================================
%               Cálculo de las matrices M. estrella y T
%=====================================================================
M_w_est=zeros(n_w,n_w);
M_w_est(1:end-1,2:end)=eye(n_w-1,n_w-1);
M_w_est(end,end)=1;
M_theta_est=zeros(n_theta,n_theta);
M_theta_est(1:end-1,2:end)=eye(n_theta-1,n_theta-1);
M_theta_est(end,end)=1;
M_est=M_w_est;
M_est(end+1:end+n_theta,end+1:end+n_theta)=M_theta_est;
T=0;
for i=1:2;
    switch i
        case 1
            p=p_w;
            n=n_w;
            M=M_w_est;
        case 2
            p=p_theta;
            n=n_theta;
            M=M_theta_est;
    end
    Taux=zeros(p,n);
    Taux(1:n-1,2:end)=eye(n-1,n-1);
    Taux(n:end,end)=1.0;
    T(end+1:end+p,end+1:end+n)=Taux;
end
T=T(2:end,2:end);
%=====================================================================
%                        Matrices para el algoritmo QP
%=====================================================================
Q=[[G'*alfa*G+lambda,zeros(c_w1+c_w2,2)];[zeros(2,c_w1+c_w2),[rho_w
0;0 rho_theta]]];
mC1=-G'*alfa;
D1=zeros(c_w1+c_w2,c_w1+c_w2);
for j=1:2
    if j==1;
        a=1;
        b=c_w1;
    elseif j==2
        a=c_w1+1;
        b=c_w1+c_w2;
    end
    for k=a:b;
        D1(k,a:k)=1;
    end
end
D1=[D1, zeros(size(D1,1),2)];
```

```
a=zeros(p_w+p_theta,2);
a(1:p_w,1)=-1;
a(p_w+1:p_w+p_theta,2)=-1;
D2=[[G; G],[a;-a]];
b=[zeros(2,c_w1+c_w2),[-1 0;0 -1]];
D=[D1;D2;b;-b];
Urestmax=[w1_max*ones(c_w1,1); w2_max*ones(c_w2,1)];
Urestmin=[w1_min*ones(c_w1,1); w2_min*ones(c_w2,1)];
Yrestmax=[w_max*ones(p_w,1); theta_max*ones(p_theta,1);...
    inf*ones(p_w+p_theta,1);0;0;inf;inf];
Yrestmin=[-inf*ones(p_w+p_theta,1); w_min*ones(p_w,1);
theta_min*ones(p_theta,1);...
    -inf;-inf;0;0];
restmax=[Urestmax; Yrestmax];
restmin=[Urestmin; Yrestmin];
xmin=[delta_w12_min*ones(c_w1,1); delta_w12_min*ones(c_w2,1); 0; 0];
xmax=[delta_w12_max*ones(c_w1,1); delta_w12_max*ones(c_w2,1); inf;
inf];
%=====================================================================
%                        matrices para el inicio
%=====================================================================
libre0=[wmedida*ones(n_w,1);thetamedido*ones(n_theta,1)];%inicializa
cion de la resuesta libre
Gesp=[G(1,:);G(p_w+1,:)];
```

QDMC, panel frontal en Labview



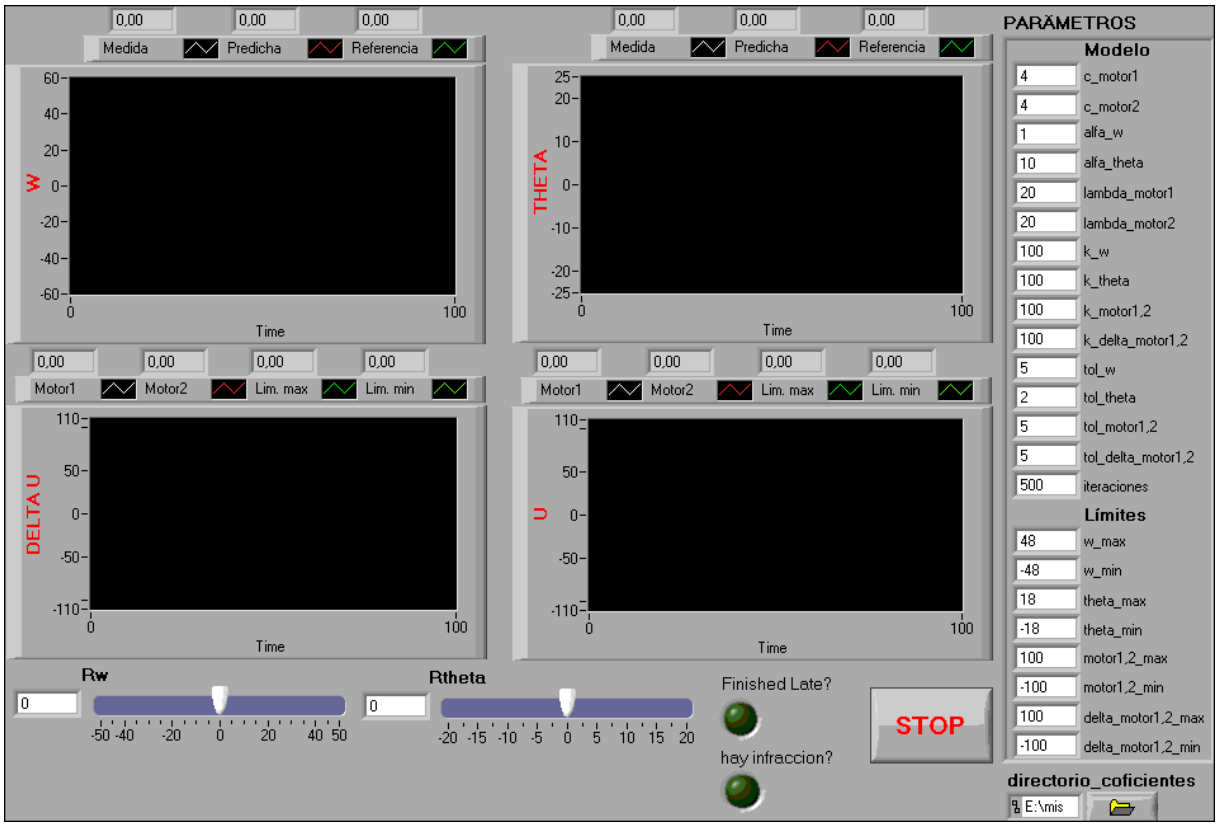
QDMC, diagrama de bloques en Labview

CÓDIGO PARA EL MODELO DMC RLS

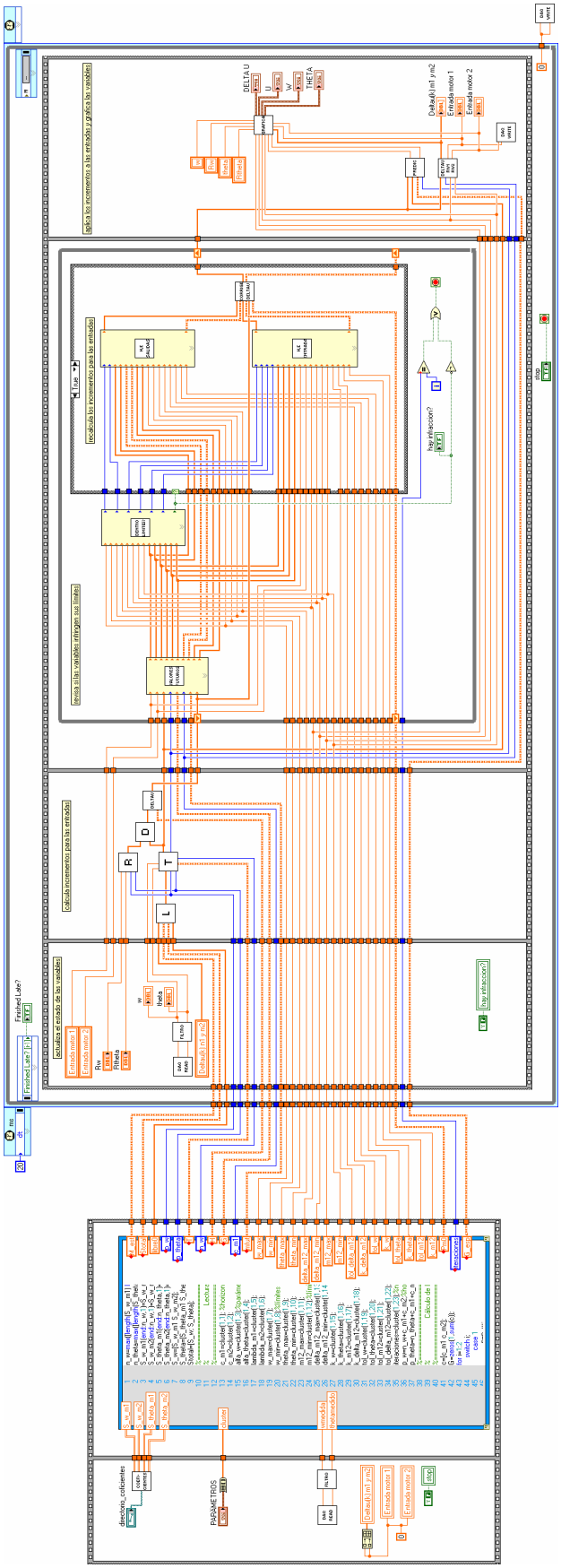
```
n_w=max([length(S_w_m1) length(S_w_m2)]);
n_theta=max([length(S_theta_m1) length(S_theta_m2)]);
S_w_m1(end:n_w,1)=S_w_m1(end);
S_w_m2(end:n_w,1)=S_w_m2(end);
S_theta_m1(end:n_theta,1)=S_theta_m1(end);
S_theta_m2(end:n_theta,1)=S_theta_m2(end);
S_w=[S_w_m1 S_w_m2];
S_theta=[S_theta_m1 S_theta_m2];
Stotal=[S_w; S_theta];
%=====================================================================
%                              Lectura Parámetros
%=====================================================================
c_m1=cluster(1,1); %horizontes de control
c_m2=cluster(1,2);
alfa_w=cluster(1,3);%parámetros alfa lambda y W
alfa_theta=cluster(1,4);
lambda_m1=cluster(1,5);
lambda_m2=cluster(1,6);
w_max=cluster(1,7);
w_min=cluster(1,8);%límites máximos y mínimos de las VC
theta_max=cluster(1,9);
theta_min=cluster(1,10);
m12_max=cluster(1,11);
m12_min=cluster(1,12);%límites máximos y mínimos y de incrementos de
las VM
delta_m12_max=cluster(1,13);
delta_m12_min=cluster(1,14);
k_w=cluster(1,15);
k_theta=cluster(1,16);
k_m12=cluster(1,17);
k_delta_m12=cluster(1,18);
tol_w=cluster(1,19);
tol_theta=cluster(1,20);
tol_m12=cluster(1,21);
tol_delta_m12=cluster(1,22);
iteraciones=cluster(1,23);%número de veces que se ejecuta el
algoritmo recursivo para encontrar solución
p_w=n_w+c_m1+c_m2;%horizontes de predicción
p_theta=n_theta+c_m1+c_m2;
%=====================================================================
%                         Cálculo de la matriz G
%=====================================================================
c=[c_m1 c_m2];
G=zeros(1,sum(c));
for i=1:2
    switch i;
        case 1
            p=p_w;
            n=n_w;
            S=S_w;
        case 2
            p=p_theta;
            n=n_theta;
            S=S_theta;
    end
    Saux=[S; ones(p-n,2)*[S(end,1) 0; 0 S(end,2)]];
```

```
    Gaux=zeros(p,sum(c));
    k=0;
    for j=1:2
        for i=1:c(j);
            Gaux(i:end,k+i)=Saux(1:p-i+1,j);
        end
        k=sum(c(1:j));
    end
    G=[G;Gaux];
end
G=G(2:end,:);
%=====================================================================
%                Cálculo de las matrices alfa y lambda H y Pm
%=====================================================================
alfa=alfa_w*eye(p_w,p_w);
alfa(end+1:end+p_theta,end+1:end+p_theta)=alfa_theta*eye(p_theta,p_t
heta);
lambda=lambda_m1*eye(c_m1,c_m1);
lambda(end+1:end+c_m2,end+1:end+c_m2)=lambda_m2*eye(c_m2,c_m2);
%=====================================================================
%               Cálculo de las matrices M. estrella y T
%=====================================================================
M_w_est=zeros(n_w,n_w);
M_w_est(1:end-1,2:end)=eye(n_w-1,n_w-1);
M_w_est(end,end)=1;
M_theta_est=zeros(n_theta,n_theta);
M_theta_est(1:end-1,2:end)=eye(n_theta-1,n_theta-1);
M_theta_est(end,end)=1;
M_est=M_w_est;
M_est(end+1:end+n_theta,end+1:end+n_theta)=M_theta_est;
T=0;
for i=1:2;
    switch i
        case 1
            p=p_w;
            n=n_w;
            M=M_w_est;
        case 2
            p=p_theta;
            n=n_theta;
            M=M_theta_est;
    end
    Taux=zeros(p,n);
    Taux(1:n-1,2:end)=eye(n-1,n-1);
    Taux(n:end,end)=1.0;
    T(end+1:end+p,end+1:end+n)=Taux;
end
T=T(2:end,2:end);
%=====================================================================
% matrices de la patente y condiciones iniciales
%=====================================================================
H=inv(G'*alfa*G+lambda)*G'*alfa;
Pm0=inv(G'*alfa*G+lambda);
mfut=zeros(c_m1+c_m2,c_m1+c_m2);
for j=1:2
    if j==1;
        a=1;
        b=c_m1;
    elseif j==2
```

```
        a=c_m1+1;
        b=c_m1+c_m2;
    end
    for k=a:b;
        mfut(k,a:k)=1;
    end
end
libre0=[wmedida*ones(n_w,1);thetamedido*ones(n_theta,1)];%inicializa
cion de la resuesta libre
G_esp=[G(1,:);G(p_w+1,:)];
```



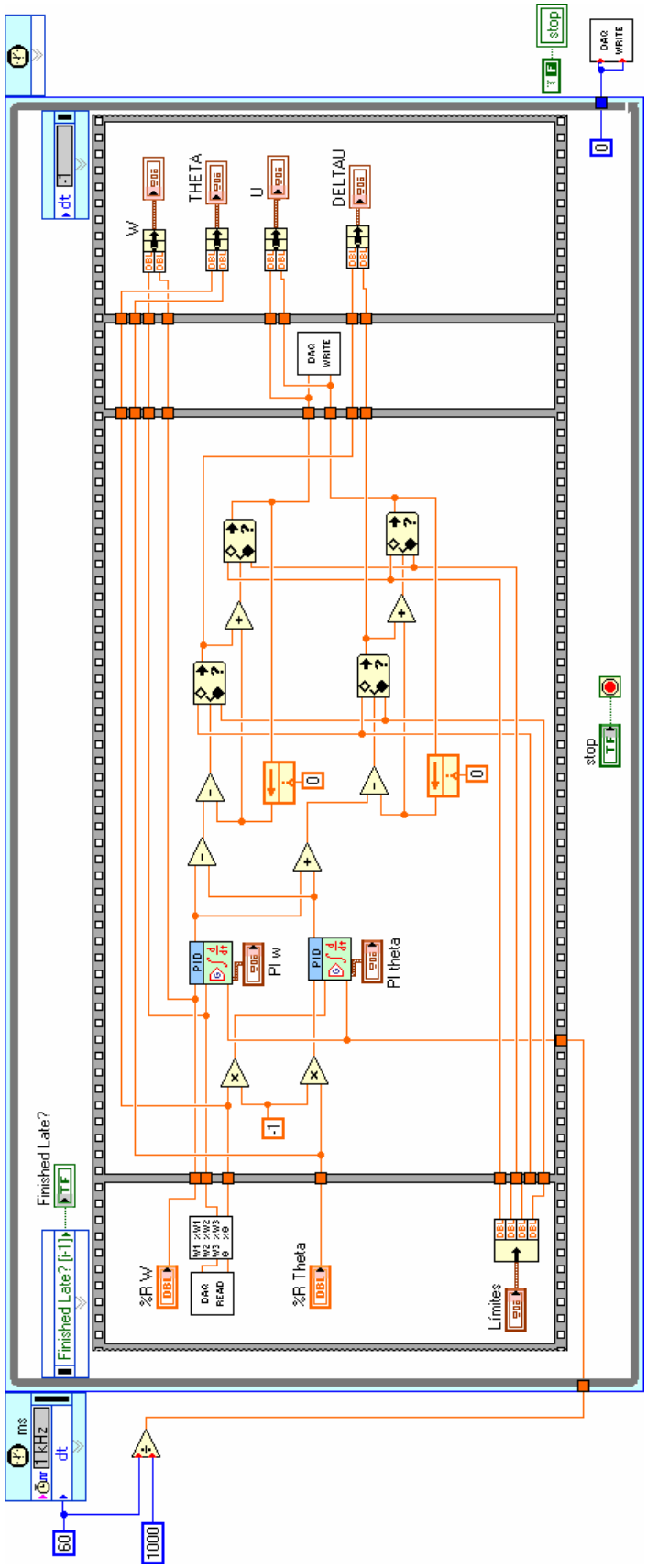
DMC RLS, panel frontal en Labview



DMC RLS, diagrama de bloques en Labview

IMÁGENES DEL PANEL DE CONTROL Y DIAGRAMA DE BLOQUES DEL ESQUEMA DE CONTROL CLÁSICO

Control clásico, panel frontal en Labview



Control clásico, diagrama de bloques en Labview