UNIVERSIDAD POLITECNICA DE VALENCIA

Departamento de
Ingeniería de Sistemas y Automática

grupo de control predictivo y optimización heurística

Control predictivo basado en modelos mediante técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables

F. Xavier Blasco Ferragud

Tesis Doctoral dirigida por:

Miguel A. Martínez Iranzo

Agradecimientos

En primer lugar y por encima de todo quiero agradecer a Miguel Martínez por todo el esfuerzo que ha invertido en este trabajo, sin su apoyo y ayuda probablemente esto no existiría. También son responsables de que esto acabe bien mis compañeros de grupo de investigación Juanse y Javi, puesto que les ha tocado echar una mano cuando ha sido necesario y aguantar mis lamentos a lo largo de este casi interminable proceso.

Agradecer a ese grupo de gente que forman parte del departamento de Ingeniería de Sistemas y Automática, puesto que en algún momento me han facilitado la tarea de una u otra forma.

A Sebastián Dormido, Eduardo Fernández y Cesar De Prada agradecer los comentarios del primer borrador que han contribuido a mejorar el documento final.

Por último, pero tan importante como el primero, un agradecimiento especial a aquellos que no teniendo ni idea de que va esto (ni falta que les hace) han contribuido a que mantenga la salud mental, me refiero sobre todo a Charo, mi familia y a esos amigos que nunca han podido contener sus risas cuando me preguntaban por la tesis y yo trataba de recitarles el título.

A Vicenta

Índice General

Capítulo 1

Motivación y objetivos de la tesis

El potencial de la metodología de Control Predictivo Basado en Modelos (MBPC) para controlar procesos industriales resulta cada vez más significativo sin más que analizar el número y tipo de procesos industriales en que se vienen empleando [80], lo cual resulta ya de por sí un elemento motivador de aquellos investigadores que pretendan trasladar sus experiencias, en la medida de lo posible, al ámbito industrial.

Un elemento fundamental y al mismo tiempo limitante de la metodología de MBPC lo constituye la Optimización de índices o funciones de costo, puesto que a medida que la complejidad del problema de MBPC crece (modelos no lineales, restricciones de entrada y salida, estabilidad, problemas de tiempo real, etc.) se va a requerir, en general, algoritmos de optimización que garanticen el mínimo global en un tiempo acotado.

Esta Tesis Doctoral se fundamenta, principalmente, en la exploración de ese factor limitante con el fin de establecer nuevos métodos de diseño para MBPC basados en herramientas de optimización que permitan abordar problemas difíciles, como son, entre otros, la presencia de óptimos locales en las funciones de coste.

En ese contexto, la Tesis Doctoral presenta las metodologías Heurísticas de Simulated Annealing y Algoritmos Genéticos como candidatas a la resolución de ese tipo de problemas en MBPC procurando la mejor adaptación de estas técnicas al MBPC y la mejora del ajuste de sus parámetros para la obtención de mejores prestaciones, sin olvidar su coste computacional.

La Tesis Doctoral también presenta ciertas formulaciones o alternativas de otros elementos de la metodología de MBPC, como son la formulación de los índices de costo y de los modelos de predicción (incluyendo las restricciones) las cuales son tratadas por los métodos de optimización propuestos en la tesis de una manera natural y sin presentar las restricciones de los métodos tradicionales.

Se presenta a continuación un resumen del Estado del Arte sobre CPBM que termina con la descripción de las principales líneas de investigación que se siguen actualmente, tratando de situar a la presente Tesis dentro de ellas. También se presenta un resumen del Estado del Arte sobre Optimización con la pretensión de describir hasta donde son capaces de llegar las técnicas conocidas y por qué.

Finaliza este capítulo con el planteamiento de los objetivos específicos que se pretenden conseguir en la presente Tesis.

1.1 Control predictivo basado en modelo (MBPC)

El control predictivo basado en modelos, más que un controlador concreto es una metodología para el cálculo de las acciones de control. Se trata además de una metodología comprensible, en cierta forma trata de reproducir la forma de actuar que tendría un operador experto en el control de un determinado proceso.

Los pasos que seguiría un operador experto para conseguir controlar un proceso serían:

1. El operador conoce bien el proceso y por tanto, sería capaz de predecir con mayor o menor exactitud cual será la evolución dinámica de las variables de un proceso si le aplica unas acciones de control determinadas.

2. El mismo operador puede además, decidir si esa evolución es adecuada en comparación a los objetivos que se ha marcado. Es capaz por tanto de valorar las distintas combinaciones de las acciones de control en función del grado de cumplimiento de unas especificaciones.

3. Con todo esto, podría decidir cual es la mejor combinación de acciones de control dentro de un conjunto de posibilidades. El resultado final es que este operador es capaz de obtener cual debe ser la combinación de acciones de control que hay que aplicar al proceso, todo ello basado en los conocimientos que tiene del proceso y la información del estado pasado y actual del mismo.

4. Para conseguir una mayor calidad en el control, este mismo operador repetiría todos los cálculos cada vez que disponga de información actualizada, bien sean nuevas medidas del estado del proceso, bien conocimientos actualizados acerca del comportamiento del proceso (información nueva del modelo).

Este ejemplo nos da a entender que los primeros controles realizados manualmente por operadores que conocían bien el proceso se podían haber englobado en el área del control

predictivo basado en modelos. En definitiva se trata de una metodología muy intuitiva para abordar el control de un proceso y esto ha influido en su difusión a nivel industrial.

Para concretar, se entiende que pertenecen a la familia de los controladores MBPC aquellos que comparten las siguientes características:

- Se hace uso explícito de un modelo del proceso en el cálculo de predicciones de la evolución dinámica del proceso.
- La ley de control (conjunto de acciones de control en un horizonte de tiempo) se obtiene de la minimización de una cierta función de coste en la que intervienen las predicciones. La función de coste es la encargada de fijar el comportamiento que se pretende conseguir (especificaciones).
- Se aplica el concepto de horizonte móvil (*receding horizon*): en cada periodo de muestreo se calcula la ley de control incorporando nuevas medidas de la evolución dinámica del proceso, pero sólo se aplica la primera acción de dicha ley.

El análisis de esta metodología de control muestra que, sea cual sea la implementación que se realice, cualquier control predictivo basado en modelos se puede entender como un problema de optimización en cada periodo de muestreo (horizonte móvil) que consta de tres elementos fundamentales (figura 1.1):

- Un **predictor**, en el que se utilizan los modelos, y que es el encargado de calcular predicciones de la evolución dinámica del proceso ($[y(t+N_1|t),\ldots,y(t+N_2|t)]$) a partir de la información dinámica que se tiene hasta ese instante (medidas de las variables del proceso hasta el instante actual 't') y una ley de control postulada ($[u(t),\ldots,u(t+N_2)]$), a lo largo del horizonte de predicción ($[N_1,N_2]$).
- Una **función de coste** que asigna un valor (coste $J(u)$) a cada predicción y por tanto a cada ley de control postulada. Este valor trata de mostrar el grado de cumplimiento de las especificaciones estáticas y dinámicas, así como incluir las posibles restricciones de funcionamiento.
- Un **optimizador** que debe encontrar la ley de control que ofrece un mejor valor de la función de coste. Generalmente en este proceso de búsqueda el optimizador realiza postulados de la ley de control e iterativamente trata de acercarse a la ley de control óptima ($\hat{u}=[\hat{u}(t),\ldots,\hat{u}(t+N_2)]$).

Combinando distintas variaciones de estos tres elementos fundamentales se pueden obtener un gran número de controladores que formarían parte de la familia de los controladores predictivos. En la bibliografía se pueden encontrar distintos trabajos que recogen una gran

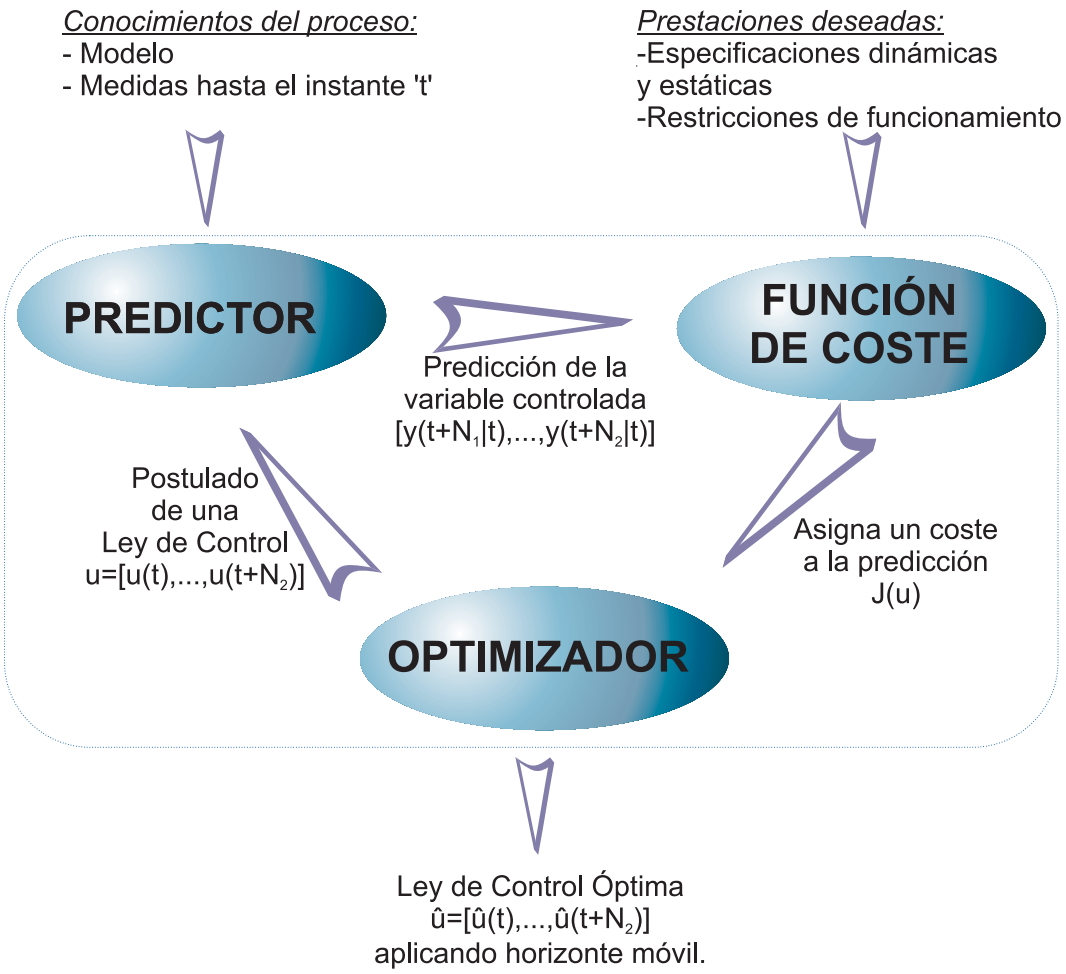


Figura 1.1: Elementos fundamentales de un control predictivo.

parte de los controladores predictivos que se han desarrollado tanto a nivel industrial como a nivel teórico ([80], [37], [69], [72], [16]).

Para poder plantear cualquier tipo de mejora se debe pasar por un primer paso de análisis de estos tres elementos fundamentales.

1.1.1 Predictor

Este elemento del control predictivo es el que se encarga de calcular las predicciones de la evolución dinámica de las variables que se quieren controlar y debe utilizar para ello un modelo. En general este modelo consta de dos componentes (figura 1.2):

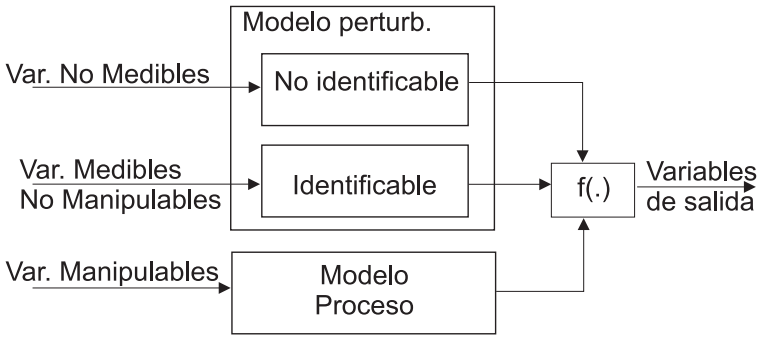


Figura 1.2: Estructura general de un modelo.

- **Modelo del proceso** que relaciona todas las variables de entrada que se pueden manipular con las variables de salida que se quieren controlar.
- **Modelo de perturbaciones** que se puede dividir en dos partes. Una que incluye la relación entre variables de entrada que se pueden medir pero no manipular, con las salidas (modelo de perturbaciones identificable) y otra parte que trata de describir la parte de la salida medida que no es explicada por el resto de modelos (modelo de perturbaciones no identificable).

Estos modelos se combinan a través de una función $f(.)$ para producir un modelo de las variables de salida.

Para modelar cada uno de estos componentes existen varias posibilidades:

- Respuesta ante un impulso.
- Respuesta ante un escalón.

- Función de transferencia.
- Representación en espacio de estados.
- Modelos de Volterra.
- Modelos mediante redes neuronales.
- Modelos fuzzy.

Las diferencias entre los distintos tipos de modelos son conocidas y son objeto de campos de estudio en los que se describe cómo se obtienen, que tipo de procesos pueden modelar, cuáles son sus limitaciones y cómo se utilizan para realizar predicciones de la evolución dinámica de las variables de un proceso. En [77] se puede encontrar una descripción del modelado mediante respuesta impulsional, función de transferencia y representación en espacio de estados, su utilización en control predictivo se puede ver en [16]. Un ejemplo de utilización en control predictivo de modelos de Volterra se puede encontrar en [59]. Las redes neuronales son en algunos casos una alternativa para el modelado de sistemas no lineales [21], ejemplos de utilización en control predictivo se pueden ver en [57], [74] y [105]. En cuanto a los modelos con técnicas fuzzy, se puede ver en [102] el modelado, identificación y control (aunque no son aplicaciones de control predictivo).

Evidentemente, la calidad de la predicción debería ser el factor que marque el tipo de técnica de modelado que se debe utilizar, al menos a nivel teórico. Es natural pensar que a mayor calidad en las predicciones más posibilidades se tienen de obtener un control adecuado. En las aplicaciones prácticas aparecen otros factores como los económicos que pueden determinar, más allá de las cuestiones teóricas, el tipo de modelo seleccionado.

Cualquiera de las técnicas de modelado mencionadas, puede ser utilizada para predecir el efecto, sobre las variables de salida, de las variables manipulables (modelo del proceso). Esto es así puesto que cualquiera de las acciones de control que se postule en el proceso de localización del óptimo, puede aplicarse posteriormente al proceso. Por tanto la calidad de la predicción sólo depende de la calidad del modelo.

No parece tan evidente utilizar estos modelos para predecir el efecto de las variables no manipulables (medibles o no). Cuando se tienen variables no manipulables no se puede saber con certeza que valores van a tomar en el futuro y por tanto en el horizonte de predicción (intervalo de tiempo en el que se quiere realizar la predicción). El modelo puede ser muy exacto, pero el problema es que se desconoce el valor futuro de estas variables y por tanto las predicciones pueden ser malas. En estos casos se debe, al menos, conocer alguna de sus propiedades estadísticas que nos permita realizar una estimación (por ejemplo, la media). Si estas variables son medibles se pueden utilizar las medidas para realizar unas estimaciones de mayor calidad (por ejemplo, evaluar la tendencia y extrapolar). La calidad de la predicción, en estos casos, depende tanto de la calidad

del modelo como de la calidad de las estimaciones que se realizan de las variables no manipulables.

Para tratar de compensar la dinámica no modelada (parte no identificable del modelo) se añaden al modelo elementos artificiales en su estructura para tratar de incrementar la calidad del control, un ejemplo clásico consiste en utilizar modelos CARMA o CARIMA con polinomios de filtrado ($T(z^{-1})$) para mejorar la robustez.

1.1.2 Función de coste

El objetivo de esta función es plasmar en una formulación matemática una medida cuantitativa del funcionamiento de un sistema. En la teoría de control aparecen distintos indicadores que tratan de describir la evolución dinámica del proceso, en general se pueden englobar en dos grandes grupos:

a) Parámetros descriptivos de la evolución temporal de la variable controlada: error en régimen permanente, valor final, tiempo de establecimiento, sobreoscilación, frecuencia de las oscilaciones, tiempo de subida, etc.

b) Medidas de la desviación de la variable de salida respecto de la referencia (error $e(t) = r(t) - y(t)$). Generalmente se han utilizado integrales de una función del error, siendo los más extendidos en su uso: IAE (integral del valor absoluto del error), ICE (integral del error al cuadrado), ITAE (integral del producto del tiempo por el valor absoluto del error), ITEC (integral del producto del tiempo por el error al cuadrado), etc. (ver [60]).

$$IAE = \int_0^\infty |r(t) - y(t)|dt \tag{1.1}$$

$$ICE = \int_0^\infty (r(t) - y(t))^2 dt \tag{1.2}$$

$$ITAE = \int_0^\infty t|r(t) - y(t)|dt \tag{1.3}$$

$$ITEC = \int_0^\infty t(r(t) - y(t))^2 dt \tag{1.4}$$

En general un índice de funcionamiento responde a una expresión:

$$I = \int_0^\infty f\left(r(t), y(t), t\right) dt \tag{1.5}$$

Entre estas opciones, parece claro que es más directo e inteligible, para evaluar cómo funciona un sistema, utilizar parámetros del primer grupo, las medidas de tipo integral no reflejan directamente cómo funciona el sistema. Sólo con el valor del índice un operador es incapaz de saber con cierto grado de exactitud como está funcionando el sistema. A esto se une que es más usual que las especificaciones de funcionamiento se suelen plantear en estos términos, por ejemplo, es normal tener restricciones del tipo tiempo de establecimiento y sobreoscilación menores que unos valores determinados.

Por otra parte, con los indicadores de tipo integral se pueden incorporar en la evaluación del funcionamiento otro tipo de elementos como por ejemplo el valor de la acción de control o ponderaciones de las distintas variables. Generalmente se utilizan para integrar en el índice la evaluación de costes económicos.

En general se deduce que no existe una formulación universal que sea válida para todos los problemas de control, cada tipo de indicadores tiene sus ventajas e inconvenientes. El índice de funcionamiento más adecuado para un problema concreto depende de los objetivos que se impongan y de las herramientas disponibles.

Una alternativa que trata de combinar los dos tipos de indicadores es la de establecer unas trayectorias de referencia en un índice de tipo integral. En lugar de utilizar una referencia de tipo escalón ($r(t)$), se filtra mediante una función de transferencia ($P(s)$) que cumpla con los objetivos del tipo: tiempo de establecimiento, sobreoscilación y régimen permanente, y se utiliza esta respuesta ($w(t)$) como referencia en un índice de funcionamiento de tipo integral.

$$w(s) = P(s)r(s) \rightarrow w(t)$$
$$I = \int_0^\infty f\left(w(t), y(t), u(t), t\right) dt$$

La traducción al entorno de un control predictivo es directa, el índice de coste debe, de alguna forma, imponer el funcionamiento deseado del proceso utilizando estos indicadores de funcionamiento. Sin embargo, el problema se complica más todavía puesto que el control predictivo introduce nuevas características y variables en el índice para posibilitar su aplicación al control en línea (se debe conseguir que el volumen de los cálculos a realizar sea razonable).

Una característica común en las funciones de coste es que operan con señales discretas, aparecen además parámetros adicionales del índice que se deben ajustar. Se puede generalizar como función de coste la expresión siguiente:

$$J(u) = \sum_{k=N_1}^{N_2} f\left(w(t+k), y(t+k|t), u(t+k), \delta(k), \lambda(k), N_u, t\right) + J_e \qquad (1.6)$$

- Horizonte de predicción ($[N_1, N_2]$), intervalo de tiempo en el que se realiza la predicción, debe ser finito para que sea posible su aplicación al control en línea. Un horizonte de predicción infinito o demasiado grande hace que los cálculos de las predicciones no se puedan realizar en línea, salvo que exista una formulación analítica.

- Horizonte de control (N_u), intervalo del horizonte de predicción en el que se permiten variaciones de la variable manipulada. Este parámetro se utiliza para simplificar el problema de optimización, si se reduce el número de variaciones se reduce el número de variables en el problema. Cuando se limita el horizonte de control cabe la posibilidad de realizar distintas estructuraciones de las leyes de control a lo largo del horizonte de predicción (influye en las prestaciones). Por tanto, junto con el parámetro N_u se debe establecer una estructura de la ley de control.

- Ponderaciones entre los términos de los errores ($\delta(k)$) y los de las acciones de control ($\lambda(k)$). En el caso multivariable, además, ponderaciones entre distintas variables. Las ponderaciones se utilizan para conseguir distintos efectos en el comportamiento de bucle cerrado, por ejemplo, que tenga más importancia las acciones de control, o que tengan menos influencia en el índice los valores de predicciones muy alejadas, o ponderar más determinadas variables en detrimento de otras, etc.

- Puede aparecer un además un conjunto de restricciones adicionales sobre variables de entrada, salida e internas. Estas restricciones pueden ser debidas tanto a limitaciones físicas del proceso como a especificaciones de funcionamiento.

- Factores adicionales como el *coste terminal* para asegurar estabilidad (J_e), [67, 19, 93].

1.1.3 Optimizador

Finalmente el control predictivo basado en modelos no es más (ni menos) que un problema de optimización. El objetivo de este elemento está claro, debe conseguir la combinación de acciones de control que optimiza la función de coste. La solución ideal es la que resultaría de la optimización analítica fuera de línea, que implica que se dispone de una expresión matemática que da el óptimo en función de las medidas hasta el instante 't'. Esto se puede conseguir cuando el modelo es lineal y la función de coste cuadrática sin restricciones. En estos casos, el problema de optimización se resuelve fuera de línea.

Fuera de este contexto, es decir, con problemas no lineales (bien sea por los modelos utilizados, bien por la aparición de restricciones, etc.), el control predictivo se convierte en un problema de optimización en cada periodo de muestreo con el consiguiente incremento de complejidad que sólo se justifica si se consiguen mejores prestaciones en el control.

En principio este elemento del control predictivo no aparece como un elemento clave para la consecución de unas buenas prestaciones, más bien queda como una herramienta en segundo plano. Parece más directa la influencia del modelo utilizado: mayor exactitud en el modelo implica mejores predicciones y por tanto un control más exacto. También parece más importante la función de coste, sin una función de coste que valore adecuadamente las predicciones no se va a conseguir un control correcto.

Sin embargo, la herramienta de optimización cobra más protagonismo cuando se pretende aplicar el control predictivo en línea. Por muy bueno que sea el modelo o muy acertada la función de coste, el cuello de botella es la técnica de optimización. **Una selección o ajuste inadecuado de esta técnica puede provocar, en algunos problemas de control, una pérdida notable de las prestaciones del control puesto que no se localiza correctamente el óptimo**. Por tanto la herramienta de optimización debe poseer unas características mínimas que no perturben las prestaciones que se obtienen por la utilización de un modelo y una función de coste determinados.

Puesto que el modelo, restricciones y funciones de coste pueden presentar, en principio, cualquier forma, el optimizador deberá adaptarse a cada problema, en concreto:

- Necesidad de soluciones a funciones y restricciones no convexas que implican la presencia de óptimos locales.
- Que no imponga ninguna condición al problema, por ejemplo que no exija la derivabilidad de la función de coste (la función de coste puede presentar discontinuidades o no linealidades duras).
- Coste computacional bajo. Esta es una característica imprescindible para su aplicación a una amplia gama de procesos.

Desafortunadamente parece que no existe el optimizador ideal, ninguna de las técnicas que se han desarrollado hasta el momento presenta todas las características anteriores simultáneamente [43]. Por tanto la aplicación del control predictivo con altas prestaciones requiere de una etapa de análisis y adaptación de varias técnicas de optimización hasta conseguir la más adecuada para el problema en cuestión.

1.2 Evolución del MBPC

Los conceptos teóricos iniciales que podían asociarse al control predictivo se basan en los trabajos realizados en control óptimo. Utilizando un modelo discreto lineal en espacio de estados se calcula la ley de control de minimizando una función cuadrática de los estados

y las acciones de control (regulador LQR). Se obtiene un controlador que consiste en una realimentación del estado [70] [55]. La optimización se realiza fuera de línea.

Esta técnica tuvo relativamente poco impacto en la tecnología del control de procesos debido a que no contemplaba las no linealidades de los proceso, las restricciones de funcionamiento, incertidumbres en el modelo y sólo se disponía de un índice cuadrático para medir las prestaciones. Únicamente podía aplicarse en áreas donde se podían conseguir modelos muy exactos (por ejemplo, la industria aerospacial).

La industria contribuyó decisivamente en el desarrollo de un control 'óptimo' aplicable, apareciendo los primeros controles predictivos basados en modelos. Se podían utilizar cualquier tipo de modelo y las restricciones de funcionamiento se podían tener en cuenta en la formulación. La optimización se realizaba en cada periodo de muestreo y se utilizaban horizontes de predicción finitos. Sólo se aplica la primera acción de control y se actualiza la información con nuevas medidas.

Las primeras implementaciones a nivel industrial de estos algoritmos se desarrollaron en paralelo IDCOM y el DMC.

- **IDCOM** [82] que es el nombre del software desarrollado a partir del algoritmo conocido como *Model Algorithmic Control* (MAC), también referenciado como *Model Predictive Heuristic Control* (MPHC). Sus características más importantes son:
  - Modelo lineal por respuesta impulsional.
  - Función de coste con un índice cuadrático y horizonte de predicción finito.
  - No hace uso del concepto de horizonte de control.
  - Especificaciones mediante trayectoria de referencia generadas mediante un sistema de primer orden.
  - Se incluye un modelo de perturbaciones:

$$n(t+k|t) = an(t+k-1|t) + (1-a)(y(t) - y_u(t)) \text{ donde } 0 \leq a < 1$$

  - Restricciones en la entrada y la salida incluidas en la formulación.
  - Optimización mediante un algoritmo iterativo.
- **DMC** *Dynamic Matrix Control* [32], sus principales características son:
  - Modelo lineal por respuesta a un escalón.
  - Función de coste con un índice cuadrático de los errores futuros y puede incluir términos referentes a las acciones de control.
  - Horizonte de predicción finito.

- Utiliza un horizonte de control.
- Especificaciones se fija tratando de seguir un referencia de tipo escalón.
- Se incluye un modelo de perturbaciones:

$$n(t+k|t) = y(t) - y_u(t)$$

- La acción de control óptima se obtiene como la solución de un problema de mínimos cuadrados.

A partir de estas dos metodologías se han ido desarrollando, con el tiempo, distintas alternativas tanto a nivel industrial como a nivel teórico, entre las que caben destacar:

- **QDMC** [38]. Basado en el DMC incluye explícitamente las restricciones en las entradas y salidas del proceso. Utiliza programación cuadrática (QP) para la minimización del índice.

- **IDCOM-M** [44], también referenciado como HIECON (*Hierarchical constraint control*). Esta basado en el IDCOM presentando cambios en la función objetivo y en la priorización de las restricciones. Utiliza dos funciones objetivo, primero se optimiza la que evalúa los errores de predicción y si quedan grados de libertad se optimiza respecto a las acciones de control. Para simplificar el proceso se utiliza horizonte de control 1, sólo se permite un cambio en la variable manipulada.

- **GPC** *Generalized Predictive Control* [29] [30].

  - Utiliza modelos basados en funciones de transferencia incluyendo modelo de perturbaciones (modelo CARIMA).
  - El índice es cuadrático e incluye un término de errores de predicción y otro de acciones de control.
  - Utiliza los conceptos de horizonte de predicción y control.
  - La optimización es analítica, por tanto se puede calcular un regulador lineal fuera de línea. Permite implementar un control adaptativo.

  Este controlador predictivo engloba muchos de los últimos avances en el campo de control predictivo, al menos para procesos lineales [28], [34], [35], [51], [1] y [76]. Se pueden tratar algunos tipos de restricciones [23], [53], [101], [15] y se ha planteado el control predictivo adaptativo [27]. Algunas de sus aplicaciones industriales se pueden encontrar en [26], [24], [56], [84].

En el campo industrial las distintas implementaciones que existen actualmente comparten las ideas de los que se han descrito. En [80], [37], [69], [72], [16], [39], [68], [49] y [25] se pueden encontrar descripciones más o menos detalladas del estado actual del control predictivo tanto a nivel tecnológico como a nivel teórico. El número de aplicaciones de este tipo de control crece cada día, algunas de estas se detallan en [4], [31], [61], [81] y [83].

Los últimos avances en esta tecnología apuntan a un control predictivo, que podría tener, entre otras, las siguientes características:

- Utilización de modelos en espacio de estados, permiten trabajar con modelos inestables, multivariables y no lineal con una formulación compacta. En este campo se debe investigar nuevas representaciones de proceso no lineales y como identificarlos a partir de los datos experimentales. Hasta el momento la alternativa más extendida es la de obtener estos modelos no lineales en espacio de estado a partir de principios fundamentales.

- De los últimos avances para asegurar la estabilidad [93], [67] [19] y [20] se intuye la utilización de horizontes de predicción casi-infinitos.

- Aparecen nuevos índices de funcionamiento, índices multiobjetivos, estructuración de la ley de control, inclusión de restricciones, etc. [91], [93] y [11]. En definitiva formas más adecuadas de plantear el problema de control.

- Utilización de otras técnicas de optimización para hacer frente a los nuevos problemas de optimización que se plantean con los modelos no lineales, restricciones de funcionamiento y nuevos índices (problemas no convexos y/o discontinuidades a resolver en tiempo real) [33], [63], [75], [94], [42], [8].

1.3 Optimización

Puesto que parte de esta tesis se centrará en la exploración de técnicas de Optimización Heurística y Monte Carlo para la resolución de problemas de control predictivo basado en modelos, es conveniente presentar un resumen del estado del arte sobre métodos de Optimización poniendo de manifiesto hasta donde puede llegar cada técnica y donde están indicadas o no. El análisis del potencial de alguna de estas técnicas y el conjunto de problemas complejos que pueden presentarse en un CPBM serán la base del planteamiento en el que se fundamentarán los objetivos de la tesis en un punto posterior.

1.3.1 Métodos de optimización clásica

Un problema de optimización [98, 46], en concreto un problema de minimización[1] se puede formular:

Dada una función: $f(x) : R^n \rightarrow R$

Encontrar $\hat{x} \in R^n$, mínimo de la función $f(x)$, *tal que:*

$$\hat{f} := f(\hat{x}) \leq f(x) \; ; \; \forall \, x \in R^n$$

$\hat{f}$ *es lo que se conoce como el* valor mínimo de la función $f(x)$.

$$\hat{f} := \min \{ f(x) \; : \; x \in R^n \}$$

En algunos casos, el interés se centra en encontrar $\hat{f}$, en otros sin embargo, el objetivo es encontrar $\hat{x}$. *Generalmente en las aplicaciones de control predictivo de procesos en las que se van a utilizar el optimizador se busca* $\hat{x}$.

Tal y como se ha formulado se trata de un problema sin restricciones (se busca el óptimo en $x \in R^n$), pero en general un problema de optimización puede venir acompañado de restricciones que se suelen representar mediante funciones de ligadura:

$$g_i(x) : R^n \rightarrow R$$

$$h_j(x) : R^n \rightarrow R$$

que generan el subespacio:

$$X = \{ x : \; x \in R^n, \; g_i(x) = 0, \; i = 1..m_1, \; h_j(x) \leq 0, \; j = 1..m_2 \}$$

O bien de forma más compacta:

$$X = \{ x : \; x \in R^n, \; g(x) = 0, \; h(x) \leq 0 \}$$

Donde:

- $g(x) = (g_1(x), g_2(x), \ldots, g_{m_1}(x))^T$
- $h(x) = (h_1(x), h_2(x), \ldots, h_{m_2}(x))^T$

[1] Un problema de maximización se puede convertir en uno de minimización simplemente cambiando el signo de la función.

Si las restricciones son del tipo $h'_j(x) \geq 0$ se pueden convertir a $h_j(x) = -h'_j(x) \leq 0$

Por tanto el problema de minimización más general se puede formular como sigue:

Dada una función: $f(x) : R^n \rightarrow R$

Encontrar $\hat{x}$ *tal que:*

$$f(\hat{x}) := \min\{f(x) \ : \ x \in X\}$$

Definición 1

$x_o \in R^n$ *es un* **mínimo local** *de* $f(x)$ *si:*

$$\exists \epsilon > 0 \text{ tal que: } \|x - x_o\| < \epsilon \rightarrow f(x_o) \leq f(x)$$

Definición 2

$x_o \in R^n$ *es un* **mínimo global** *de* $f(x)$ *si:*

$$f(x_o) \leq f(x) \ \forall x \in R^n$$

Optimización sin restricciones

Se puede demostrar [98] que un mínimo de una función en un problema de minimización sin restricciones cumple el siguiente teorema:

Teorema 1

$$\left.\begin{array}{l} f(x) \text{ es dos veces diferenciable en el punto } \hat{x} \\ Si \ \ \nabla f(\hat{x}) = 0 \text{ (\textbf{Condición necesaria})} \\ \nabla^2 f(\hat{x}) \text{ definida positiva, (\textbf{Condición suficiente})} \end{array}\right\} \rightarrow \hat{x} \text{ es un mínimo de } f(x)$$

Donde:

- $\nabla f(x)$ *es el gradiente de la función* $f(x)$*:*

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, ..., \frac{\partial f(x)}{\partial x_n}\right)$$

- $\nabla^2 f(x)$ *es la matriz de Hesse (o Hesiano) de la función* $f(x)$*:*

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

Los puntos que cumplen la condición necesaria se denominan **puntos estacionarios**, pueden ser mínimos (si cumplen la condición necesaria), máximos o puntos de silla.

A partir del teorema 1 la solución de un problema de minimización sin restricciones: se formula:

1. *Calcular los puntos estacionarios* $x \in R^n$ *donde* $\nabla f(x) = 0$ *a partir del sistema de ecuaciones:*

$$\begin{aligned} \frac{\partial f(x)}{\partial x_1} &= 0 \\ &\cdots \\ \frac{\partial f(x)}{\partial x_n} &= 0 \end{aligned}$$

2. *Evaluar la condición:* $\nabla^2 f(x)$ *es definida positiva en los puntos anteriores para conocer cuales son mínimos.*

3. *Evaluar f(x) en los puntos de los mínimos para localizar el mínimo global.*

Optimización con restricciones de tipo igualdad

Para resolver un problema de minimización que presenta restricciones de igualdad:

$$\begin{aligned} &\min \{f(x) : x \in X\} \\ &X = \{x : x \in R^n,\ g_i(x) = 0,\ i = 1..m_1\} \end{aligned} \tag{1.7}$$

Existen dos alternativas [98]:

1. **Método del Jacobiano**

   Este método se basa en el cálculo del gradiente restringido de la función $f(x)$, denominado $\nabla_c f(x)$, y que es el *gradiente de* $f(x)$ *en los puntos que satisfacen las restricciones.*

   El método se desarrolla dividiendo x en variables dependientes ($y = (y_1, \ldots, y_{m_1})$) e independientes ($z = (z_1, \ldots, z_{n-m_1})$).

$$x = (y, z)$$

   Se demuestra [98] que:

$$\nabla_c f(x) = \frac{\partial_c f(x)}{\partial z} = \nabla_z f(x) - \nabla_y f(x) J^{-1} C$$

   Donde:

- $\nabla_z f(x) = \left(\frac{\partial f(x)}{\partial z_1}, \ldots, \frac{\partial f(x)}{\partial z_{n-m_1}}\right)$
- $\nabla_y f(x) = \left(\frac{\partial f(x)}{\partial y_1}, \ldots, \frac{\partial f(x)}{\partial y_{m_1}}\right)$
- Matriz Jacobiana:

$$J = \nabla_y g(x) = \begin{pmatrix} \frac{\partial g_1(x)}{\partial y_1} & \frac{\partial g_1(x)}{\partial y_2} & \cdots & \frac{\partial g_1(x)}{\partial y_{m_1}} \\ \frac{\partial g_2(x)}{\partial y_1} & \frac{\partial g_2(x)}{\partial y_2} & \cdots & \frac{\partial g_2(x)}{\partial y_{m_1}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_{m_1}(x)}{\partial y_1} & \frac{\partial g_{m_1}(x)}{\partial y_2} & \cdots & \frac{\partial g_{m_1}(x)}{\partial y_{m_1}} \end{pmatrix}$$

- Matriz de control

$$C = \nabla_z g(x) = \begin{pmatrix} \frac{\partial g_1(x)}{\partial z_1} & \frac{\partial g_1(x)}{\partial z_2} & \cdots & \frac{\partial g_1(x)}{\partial z_{n-m_1}} \\ \frac{\partial g_2(x)}{\partial z_1} & \frac{\partial g_2(x)}{\partial z_2} & \cdots & \frac{\partial g_2(x)}{\partial z_{n-m_1}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_{m_1}(x)}{\partial z_1} & \frac{\partial g_{m_1}(x)}{\partial z_2} & \cdots & \frac{\partial g_{m_1}(x)}{\partial z_{n-m_1}} \end{pmatrix}$$

- La relación entre las variaciones de las variables dependientes respecto de las variables independientes viene dada por:

$$\frac{\partial y}{\partial z} = -J^{-1}C \tag{1.8}$$

A partir de la derivada restringida (aplicando condición necesaria del teorema 1) y las ecuaciones de las restricciones se calcula el punto estacionario (x_o) resolviendo el sistema de ecuaciones:

$$\begin{aligned} \nabla_c f(x) &= 0 \\ g(x) &= 0 \end{aligned}$$

Para evaluar si se trata de un mínimo (condición de suficiencia del teorema 1) se debe comprobar que:

$$\frac{\partial \nabla_c f(x_o)}{\partial z} \text{ es definida positiva}$$

En general es necesario utilizar la expresión 1.8 en los cálculos de esta matriz.

Este método presenta dificultades por la necesidad de invertir una matriz (J^{-1}), lo cual es siempre costoso y en ocasiones puede presentar problemas numéricos.

2. **Método de los multiplicadores de Lagrange**

Se puede demostrar [98] que el problema de minimización con restricciones 1.7, equivale al problema de minimización sin restricciones de la función Lagrangiana $L(x, \lambda)$:

$$\min \left\{ L(x, \lambda) = f(x) + \lambda g(x) \ : \ (x, \lambda) \in R^{n+m_1} \right\}$$

Donde:

- $g(x) = (g_1(x), g_2(x), \ldots, g_{m_1}(x))^T$
- $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{m_1})$ son los multiplicadores de Lagrange. Los multiplicadores de Lagrange se interpretan como *coeficientes de sensibilidad* (evalúan las variaciones de f respecto de las variaciones de g), ver [98]. También se puede interpretar de forma geométrica, se trata de los coeficientes que permiten cancelar vectores gradientes ver [100].

$$\lambda = -\frac{\partial f}{\partial g}$$

A este problema sin restricciones se le puede aplicar el teorema 1 para el cálculo del punto estacionario. El sistema de ecuaciones a resolver que resulta de la condición necesaria es:

$$\begin{aligned} \frac{\partial L(x, \lambda)}{\partial x} &= \nabla f(x) + \lambda \nabla g(x) = 0 \\ \frac{\partial L(x, \lambda)}{\partial \lambda} &= g(x) = 0 \end{aligned}$$

Donde $\nabla g(x) = (\nabla g_1(x), \nabla g_2(x), \ldots, \nabla g_{m_1}(x))^T$.

Para determinar si se trata de un mínimo hay que verificar la condición suficiente del teorema 1.

Optimización con restricciones de tipo desigualdad

Si el problema presenta además restricciones que son desigualdades ($h_j(x) \leq 0$):

$$\begin{aligned} &\min \{ f(x) \ : \ x \in X \} \\ &X = \{ x : \ x \in R^n, \ g_i(x) = 0, \ i = 1..m_1, \ h_j(x) \leq 0, \ j = 1..m_2 \} \end{aligned} \tag{1.9}$$

Se pueden utilizar dos alternativas:

1. **Método de las restricciones activas**

   Este método consiste en:

(a) Resolver el problema teniendo en cuenta únicamente las restricciones de tipo igualdad:

$$\min\{f(x) \,:\, x \in X\}$$
$$X = \{x :\; x \in R^n,\; g_i(x) = 0,\; i = 1..m_1\}$$

por ejemplo utilizando multiplicadores de Lagrange se calcula x_o (mínimo de la función).

(b) Con la solución obtenida evaluar si se satisfacen las restricciones de tipo desigualdad. Si no se satisface alguna de ellas, se nombran como *restricciones activas.*

$$h_j(x_o) > 0, \quad \text{Restricciones activas}$$
$$h_k(x_o) \leq 0,\; k \neq j \;, \text{ Restricciones inactivas}$$

(c) Se vuelve a resolver el problema de optimización incorporando las restricciones de desigualdad activas como si fuesen restricciones de tipo igualdad.

$$\text{Si, } \; h_a(x) \text{ son las restricciones activas}$$

$$\min\{L(x, \lambda, \mu) = f(x) + \lambda g(x) + \mu h_a(x)\}$$

(d) Con la nueva solución se vuelve al paso (b) hasta que se cumplan todas las restricciones simultaneamente.

Este método no asegura la obtención del mínimo, salvo que se evalúen todas las combinaciones de restricciones de tipo desigualdad como si fuesen de tipo igualdad, lo cual incrementa exponencialmente los tiempos de cálculo ver [98].

2. **Variables de holgura y condiciones de Kuhn-Tucker**

Las restricciones de tipo desigualdad se pueden convertir en restricciones de tipo igualdad utilizando variables de holgura (s_j):

$$h_j(x) \leq 0 \;\rightarrow\; h_j(x) + s_j^2 = 0$$

En general, para un conjunto de restricciones de tipo desigualdad:

$$h(x) \leq 0 \;\rightarrow\; h(x) + Ss^T = 0$$

Donde:

- $h(x) = (h_1(x), h_2(x), \ldots, h_{m_2}(x))^T$
- $s = (s_1, s_2, \ldots, s_{m_2})$
- $S = diag\{s\}$

En el caso más general en que aparecen los dos tipos de restricciones se puede convertir el problema con restricciones en uno sin restricciones mediante la utilización de multiplicadores de Lagrange y variables de holgura. El problema:

$$\begin{array}{l}\min\{f(x) \,:\, x \in X\} \\ X = \{x \,:\, x \in R^n,\ g_i(x) = 0,\ i = 1..m_1,\ h_j(x) \leq 0,\ j = 1..m_2\}\end{array}$$

Se convierte en:

$$\min\left\{L(x,\lambda,\nu,s) = f(x) + \lambda g(x) + \nu(h(x) + Ss^T) \,:\, (x,\lambda,\nu,s) \in R^{n+m_1+2m_2}\right\}$$

En este caso el sistemas de ecuaciones a resolver para la localización del punto estacionario es [42]:

$$\begin{array}{rcl}\dfrac{\partial L(x,\lambda,\nu,s)}{\partial x} & = & \nabla f(x) + \lambda \nabla g(x) + \nu \nabla h(x) = 0 \\ \dfrac{\partial L(x,\lambda,\nu,s)}{\partial \lambda} & = & g(x) = 0 \\ \dfrac{\partial L(x,\lambda,\nu,s)}{\partial \nu} & = & h(x) + Ss^T = 0 \\ \dfrac{\partial L(x,\lambda,\nu,s)}{\partial s} & = & SVe^T = 0\end{array}$$

Donde:

- $V = diag\{\nu\}$
- $e = (1, 1, \ldots, 1)$

La resolución de este sistema de ecuaciones sólo establece la condición necesaria, para verificar si se trata de un mínimo se debería evaluar la condición de suficiencia. En el caso en que la solución del sistema de ecuaciones cumpla: $\nu \geq 0$, función objetivo ($f(x)$) y el espacio de soluciones (delimitado por las restricciones) sean convexos, se garantiza que la solución obtenida es un mínimo (condiciones de optimalidad de Kuhn-Tucker, ver [98]).

En resumen, la optimización de una función sin restricciones se resuelve aplicando el teorema 1, a partir de $\nabla f(x) = 0$ (sistema de ecuaciones) se obtienen los posibles mínimos (puntos estacionarios) y verificando que $\nabla^2 f(x)$ es definida positiva para esos puntos, se asegura que se trata de los mínimos.

Cuando el problema presenta restricciones de tipo igualdad: el método del Jacobiano permite obtener un sistema de ecuaciones del que se obtienen puntos que cumplen la condición necesaria ($\nabla_c f(x) = 0$) y la condición suficiente a partir de $\nabla_c^2 f(x)$. El método de los multiplicadores de Lagrange lo convierten en un problema sin restricciones al que se le puede aplicar el teorema 1.

Si aparecen restricciones de tipo desigualdad se dispone de dos alternativas: el método de las restricciones activas o la utilización de variables de holgura, en ambos casos se utilizan los multiplicadores de Lagrange para obtener un problema sin restricciones.

En todos los casos presentados, la obtención de un mínimo pasa por plantear y resolver un sistema de ecuaciones que en general será no lineal, y donde la totalidad o parte de los coeficientes del sistema deben ser obtenidos a partir de las derivadas de primera y segunda especie de las funciones implicadas, que pueden ser difícil o imposible de obtener. Es necesario además la inversión de una matriz, lo cual es siempre dificultoso y costoso en tiempo.

En algunas situaciones particulares, como los conocidos problemas de programación lineal (LP) y cuadrática (QP) con función a minimizar lineal y cuadrática respectivamente y restricciones lineales en ambos casos se pueden encontrar soluciones directas [98]. Sin embargo no es así en el caso de programación no lineal (NLP), donde tanto las funciones como las restricciones pueden ser no lineales.

En estos casos se requiere de soluciones iterativas que plantean, muchas de ellas, una dirección de búsqueda y un tamaño del paso en cada iteración, así como una condición de finalización que asegure las condiciones de suficiencia. En el punto siguiente se describirán estos métodos.

1.3.2 Métodos de optimización iterativos

Como se acaba de decir, un problema de minimización que no tiene una solución analítica fácil de encontrar puede ser resuelto mediante algoritmos iterativos. Estos algoritmos presentan la estructura general que se muestra en la figura 1.3.

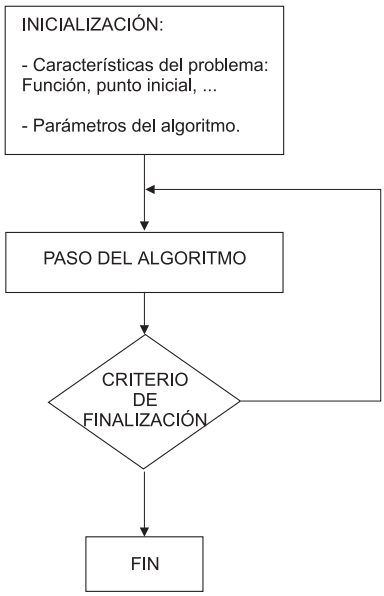


Figura 1.3: Diagrama de flujo global de un algoritmo de optimización iterativo.

Si estos algoritmos consiguen una solución, el problema radica en reconocer si se trata del mínimo global o simplemente se trata de un mínimo local, influyendo en el 'criterio de finalización' y por tanto en la calidad de la solución y el coste computacional. Con métodos iterativos la obtención del mínimo global sólo esta resuelto en el caso en que la función y la región de las restricciones sea convexa[2].

Definición 3

Una función $L(x)$, $x \in R^n$ *es* **Convexa**, *si:*

$$\forall x_1,\ x_2\ R^n;\ \ \forall \lambda \in [0,1]$$

$$L(\lambda x_1 + (1-\lambda)x_2) \leq \lambda L(x_1) + (1-\lambda)L(x_2)$$

Una función convexa diferenciable tiene las siguientes propiedades:

1. $\forall x, h \in R^n\ ,\ \ L(x+h) - L(x) \geq \nabla L(x)^T h$
2. $\nabla^2 L(x)$ *es definida positiva.*
3. $L(x)$ *sólo presenta un mínimo.*

[2]Esta característica es la que se utiliza en las condiciones de Kuhn-Tucker, la solución del sistema de ecuaciones de Kuhn-Tucker asegura el mínimo global si se cumple la convexidad de la función y del conjunto de restricciones.

Minimización de problemas de una variable

Es importante destacar estas técnicas puesto que muchos de los métodos de minimización multivariables necesitan de la resolución de problemas auxiliares que son univariables (obtención de la longitud del paso en la dirección de mejora). En las técnicas que se enumeran a continuación se supone que la función es unimodal. El problema unimodal se formula:

$$\min \{f(x) \, : \, x \in R\}$$

o bien, si existe restricción:

$$\min \{f(x) \, : \, x \in [a, b]\}$$

1. **Métodos de reducción de intervalos.** Asumiendo que la función es unimodal, se puede asegurar que el mínimo se encuentra en un intervalo $[x_k - \Delta, x_k + \Delta]$, $\Delta > 0$ que cumple:

   $$f(x_k - \Delta) \geq f(x_k) \leq f(x_k + \Delta)$$

   Estos métodos empiezan por una fase de acotamiento inicial una posibilidad es la que se presenta en [52]. Los pasos siguientes consisten en ir reduciendo este intervalo inicial sucesivamente. Para ello existe distintas posibilidades, entre las más utilizadas:

   (a) Reducción de la mitad del intervalo en cada paso (*Interval halving*).

   (b) Reducción del intervalo mediante *Golden Ratio*.

   (c) Utilización del los números de Fibonacci para la reducción del intervalo.

2. **Método de las aproximaciones polinomiales.** En cada iteración se aproxima la función en el entorno de un punto a una función polinomial y se calcula el mínimo de esta función. En este caso también aparecen varios métodos:

   (a) Aproximación cuadrática de Powell. Para que converja el método es necesario que la función sea convexa.

   (b) Aproximación cúbica de Davidon. Este método es más eficiente que el de Powell salvo que se tengan que calcular las derivadas de forma discreta.

Minimización de problemas multivariables sin restricciones

Las posibilidades más comunes para resolver el problema son:

1. **Método del gradiente**. El método consiste en localizar en cada iteración un punto (x_k) con un valor de la función menor que el punto de la iteración anterior, siempre en la dirección contraria del gradiente. El criterio de finalización se basa en $\nabla f(x) = 0$ (condición necesaria para un mínimo).

   En definitiva el método se basa en la fórmula recursiva:

   $$x_{k+1} = x_k - r_k \nabla f(x_k)$$

   Donde el valor de r_k se obtiene de la resolución del problema de minimización de univariable:

   $$\min \{h(r) = f(x_k - r\nabla f(x_k)) \ : \ r \in R\}$$

   Este algoritmo se ralentiza según se aproxima al mínimo, y no puede resolver problemas con valles estrechos.

2. **Método de Newton**. Este método se obtiene a partir del desarrollo de Taylor de la función a minimizar. En el entorno de un punto x_k:

   $$f(x) = f(x_k) + \nabla f(x_k)(x - x_k)^T + \frac{1}{2}(x - x_k)\nabla^2 f(x_k)(x - x_k)^T + o(||x - x_k||^2)$$

   Despreciando los términos de orden 3 y superior se obtiene una aproximación cuadrática de $f(x)$ en el entorno del punto x_k:

   $$f_k(x) = f(x_k) + \nabla f(x_k)(x - x_k)^T + \frac{1}{2}(x - x_k)\nabla^2 f(x_k)(x - x_k)^T$$

   Para obtener el punto estacionario de esta función se calcula el gradiente y se iguala a cero:

   $$\nabla f_k(x) = \nabla f(x_k) + (x - x_k)\nabla^2 f(x_k) = 0$$

   El mínimo (si se cumple la condición suficiente, Hessiano definido positivo) se puede obtener analíticamente:

   $$x_{min} = x_k - \nabla f(x_k)(\nabla^2 f(x_k))^{-1}$$

   De esta expresión se deriva directamente la fórmula recursiva para la localización del mínimo de una función:

   $$x_{k+1} = x_k - \nabla f(x_k)(\nabla^2 f(x_k))^{-1}$$

   Para mejorar el acercamiento al mínimo existe una alternativa conocida como **Método de Newton modificado**. Esta modificación utiliza una fórmula recursiva similar pero con un parámetro adicional que se obtiene de la resolución de un problema de minimización univariable:

   $$x_{k+1} = x_k - r_k \nabla f(x_k)(\nabla^2 f(x_k))^{-1}$$

Donde el valor de r_k se obtiene de la resolución de:

$$\min\left\{h(r) = f(x_k - r\nabla f(x_k)(\nabla^2 f(x_k))^{-1}) \ : \ r \in R\right\}$$

Para los métodos de Newton se debe realizar la inversión de una matriz, además se debe garantizar que $\nabla^2 f(x)$ sea definida positiva, en caso contrario el método diverge.

3. **Método Quasi-Newton**. Se trata del método de Newton donde el $H_k^{-1} = \nabla^2 f(x_k))^{-1}$ se obtiene iterativamente a partir de la evaluación exclusiva del gradiente [52]:

$$H_{k+1}^{-1} = H_k^{-1} + U_k$$

La *matriz de corrección*, U_k, se obtiene como función del gradiente, existiendo varias alternativas [52]:

- **DFP** Davidon-Fletcher-Powell.
- **BFGS** Broyden, Fletcher, Goldfarb y Shanno.

4. **Métodos de búsqueda directa**. El método es originalmente debido a *Nelder and Mead* (*Downhill Simplex Method*), y únicamente requiere evaluaciones de la función a minimizar y no sus derivadas. La idea general que subyace en este grupo de métodos es la de partir de un entorno en el que se puede asegurar que está el mínimo e iterativamente reducir el tamaño de dicho entorno [79, 52]. Los métodos de *reducción de intervalos* de una variable forman parte de esta familia de métodos de búsqueda directa.

   Estos métodos no son muy eficientes en cuanto al número de evaluaciones de la función a minimizar [79].

Minimización de problemas multivariables con restricciones

1. **Método de Simplex**. Este método resuelve problemas de programación lineal:

$$\min\left\{f(x) = cx^T \ : \ x \in X\right\}$$

$$X = \left\{x : \ x \in R^n, \ x \geq 0, \ g(x) = Ax^T - b^T = 0\right\}$$

$$c \in R^n \ , \ b \in R^m \ , \ A \in R^{n \times m}$$

   Este algoritmo explora los vértices dentro de la zona de restricciones, se pasa de un vértice a otro que haga menor la función objetivo de manera que iterativamente se llega al mínimo.

2. **Métodos de funciones de penalización y de barrera**, también conocidos como **métodos de punto exterior e interior** respectivamente.

   Esta técnica se basa en transformar, en cada iteración, el problema de minimización con restricciones en uno sin restricciones mediante la incorporación de funciones de penalización. El problema:

   $$\min\{f(x) : x \in X\}$$
   $$X = \{x : x \in R^n,\ g_i(x) = 0,\ i = 1..m\}$$

   Se transforma en:

   $$\min\{\phi(x,r) = f(x) + P(x,r) : x \in R^n\}$$

   Si se utilizan funciones de penalización, las más comunes son:

   $$P(x,r) = r\sum_{i=1}^{m}(g_i(x))^2$$
   $$P(x,r) = r\sum_{i=1}^{m}|g_i(x)|^s,\ s > 0$$

   Si $g_i(x)$ es una restricción de tipo desigualdad se debe reconvertir en otra restricción que cumple:

   $$g_i(x) \le 0 \rightarrow g_i'(x) = \left\{ \begin{array}{ll} 0 & g_i(x) \le 0 \\ g_i(x) & g_i(x) > 0 \end{array} \right\} = 0$$

   Es decir, si un punto x cumple la restricción $(g_i(x) \le 0)$ no introduce penalización $(P(x,r) = 0)$ en la función a minimizar, en caso contrario modifica la función a minimizar en el valor $P(x,r) > 0$.

   Para localizar el mínimo se parte de un punto x_0 y un valor inicial de r_0. De la resolución del problema de minimización sin restricciones se obtiene x_k, que será, junto con un nuevo valor de r_k el punto inicial para la resolución del siguiente problema sin restricciones. El algoritmo se debe detener cuando x_k no varíe demasiado en iteraciones sucesivas. El que toma r_k se define antes de lanzar el algoritmo, y debe cumplir:

   $$\{r_k\} \text{ secuencia tal que: } \lim_{k\to\infty} r_k = \infty$$

   Si las restricciones son de tipo desigualdad $g_i(x) \ge 0$ se pueden utilizar funciones de barrera, las más comunes son:

   $$P(x,r) = \frac{1}{r}\sum_{i=1}^{m}\frac{1}{g_i(x)}$$
   $$P(x,r) = -\frac{1}{r}\sum_{i=1}^{m}\ln g_i(x)$$

En estos casos para localizar el mínimo, el punto inicial x_0 debe ser un punto interior a la zona de restricciones (nunca se debe tomar un punto en la frontera). Se garantiza que la solución del nuevo problema de optimización es también un punto interior, puesto que en las proximidades de las fronteras $\phi(x, r)$ toma valores muy elevados, el problema equivale a la resolución de un problema sin restricciones. Los valores de r_k en cada iteración se toman de la misma forma que en las funciones de penalización.

3. **SQP, programación cuadrática secuencial**. Bajo esta denominación se encuentran un conjunto de técnicas que resuelven el problema de minimización no lineal aproximándolo en cada iteración a un problema de programación cuadrática (QP).

   El problema inicial:

   $$\min\{f(x) : x \in X\}$$
   $$X = \{x : x \in R^n,\ g_i(x) = 0,\ i = 1..m_1,\ h_j(x) \leq 0,\ j = 1..m_2\}$$

   Se resuelve mediante una fórmula iterativa, la forma más común es:

   $$x_{k+1} = x_k + \alpha_k d_k$$

   - Para obtener la dirección de mejora d_k, se hace una aproximación cuadrática de la función y lineal de las restricciones en el punto x_k convirtiéndose en cada iteración k en un problema QP:

     $$\min\left\{f_k(d) = \nabla f(x_k)d^T + \tfrac{1}{2}dHd^T : d \in D\right\}$$
     $$D = \left\{\begin{array}{l} d : d \in R^n \\ \nabla g_i(x_k)d^T + g_i(x_k) = 0,\ i = 1..m_1 \\ \nabla h_j(x_k)d^T + h_j(x_k) \leq 0,\ j = 1..m_2 \end{array}\right\}$$

     Donde H es el Hesiano del Lagrangiano.

   - El valor de H se obtiene recursivamente como en los métodos Quasi-Newton.
   - El valor de la longitud del paso, α_k, se obtiene de una minimización unidimensional, de forma similar al método de Newton modificado.

   Existen otras alternativas bajo la denominación de SQP que incorporan la técnica de punto interior, en cualquier caso se basan en realizar aproximaciones cuadráticas de la función y lineal de las restricciones (problema QP) en cada iteración [42].

1.3.3 Métodos de optimización heurísticos

Las técnicas que se han mostrado hasta el momento presentan limitaciones que para algunos problemas son determinantes. Por ejemplo, los métodos basados en aproximaciones

cuadráticas requieren, para una convergencia adecuada y asegurar la localización del mínimo global, que la función sea convexa y el espacio de soluciones sea un conjunto convexo pues de lo contrario aparecen mínimos locales. Por otro lado, salvo los métodos de búsqueda directa (que son menos eficientes), los métodos descritos requieren que la función y las restricciones sean derivables (que no presente discontinuidades). Por tanto existe un rango de problemas donde los métodos descritos, hasta el momento, no alcanzan una solución satisfactoria.

Para este tipo de problemas (conocidos como problemas de optimización global), se presentan varias alternativas que se pueden enmarcar en el grupo de la optimización heurística. Este tipo de métodos ha abierto un campo muy amplio de investigación dada la enorme importancia que tienen la localización de óptimos en todas las áreas científicas y la complejidad cada vez mayor de los problemas que se pretenden abordar. La lista de métodos es enorme, existen incluso métodos específicos o particularizaciones para problemas concretos (ver [43], [78], [36], [104]). Dos de los más relevantes y que se analizarán con detalle en esta tesis son: *Simulated Annealing* (se enmarca en los métodos de Monte Carlo para optimización) y Algoritmos Genéticos.

Métodos de Monte Carlo

Los métodos Monte Carlo aplicados al área de optimización están asociados a la utilización de números aleatorios en la inspección del espacio de búsqueda para determinar el óptimo global. En [85] aparecen documentados algunos de estos métodos que se resumen a continuación:

- **Búsqueda aleatoria no adaptativa** (*Nonadaptative random search algorithm*). Se trata del algoritmo más simple e intuitivo de los que componen esta familia de métodos. Consiste en:

  1. Generar una conjunto $(x_1, \ldots, x_N)$ de N puntos de forma aleatoria distribuidos uniformemente (se pueden utilizar otros tipos de distribuciones si hay conocimiento a priori de la zona donde se puede encontrar el mínimo) en el espacio de búsqueda X.
  2. Calcular $y_i = f(x_i),\ i = 1, \ldots, N$
  3. La estimación de $min\,\{f(x)\ :\ x \in X\}$ se obtiene de:

     $$min(y_1, \ldots, y_N)$$

- **Algoritmo de arranque múltiple** (*Multistart algorithm*). Consiste en:

  1. Generar un conjunto $(x_1, \ldots, x_N)$ de N puntos de forma aleatoria (se pueden utilizar varios tipos de funciones de distribución) en el espacio de búsqueda X.

2. Se toman estos puntos como puntos iniciales para N ejecuciones de un algoritmo de optimización local (se puede usar por ejemplo el método del gradiente) obteniéndose de cada ejecución un óptimo $(x'_1, \ldots, x'_N)$.
3. Calcular $y_i = f(x'_i),\ i = 1, \ldots, N$
4. La estimación de $min\,\{f(x)\ :\ x \in X\}$ se obtiene de:

$$min(y_1, \ldots, y_N)$$

- **Búsqueda aleatoria adaptativa** (*Adaptative random search algorithm*). En esta familia de métodos de tipo Monte Carlo entran varios algoritmos iterativos. El que describe de forma general este conjunto de algoritmos podría ser el de Matyas (1965) (ver [85]):

  1. Se parte de un punto $x_i \in X$ y se calcula el valor de la función $f(x_i)$.
  2. Se genera un incremento Δx_i aleatoriamente mediante una distribución determinada (densidad de probabilidad $D(\Delta x)$) de media nula.
  3. Se debe asegurar que $x_i + \Delta x_i \in X$ si no se vuelve al paso anterior.
  4. Se salta al siguiente punto:

$$x_{i+1} = \begin{cases} x_i + \Delta x_i & , \ \textbf{Si}\ \ f(x_i + \Delta x_i) < f(x_i) \\ x_i & , \ \text{En los demás casos} \end{cases}$$

  5. Se vuelve al paso 2 si no se cumple el criterio de finalización que se haya establecido.

  La convergencia al optimo global está asegurada (cuando $i \to \infty$). Por ejemplo, para la función $f(x)$ de la figura 1.4, si se parte de un punto x_1 y Δx_1 se toma de una distribución con una función de densidad $D(\Delta x)$. La probabilidad de obtener un punto con $f(x) < f(x_1)$ es la suma de las áreas sombreadas A y B. La probabilidad de que este punto corresponda a un punto del intervalo $[x_1, x_d^a]$ es el área sombreada A. La probabilidad de que este punto corresponda a un punto del intervalo $[x_i^b, x_d^b]$ es el área sombreada b. Mientras no se llega al punto x_2^* existe probabilidad positiva de obtener un punto cada vez más cercano, incluso cuando el algoritmo llega al mínimo local x_1^* la probabilidad de encontrar x_2^* sigue siendo positiva.

- **Simulated Annealing**. Se trata de un algoritmo de búsqueda aleatoria adaptativa más evolucionado y sofisticado. Este algoritmo se basa en una analogía con principios de la termodinámica (o mejor la mecánica estadística), concretamente en la forma en que los líquidos se congelan y cristalizan o los metales se enfrían y templan (*anneal*).

  Las diferencias con el algoritmo de búsqueda aleatoria adaptativa descrito anteriormente, son básicamente dos:

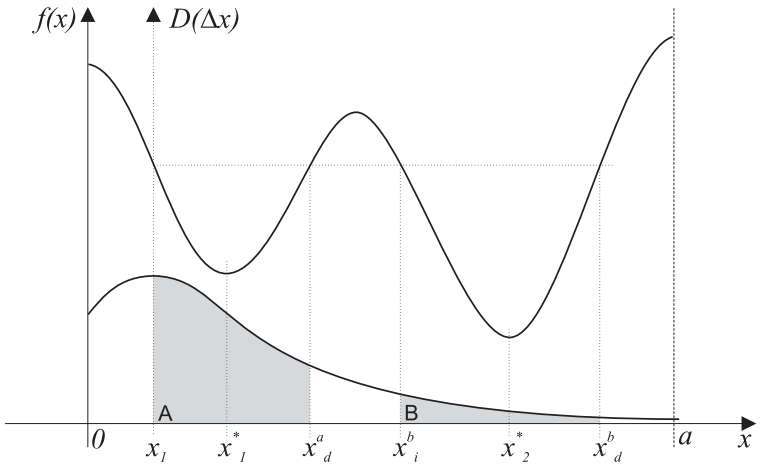


Figura 1.4: Función multimodal $f(x)$ y función de densidad de la distribución $D(\Delta x)$.

- La distribución utilizada para obtener Δx varía con el tiempo, concretamente la desviación de la distribución cambia con un parámetro denominado *temperatura* que es el que varía con el tiempo según lo que se denomina *ley de enfriamiento*.
- La ley de aceptación permite saltar a puntos con valores de la función mayores con una probabilidad que varía con la temperatura.

En definitiva el algoritmo de *Simulated Annealing* básico sería:

1. Se parte de un punto $x_i \in X$ y se calcula el valor de la función $f(x_i)$.
2. Se genera un incremento Δx_i aleatoriamente mediante una distribución determinada (densidad de probabilidad $D(\Delta x)$) de media nula y desviación variable con la temperatura.
3. Se debe asegurar que $x_i + \Delta x_i \in X$ si no se vuelve al paso anterior.
4. Se salta al siguiente punto:

$$x_{i+1} = \begin{cases} x_i + \Delta x_i & , \ \textbf{Si} \ \ f(x_i + \Delta x_i) < f(x_i) \\ x_i + \Delta x_i & , \ \textbf{Si} \ \ f(x_i + \Delta x_i) \geq f(x_i) \\ & \quad \text{con una probabilidad variable según la temperatura} \\ x_i & , \ \text{En los demás casos} \end{cases}$$

5. Se vuelve al paso 2 si no se cumple el criterio de finalización que se haya establecido, y se cambia la temperatura según una ley de enfriamiento.

En el capítulo siguiente se trata con más detalle este algoritmo puesto que se muestra como una alternativa atractiva para la resolución de problemas de minimización

complicados (mínimos locales, discontinuidades, etc.) y únicamente requiere la función objetivo (no se utilizan sus derivadas).

Algoritmos genéticos

Los Algoritmos Genéticos (GA) [40, 66, 47] son técnicas de optimización basadas en simular los fenómenos que se dan en la evolución de las especies adaptándolos a un problema de optimización. En definitiva, estos algoritmos tratan de replicar unas leyes de selección natural que van a afectar a una población hasta conseguir individuos mejor adaptados a su entorno.

La *población* sobre la que opera el GA no es más que un conjunto de puntos del espacio de búsqueda (posibles soluciones al problema de optimización). Lo que se denomina *individuo* es un punto del espacio de búsqueda, cada uno de los individuos diferentes que pueden existir en una población debe tener asociado un *cromosoma* diferente (un código que lo identifica), es decir, debe existir un tipo de codificación para el espacio de soluciones. Dos individuos iguales tienen el mismo cromosoma. El grado de adaptación al entorno y por tanto las posibilidades de supervivencia de un individuo vienen dados por la función objetivo (función que se quiere optimizar). Dicho de otro modo, la función objetivo no es más que el entorno en el que tiene que sobrevivir un individuo, el valor que toma dicha función para un individuo concreto marcará sus posibilidades de supervivencia y reproducción.

Partiendo de una población inicial se van modificando estos individuos (evolución) mediante la aplicación de operadores genéticos:

- *Selección.* Seleccionará de los individuos de la población que formarán parte de la siguiente generación. Los mejor adaptados tienen más probabilidades de seguir en la siguiente generación.
- *Cruce.* Operación con la que se generan nuevos tipos de individuos (cromosomas diferentes) mediante la combinación de los cromosomas de dos individuos. Con este operador se buscan tener individuos todavía mejor adaptados.
- *Mutación.* A partir de un individuo se genera otro diferente mediante la variación aleatoria de alguna de las partes del cromosoma. Este operador permite encontrar cromosomas con buenas cualidades que no existían en la población y que no se podían encontrar mediante cruces.

El algoritmo consiste básicamente en:

1. Generar una población inicial.

2. Evaluar la función a minimizar para los individuos de la población y verificar si esta población cumple con el criterio de finalización que se ha establecido.

3. Generar una nueva población mediante la aplicación de los operadores genéticos: Selección, Cruce y Mutación.

4. Volver al paso 2.

Estas técnicas heurísticas han permitido resolver una gran cantidad de problemas de optimización en todos los campos científicos y técnicos, al igual que el algoritmo de *Simulated Annealing*, se trata de una alternativa que resuelve problemas complejos y que sólo requiere de la función a minimizar (no necesita sus derivadas). Estos algoritmos están sufriendo una evolución continua para conseguir cada vez mejores prestaciones, parece pues que se trata de una opción atractiva para su aplicación al MBPC y por ello se analizarán con más detalle en el capítulo siguiente.

1.4 Objetivos de la tesis

Como ya se ha mostrado existe una gama muy estudiada de algoritmos de optimización cuando tanto la función que se quiere minimizar como el espacio de búsqueda son convexos, y las funciones son continuas. La pregunta que se plantea es si pueden existir problemas no convexos y/o con discontinuidades en control predictivo, donde estas técnicas resultan ineficientes o imposibles de aplicar. En este sentido se plantean algunos ejemplos:

- Cuando se incorporan al modelo no linealidades tales como histéresis, zonas muertas, etc., se obtienen funciones a minimizar como la que se muestra en la figura 1.5 (cambian en cada periodo de muestreo). Se observan discontinuidades importantes (en el ejemplo, estas discontinuidades son debidas al efecto de histéresis) y la función a minimizar no es convexa, no cumple la definición 3.

- Algunos modelos no lineales y/o índices de coste diferentes al cuadrático dan lugar a funciones a minimizar no convexas. Este es el caso del control climático de cultivo de un invernadero[3] como se ve en la figura 1.6 tampoco cumple la definición 3.

Evidentemente la complejidad que incorporan al problema de optimización el planteamiento con modelos no lineales o nuevos índices de coste, sólo esta justificada si se consiguen mejores prestaciones que con planteamientos aproximados pero más simples de

[3]Este problema se trata con detalle en un capítulo de esta tesis

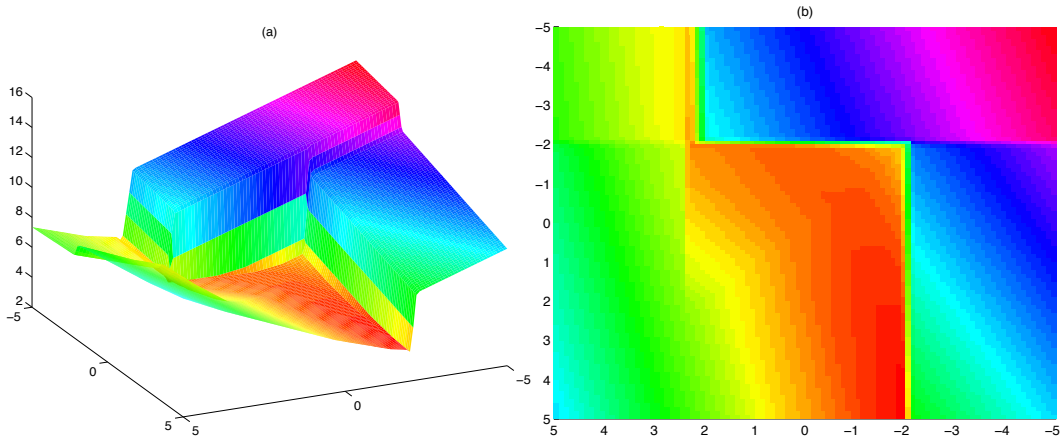


Figura 1.5: Función objetivo resultante en un problema MBPC de un doble integrador con histéresis.

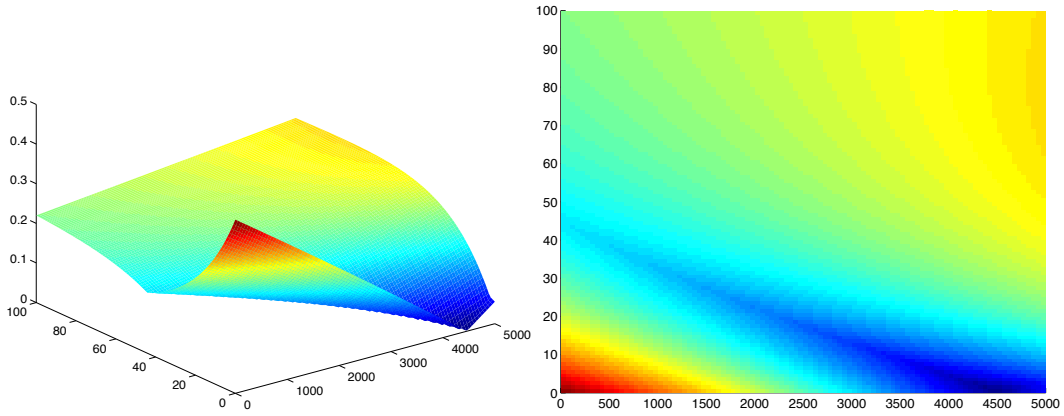


Figura 1.6: Función objetivo resultante en un problema MBPC del control climático de un invernadero.

tratar (por ejemplo, obteniendo modelos aproximados lineales, con funciones de coste cuadráticas, etc.) En cualquier caso se trata de abrir las puertas a la resolución de problemas más complejos que se presentan en aplicaciones reales.

Con estas ideas, este trabajo trata de contribuir al desarrollo del MBPC mediante la utilización de dos técnicas de optimización heurística: algoritmos genéticos (GA) y *simulated annealing* (SA), ya que pueden resolver problemas de minimización más complejos que las técnicas clásicas.

Los objetivos que se plantean se enumeran a continuación y conforman la estructura de los siguientes capítulos de la tesis.

1. **Mostrar el potencial de las técnicas de optimización heurística (GA y SA) y sus limitaciones.**

   (a) Evaluar las características de las versiones actuales de los algoritmos de este tipo.

   (b) Proponer mejoras en los algoritmos, con el objetivo de disminuir el coste computacional sin perder calidad en la solución.

   (c) Obtener una metodología para ajustar los parámetros característicos de los algoritmos asociados a estas técnicas facilitando su aplicación al MBPC.

2. **Adaptar estas técnicas de optimización al problema del MBPC para disponer de un controlador con una estructura lo más flexible posible.**

   El objetivo es poder incluir fácilmente no linealidades y restricciones de funcionamiento en todos los elementos del predictor (modelo del proceso y modelos de perturbaciones), cambiar la función de coste o cambiar el optimizador. En definitiva, permitir aprovechar mejor todo el potencial que presenta el MBPC.

3. **Mostrar qué aporta al MBPC la utilización de optimización heurística:**

   (a) Facilidad para interpretar y resolver nuevos índices de coste que reflejen mejor los objetivos de control.

   (b) Mejoras en el comportamiento del control predictivo con optimización heurística ante modelos que presentan no linealidades en los actuadores, detallando las más usuales como son las saturaciones, zonas muertas, *backlash* e histéresis.

   (c) Aplicabilidad de este tipo de control a modelos no lineales (por ejemplo, con representación en espacio de estados) y las mejoras que aporta.

4. **Extender la aplicación del control predictivo con optimización heurística a sistemas MIMO no lineales.**

5. **Plantear un ejemplo de aplicación con un proceso MIMO no lineal**. Se tratará de controlar el clima de un invernadero para la producción de rosas.

Capítulo 2

Técnicas de optimización heurística

Como ya se ha mencionado uno de los elementos básicos del MBPC es el optimizador. Es sobre este elemento donde recae la responsabilidad de obtener el óptimo de una función en la que se incluye la información disponible del proceso y la forma en que se alcanzan las especificaciones de control. Por ello, tal y como se ha visto en el capítulo anterior, disponer de unas técnicas de optimización adecuadas permitiría abordar problemas de control complejos pudiendo alcanzar además buenas prestaciones.

Se deben tener en cuenta dos aspectos generales en la selección de la técnica de optimización: por una parte el tipo de función que se quiere optimizar y por otra parte los recursos que consume el optimizador.

- En principio, para su aplicación al control de procesos, interesa que el optimizador pueda obtener el mínimo de todo tipo de funciones o al menos de una gran variedad. En el caso del MBPC, si se dispone de un modelo no lineal del proceso y/o con restricciones (saturaciones, zonas muertas, etc.) puede ser conveniente su inclusión en la función objetivo para incrementar las prestaciones del control, y en ese caso esa función puede resultar compleja a la hora de localizar el mínimo. Incluso disponiendo de modelos lineales, puede ser que el índice de coste requiera de una formulación no lineal para obtener una mejora en la calidad del control pero con ello se incrementa la complejidad de la función a minimizar. Las características de estas funciones determinarán, en parte, el tipo de técnica de minimización adecuada.

- Desde el punto de vista de la implementación de un sistema de control con alguna herramienta de optimización, interesa que no consuma excesivos recursos: tiempo, memoria, cpu, etc. Las limitaciones en cuanto a los recursos vendrá impuesta por el problema de control concreto. Periodos de muestreo elevados y/o procesadores muy rápidos facilitarán la aplicación de técnicas de optimización más sofisticadas.

Como ya se ha mencionado en el capítulo anterior las técnicas de optimización heurísticas son la alternativa a los métodos 'clásicos' cuando el problema se complica. En este conjunto de técnicas de optimización heurística podemos encontrar una amplia y variada gama de métodos [36, 78, 43]. De todas ellas caben destacar los métodos de *Simulated Annealing* (SA) y Algoritmos Genéticos (GA) por su amplia utilización en casi todos los campos científicos y tecnológicos. Se trata por tanto de dos metodologías que presentan un estado de desarrollo elevado. Estas técnicas, además, pueden resolver problemas con variables reales, frente a otras técnicas heurísticas que actualmente se restringen a problemas de optimización combinatoria. Son pues dos alternativas posibles para su aplicación al MBPC que, aunque no son las únicas, actualmente parecen las más desarrollas.

En este capítulo se realiza un análisis de todos los elementos que componen estas dos técnicas de optimización heurística: *Simulated Annealing* y Algoritmos Genéticos.

1. En el caso del algoritmo de *Simulated Annealing*, además del análisis de sus componentes, se propone una nueva alternativa (ASA) en la que se cambia la fase de agitación térmica, se seleccionan todos los nuevos estados simultáneamente y se toma el que presente menor energía para continuar el algoritmo. Esta nueva propuesta muestra menor coste y mayor calidad que los algoritmos de *Simulated Annealing* clásico y rápido.

2. En el caso de los Algoritmos Genéticos se realiza un análisis de dos codificaciones diferentes (binaria y real) y varios operadores genéticos englobados en tres grupos: operadores de selección, reproducción y mutación, todo ello para cada una de las codificaciones. Con toda esta información se llega a un conocimiento imprescindible para plantear unas combinaciones de operadores genéticos adecuadas que consigan una buena relación calidad/coste computacional. Todo ello con vistas a su aplicación a un entorno de control predictivo en tiempo real.

Estas técnicas de optimización tienen un coste computacional no despreciable que se debe tener en cuenta si se quiere aplicar al control de un proceso. En ese sentido, todas las modificaciones de estos algoritmos y conocimientos acerca del ajuste de sus parámetros, que permitan disminuir el coste, manteniendo la calidad de la solución, facilitan su aplicación al control. Debido a la envergadura de esta tarea, la evaluación de estos algoritmos se realizará en el capítulo siguiente.

2.1 Simulated Annealing (SA)

El algoritmo de *Simulated Annealing* ([50]) es otra de las técnicas de optimización heurística que permite resolver problemas de optimización complicados (con varios óptimos, funciones no derivables, funciones con restricciones, etc.).

Este algoritmo se basa en una analogía con principios de la termodinámica (o mejor la mecánica estadística), concretamente en la forma en que los líquidos se congelan y cristalizan o los metales se enfrían y templan (*anneal*). Con altas temperaturas las moléculas se mueven libremente unas respecto de otras (agitación térmica) y consiguen alcanzar su posición de mínima energía (para una temperatura dada). Si se enfría el líquido lentamente se va perdiendo gradualmente la movilidad térmica pero las moléculas van alcanzando posiciones de mínima energía sucesivamente. El proceso acaba cuando se alcanza una temperatura suficientemente baja (y solidifica) con lo que se obtiene una cristalización de mínima energía. Si el proceso de enfriamiento es muy rápido se llega a cristales amorfos de estados energéticos mayores.

La esencia del proceso está en un enfriamiento muy lento que de tiempo a alcanzar estados de mínima energía.

En que consiste la analogía:

- La función de coste, $f(x)$, correspondería a la función de energía, evalúa el estado energético.
- Cada punto del espacio de búsqueda, x, corresponde a una de las posibles posiciones de las moléculas.

Se parte de una temperatura inicial elevada, $T_i = T_o$, y un valor inicial del espacio de búsqueda, $x_i = x_o$ (corresponde a una posición de la molécula)[1].

El movimiento debido a la agitación térmica debe resultar en un recorrido por distintos puntos del espacio de búsqueda de una forma aleatoria. Por ejemplo se simula tomando un punto aleatorio del espacio de búsqueda, x_{i+1}, este salto a un nuevo punto se realiza con una determinada distribución. Existen algoritmos de *Simulated Annealing* que utilizan distintos tipos de distribuciones como se verá más adelante.

Este nuevo punto tendrá un valor de la función objetivo (estado energético de la molécula).

Se toma este punto como nuevo punto de partida si el valor de la función objetivo es menor que el anterior (las moléculas deben tender a posiciones de más baja energía). Si el

[1]Los subíndices 'i' corresponden a instante de tiempo o iteración en que se encuentra el algoritmo. Los subíndices 'o' corresponden al instante inicial.

valor de la función objetivo es mayor, se toma como punto de partida con una determinada probabilidad que va a depender de la temperatura en ese instante (en los movimientos provocados por la agitación térmica algunas de las moléculas pueden pasar a estados energéticos superiores) y del valor del incremento en la función de coste $(f(x_{i+1}) - f(x_i))$. Existen distintas alternativas para determinar la probabilidad de aceptación dando lugar a distintas formas del algoritmo.

Por último el algoritmo debe establecer una ley de disminución para el parámetro T_i, esto correspondería a la *ley de enfriamiento*.

La figura 2.1 representa, de forma esquemática, la estructura general de un algoritmo de *Simulated Annealing*.

En la bibliografía se describen diferentes implementaciones de los algoritmos de *Simulated Annealing* (SA clásico y SA rápido) que se diferencian en tres aspectos:

- La función de distribución utilizada en la agitación térmica.
- La ley de aceptación de un nuevo punto cuando presenta un 'estado energético peor' (un valor de la función objetivo peor).
- En el tipo de ley de enfriamiento.

2.1.1 Funciones de distribución

Su utilidad es recorrer el espacio de búsqueda durante el proceso de agitación térmica. Una característica necesaria es que la distribución sea lo más extensa posible para que se puedan salvar los mínimos locales. En este sentido se puede utilizar una distribución uniforme sobre el espacio de búsqueda.

Aunque es importante esta capacidad de exploración no lo es menos que el algoritmo converja y para ello se requiere que la distribución explore de manera exhaustiva las zonas cercanas al punto del que se parte. Esto permite que el algoritmo se vaya acercando con mayor precisión al óptimo. Utilizando una distribución normal se puede conseguir este efecto sin perder la posibilidad de explorar todo el espacio de búsqueda. La función de densidad de la distribución normal (Gauss) de media nula corresponde a (para una sola dimensión):

$$g(x) = exp\left(-\frac{x^2}{\sigma^2}\right) \tag{2.1}$$

Una tercera posibilidad es utilizar la distribución de Cauchy, su función de densidad corresponde a:

$$g(x) = \frac{\sigma}{\sigma^2 + x^2} \tag{2.2}$$

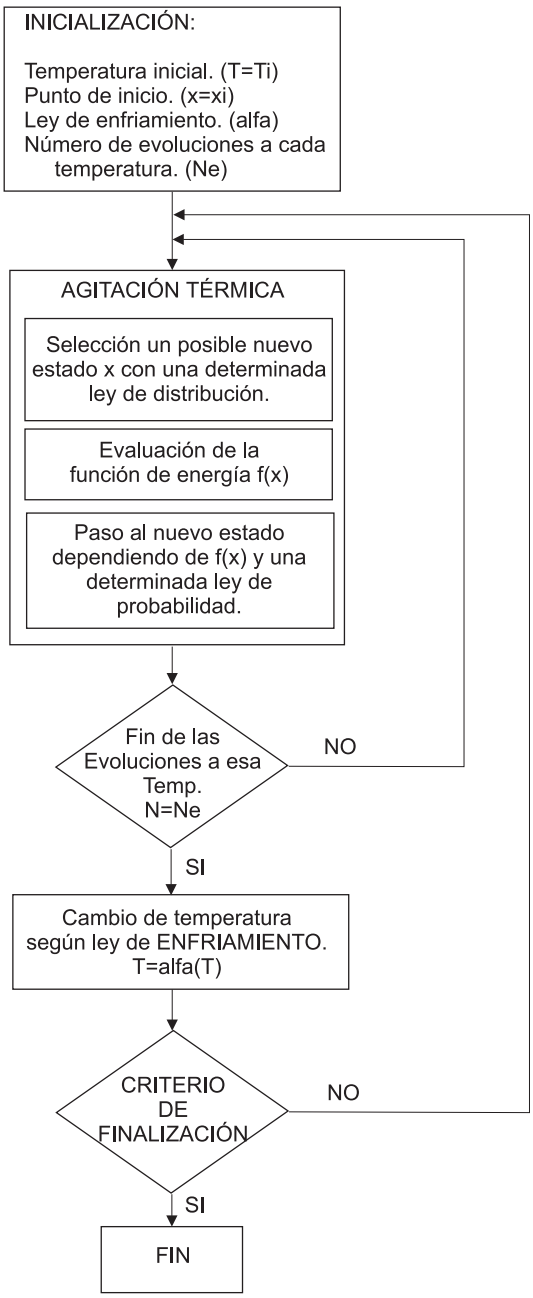


Figura 2.1: Diagrama de flujo de un algoritmo de *Simulated Annealing*.

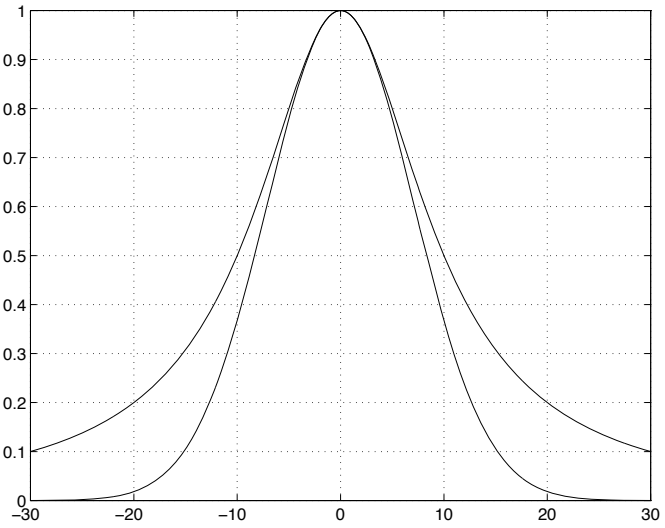


Figura 2.2: Función de densidad normal y de Cauchy con varianza 10.

Esta distribución es similar a la normal (figura 2.2) pero tiene una mayor probabilidad de explorar puntos alejados, por tanto comparte dos cualidades: la exploración exhaustiva del entorno cercano y una exploración de puntos alejados.

Estos efectos se pueden observar mejor en la figura 2.3. Muestra como se distribuyen 200 puntos con una distribución uniforme, normal y de Cauchy respectivamente en un espacio bidimensional. El punto central de las distribuciones es el $(0, 0)$ y la varianza para las distribuciones normal y de Cauchy es $\sigma = 0.5$.

En algunos algoritmos de *Simulated Annealing* se varía la varianza de la distribución en el tiempo o la temperatura de manera que se favorece la convergencia. Las expresiones que se utilizan para controlar esta disminución pueden ser:

$$\sigma(t) = \sigma_o \beta^t \tag{2.3}$$

$$\sigma(t) = \sigma_o \beta^{T(t)} \tag{2.4}$$

Otro aspecto a tener en cuenta cuando se genera un número aleatorio con alguna de las distribuciones descritas es la posibilidad de que se salga del espacio de búsqueda. Se debe plantear algún mecanismo para reinsertar el punto en la zona de búsqueda. Para ellos existen varias posibilidades:

1. Ignorar los puntos que se salen y generar otro.

2. Llevar los puntos que se salen al límite de la zona de búsqueda más cercana.

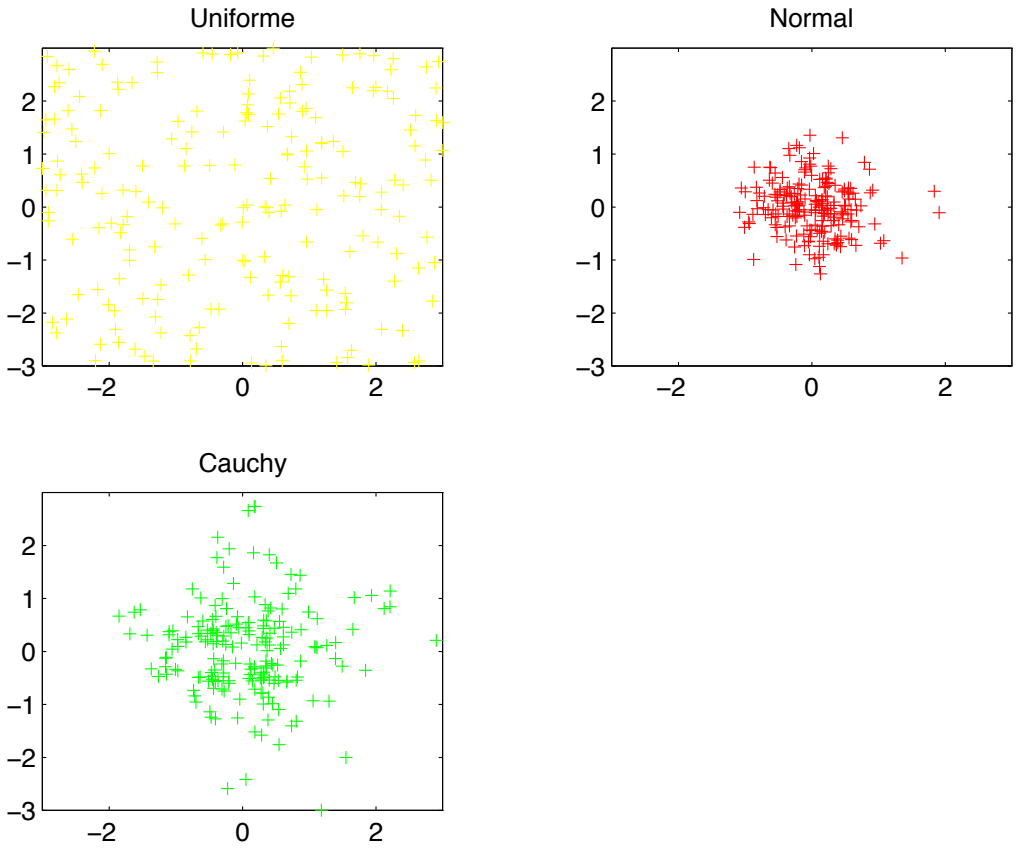


Figura 2.3: 200 puntos generados con distribución uniforme, normal y Cauchy en un espacio bidimensional.

3. Eliminar las zonas exteriores, hacer que cada extremo de la zona toque al extremo opuesto. Cuando un punto se sale de la zona se reinserta por el extremo opuesto.

En la figura 2.4 se presenta un ejemplo de las dos últimas variantes descritas para espacios bidimensionales. Se generan 150 puntos con una distribución normal de varianza $\sigma = 1$, el punto central de la distribución está marcado con un círculo. El espacio de búsqueda es $[-5, 5] \times [-5, 5]$. Con el tercer método se observa una mayor exploración de zonas muy alejadas del punto central (figura 2.4(B)), de esta forma se utilizan los puntos que se salen para incrementar la capacidad de exploración. Con el segundo método (ver figura 2.4(A)) los puntos que se salen del intervalo de búsqueda se usan para explorar los límites de dicho intervalo.

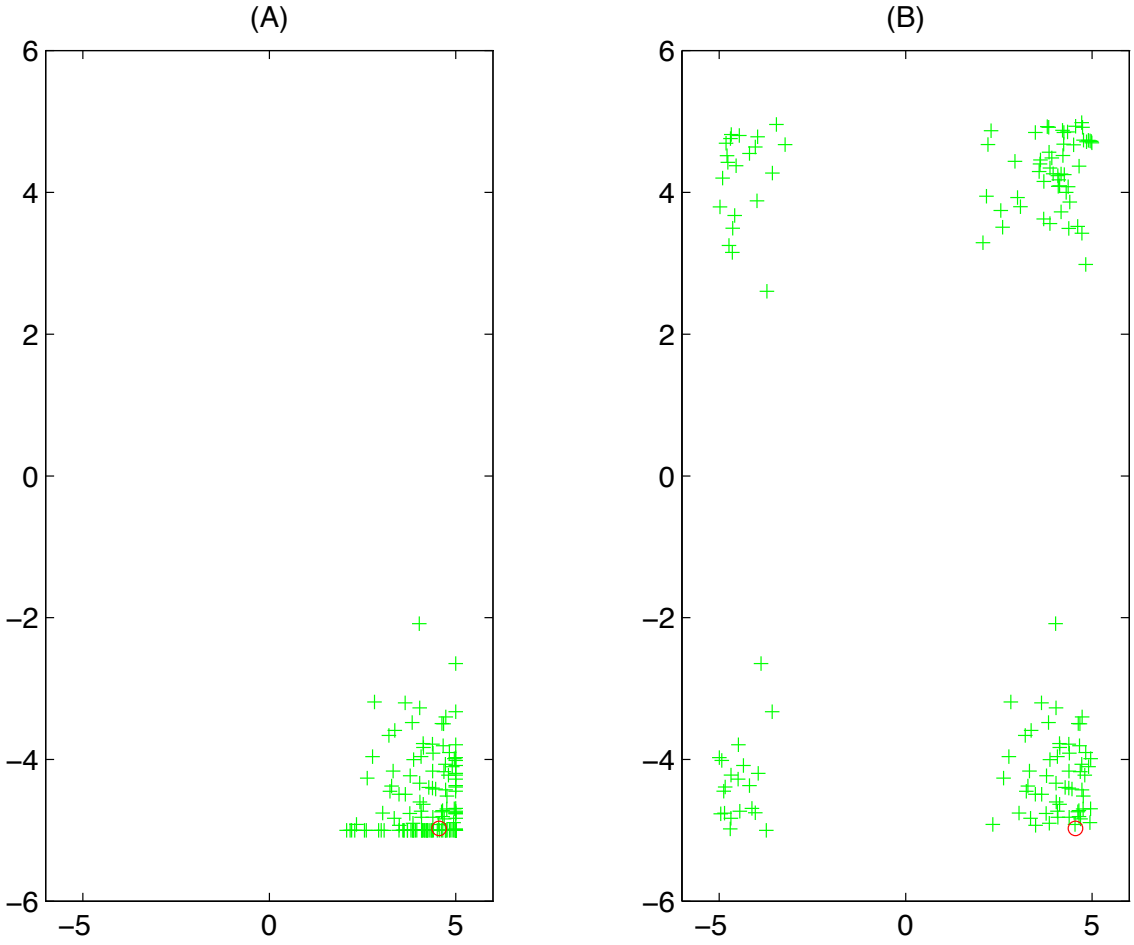


Figura 2.4: (A) Lleva los puntos a los límites de la zona de búsqueda. (B) Eliminación de las zonas exteriores, los puntos que se salen del espacio de búsqueda se reinsertan por el otro extremo.

2.1.2 Leyes de aceptación

El siguiente punto a tratar es la estructura de las leyes de aceptación de los nuevos puntos que se recorren durante la agitación térmica. Una característica general de todas las implementaciones de SA es que si el valor de la función objetivo es mejor que el del punto anteriormente aceptado se pasa a ese valor. Esto permite mejorar gradualmente el valor de la solución. En cuanto a los puntos que presentan un valor peor, se debe establecer algún mecanismo para su aceptación. Si no se aceptasen valores peores el algoritmo correría el peligro de converger, casi siempre, a un óptimo local. Por tanto, esta característica de aceptación marcará enormemente la capacidad de superar óptimos locales.

Una ley de aceptación que se utiliza en muchos casos es la que corresponde a la distribución de Boltzman [50]:

$$P_a = e^{-\frac{f(x_{i+1})-f(x_i)}{T_i}} \tag{2.5}$$

La probabilidad de aceptación (P_a) de un nuevo punto x_{i+1} depende de la temperatura en ese instante T_i y la diferencia del valor de la función objetivo entre el punto actual $f(x_i)$ y el nuevo punto $f(x_{i+1})$. Valores muy elevados de la temperatura incrementan la posibilidad de aceptación. Lo mismo sucede con valores bajos de $\Delta f_{i+1} = f(x_{i+1}) - f(x_i)$.

$$\frac{\Delta f_{i+1}}{T_i} = \frac{f(x_{i+1}) - f(x_i)}{T_i} \Uparrow \rightsquigarrow P_a \approx 0$$

$$\frac{\Delta f_{i+1}}{T_i} = \frac{f(x_{i+1}) - f(x_i)}{T_i} \Downarrow \rightsquigarrow P_a \approx 1$$

Para esta ley de aceptación la probabilidad P_a se mueve en el intervalo $[0, 1]$.

Otra posible ley de aceptación ([97]) es la que viene dada por la expresión:

$$P_a = \frac{1}{1 + e^{\frac{f(x_{i+1})-f(x_i)}{T_i}}} \tag{2.6}$$

En este caso el comportamiento corresponde a:

$$\frac{\Delta f_{i+1}}{T_i} = \frac{f(x_{i+1}) - f(x_i)}{T_i} \Uparrow \rightsquigarrow P_a \approx 0$$

$$\frac{\Delta f_{i+1}}{T_i} = \frac{f(x_{i+1}) - f(x_i)}{T_i} \Downarrow \rightsquigarrow P_a \approx 0.5$$

Se reduce el rango de variación de la probabilidad de aceptación al intervalo $P_a \subset [0, 0.5]$ se trata de un criterio de aceptación más conservador.

Básicamente estas dos leyes de aceptación tienen una forma similar aunque a distinta escala, esto se ve mejor en la figura 2.5. Cabe la posibilidad de diseñar leyes de aceptación

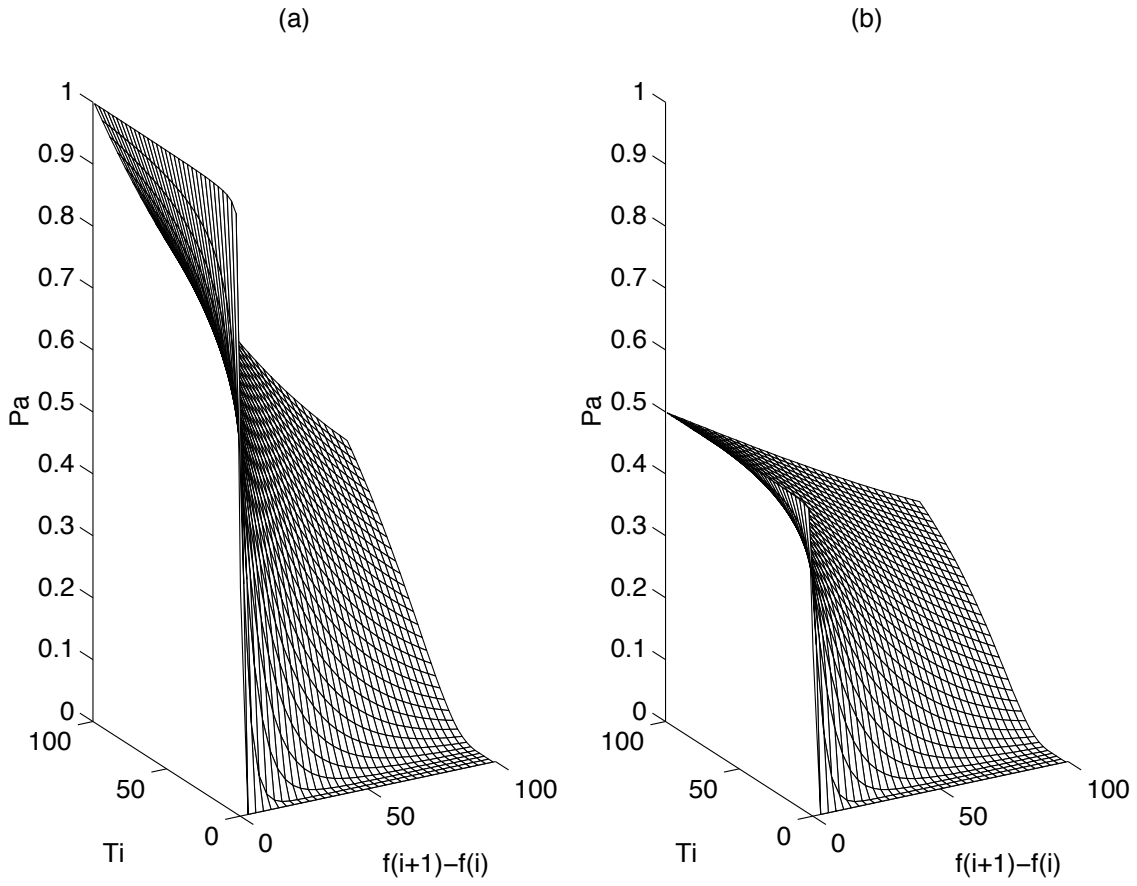


Figura 2.5: Probabilidad de aceptación en función de la temperatura T_i y el incremento de la función de coste $f(x_{i+1}) - f(x_i)$. (a) Ley de aceptación $P_a = e^{-\frac{f(x_{i+1})-f(x_i)}{T_i}}$. (B) Ley de aceptación $P_a = \frac{1}{1+e^{\frac{f(x_{i+1})-f(x_i)}{T_i}}}$.

diferentes, funciones de las variables T_i y Δf_{i+1} que den un valor en el intervalo $[0,1]$. Estas leyes pueden perseguir diferentes objetivos por ejemplo, disminuir la probabilidad de aceptación si Δf_{i+1} es muy pequeña[2], o reducir el coste computacional diseñando leyes que no incluyan exponenciales ni operaciones costosas.

2.1.3 Curva de enfriamiento

El siguiente aspecto importante en la implementación de un SA es la curva de enfriamiento, es decir, cómo se decrementa la temperatura con el tiempo. La evolución de la temperatura afecta tanto a la varianza de la distribución durante la agitación térmica como a la probabilidad de aceptación de nuevos puntos.

Curvas de enfriamiento comunes en la bibliografía ([97], [50]) son:

$$T(t) = \frac{T_o}{1 + log(t+1)} \tag{2.7}$$

$$T(t) = \frac{T_o}{1+t} \tag{2.8}$$

$$T(t) = T_o \alpha^t \ , \ 0 < \alpha < 1 \tag{2.9}$$

T_o es la temperatura inicial y t el tiempo transcurrido. Para una evolución discreta en el tiempo las expresiones se pueden reescribir como (i y T corresponden al instante y periodo de muestreo respectivamente):

$$T_i = \frac{T_o}{1 + log(iT+1)}$$

$$T_i = \frac{T_o}{1+iT}$$

$$T_i = T_o \alpha^{iT} \ , \ 0 < \alpha < 1$$

La diferencia entre estas curvas de enfriamientos se puede apreciar en la figura 2.6. La curva (a) tiene un enfriamiento inicial muy rápido, pero tarda mucho más que las demás en enfriarse del todo, apenas a rebasado el 20% de valor final cuando las demás curvas se sitúan en valores muy bajos (inferiores al 2%). Si se utiliza está curva de enfriamiento se mantiene una alta temperatura lo que influye en un convergencia más lenta del algoritmo aunque esto puede beneficiar en algunos casos una exploración más exhaustiva del espacio. La curva (b) tiene un comportamiento similar al de (a) aunque a distinta escala, se enfría muy rápido al principio y mantiene un enfriamiento muy lento al final aunque los valores son muy inferiores a los de (a). En cuanto a las curvas (c) y (d) tienen enfriamientos

[2]En algunos caso no tiene sentido aceptar valores ligeramente superiores de $f(x_{i+1})$.

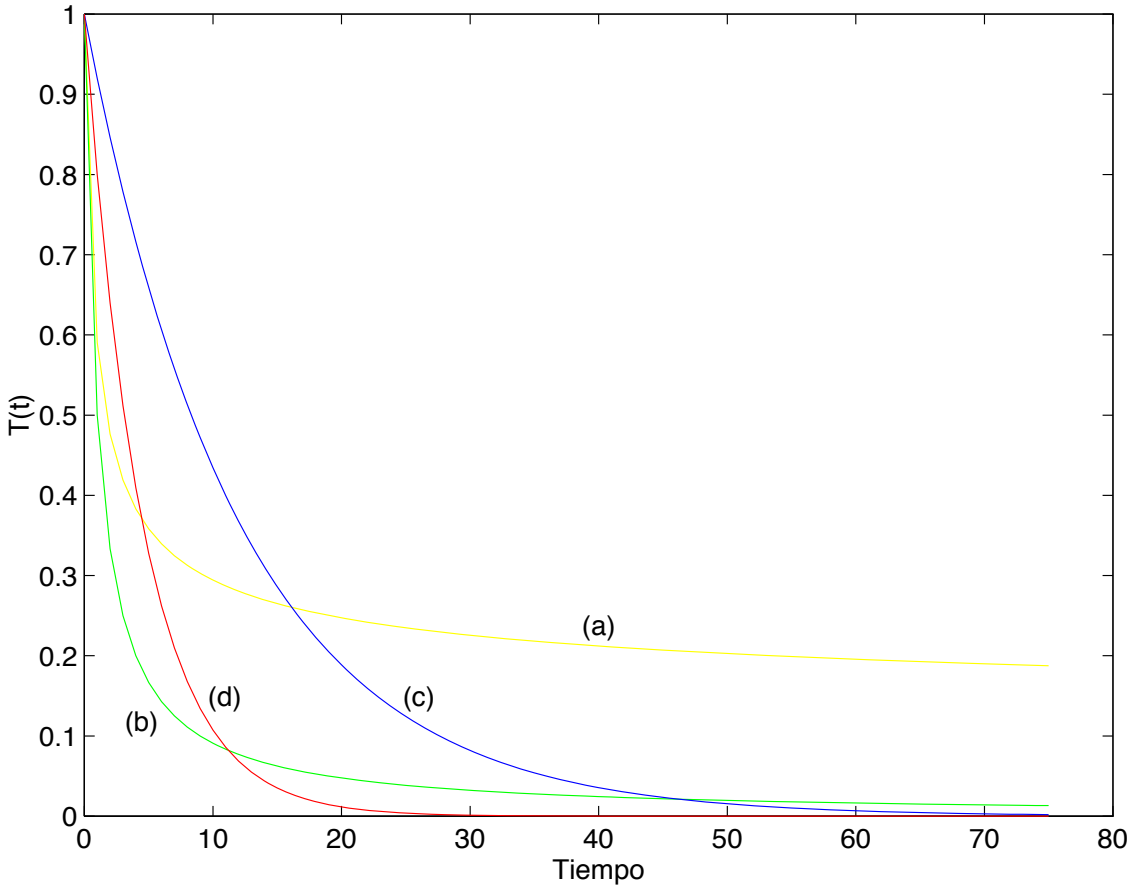


Figura 2.6: Curvas de enfriamiento $T_o = 1$: (a) $T(t) = \frac{T_o}{1+log(t+1)}$; (b) $T(t) = \frac{T_o}{1+t}$; (c) $T(t) = T_o 0.92^t$; (d) $T(t) = T_o 0.8^t$.

iniciales más lentos y durante más tiempo alcanzado valores muy bajos de temperatura. Una ventaja de estas últimas curvas es que son computacionalmente menos costosas de calcular.

2.1.4 Algoritmos de *Simulated Annealing*

En este apartado se presentan tres implementaciones del algoritmo de SA, las dos primeras corresponden a algoritmos usuales en la bibliografía [50, 97], la tercera es una nueva alternativa para mejorar tanto los recursos consumidos como la calidad de la solución obtenida. Estos tres algoritmos son:

1. **CSA**, *Simulated Annealing clásico*: Para la selección de los puntos durante la agitación térmica se utiliza una distribución normal y la ley de aceptación corresponde a la ecuación 2.5. La varianza de la distribución normal disminuye con el tiempo según la expresión:

   $$\sigma(t) = \sigma_o \cdot \beta^t \tag{2.10}$$

   Donde σ_o es la varianza inicial y β es un factor que controla la disminución de la varianza con el tiempo. La curva de enfriamiento que se utiliza es la correspondiente a la ecuación 2.7.

2. **FSA**, *Simulated Annealing Rápido*: En la agitación térmica se utiliza una distribución de Cauchy con varianza variable con el tiempo según la ecuación 2.10. La ley de aceptación es la de la expresión 2.6, y la curva de enfriamiento correspondiente a 2.8.

3. **ASA**, *Simulated Annealing Alternativo*: En esta propuesta se varía el proceso en la fase de agitación térmica, en lugar de generar un punto y evaluar si se acepta o no cada vez, se elige un conjunto de puntos con una distribución de Cauchy de los cuales se toma el mejor para evaluar si se acepta o no. Esta propuesta trata de mejorar la convergencia del algoritmo y no incrementa el coste computacional.

   La ley de aceptación es la de la expresión 2.6 con la varianza dada por la expresión 2.10. La curva de enfriamiento es la correspondiente a 2.8 o 2.9 (se han probado las dos versiones).

Para mostrar las diferencias en el comportamiento de estos algoritmos, a continuación se analizan los resultados obtenidos con dos problemas de optimización, el primero consiste en un problema univariable con mínimos locales y el segundo es una función de dos variables sin mínimos locales (función de Rosenbrock). Hay que destacar que las conclusiones se restringen a los problemas concretos que se han estudiado, aunque estos aparecen en la bibliografía como funciones válidas para la comparación de algoritmos de

optimización [97, 100], se debería realizar un estudio más amplio para comprobar si estas características se mantienen con otro tipo de problemas. En cualquier caso se ve en ambos ejemplos que el algoritmo propuesto (ASA) apunta buenas prestaciones aunque quedaría por resolver cómo realizar el ajuste de sus parámetros.

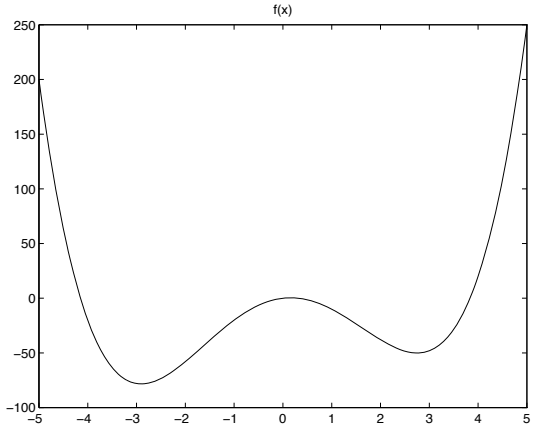
Ejemplo de minimización de un problema univariable

En este caso se utiliza la función que aparece en el artículo *Fast simulated annealing* [97] donde se compara el CSA y FSA. La función que se utiliza para la comparación es la siguiente:

$$f(x) = x^4 - 16x^2 + 5x$$

La función presenta dos mínimos:

$$x = -2.9035 \quad \rightarrow \quad f(x) = -78.3323$$
$$x = 2.7468 \quad \rightarrow \quad f(x) = -50.0589$$



El espacio de búsqueda en todas las ejecuciones es $x \in [-20,\ 20]$ (en la figura sólo se muestra el intervalo $[-5,\ 5]$, zona en que se encuentran el óptimo local y global), los puntos que se salen del espacio de búsqueda se reinsertan suponiendo que no existen zonas exteriores (ver apartado 2.1.1).

En la agitación térmica se producen 15 saltos a distintos puntos sucesivamente (en los algoritmos CSA y FSA), para el algoritmo ASA se evalúan 15 puntos para determinar el mejor de ellos. La varianza de las distribuciones sigue la siguiente expresión 2.10 con los siguientes parámetros:

$$\sigma_o = \frac{\text{Rango del espacio de búsqueda}}{fstdi}$$

$$\beta = 0.9965$$

En este caso el rango del espacio de búsqueda es de 40 y se toma como factor inicial $fstdi = 12$, la varianza inicial es de $\sigma_o = 40/12 = 3.33$.

La temperatura inicial en todos los algoritmos es de $T_{ini} = 450$ y el parámetro de temperatura final es $T_{fin} = 10^{-8}$ (poner 0 provoca problemas numéricos).

Todos los parámetros: β, $fstdi$, T_{ini} y T_{fin} se han elegido experimentalmente de manera que los resultados, para el algoritmo CSA, sean similares a los que se presentan en [97].

La detención del algoritmo se realiza por temperatura, se para el algoritmo cuando la temperatura sea inferior a un valor determinado, este procedimiento exige ajustar una temperatura de detención diferente para cada ley de enfriamiento puesto que su evolución es diferente. Se intenta que la detención del algoritmo no sea antes de la convergencia del mismo pero que se realice en un tiempo razonable (que no tarde demasiado). Con estas ideas la finalización de cada algoritmo viene dada por:

(a) Algoritmo CSA. Finaliza la ejecución cuando la temperatura es inferior a $T_{fin} + 0.12 \cdot T_{ini}$ (temperatura final más un 12 % de la temperatura inicial).

(b) Algoritmo FSA. Finaliza la ejecución cuando la temperatura es inferior a $T_{fin} + 0.001 \cdot T_{ini}$ (temperatura final más un 0.1% de la temperatura inicial).

(c) Algoritmo ASA con la ley de enfriamiento 2.8. Finaliza la ejecución cuando la temperatura es inferior a $T_{fin} + 0.001 \cdot T_{ini}$ (temperatura final más un 0.1% de la temperatura inicial).

(d) Algoritmo ASA con ley de enfriamiento 2.9, $\alpha = 0.92$. Finaliza la ejecución cuando la temperatura es inferior a T_{fin}.

De los resultados obtenidos (ver figura2.7) se deduce que:

(a) El algoritmo clásico CSA converge de forma muy lenta y con mucha variabilidad, incluso durante un tiempo se queda atrapado en el mínimo local, se trata del algoritmo que presenta peores prestaciones de los cuatro analizados.

(b) El algoritmo FSA converge bastante rápido al mínimo global con una pequeña variabilidad. No se queda atrapado en ningún momento en el mínimo local.

(c) El comportamiento del algoritmo ASA que se ve en (c) es muy similar al del algoritmo FSA aunque con una variabilidad ligeramente menor.

(d) El algoritmo que presenta las mejores prestaciones es el ASA que se ve en (d), que además tiene menor coste computacional que el (c).

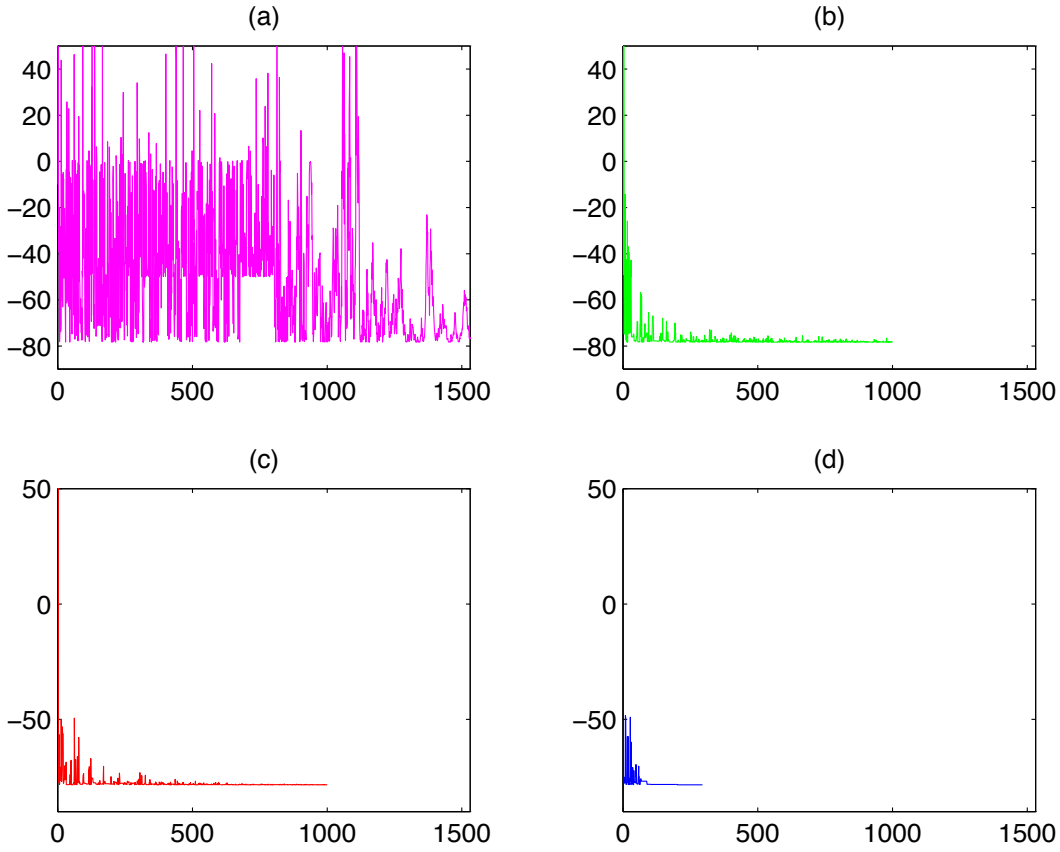


Figura 2.7: Evolución del valor de $f(x)$ después de cada cambio de temperatura. (a) Algoritmo CSA. (b) Algoritmo FSA. (c) Algoritmo ASA con la ley de enfriamiento $T(t) = \frac{T_o}{1+t}$. (d) Algoritmo ASA con la ley de enfriamiento de la ecuación $T(t) = T_o\alpha^t$, $\alpha = 0.92$.

Ejemplo de minimización de un problema con dos variables

Para la comparación de las distintas variantes de *Simulated Annealing* en una función de dos variables se utiliza la función de Rosenbrock (ver anexo B, función 'f2'), con la que se evalúan distintos algoritmos de optimización en [100]. Esta función aunque es unimodal presenta complejidad suficiente para servir de problema en la evaluaciones de algoritmos de optimización [40, 104, 100].

La estructura y valor de los parámetros de cada una de las variantes es la misma que en el problema anterior.

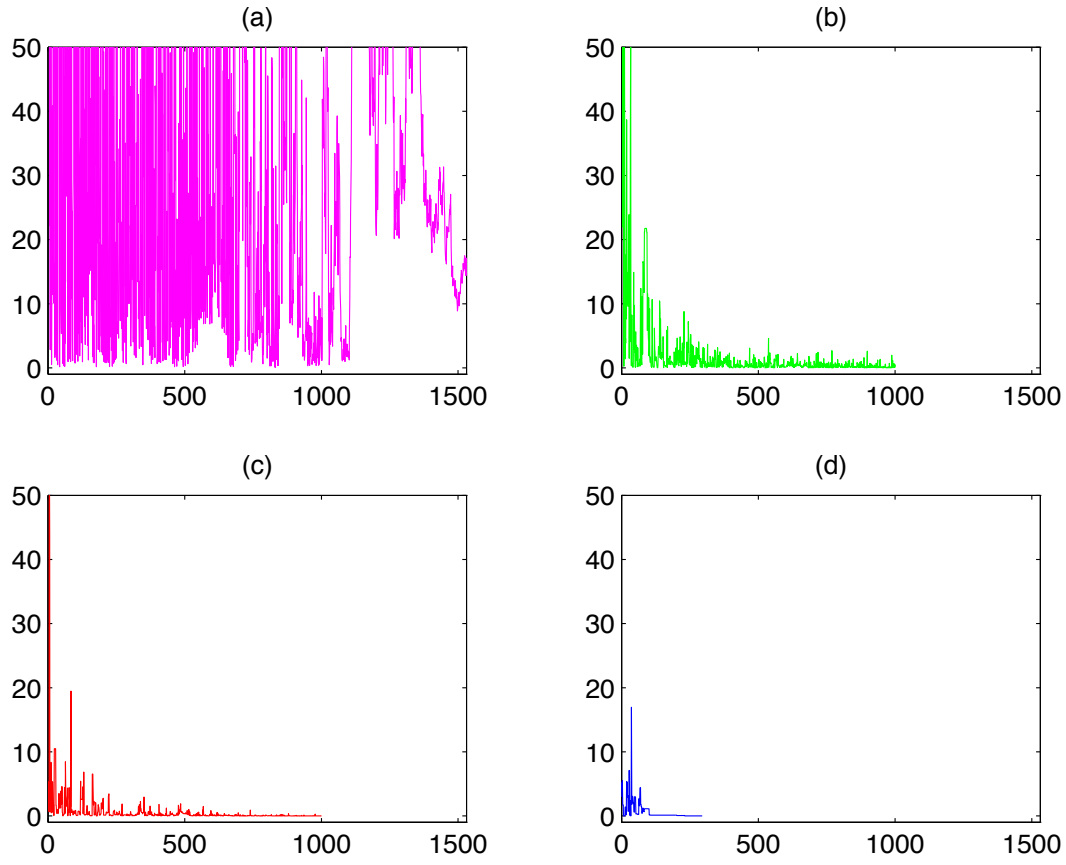


Figura 2.8: Evolución del valor de $f(x)$ después de cada cambio de temperatura para la función de Rosenbrock. (a) Algoritmo CSA. (b) Algoritmo FSA. (c) Algoritmo ASA con la ley de enfriamiento $T(t) = \frac{T_o}{1+t}$. (d) Algoritmo ASA con la ley de enfriamiento de la ecuación $T(t) = T_o\alpha^t$, $\alpha = 0.92$.

A la vista de los resultados obtenidos (ver figura 2.8), las conclusiones para la función de dos variables (función de Rosenbrock) son similares a las que se han obtenido en el

caso de una variable:

(a) El algoritmo clásico CSA converge de forma muy lenta y con mucha variabilidad, se trata del algoritmo que presenta peores prestaciones de los cuatro analizados.

(b) El algoritmo FSA converge mejor al mínimo con una pequeña variabilidad.

(c) El comportamiento del algoritmo ASA que se ve en (c) es muy similar al del algoritmo FSA aunque mejora en cuanto a la variabilidad.

(d) El algoritmo que presenta las mejores prestaciones es el ASA que se ve en (d).

2.2 Algoritmos Genéticos (GA)

Los Algoritmos Genéticos (GA) [40, 66, 47] son técnicas de optimización basadas en simular los fenómenos que se dan en la evolución de las especies adaptándolos a un problema de optimización. En definitiva, estos algoritmos tratan de replicar unas leyes de selección natural que van a afectar una población hasta conseguir individuos mejor adaptados a su entorno.

La *población* sobre la que opera el GA no es más que un conjunto de puntos del espacio de búsqueda (posibles soluciones al problema de optimización). Lo que se denomina *individuo* es un punto del espacio de búsqueda, cada uno de los individuos diferentes que pueden existir en una población debe tener asociado un *cromosoma* diferente (un código que lo identifica), es decir, debe existir un tipo de codificación para el espacio de soluciones. Dos individuos iguales tienen el mismo cromosoma. El grado de adaptación al entorno y por tanto las posibilidades de supervivencia de un individuo vienen dados por la función objetivo (función que se quiere optimizar). Dicho de otro modo, la función objetivo no es más que el entorno en el que tiene que sobrevivir un individuo, el valor que toma dicha función para un individuo concreto marcará sus posibilidades de supervivencia y reproducción.

Establecidas la codificación (estructura de los cromosomas) y la función objetivo (entorno) el mecanismo de evolución se simula mediante la aplicación de unas funciones que se conocen como *operadores genéticos*. Partiendo de una población inicial se van modificando estos individuos (evolución) mediante la aplicación de operadores genéticos, los grupos más comunes son:

- *Selección.* Seleccionará los individuos de la población que formarán parte de la siguiente generación. Esta operación depende básicamente del valor de la función objetivo de cada uno de los individuos. Los mejor adaptados tienen más probabilidades de seguir en la siguiente generación, es decir, se trata de hacer desaparecer los cromosomas que no presentan buenas cualidades.

- *Cruce.* Operación con la que se generan nuevos tipos de individuos (cromosomas diferentes) mediante la combinación de los cromosomas de dos individuos. Con este operador se buscan nuevos cromosomas que combinen cualidades de los cromosomas que ya existen con el objeto de tener individuos todavía mejores.

- *Mutación.* A partir de un individuo se genera otro diferente mediante la variación aleatoria de alguna de las partes del cromosoma. Este operador permite encontrar cromosomas con buenas cualidades que no existían en la población y que no se podían encontrar mediante cruces.

El diagrama de flujo de la figura 2.9 muestra los pasos que se siguen en un algoritmo genético. La condición de finalización para el algoritmo siguiendo criterios de tiempo transcurrido o calidad de la solución que se ha encontrado hasta el momento.

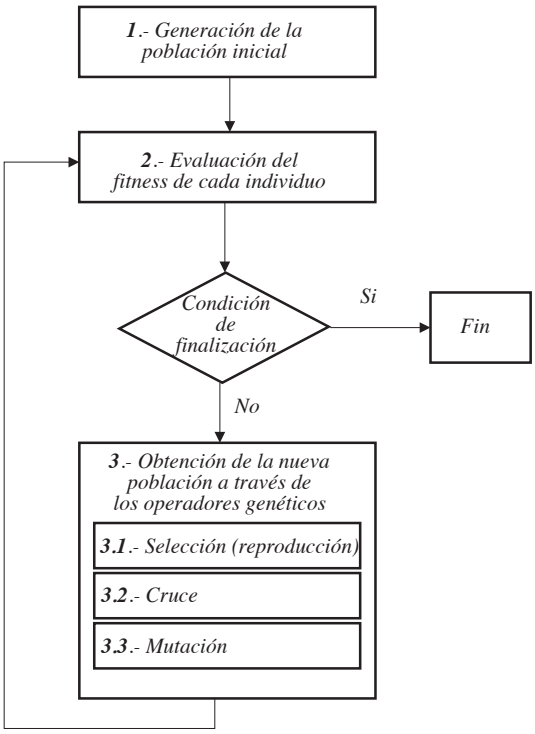


Figura 2.9: Diagrama de flujo básico de un algoritmo genético.

Las distintas variantes de algoritmos genéticos que se pueden dar se diferencian por:

- El tipo de codificación utilizada para los cromosomas. Las más comunes son la codificación binaria y la real.
- Los operadores genéticos utilizados. Dependiendo del tipo de codificación se deben implementar los operadores genéticos de una forma diferente. Además, para un mismo tipo de codificación existen muchas posibilidades para la implementación de un mismo operador genético.

Además de las obras básicas ([40, 66, 47]), en la bibliografía se pueden encontrar varios textos que recogen parte del estado actual de estas técnicas así como aspectos concretos de implementación ([22, 6, 7, 17, 103, 45].

En lo que sigue se describen detalladamente los dos tipos de codificación mencionados y un conjunto de operadores genéticos para cada una de las codificaciones.

2.2.1 Codificación binaria

Por codificación se entiende la estructura y contenido del cromosoma que se asigna a una solución del espacio de búsqueda. La codificación binaria consiste en asignar a cada posible solución una cadena binaria, es decir, una cadena de unos y ceros. Por ejemplo, el cromosoma de un determinado individuo podría ser la cadena *011110001101*.

Evidentemente, cada punto del espacio de soluciones debe quedar representado por una cadena con diferentes valores. El hecho de que los cromosomas sean de longitud finita limita el número de puntos del espacio de soluciones que pueden ser codificado, por ejemplo, con un cromosoma de longitud siete, sólo disponemos de 2^7 cadenas diferentes. Por ello, se deben seleccionar los puntos del espacio a los que se asignará cada cadena. Para realizar esta discretización existen distintas posibilidades y la que se utiliza generalmente es una discretización lineal.

Ejemplo 2.1 *Si se quiere codificar los puntos de un espacio de búsqueda unidimensional* $[u_{min}, u_{max}[= [-5, 5[$ *mediante una codificación binaria con cadenas de longitud 10 y con una discretización lineal (ver figura 2.10), se obtiene:*

Valor	*Cromosoma*
-5	*0000000000*
-5+d	*0000000001*
-5+2d	*0000000010*
-5+3d	*0000000011*
...	...
-5+(2^{10}-1)d	*1111111111*

Donde la precisión en este caso es $d=10/2^{10}$

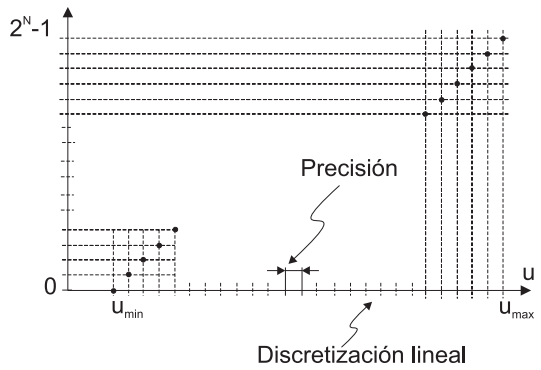


Figura 2.10: Discretización lineal de un espacio unidimensional.

Esta codificación se puede extender al caso de espacios multidimensionales sin más que asignar una parte del cromosoma a la codificación de cada dimensión. Por ejemplo, si se quiere codificar un espacio tridimensional, $[u_{1min}, u_{1max}] \times [u_{2min}, u_{2max}] \times [u_{3min}, u_{3max}]$, con unos cromosomas de longitud 18 bits, se pueden asignar seis bits a cada una de las dimensiones.

$$\text{Cromosoma} \rightarrow \underbrace{100111}_{u_1}\underbrace{001101}_{u_2}\underbrace{101100}_{u_3}$$

En el caso de la codificación binaria, y salvo que la función de coste se pueda evaluar a partir del código binario, es necesaria una *función de decodificación* que facilite el punto del espacio de búsqueda que corresponde a una cadena binaria.

Código Gray

La codificación más utilizada es la codificación binaria estándar aunque se pueden emplear otro tipo de representaciones como por ejemplo la codificación 'Gray' [18] ([79] incluye una descripción de este código y algunas aplicaciones). Este código representa cada número con una cadena de unos y ceros (cadena binaria) pero con una propiedad particular: dos números consecutivos difieren únicamente en un bit. Existen diversas formas para esta codificación pero la que más se utiliza es la que se conoce como código Gray reflejado. Este código se genera partiendo de una cadena con todos los bits a cero y se va cambiando el bit que se encuentra más a la derecha y que produzca un nuevo código. En la tabla 2.1 se muestra un ejemplo comparándolo con la codificación binaria estándar. Se puede generar una cadena en código Gray (B_{Gray}) a partir de una cadena binaria estándar (B_{est}) aplicando los siguiente pasos:

1. El bit más significativo de la cadena binaria estándar se mantiene en la cadena binaria Gray. Por ejemplo, para una cadena de longitud N:

$$B_{Gray}(N) = B_{est}(N)$$

2. Para los demás bits ($i < N$) se aplica una operación de OR exclusivo:

$$B_{Gray}(i) = XOR\,(B_{est}(i+1), B_{est}(i))$$

Para la operación inversa se siguen los siguientes pasos:

1. Para i=N (bit más significativo): $B_{est}(N) = B_{Gray}(N)$
2. Para los demás bits ($i < N$):

$$B_{est}(i) = XOR\,(B_{est}(i+1), B_{gray}(i))$$

Número Entero	Código binario Estándar	Código binario Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tabla 2.1: Comparación de los códigos binarios estándar y Gray para la codificación de enteros del 0 al 7.

Cuando se aplica la codificación binaria Gray en un algoritmo genético, no se aprecia un comportamiento muy diferente al de la codificación binaria estándar, los resultados que se obtienen son similares [9].

2.2.2 Operadores genéticos para la codificación binaria

Una vez definida la estructura de los cromosomas y una operación de decodificación, se debe seleccionar el conjunto de operadores que se aplicarán sobre los individuos de la población, operadores genéticos. Pero para iniciar el proceso evolutivo es necesaria la generación de una población inicial. Esta se obtiene normalmente de forma aleatoria[3], es decir, se generan N_{ind} (número de individuos) cadenas binarias aleatorias, cada una de ellas corresponderá al cromosoma de un individuo de la población. Pueden aparecer cadenas binarias idénticas, esto significa que se han generados individuos iguales.

Operador de selección

El primer operador genético que se aplica es el que selecciona los individuos que van a componer la siguiente generación y la cantidad de cada uno de ellos. Esta operación depende del valor de la función de coste de los individuos. El operador de selección debe escoger con mayor probabilidad aquellos individuos que presentan un valor de la función

[3]Cabe la posibilidad de generar poblaciones iniciales que no sean aleatorias.

mejor, incluyendo además la posibilidad de que individuos que no son tan buenos tengan cierta probabilidad de aparecer. Esto último pretende mantener una cierta diversidad en cuanto a la información que contienen los cromosomas. No es bueno que desaparezcan rápidamente todos los individuos malos puesto que pueden contener parte de la información genética necesaria (parte de un cromosoma) para obtener el óptimo.

El objetivo que se persigue en los métodos que se describen es que la cantidad de cada uno de los individuos de la nueva población venga dado por la siguiente proporción (2.11):

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N_{ind}} f(x_j)} \tag{2.11}$$

Donde x_i representa un individuo presente en la población de la generación actual y $f(x_i)$ representa su valor de la función objetivo.

Dos métodos utilizados habitualmente son (ver figura 2.11) [41], [3]:

- **Mecanismo de la ruleta simple (RWS)**. El mecanismo se puede describir como un símil con una ruleta. La parte fija de la ruleta esta dividida en porciones que corresponden a los distintos tipos de cromosomas que existen en la población inicial. Por ejemplo, la población antes de la selección está compuesta por 12 individuos de los cuales sólo existen 8 tipos diferentes (cromosomas diferentes). Las porciones de la parte fija de la ruleta no son iguales, dependen del valor de la función objetivo del individuo al que representan según la expresión 2.11. Para obtener todos los individuos de la nueva población se hace girar la ruleta tantas veces como individuos se quieran seleccionar.

  Por ejemplo, para obtener una nueva población de 12 individuos se hace girar 12 veces la ruleta, el puntero indica el tipo de individuo que se genera en cada lanzamiento.

  Este mecanismo no asegura que la siguiente generación que se obtiene mediante la selección cumpla las proporciones de la expresión 2.11, puesto que depende de los lanzamientos que se realicen.

- **Stochastic Universal Sampling (SUS)**. El mecanismo también es similar al de una ruleta, las porciones se ajustan de la misma manera que se ha descrito anteriormente. La diferencia es que la parte móvil de la ruleta no tiene un único puntero, sino tantos como individuos se quieren seleccionar para la siguiente generación. Estos punteros están uniformemente situado. Haciendo girar una única vez la ruleta se obtienen todos los individuos de la población.

  Con este mecanismo si que se asegura que la probabilidad de selección de un individuos es el que corresponde a la expresión 2.11.

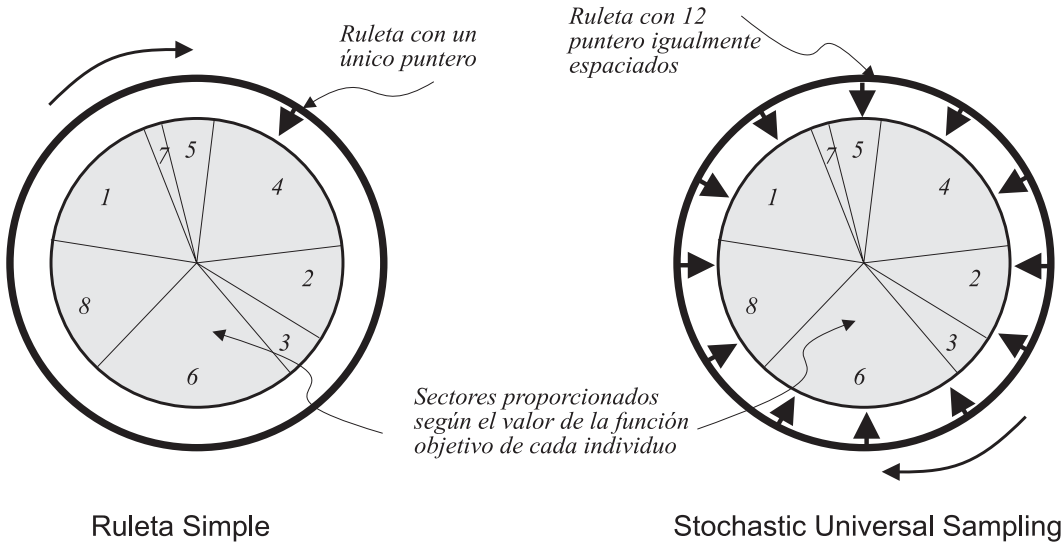


Figura 2.11: Representación gráfica de los mecanismos de selección RWS y SUS. Las porciones de las partes fijas de ambas ruletas satisfacen la expresión: $\frac{f(x_i)}{\sum_{j=1}^{N_{ind}} f(x_j)}$.

Para conseguir un adecuado funcionamiento del algoritmo, se deben evitar convergencias prematuras del mismo, es decir, que rápidamente toda la población este compuesta por un único tipo de individuo. De no hacerlo, se incrementa notablemente el riesgo de que el algoritmo se bloquee en un óptimo local. Este fenómeno ocurre cuando existen individuos que tienen un valor de la función de coste mucho mejor que los demás, el operador de selección escogerá muchos individuos de este tipo para la siguiente generación disminuyendo las posibilidades para otros tipos no tan buenos.

Una solución a este problema es modificar el valor de la función objetivo haciendo que los valores muy buenos y muy malos no lo sean tanto. Este efecto se puede conseguir mediante una operación previa a la selección, denominada **operación de Ranking**. Ésta consistiría por ejemplo, en ordenar los individuos por orden según el valor de la función de coste y asignar como nuevo valor de la función de coste el orden en esa clasificación. Esta operación se conoce como *ranking* lineal. Evidentemente esta no es la única forma de reajustar los valores de la función de coste, se pueden realizar otros tipos de asignaciones de valores que no sean lineales.

La tabla siguiente muestra un ejemplo de ranking lineal para en un problema de minimización.

Individuo	Valor función de coste	Valor función de coste con *Ranking*
x_1	3.73	4
x_2	0.012	6
x_3	50.3	1
x_4	2.145	5
x_5	15.12	3
x_6	25.72	2

Operador de cruce

El objetivo de esta operación es la de mezclar la información de los cromosomas con el fin de obtener individuos diferentes a los que ya existen. Se trata de explorar el espacio de búsqueda. La operación de cruce no se realiza sobre todos los individuos de la población, cada individuo tiene una probabilidad de ser cruzado, probabilidad de cruce P_c. Si se cruzan todos los individuos, cada generación puede resultar totalmente diferente de la anterior, el algoritmo tiene dificultades para converger.

Para la operación de cruce existen varias posibilidades entre las que están:

- **Punto de cruce simple**. Se selecciona aleatoriamente una posición por donde se van a cortar los cromosomas padres para combinarse (figura 2.12).

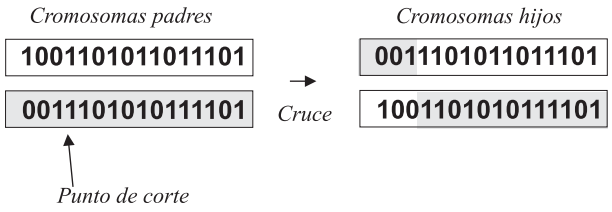


Figura 2.12: Cruce simple en la codificación binaria.

- **Cruce multipunto**. En el cruce multipunto, los individuos hijos se obtienen de cortar los cromosomas de los padres por varios puntos en lugar de por uno sólo (figura 2.13). Esto permite una mayor variedad de cromosomas hijos, más posibilidades de explorar el espacio de búsqueda.

- **Cruce uniforme**. Uno de los problemas que pueden presentar los algoritmos genéticos es la convergencia prematura del algoritmo debido, básicamente, a una pérdida de diversidad de la población (los individuos presentes en la población tienen cromosomas muy similares, por tanto es difícil que varíen mucho los cromosomas de la

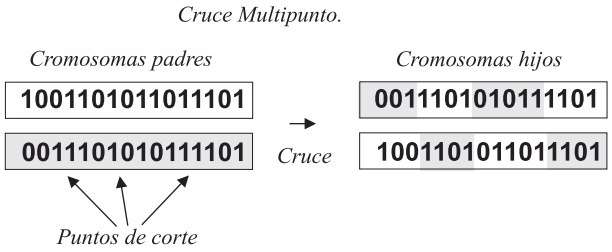


Figura 2.13: Cruce multipunto en la codificación binaria.

siguiente generación). Con el cruce uniforme se pretende ampliar las posibilidades de combinación de los cromosomas padre para generar cromosomas hijos.

El cruce uniforme ([96], [48]) puede considerarse como una extensión del *cruce multipunto*. En el cruce uniforme, los cromosomas hijos se obtienen de la siguiente forma: cada uno de los bits del cromosoma del primer hijo puede obtenerse de un cromosoma padre u otro dependiendo de una determinada probabilidad (α). El cromosoma del segundo hijo se obtiene de la misma forma pero con una probabilidad $1-\alpha$.

En la figura 2.14, las cadenas *origen* indican de qué padre se generan cada uno de los bits de los hijos, si el valor es 1 significa que el bit se toma del padre 1 (P1), si el valor es 2 el bit se toma de P2. Cada una de las cadenas corresponde al origen de uno de los hijos y se han generado aleatoriamente con una probabilidad α, la primera cadena corresponde a H1 y la segunda a H2 siendo la inversa de H1. Dependiendo de la implementación que se realice H2 no tiene por que ser complementario, en ese caso se genera con una probabilidad $1-\alpha$, esta implementación es un poco más costosa computacionalmente (se deben generar más números aleatorios).

Para $\alpha = 0.5$ los cromosomas hijos tienen (en promedio) tanta información genética de un padre como del otro. Si se aumenta (o disminuye) el valor de α los cromosomas hijos contienen más información genética de uno de los padres (se va a parecer más a ese padre).

Operador de mutación

Variación aleatoria de los bits del cromosoma de los individuos de la población con una probabilidad P_m (figura 2.15). Este operador permite incrementar la capacidad de exploración del espacio de búsqueda puesto que posibilita la aparición de información genética que no existe en la población. En este sentido se complementa al operador de cruce, este operador permite que aparezcan cromosomas nuevos pero combinando información genética que existe en la población. El operador de mutación puede (aunque no siempre) hacer aparecer nueva información. La probabilidad de mutación debe ser baja puesto

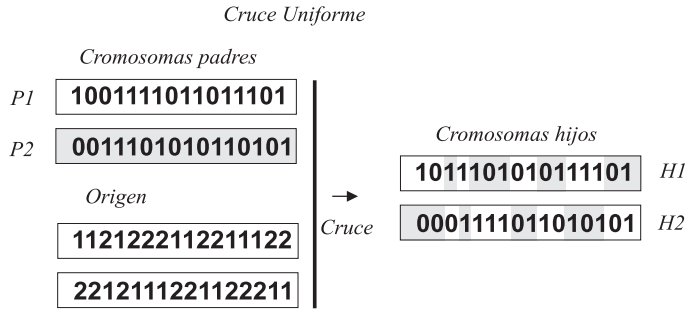


Figura 2.14: Cruce uniforme en la codificación binaria.

que de lo contrario se dificulta la convergencia del algoritmo. Si varían aleatoriamente muchos bits de los cromosomas es muy difícil que los individuos se parezcan cada vez más (convergencia del algoritmo).

Figura 2.15: Mutación en la codificación binaria.

2.2.3 Codificación real

Una de las características de un algoritmo genético es el tipo de alfabeto que se utiliza para la codificación de los parámetros de la función objetivo. Hasta el momento, se ha estudiado la codificación binaria. Otra de las posibles codificaciones, que se va a estudiar a continuación, es la codificación real ([66]). Cada parámetro de la función objetivo está representado por un número real, por tanto cada cromosoma esta formado por una cadena de números reales, tantos como la dimensión del espacio de búsqueda.

Por ejemplo, si se quiere codificar un espacio tridimensional como: $[u_{1min}, u_{1max}] \times [u_{2min}, u_{2max}] \times [u_{3min}, u_{3max}]$.

$$\text{Cromosoma} \rightarrow \underbrace{34.1235}_{u_1} \mid \underbrace{123.0293}_{u_2} \mid \underbrace{0.1203}_{u_3}$$

La característica fundamental de este tipo de codificación es que no discretiza el espacio de búsqueda[4], esto se puede considerar como una ventaja salvo que el espacio de búsqueda sea discreto y por lo tanto esta codificación no se puede aplicar. En la codificación binaria,

[4]Si no se tiene en cuenta la discretización propia del computador que realiza los cálculos

existe un número finito de punto en el espacio de búsqueda que depende de la longitud del cromosoma. En el caso de la codificación real, el espacio es continuo, existen infinitos puntos en él, se pueden alcanzar soluciones con una precisión mayor sin incrementar la longitud del cromosoma y por lo tanto sin incrementar el coste computacional (la longitud del cromosoma afecta al coste del algoritmo en la codificación binaria).

2.2.4 Operadores genéticos para la codificación real

Al cambiar de codificación se deben modificar los operadores genéticos que van a hacer evolucionar la población en cada generación. La generación de la población inicial es similar a la de la codificación binaria. Se genera, para cada uno de los parámetros de cada individuo un número real aleatorio (distribución uniforme) dentro del espacio de búsqueda.

Otra ventaja de la codificación real es que la evaluación de la función de coste no necesita una operación de decodificación para extraer los valores de los parámetros de la cadena que forma el cromosoma, se utilizan directamente los componentes del cromosoma.

Operación de selección

Con el mismo objetivo que en la codificación binaria, se realiza una operación de **Ranking** previa a la selección. La operación de selección no varía respecto de la que se describe en la codificación binaria, igual que en ese caso es más aconsejable utilizar el método de *Stochastic Universal Sampling*.

Operación de cruce

También en este caso se define una probabilidad de cruce P_c, no tienen por que cruzarse todos los individuos.

Para realizar las operaciones de cruce con codificación real se puede utilizar las operaciones que se han definido para la codificación binaria, siempre que el espacio de búsqueda sea de dimensión mayor o igual a dos (para una dimensión sólo se pueden aplicar los operadores de cruce por recombinación que se describen más adelante):

- Cruce punto simple.
- Cruce multipunto.
- Cruce uniforme.

La forma de realizar estas operaciones es utilizar cada una de las variables del cromosoma de la codificación real como un bit de la codificación binaria, por tanto los cruces se producirán en las posiciones que separan las variables (figura 2.16).

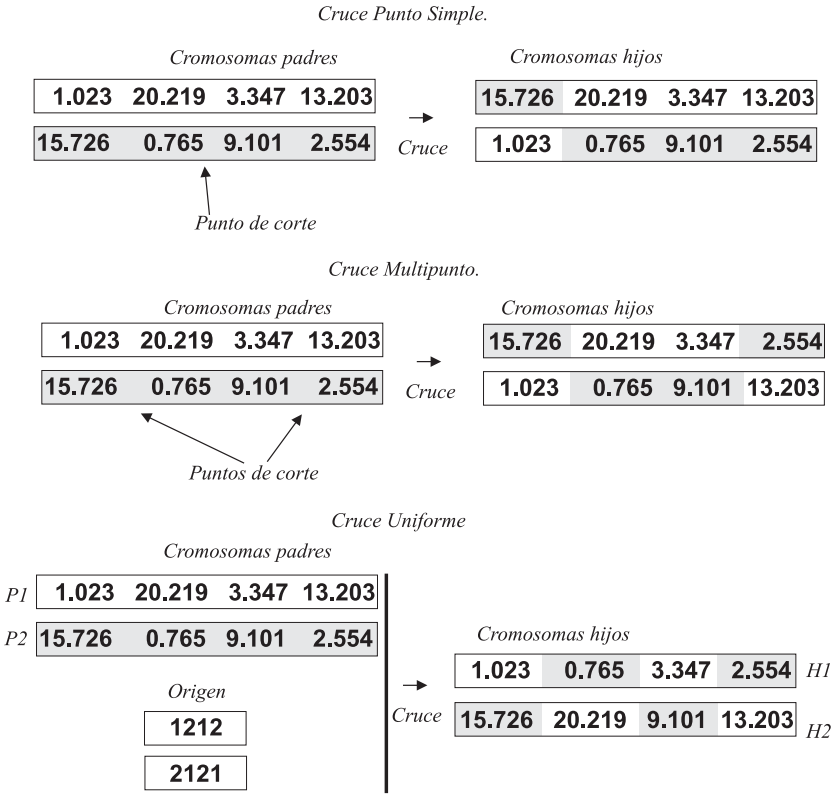


Figura 2.16: Operaciones de cruce en la codificación real. Cruce simple, multipunto y uniforme

Con la codificación real aparece otro conjunto de posibilidades para realizar la operación de cruce que permiten, además, una mayor diversidad en la población resultante si se compara con los operadores de cruce anteriores. Estos operadores de cruce si que se pueden utilizar para un espacio unidimensional.

- **Recombinación lineal**. Los cromosomas hijos se obtienen realizando una combinación lineal de los cromosomas padre, se aplica sobre cada variable del cromosoma las siguientes operaciones:

$$H1 = \alpha \cdot P1 + (1 - \alpha) \cdot P2$$

$$H2 = \alpha \cdot P2 + (1 - \alpha) \cdot P1$$

El valor del parámetro α es el mismo para todas las variables del cromosoma y se puede elegir en el intervalo $[-d, 1 + d]$. Para la recombinación lineal se utiliza $d = 0$, para la recombinación lineal extendida un valor usual es $d = 0.25$.

Si se asigna un valor de $d > 0$ se expande la zona donde pueden aparecer los hijos, se incrementa la propiedad de exploración del espacio de búsqueda, aunque un valor de d demasiado elevado dificulta la convergencia del algoritmo (la población se puede dispersar demasiado).

- **Recombinación intermedia**. Se realizan las mismas operaciones que en la recombinación lineal, la única diferencia es que se utiliza un valor diferente de α para cada una de las variables.

El efecto de este tipo de operadores de cruce se puede ver gráficamente (figura 2.17) para el caso de un espacio de búsqueda bidimensional, se muestran donde pueden aparecer los individuos resultantes del cruce que sean diferentes a los padres. En la figura 2.17 los cromosomas padres son:

$$\text{Padre P1} \rightarrow [u_{1(P1)}, u_{2(P1)}]$$

$$\text{Padre P2} \rightarrow [u_{1(P2)}, u_{2(P2)}]$$

Se puede ver cualitativamente la capacidad de exploración del espacio de búsqueda de cada uno de los operadores de cruce. Para un espacio bidimensional los cruces por punto simple, multipunto y uniforme son equivalentes, en la figura sólo se muestra uno de ellos (punto simple).

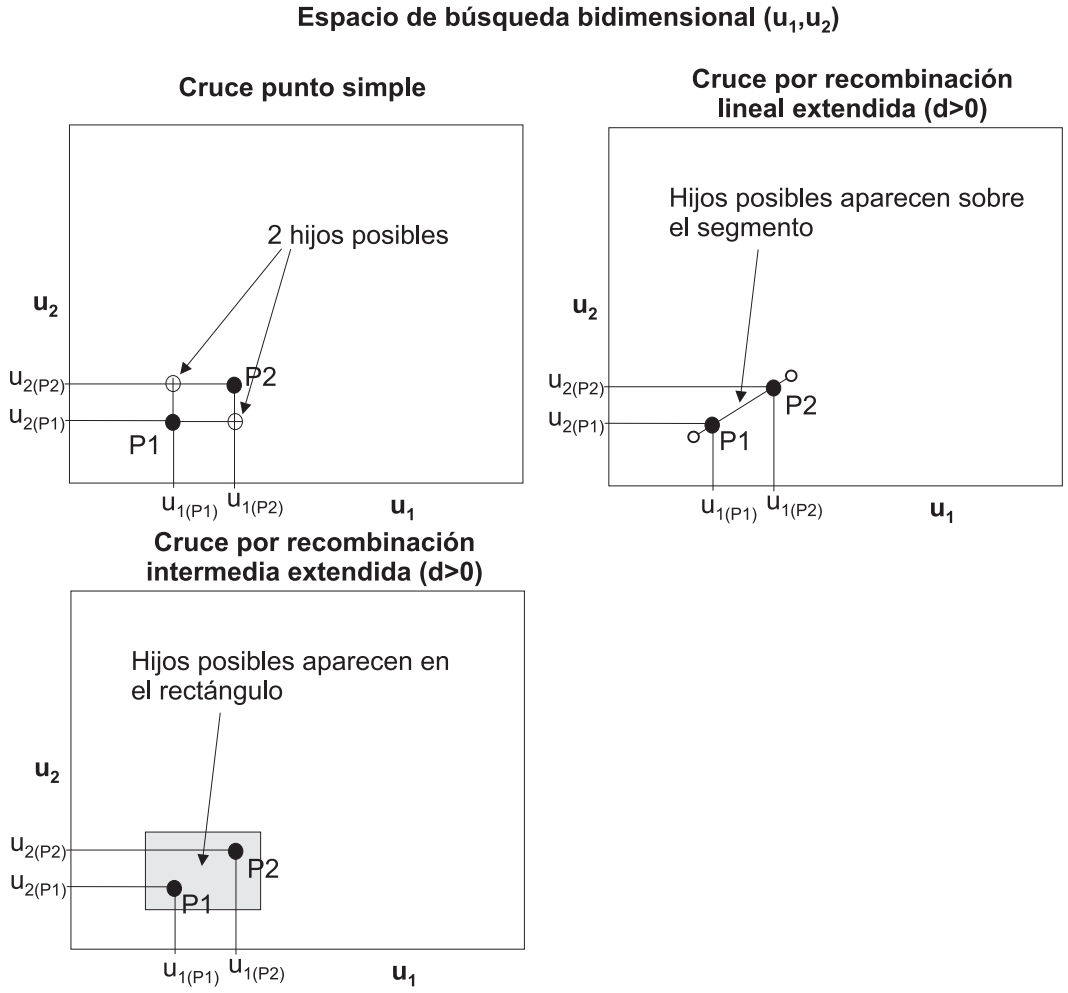


Figura 2.17: Efecto de las operaciones de cruce en la codificación real. Cruce simple y recombinaciones lineal e intermedia extendidas ($d > 0$).

Operación de mutación

Igual que para la codificación binaria, el operador de mutación se encarga de incrementar la capacidad de exploración del espacio de búsqueda, en concreto trata de hacer que aparezca en la población información genética que no existía. Cuando se utiliza los operadores de cruce binarios esta operación de mutación es imprescindible para que aparezca nueva información genética, pero si los operadores que se utilizan son los de recombinación puede que no sea necesaria (aunque si recomendable) la utilización de operación de mutación. Los operadores por recombinación son capaces de generar individuos con información genética diferente a la de los padres. Esta característica puede ser interesante desde el punto de vista de coste computacional.

La forma más simple de realizar la mutación de un parámetro en un cromosoma consiste en añadir un valor aleatorio (con una distribución uniforme) a dicho parámetro manteniendo el parámetro resultante en el espacio de búsqueda, **Mutación aleatoria**.

Se puede plantear otra forma de mutación. También consiste en añadir un valor al parámetro que se quiere mutar pero teniendo en cuenta que en algunos casos es conveniente que las mutaciones provoquen variaciones muy pequeñas (el parámetro mutado se mantiene cerca del parámetro original) para tratar de aumentar la precisión de la solución encontrada y en otras ocasiones interesa justamente lo contrario para explorar nuevas zonas.

El operador (**mutación orientada**) que se describe ([71]) provoca con mayor probabilidad mutaciones de poco valor, aunque también ocurren mutaciones con valores altos:

$$Variable_{mutada} = Variable \pm Rango \cdot \delta$$

Donde:

$$Rango = 0.5 \cdot \text{dominio de la variable}$$

$$\delta = \sum_{i=1}^{m} \alpha_i 2^{-i}$$

$$\alpha_i = \begin{cases} 1 & \text{con una probabilidad: } \frac{1}{m} \\ 0 & \text{en el resto de los casos} \end{cases}$$

$$m = 20$$

Se puede conseguir un efecto intermedio si se utiliza un operador de mutación aleatoria pero con una distribución distinta de la uniforme, por ejemplo, con una distribución normal (ver figura 2.18).

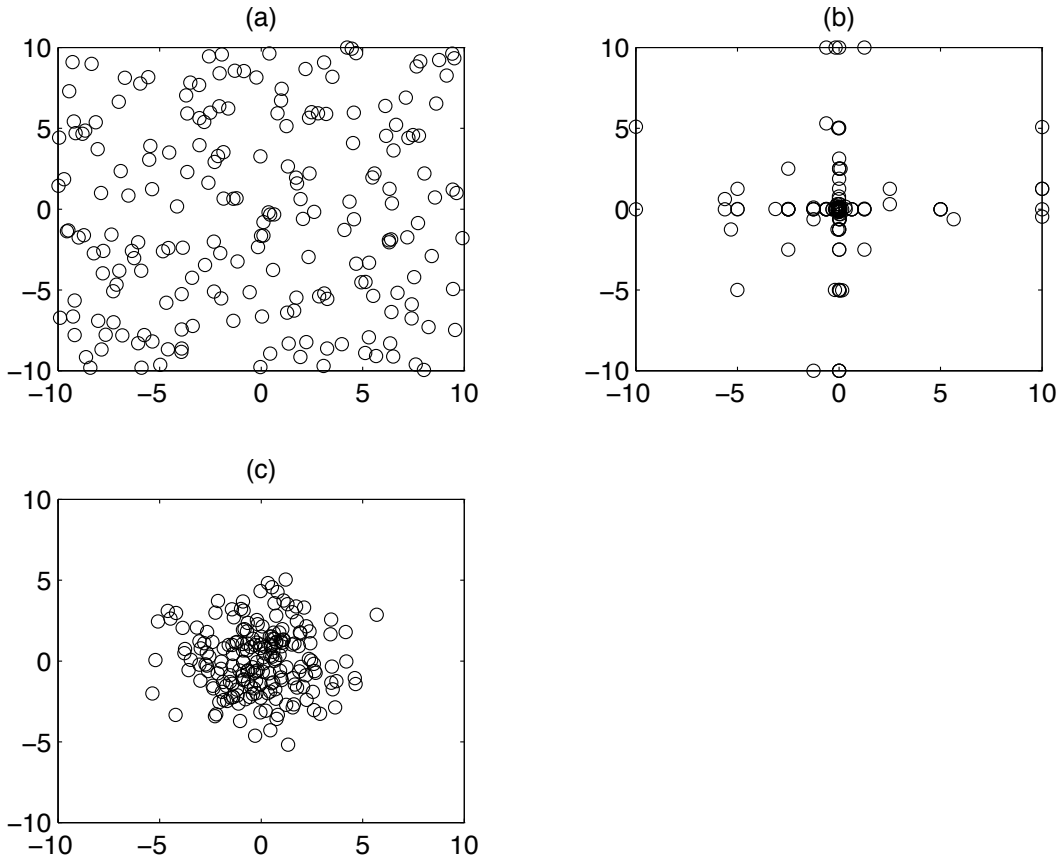


Figura 2.18: 200 mutaciones del punto (0,0), en el espacio $[-10, 10] \times [-10, 10]$, con distintos operadores de mutación. (a) Mutación aleatoria con distribución uniforme, (b) Mutación orientada, (c) Mutación aleatoria con distribución normal de varianza 2 .

2.2.5 Ejemplo de evolución

Cualquiera de las codificaciones con una combinación de operadores genéticos podría tener una evolución, según van pasando las generaciones, como la que se muestra a continuación.

El problema que se quiere resolver es el de obtener el máximo de la función de dos variables siguiente (figura 2.19):

$$f(u_1, u_2) = R(u_1, u_2) - \sum_{i=1}^{10} (y_i - r_i)^2$$

Donde:

- $y_i = 2y_{i-1} - y_{i-2} + 0.005u_{i-1} + 0.005u_{i-2}$
- $[y_0, y_{-1}, u_0, u_{-1}] = [0.5222, 0.1994, -9.8052, -15.2083]$
- $-40 \leq u_1 \leq 40$; $-40 \leq u_2 \leq 40$
- $u_i = u_2,\ i > 2$
- $r_i = 1,\ \forall\, i$
- $\Delta u_1 = u_1 - u_o$; $\Delta u_2 = u_2 - u_1$

$$R(u_1, u_2) = \begin{cases} -1000 & ; \text{ Si } (-5 < \Delta u_1 < 5) \text{ OR } (-5 < \Delta u_2 < 5) \\ 0 & ; \text{ Para los demás valores} \end{cases}$$

El problema se resuelve con un algoritmo genético con las siguientes características: Codificación binaria estándar, Ranking lineal, operador de selección SUS, cruce por punto simple con una probabilidad $P_c = 0.7$ y mutación aleatoria con una probabilidad $P_m = 0.02$.

La figura 2.20 muestra la distribución de los individuos de la población (indicados mediante círculos) para distintas generaciones. Además se muestra la función a maximizar mediante las curvas de nivel, de manera que se puede observar de forma cualitativa y aproximada el valor de la misma observándose que la mayoría de los puntos se acercan al máximo según transcurren las generaciones.

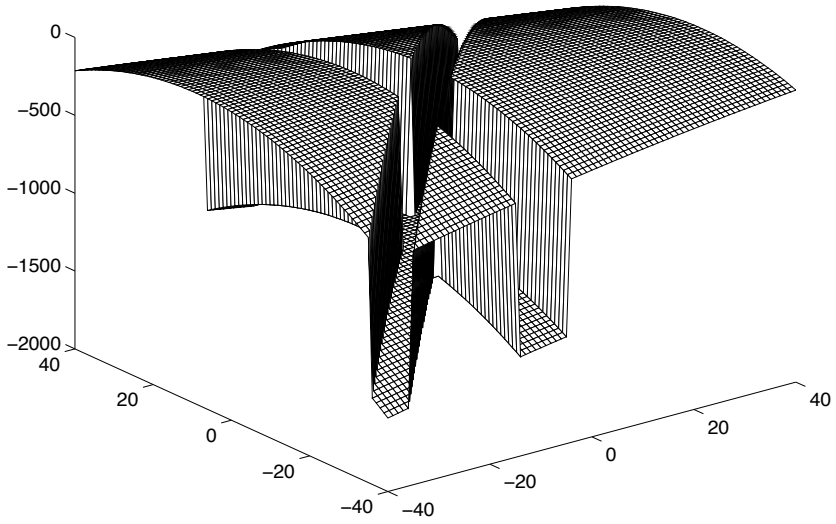
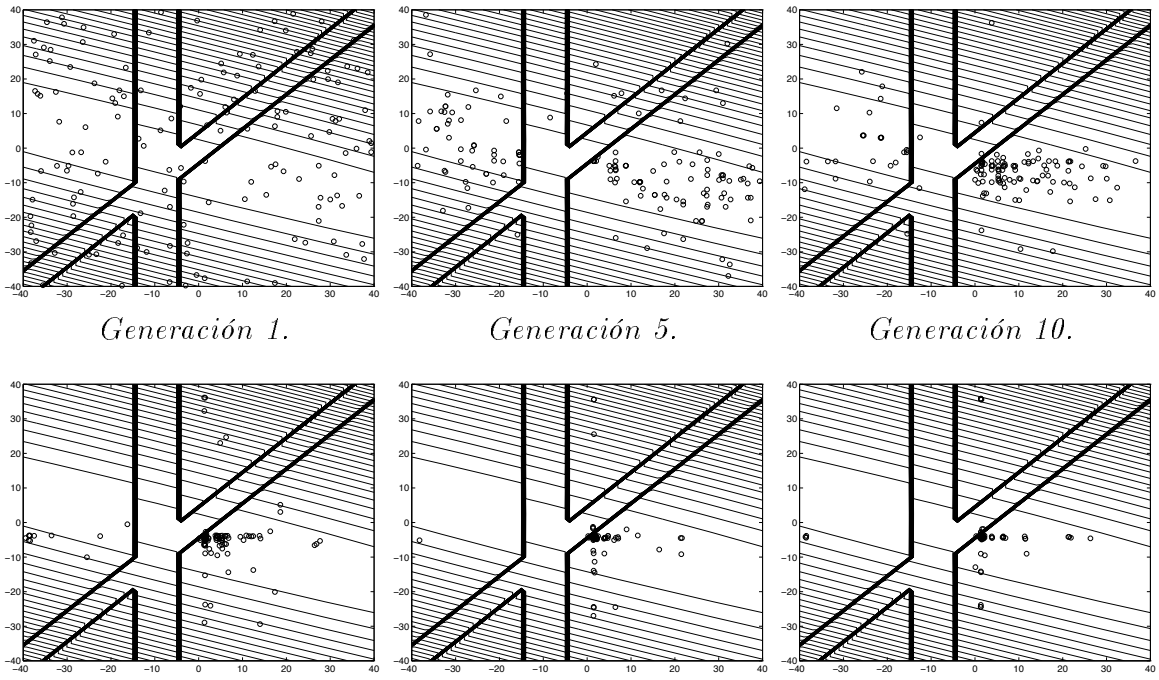


Figura 2.19: Representación gráfica de $f(u_1, u_2)$, $(u_1, u_2) \in [-40, 40] \times [-40, 40]$



Generación 1. *Generación 5.* *Generación 10.*

Generación 15. *Generación 20.* *Generación 25.*

Figura 2.20: Localización de los individuos de una población en distintas generaciones para un problema de maximización.

Capítulo 3

Evaluación de los algoritmos GA y SA

La evaluación de unos algoritmos de optimización heurística está afectada por numerosos parámetros (figura 3.1) y por tanto resulta muy costoso realizar una evaluación exhaustiva teniendo en cuenta todas las variables. Si además se deben comparar varios algoritmos entre si, el problema de la evaluación se incrementa enormemente. En cualquier caso siempre se puede realizar una evaluación parcial acorde con las necesidades, es decir, teniendo presente para qué se van a utilizar los algoritmos y conocer los rangos de validez de esta comparación.

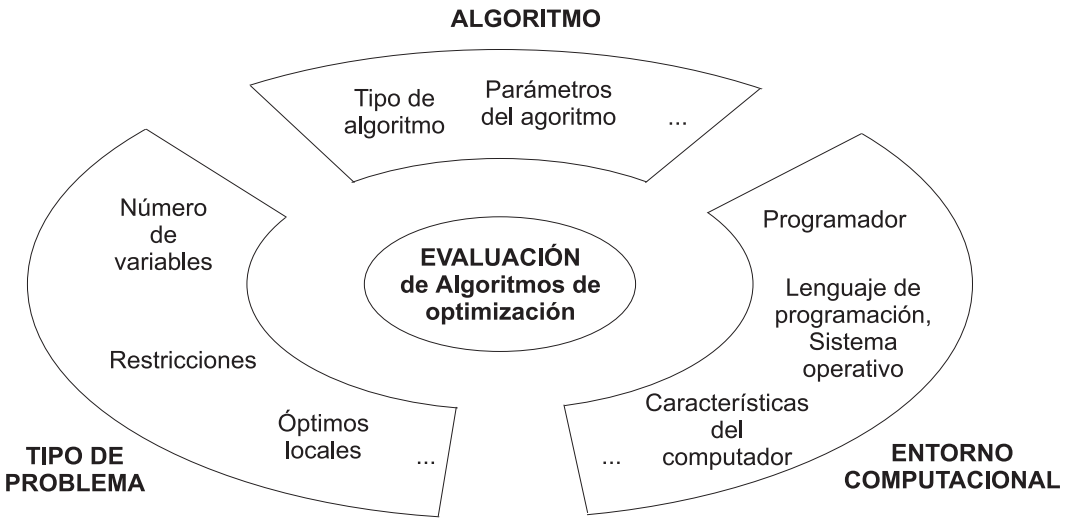


Figura 3.1: Evaluación de algoritmos de optimización.

Dada la enorme dificultad que presenta realizar una comparación objetiva de métodos heurísticos, se recurrió a la búsqueda de bibliografía significativa que tratase este tema. En esta búsqueda fue posible encontrar el artículo *Designing and reporting on computational*

experiments with heuristic methods [5], de la revista *Journal of Heuristics*, donde los autores reconocen esas mismas dificultades, debido fundamentalmente a los problemas de generalización que este tipo de comparación plantea, pero proponen una serie de directrices que deberían ser tenidas en consideración cuando se aborda un estudio comparativo de este tipo. Parece que los editores de esta revista, dedicada íntegramente a los métodos heuristicos, consideran que el problema es de la entidad suficiente como para que el primer artículo trate de marcar las bases para la comparación objetiva de métodos heurísticos.

En ese artículo se recomienda que para una comparación más objetiva es mejor programar todos los algoritmos en un mismo entorno computacional[1] que comparar los resultados que aparecen en la bibliografía, puesto que normalmente estos resultados presentan una gran heterogeneidad de los elementos de comparación. Esto no siempre es posible puesto que requiere que el programador conozca bien todos los algoritmos que se deben programar. En este trabajo es posible una comparación de este tipo entre los algoritmos de *Simulated Annealing* y los Algoritmos Genéticos, puesto que se han estudiado en detalle.

La elección del entorno computacional ha estado restringida a los componentes disponibles en el laboratorio, aún así el entorno para los test es adecuado y está constituido por:

- Pentium 200MHz y 32Mb RAM, con sistema operativo Windows 95.
- La programación de los algoritmos descritos en las secciones anteriores se ha realizado en Matlab 4.2. Se podría haber utilizado algún lenguaje de alto nivel como el lenguaje *C* pero se optó por realizar los programas en Matlab por simplificar el proceso de programación y representaciones gráficas.

Este mismo artículo también recomienda, en cuanto a las medidas que se deben tomar para evaluar las prestaciones de cada algoritmo, lo siguiente:

- **Calidad de la solución.** Esta sólo se puede evaluar si se conoce el mínimo que se busca, es necesario disponer de un conjunto de funciones de test bien conocidas. En este sentido, hay que mencionar que cuando se intenta aplicar un algoritmo de optimización al control predictivo, pueden aparecer una gran variedad de problemas: fuertes restricciones, varios mínimos, mínimos locales próximos a los mínimos globales, mínimos en zonas muy planas, etc. Se debe tener en cuenta esta variedad de problemas en los ensayos de evaluación que se realicen.

  Se ha elegido un conjunto de siete funciones de test (F1, F2, F3, F4, F5, F6 y F7) que se describen con detalle más adelante en el anexo B, estas funciones se utilizan

[1] El entorno computacional incluye la máquina sobre la que se ejecuta, el sistema operativo, la carga del sistema, el compilador utilizado y la habilidad del programador.

como banco de pruebas en trabajos publicados para evaluar las capacidades de un algoritmo de optimización (por ejemplo ver [40] y [104]). Con la elección de estas funciones se trata de recoger una amplia variedad de problemas de optimización.

Para evaluar la calidad de la solución que encuentra un algoritmo de optimización, se pueden tomar dos tipos de valores:

- Evaluar la distancia del valor mínimo encontrado por el algoritmo $f(x_s)$ al valor mínimo de la función $f(\hat{x})$, $|f(x_s) - f(\hat{x})|$.
- Evaluar la distancia de la solución encontrada x_s al punto mínimo $\hat{x}$. La distancia se obtiene de la norma $\|x_s - \hat{x}\|$.

- **Coste computacional.** El coste se suele evaluar según la memoria que utiliza el algoritmo y el tiempo. De estas dos medidas, la que más nos interesa es la del tiempo puesto para poder aplicar el algoritmo en un entorno de tiempo real[2]. En todo el trabajo que se presenta a continuación, cuando se hable de coste computacional, sólo se tendrá en cuenta el coste en tiempo del algoritmo. El problema que aparece es que es complicado medir únicamente el tiempo del algoritmo sin que influyan otras tareas que se puedan ejecutar por requerimientos del sistema operativo. Para tratar de detectar posibles errores en estas medidas se utilizan dos indicadores:

  - Contabilizar el número de operaciones en coma flotante (FLOPS). En este caso si que es posible medir únicamente las operaciones en coma flotante del algoritmo, Matlab dispone de unas funciones que facilitan estas medidas. El problema es que algunas de las operaciones que se realizan en el algoritmo no son de coma flotante y por tanto no quedan reflejadas. Se trata pues de una medida parcial pero junto con la medida de tiempo da una valoración razonable del coste computacional.
  - Contabilizar el tiempo invertido en el algoritmo (Unidades de tiempo). Como ya se ha mencionado, esta medida puede estar perturbada por otras operaciones del sistema y por tanto no conviene que se tome como única medida del coste computacional.

  Cada una de estas medidas se puede tomar de diferentes formas: sobre el total del algoritmo, midiendo el coste por etapas del algoritmo, por funciones, etc. En cualquier caso, ninguno de estos indicadores tiene validez por separado puesto que están altamente influidos por el entorno computacional. Por ejemplo, un algoritmo que este muy dividido en funciones (que son llamados por un programa principal), aunque sea más legible tendrá un coste computacional superior a su equivalente menos fragmentado. Además estas medidas no son comparables entre computadores. Por ello aunque es conveniente tomar dichas medidas, puesto que ofrecen órdenes de

[2] La restricción de memoria es cada vez menos restrictiva debido a los avances tecnológicos

magnitud, es deseable tomar otro tipo de medidas que sean más objetivas, menos dependiente del entorno computacional.

Una solución generalmente adoptada es la de contar el número de operaciones propias del algoritmo en lugar de FLOPS o unidades de tiempo, en este sentido se suele utilizar el **número de evaluaciones de la función objetivo**. Este indicador tiene la ventaja adicional de que facilita la comparación entre computadores, parece por tanto una medida más objetiva.

- **Robustez del algoritmo.** Por robustez del algoritmo se entiende:
  - Grado de habilidad para resolver un amplio abanico de problemas.
  - Grado de influencia en la resolución de un problema de las variaciones de los parámetros de ajuste del algoritmo (por ejemplo, probabilidades de cruce, mutación en un algoritmo genético) .

  Las medidas que se deben tomar para evaluar la robustez deben ser tanto de calidad de la solución como de coste computacional, todo ello para distintos valores de los parámetros de ajuste del algoritmo.

Tratando de seguir estas ideas, se han realizados dos tandas de ensayos, en la primera se pretende comparar varios de los algoritmos heurísticos descritos, tratando de obtener órdenes de magnitud de los costes computacionales y la calidad de la solución.

La segunda tanda de ensayos se ha centrado en el algoritmo que ha dado mejores resultados en la primera fase, y el objetivo en este caso es tratar de establecer la robustez y sobre todo establecer unas reglas de ajuste de los parámetros del mismo. Esta información será de gran utilidad para la utilización posterior de estos algoritmos en un entorno MBPC.

3.1 Análisis comparativo de varios algoritmos

En esta primera fase de los ensayos se realiza un estudio comparativo de distintos algoritmos de optimización que se han descrito en secciones anteriores. Los cinco algoritmos que se van a estudiar son los que se describen en la tabla 3.1 y se nombran con: AO1, AO2, AO4, AO4 y AO5.

La selección de estos algoritmos se ha realizado por los siguientes criterios:

1. En cuanto a los algoritmos genéticos, el primero (AO1) corresponde al estándar y sirve de comparación con los demás.

ALGORITMOS GENÉTICOS

	Codificación	Ranking	Selección	Cruce	mutación
AO1	Binaria estándar	Lineal	Stochastic Universal Sampling	Punto simple Prob. P_c	Aleatoria Prob. P_m
AO2	Binaria estándar	Lineal	Stochastic Universal Sampling	Uniforme Prob. P_c Recomb. α	Aleatoria Prob. P_m
AO3	Binaria estándar	Lineal	Stochastic Universal Sampling	Uniforme Prob. P_c Recomb. α	NO
AO4	Real	Lineal	Stochastic Universal Sampling	Recombinación lineal Prob. P_c Recomb. α	Orientada Prob. P_m

SIMULATED ANNEALING

	Agitación térmica	Ley de aceptación	Curva de enfriamiento
AO5	- Distribución normal de un conjunto de puntos. - Se evalúa la ley de aceptación sobre el mejor del conjunto. - Puntos fuera del espacio de búsqueda se llevan al extremo más cercano.	Boltzman	$T(t) = T_0\alpha^t$

Tabla 3.1: Características de los algoritmos comparados.

2. En todos los casos se ha empleado un ranking lineal, otro tipo de ranking es adecuado, en algunos casos, cuando se tiene información de la función objetivo. En esta comparación se supone que no se tiene ninguna información acerca de la misma, se opta pues por el ranking lineal para evitar convergencias prematuras del algoritmo.

3. El operador *Stochastic Universal Sampling* también se utiliza en todos los casos porque presenta mejores cualidades que el de ruleta simple.

4. Se testea el operador de cruce uniforme con y sin mutación, AO2 y AO3 respectivamente. Se quiere evaluar si el operador de cruce uniforme presenta unas buenas cualidades de exploración del espacio de búsqueda evitando el operador de mutación, está cualidad sería importante para ahorrar tiempos de ejecución.

5. El algoritmo genético con codificación real (AO4) y el simulated annealing (AO5) que se han seleccionado corresponden a dos posibles implementaciones que se consideran razonables, las estructuras seleccionadas han mostrado un comportamiento correcto en unas pruebas previas.

Además de seleccionar los algoritmos, se deben ajustar los parámetros de cada uno de ellos. La tabla 3.2 recoge los parámetros para cada uno de los algoritmos:

- En los algoritmos genéticos **NIND** corresponde al número de individuos de la población. En AO5 corresponde al número de evoluciones a una temperatura (número de valores generados aleatoriamente a la misma temperatura).

- **MAXGEN** es el número máximo de generaciones en los algoritmos genéticos y el número de veces que se cambia de temperatura según la ley de enfriamiento en AO5 (se mantiene el mismo nombre de esta variable en los dos algoritmos puesto que tienen un significado similar). Estos valores se han asignado de manera que se mantenga constante el número de evaluaciones de la función objetivo respecto de los algoritmos genéticos.

- **NVAR** corresponde en todos los casos a la dimensión del espacio de búsqueda.

- Los parámetro T_i y σ corresponden a la temperatura inicial de OA5 y el coeficiente de ajuste del rango de la desviación estándar de la distribución Normal.

- P_m y P_c corresponden a las probabilidades de mutación y cruce respectivamente (en los algoritmos genéticos AO1 ... AO4.

- El **parámetro** α es en los casos AO2 y AO3 el coeficiente de recombinación del cruce uniforme, en el caso AO4 el coeficiente de recombinación lineal del operador cruce en la codificación real y en el caso AO5 se trata del coeficiente de la ley de enfriamiento.

Para ajustar los parámetros de cada uno de los algoritmos, se ha partido de un ensayo inicial con AO1 y la función F2. Para encontrar una solución más o menos adecuada, al algoritmo se le asignaron los parámetros indicados en la tabla 3.2 en la fila AO1. A partir de estos valores se han seleccionado los demás manteniendo constante el número de individuos, número máximo de generaciones, probabilidades de cruce y mutación (cuando era necesario) en todos los GAs. Para el algoritmo de *simulated annealing*, se ha mantenido el número total de evaluaciones de la función objetivo ($NIND \times MAXGEN$) y los demás parámetros se han ajustado experimentalmente hasta obtener un resultado razonable con la función F2.

Algoritmo	NIND	MAXGEN	NVAR	PRECI	P_c	P_m	α
AO1	500	35	2	16	0.7	0.02	-
AO2	500	35	2	16	0.7	0.02	0.5
AO3	500	35	2	16	0.7	-	0.5
AO4	500	35	2	Real	0.7	0.02	0.5
Algoritmo	NIND	MAXGEN	NVAR	PRECI	T_i	σ	α
AO5	35	500	2	Real	600	1/8	0.85
Número de repeticiones de los ensayos por función					50		

Tabla 3.2: Tabla de parámetros de los algoritmos.

Para poder comparar el coste computacional y la calidad de la solución de cada uno de los algoritmos se evalúan los siguientes parámetros.

Coste computacional:

- **Evaluaciones de la función objetivo**. Se ha optado por mantener constante este valor para todos los ensayos. $N_{eval} = NIND \times MAXGEN = 17500$. En el algoritmo de *Simulated Annealing* se ajusta la temperatura final en un valor muy bajo de manera que siempre se produzcan $MAXGEN$ cambios de temperatura, es decir, que no acabe el algoritmo antes de evaluar la función objetivo el mismo número de veces que en los algoritmos genéticos.
- **Tiempos de ejecución**. Se toman los tiempos totales del algoritmo (y en el caso de los algoritmos genéticos, además, la media y desviación estándar de las generaciones).
- **Operaciones en coma flotantes**. Se toma el número total (y en el caso de los algoritmos genéticos, además la media y desviación estándar de las generaciones).

Calidad de la solución:

- Distancia al mínimo real (x_o), $\|x - x_o\|$, x es el punto mínimo encontrado por el algoritmo.
- Distancia al valor mínimo de la función $|f(x) - f(x_o)|$, $f(x)$ es el valor de la función en el punto mínimo encontrado.

El análisis que se realiza compara, para cada función, todos los algoritmos, incluyendo los tiempos y Flops de los 50 ensayos, y la calidad de las soluciones.

3.1.1 Resultados con GA y SA

Función F1

CALIDAD DE LA SOLUCIÓN

	$\|x - x_o\|$		$\|f(x) - f(x_o)\|$	
	Media	Desv.	Media	Desv.
AO1	0.00063	0.00041	0	0
AO2	0.0019	0.0024	0.0000086	0.000025
AO3	0.00014	0.00022	0	0
AO4	0.000064	0.000077	0	0
AO5	0.0141	0.07	0.000025	0.00023

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	8.765	0.0617	0.57%	1987476.1	176.6	2.64%
AO2	12.722	0.1773	0.39%	2346684.1	3076.8	2.24%
AO3	10.923	0.1287	0.46%	1787548.1	2965.8	2.94%
AO4	6.846	0.0780	0.73%	1110699.4	3634.6	4.73%
AO5	1.086	0.0234	4.60%	580896.9	9.1	9.04%

Tabla 3.3: Resultados para la función F1.

Las medidas que se muestran en la tabla 3.3 corresponden a las medias y desviaciones estándar sobre los 50 ensayos realizados con cada algoritmo.

De la tabla se concluye que todos los algoritmos genéticos analizados presentan una precisión aceptable, la distancia al mínimo es inferior 0.002 en todos los caso. Cabe destacar la excelente calidad de AO4 con una precisión en la distancia al mínimo del orden de 0.00007 de media. AO5 presenta las peores soluciones, la precisión alrededor de 0.014 de media.

Se observa que la función con menor coste computacional es la AO5 con bastante diferencia sobre los demás algoritmos, este resultado es de esperar puesto que en los

algoritmos genéticos se deben realizar muchas más operaciones. El siguiente algoritmo interesante, desde este punto de vista, es el AO4.

Para tratar de evaluar el coste propio del algoritmo independiente de la función de coste se puede comparar el coste total tanto de tiempos como de operaciones en coma flotante (T_{total} y F_{total}) con el coste de evaluar la función objetivo 17500 veces ($NIND \times MAXGEN$) es $\Delta F = 52500$ Flops y $\Delta T = 0.05$ segundos. Los resultados que se obtienen se muestran en la tabla en porcentaje:

$$\frac{\Delta T}{T_{total}} \cdot 100$$
$$\frac{\Delta F}{F_{total}} \cdot 100$$

Se observa una elevada carga computacional debida al algoritmo. El coste temporal es muy elevado debido a la fragmentación de los programas. Cada algoritmo está compuesto por varias funciones programadas por separado para que sea más legible, las llamadas a estas funciones consumen bastante tiempo. En cuanto al coste en operaciones en coma flotante los algoritmos genéticos con codificación binaria tienen la carga mayor[3]. El GA con codificación real presenta menos carga que los de codificación binaria. El mejor algoritmo en cuanto a los costes computacionales es el *Simulated Annealing* (AO5).

[3]Esto es en parte debido a que se ha implementado en Matlab y cada bit del cromosoma es considerado como un número real.

Función F2

CALIDAD DE LA SOLUCIÓN

	$\lvert x - x_o \rvert$		$\lvert f(x) - f(x_o) \rvert$	
	Media	Desv.	Media	Desv.
AO1	0.0673	0.0544	0.0028	0.0544
AO2	0.1051	0.1072	0.0074	0.0125
AO3	0.1154	0.1297	0.0114	0.0272
AO4	0.0032	0.0045	0.00001	0.000024
AO5	0.0568	0.0348	0.0018	0.0025

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	8.879	0.448	0.67%	2057410.9	158.5	5.95%
AO2	12.661	0.445	0.47%	2416256.7	3349.3	5.07%
AO3	11.065	0.476	0.54%	1856553.9	3034.7	2.94%
AO4	6.752	0.315	0.89%	1180699.4	3634.6	6.60%
AO5	1.215	0.0312	4.94%	650918.8	9.0	18.82%

Tabla 3.4: Resultados para la función F2.

De la tabla 3.4 se observa que, para la función F2, el algoritmo que encuentra mejor solución es AO4, con una precisión en la distancia al mínimo del orden de 0.0032. Le siguen AO1 y AO5 con una precisión del orden de 0.06 y finalmente los peores resultados se obtienen con AO2 y AO3 .

De la tabla se puede concluir que se observan las mismas características de coste computacional que se observaban en F1. AO5 presenta un coste sensiblemente menor seguido por AO4. Coste computacional de evaluar F2 17500 veces: $\Delta T = 0.06$ seg y $\Delta F = 122500$ Flops.

Función F3

CALIDAD DE LA SOLUCIÓN

	$\lvert x - x_o \rvert$		$\lvert f(x) - f(x_o) \rvert$	
	Media	Desv.	Media	Desv.
AO1	0	0	0	0
AO2	0	0	0	0
AO3	0	0	0	0
AO4	0	0	0	0
AO5	0	0	0	0

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	8.863	0.065	0.56%	1969893.3	163.3	1.78%
AO2	12.860	0.098	0.39%	2329125.1	3418.9	1.50%
AO3	10.853	0.102	0.46%	1769419.6	2748.5	1.98%
AO4	6.421	0.049	0.79%	1467333.1	3954.7	2.38%
AO5	1.134	0.026	4.41%	563489.0	4.3	6.21%

Tabla 3.5: Resultados para la función F3.

Como se ve en la tabla 3.5, para la función F3 todos los algoritmos encuentran una solución exacta. Coste computacional de evaluar F3 17500 veces: $\Delta T = 0.05$ seg y $\Delta F = 35000$ Flops. Se observan las mismas características de coste computacional que se observaban en F1 y F2. El menor coste sigue correspondiendo a AO5 seguido por AO4.

Función F4

CALIDAD DE LA SOLUCIÓN

	$\lvert x - x_o \rvert$		$\lvert f(x) - f(x_o) \rvert$	
	Media	Desv.	Media	Desv.
AO1	0.0108	0.0075	0	0
AO2	0.1316	0.1473	0.0031	0.0179
AO3	0.0535	0.0937	0.000098	0.00045
AO4	0.0033	0.0016	0	0
AO5	0.1602	0.0794	0.000067	0.00019

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	24.712	0.611	59.32%	5479567.1	160.4	64.69%
AO2	28.615	0.135	51.06%	5839752.7	3139.6	60.70%
AO3	27.120	0.151	54.06%	5279672.0	2789.8	67.14%
AO4	22.889	0.151	64.05%	4977463.5	3813.4	71.24%
AO5	18.107	0.074	80.96%	4201457.6	9.0	84.37%

Tabla 3.6: Resultados para la función F4.

Para la función F4, AO4 encuentra claramente las mejores soluciones y los algoritmos AO2 y AO5 las peores (ver tabla 3.6). Coste computacional de evaluar F4 17500 veces: $\Delta T = 14.66$ seg y $\Delta F = 3544660$ Flops. Para esta función el menor coste sigue correspondiendo a AO5 seguido por AO4. Aunque hay que destacar que las diferencias entre los algoritmos son mucho menores, esto es debido a que el coste computacional de evaluar la función objetivo es muy alto. En estos casos el coste computacional del algoritmo ya no es tan relevante para la elección de un algoritmo adecuado.

Función F5

CALIDAD DE LA SOLUCIÓN

	$\|x - x_o\|$		$\|f(x) - f(x_o)\|$	
	Media	Desv.	Media	Desv.
AO1	0.00099	0.00086	0.000022	0.000032
AO2	0.0096	0.1	0.0022	0.004
AO3	0.0025	0.0053	0.0000064	0.0021
AO4	0.00025	0.00017	0	0
AO5	0.0222	0.0127	0.0074	0.0065

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	8.708	0.073	1.26%	2127408.0	224.6	9.05%
AO2	12.680	0.086	0.87%	2487149.6	2774.0	7.74%
AO3	11.039	0.108	1.00%	1926394.6	3124.3	9.99%
AO4	6.414	0.053	1.71%	1625108.5	3802.9	11.85%
AO5	1.282	0.024	6.02%	721422.1	10.1	26.69%

Tabla 3.7: Resultados para la función F5.

Para la función F5 AO4 y AO1 encuentra la soluciones muy buenas, los algoritmos AO2 y AO3 encuentran soluciones bastante buenas y finalmente AO5 presenta las soluciones con menor precisión (ver tabla 3.7). Coste computacional de evaluar F5 17500 veces: $\Delta T = 0.11$ seg $\Delta F = 192535$ Flops. Los costes computacionales siguen las mismas pautas que en los casos anteriores AO5 es el menos costoso seguido de AO4.

Función F6

CALIDAD DE LA SOLUCIÓN

	$\lvert x - x_o \rvert$		$\lvert f(x) - f(x_o) \rvert$	
	Media	Desv.	Media	Desv.
AO1	0.0282	0.0287	0.00023	0.00045
AO2	0.1175	0.1042	0.0031	0.0061
AO3	0.0913	0.1567	0.0041	0.0113
AO4	6.3873	3.7165	6.81	7.9112
AO5	684.3695	389.925	139.9733	91.6767

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	8.932	0.083	2.46%	2057402.2	189.6	5.95%
AO2	12.900	0.126	1.94%	2417407.5	2681.1	7.74%
AO3	11.339	0.179	1.70%	1856426.5	2962.1	6.66%
AO4	6.625	0.044	3.32%	1555834.8	4081.0	7.87%
AO5	1.405	0.029	15.66%	650958.2	9.4	18.82%

Tabla 3.8: Resultados para la función F6.

En el caso de la función F6 las mejores soluciones las encuentran los algoritmos genéticos con codificación binaria; AO1, AO2 y AO3. El algoritmo genético con codificación real AO4 encuentra soluciones peores y el AO5 presenta soluciones bastante malas.

Coste computacional de evaluar F6 17500 veces: $\Delta T = 0.22$ seg y $\Delta F = 122500$ Flops. En cuanto a los costes computacionales se observan comportamientos similares a los que se ven en las funciones anteriores.

Función F7

CALIDAD DE LA SOLUCIÓN

	$\|x - x_o\|$		$\|f(x) - f(x_o)\|$	
	Media	Desv.	Media	Desv.
AO1	4.3294	2.8007	0.0108	0.0055
AO2	4.2922	3.2756	0.0141	0.0102
AO3	3.1906	2.8178	0.0075	0.0053
AO4	1.3778	2.3122	0.0048	0.0032
AO5	9.0721	4.7010	0.0534	0.0293

COSTE COMPUTACIONAL

	Tiempo			Flops		
	T_{total}			F_{total}		
	Media (seg.)	Desv. (seg.)	$\frac{\Delta T}{T_{total}}$ (%)	Media (flops)	Desv. (flops)	$\frac{\Delta F}{F_{total}}$ (%)
AO1	8.714	0.072	1.26%	2109995.4	172.1	8.29%
AO2	11.705	0.131	0.94%	2469666.7	3200.5	7.74%
AO3	11.210	0.097	0.98%	1909682.5	2312.1	7.09%
AO4	6.382	0.049	1.72%	1608467.1	3612.1	10.88%
AO5	1.257	0.020	8.75%	703916.1	10.1	24.86%

Tabla 3.9: Resultados para la función F7.

Para esta función el algoritmo AO4 encuentra las mejores soluciones, los demás algoritmos genéticos (AO1, AO2 y AO3) encuentran similares peores que AO4 (tabla 3.9). El algoritmo AO5 tiene muchas dificultades para encontrar el mínimo global, aunque se queda en mínimos cercanos.

Coste computacional de evaluar F7 17500 veces: $\Delta T = 0.11$ seg y $\Delta F = 175035$ Flops. Los costes computacionales tienen las mismas características que en las funciones anteriores.

3.1.2 Resultados de otros algoritmos comerciales

Para tratar de evaluar con más garantías los algoritmos estudiados, se ha realizado una comparación con otros algoritmos comunes en los paquetes de optimización comerciales. En concreto se han utilizados los métodos:

- Método Quasi-Newton que utiliza para aproximar la matriz Hesiana la fórmula BFGS ([100]). No permite manipular restricciones, pero se puede aplicar en las funciones de test puesto que estas no tienen restricciones salvo los límites del espacio de búsqueda.
- Programación cuadrática secuencial (SQP) utilizando la fórmula BFGS para aproximar la matriz Hesiana del Lagrangiano ([100]). Este método si que incorpora el tratamiento de restricciones.

Estos dos métodos están implementados en el *Optimization Toolbox* de Matlab con las funciones: *fminu* y *constr* respectivamente. Se ha optado por la implementación de Matlab por mantener la homogeneidad de la plataforma de ensayos, los algoritmos de *Simulated Annealing* y Algoritmos Geneticos han sido programados en Matlab. Se podrían haber utilizado las funciones implementadas en *NAG Fortran Library* [73], este punto queda para trabajos posteriores puesto que se tiene previsto trasladar los algoritmos heurísticos a lenguaje C de cara a la implementación en línea de MBPC con optimización heurística.

Todas las opciones de los algoritmos se han mantenido en los valores por defecto propuestos, excepto el número máximo de evaluaciones de la función objetivo que se ha fijado en 500 (este valor es suficiente en todos los casos para que los algoritmos ensayados converjan).

Ambos algoritmos necesitan un punto inicial, este se ha generado aleatoriamente dentro del espacio de búsqueda.

Los test realizados corresponden a 50 ensayos con cada función de prueba. Los datos que se han tomado son la distancia de la solución encontrada por el algoritmo con el mínimo. Se han tomado también el número de evaluaciones de la función objetivo, el tiempo total de ejecución y el número de operaciones en coma flotante.

En las tablas 3.10 y 3.11 se observa que, como cabía esperar (ver capítulo 1), estos algoritmos funcionan bien cuando las funciones son convexas (funciones F1 y F2) pero en los demás casos la solución que obtienen es un mínimo local. Para los casos en que encuentran el mínimo, el coste computacional si que es claramente inferiores a los algoritmos de *Simulated Annealing* y Algoritmos Genéticos[4].

[4]Parte de esta ventaja puede ser debida a que se trata de un paquete comercial y puede estar mejor optimizado en su programación

Algoritmo Quasi-Newton

	Calidad		Coste computacional					
	$\lvert x - x_o \rvert$		Núm. Eval.		Tiempo (seg.)		Flops	
	Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.
F1	$2.1 \cdot 10^{-8}$	$2.5 \cdot 10^{-8}$	9	0	$1.44 \cdot 10^{-2}$	$2.47 \cdot 10^{-2}$	468.32	4.51
F2	$5.4 \cdot 10^{-5}$	$7.9 \cdot 10^{-5}$	188.24	100.9	$4.15 \cdot 10^{-1}$	$2.11 \cdot 10^{-1}$	$1.48 \cdot 10^{4}$	$7.12 \cdot 10^{3}$
F3	7.39	2.72	4	0	$7.4 \cdot 10^{-3}$	$1.86 \cdot 10^{-2}$	95	0
F4	37.09	22.63	51.62	35.47	$2.59 \cdot 10^{-1}$	$1.85 \cdot 10^{-1}$	$2.87 \cdot 10^{4}$	$1.98 \cdot 10^{4}$
F5	2.70	1.82	32.92	9.30	$7.44 \cdot 10^{-2}$	$3.08 \cdot 10^{-2}$	$2.67 \cdot 10^{3}$	$7.02 \cdot 10^{2}$
F6	667.36	313.29	31.68	7.47	$7.88 \cdot 10^{-2}$	$3.07 \cdot 10^{-2}$	$2.39 \cdot 10^{3}$	$6.16 \cdot 10^{2}$
F7	358.43	187.52	35.2	16.56	$7.74 \cdot 10^{-2}$	$4.38 \cdot 10^{-2}$	$2.92 \cdot 10^{3}$	$1.46 \cdot 10^{3}$

Tabla 3.10: Resultados con el algoritmo Quasi-Newton.

Algoritmo SQP

	Calidad		Coste computacional					
	$\lvert x - x_o \rvert$		Núm. Eval.		Tiempo (seg.)		Flops	
	Media	Desv.	Media	Desv.	Media	Desv.	Media	Desv.
F1	$3.7 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$	9.86	1.82	$4.84 \cdot 10^{-2}$	$1.87 \cdot 10^{-2}$	$1.03 \cdot 10^{3}$	$1.75 \cdot 10^{2}$
F2	$4.1 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	106.24	45.95	$4.28 \cdot 10^{-1}$	$1.77 \cdot 10^{-1}$	$1.30 \cdot 10^{4}$	$5.51 \cdot 10^{3}$
F3	7.44	2.83	4	0	$2.08 \cdot 10^{-2}$	$2.70 \cdot 10^{-2}$	$3.43 \cdot 10^{2}$	0
F4	30.80	16.16	104	43.7	$6.80 \cdot 10^{-1}$	$2.81 \cdot 10^{-1}$	$6.13 \cdot 10^{4}$	$2.55 \cdot 10^{4}$
F5	3.26	2.02	32.1	5.44	$1.32 \cdot 10^{-1}$	$2.91 \cdot 10^{-2}$	$4.00 \cdot 10^{3}$	$7.26 \cdot 10^{2}$
F6	687.43	323.57	36.88	13.56	$1.64 \cdot 10^{-1}$	$4.50 \cdot 10^{-2}$	$4.61 \cdot 10^{3}$	$1.11 \cdot 10^{3}$
F7	453.62	205.76	37.32	19.23	$1.46 \cdot 10^{-1}$	$6.71 \cdot 10^{-2}$	$4.30 \cdot 10^{3}$	$1.81 \cdot 10^{3}$

Tabla 3.11: Resultados con el algoritmo SQP.

3.1.3 Conclusiones

En la figura (3.2) se resumen los resultados obtenidos. Se clasifican los algoritmos de mayor a menor calidad en la solución encontrada (de 5 a 1) y de menor a mayor coste computacional (de 1 a 5). Se observa que el mejor algoritmo desde el punto de vista de la calidad es el AO4, Algoritmo Genético con codificación real, obtiene siempre la mejor calidad excepto en la función F6. Los algoritmos genéticos con codificación binaria presentan una calidad intermedia y por último el algoritmo de *Simulated annealing* es el que presenta peores resultados.

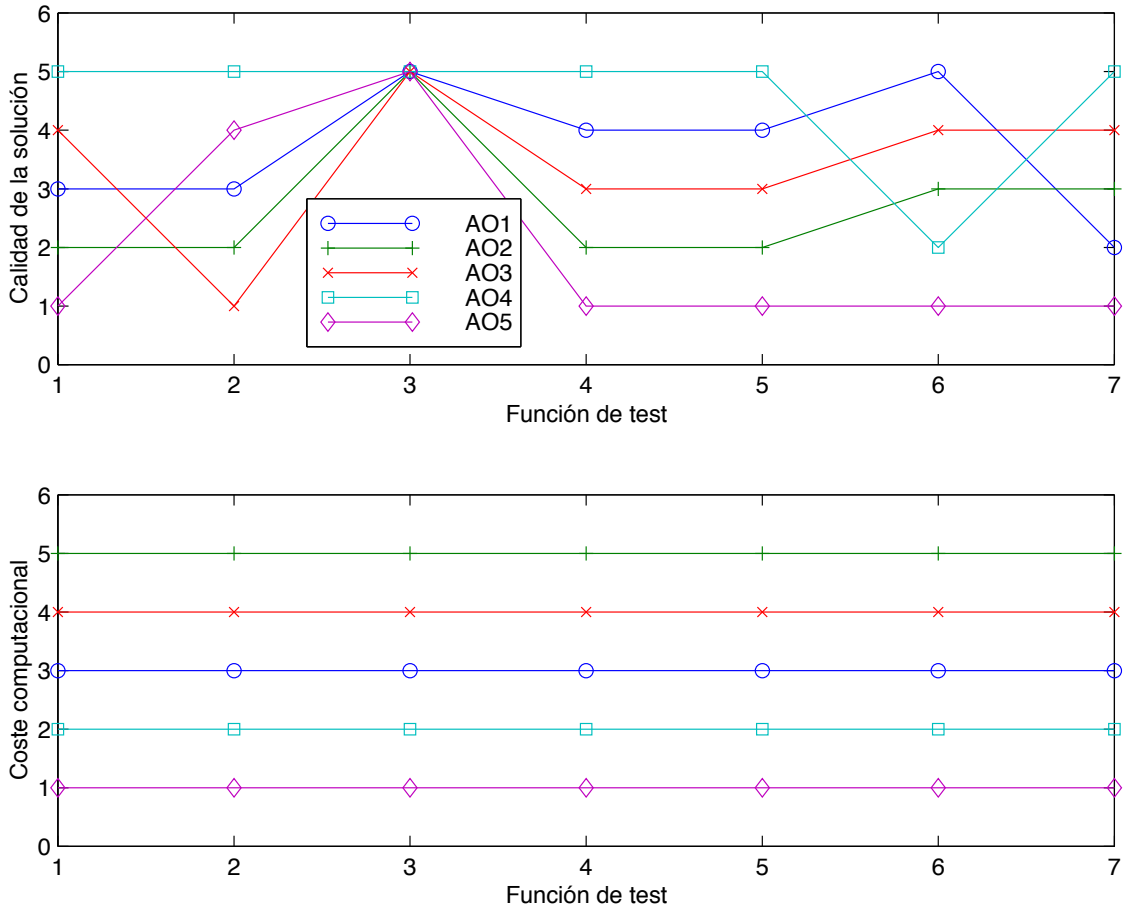


Figura 3.2: Comparación de la calidad de la solución y del coste computacional de los algoritmos $AO1 \ldots AO5$.

En cuanto al coste computacional se repiten los mismos resultados para todas la funciones: el algoritmo de *Simulated Annealing* (AO5) tiene un coste sensiblemente menor que los Algoritmos Genéticos. En cuanto a estos últimos el de codificación real (AO4) es el que presenta el menor coste computacional, seguido por AO1.

	Algoritmos				
	AO1	AO2	AO3	AO4	AO5
Media de la calidad para todas las funciones	3.7143	2.7143	3.4286	4.5714	2.0000
Media de coste para todas las función	3	5	4	2	1
Relación calidad coste	1.2381	0.5429	0.8571	2.2857	2.0000

Tabla 3.12: Relación calidad/coste para los algoritmos $AO1 \ldots AO5$.

En resumen el algoritmo AO4 presenta las mejores prestaciones calidad/coste (ver tabla 3.12), siendo un candidato para su aplicación al control predictivo cuando las funciones a minimizar que aparezcan presenten problemas. El algoritmo AO5, a pesar de que la calidad de la solución no es excesivamente buena, es competitivo debido a su bajo coste computacional, su campo de aplicación debe ser en los problemas complejos donde las restricciones temporales hacen que el algoritmo AO4 pierda demasiada calidad frente al algoritmo AO5.

Evidentemente el diseño de unas reglas de ajuste adecuadas para estos algoritmos facilitaría su aplicación a la resolución de problemas de control en general y de control predictivo en particular cuando las funciones a minimizar sean difíciles. El apartado siguiente se centra en el diseño de estas reglas de ajuste para el algoritmo de tipo AO4.

3.2 Evaluación de robustez y ajuste de parámetros de un GA real

El algoritmo que se va a evaluar es el Algoritmo Genético con codificación real con las siguientes características:

- Operación de *Ranking* lineal.
- Operador de selección: *Stochastic Universal Sampling.*
- Operador de cruce: Cruce por recombinación lineal con probabilidad de cruce P_c y parámetro de recombinación α se evalúa aleatoriamente para cada uno de los individuos sometidos al operador.
- Operador de mutación: mutación orientada, con probabilidad de mutación P_m.

Corresponde al algoritmo AO4 del apartado anterior con la variante de que el parámetro α del operador de cruce no es constante. Ya se ha visto que AO4 presentaba unas prestaciones muy altas de calidad y razonables en cuanto al coste computacional por tanto se trata de un buen candidato para su aplicación al control predictivo. En esta sección se pretende evaluar la robustez del algoritmo cuando se varían sus parámetros: número de individuos $NIND$, probabilidad de cruce P_c y probabilidad de mutación P_m. Se han realizado los siguientes ensayos:

- Número de individuos: se varía de 20 a 220 con un incremento de 20.
- Probabilidad de cruce: se varía de 0 a 1 con un incremento de 0.1.
- Probabilidad de mutación: se varia de 0 a 0.4 con un incremento de 0.05.

Para cada valor de $NIND$, P_c y P_m se realizan 10 ensayos.

La **calidad de la solución** obtenida se evalúa mediante $||x - x_o||$ y el **coste computacional** se mide con el número de evaluaciones de la función objetivo.

Para estos ensayos no se utiliza la función F1 puesto que se trata de una función con un sólo mínimo sin ninguna complicación y no justifica la utilización de una técnica de optimización heurística. Se ha podido ver anteriormente que otros métodos como el Quasi-Newton o el SQP resuelven el problema con muy buena calidad y bajo coste. En cualquier caso la función F2 posee una característica básica similar, ausencia de mínimo locales, aunque con la dificultad añadida de que le mínimo global se encuentra en una zona

muy plana. Esta función también es resuelta adecuadamente por los algoritmos Quasi-Newton y el SQP pero se incluye en esta comparación para analizar el comportamiento del Algoritmo Genético real ante problemas con un sólo mínimo.

Los criterios de finalización del algoritmo son dos (tabla 3.13):

1. Cuando el 95% de los individuos de la población están en un radio fijado ($DIST$) alrededor del mejor individuo de esa población (x_p). Se estima que en esas condiciones el algoritmo se ha extinguido, ha perdido su capacidad de exploración. El algoritmo se detiene cuando el 95% de los individuos de la población cumplen:

$$||x - x_p|| < DIST$$

   El valor de $DIST$ se asigna dependiendo de la precisión que se desea. En este caso se relaciona con la precisión de un Algoritmo Genético binario (puede servir para realizar comparaciones). En un Algoritmo Genético binario, la precisión viene dada por el número de bits que se le asignan a cada variable ($PRECI$), por ejemplo, con $PRECI = 16$ se discretiza el espacio de búsqueda en 2^{16} puntos y por tanto la máxima precisión que se puede obtener a partir del rango del espacio de búsqueda (Δr) y $PRECI$:

$$\epsilon = \frac{\Delta r}{2^{PRECI}}$$

   Para el criterio de finalización el valor del radio se fija en cada caso según la precisión:

$$DIST = 4\epsilon$$

2. Por número máximo de generaciones, $MAXGEN = 500$. En caso de que el algoritmo no se extinga (no se cumpla nunca la condición anterior) este criterio evita que se este ejecutando indefinidamente.

Los datos que se obtienen quedan recogidos en [10] donde aparecen además de los datos, unas representaciones gráficas que muestran las relaciones entre calidad de la solución, coste computacional y convergencia frente a los parámetros: P_c, P_m con $NIND$ constante, P_c, $NIND$ con P_m constante y P_m, $NIND$ con P_c constante, estas representaciones permiten realizar un análisis cualitativo de las distintas relaciones entre los parámetros pero es muy complejo extraer conclusiones más concretas. Para posibilitar la realización del análisis se ha condensado la información en tablas. Estas reflejan las relaciones entre los tres parámetros que se han estudiado, P_c, P_m y número de individuos $NIND$ en base a la calidad de la solución obtenida, criterios de convergencia y coste computacional.

Para cada función de test se presentan dos tablas (apéndice C), la primera nos da para cada probabilidad de mutación y de cruce, por una parte, el rango de individuos que

	Funciones de test					
	F2	F3	F4	F5	F6	F7
u_{min}	-5.12	-5.12	-65.536	-5.12	-500	-600
u_{max}	5.12	5.12	65.536	5.12	500	600
Δr	10.24	10.24	131.072	10.24	1000	1200
$PRECI$	16	16	17	16	20	20
ϵ	$1.6 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	0.001	$1.6 \cdot 10^{-4}$	$9.5 \cdot 10^{-4}$	0.00114
$DIST$	$6.25 \cdot 10^{-4}$	$6.25 \cdot 10^{-4}$	0.004	$6.25 \cdot 10^{-4}$	0.0038	0.00456
$MAXGEN$	500	500	500	500	500	500

Tabla 3.13: Parámetros para los criterios de finalización del algoritmo: $DIST$ y $MAXGEN$.

cumple el criterio de calidad de la solución ($E(||x - x_o||) < \epsilon$) y por otra parte el rango de inviduos en que se da una convergencia del algoritmo mayor de un determinado factor F_c (cumpliendo simultáneamente el criterio de calidad). La segunda tabla ofrece los costes computacionales medios (para los 10 ensayos) correspondientes al límite inferior, media y limite superior respectivamente de cada uno de los rangos de la primera tabla. El coste computacional se contabiliza en miles de evaluaciones de la función objetivo y sólo cuenta el número de evaluaciones de la función objetivo hasta que se encuentra la mejor solución. En el caso de que el algoritmo no converja el coste total del mismo es superior (o igual) al coste que se muestre en la tabla correspondiente puesto que se realizan en todos los caso $MAXGEN \times NIND$.

El factor ϵ corresponde al que aparece en la tabla 3.13, esto puede permitir la comparación con un Algoritmo Genético binario con cromosomas de longitud $2 \times PRECI$ (las funciones son de dos variables). Las tablas se han formado a partir de la media de $||x - x_o||$ de los 10 ensayos que se han realizado para cada P_c, P_m y $NIND$.

Para la función F7 no existe ningún conjunto de parámetros (P_c, P_m, $NIND$) que satisfaga la condición $E(||x - x_o||) < \epsilon$. Con el fin de realizar un análisis del comportamiento del algoritmo con las condiciones del ensayo se ha relajado la condición. Se ha considerado como condición $E(||x - x_o||) < 0.1$, si el mejor valor que se obtiene está en ese entorno, un optimizador de tipo *Hill climbing* encontraría el óptimo global. Los óptimos más cercanos al óptimo global de la función F7 se encuentran en:

$$\begin{aligned} x = (3.14, 4.4385) &\rightarrow F7(x) = 0.0074 \\ x = (3.14, -4.4385) &\rightarrow F7(x) = 0.0074 \\ x = (-3.14, -4.4385) &\rightarrow F7(x) = 0.0074 \\ x = (-3.14, 4.4385) &\rightarrow F7(x) = 0.0074 \end{aligned}$$

La distancia entre estos óptimos locales y el óptimos globales es $||x - x_o|| = 5.4369$, por tanto, establecer la condición $E(||x - x_o||) < 0.1$ para ésta función no parece descabellado para la posterior localización del óptimo local de forma rápida.

El factor de convergencia F_c se calcula de la siguiente forma. Para un ensayo, se asigna a una variable $Conv$ los siguientes valores:

- $Conv = 1$ si el algoritmo ha finalizado cuando el 95% de la población se encuentra en un entorno establecido (criterio de finalización 1).
- $Conv = 0$ si el algoritmo finaliza con el criterio del número máximo de generaciones (criterio de finalización 2).

Para obtener F_c se realiza la media de $Conv$ para todos los ensayos realizados (en este caso 10).

$$F_c = E(Conv)$$

La interpretación de este factor es: $F_c = 1$ indica que el algoritmo converge en el 100% de los casos, $F_c = 0$ el algoritmo no converge en ningún caso, $F_c = 0.5$ el algoritmo converge en el 50% de los casos, etc.

En las tablas se muestran los rangos de número de individuos, para cada P_m y P_c, que satisfacen:

$$F_c \geq 0.9$$

A continuación se presentan las conclusiones que se obtiene para cada función, a partir de la información que aparece estas tablas.

Función F2

Para esta función el operador de cruce resulta fundamental, por debajo de $P_c = 0.3$ el algoritmo tiene dificultades para encontrar soluciones adecuadas sobre toda con valores de $P_m < 0.2$ aunque mejora un poco para valores mayores de P_m. Con probabilidades de cruce elevadas el algoritmo encuentra la solución, sólo presenta dificultades en combinación con P_m elevadas.

La convergencia sólo se da para valores de $P_m \leq 0.1$. Es más fácil converger cuanto más pequeño es el valor de P_m. Se observa que para $P_m = 0.1$ un número de individuos elevado impide la convergencia del algoritmo.

Se observa que el coste computacional es sensiblemente menor si el algoritmo converge. El coste disminuye en general si la probabilidad de mutación disminuye o la probabilidad de cruce aumenta.

La zona que presenta unas buenas cualidades de calidad/coste es pues:

- $0.4 \leq P_c \leq 1$
- $0 \leq P_m \leq 0.1$
- $100 \leq NIND \leq 220$

Para este rango de valores el coste está alrededor de 10.000 evaluaciones de la función objetivo.

En cualquier caso, para funciones unimodales como es el caso de la función F2 (ver figura B.3) es mejor utilizar otros algoritmos de optimización con menor coste.

Función F3

El algoritmo encuentra el óptimo global en todos los casos excepto para el caso $P_c = 0$ y $P_m = 0$. En casi los casos, el rango de individuos es $[20, 220]$, sólo se reduce el rango cuando la probabilidad de mutación es nula o muy baja.

En cuanto a la convergencia, el algoritmo sólo tiene un factor superior a 0.9 para $P_c = 0.1$ y $P_m = 0.05$, en definitiva, probabilidades de cruce y mutación muy bajas. Se puede generalizar que para este tipo de funciones el algoritmo no converge según el criterio de convergencia que se ha establecido.

El motivo es que la función tiene zonas planas (ver figura B.5), tanto los óptimos locales como el óptimo global coinciden con zonas y no un punto único punto. En este caso existen muchos individuos con el mismo valor de la función y salvo que la zona para el criterio de finalización sea de un tamaño comparable a la zona que corresponde a los óptimos es muy difícil que el 95% de los individuos se concentren en un radio pequeño. Una alternativa para conseguir una convergencia del algoritmo es modificar el criterio de finalización. Se puede establecer que el algoritmo también finalice cuando, por ejemplo, el 95% de los individuos tengan el mismo valor de la función, es decir, que esos individuos se encuentran en la misma zona.

El coste para esta función es muy bajo comparado con el de las demás funciones y disminuye cuanto mayores son las probabilidades de mutación y cruce.

La zona que presenta unas buenas cualidades de calidad/coste es pues:

- $0.2 \leq P_c \leq 1$
- $0.05 \leq P_m \leq 0.4$
- $20 \leq NIND \leq 220$

Para este rango de valores el coste está alrededor de 300 evaluaciones de la función objetivo.

Función F4

El algoritmo encuentra siempre soluciones adecuadas excepto para combinaciones de P_c y P_m de valores bajos. El aumento de la probabilidad de mutación aumenta el rango de individuos para los que se encuentra solución, aunque este rango vuelve a disminuir levemente cuando se dan combinaciones de P_c y P_m de elevado valor.

La convergencia depende básicamente de la probabilidad de mutación para valores de $P_m \leq 0.1$ el algoritmo converge para un rango bastante amplio de número de individuos (empeorando si el número de individuos es muy elevado) para $0.1 < P_m \leq 0.2$ el algoritmo converge sólo para valores muy bajos del número de individuos (20 o 40).

El coste computacional en las zonas en que no converge el algoritmo es elevado. Cuando no converge el algoritmo, las combinaciones de probabilidades de mutación bajas y probabilidades de cruce muy altas o muy bajas incrementan el coste computacional.

La zona que presenta unas buenas cualidades de calidad/coste es pues:

- $0.3 \leq P_c \leq 1$
- $0 \leq P_m \leq 0.2$
- $40 \leq NIND \leq 160$

Para este rango de valores el coste está alrededor de 6.000 evaluaciones de la función objetivo.

Se observa que para estas funciones con los mínimos en depresiones muy pronunciadas (ver figura B.7) el algoritmo no tiene demasiados problemas para encontrar el mínimo global, al menos si el valor en el mínimo global es sensiblemente menor que en los demás mínimos locales. El algoritmo encuentra una solución adecuada para casi todos los rangos de valores de P_c y P_m, y converge en un amplio rango, aunque si P_m es muy baja, el número mínimo de individuos debe se mayor de 140 para tener ciertas garantías.

Función F5

El algoritmo encuentra una solución en todos los casos excepto para combinaciones de P_c y P_m con valores muy bajos, en general con un rango de individuos bastante amplio, el límite superior es siempre 220 y el inferior varia entre 40 y 160 individuos por generación.

Para la convergencia del algoritmo el parámetro significativo es la probabilidad de mutación, $P_m \geq 0.15$ impiden la convergencia del algoritmo. Con $P_m \leq 0.05$ el algoritmo

encuentra la solución siempre convergiendo. Con $P_m = 0.1$ el número de individuos influye en la convergencia, a más individuos menor probabilidad de converger.

El coste computacional en las zonas en que no converge el algoritmo es mayor que cuando converge. En general, probabilidades de mutación alta incrementan el coste computacional.

La zona que presenta unas buenas cualidades de calidad/coste es pues:

- $0.3 \leq P_c \leq 1$
- $0 \leq P_m \leq 0.1$
- $60 \leq NIND \leq 200$

Para este rango de valores el coste está alrededor de 6.000 evaluaciones de la función objetivo.

Para este tipo de funciones (ver figura B.9) el algoritmo tampoco tiene problemas, encuentra solución adecuada para un amplio rango de valores de los parámetros.

Función F6

Para este tipo de función el aumento de la probabilidad de cruce tiene un efecto negativo, para $P_c > 0.5$ el algoritmo no encuentra solución. Este comportamiento, que sólo ocurre para esta función de test, parece debido a que la función presenta muchos óptimos locales y varios de estos óptimos locales tiene un valor de la función muy parecido o comparable al óptimo global pero se encuentra muy alejado del mismo (ver figura B.11).

En estas circunstancias la operación de cruce tiene dificultades para hacer converger el algoritmo, por ejemplo, el cruce entre dos individuos muy buenos pero muy alejados (uno en el entorno del óptimo global y otro en el entorno del óptimo local) produce individuos que pierden gran parte de la información para alcanzar el óptimo global. Por lo tanto, si se producen muchos cruces es muy complicado que los individuos de la población se parezcan al óptimo global.

La probabilidad de mutación influye en el rango de individuos en que se encuentra una solución, un aumento de esta probabilidad provoca una aumento del rango salvo si se combina con una probabilidad de cruce elevada (en este caso $P_c = 0.5$.

En cuanto a la convergencia, el comportamiento es similar a las demás funciones, un aumento excesivo de la misma impide la convergencia del algoritmo, para $P_m > 0.1$ no converge nunca.

El coste computacional en las zonas en que no converge el algoritmo es ligeramente mayor que cuando converge. En general, probabilidades de mutación alta incrementan el coste computacional.

La zona que presenta unas buenas cualidades de calidad/coste es pues:

- $0 \leq P_c \leq 0.5$
- $0.05 \leq P_m \leq 0.1$
- $100 \leq NIND \leq 220$

Para este rango de valores el coste está alrededor de 7.000 evaluaciones de la función objetivo.

Función F7

Esta función tiene muchos óptimos locales pero los que tienen valor parecido al óptimo global se encuentran cerca del mismo (ver figura B.13), por tanto no se produce el efecto que se observaba en la función anterior. En este caso se encuentran soluciones adecuadas para valores muy elevados de P_c, siendo mejores los resultados si se aumenta P_m aunque a partir de un valor de 0.35 se aprecia un leve empeoramiento.

La convergencia se produce como siempre para valores pequeños de P_m.

El coste computacional en las zonas en que no converge el algoritmo es mayor que cuando converge. En general, probabilidades de mutación alta incrementan el coste computacional.

La zona que presenta unas buenas cualidades de calidad/coste es pues:

- $0.7 \leq P_c \leq 1$
- $0 \leq P_m \leq 0.05$
- $160 \leq NIND \leq 220$

Para este rango de valores el coste está alrededor de 11.000 evaluaciones de la función objetivo.

Para este tipo de funciones en los que existen muchos óptimos locales y todos muy juntos es conveniente incrementar la capacidad de exploración del algoritmo. Mantener valores de P_c y P_m altos y es muy aconsejable que el número de individuos de la población sea elevado aunque esto revierte en un incremento del coste computacional.

3.3 Propuesta de una metodología de ajuste

El objetivo de esta evaluación ha sido conseguir un rango de parámetros P_c, P_m y $NIND$ que proporcione buenas prestaciones tanto en calidad como en coste, todo ello para un rango de funciones los más amplio posible. Del análisis de las tablas de resultados anteriores para cada una de las funciones se extraen las conclusiones que se presentan a continuación[5].

1. **Coste computacional**
   El primer resultado a destacar es que si el algoritmo converge el coste computacional es menor que si no converge. Los tres parámetros estudiados facilitan la convergencia con:

   - Una probabilidad de cruce baja.
   - Una probabilidad de mutación baja.
   - Pocos individuos por generación.

   De estos parámetros el que se ha destacado por su mayor influencia es la probabilidad de mutación.

2. **Calidad de la solución** Evidentemente no sólo hay que asegurar la convergencia, sino la convergencia al óptimo global, esto influye en que es necesaria una correcta exploración del espacio de búsqueda para poder obtener una solución con una calidad aceptable. En este sentido, influyen negativamente en la exploración del espacio de búsqueda y por tanto en la calidad de la solución:

   - Una probabilidad de cruce baja.
   - Una probabilidad de mutación baja.
   - Pocos individuos por generación.

   Esta conclusión se extrae de las tablas anteriores, las casillas que están vacías corresponden a combinaciones de los parámetros que no consiguen solución adecuada. En general, se trata pues de los mismos valores que influyen positivamente en la convergencia del algoritmo. Una combinación de parámetros que explora exhaustivamente el espacio de búsqueda dificulta e incluso imposibilita la convergencia del algoritmo y viceversa.

[5] La validez de las conclusiones que se presentan en este apartado quedan restringidas a funciones de dos variables y al Algoritmo Genéticos descrito.

Por tanto los objetivos de exploración del espacio de búsqueda y convergencia del algoritmo son en general incompatibles, deben de ponderarse correctamente para alcanzar una relación adecuada de estos factores. En definitiva, de lo que se está hablando es de la relación entre la calidad de la solución y el coste.

Los dos objetivos se pueden conseguir de forma aceptable mediante un buen ajuste de los parámetros del algoritmo. La tabla 3.14 muestra los rangos de los parámetros en los que se consigue una solución adecuada con el menor coste para las funciones de test utilizadas, es decir, son los rangos para los que se obtiene una buena calidad y el algoritmo converge (coste bajo). Además en la misma tabla se incluye información de cada una de las funciones acerca de las características del problema de optimización que se plantea con cada función. Más adelante se propone una metodología de ajuste del Algoritmo Genético con codificación real descrito en base a la información que se muestra en la tabla

Otra característica que se puede derivar del análisis de las tablas es que el Algoritmo Genético con codificación real es bastante robusto, esta conclusión se extrae de la amplitud de los rangos de valores de parámetros en los que se obtienen soluciones de calidad. Esta característica facilita enormemente el ajuste del algoritmo. Además se pueden ampliar los rangos que se muestran en la tabla sobre todo los de número de individuos y/o la probabilidad de mutación pero siempre a cambio de un coste superior. Como recomendación general, para la mayoría de las funciones, es peor aumentar excesivamente la probabilidad de mutación que el número de individuos o la probabilidad de cruce. No es aconsejable en ningún caso que la probabilidad de mutación supere $P_m = 0.2$.

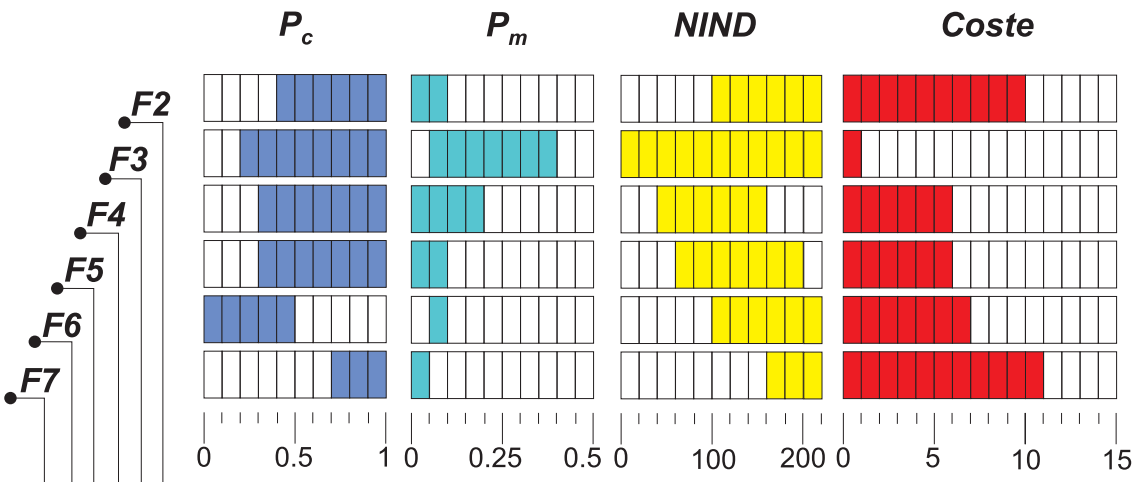
A partir de la tabla 3.14 se pueden extraer varias reglas para el ajuste del algoritmo:

1. Para el caso en el que se conozcan algunas características de la función se puede tratar de relacionar las características de dicha función con las funciones de test que se presentan (la tabla 3.14 indica brevemente las problemáticas de cada función) y realizar un ajuste de los parámetros según en los intervalos sugeridos por las tablas, ver ejemplo 3.1.

2. En el caso de que no se tenga ninguna información a priori acerca de la función a minimizar, el ajuste de los parámetros que se aconseja para este algoritmo es:

   - $P_c = 0.6$
   - $P_m = 0.1$
   - $NIND = 120$

   Este conjunto de parámetros es el que presenta unos resultados razonables para todas las funciones.



F2	Función unimodal con el mínimo en una zona complicada.
F3	Función multimodal. Tipo escalones, zonas completamente planas.
F4	Función multimodal. Óptimos localizados en depresiones profundas aunque con claras diferencias entre los valores de los óptimos.
F5	Función multimodal. Forma acanalada.
F6	Función multimodal. Óptimos con valor de la función similar al óptimo global pero alejados del mismo.
F7	Función multimodal. Gran cantidad de óptimos y muy juntos.

Tabla 3.14: Rango de ajuste aconsejados para los parámetros P_c, P_m y $NIND$ de un Algoritmo Genético con codificación real, si se conocen a priori algunas características del problema de minimización.

Ejemplo 3.1 Metodología de ajuste de un GA con codificación real
Se quiere ajustar un Algoritmo Genético para obtener el mínimo de una función que consiste en la función de coste de tipo valor absoluto de un control predictivo. El proceso viene modelado por un doble integrador y un backlash y el índice de coste está formado por el valor absoluto de los errores en el horizonte de predicción, este ejemplo se detalla en el capítulo correspondiente a las no linealidades en los actuadores, aquí sólo se pretende mostrar el procedimiento de ajuste de un Algoritmo Genético con codificación real.

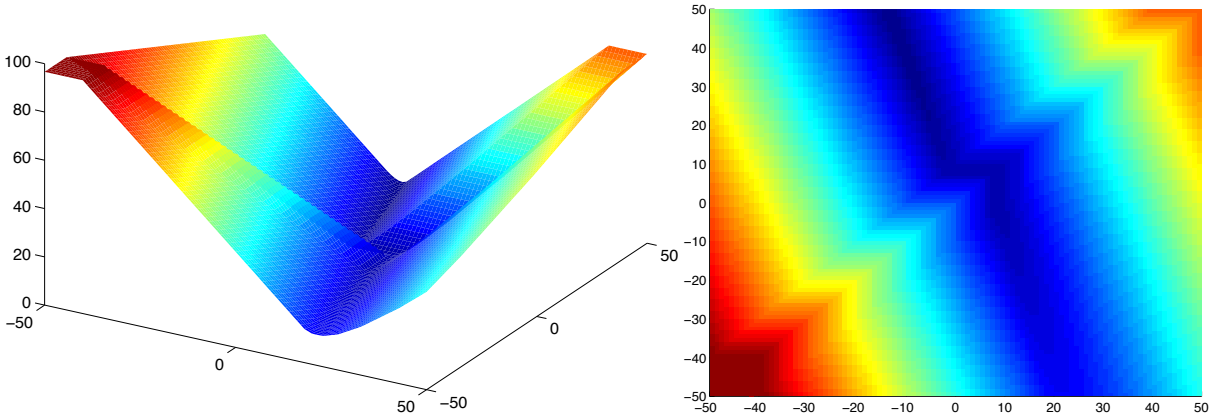


Figura 3.3: Representación gráfica de la función a minimizar en el ejemplo 3.1.Vistas 3D y superior.

Puesto que se dispone de la función a minimizar se pueden realizar varias representaciones gráficas para lo que serían distintos instantes de muestreo y disponer de la forma que puede presentar dicha función. Un ejemplo de la función se muestra en la figura 3.3 y a partir de esta representación se pueden ajustar los parámetros del Algoritmo Genético.

De la representación gráfica se observa que la función parece unimodal pero puede tener ciertos problemas en una franja diagonal debido al backlash. En principio, este problema se puede relacionar con la función de test F2 (función de Rosenbrock) que también es unimodal y el mínimo se encuentra en una zona problemática. Para esta función, unos valores adecuados de los parámetros del Algoritmo Genético son (tabla 3.14):

- $0.4 \leq P_c \leq 1$
- $0 \leq P_m \leq 0.1$
- $100 \leq NIND \leq 220$

Tomando unos valores intermedios en estos rango: $P_c = 0.7$, $P_m = 0.05$ *y* $NIND = 150$. *Se obtienen los siguientes resultados:*

- *El algoritmo converge en 38 generaciones, estos supone un coste computacional de 5.7 miles de evaluaciones de la función objetivo. Este valor coincide con las estimaciones de coste que se presentan en la tabla 3.14.*
- *El algoritmo converge al mínimo global con una buena calidad (error* $< 10^{-4}$*), el valor encontrado es* $x = [36.5791, -10.2632]$.

Tomando unos valores fuera de estos rangos: $P_c = 0.2$, $P_m = 0.25$ *y* $NIND = 40$. *Se obtienen los siguientes resultados:*

- *El algoritmo no converge, se detiene por la limitación de generaciones cuando llega a 500 generaciones. El coste es de 20 miles de evaluaciones de la función objetivo. Este valor es netamente superior al coste cuando el algoritmo converge (ver tabla 3.14.*
- *La mejor solución encontrada es* $x = [35.9049, -10.0841]$, *que corresponde a* $||x - x_{min}|| = 0.6976$, *es decir, la calidad de la solución es baja.*

3.4 Conclusiones

Con el estudio que se ha realizado en el capítulo anterior, se disponen de alternativas contrastadas para la resolución de los problemas de optimización multimodal, no convexa y/o con discontinuidades que puedan aparecen en el MBPC. En este capítulo se han evaluado las implementaciones potencialmente mas adecuadas mediante un estudio comparativo de las mismas según se indica en el artículo *Designing and reporting on computational experiments with heuristic methods* del *Journal of Heuristics*.

Concretamente se comparan cinco algoritmos obteniendo órdenes de magnitud tanto en el coste computacional como en la calidad de la solución, todo ello para un conjunto de siete funciones de test que tratan de simular un amplio rango de problemas de optimización.

De estos ensayos se puede concluir que:

1. Entre los algoritmos testeados el que presenta mejor relación calidad/coste es el algoritmo genético con codificación real.

2. El peor algoritmo en cuanto a la calidad obtenida es el de *simulated annealing*, sin embargo es el de menor coste computacional.

3. Los algoritmos genéticos basados en una codificación binaria consiguen calidades intermedias entre el de codificación real y el algoritmo de *simulated annealing*, pero presentan los peores costes computacionales.

En el estudio se ha intentado, además, indicar el coste propio del algoritmo y el de evaluación de la función a minimizar. Se ve que si esta función es muy costosa computacionalmente, el coste propio del algoritmo tiene menor importancia en el coste total. Esta conclusión puede afectar a la elección de un Algoritmo Genético frente al de *Simulated Annealing*. La ventaja principal de este último es su bajo coste computacional pero esta cualidad no es tan relevante en problemas en que el coste de la evaluación de la función a optimizar es alto.

A pesar del mayor coste de los Algoritmos Genéticos basados en la codificación binaria, estos pueden ser de utilidad en su aplicación al control de proceso. En un entorno de control digital el conversor D/A presenta una cuantificación determinada (por ejemplo, 12 bits), si se plantea el problema de optimización con la longitud de cada uno de los parámetros del cromosoma correspondiente a la del conversor D/A puede resultar un problema de optimización menos costoso. El algoritmo genético con codificación binaria puede llegar a ser competitivo en estos caso. Este aspecto no ha sido tratado en este trabajo y puede plantearse en un futuro análisis.

Para completar la comparación se han efectuado ensayos con las mismas funciones de test y algoritmos comerciales usuales: SQP (programación cuadrática secuencial) y el QN (Quasi-Newton). La conclusión principal es que estos algoritmos son muy potentes para los casos unimodales sin problemas de discontinuidades o no convexidad (funciones F1 y F2) pero tiene serias dificultades con las demás funciones. Por tanto, se aconseja el uso de estos algoritmos frente a los estudiados únicamente en el caso unimodal y si las funciones no presentan discontinuidades y son convexas.

De todo este estudio se destaca que el Algoritmo Genético con codificación real presenta el mejor comportamiento, pero quedaba por resolver el problema del ajuste de sus parámetros y el análisis de la robustez del mismo. En la última parte del capítulo se tratado con detalle este punto.

El estudio evalúa el algoritmo genético con codificación real propuesto con los siguientes parámetros variables: Número de individuos de la población, probabilidad de cruce y probabilidad de mutación. Para los experimentos se utilizan 6 funciones de test (ya usadas en los experimentos anteriores). Para cada función y combinación de parámetros (número de individuos, probabilidad de cruce y de mutación) se realizan los experimentos necesarios para conseguir un fiabilidad adecuada de los resultados.

Se elabora una tabla 3.14 con los rangos de ajuste aconsejados para los parámetros P_c, P_m y $NIND$ de un Algoritmo Genético con codificación real, si se conocen a priori algunas características del problema de minimización, y se muestra un ejemplo de utilización.

Capítulo 4

Control predictivo basado en modelos con optimización heurística

El objetivo fundamental de la propuesta de un MBPC utilizando optimización heurística es tratar de aprovechar todo el potencial que encierra la propia técnica MBPC. En ese sentido, la inclusión de un optimizador heurístico adecuado evita ciertas limitaciones que tienen algunas de las implementaciones, sobre todo con la aparición de funciones de coste multimodales y discontínuas. La herramienta posibilita:

- Utilizar índices de coste y estructuraciones de la ley de control no convencionales que incrementen las prestaciones del controlador.
- Utilizar modelos de predicción más exactos (uso de toda la información del proceso que esté disponible), incorporando las no linealidades del proceso en el modelo y las restricciones tanto en las acciones de control como en las variables del proceso.

Este capítulo se centra en la descripción de esta propuesta de control predictivo basado en modelos con alguna técnica de optimización heurística y mostrar la flexibilidad que aporta para establecer una estructuración de la ley de control alternativa y/o un índice de coste diferente. En el capítulo siguiente se tratará el aspecto referente a la inclusión de modelos no lineales y las restricciones en las acciones de control y variables del proceso.

La estructura general de un control predictivo con optimización heurística se muestra en la figura 4.1. Para cada particularización de los elementos que componen el controlador se obtiene una alternativa diferente, así variando el modelo de predicción y/o el índice de coste se pueden obtener prestaciones diferentes. Como optimizador heurístico se puede utilizar cualquiera de las técnicas que se han descrito en el capítulo anterior.

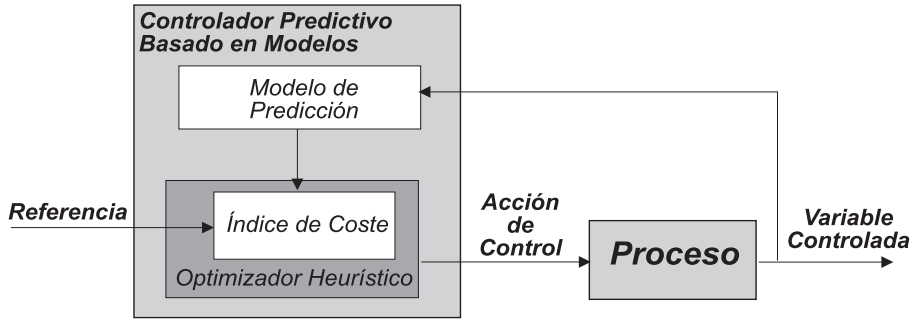


Figura 4.1: Estructura del control predictivo con técnica de optimización heurística.

La propuesta que se realiza se particulariza al Controlador Predictivo Generalizado (GPC) [29], ya que incluye modelo del proceso y modelo de perturbaciones, y el índice de coste incorpora gran parte de las posibilidades y avances que se han incluido en las últimas generaciones de control predictivo. Como algoritmo de optimización heurística se describen tanto *Simulated Annealing* como Algoritmos Genéticos con codificación real y binaria, cualquiera de estas técnicas puede ser aplicada al MBPC variando la relación coste computacional/calidad de la solución tal y como se ha visto en el capítulo anterior. En general, la mejor relación la obtiene el Algoritmo Genético con codificación real y los operadores genéticos descritos en el capítulo anterior. Si no existen limitaciones de coste computacional se puede conseguir ajustar cualquiera de los algoritmos para que den la misma calidad en la solución.

Esta configuración del MBPC permite describir la integración de una técnica de optimización heurística con uno de los tipos de control predictivo más completo, y a partir de esta implementación se pueden realizar multitud de variantes y proponer mejoras en todos los aspectos (modelo, indice de coste y optimizador).

4.1 GPC con optimización heurística

Siguiendo la estructura general de la figura 4.1 se debe definir el índice de coste y el modelo de predicción que se van a utilizar en este caso.

4.1.1 Índice de coste

El índice de coste que se utiliza es el clásico para un GPC:

$$J(\Delta U) = \sum_{j=N_1}^{N_2} [y(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda_j [\Delta u(t+j-1)]^2 \qquad (4.1)$$

Donde $\Delta U = [\Delta u(t), ..., \Delta u(t+N_u-1)]^T$, vector de los incrementos de la acción de control futuras respecto del cual se debe minimizar el índice de coste. Evidentemente el índice depende de varios parámetros como: N_1, N_2, N_u y λ pero estos se ajustan antes de realizar la minimización.

4.1.2 Modelo de predicción

En el esquema de un GPC la predicción $y(t+j|t)$ se obtiene a partir de un modelo CARIMA:

$$A(z^{-1})y(t) = B(z^{-1})u(t-1) + C(z^{-1})\xi(t)/\Delta \tag{4.2}$$

En general, es difícil obtener un modelo adecuado del polinomio $C(z^{-1})$ pero se sustituye por un polinomio de diseño $T(z^{-1})$ que ajustado adecuadamente incrementa las cualidades de robustez y atenuación de posibles perturbaciones [65].

$$A(z^{-1})y(t) = B(z^{-1})u(t-1) + T(z^{-1})\xi(t)/\Delta \tag{4.3}$$

Donde:

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{-n_a} \\ B(z^{-1}) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + a_{n_b} z^{-n_b} \\ T(z^{-1}) &= t_0 + t_1 z^{-1} + t_2 z^{-2} + \cdots + t_{n_t} z^{-n_t} \\ \Delta &= 1 - z^{-1} \end{aligned}$$

Representado en forma de diagrama de bloques se tiene la figura 4.2. El modelo se compone de una parte que corresponde al modelo del proceso y otra al modelo de perturbaciones

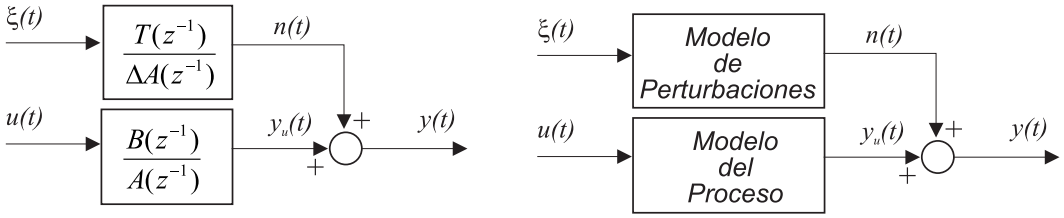


Figura 4.2: Diagrama de bloques del modelo CARIMA.

La formulación clásica de un GPC $y(t+j|t)$ se obtiene mediante la resolución de determinadas ecuaciones diofánticas. Este método, que ofrece una formulación compacta del regulador, está restringido a modelos lineales sin restricciones. En los cálculos del regulador se pierde la separación entre la parte del modelo que corresponde a las perturbaciones y el resto de elementos. Para poder incrementar las posibilidades de aplicación

a otros modelos e índices de coste es necesario una formulación más general del cálculo de las predicciones. Una forma lógica de flexibilizar la utilización de distintos tipos de modelos es independizar las predicciones del modelo del proceso de las predicciones del modelo de perturbaciones.

Cálculo de las predicciones A continuación se presenta un método alternativo que no necesita la resolución de las ecuaciones diofánticas mencionadas. Se basa en la resolución numérica separando el cálculo de la salida del modelo del proceso y la salida del modelo de perturbaciones. Como se verá más adelante, esta alternativa permitirá utilizar la estructura de un GPC en procesos con modelos no lineales [64].

La predicción de la salida en el instante '$t+j$' con la información disponible hasta el instante 't', $y(t+j|t)$, viene dada por:

$$y(t+j|t) = y_u(t+j|t) + n(t+j|t) \tag{4.4}$$

Donde $y_u(t+j|t)$ se obtiene del modelo del proceso y las acciones de control futuras, y $n(t+j|t)$ se obtiene del modelo de perturbaciones.

- Cálculo de $y_u(t+j|t)$:
  Si el modelo del proceso es una función de transferencia (como es el caso en la formulación del GPC), $y_u(t+j|t)$ se obtiene recursivamente mediante la ecuación en diferencias 4.5 con los datos conocidos hasta en el instante 't':

  $$A(z^{-1})y_u(t+j|t) = B(z^{-1})u(t+j-1) \tag{4.5}$$

  Esta predicción es independiente del modelo de perturbaciones y se puede sustituir, si se considera oportuno, por cualquier tipo de modelo no linealidad que genere $y_u(t+j|t)$.

- Cálculo de $n(t+j|t)$:
  $n(t)$ depende de una señal no medible, $\xi(t)$, pero con propiedades estadísticas establecidas, se trata de un ruido blanco:

  $$n(t) = \frac{T(z^{-1})}{\Delta A(z^{-1})}\xi(t) \tag{4.6}$$

  Para simplificar los cálculos posteriores se puede reescribir la ecuación 4.6, realizando el cambio de variable $n^f(t) = n(t)/T(z^{-1})$:

  $$n^f(t) = \frac{1}{\Delta A(z^{-1})}\xi(t) \tag{4.7}$$

El cambio de variable se interpreta como el resultado de filtrar la señal $n(t)$ con el filtro $1/T(z^{-1})$ (tanto este filtro como $T(z^{-1})$ son realizables). En cualquier momento se puede realizar el cambio inverso y obtener $n(t) = T(z^{-1})n(t)^f$.

Para obtener la mejor predicción de n^f se parte de la ecuación en diferencias siguiente:

$$n^f(t) = (\Delta A(z^{-1}))' n^f(t) + \xi(t) \tag{4.8}$$

Donde:

$$\begin{aligned} \Delta A(z^{-1}) &= 1 + \hat{a}_1 z^{-1} + \hat{a}_2 z^{-2} + \cdots + \hat{a}_{n_a+1} z^{-(n_a+1)} \\ (\Delta A(z^{-1}))' &= -\hat{a}_1 z^{-1} - \hat{a}_2 z^{-2} - \cdots - \hat{a}_{n_a+1} z^{-(n_a+1)} \end{aligned}$$

La predicción para el siguiente instante de tiempo será:

$$n^f(t+1) = (\Delta A(z^{-1}))' n^f(t+1) + \xi(t+1) \tag{4.9}$$

Como $\xi(t+1)$ es desconocido pero de media cero, la mejor predicción que se puede realizar es, considerar $\xi(t+1) = 0$:

$$\begin{aligned} n^f(t+1|t) &= (\Delta A(z^{-1}))' n^f(t+1|t) \\ &= -\hat{a}_1 n^f(t) - \hat{a}_2 n^f(t-1) - \ldots - \hat{a}_{n_a+1} n^f(t-n_a) \end{aligned} \tag{4.10}$$

Donde $n^f(t-j)$, $j \geq 0$ (últimos valores de n^f) se pueden obtener por filtrado utilizando $1/T(z^{-1})$. Los valores $n(t-j)$ no son más que la diferencia entre los valores medidos de la salida y los valores que resultan del modelo del proceso:

$$n(t-j) = y(t-j) - y_u(t-j) \tag{4.11}$$

$$n^f(t-j) = \frac{n(t-j)}{T(z^{-1})} \tag{4.12}$$

Hay que recordar que el polinomio $T(z^{-1})$ es conocido puesto que es fijado por el diseñador.

La predicción a dos instantes de muestreo vista queda:

$$n^f(t+2) = (\Delta A(z^{-1}))' n^f(t+2) + \xi(t+2) \tag{4.13}$$

De nuevo, puesto que $\xi(t+2)$ es desconocido pero de media cero, la mejor predicción que se puede realizar corresponde a:

$$\begin{aligned} n^f(t+2|t) &= (\Delta A(z^{-1}))' n^f(t+2|t) = -\hat{a}_1 n^f(t+1|t) - \\ &\quad -\hat{a}_2 n^f(t) - \hat{a}_3 n^f(t-1) - \ldots - \hat{a}_{n_a+1} n^f(t-n_a+1) \end{aligned} \tag{4.14}$$

Donde $n^f(t-j)$, $j \geq 0$ se obtiene de la misma forma que en la ecuación (4.12) y $n^f(t+1|t)$ se ha calculado en la predicción anterior.

Generalizando, la predicción para el instante '$t+j$' es:

$$\begin{aligned} n^f(t+j|t) &= (\Delta A(z^{-1}))' n^f(t+j|t) = -\hat{a}_1 n^f(t+j-1|t) - \\ &\quad -\hat{a}_2 n^f(t+j-2|t) - \ldots - \hat{a}_{j-1} n^f(t+1|t) - \hat{a}_j n^f(t) - \\ &\quad -\hat{a}_{j+1} n^f(t-1) - \ldots - \hat{a}_{n_a+1} n^f(t+j-n_a-1) \end{aligned} \tag{4.15}$$

A partir de $n^f(t+j|t)$ se obtiene la mejor predicción de la salida del modelo de perturbaciones $n(t+j|t)$:

$$n(t+j|t) = T(z^{-1}) n^f(t+j|t) \tag{4.16}$$

4.1.3 Adaptación de una técnica de optimización

Una vez se ha establecido el índice de coste y el método para obtener las predicciones de la salida, el cálculo de las acciones de control que minimiza el índice se reduce a un problema de optimización con o sin restricciones.

$$\begin{aligned} &\min_U (J(U)) \\ &U = [u(t), u(t+1), \ldots, u(t+N_u-1)] \end{aligned}$$

Para el caso de un control SISO, la dimensión del problema de optimización es N_u.

En el GPC se puede obtener una solución analítica a este problema, pero si se utilizan otro tipo de índices o el modelo de proceso no es lineal, generalmente es imposible obtener una solución analítica del problema. En estos casos la alternativa que se propone es la aplicación de una técnica de optimización heurística como los Algoritmos Genéticos o *Simulated Annealing*, que como se ha mostrado en el capítulo anterior permiten abordar con garantías una amplia gama de problemas.

Para resolver el problema de minimización del MBPC en cada periodo de muestreo:

- La técnica de *Simulated Annealing* sólo requiere la función a minimizar $J(U)$ y un ajuste adecuado de los parámetros del optimizador. Como adaptación concreta del optimizador, se puede utilizar como punto de inicial del algoritmo la mejor solución obtenida en el instante de muestreo anterior (mejor solución del problema de optimización anterior) en lugar de un punto aleatorio. Esta adaptación puede distorsionar los algoritmos CSA y FSA puesto que utilizan un único punto en cada evolución, pero no afecta al algoritmo ASA puesto que intervienen varios puntos en cada evolución.

- Para el caso de aplicar Algoritmos Genéticos, tanto con codificación binaria como real, es necesario además decidir cómo se estructuran los cromosomas.

  Si se elige un Algoritmo Genético con codificación binaria, se debe decir, por una parte los bits dedicados a cada variable dependiendo de la precisión que se quiere alcanzar y por otra donde se sitúa cada variable en el cromosoma. La opción elegida en todos los casos (para control SISO) es la que se muestra en la figura 4.3, se sitúan los parámetros consecutivamente en el cromosoma.

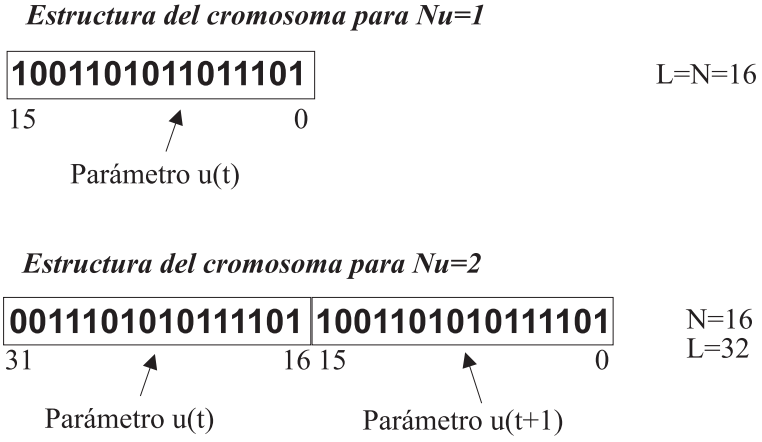


Figura 4.3: Estructura de los cromosomas en una codificación binaria (N: bits por parámetro, L: longitud del cromosoma). Ejemplo para $N_u = 1$ y $N_u = 2$.

  Si el Algoritmo Genético utiliza una codificación real, la estructura del cromosoma es del mismo tipo, se sitúan los distintos parámetros consecutivamente: $[u(t), u(t+1), \ldots, u(t+N_u-1)]$. Evidentemente en este caso no hay que elegir el número de bits para codificar cada parámetro.

  Una adaptación de estos algoritmos que permite mejorar la respuesta del controlador predictivo consiste en introducir, en la población inicial de cada periodo de muestreo, un mínima cantidad de duplicados del mejor individuo del periodo de muestreo anterior. La cantidad de individuos no debe ser muy alta para mantener la diversidad de la población inicial, con uno o dos individuos es suficiente.

  En la descripción del Algoritmo Genético que se realizó en un capítulo anterior, la población inicial es totalmente aleatoria, esto es adecuado para la resolución de un problema aislado. En el caso del MBPC, el problema de minización en un instante de muestreo está más o menos relacionado con el problema del instante de muestreo anterior. El objeto de introducir una pequeña cantidad de información genética de los mejores individuos del problema anterior permite asegurar que el algoritmo pueda realizar una exploración de estas zonas (además de la exploración de otras zonas) sin depender de factores aleatorios. Esto mejora sensiblemente el comportamiento,

al menos en régimen permanente, y no empeora el régimen transitorio. Cuando se trabaja con poblaciones con pocos individuos (para conseguir coste computacionales bajos) esta adaptación mejorar la respuesta tanto en régimen permanente como en régimen transitorio, es por tanto, una aportación en la reducción de los costes computacionales del algoritmo de optimización.

En el anexo A se incluye una validación de esta estructura para el caso de un GPC clásico (modelo lineal con índice cuadrático), en ese caso no está justificada la incorporación de optimización heurística pero se trata de verificar si la estructura propuesta es equivalente a la clásica para, a partir de ella, incorporar las mejoras que se presentarán en los apartados y capítulos siguientes.

4.2 Incremento de las prestaciones por modificación del índice

Se presentan en este apartado dos características que pueden adaptarse de forma sencilla al planteamiento MBPC con optimización heurística:

- Redistribución de la acción de control.
- Índices de coste distintos del cuadrático.

Se persigue alcanzar mejores prestaciones en el control, manteniendo la estructura de control predictivo.

4.2.1 Redistribución de la acción de control.

En este apartado se introduce una variación sobre el índice de minimización convencional que aporta un grado de libertad más en el ajuste del controlador. Como se verá, esta modificación puede permitir incrementar la calidad del control. Esta redistribución no es exclusiva de una estructura MBPC con optimización heurística pero es más simple e intuivo introducirlo si se utiliza un método heurístico, en el apartado siguiente se muestra cómo se puede incorporar en la formulación analítica de un GPC aunque queda restringido a procesos lineales y con indice de coste cuadrático.

La modificación que se realiza en la minimización del índice es una redistribución no convencional de las acciones de control en el horizonte de predicción. Cuando se realiza una minimización con, por ejemplo, $N_u = 2$ (el horizonte de control es dos, es decir, sólo

se permiten dos cambios en la acción de control) se supone que las acciones de control futuras (en el horizonte de predicción) se van a aplicar consecutivamente de la siguiente forma (ver figura 4.4):

$$\begin{aligned} u(t) &= u_0 \\ u(t+1) &= u_1 \\ u(t+2) &= u_1 \\ u(t+3) &= u_1 \\ &\dots \end{aligned}$$

A partir del segundo cambio en la acción de control ésta se mantiene constante. Esto mismo se puede expresar en forma de incrementos en la acción de control:

$$\begin{aligned} \Delta u(t) &= u_0 - u_{-1} \\ \Delta u(t+1) &= u_1 - u_0 \\ \Delta u(t+2) &= 0 \\ \Delta u(t+3) &= 0 \\ &\dots \end{aligned}$$

Otra alternativa es distribuir estas acciones de una forma diferente como (ver figura 4.4):

$$\begin{aligned} u(t) &= u_0 \\ u(t+1) &= u_0 \\ u(t+2) &= u_0 \\ u(t+3) &= u_1 \\ u(t+4) &= u_1 \\ u(t+5) &= u_1 \\ &\dots \end{aligned}$$

Escrito como incrementos:

$$\begin{aligned} \Delta u(t) &= u_0 - u_{-1} \\ \Delta u(t+1) &= 0 \\ \Delta u(t+2) &= 0 \\ \Delta u(t+3) &= u_1 - u_0 \\ \Delta u(t+4) &= 0 \\ \Delta u(t+5) &= 0 \\ &\dots \end{aligned}$$

Para distinguir esta distribución de la acción de la distribución clásica podemos plantear la siguiente notación, para el caso de dos cambios en la acción de control la notación para la distribución clásica es $N_u = 2$. Para las redistribuciones alternativas es necesario indicar la posición del segundo cambio en la acción de control, esto se puede indicar de la siguiente forma: $N_u = 1.xx$ donde xx es un entero que indica la posición del segundo cambio. En el ejemplo de la figura 4.4 $N_u = 1.3$ indica que el cambio en la acción de control se produce en el instante '$t + 3$'.

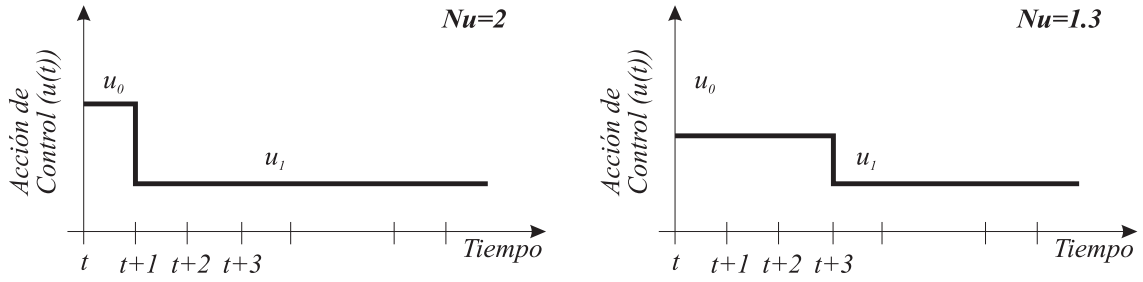


Figura 4.4: Redistribución de la acción de control.

Se mantiene la primera acción de control durante más tiempo. Este tipo de redistribución permite dar más importancia en la minimización a la acción de control que realmente se va a aplicar al proceso, es decir, la primera (con el *receding horizon* sólo se aplica la primera acción de control). Esto es debido a que permite mantener dicha acción de control durante más instantes de muestreo y por tanto influye más en el índice.

En la figura 4.5 se compara el efecto del aplicar distintas distribuciones de las acciones de control. El proceso que se ha utilizado es el doble integrador muestreado a 10 Hz. Los parámetros del controlador predictivo con algoritmo genético son:

- $N_u = 2$ (para $C2$, $C3$, y $C4$) y $N_u = 1$ (para $C1$)
- $N_1 = 1$, $N_2 = 10$.
- 25 generaciones con 150 individuos y 32 bits por cromosoma (codificación binaria).

Las redistribuciones aplicadas son:

- $C1$: $[u_0, u_0, u_0, u_0, u_0, u_0, u_0, u_0, u_0, u_0]$.
- $C2$: $[u_0, u_0, u_0, u_0, u_0, u_0, u_1, u_1, u_1, u_1]$.
- $C3$: $[u_0, u_0, u_0, u_1, u_1, u_1, u_1, u_1, u_1, u_1]$.
- $C4$: $[u_0, u_1, u_1, u_1, u_1, u_1, u_1, u_1, u_1, u_1]$.

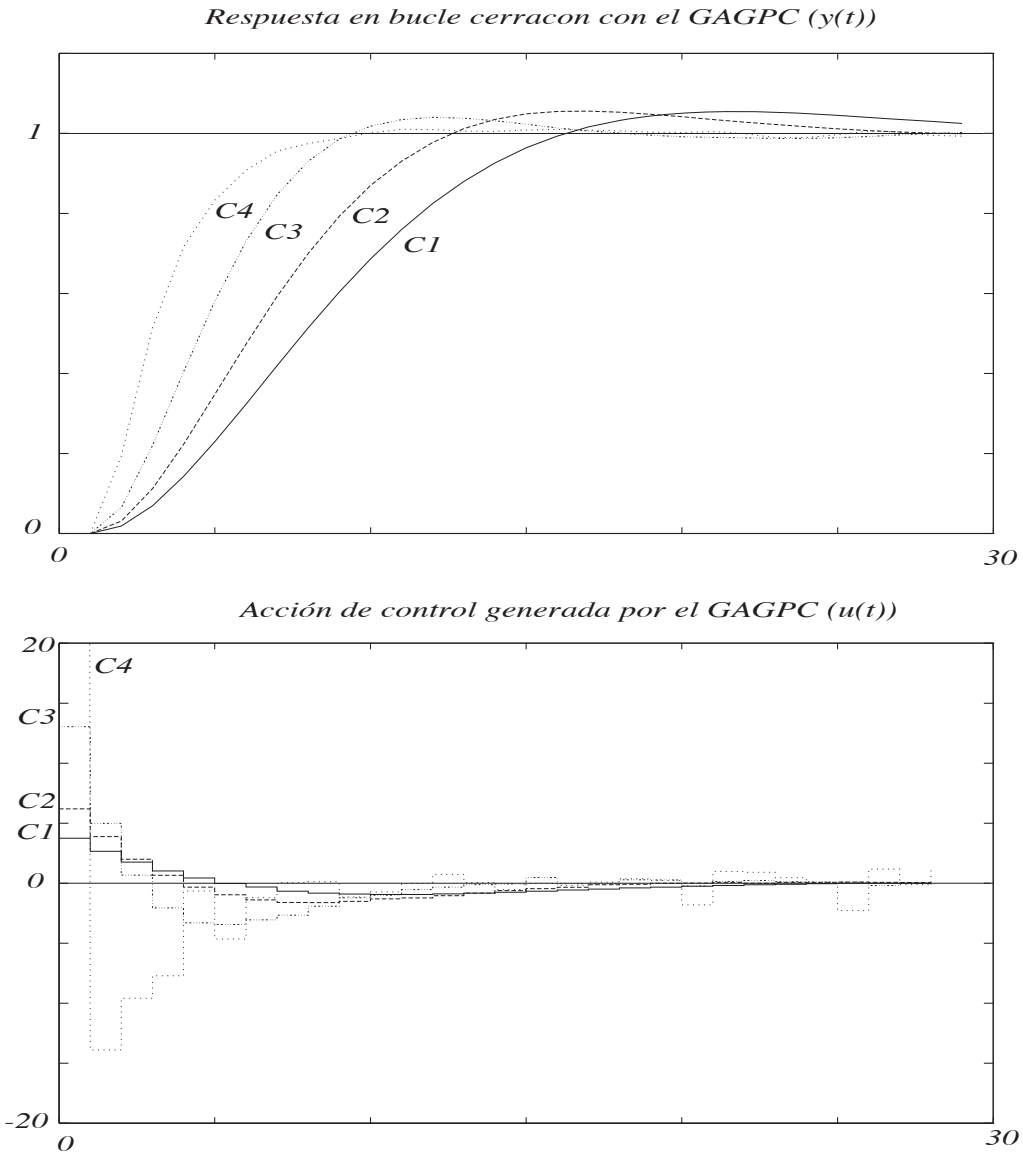


Figura 4.5: Control del doble integrador con distinta redistribución de la acción de control.

Cuanto mayor sea el número de u_0 que se permiten (es la acción de control que se va a aplicar realmente puesto que se utiliza un horizonte móvil), se observa una respuesta más suave (acciones de control menos enérgicas). Con esta técnica se puede decir que el horizonte de control esta entre $N_u = 1$ y $N_u = 2$. Evidentemente, esta técnica no está restringida a $N_u = 2$ se puede realizar con cualquier valor de $N_u > 1$. La notación para varios cambios de la acción de control puede ser: $N_u = 1.3.7$, es decir, se produce cambios en la acción de control en los instantes '$t+3$' y '$t+7$'.

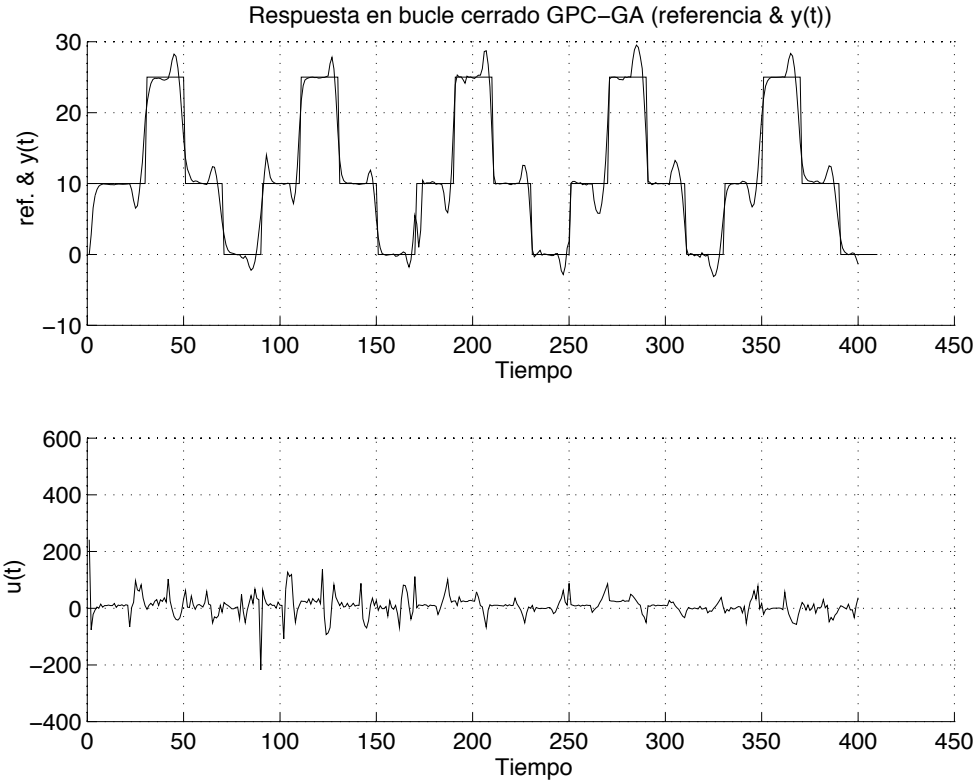


Figura 4.6: Controlador predictivo genético con $N_u = 2$, el controlador conoce los cambios en las referencias futuras. Control de seguimiento de trayectorias predeterminadas.

Como ejemplo de aplicación de este índice se trata de resolver el problema que aparece en la figura 4.6. Corresponde al mismo control que el de la figura A.2 (ver anexo A) pero en este caso el controlador conoce las referencias futuras. En el caso de la figura A.2 el controlador suponía que las referencias futuras son la misma que en el instante 't'. Para esta simulación (figura 4.6) se observan oscilaciones no deseadas cuando varía la referencia.

Para paliar este efecto se propone una modificación del índice de coste aplicando una redistribución diferente de las acciones de control. En lugar de permitir variaciones en las

dos primeras acciones (sólo existen dos variaciones puesto que se trabaja con un horizonte de control de $N_u = 2$), se posibilita que las variaciones no sean consecutivas. Las figuras 4.7 y 4.8 muestran los resultados que se obtienen con redistribuciones como $C2$ y $C3$ respectivamente. Con esto se obtiene respuestas más suaves, cuanta más importancia se da a la acción de control inicial la respuesta se parece más a un control con $N_u = 1$, más lenta y sobreamortiguada.

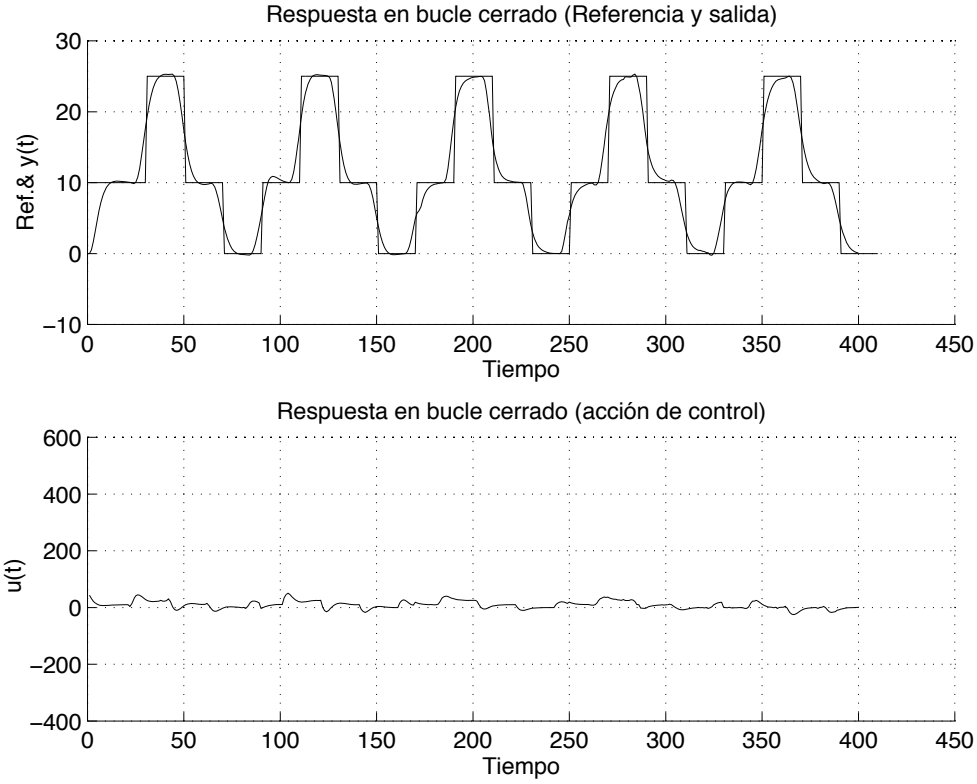


Figura 4.7: Controlador predictivo genético con $N_u = 2$ con el índice modificado, redistribución de las acciones de control según $C2$.

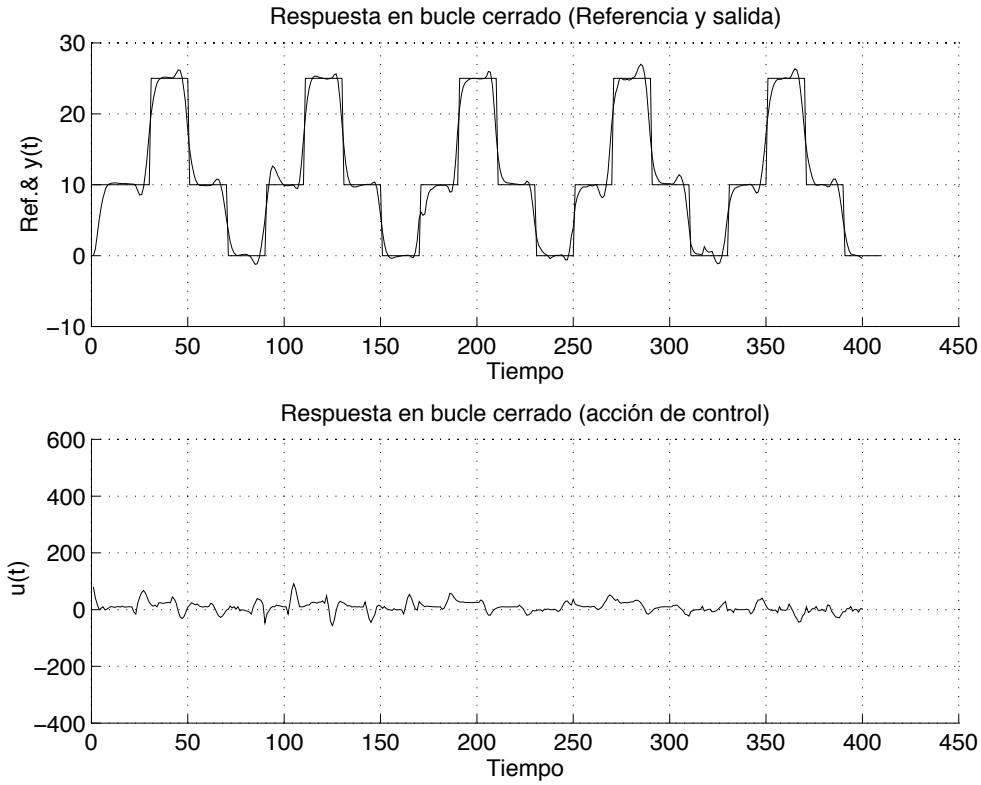


Figura 4.8: Controlador predictivo genético con $N_u = 2$ con el índice modificado, redistribución de las acciones de control según $C3$.

Además se puede combinar una redistribución de la acción de control con la ponderación de los errores de predicción, el índice sería:

$$J_{GPC}(u) = \sum_{j=N_1}^{N_2} \alpha_j e^2(t+j|t) \tag{4.17}$$

Este índice permite ponderar la importancia de cada uno de los errores de predicción. Si se desea que el error futuro no tenga tanta importancia se pondera con un valor menor. Los resultados obtenidos combinando esta índice y con una redistribución de la acción de control se muestran en la figura 4.9. La ponderación utilizada es $\alpha_j = 0.5^{j-1}$ y la redistribución $[u_0, u_0, u_1, u_1, u_1, u_1, u_1, u_1, u_1, u_1]$. Con esto se consigue un mejor seguimiento de la referencia.

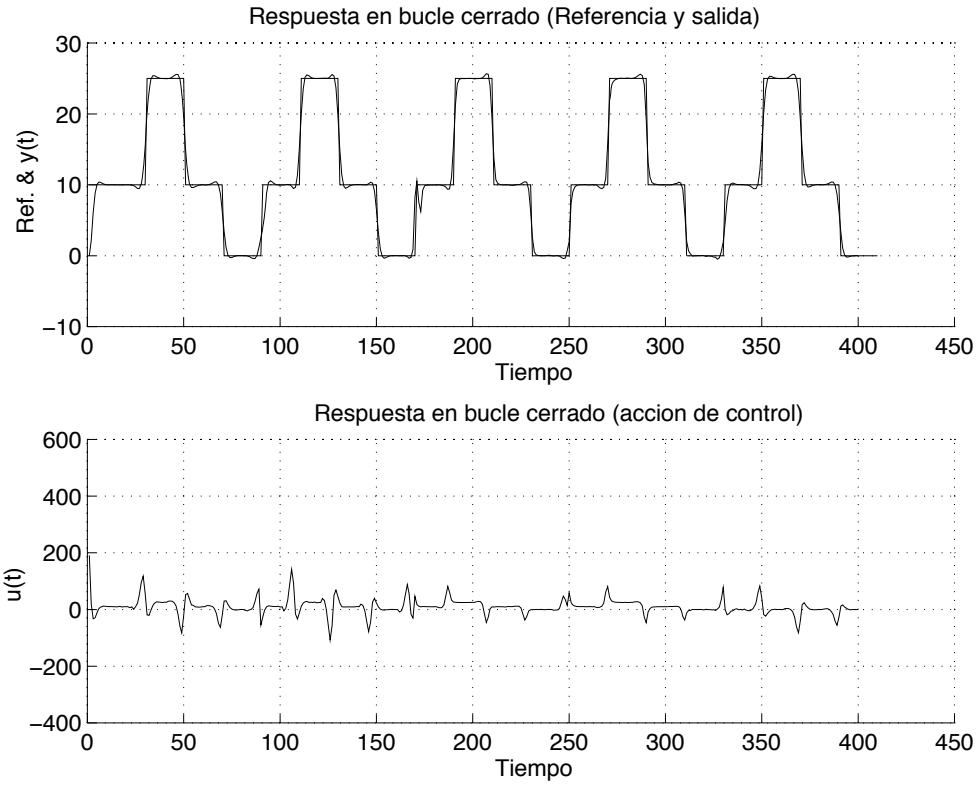


Figura 4.9: Controlador predictivo genético con $N_u = 2$ con el índice modificado.

4.2.2 Redistribución de la acción de control en un GPC lineal

Como se comenta en el apartado anterior la redistribución de la acción de control fuerza incrementos de la acción de control nulos en instantes no consecutivos, no es más que una nueva alternativa de estructuración de la ley de control.

A partir del planteamiento general que se sigue en el desarrollo de un GPC lineal cuyo índice de coste viene dado por:

$$J = \sum_{i=N_1}^{N_2} [y(t+i|t) - w(t+i)]^2 + \sum_{j=1}^{N_u} \lambda_j [\Delta u(t+j-1)]^2 \tag{4.18}$$

Escrito de forma matricial, se tiene:

$$\begin{aligned} J &= (Y-W)^T A (Y-W) + \hat{u}^T \lambda \hat{u} \\ Y &= Gu + \Gamma \Delta U^f + F Y^f \end{aligned}$$

$$Y = \begin{pmatrix} y(t+N_1|t) \\ y(t+N_1+1|t) \\ \cdots \\ y(t+N_2|t) \end{pmatrix} \quad ; \quad u = \begin{pmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \cdots \\ \Delta(t+N_u) \end{pmatrix}$$

Y minimizando este índice se obtiene la ley de control a lo largo del horizonte de predicción (ver [88]):

$$\hat{u} = (G^T A G + \lambda)^{-1} G^T A (W - \Gamma \Delta U^f - F Y^f) \tag{4.19}$$

En este caso, para incorporar la redistribución de la ley de control que se describe en el apartado anterior se presentan dos alternativas:

1. **Reajuste de los parámetros N_u y λ_j.**

   La primera, y quizá la más intuitiva, consiste en la asignación de los coeficientes de ponderación de las acciones de control $\lambda_j = \infty$ para los $\Delta u(t+j-1)$ que se quiera forzar cero e incrementar el horizonte de control $N_u = N_{u1} \leq N_2$ que se debe

interpretar como horizonte de control en el que se producen todos los $\Delta u(t+j-1) \neq 0$ [1].

El efecto de esta alternativa se puede analizar con un ejemplo, sín pérdida de generalidad, para el caso $N_2 = 3$ y $N_1 = 1$, se trata de permitir dos cambios en la acción de contro pero forzando $\Delta u(t+1) = 0$ y por tanto $N_u = 3$. Para el desarrollo que sigue se asigna $\lambda_2 = N$, $N \to \infty$:

$$G^T A G = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \; ; \; \lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & N & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

$$G^T A G + \lambda = \begin{pmatrix} a_{11}+\lambda_1 & a_{12} & a_{13} \\ a_{21} & a_{22}+N & a_{23} \\ a_{31} & a_{32} & a_{33}+\lambda_3 \end{pmatrix} = \begin{pmatrix} a'_{11} & a_{12} & a_{13} \\ a_{21} & N' & a_{23} \\ a_{31} & a_{32} & a'_{33} \end{pmatrix}$$

$$\begin{aligned} det(G^T A G + \lambda) &= N' a'_{11} a'_{33} + a_{21} a_{32} a_{13} + a_{31} a_{12} a_{23} - \\ &\quad - N' a_{13} a_{31} - a_{23} a_{32} a'_{11} - a'_{33} a_{12} a_{21} \\ &= N'(a'_{11} a'_{33} - a_{13} a_{31}) + K \end{aligned}$$

$$(G^T A G + \lambda)^{-1} = \frac{1}{det(G^T A G + \lambda)} \begin{pmatrix} N' a'_{33} - a_{23} a_{32} & a_{13} a_{32} - a_{12} a'_{33} & a_{12} a_{23} - a_{13} N' \\ a_{23} a_{31} - a'_{33} a_{21} & a'_{11} a'_{33} - a_{13} a_{31} & a_{13} a_{21} - a'_{11} a_{23} \\ a_{21} a_{32} - N' a_{31} & a_{12} a_{31} - a'_{11} a_{32} & N' a'_{11} - a_{12} a_{21} \end{pmatrix}$$

$$(G^T A G + \lambda)^{-1} = \frac{1}{det(G^T A G + \lambda)} \begin{pmatrix} N' a'_{33} - k_{11} & k_{12} & k_{13} - a_{13} N' \\ k_{21} & k_{22} & k_{23} \\ k_{31} - N' a_{31} & k_{32} & N' a'_{11} - k_{33} \end{pmatrix}$$

$$(G^T A G + \lambda)^{-1} = \begin{pmatrix} \frac{N' a'_{33} - k_{11}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} & \frac{k_{12}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} & \frac{k_{13} - a_{13} N'}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} \\ \frac{k_{21}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} & \frac{k_{22}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} & \frac{k_{23}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} \\ \frac{k_{31} - N' a_{31}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} & \frac{k_{32}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} & \frac{N' a'_{11} - k_{33}}{N'(a'_{11} a'_{33} - a_{13} a_{31}) + K} \end{pmatrix}$$

$$\lim_{N \to \infty} (G^T A G + \lambda)^{-1} = \lim_{N' \to \infty} (G^T A G + \lambda)^{-1} = \begin{pmatrix} \frac{a'_{33}}{a'_{11} a'_{33} - a_{13} a_{31}} & 0 & \frac{-a_{13}}{a'_{11} a'_{33} - a_{13} a_{31}} \\ 0 & 0 & 0 \\ \frac{-a_{31}}{a'_{11} a'_{33} - a_{13} a_{31}} & 0 & \frac{a'_{11}}{a'_{11} a'_{33} - a_{13} a_{31}} \end{pmatrix}$$

[1] Por ejemplo, si $N_2 = 20$, y se quiere forzar tres cambios de la acción de control en t, $t+3$ y $t+6$, entonces $N_u = 7$ para que recoja todo el horizonte en el que se producen los cambios en la acción de control.

Con lo que se comprueba que se consigue el objetivo de forzar $\Delta u(t+1) = 0$ gracias a la fila de ceros que aparece en la matriz $(G^T AG + \lambda)^{-1}$:

$$\hat{u} = \begin{pmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \Delta u(t+2) \end{pmatrix} = \begin{pmatrix} \frac{a'_{33}}{a'_{11}a'_{33}-a_{13}a_{31}} & 0 & \frac{-a_{13}}{a'_{11}a'_{33}-a_{13}a_{31}} \\ 0 & 0 & 0 \\ \frac{-a_{31}}{a'_{11}a'_{33}-a_{13}a_{31}} & 0 & \frac{a'_{11}}{a'_{11}a'_{33}-a_{13}a_{31}} \end{pmatrix} G^T A(W - \Gamma\Delta U^f - FY^f)$$

Con esta alternativa la redistribución de la acción de control se consigue mediante una asignación adecuada de los parámentros: N_u y λ_j en un GPC clásico.

El horizonte de control se debe incrementar para englobar todas las variaciones de la acción de control y se deben asignar $\lambda_j = \infty$ para forzar incrementos de la acción de control nulos en dicho horizonte.

Desde un punto de vista práctico esta forma de operar tiene inconvenientes, pues requiere trabajar con factores $\lambda_j = \infty$, y esto, computacionalmente requiere aproximaciones y puede presentar problemas en la inversión de la matriz.

2. **Reconfiguración de la matriz G.**

La otra alternativa para incluir la redistribución de la acción de control en la formulación analítica del GPC pasa por definir una nueva matriz G^* basada en la matriz G en la que se excluyen de antemano las columnas 'j' correspondientes a los $\Delta u(t+j-1)$ que se desean anular, quedando N_u columnas[2] y se reduce la dimensión de la matriz λ que sólo incluye los factores de ponderación de los incrementos de las acciones de control que no son nulas a priori.

Para el ejemplo anterior: $N_2 = 3$, $N_1 = 1$ y se trata permitir dos cambios en la acción de contro pero forzando $\Delta u(t+1) = 0$, por tanto, $N_u = 2$. Se opera eliminando la segunda fila de G para obtener G^*, obteniendo:

$$G^{*T} AG^* = \begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix} \; ; \; \lambda^* = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_3 \end{pmatrix}$$

$$G^{*T} AG^* + \lambda^* = \begin{pmatrix} a_{11} + \lambda_1 & a_{13} \\ a_{31} & a_{33} + \lambda_3 \end{pmatrix} = \begin{pmatrix} a'_{11} & a_{13} \\ a_{31} & a'_{33} \end{pmatrix}$$

$$det(G^{*T} AG^* + \lambda^*) = a'_{11}a'_{33} - a_{13}a_{31}$$

[2]Entendiendo N_u como el número de variaciones no nulas de la acción de control que se permiten a lo largo del horizonte de predicción, estas variaciones no tienen por qué ser consecutivas.

$$(G^{*T}AG^{*}+\lambda^{*})^{-1}\begin{pmatrix} \frac{a'_{33}}{a'_{11}a'_{33}-a_{13}a_{31}} & \frac{-a_{13}}{a'_{11}a'_{33}-a_{13}a_{31}} \\ \frac{-a_{31}}{a'_{11}a'_{33}-a_{13}a_{31}} & \frac{a'_{11}}{a'_{11}a'_{33}-a_{13}a_{31}} \end{pmatrix}$$

Esta alternativa disminuye la dimensión de $G^TAG+\lambda$ frente a la alternativa anterior y por tanto se simplifica la inversión de esta matriz.

Las diferencias en las dimensiones de las matrices que intervienen en los cálculos en ambos métodos se pueden ver fácilmente:

$$\begin{aligned} \hat{u}_{N_{u1}\times 1} &= (G^T_{N_{u1}\times N}A_{N\times N}G_{N\times N_{u1}}+\lambda_{N_{u1}\times N_{u1}})^{-1}G^T_{N_{u1}\times N}A_{N\times N}(W-\Gamma\Delta U^f-FY^f)_{N\times 1} \\ \hat{u}^{*}_{N_u\times 1} &= (G^{*T}_{N_u\times N}A_{N\times N}G^{*}_{N\times N_u}+\lambda^{*}_{N_u\times N_u})^{-1}G^{*T}_{N_u\times N}A_{N\times N}(W-\Gamma\Delta U^f-FY^f)_{N\times 1} \end{aligned}$$

Donde:

- $N=N_2-N_1$
- $N_{u1}\geq N_u$. Por ejemplo, si $N_u=2$ (número de cambios de la acción de control a lo largo del horizonte de predicción) y se quiere: $\Delta u(t)\neq 0$, $\Delta u(t+k)\neq 0$.

$$\hat{u}_{N_{u1}\times 1}=\begin{pmatrix} \Delta u(t) \\ 0 \\ \cdots \\ 0 \\ \Delta u(t+k) \end{pmatrix} \quad ; \quad \hat{u}^{*}_{N_u\times 1}=\begin{pmatrix} \Delta u(t) \\ \Delta u(t+k) \end{pmatrix}$$

$$N_{u1}=k+1>2$$

La segunda alternativa, aunque menos intuitiva, es menos compleja computacionalmente y no requiere aproximaciones. Hay que destacar que estos desarrollos son exclusivos para sistemas lineales y con un índice cuadrático, es mucho más intuitiva y fácil de manejar la redistribución de la acción de control si se utiliza una técnica de optimización heurística, siendo directamente aplicable para procesos no lineales e índices de coste alternativos al cuadrático

4.2.3 Índices no cuadráticos

La utilización de un índice cuadrático está justificado en el desarrollo del GPC convencional, puesto que, además de evaluar el error que se produce (discrepancia entre la salida y la referencia a lo largo del horizonte de predicción), evitando que se compensen errores positivos y negativos, resulta ser derivable (indispensable para la minimización analítica que se realiza en el GPC lineal).

Con la utilización de alguna de la técnicas de optimización heurística descritas para la minimización, la derivabilidad del índice no es un requisito, por tanto se pueden buscar otros índices en función de la evaluación del error a lo largo del horizonte de predicción, e incluso plantear índices que utilicen parámetros diferentes de este error para establecer el comportamiento en bucle cerrado [93]. De hecho, como ya se comentaba en el capítulo 1, es usual encontrar en la bibliografía distintos tipos de índices para evaluar las prestaciones de un sistema de control y cada uno de ellos esta indicado para distintas situaciones y procesos. Por ejemplo:

- Según *Stephanopoulos* [95]:

  "...Respecto a cual de los tres criterios (ISE, IAE, o ITAE) se debe utilizar, depende de las características del sistema a controlar y de los requerimientos adicionales que impongamos a la respuesta controlada del proceso. ..."

  "...Si se quiere sobre todo suprimir los errores grandes, ISE es mejor que IAE ya que los errores son elevados al cuadrado y contribuyen más en la integral.

  Para la supresión de errores pequeños, IAE es mejor que ISE ya que cuando se eleva al cuadrado un número pequeño (menor que 1) se convierte en un número más pequeño aún.

  Para suprimir errores que persisten durante largo tiempo, el criterio ITAE ajustará mejor el controlador ya que la presencia de 't' grandes amplifica el efecto de los errores, incluso pequeños, en el valor de la integral. ..."

- Según *Marlin* [60]:

  "...IAE es una medida apropiada de las prestaciones del control cuando el efecto sobre las prestaciones es lineal con la magnitud de la desviación. ISE es apropiado cuando grandes desviaciones causan una mayor degradación de las prestaciones que pequeñas desviaciones. ITAE penaliza las desviaciones que persisten durante largo tiempo. ..."

Estos mismos razonamientos se podrían trasladar directamente al índice de coste de un MBPC, en cada periodo de muestreo se minimiza un índice que evalúa las prestaciones de las distintas trayectorias de la variable que se controla.

La idea de índices diferentes al cuadrático también aparece en otros campos de control, Aström y Wittenmark [2] plantean la posibilidad de utilizar un índice que evalúa el valor absoluto del error para la implementación de un controlador MRAS como alternativa al índice cuadrático puesto que este introduce distorsiones.

En definitiva, parece que el índice cuadrático ha sido de los más utilizados más para facilitar la tarea al optimizador que por la calidad de las prestaciones que pueda conseguir en el control.

En este apartado se muestra que la utilización de distintas funciones del error deriva en un comportamiento diferente en bucle cerrado.

Una de las características del índice cuadrático es que distorsiona los errores, se atenúan más los errores menores de la unidad frente a los que son superiores a la unidad. Este efecto influye en la minimización, por ejemplo, si se dan los dos tipos de errores a lo largo del horizonte de predicción, los más grandes pueden ser sobrevalorados y quizá no sea lo adecuado dependiendo de la aplicación. Incluso en el caso en que todos los errores en el horizonte de predicción sean menores que la unidad aparece una distorsión, cuanto menor sea el valor su influencia en el índice está más atenuada.

Para evitar esta distorsión, según el valor del error, la alternativa más clara es la utilización en el índice de la función valor absoluto. El índice quedaría la forma siguiente:

$$J(u) = E\left\{\sum_{j=N_1}^{N_2} |e(t+j)| + \sum_{j=1}^{N_u} \lambda_j \, |\Delta u(t+j-1)|\right\} \tag{4.20}$$

Otro tipo de índice que se puede plantear es el de la raíz cuadrada del módulo del error. El efecto sería el inverso al que se da con el cuadrado. En este caso, se atenúan más los errores mayores de la unidad frente a los errores menores de la unidad. El índice sería:

$$J(u) = E\left\{\sum_{j=N_1}^{N_2} \sqrt{|e(t+j)|} + \sum_{j=1}^{N_u} \lambda_j \sqrt{|\Delta u(t+j-1)|}\right\} \tag{4.21}$$

Aparece pues una gran variedad de posibilidades en la forma de afectar al error y al término de las acciones de control, a continuación únicamente se describen algunas de las características de las alternativas descritas. Para ver las diferencias entre los índices se plantea el caso hipotético caso en el que $N_2 = 2$, en los que los factores de ponderación de las acciones de control se ajustan $\lambda_j = 0$ para simplificar el análisis. Los índices quedarían

de la siguiente forma:

$$C1 \rightarrow J_{C1} = e_1^2 + e_2^2 \tag{4.22}$$
$$C2 \rightarrow J_{C2} = |e_1| + |e_2| \tag{4.23}$$
$$C3 \rightarrow J_{C3} = \sqrt{|e_1|} + \sqrt{|e_2|} \tag{4.24}$$
$$\tag{4.25}$$

Se pueden ver las formas que presentan los distintos índices en el espacio $(e_1, e_2) \in [-1, 1] \times [-1, 1]$ en la figura 4.10. Como cabría esperar, cada uno de los índices resulta en una

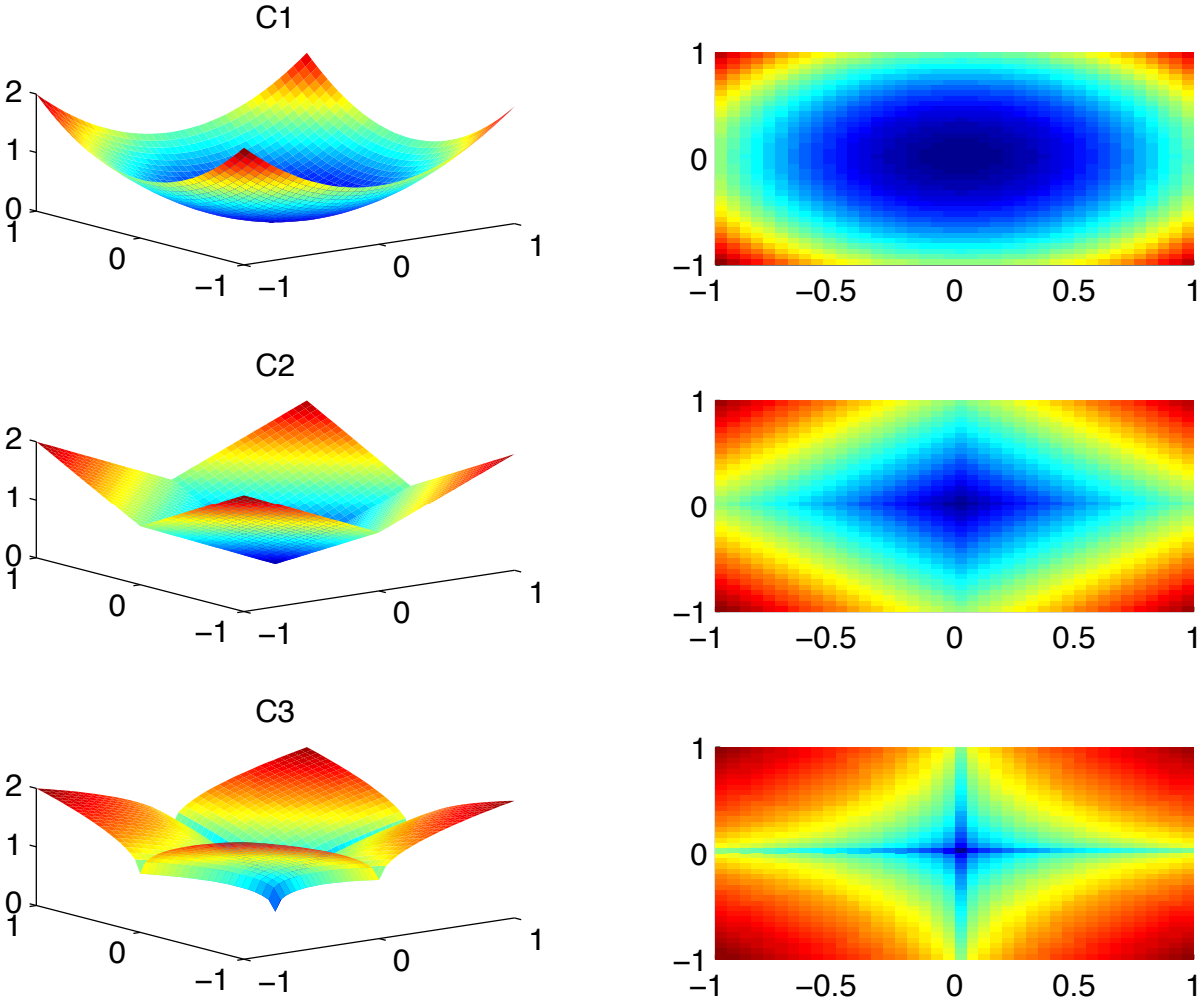


Figura 4.10: Índices $J(e_1, e_2)$ en el espacio $(e_1, e_2) \in [-1, 1] \times [-1, 1]$. *C1* índice cuadrático, *C2* índice con el módulo del error y *C3* raíz cuadrada del módulo del error.

función diferente pero esto no implica necesariamente que el resultados de la minimización

sea diferente. Depende de la zona en que se encuentren los errores que intervienen en el índice, para comprobar este hecho, se compara tres puntos diferentes del plano:

$$\begin{aligned} \text{Punto A} &\rightarrow (-0.48, 0.12) \\ \text{Punto B} &\rightarrow (-0.7, 0.0267) \\ \text{Punto C} &\rightarrow (-0.3, -0.3317) \end{aligned}$$

Para cada punto se obtienen los siguientes valores de las funciones:

$$\begin{aligned} \text{Punto A} &\rightarrow C1(A) = 0.2448 \; ; \; C2(A) = 0.6 \; ; \; C3(A) = 1.0392 \\ \text{Punto B} &\rightarrow C1(B) = 0.4907 \; ; \; C2(B) = 0.7267 \; ; \; C3(B) = 1 \\ \text{Punto C} &\rightarrow C1(C) = 0.2 \; ; \; C2(C) = 0.6317 \; ; \; C3(C) = 1.236 \end{aligned}$$

Por tanto la minimización para cada función y estos tres puntos quedaría:

$$\begin{aligned} \text{Función C1} &\rightarrow C1(C) < C1(A) < C1(B) \\ \text{Función C2} &\rightarrow C2(A) < C2(C) < C2(B) \\ \text{Función C3} &\rightarrow C3(B) < C3(A) < C3(C) \end{aligned}$$

Esto mismo se puede ver gráficamente en la figura 4.11 donde se muestran las curvas de nivel de valor 0.2, 0.6, 1, 1.4 y 1.8 para las tres funciones, además de la posición de los tres puntos.

Las pruebas de la figura 4.11 muestran la diferencia que existe entre los diferentes tipos de índices descritos pero no muestran como afectan estas diferencias al control de un proceso. Como ejemplo se muestra el control de un doble integrador muestreado a 10 Hz mediante un control predictivo con algoritmo genético (codificación real). El horizonte de control utilizado es $N_u = 2$. La simulación *C1* corresponde al índice cuadrático convencional, *C2* corresponde al índice con el módulo del error y *C3* al índice con la raíz cuadrada del módulo del error.

Los resultados de la figura 4.12 muestran un comportamiento similar de todos los índices en los primeros instantes (coincide con la zona en que la acción de control está saturada) y cuando el proceso se encuentra en régimen permanente (en esta zona, cualquier control que lleve el sistema al punto de funcionamiento ofrece la misma acción de control). Las diferencias aparecen en la zona intermedia, cuando el sistema se está acercando a la referencia, afectando al régimen transitorio. En este ejemplo el índice C3 muestra un mejor comportamiento, pero no es difícil demostrar que este resultado sea generalizable.

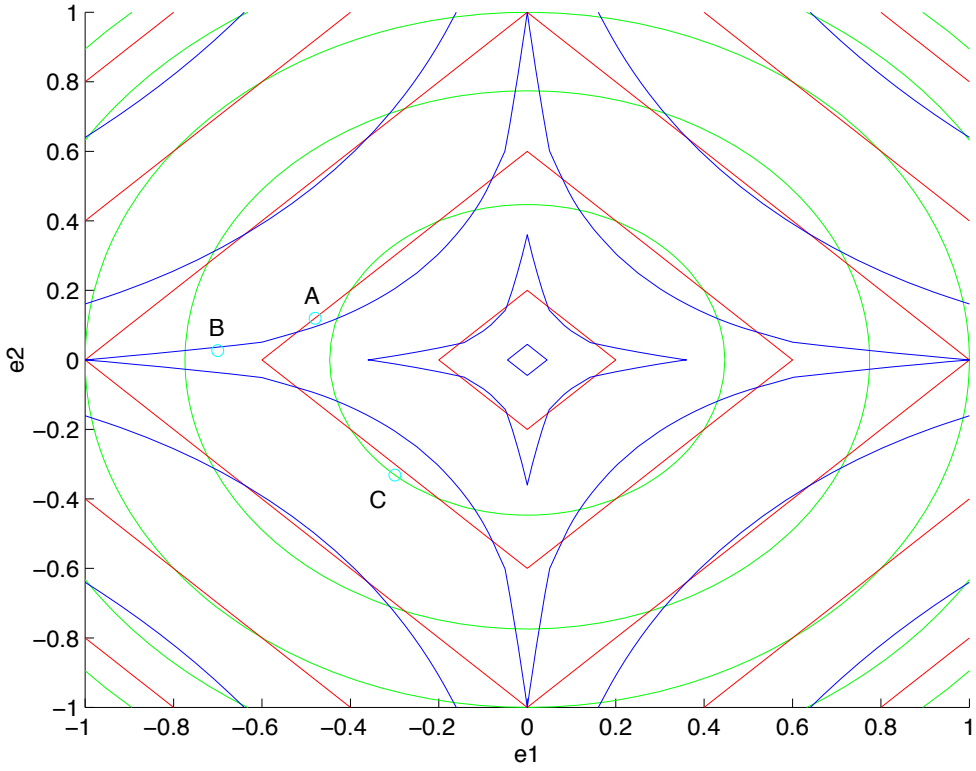


Figura 4.11: Puntos A, B y C, y curvas de nivel 0.2, 0.6, 1, 1.4 y 1.8 para las tres funciones. *C1* en verde , *C2* en rojo y *C3* en azul.

En cualquier caso se trata de poner de manifiesto que la utilización de una técnica de optimización heurística como los Algoritmos Genéticos o *Simulated Annealing* permite plantearse el cambio de índices sin dificultades en la formulación y adecuándolos a problemas concretos.

En los apartados y capítulos siguientes se tiende a utilizar el índice con el valor absoluto del error evitando las distorsiones que presentan los demás índices.

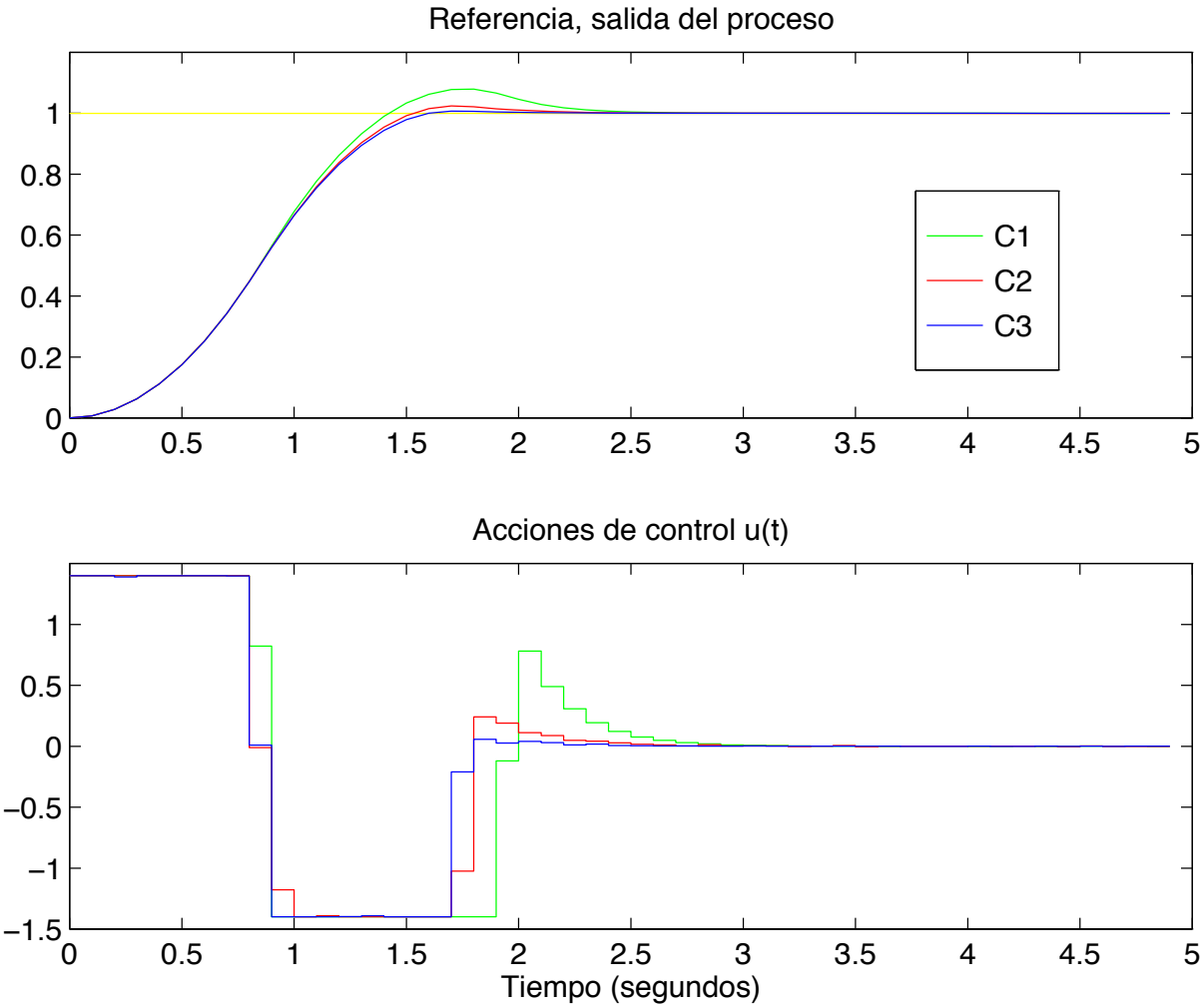


Figura 4.12: Control del doble integrador con diferentes índices: *C1* índice cuadrático, *C2* índice con el módulo del error y *C3* raíz cuadrada del módulo del error.

4.3 Conclusiones

En este capítulo se ha descrito la utilización de técnicas de optimización heurística al MBPC. La propuesta se ha basado en un GPC por ser un tipo de control predictivo que recoge muchos de los avances que se han dado en este campo. En cualquier caso las estructura que se plantea abre las posibilidades a otro tipo de modelos y funciones de coste, lo que permite tener un punto de partida sobre el que trabajar para conseguir mejoras en las prestaciones.

La parte final del capítulo ha mostrado dos aspectos con los que se pueden variar las prestaciones del controlador, independientemente del modelo del proceso que se utilice:

1. **Redistribución de la acción de control.** Se ha descrito un grado de libertad adicional, que se obtiene mediante unas redistribuciones alternativas de las acciones de control a lo largo del horizonte de predicción. Esta nueva posibilidad permite, por ejemplo, conseguir respuestas intermedias entre las conseguidas con $N_u = 1$ y $N_u = 2$ en cuanto a la respuesta transitoria del sistema y agresividad de las acciones de control.

   Una característica importante es que el MBPC con optimización heurística permite estructurar la ley de control con más libertad sin introducir mayores complicaciones en la formulación. Este aspecto se muestra comparando la redistribución de la acción de control en un MBPC con optimización heurística y en un GPC lineal.

2. **Índices de coste no cuadráticos.** Se han mostrado las diferencias entre el índice cuadrático clásico, y el que se basa en el valor absoluto del error y la raíz cuadrada del valor absoluto del error. La conclusión más destacada es que la utilización de una técnica de optimización heurística permite incorporar índices de coste alternativos que se ajusten más a las necesidades de cada problema MBPC y que permitirían mejorar sus prestaciones.

Capítulo 5

MBPC con optimización heurística en procesos no lineales

En el MBPC las mejoras en las prestaciones pueden aparcer principalmente por dos vías:

- **Modificación de los elementos del controlador**. En este sentido el capítulo anterior mostraba dos aspectos en los que el MBPC con optimización heurística podía contribuir para mejorar la calidad del control: índices de coste y estructuraciones de la ley de control alternativos. Otro aspecto que puede mejorar las prestaciones en ciertas aplicaciones es la posibilidad de incorporar de forma simple las restricciones de funcionamiento sobre las variables del proceso.
- **Utilización de toda la información disponible del proceso**. Esto debe entenderse como la posibilidad de utilizar modelos muy elaborados del proceso que incorporen las no linealidades del proceso, sensores y actuadores, además de modelos generales de perturbaciones deterministas y estocásticas, medibles o no. Las mejoras en las prestaciones pasan por utilizar esta información más exacta del comportamiento dinámico de todas las partes del sistema e incorporarlas al diseño[1].

En este capítulo se presentan las nuevas alternativas que ofrece el control predictivo con optimización heurística relacionadas con la utilización de modelos no lineales y la inclusión de restricciones en las variables del proceso. La optimización heurística permite además combinar toda esta información del proceso con índices de coste y estructuraciones de la ley de control alternativos lo que permite conseguir buenas prestaciones en problemas de control difíciles.

[1]Esta idea esta presente en muchas áreas del control siempre con el objetivo de obtener mejores prestaciones, se pueden ver algunos ejemplos de utilización de modelos no lineales en el control en [99] y [54].

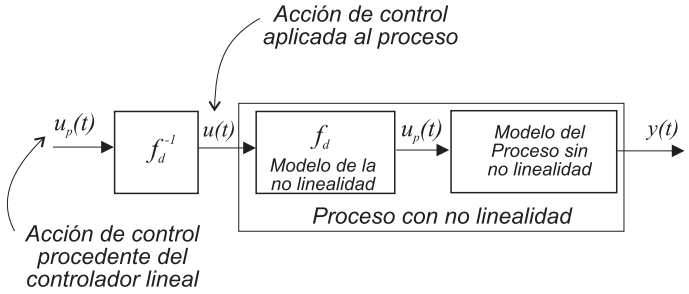


Figura 5.1: Compensación de las no linealidades invertibles con controladores lineales.

5.1 Introducción

En muchos problemas prácticos, para el diseño del control de un proceso no lineal, se puede aproximar un modelo lineal o localmente linealizado sin perder excesiva calidad en el control resultante. Sin embargo, algunos actuadores pueden presentar unas no linealidades fuertes que pueden tener un efecto dominante en el comportamiento en bucle cerrado. Algunos ejemplos de estas no linealidades fuertes aparecen en [23, 99, 101] y entre las más características están:

- Saturación en la amplitud de la acción de control.
- Saturación en el incremento de la acción de control.
- Zonas muertas.
- Backlash.
- Histéresis.

Este tipo de no linealidades en los actuadores son normalmente conocidas, se dispone de una descripción matemática, **función de descripción** ($u_p = f_d(u)$), que permite obtener la acción de control efectiva sobre el proceso 'u_p' a partir de la acción de control que suministraría el controlador 'u'.

Los controladores lineales son calculados sin tener en cuenta las no linealidades, por la dificultad que entraña su inclusión en el diseño, a costa de un posible pobre funcionamiento. En estos casos, se supone que la acción de control que calcula el controlador coincide con la acción de control efectiva, $u = u_p$. A efectos prácticos es como si el controlador calculase u_p. En algunos casos se podría compensar la no linealidad siempre que su función de descripción sea invertible, es decir, que se pueda calcular u a partir de u_p, $u = f_d^{-1}(u_p)$ (figura 5.1).

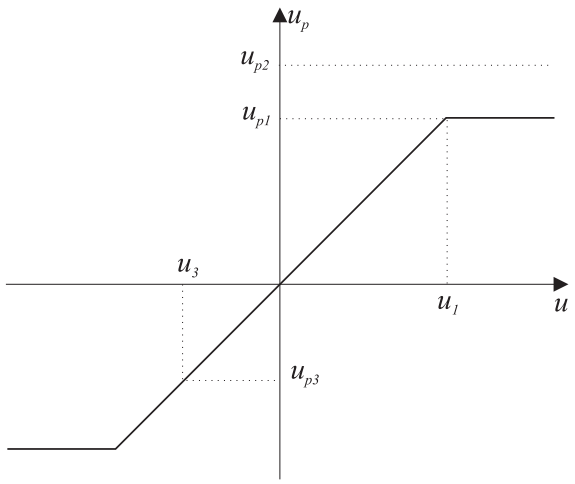


Figura 5.2: Función de descripción de una saturación. Para punto u_{p1} existen infinitos $u \in [u_1, \infty[$. Para el punto u_{p2} no existe ningún valor de u. Para el punto u_{p3} hay un único valor de $u = u_3$.

Esta compensación no siempre es posible, en la figura 5.2 se muestra un ejemplo con la función de descripción de una saturación que presenta zonas no invertibles.

Cuando una función de descripción no es invertible puede ser debido a:

- La aparición de varios, e incluso, infinitos valores de u para cada valor de u_p. Este problema se puede salvar, en algunas ocasiones, fijando un criterio para seleccionar un valor de u entre las varias posibilidades que aparezcan, por ejemplo, el criterio de menor energía:

  $$u = \min\{|f_d^{-1}(u_p)|\}$$

  En el ejemplo de la figura 5.2, para el punto u_{p1} se tomaría:

  $$u_1 = \min\{|f_d^{-1}(u_{p1})|\}$$

- Que no exista ningun valor de u para un u_p determinado. En estos casos cualquier selección de un valor de u produce un control que no es óptimo puesto que no se siguen las indicaciones del controlador en cuanto a u_p.

Si no se tiene en cuenta la no linealidad su efecto aparece como parte de la dinámica no modelada, por tanto, dependiendo de la robustez del controlador se obtendrá un funcionamiento más o menos correcto en bucle cerrado. En muchas implementaciones prácticas del controlador se asume que, sea cual sea la acción de control que calcula el algoritmo de control, el actuador puede llevarla a cabo sin saturaciones, zonas muertas o

histéresis. Esta asunción puede llevar a un comportamiento no deseado o incluso inestable del proceso.

Por tanto, de ser posible, sería una buena idea incorporar el conocimiento que *a priori* se tiene de la no linealidad del actuador en el diseño del controlador. Un ejemplo clásico en la consideración de estos efectos es el *Antiwindup* en los reguladores con acción integral, que permite paliar el efecto negativo que introduce una saturación en la acción integral. En cualquier caso el *Antiwindup* en un regulador de tipo PID, sólo reduce los defectos que puede producir la saturación pero el controlador no ofrece las mejores acciones de control para conseguir las especificaciones, se trata de una solución subóptima puesto que no tiene en cuenta la saturación en los cálculos de u_p.

Para el caso del MBPC, algunas de las no linealidades pueden ser tenidas en cuenta mediante restricciones en las acciones de control, por tanto un MBPC que use una técnica de optimización que sea capaz de manejar restricciones (por ejemplo, mediante el método de los multiplicadores de Lagrange) puede incorporar dicho conocimiento [23]. En algunos casos el problema puede resultar poco manejable en el proceso de optimización (resultando costoso computacionalmente) o incluso resultar un problema de optimización no convexo. En otros casos, la no linealidad no es trasladable a un conjunto de restricciones apto para un optimizador clásico. En definitiva, la inclusión de este tipo de no linealidades fuertes puede provocar que el problema a minimizar sobrepase las capacidades de un optimizador clásico, con lo cual aparece la alternativa de las técnicas de optimización heurísticas.

En los siguientes apartados, primero se propondrán dos métodos alternativos para la inclusión de no linealidades en los actuadores en el MBPC cuando se utiliza un optimizador heurístico y posteriormente se realizará un análisis de los resultados que se obtienen para distintos modelos de procesos combinados con estas no linealidades.

Otro de las apartados que se presentan en este capítulo estudia la inclusión de 'restricciones en la variable de salida', se muestra que el tratamiento es similar a la inclusión de restricciones en la entrada (debidas por ejemplo a no linealidades en los actuadores).

La última parte trata el funcionamiento del MBPC con optimización heurística con otros tipos de modelos no lineales distintos de saturaciones, zonas muertas, backlash e histéresis. Para ilustrar este caso se realiza el control MBPC de un brazo robot con unión flexible con un modelo no lineal en espacio de estados.

5.2 Incorporación de no linealidades en los actuadores

A la hora de incluir la información de las no linealidades de los actuadores se dispone de dos alternativas, la primera es aplicable a todos los tipos de no linealidades de las que se disponga de un modelo matemático, la segunda sólo a aquellas no linealidades que se

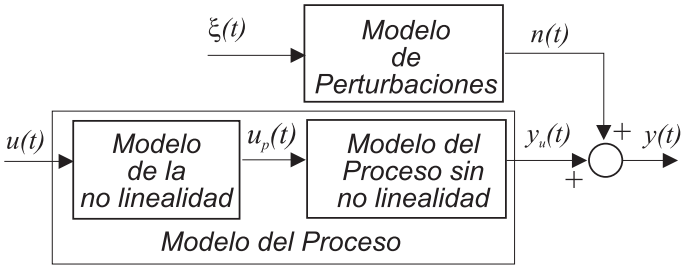


Figura 5.3: Incorporación del modelo de la no linealidad en el modelo de predicción. $u(t)$ es la acción de control que calcula el controlador y $u_p(t)$ es la acción de control efectiva sobre el proceso.

puedan convertir en restricciones en el problema de optimización de un MBPC. Estas dos alternativas son:

- Inclusión en el modelo de predicción.
- Inclusión como restricciones.

5.2.1 Inclusión en el modelo de predicción

Este método consiste en:

- Incluir la información de la no linealidad en el modelo del proceso utilizado para realizar las predicciones (figura 5.3).
- El índice de coste está en función de las acciones de control 'u' que se va a aplicar al proceso, $J(u)$.
- El espacio de búsqueda de soluciones es infinito (las variables $u(t), u(t+1), \ldots, u(t+N_u-1)$ pueden tomar cualquier valor), las restricciones están incluidas en el modelo.

El cálculo de la acción de control pasaría por la resolución de un problema de minimización sin restricciones en el que la función puede ser tan compleja (no convexa, discontinuidades, etc.) que invalida un método de optimización clásico.

En este caso se pretende que el modelo de predicción incorpore el conocimiento de la no linealidad, se obtienen por tanto predicciones más precisas. Las únicas modificaciones que se deben llevar a cabo se realizan sobre el modelo de predicción, siguiendo el esquema que se muestra en la figura 5.3.

Las características de esta alternativa son:

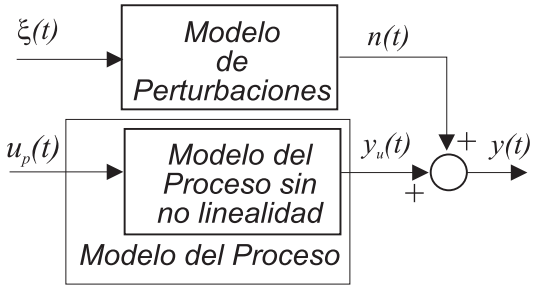


Figura 5.4: Modelo utilizado para las predicciones cuando la no linealidad se tiene en cuenta como restricción del problema de optimización 5.2.2.

- Se puede utilizar con cualquier tipo de no linealidad de la que se disponga un modelo y no está restringida a no linealidades de los actuadores.
- No conlleva ninguna modificación sobre el índice de coste para tener en cuenta las no linealidades.
- Puede producir índices de coste multimodales con numerosos o infinitos mínimos con el mismo valor.

5.2.2 Inclusión como restricciones

Esta alternativa consiste en incluir la información en forma de restricciones o ligaduras en el problema de optimización.

- El modelo utilizado para realizar las predicciones no tiene en cuenta la no linealidad (figura 5.4).
- La información de la no linealidad será incluida en el problema de optimización como un conjunto de restricciones sobre las acciones de control efectivas 'u_p' (figura 5.5):

  $$X = \{u_p \ : \ u_p \in R^{N_u} \ ; \ g(u_p) \leq 0\}$$

  El problema de minimización se convierte en:

  $$\min\{J(u_p) \ ; \ u_p \in X\}$$

- Finalmente se debe obtener la acción de control a aplicar 'u' a partir de la solución 'u_p' del problema de optimización anterior mediante la inversión de la función de descripción de la no linealidad.

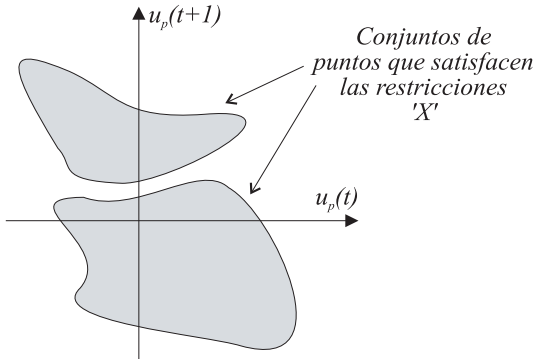


Figura 5.5: Ejemplo de dos dimensiones. El optimizador debe buscar el mínimo de $J(u_p)$ entre los valores del conjunto de puntos que satisfacen las restricciones 'X'.

El optimizador debe buscar el mínimo de la función sin violar ninguna de las restricciones. Hay que resaltar que no siempre es posible trasladar una no linealidad fuerte a un conjunto de restricciones, por tanto el uso de esta alternativa está limitado a aquellas no linealidades que puedan ser expresadas como restricciones.

Las características generales de esta alternativas son:

- No es necesario modificar el modelo del proceso (ni el de perturbaciones), pues es la función de coste la que tiene en cuenta las no linealidades.
- No es directamente aplicable a todo tipo de no linealidades, es necesario que las no linealidades sean traducibles a un conjunto de restricciones para conformar el espacio de búsqueda 'X' y que sea posible obtener una acción de control u a partir de la solución del problema de optimización u_p, es decir, que la función de descripción de la no linealidad sea invertible o en el caso de que existan muchas soluciones establecer un criterio adicional para la selección de la acción de control.

Esta alternativa se puede implementar de dos forma:

1. **Penalización del índice de coste.**

   Penalizando el índice de coste según el grado de incumplimiento de las restricciones impuestas por las no linealidades, esto afecta a la función a minimizar (se corrige con una función de penalización).

   Este método convierte un problema con restricciones, $J(u_p)$, en uno sin restricciones, $J'(u_p)$. El método consiste en asociar un coste a todas las violaciones de las restricciones, este coste es incluido en la evaluación del índice. El índice de coste que realmente se minimiza es:

$$J'(u_p) = J(u_p) + r\phi(u_p) \tag{5.1}$$

donde:

r : coeficiente de penalización.
$\phi(u_p)$: función de penalización [2].

Además de las características generales mencionadas anteriormente, esta alternativa puede provocar:

- Funciones a minimizar con fuertes discontinuidades produciendo mínimos aislados, es decir, el mínimo está rodeado de valores de la función muy altos. Este efecto puede provocar la convergencia a mínimos locales si no selecciona y ajusta adecuadamente la técnica de optimización.

2. **Modificación del espacio de búsqueda**.

   Modificando el espacio de búsqueda de manera que no estén incluidas las zonas que corresponden a acciones de control no permitidas por las no linealidades, esto implica una codificación del espacio de búsqueda más sofisticada que no siempre es factible.

   A primera vista parece que la manera más sencilla de incluir las restricciones sea la de no considerar aquellas soluciones que violen las restricciones:

   - En el transcurso de la optimización mediante *Simulated Annealing* cuando un punto de los que aparece por agitación térmica viola las restricciones se ignora y se repite la operación hasta que el punto generado no viole las restricciones.
   - En el caso de un Algoritmo Genético se trataría de evitar que aparezcan, tanto por cruce como por mutación, puntos que violen las restricciones. Esta manera de proceder puede ser adecuada en algunos casos concretos pero, en general, no es aconsejable pues se pierde información genética[3], que en sucesivas generaciones nos podría llevar a obtener una solución óptima.

   En definitiva se realiza una búsqueda en todos el espacio pero se descartan aquellos puntos que violan las restricciones. Esta forma de operar es poco eficiente en tiempo (se deben crear muchas posibles soluciones que luego se descartan y no intervienen en la minimización) y en calidad (pérdida de información genética).

   El problema se puede resolver modificando el espacio de búsqueda para que no incluya puntos que violen las restricciones, se minimiza $J(u_p)$ y el espacio de búsqueda es X. En el caso de los Algoritmos Genéticos esta alternativa se puede resolver en la fase de codificación del espacio de búsqueda. Se trata de evitar que los puntos

[2] Esta función puede tomar, por ejemplo, valores booleanos que indiquen si se viola o no la restricción.
[3] Se pueden perder cromosomas que aún correspondiendo a individuos que violan las restricciones contienen información que corresponde al mínimo global.

que violan las restricciones aparezcan en el espacio de búsqueda lo que se puede conseguir si a estos puntos no se les asigna ningún cromosoma, de esta forma no entran en el mecanismo de evolución y nunca aparecen como soluciones.

El ejemplo más claro de esta alternativa es la saturación. Al codificar los parámetros se elige un rango de valores para cada uno de ellos (corresponde al espacio de búsqueda) haciéndolo coincidir con el rango de valores en que no saturan.

Para otros tipos de restricciones la codificación se puede complicar sensiblemente, se debería buscar una codificación de un espacio discontinuo. Un ejemplo de codificación de zonas discontinuas se presenta en el ejemplo 5.1 siguiente.

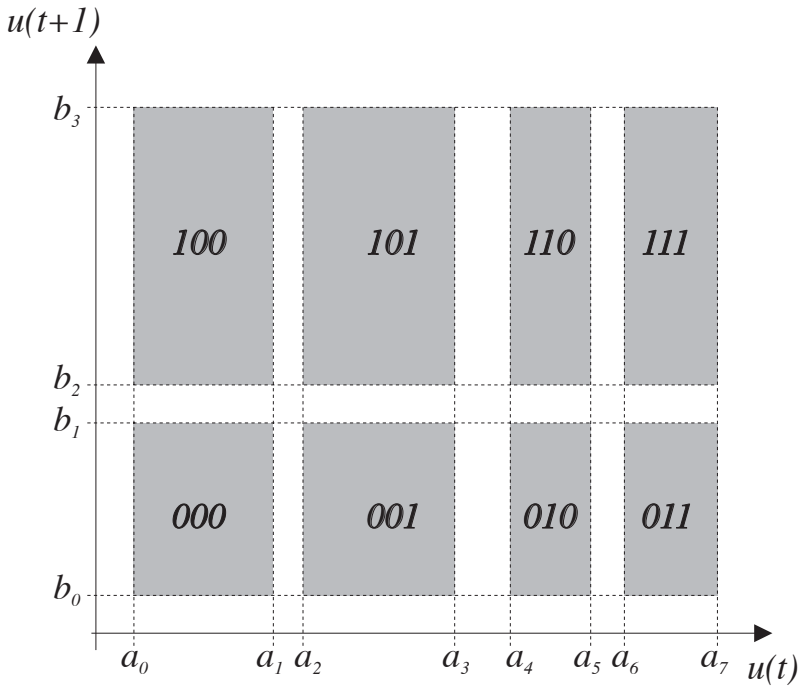


Figura 5.6: Codificación continua de parámetros discontinuos

Ejemplo 5.1 *Ejemplo de codificación de zonas discontinuas. Se supone que el espacio de búsqueda que no viola las restricciones es el que corresponde a la zona sombreada de la figura 5.6. Este espacio de búsqueda se obtiene de restricciones de*

tipo desigualdad:

Zona	$u(t)$	$u(t+1)$
0	$a_0 < u(t) < a_1$	$b_0 < u(t+1) < b_1$
1	$a_2 < u(t) < a_3$	$b_0 < u(t+1) < b_1$
2	$a_4 < u(t) < a_5$	$b_0 < u(t+1) < b_1$
3	$a_6 < u(t) < a_7$	$b_0 < u(t+1) < b_1$
4	$a_0 < u(t) < a_1$	$b_2 < u(t+1) < b_3$
5	$a_2 < u(t) < a_3$	$b_2 < u(t+1) < b_3$
6	$a_4 < u(t) < a_5$	$b_2 < u(t+1) < b_3$
7	$a_6 < u(t) < a_7$	$b_2 < u(t+1) < b_3$

Para codificar sólo los individuos que pertenecen a estas regiones, el cromosoma podría tener la estructura que se muestra en la figura 5.7, consta de tres campos: el primero marca la zona a la que pertenece el punto, y los otros dos campos corresponden a los valores de $u(t)$ y $u(t+1)$ respecto de un sistema de referencia propio de la zona (para una misma precisión no son necesarios tantos bits).

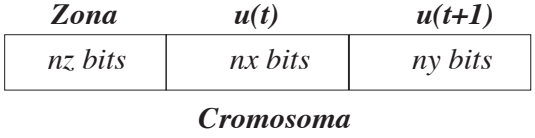


Figura 5.7: Codificación de un espacio de búsqueda discontinuo.

Además de las características generales, la modificación del espacio de búsqueda se caracteriza por:

- La disminución de zonas con mínimos rodeados de individuos con valor de la función objetivo muy altos, frente al método de penalización. En esta ocasión estos individuos no aparecen en el espacio de búsqueda.
- Suelen aparecer discontinuidades en el valor de la función objetivo.
- La fase de codificación se complica, por ejemplo en los Algoritmos Genéticos se debe incluir el marcado de la zona en la estructura del cromosoma.
- La codificación de un espacio discontinuo puede introducir distorsiones del espacio de búsqueda.

  Por ejemplo, si en la discretización del espacio de búsqueda el número de puntos por zona es el mismo, esto puede ocurrir cuando se usa el mismo número de bits por parámetro para cada zona. En estos casos, bien las zonas grandes pierden precisión o bien las zonas pequeñas tienen una precisión excesiva (mayor coste computacional). Estas alteraciones se pueden paliar tratando de que todas las

zonas tengan un tamaño similar, por ejemplo, dividiendo una zona grande en varias de tamaño más reducido. En cualquier caso esta alternativa exige un cuidado especial en la codificación.

5.3 MBPC con optimización heurística en procesos con no linealidades en actuadores

En este apartado se realiza un análisis de distintas no linealidades en los actuadores y de la incorporación de sus modelos en el control predictivo con optimización heurística. Para conseguir resultados con representatividad suficiente se utilizan varios procesos cuyas descripciones se recogen en la tabla 5.1, se trata de recoger un abanico importante de situaciones posibles en la práctica. El periodo de muestreo utilizado es de $T = 0.1seg$

En todos los ensayos se trata de seguir la referencia $r(t) = 1$ y la duración del mismo es de 50 periodos de muestreo.

Para cada no linealidad en los actuadores se realizan tres ensayos:

1. Función de coste cuadrática 5.2 y MBPC con Algoritmo Genético con codificación real.
2. Función de coste cuadrática 5.2 y MBPC con algoritmo SQP.
3. Función de coste modular 5.3 y MBPC con Algoritmo Genético con codificación real.

$$J(u) = \sum_{i=N_1}^{N_2} (y(t+i|t) - r(t+i))^2 \tag{5.2}$$

$$J(u) = \sum_{i=N_1}^{N_2} |y(t+i|t) - r(t+i)| \tag{5.3}$$

Los ensayos con el índice cuadrático se realizan para comparar los resultados entre los métodos de optimización Algoritmos Genéticos y SQP. Las simulaciones con el índice modular permiten verificar la flexibilidad del MBPC con Algoritmos Genéticos en cuanto a la utilización de distintas funciones de coste, en este caso se comparan los índices modular y cuadrático. Para el índice modular no se han realizado las simulaciones con el algoritmo SQP puesto que este algoritmo no está indicado en la optimización de este tipo de funciones.

En muchos de los ensayos el problema de optimización presenta varias soluciones con el mismo valor de la función a minimizar, en estos casos se podría plantear algún criterio para seleccionar alguna de ellas (por ejemplo, el criterios de mínima energía).

Parámetros de los ensayos		
	Modelo continuo del proceso	Parámetros del MBPC
$G_1(s)$	$\frac{e^{-2.5 \cdot s}}{s+1}$	$N_1 = 1,\ N_2 = 40,\ N_u = 2$
$G_2(s)$	$\frac{e^{-s}}{s+1}$	$N_1 = 1,\ N_2 = 20,\ N_u = 2$
$G_3(s)$	$\frac{11.25}{s^2+3s+11.25}$	$N_1 = 1,\ N_2 = 20,\ N_u = 2$
$G_4(s)$	$\frac{11.25e^{-0.5 \cdot s}}{s^2+3s+11.25}$	$N_1 = 1,\ N_2 = 20,\ N_u = 2$
$G_5(s)$	$\frac{3125}{(s+5)^5}$	$N_1 = 1,\ N_2 = 20,\ N_u = 2$
$G_6(s)$	$\frac{9(1-s)}{s^2+6s+9}$	$N_1 = 1,\ N_2 = 20,\ N_u = 2$
$G_7(s)$	$\frac{1}{s^2}$	$N_1 = 1,\ N_2 = 20,\ N_u = 2$

Tabla 5.1: Información del modelo del proceso y de los parámetros del control predictivo utilizados en los ensayos con no linealidades en los actuadores.

5.3.1 Saturación de la acción de control

Se trata de la no linealidad más común, aparece en todos los proceso. Viene definida por la siguiente expresión (gráficamente ver figura 5.8):

$$u_p(t) = \begin{cases} U_{max} & \text{Si } u(t) > U_{max} \\ u(t) & \text{Si } U_{min} \leq u(t) \leq U_{max} \\ U_{min} & \text{Si } u(t) < U_{min} \end{cases} \tag{5.4}$$

donde:

$u(t)$:	acción de control calculada por el regulador que se aplica al actuador.
$u_p(t)$:	acción de control que realmente se aplica al proceso.
U_{max}:	máxima acción de control que se puede aplicar al proceso.
U_{min}:	mínima acción de control que se puede aplicar al proceso.

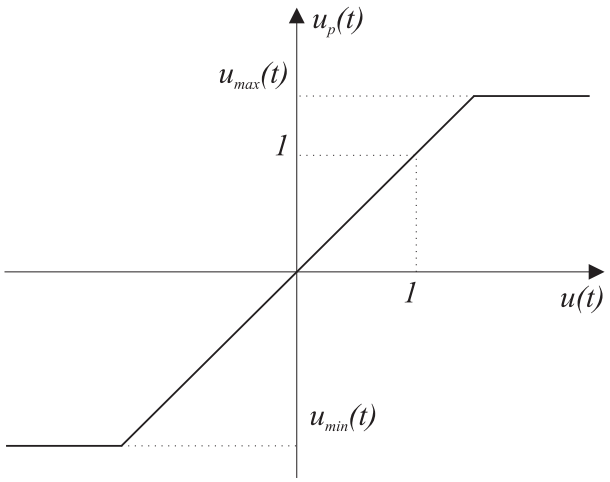


Figura 5.8: Función de descripción de una saturación en la acción de control.

Esta no linealidad es también la más fácil de incluir en un optimizador heurístico. La forma más simple es incluir la información en la fase de codificación, en definitiva consiste en limitar el espacio de búsqueda a la zona de acciones de control posibles.

Para los distintos instantes de muestreo, la función de coste cuadrática para el proceso $G_2(s)$ presenta la forma como que se muestra en la figura 5.9

La información del ajuste del GA y de los parámetros de la no linealidad se muestran en la tabla 5.2.

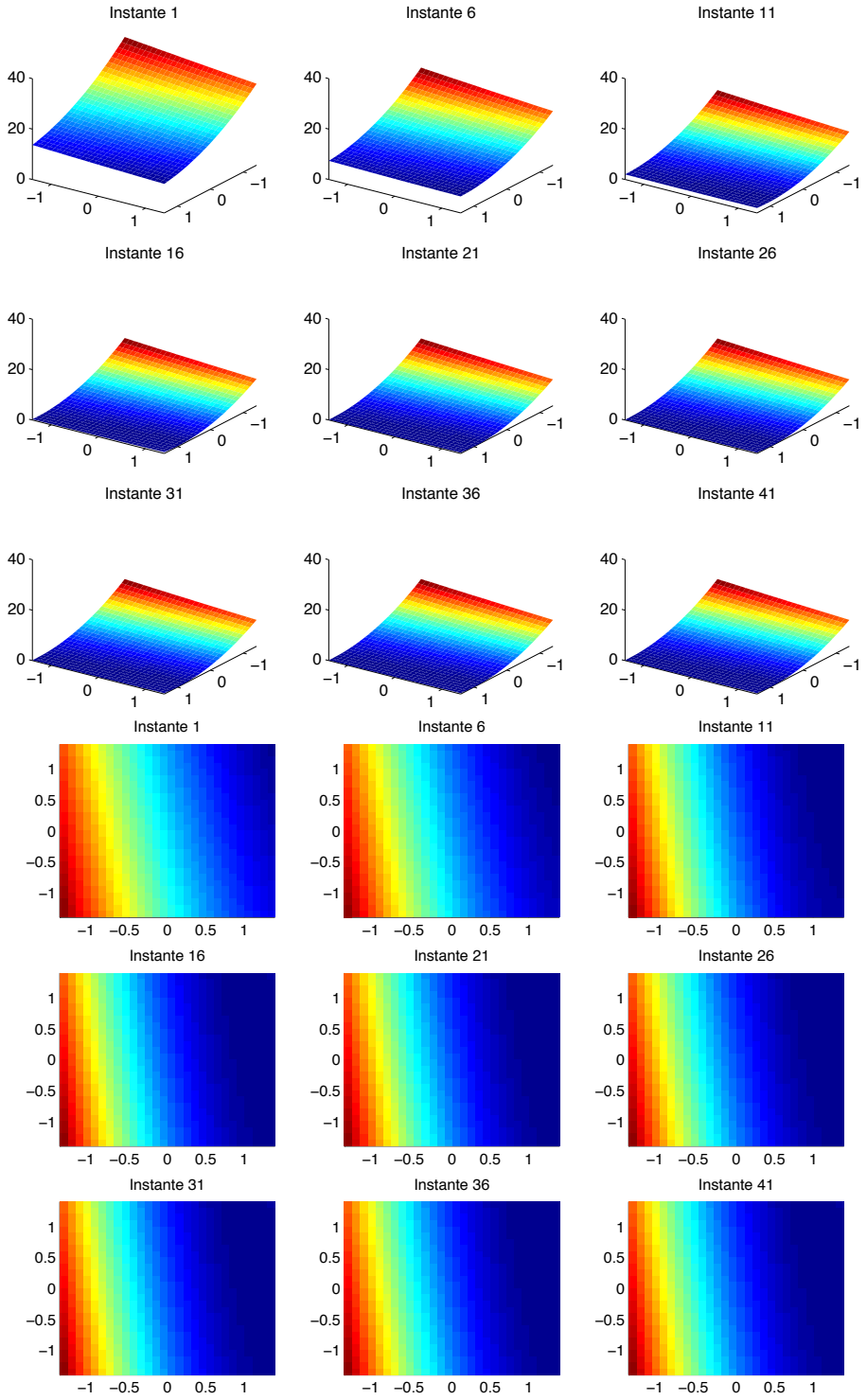


Figura 5.9: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con saturaciones en la acción de control. Proceso $G_2(s)$, índice cuadrático.

No linealidad	Saturación en $u(t)$	
	$U_{min} = -1.4$	
	$U_{max} = 1.4$	
GA real	$NumIndiv.$	100
	$MAXGEN$	500
	P_c	0.7
	P_m	0.05

Tabla 5.2: Parámetros con saturación en $u(t)$.

Los resultados obtenidos para cada proceso con el índice cuadrático (utilizando GA y SQP) y modular (utilizando GA) se muestran en el anexo D en las figuras D.1, D.2, D.3, D.4, D.5, D.6 y D.7. Para analizar estos resultados se evalúan los indicadores: IAE, ICE, ITAE e ITEC que se representan gráficamente en la figura 5.10.

Como era previsible, con el índice cuadrático tanto el GA como el SQP llegan a los mismos resultados. La función a minimizar no presentan complejidad suficiente como para invalidar el uso de un SQP, de hecho es más adecuada su utilización puesto que presenta un menor coste computacional. En cuanto a los resultados obtenidos con el índice modular se dan resultados diferentes al índice cuadrático aunque dependiendo del tipo de indicador que se evalúa se obtienen mejores o peores resultados frente al índice cuadrático. Esto es explicable por la estructura del indicador, IAE e ITAE están basados en el modulo del error y por tanto era previsible que se obtengan valores menores si el índice de coste del MBPC es modular, en cuanto a los indicadores ICE e ITEC están basados en el cuadrado del error y por tanto se da el caso opuesto.

5.3.2 Saturación del incremento de la acción de control

Se trata del mismo tipo de no linealidad que en el caso anterior pero esta vez sobre el incremento de la acción de control. Viene definida por la siguiente expresión (la representación gráfica es igual que la de la figura 5.8 cambiando u por Δu):

$$\Delta u_p(t) = \begin{cases} \Delta U_{max} & \text{Si } \Delta u(t) > \Delta U_{max} \\ \Delta u(t) & \text{Si } \Delta U_{min} \leq \Delta u(t) \leq \Delta U_{max} \\ \Delta U_{min} & \text{Si } \Delta u(t) < \Delta U_{min} \end{cases} \tag{5.5}$$

En este caso es complicado incluir la información de la no linealidad en la fase de codificación (ver modificación del espacio de búsqueda en el apartado 5.2.2). Se van a presentar los resultados obtenido de las dos alternativas restantes:

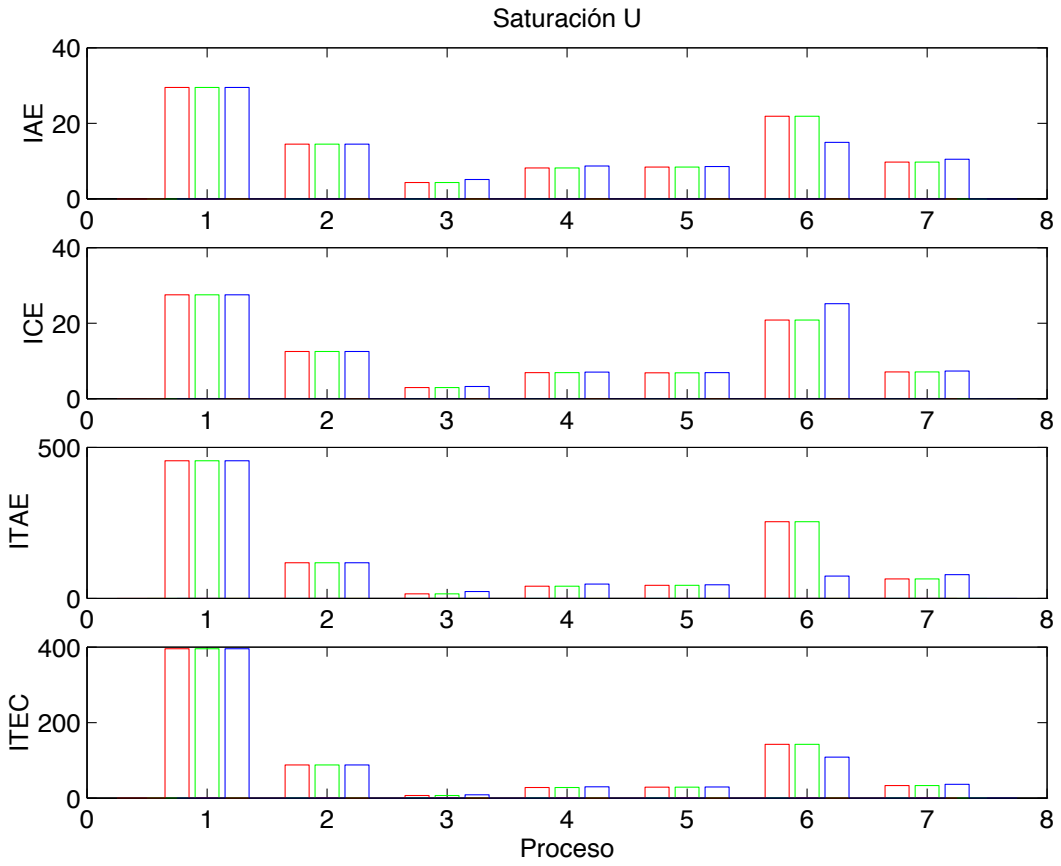


Figura 5.10: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

- Incorporando la no linealidad en el modelo del proceso (ver apartado 5.2.1).
- Se introduce la información como una restricción en el índice a minimizar, es decir, penalizando el índice cuando se produce una violación de la restricción (ver apartado 5.2.2).

Las características de las no linealidades que se van a aplicar a todos los proceso y el valor de los parámetros del Algoritmo Genético se recogen en la tabla 5.3. En todos los casos que siguen en el capítulo, se incluye la saturación en la acción de control puesto que esta aparece en la práctica en todos los problemas de control, como ya se ha descrito en el apartado anterior la información correspondiente a la saturación de u se tiene en cuenta en la codificación del espacio de búsqueda.

No linealidad	Saturación en $u(t)$ y en $\Delta u(t)$	
	$U_{min} = -2.5$	
	$U_{max} = 2.5$	
	$\Delta U_{min} = -0.5$	
	$\Delta U_{max} = 0.5$	
GA real	$NumIndiv.$	100
	$MAXGEN$	500
	P_c	0.7
	P_m	0.05

Tabla 5.3: Parámetros para la simulación del control de un proceso con saturación en $u(t)$ y $\Delta u(t)$.

Inclusión de la no linealidad mediante penalización

La función de coste que se minimiza es $J'(u)$:

$$\text{Índice cuadrático:} \qquad J(u) = \sum_{i=N_1}^{N_2} (y(t+i|t) - r(t+i))^2$$

$$\text{Índice modular:} \qquad J(u) = \sum_{i=N_1}^{N_2} |y(t+i|t) - r(t+i)|$$

$$J'(u) = J(u) + r\phi(u)$$

$\phi(u) = 0$ si Δu no viola las restricciones en caso contrario $\phi(u) = 1$ y el coeficiente de este término se ajusta en el valor $r = 10$.

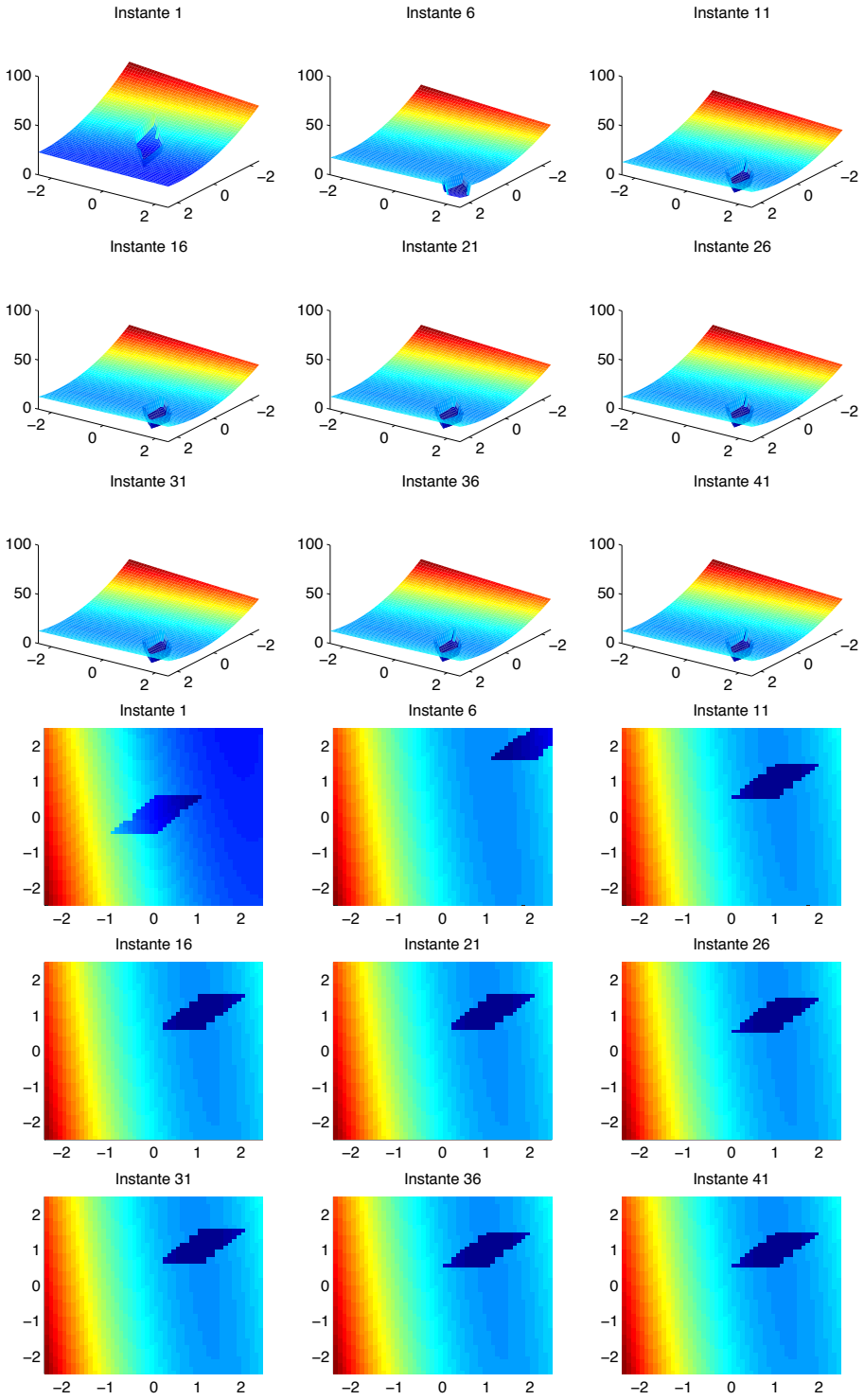


Figura 5.11: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con saturaciones en el incremento de la acción de control. Proceso $G_2(s)$, índice cuadrático.

El tipo de función a minimizar (utilizando índice cuadrático) para distintos instantes de muestreo y para el proceso $G_2(s)$ se puede ver en la figura 5.11.

Los resultados obtenidos para cada uno de los procesos se muestran en el anexo D en las figuras D.9, D.11, D.13, D.15, D.17, D.19 y D.21. El valor los indicadores: IAE, ICE, ITAE e ITEC que se representa gráficamente en la figura 5.12.

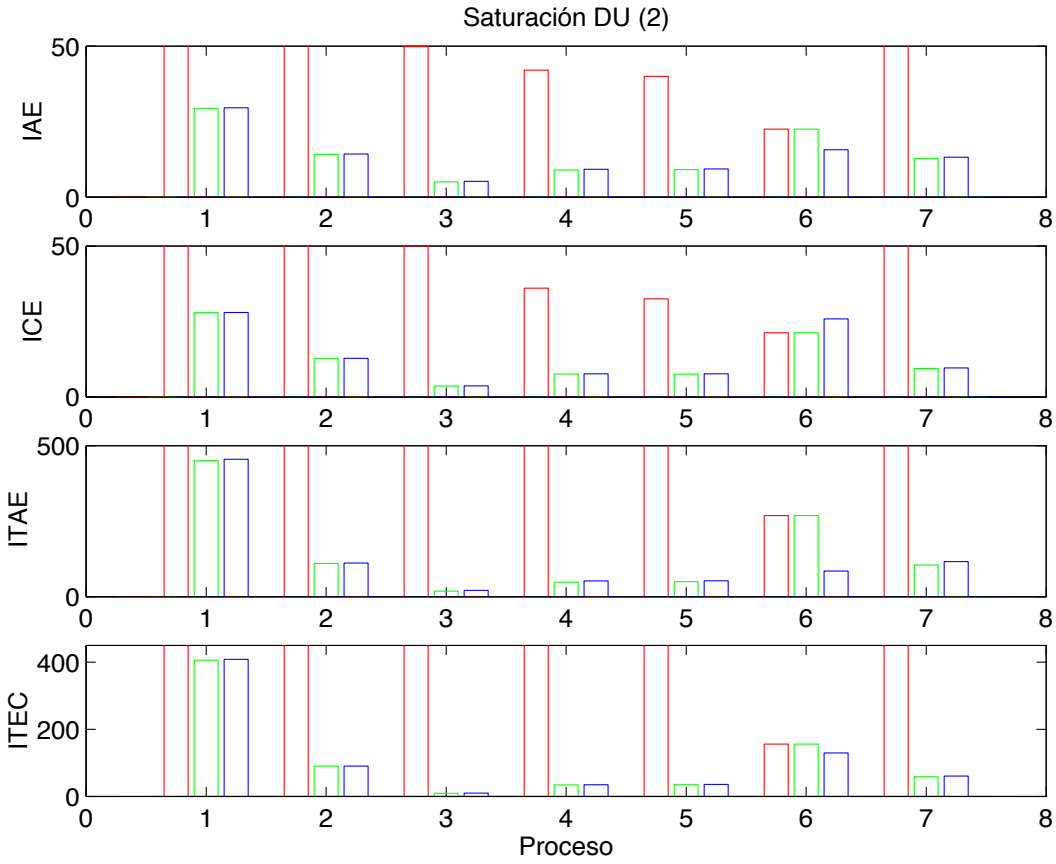


Figura 5.12: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con saturación en el incremento de la acción de control, la no linealidad se incorpora penalizando el índice de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Inclusión de la no linealidad en el modelo

Las condiciones de la simulación que se ha realizado son las mismas que en el caso anterior. La diferencia aparece en la función a minimizar, en este caso es $J(u)$ en lugar de $J'(u)$. Con la inclusión de la no linealidad en el modelo, el tipo de función a minimizar cambia

respecto al caso anterior, se obtienen funciones como las que se muestran en la figura 5.13.

Los resultados obtenidos para cada uno de los procesos se muestran en el anexo D en las figuras D.8, D.10, D.12, D.14, D.16, D.18 y D.20. El valor los indicadores: IAE, ICE, ITAE e ITEC que se representa gráficamente en la figura 5.12.

Aunque se trata del mismo problema, la resolución mediante penalización del índice o mediante inclusión de la no linealidad en el modelo de predicción cambia drásticamente el tipo de función a minimizar. Aún así, tanto en un caso como en otro, las funciones a minimizar que resultan no son convexas y por tanto el SQP no garantiza la localización de un solución adecuada. Esto queda reflejado en las simulaciones, sólo los Algoritmos Genéticos consiguen un control razonable del proceso.

En cuanto a la utilización del índice modular o cuadrático, dependiendo del indicador que se evalúe se obtienen mejores resultados con un índice u otro.

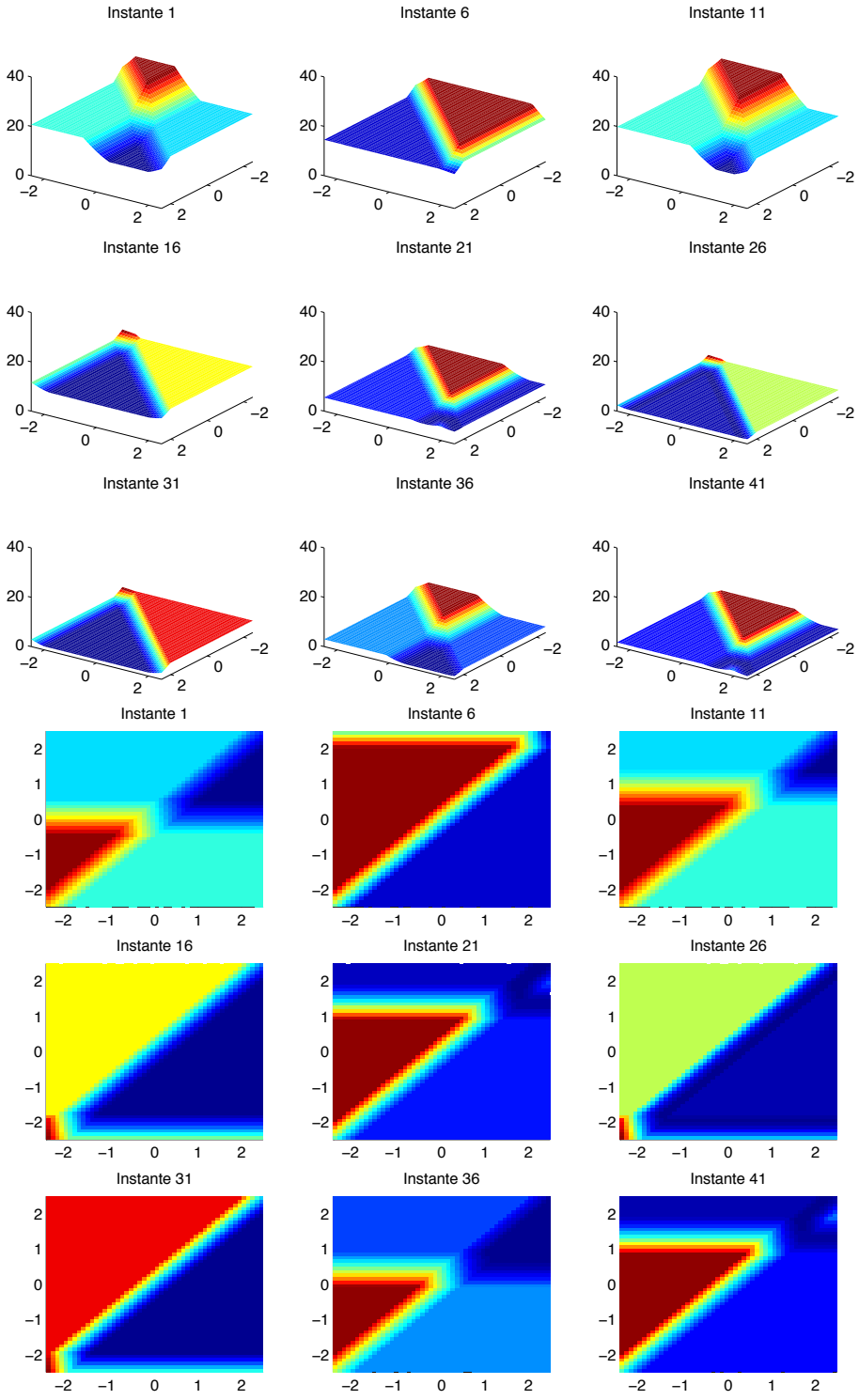


Figura 5.13: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con saturaciones en el incremento de la acción de control. Proceso $G_2(s)$, índice cuadrático.

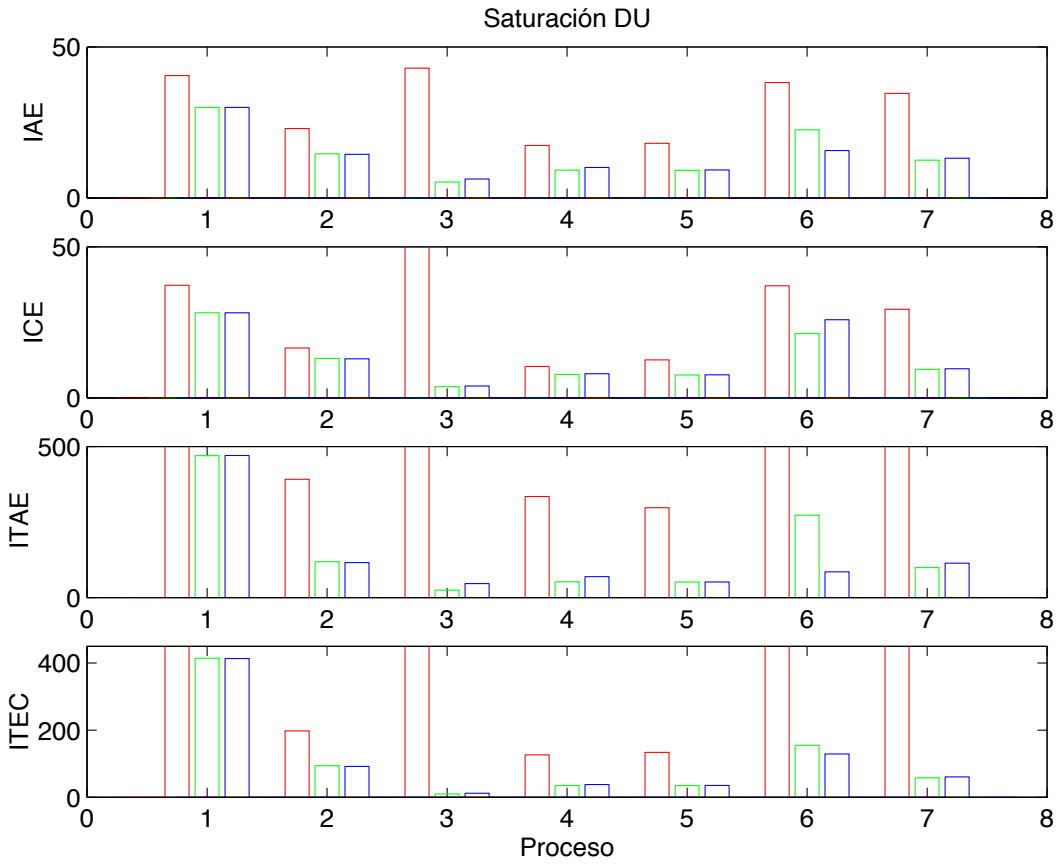


Figura 5.14: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con saturación en el incremento de la acción de control, la no linealidad se incorpora en el modelo. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

5.3.3 Zona muerta tipo I.

Las zonas muertas son también un caso muy común de no linealidades, tanto las tipo I como las de tipo II. Después de las saturaciones son las que se presentan con más frecuencia. Es muy común que los accionadores no sean capaces de suministrar al proceso valores muy bajos de acción de control y en el caso de que si que puedan, el problema es que es el proceso mismo el que no reacciona frente a entradas muy pequeñas (en este caso la no linealidad es del proceso pero se puede considerar en el accionador). La función de descripción de esta no linealidad corresponde:

$$u_p(t) = \begin{cases} m_d[u(t) - c_{d1}] & \text{Si } u(t) > c_{d1} \\ 0 & \text{Si } c_{d2} \leq u(t) \leq c_{d1} \\ m_d[u(t) - c_{d2}] & \text{Si } u(t) < c_{d2} \end{cases} \tag{5.6}$$

donde:

$u(t)$:	acción de control calculada por el regulador.
$u_p(t)$:	acción de control aplicada al proceso.
m_d:	pendiente de la recta. Relación entre $u_{p(t)}$ y $u(t)$ fuera de la zona muerta.
c_{d1}:	límite superior de la zona muerta.
c_{d2}:	límite inferior de la zona muerta.

En la figura 5.15 se puede ver la relación que hay entre $u(t)$ y $u_p(t)$ de forma gráfica.

Las características de la no linealidad y los parámetros del Algoritmo Genético se recogen en la tabla 5.4.

Se pueden tomar cualquiera de las alternativas que se han mencionado anteriormente para incorporar estas no linealidades en el control predictivo con optimización heurística. La opción de incluir la información en la fase de codificación tiene como desventaja principal el problema de codificar un espacio de búsqueda como:

$$u \in [U_{min}, c_{d2}] \cup 0 \cup [c_{d1}, U_{max}]$$

Una de las zonas a codificar consiste en un sólo punto. En cuanto a la opción de degradar el índice en la zona : $u \in]c_{d2}, 0[\cup]0, c_{d1}[$ presenta la desventaja de dejar el punto 0 aislado y rodeado por valores muy pobres del índice, esto dificulta la labor de cualquier algoritmo de optimización. Por tanto, se ha optado por incorporar la información de la no linealidad en el modelo del proceso. La función a minimizar para distintos instantes de muestreo y el proceso $G_2(s)$ se muestra en la figura 5.16.

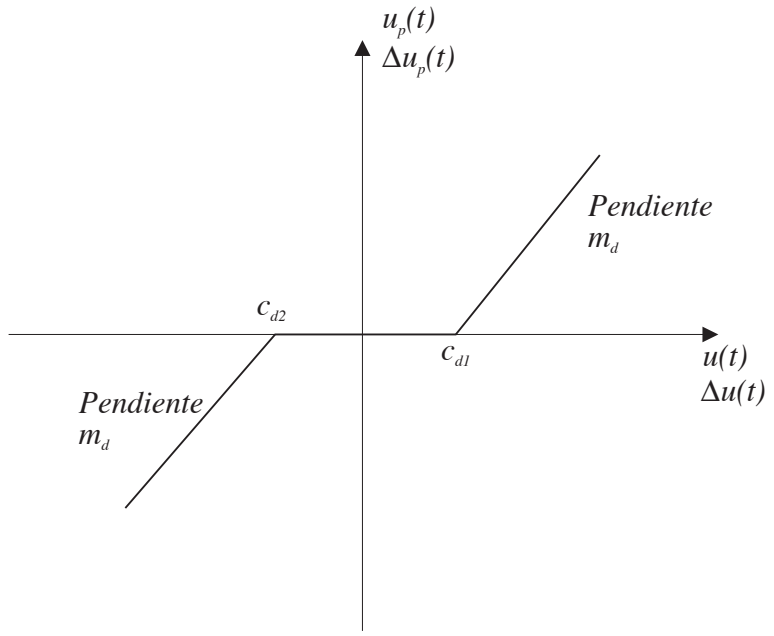


Figura 5.15: Relación entre $u(t)$ y $u_p(t)$ en una zona muerta tipo I.

No linealidad	Saturación en $u(t)$	
	$U_{min} = -5$	
	$U_{max} = 5$	
	Zona muerta tipo I	
	$m_d = 1$	
	$c_{d1} = 0.25$, $c_{d2} = -0.25$	
GA real	$NumIndiv.$	100
	$MAXGEN$	500
	P_c	0.7
	P_m	0.05

Tabla 5.4: Parámetros para la simulación del control de un proceso con saturación y zona muerta de tipo I en $u(t)$.

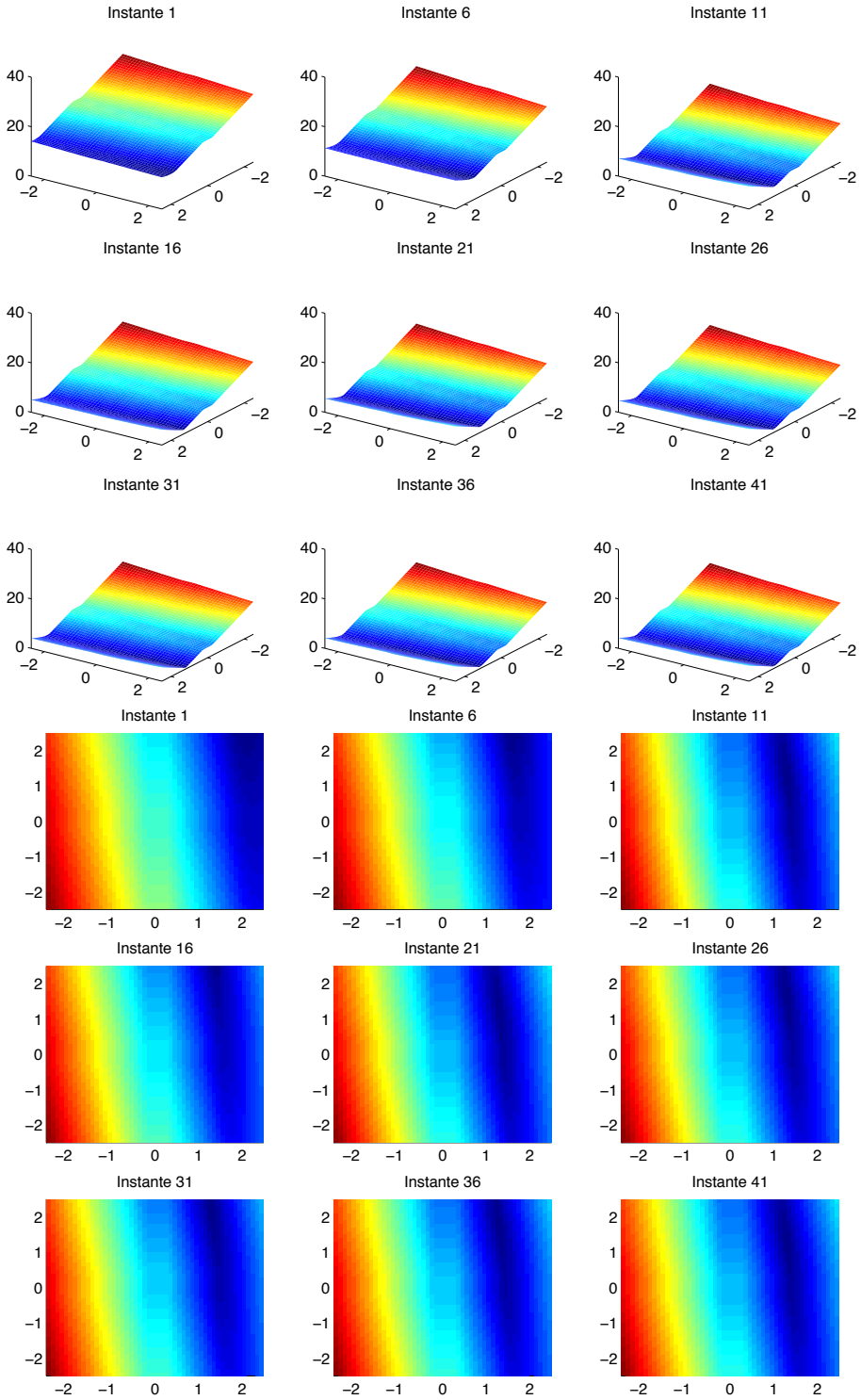


Figura 5.16: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con zona muerta de tipo I en la acción de control. Proceso $G_2(s)$, índice cuadrático.

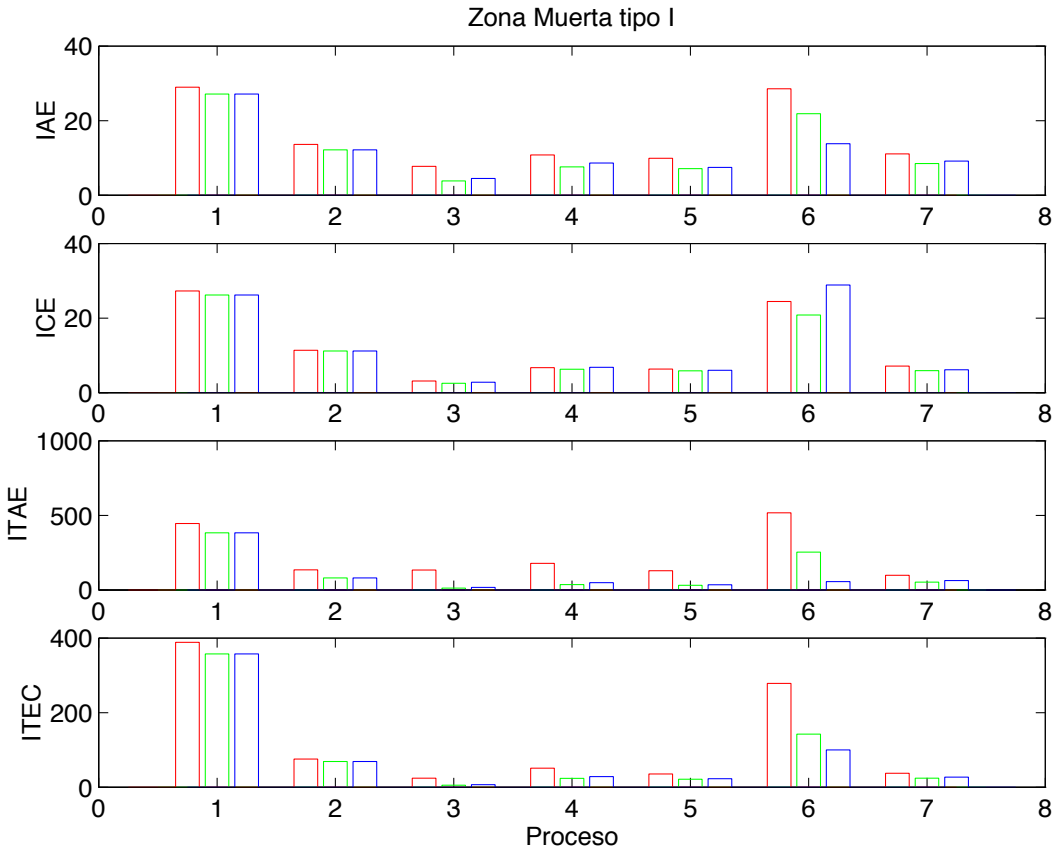


Figura 5.17: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con zona muerta de tipo I en la acción de control, la no linealidad se incorpora en el modelo. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Los resultados obtenidos para cada uno de los procesos se muestran en el anexo D en las figuras D.22, D.23, D.24, D.25, D.26, D.27 y D.28. El valor los indicadores: IAE, ICE, ITAE e ITEC que se representa gráficamente en la figura 5.17.

El control con el algoritmo SQP es el que presenta los peores resultados, la no linealidad provoca dificultades para este algoritmo debido a las zonas planas que provoca en la función de coste la no linealidad. De nuevo la elección de la función de coste modular o cuadrática depende del indicador utilizado para evaluar las calidad del control, en definitiva de las prestaciones que se quieren obtener.

5.3.4 Zona muerta tipo II.

Se trata de otro tipo de zona muerta. Una no linealidad como esta viene definida por la siguiente expresión:

$$u_p(t) = \begin{cases} m_d(u(t) - c_{d1}) + c_{d2} & \text{Si } u(t) > c_{d1} \\ 0 & \text{Si } c_{d3} \leq u(t) \leq c_{d1} \\ m_d(u(t) - c_{d3}) + c_{d4} & \text{Si } u(t) < c_{d3} \end{cases} \tag{5.7}$$

donde:

$u(t)$:	acción de control calculada por el regulador.
$u_p(t)$:	acción de control introducida en la planta.
m_d:	pendiente de la recta. Relación entre $u_{p(t)}$ y $u(t)$ fuera de la zona muerta.
c_{d1}:	límite superior de la zona muerta.
c_{d3}:	límite inferior de la zona muerta.
c_{d2}:	valor que alcanza $u(t)$ en el límite superior de la zona muerta.
c_{d4}:	valor que alcanza $u(t)$ en el límite inferior de la zona muerta.

En la figura 5.18 se puede ver la relación que hay entre $u(t)$ y $u_p(t)$ de forma gráfica.

Las características de la no linealidad y los parámetros del Algoritmo Genético se recogen en la tabla 5.5.

Este tipo de no linealidades se puede incorporar de la misma forma que la de tipo I, y al igual que en ese caso, parece más adecuado incorporarlas en el modelo del proceso. La función a minimizar para distintos instantes de muestreo y para el proceso $G_2(s)$ se muestra en la figura 5.19.

Los resultados obtenidos para cada uno de los procesos se muestran en el anexo D en las figuras D.29, D.30, D.31, D.32, D.33, D.34 y D.35. El valor los indicadores: IAE, ICE, ITAE e ITEC que se representa gráficamente en la figura 5.20.

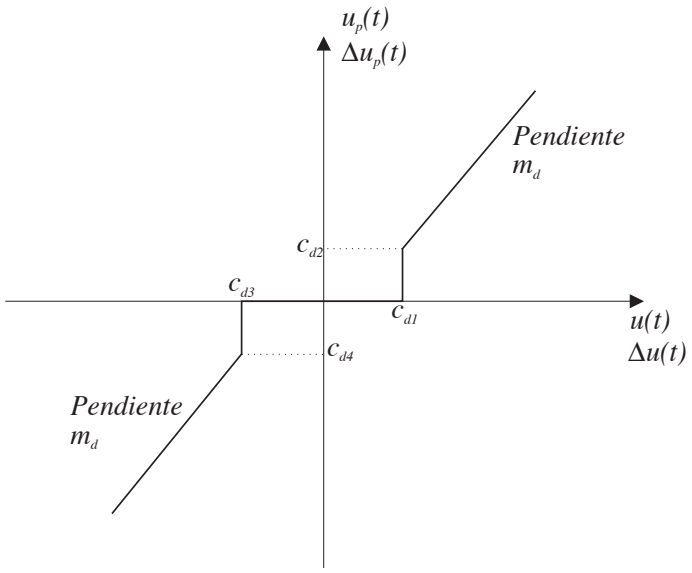


Figura 5.18: Relación entre $u(t)$ y $u_p(t)$ en una zona muerta tipo II.

No linealidad	Saturación en $u(t)$	
	$U_{min} = -2.5$	
	$U_{max} = 2.5$	
	Zona muerta tipo II	
	$m_d = 1$	
	$c_{d1} = c_{d2} = 0.25$	
	$c_{d3} = c_{d4} = -0.25$	
GA real	$NumIndiv.$	100
	$MAXGEN$	500
	P_c	0.7
	P_m	0.05

Tabla 5.5: Parámetros para la simulación del control de un proceso con saturación y zona muerta de tipo II en $u(t)$.

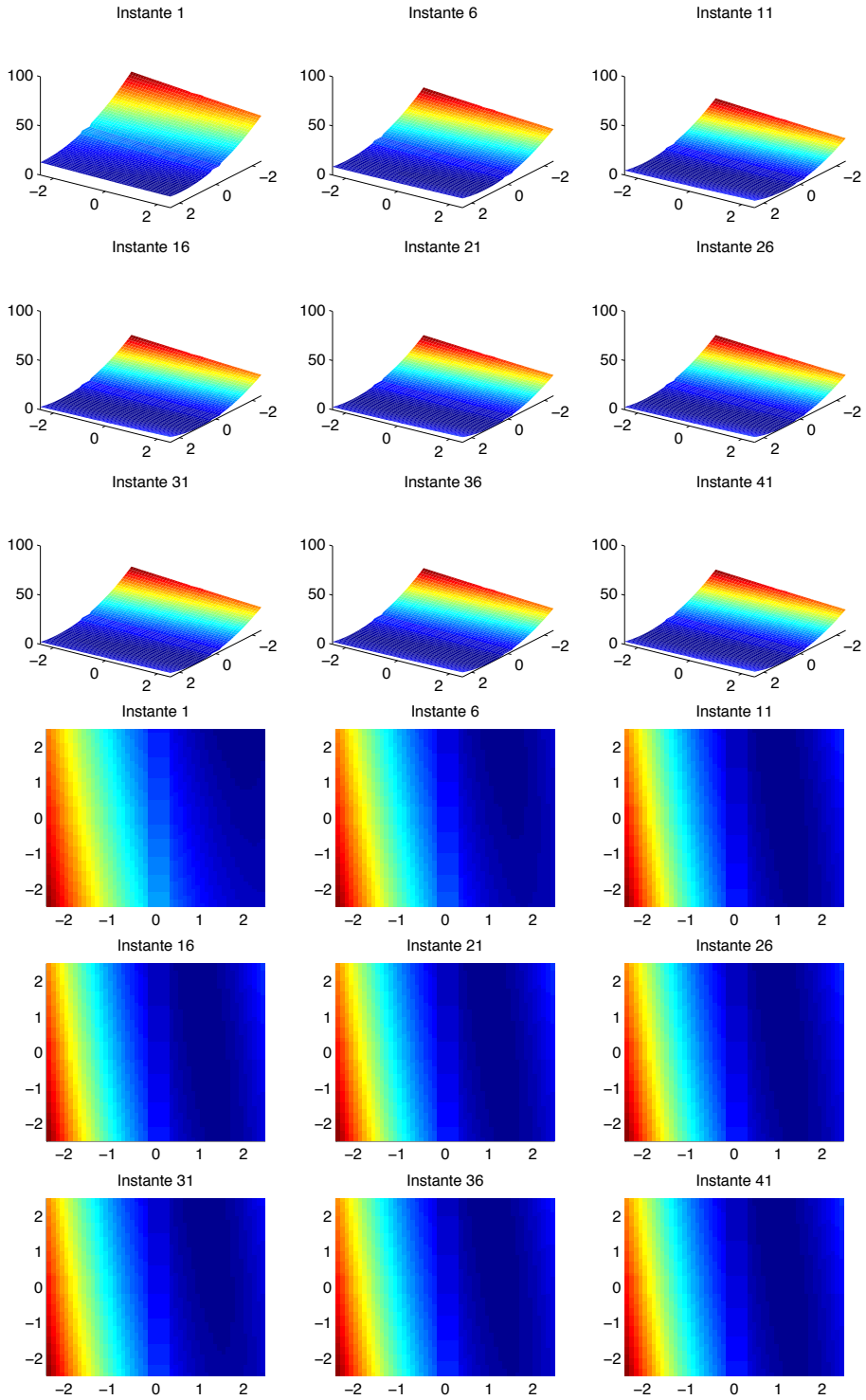


Figura 5.19: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con zona muerta de tipo II en la acción de control. Proceso $G_2(s)$, índice cuadrático.

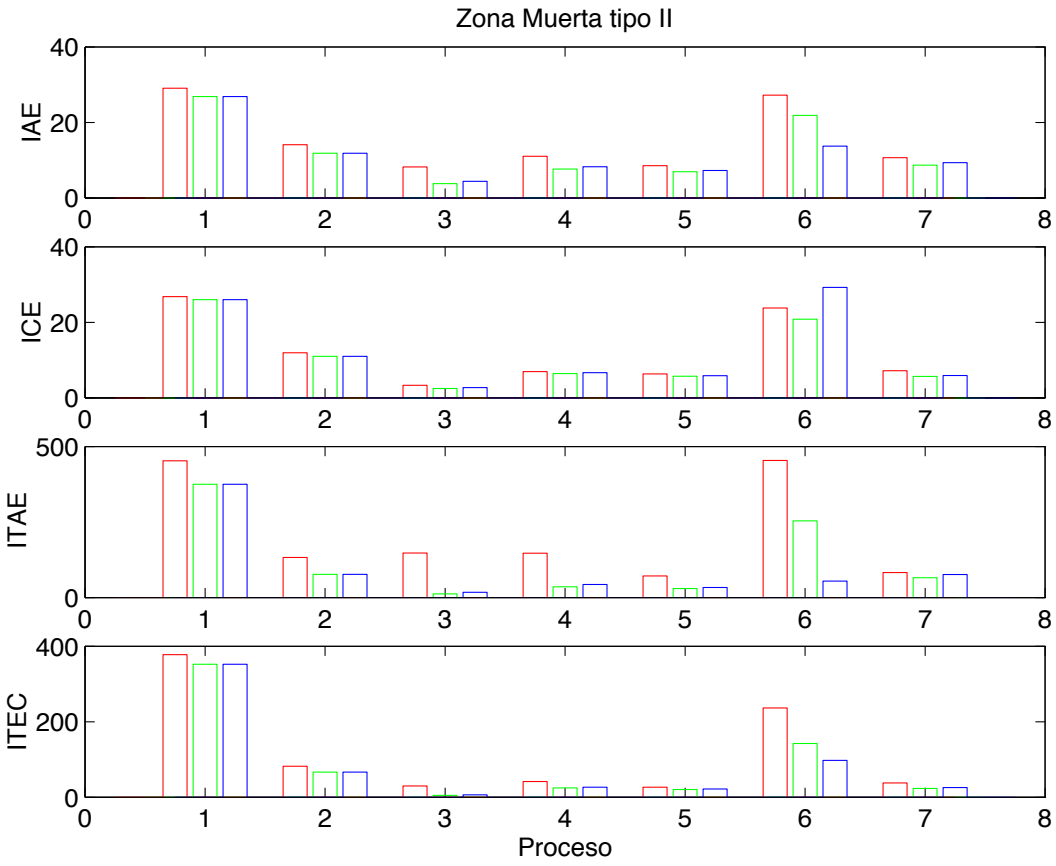


Figura 5.20: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con zona muerta de tipo II en la acción de control, la no linealidad se incorpora en el modelo. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Los resultados obtenidos son similares a los de las zonas muertas de tipo I: el algoritmo SQP tiene dificultades y no garantiza un control adecuado. El Algoritmo Genético si que consigue un control satisfactorio del proceso y dependiendo de las prestaciones que se quieran obtener es mejor un índice cuadrático o un índice modular.

5.3.5 Backlash

Este tipo de no linealidades suelen ir asociadas a sistemas con engranajes y su principal consecuencia es una especie de zona muerta sobre la acción de control cuando esta cambia de signo. En este caso, la expresión que relaciona $u_p(t)$ con $u(t)$ viene definido por la ecuación:

$$u_p(t) = \begin{cases} m_{b1}[u(t) - C_{b1}] & \text{Si } \Delta u(t) > d_p(t-1) \\ u_p(t-1) & \text{Si } d_u(t-1) \leq \Delta u(t) \leq d_p(t-1) \\ m_{b2}[u(t) - C_{b2}] & \text{Si } \Delta u(t) < d_u(t-1) \end{cases} \tag{5.8}$$

donde:

$$d_p(t) = \left[\frac{u_p(t)}{m_{b1}} + C_{b1}\right] - u(t) \tag{5.9}$$

$$d_u(t) = \left[\frac{u_p(t)}{m_{b2}} + C_{b2}\right] - u(t) \tag{5.10}$$

donde:

$u(t)$:	acción de control calculada por el regulador.
$u_p(t)$:	acción de control introducida en la planta.
$d_p(t)$:	límite superior de la no linealidad.
$d_u(t)$:	límite inferior de la no linealidad.
m_{b1} y C_{b1}:	parámetros de la recta que relaciona $u(t)$ y $u_p(t)$
m_{b2} y C_{b2}:	parámetros de la recta que relaciona $u(t)$ y $u_p(t)$.

En la figura 5.21 se puede ver la relación que hay entre $u(t)$ y $u_p(t)$ de forma gráfica.

Las características de la no linealidad y los parámetros del Algoritmo Genético se recogen en la tabla 5.6.

En esta ocasión se ha optado por incorporar la información de la no linealidad en el modelo de predicción, pues la aproximación basada en restricciones no es fácil de implementar y además cambia para cada instante de muestreo, este tipo de no linealidades depende de los valores en el instante anterior. La función a minimizar para distintos instantes de muestreo y para el proceso $G_2(s)$ se muestra en la figura 5.22.

Los resultados obtenidos para cada uno de los procesos se muestran en el anexo D en las figuras D.36, D.37, D.38, D.39, D.40, D.41 y D.42. El valor los indicadores: IAE, ICE, ITAE e ITEC que se representa gráficamente en la figura 5.23.

De nuevo la no linealidad provoca problemas en el control con el algoritmo SQP (la función a minimizar no es convexa), el Algoritmo Genético produce mejores resultados tanto con índice cuadrático como con índice modular.

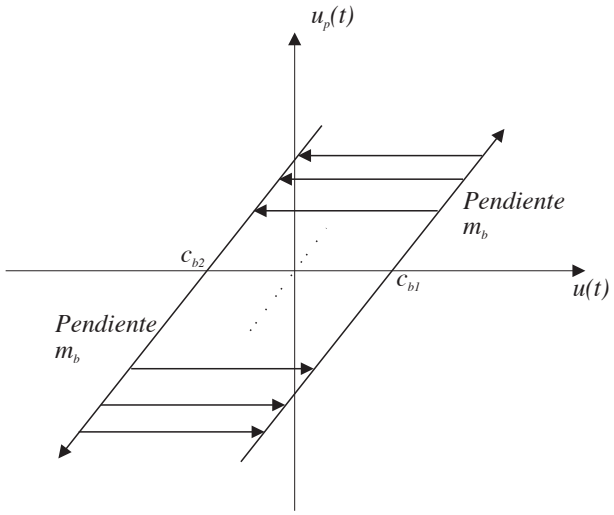


Figura 5.21: Relación entre $u(t)$ y $u_p(t)$ en presencia de un backlash.

No linealidad	Saturación en $u(t)$	
	$U_{min} = -2.5$	
	$U_{max} = 2.5$	
	Backlash	
	$m_{b1} = m_{b2} = 1$	
	$c_{b1} = 0.2$, $c_{b2} = -0.2$	
GA real	$NumIndiv.$	100
	$MAXGEN$	500
	P_c	0.7
	P_m	0.05

Tabla 5.6: Parámetros de simulación del control de un proceso con saturación y backlash en $u(t)$.

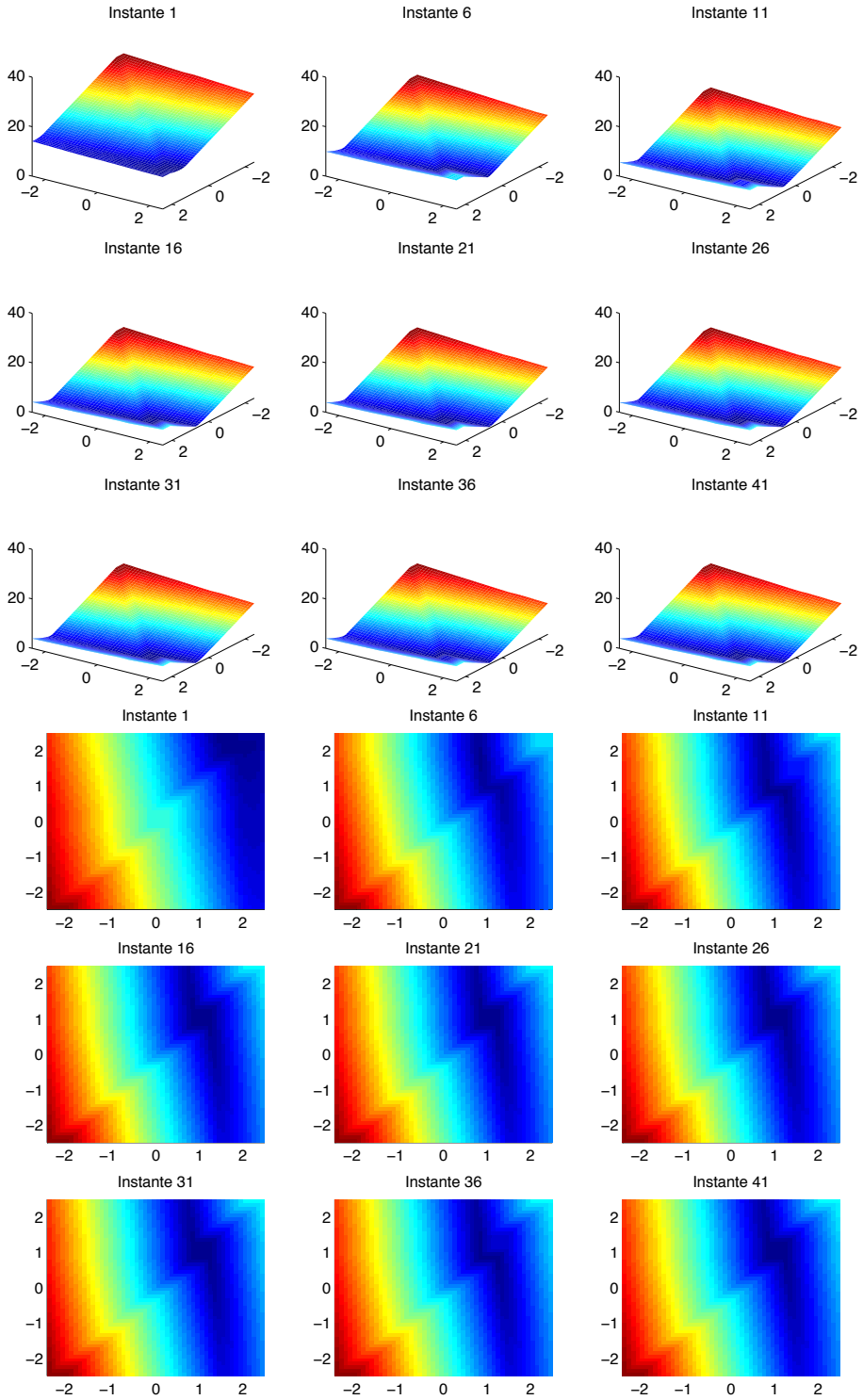


Figura 5.22: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con backlash en la acción de control. Proceso $G_2(s)$, índice cuadrático.

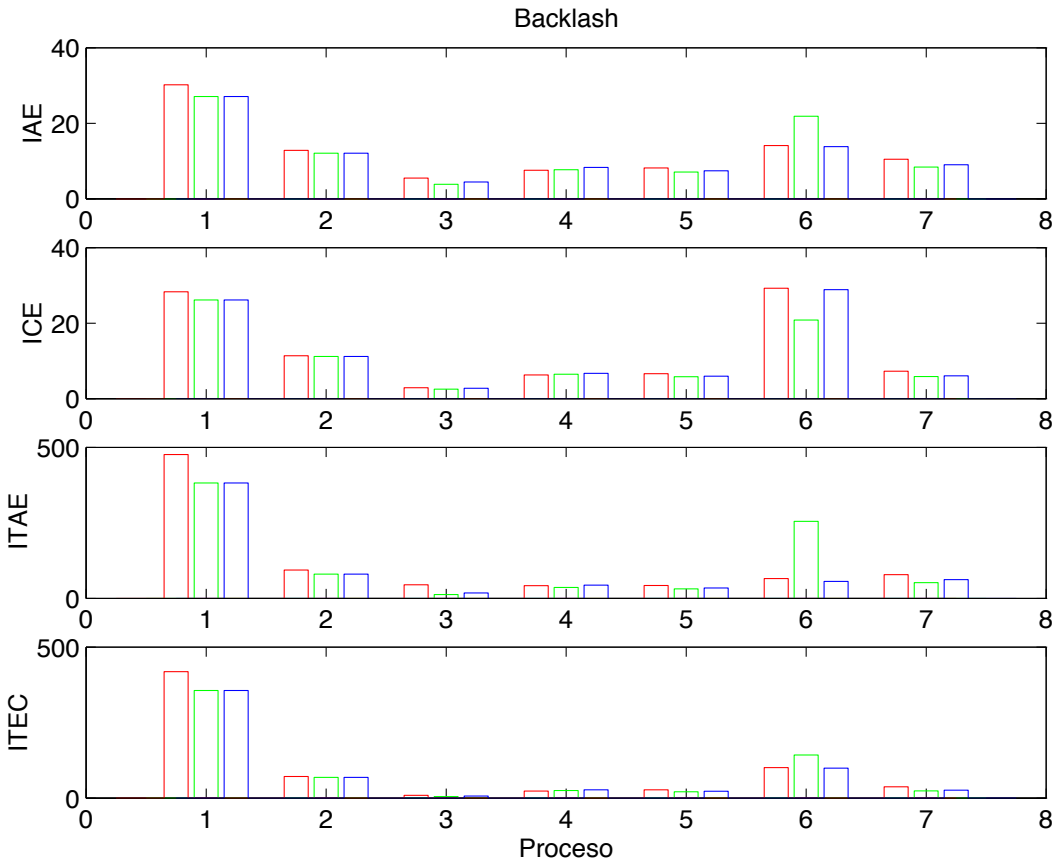


Figura 5.23: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con backlash en la acción de control, la no linealidad se incorpora en el modelo. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

5.3.6 Histéresis

La histéresis es un tipo de no linealidad que suele ir asociada a sistemas con elementos ferromagnéticos aunque este efecto se encuentra también en materiales plástico, piezoeléctricos y otros. Este tipo de no linealidad es complicada de modelar, por ello, aunque no existe una única formulación que describa todos los fenómenos de histéresis, la expresión 5.11 que relaciona $u_p(t)$ con $u(t)$ constituye un modelo relativamente simple para una gran variedad de estos fenómenos (ver [99]).

$$u_p(t) = \begin{cases} u_p(t-1) & \textbf{Si } u(t) = u(t-1) \\ m_t u(t) + c_t & \textbf{Si } u(t) \geq v_3, \textbf{o} \\ & \textbf{Si } m_t < m_b, \textbf{y} \\ & u_p(t-1) = m_t u(t-1) + c_t, \textbf{y} \\ & u(t-1) < u(t) < v_3 \\ m_b u(t) + c_b & \textbf{Si } u(t) \leq v_4, \textbf{o} \\ & \textbf{Si } m_t > m_b, \textbf{y} \\ & u_p(t-1) = m_b u(t-1) + c_b, \textbf{y} \\ & v_4 < u(t) < u(t-1) \\ m_t u(t) + c_d & \textbf{Si } v_d < u(t) < u(t-1) \\ m_b u(t) + c_u & \textbf{Si } u(t-1) < u(t) < v_u \\ m_l(u(t) - c_l) & \textbf{Si } v_d \geq u(t) \geq v_4 \\ m_r(u(t) - c_r) & \textbf{Si } v_u \leq u(t) \leq v_3 \end{cases} \tag{5.11}$$

donde:

$$\begin{aligned} c_d &= u_p(t-1) - m_t u(t-1) \\ c_u &= u_p(t-1) - m_b u(t-1) \\ v_d(t) &= \frac{m_l c_l + c_d}{m_l - m_t} \\ v_u(t) &= \frac{m_r c_r + c_u}{m_r - m_b} \end{aligned}$$

Los parámetros y variables que aparecen en las ecuaciones corresponden a:

$u(t)$:	acción de control calculada por el regulador.
$u_p(t)$:	acción de control introducida en la planta.
m_b y c_b:	parámetros que caracterizan la recta del lado inferior del cuadrilátero (pendiente y valor de corte con el eje u_p).
m_r y c_r:	parámetros que caracterizan la recta del lado derecho del cuadrilátero (pendiente y valor de corte con el eje u).
m_t y c_t:	parámetros que caracterizan la recta del lado superior del cuadrilátero(pendiente y valor de corte con el eje u_p).
m_l y c_l:	parámetros que caracterizan la recta del lado izquierdo del cuadrilátero(pendiente y valor de corte con el eje u).
v_1, v_2, v_3 y v_4:	valores de $u(t)$ en los extremos superior-izquierdo, inferior-derecho, superior-derecho e inferior-izquierdo del cuadrilátero respectivamente. Estos valores se pueden obtener a partir de los parámetros anteriores.

La figura 5.24 muestra la relación que hay entre $u(t)$ y $u_p(t)$, y aclara el significado de los parámetros.

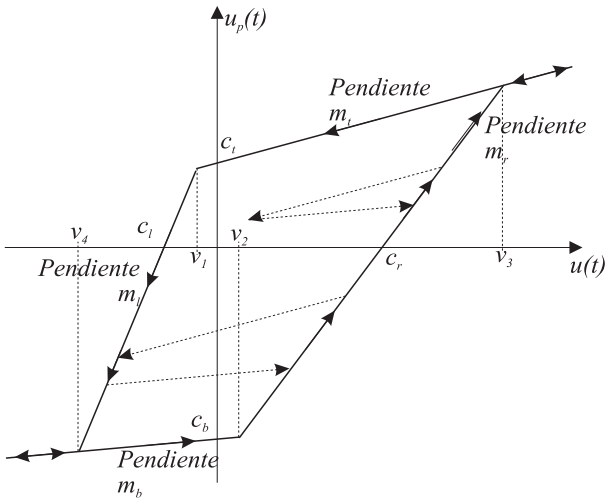


Figura 5.24: Relación entre $u(t)$ y $u_p(t)$ en presencia de una histéresis.

Las características de la no linealidad y los parámetros del Algoritmo Genético se recogen en la tabla 5.7.

La complejidad del modelo de este tipo de no linealidades hace muy difícil incorporarlo de otra manera que no sea en el modelo del proceso. Un ejemplo del tipo de funciones a minimizar para distintos instantes de muestreo correspondientes al proceso $G_1(s)$ se ve en la figura 5.25

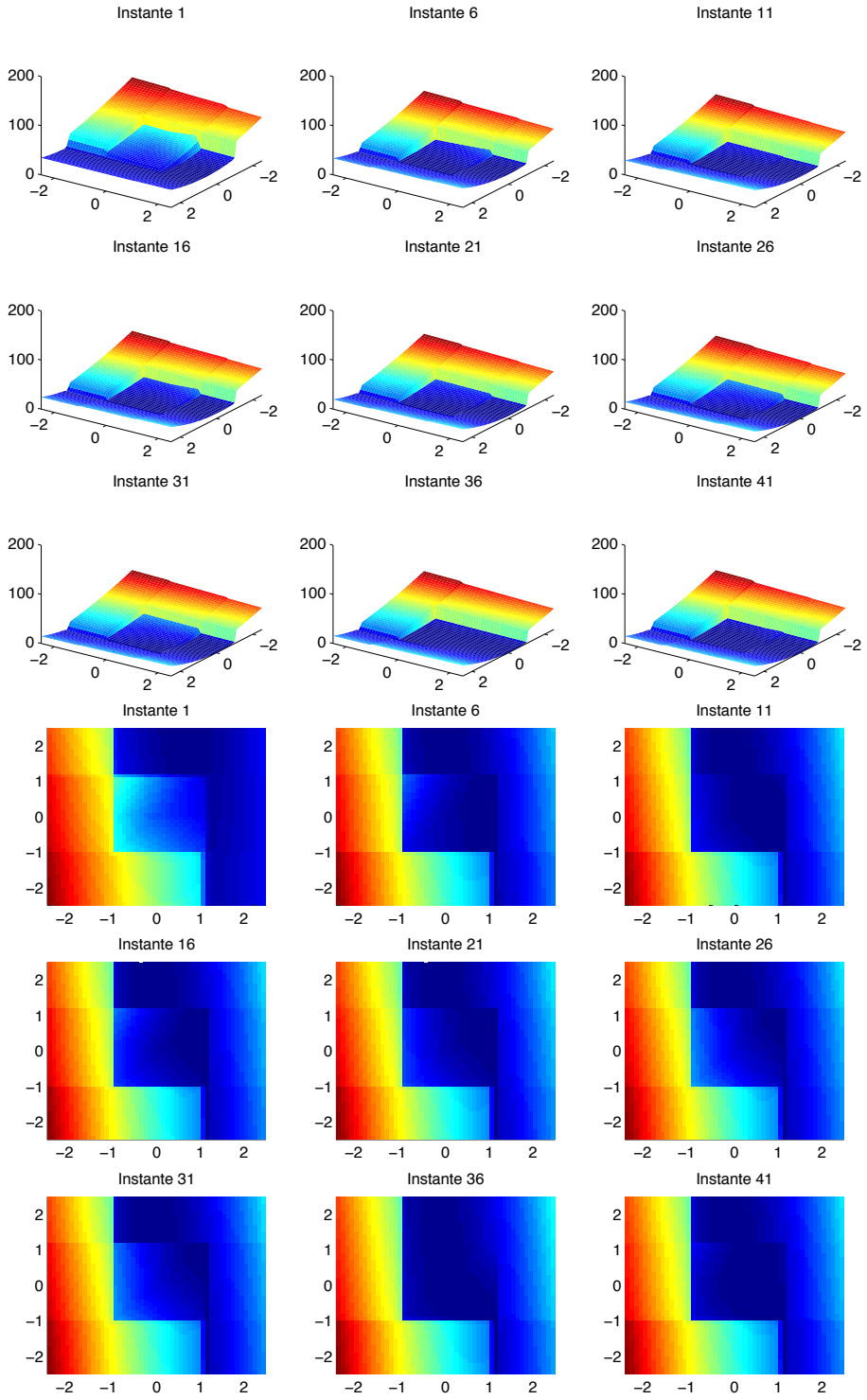


Figura 5.25: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con histéresis en la acción de control. Proceso $G_1(s)$, índice cuadrático.

No linealidad	Saturación en $u(t)$	
	$U_{min} = -2.5$	
	$U_{max} = 2.5$	
	Histéresis	
	$m_b = 0.5$; $c_b = -1$	
	$m_r = 10$; $c_r = 1$	
	$m_t = 0.8$; $c_t = 1$	
	$m_l = 15$; $c_l = -1$	
GA real	$NumIndiv.$	100
	$MAXGEN$	500
	P_c	0.7
	P_m	0.05

Tabla 5.7: Parámetros de simulación del control de un proceso con saturación e histéresis en $u(t)$.

Los resultados obtenidos para cada uno de los procesos se muestran en el anexo D en las figuras D.43, D.44, D.45, D.46, D.47, D.48 y D.49. El valor los indicadores: IAE, ICE, ITAE e ITEC que se representa gráficamente en la figura 5.26.

La no linealidad hace que la función a minimizar presente dificultades para el algoritmo SQP que no asegura la localización de una solución adecuada (la función no es convexa). Los Algoritmos Genéticos pueden resolver el problema con más garantías tanto para índices cuadráticos como modulares.

Resumen de las experiencias

En resumen, exceptuando la saturación en la acción de control (si el proceso no plantea otro tipo de dificultades), las no linealidades estudiadas requieren un algoritmo de optimización del tipo Algoritmos Genéticos. El SQP no puede resolver estos problemas puesto que las funciones a minimizar son casi siempre no convexas.

En cuanto a la inclusión de la no linealidad, la forma más general y sencilla es incorporar el modelo de la misma en el modelo del proceso. Sólo en el caso de no linealidades muy simples de modelar (por ejemplo, saturaciones) es preferible incorporarla en la modificación del espacio de búsqueda.

Comparando los resultados obtenidos con los índices modular y cuadrático (con Algoritmos Genéticos), para todos los proceso salvo el de fase no mínima, se obtienen valores muy similares de los indicadores evaluados (IAE, ICE, ITAE e ITEC), las diferencias son

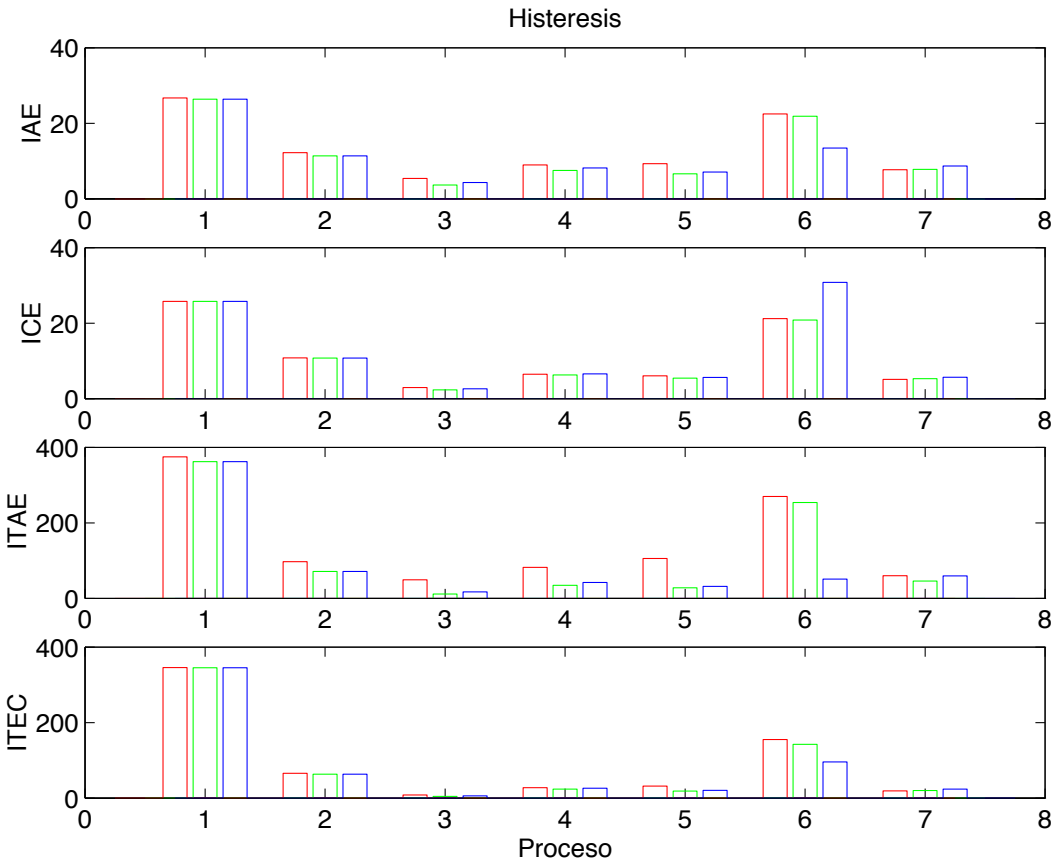


Figura 5.26: Valor de los indicadores IAE, ICE, ITAE e ITEC para los distintos proceso con histéresis en la acción de control, la no linealidad se incorpora en el modelo. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

mínimas y no determinan la utilización de uno u otro. Estas mínimas diferencias son debidas en gran parte a las saturaciones de la acción de control, en los casos estudiados durante el régimen transitorio las acciones de control suelen tomar los valores de saturación y por tanto el comportamiento será muy similar con un índice u otro, el problema es que no existe suficiente libertad en la acción de control para que la diferencia sea significativa entre un tipo de índice u otro. Sólo en el caso del proceso $G_6(s)$ se aprecian diferencias notables entre los resultados con índice modular e índice cuadrático, y en ese caso se observa que las acciones de control en el transitorio no toman valores de saturación, el controlador tiene margen de maniobra suficiente para que se aprecien las diferencias entre un índice u otro. Para este caso, el índice modular presenta mejores resultados salvo para el indicador ICE.

En definitiva, el tipo de índice que se debe utilizar depende de las prestaciones que se quieran obtener, en concreto de los indicadores que se vayan a utilizar para evaluar la calidad del control, e incluso, en la práctica, de los gustos o costumbres del operador que manipula el controlador. En cualquier caso, la gran ventaja de los Algoritmos Genéticos en particular y de la Optimización Heurística en general, es que no plantean, como lo hacen otros métodos, ninguna limitación en cuanto a las funciones de coste a minimizar ni del tipo de modelo que se emplea para diseñar el controlador incrementando así enormemente las posibilidades del MBPC.

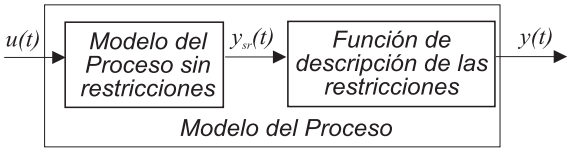


Figura 5.27: El modelo utilizado para el desarrollo del modelo de predicción incluye las restricciones de la salida.

5.4 MBPC con optimización heurística en procesos con restricciones en la salida

En este apartado se muestra dos formas de incluir las restricciones en las variables de salida en el MBPC. Antes de esto conviene diferenciar entre dos tipos de restricciones:

- **Restricciones del proceso**, esto es cuando las variables de salida no pueden físicamente tomar ciertos valores. Por ejemplo, en el nivel de un depósito de 3 metros de altura nunca podrá tomar un valor de 3.5 metros.
- **Restricciones de funcionamiento**, en este caso la restricción no es física se trata de una especificación de diseño del controlador. En el caso del depósito de 3 metros puede que una especificación de diseño sea que el nivel no debe alcanzar los 2.5 metros.

A efectos matemáticos, ambos casos no son más que un conjunto de restricciones sobre unas variables del problema de optimización aunque no conviene confundirlos. En principio, pueden ser incluidas de forma similar a las restricciones que aparecen con las no linealidades de los actuadores, es decir:

1. **Inclusión en el modelo de predicción.** Se añade al modelo del proceso sin restricciones una función de descripción de las restricciones de la variable de salida, la salida sin restricciones $y_{sr}(t)$ es filtrada por la función de descripción para obtener $y(t)$ (figura 5.27). El problema de optimización debe minimizar $J(u)$ sin restricciones pues están incluidas en el modelo.

   Esta alternativa puede presentar problemas de múltiples mínimos, se puede dar el caso de que distintos valores de las acciones de control den el mismo valor de la salida $y(t)$. En este caso siempre se puede establecer un criterio adicional para seleccionar la acción de control, por ejemplo, el de mínima energía.

   En el caso de restricciones de funcionamiento no está indicada esta alternativa pues no se trata de evitar que el modelo viole las restricciones sino que el proceso no viole

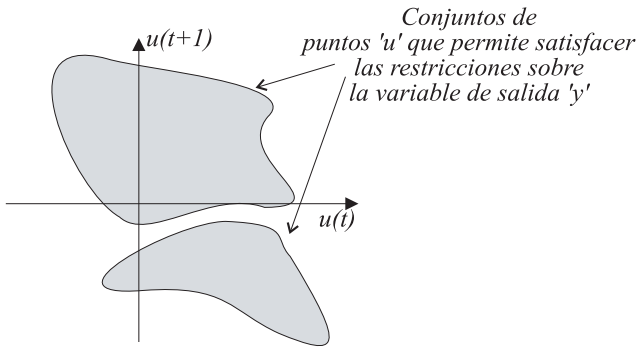


Figura 5.28: Ejemplo de dos dimensiones. El optimizador debe buscar el mínimo de $J(u)$ entre los valores del conjunto de puntos que satisfacen las restricciones de las variables de salida.

las restricciones. Al incluir las restricciones en el modelo la $y(t)$ que se obtiene no corresponde al comportamiento real del proceso[4] y por tanto puede ocurrir que $y(t)$ (respuesta del modelo con las restricciones) no viole las restricciones pero $y_{sr}(t)$ (respuesta del modelo del proceso sin restricciones que se supone modela perfectamente al proceso real) si que las viole.

Si la restricción es una restricción física del proceso no se plantea este problema puesto que incluyéndola en el modelo se consigue que $y(t)$ refleje correctamente el comportamiento real (físicamente el proceso está incapacitado para violar la restricción).

2. **Inclusión como restricciones.** En este caso el modelo utilizado para las predicciones no incluye las restricciones, en definitiva, $y(t) = y_{sr}(t)$. El problema de optimización se convierte en un problema con restricciones (figura 5.28):

$$\min\{J(u)\ ;\ u \in X\}$$

$$X = \{u\ :\ u \in R^{N_u}\ ;\ g(y) \leq 0\}$$

Las restricciones se establecen sobre las variables de salida y, pero pueden convertirse en restricciones sobre las variables de la entrada puesto que 'y' y 'u' están relacionados por el modelo del proceso.

Igual que en el caso de las no linealidades en los actuadores, aquí también se pueden utilizar dos alternativas:

- Función de penalización. Se penaliza el índice en función del cumplimiento o no de las restricciones sobre y. Se convierte en un problema de optimización sin restricciones cambiando la función a minimizar.

[4]Se supone que el modelo sin restricciones refleja correctamente el comportamiento real del proceso.

- Modificación del espacio de búsqueda. Se trata de obtener el subespacio 'X' de las variables 'u' que satisfacen las restricciones sobre 'y'. Este procedimiento presenta, a priori, cierta complejidad puesto que a partir de las variables de salida que satisfacen la restricciones se debe invertir el modelo del proceso para conseguir los valores de u que producen esas salidas.

En resumen, a efectos prácticos, la primera alternativa sólo está indicada para restricciones del proceso y la segunda alternativa está indicada sobre todo para las restricciones de funcionamiento.

5.4.1 Ejemplo: MBPC con restricciones de funcionamiento

En este apartado se muestran los resultados que se obtienen de un control MBPC con optimización heurística para un proceso con no linealidad en el actuador y restricción de funcionamiento en la variable de salida.

El proceso que se utiliza es el $G_6(s)$ de la tabla 5.1, se trata de un proceso de fase no mínima. Se incorpora además una no linealidad en el actuador que corresponde a una saturación y una zona muerta de tipo I en la acción de control caracterizadas por: $U_{min} = -5$, $U_{max} = 5$, $m_d = 1$, $c_{d1} = 0.25$, $c_{d2} = -0.25$. La saturación se tiene en cuenta limitando el espacio de búsqueda y la zona muerta se incorpora en el modelo del proceso.

La restricción en la salida corresponde a: $y(t) \geq -0.5$. Para incorporarla en el control se utiliza una función de penalización en el índice de coste.

$$J'(u) = J(u) + r\phi(y)$$

$$J(u) = \sum_{i=N_1}^{N_2} |y(t+i|t) - r(t+i)|$$

$\phi(y) = 0$ si $y \geq -0.5$ (no viola las restricciones) en caso contrario $\phi(y) = 1$ y el coeficiente de este término se ajusta en el valor $r = 25$. Con todo esto la funciones a minimizar que se obtienen para distintos instantes de muestreo se muestran en la figura 5.29, siendo claramente no convexas.

Los parámetros del Algoritmo Genético se ajustan según: $NumIndiv. = 100$, $MAXGEN = 500$, $P_c = 0.7$, $P_m = 0.05$. Los resultados obtenidos se muestran en la figura 5.30

Se observa que la utilización de un Algoritmo Genético permite incluir restricciones con facilidad obteniéndose resultados adecuados.

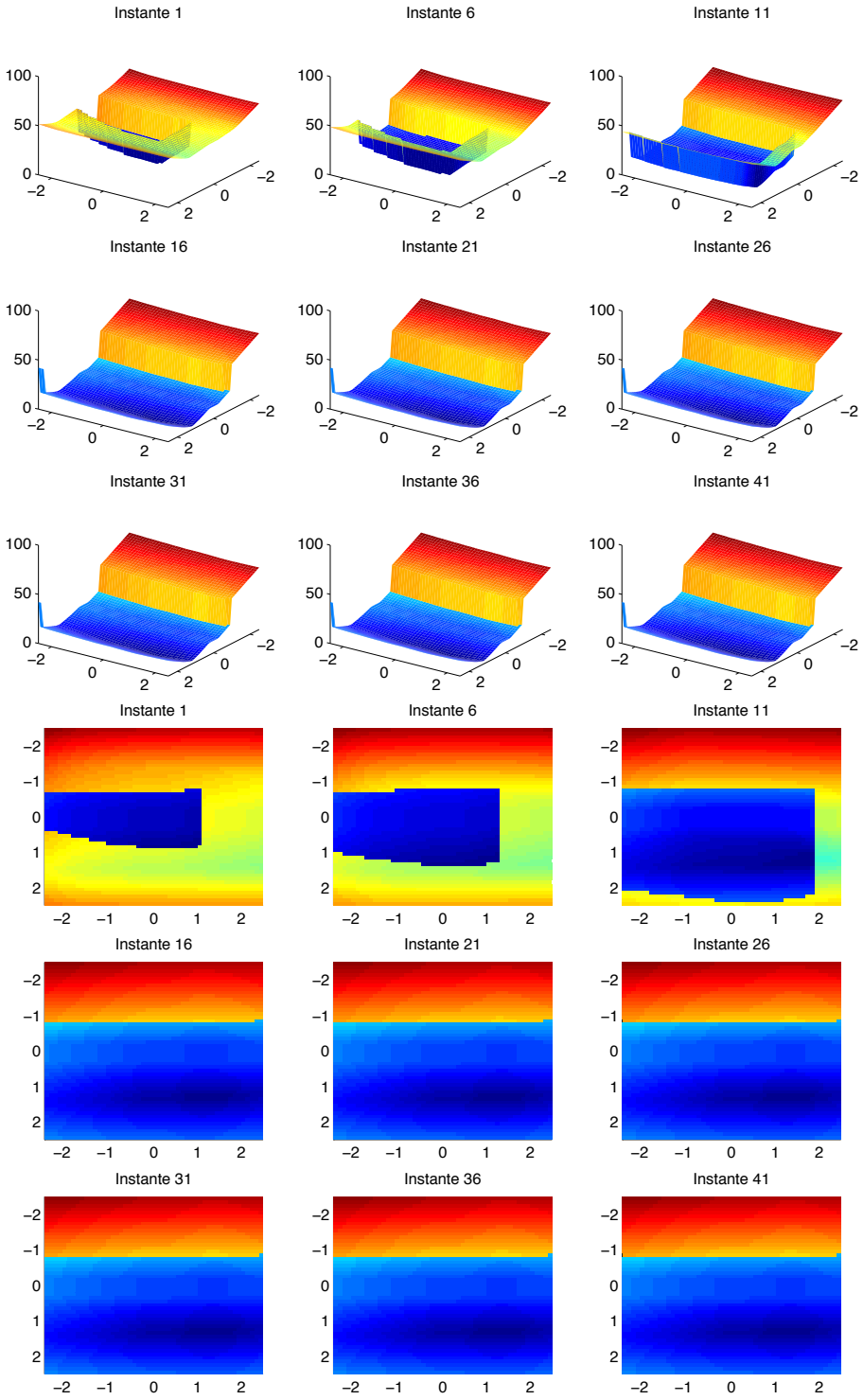


Figura 5.29: Representación gráfica de las funciones a minimizar en distintos instantes para un proceso con saturaciones y zona muerta de tipo I en la acción de control. Proceso $G_6(s)$ con restricción $y \geq -0.5$ en la salida, índice modular.

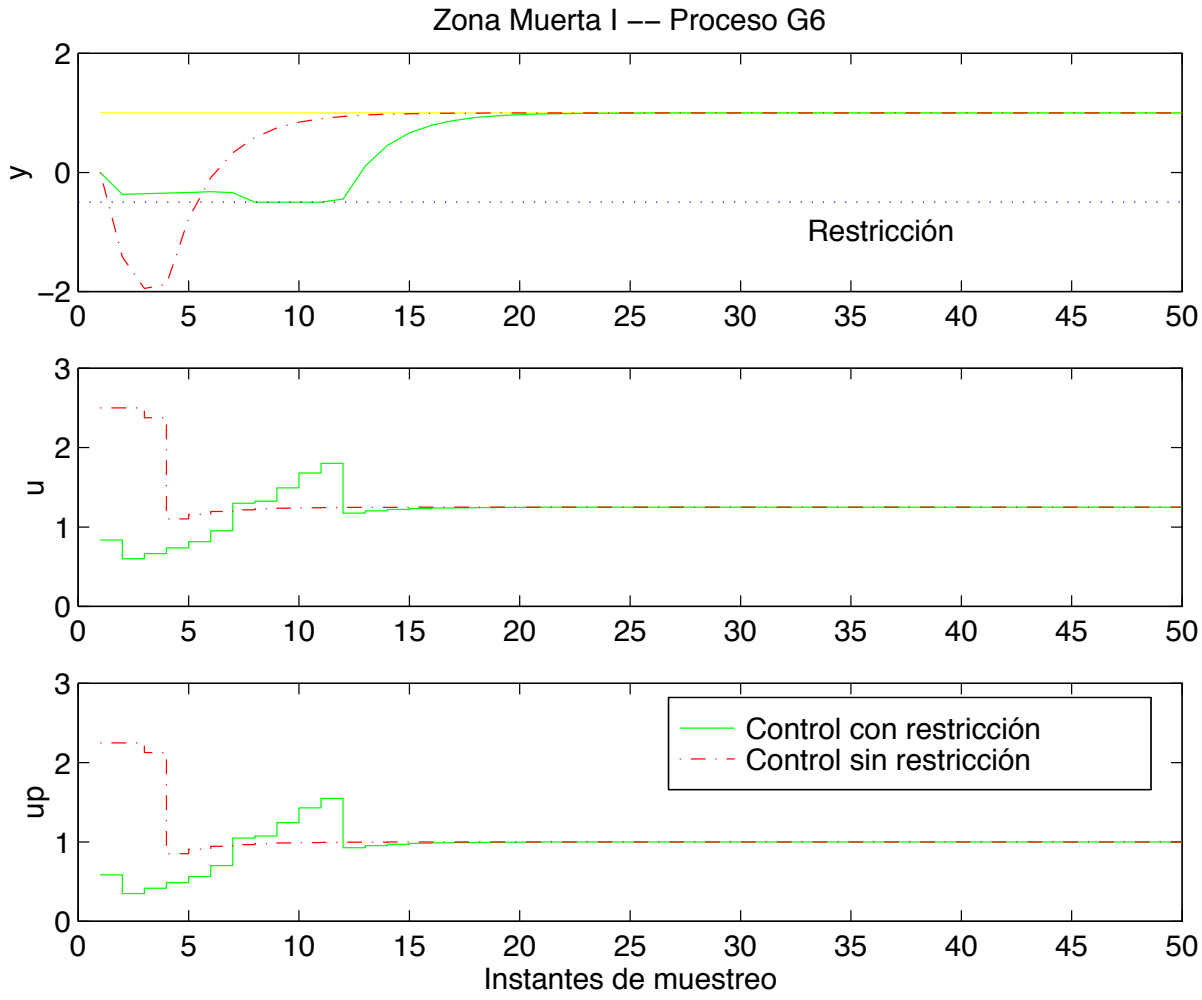


Figura 5.30: Control del un proceso $G_6(s)$ con saturaciones y zona muerta de tipo I en la acción de control y restricción en la variable de salida. Comparación con el control sin restricciones.

5.5 MBPC con optimización heurística con modelos de procesos no lineales

En los apartados anteriores se han tratado procesos lineales no linealidades en los actuadores y restricciones en la variable de salida. En este apartado se plantea la aplicación del control MBPC con optimización heurística al control de procesos no lineales, siempre y cuando se disponga de una adecuado conocimiento del mismo, es decir, un modelo que refleje dichas no linealidades. En principio, el modelo puede ser de cualquier tipo, basta que sea capaz de suministrar N_2 salidas futuras (predicción de la respuesta del proceso a lo largo del horizonte de predicción) a partir de unas entradas dadas.

La primera opción para resolver el control de un proceso no lineal mediante la aplicación una metodología MBPC, consiste en realizar una linealización del proceso en torno a un punto de funcionamiento y realizar el diseño del controlador MBPC con este modelo lineal. Si la calidad del control que se obtiene no es suficiente cabe la posibilidad de aplicar una metodología de Planificación de Ganancia, linealizando en torno a varios puntos con lo que se dispone de un conjunto de modelos lineales que permiten diseñar un controlador MBPC lineal más o menos adecuado para cada una de las zonas, pero es necesario habilitar un mecanismo para cambiar de un control a otro según el valor de la variable. Este forma de operar puede ser suficiente para conseguir un control adecuado en algunas aplicaciones, pero en cualquier caso no se trata de un diseño simple puesto que requiere: seleccionar las distintas zonas de funcionamiento y obtener un modelo lineal ajustado en cada zona, diseñar para cada una de ellas el control MBPC lineal adecuado y ajustar cuidadosamente la forma de cambiar de un controlador a otro. En el mejor de los casos, la utilización de varios controladores MBPC lineales puede conseguir una calidad similar a la del MBPC con el modelo no lineal.

La inclusión de un optimizador heurístico, en primer lugar, simplifica notablemente el diseño del regulador, la estructura del controlador no varía, tanto si se utiliza un modelo lineal como si el modelo es no lineal, en segundo lugar, permite utilizar la información del modelo no lineal sin necesidad de realizar aproximaciones lineales lo cual produce mejores predicciones y por tanto cabe esperar un control de mayor calidad. Por último, como ya se ha mostrado en apartados anteriores, la utilización de un optimizador heurístico permite la inclusión, de forma muy simple, de no linealidades en los actuadores y restricciones en la salida.

En el siguiente punto se muestra un ejemplo de utilización de un modelo no lineal en espacio de estados para realizar un control MBPC con un optimizador heurístico. Cabe destacar que aunque se utilice un modelo en espacio de estados (por tratarse de la forma más común de disponer de un modelo no lineal), no se presupone que el estado es medible, ni que existe un observador del mismo. Se pretende controlar el proceso con la información de entrada/salida, esto permite intercambiar este tipo de modelos no lineales

por otros sin perder validez la estructura que se plantea. En cualquier caso, si en algún problema concreto es posible medir alguno de los estados del proceso esta información puede ser fácilmente introducida en el modelo sin que ello suponga cambios sustanciales en el planteamiento del controlador.

El utilizar representación en espacio de estados es una forma usual y razonable de modelar un proceso no lineal, pero no la única. En el planteamiento que se realiza, no se excluye cualquier otra forma de modelado no lineal (por ejemplo, redes neuronales, fuzzy, etc.). Sea cual sea la forma del modelo el controlador sólo requiere información entrada/salida. La complejidad del problema se traslada a la optimización pero el Algoritmo Genético ha mostrado una gran robustez ante una gran variedad de problemas de optimización, tal y como se demostró en el capítulo dedicado a la evaluación de GA y SA.

5.5.1 Ejemplo de control MBPC para un proceso no lineal

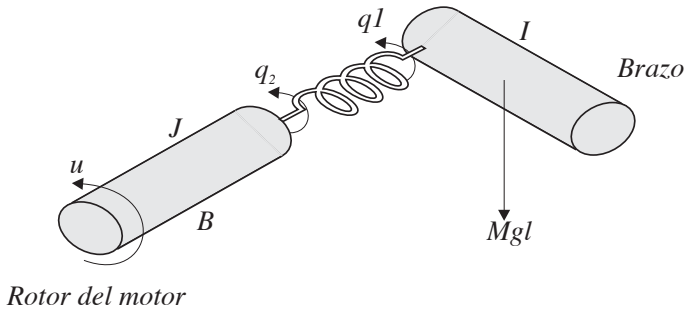


Figura 5.31: Brazo robot con unión flexible.

El sistema que se va ha tratar de controlar es un brazo con una unión flexible (figura 5.31) cuya dinámica viene dada por las siguientes ecuaciones:

$$\begin{aligned} I\ddot{q}_1 + Mgl\sin(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 + B\dot{q}_2 - K(q_1 - q_2) &= u \end{aligned}$$

Donde:

- q_1 y q_2 representan el desplazamiento angular del extremo del brazo y del eje del motor respectivamente.
- u es el par generado por el motor.
- I, J, B, Mgl y K son parámetros físicos del sistema cuyos valores aparecen en la tabla 5.8.

Parámetro	Valor
I	$0.032\ Kg.m^2$
J	$0.004\ Kg.m^2$
B	$0.007\ N.m.seg/rad$
Mgl	$0.8\ N.m$
K	$7.13\ N.m/rad$

Tabla 5.8: Valores nominales de los parámetros del brazo.

Tomando como variables de estado:

- x_1: posición del brazo q_1(en radianes).
- x_2: velocidad angular del brazo (en rad/seg).
- x_3: posición del rotor del motor q_2 (en radianes).
- x_4: velocidad angular del rotor del motor (en rad/seg).

Obtenemos las siguiente representación en espacio de estados:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -M_1 \sin(x_1) - K_1(x_1 - x_3) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= -B_1 x_4 + K_2(x_1 - x_3) + u/J \\
y &= x_1
\end{aligned}$$

Donde:

$$M_1 = \frac{Mgl}{I} \quad , \quad K_1 = \frac{K}{I}$$
$$B_1 = \frac{B}{J} \quad , \quad K_2 = \frac{K}{J}$$

A partir de este modelo continuo, obtenemos el modelo del proceso que se utilizará para las predicciones. Discretizando por el método de Euler con un periodo de muestreo de $T = 0.01$ segundos obtenemos:

$$\begin{aligned}
x_1(k+1) &= x_1(k) + T x_2(k) \\
x_2(k+1) &= x_2(k) - T M_1 \sin(x_1(k)) + T K_1(x_1(k) - x_3(k)) \\
x_3(k+1) &= x_3(k) + T x_4(k) \\
x_4(k+1) &= x_4(k) - T B_1 x_4(k) + T K_2(x_1(k) - x_3(k)) + T u(k)/J \\
y(k) &= x_1(k)
\end{aligned}$$

Evidentemente, la calidad de este modelo depende del periodo de muestreo, cuanto menor sea, mejor será el modelo y por lo tanto las predicciones serán mejores. Para verificar la calidad del modelo discreto se ha realizado una simulación con el mismo y se compara con la respuesta del proceso. La figura 5.32 muestra que existe una discrepancia y por tanto un error de modelado.

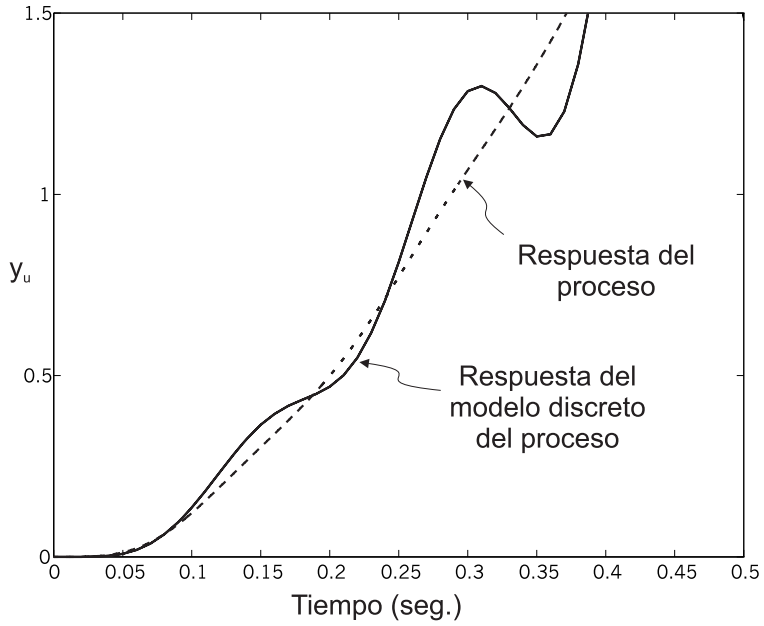


Figura 5.32: Respuesta del proceso y del modelo discreto para una entrada $u(t) = 1$.

En cuanto al modelo de perturbación que se utiliza, viene dado por la siguiente expresión:

$$n(t) = \frac{1}{\Delta}\xi(t) \tag{5.12}$$

La simulación se ha realizado con los siguientes parámetros: $N_u = 1$ ($N_u = 2$ en la segunda simulación), $N_1 = 1$, $N_2 = 10$. El optimizador es un algoritmo genético con codificación binaria con las siguientes características: 40 individuos por generación, 20 generaciones, cromosomas de 16 bits y limitando la acción de control entre ± 50.

Para mostrar las mejoras obtenidas frente a un control MBPC lineal, se compara con la respuesta que se obtiene con un GPC que se calcula a partir de un modelo linealizado alrededor del punto:

$$(x_1, x_2, x_3, x_4, u) = (0, 0, 0, 0, 0)$$

El modelo continuo que se obtiene es:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -M_1 - K_1 & 0 & K_1 & 0 \\ 0 & 0 & 0 & 1 \\ K_2 & 0 & -K_2 & -B_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/J \end{pmatrix} u$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Los resultados de la simulación para $N_u = 1$ se muestran en la figura 5.33 y para $N_u = 2$ en la figura 5.34.

Como era previsible, en la figura 5.33 se observa un mejor comportamiento cuando se utiliza el modelo no lineal, el transitorio presenta menos oscilaciones y tiempo de establecimiento. En el caso de la figura 5.34 el GPC convencional no consigue ni siquiera una respuesta estable. En definitiva, la calidad del control es mayor utilizando el modelo no lineal.

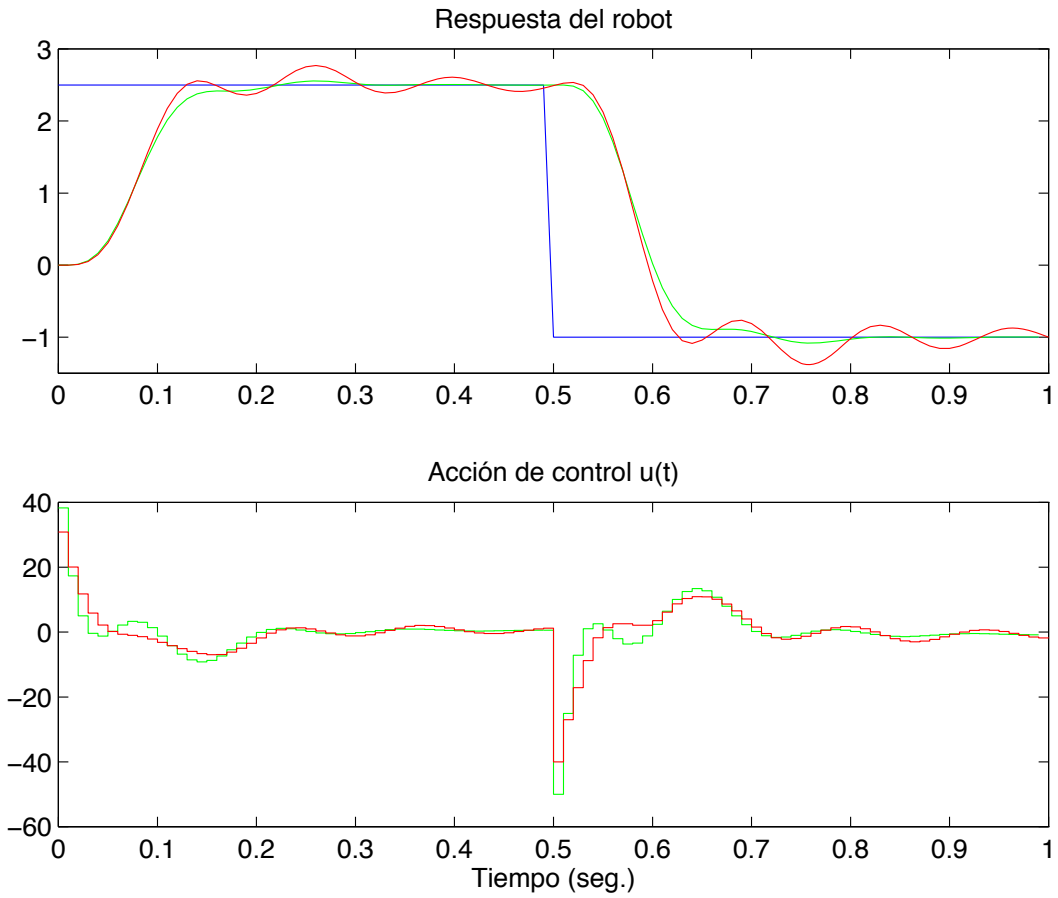


Figura 5.33: Resultados del control predictivo con $N_u = 1$ con un GPC convencional (en rojo) y un control predictivo con optimización heurística (en verde).

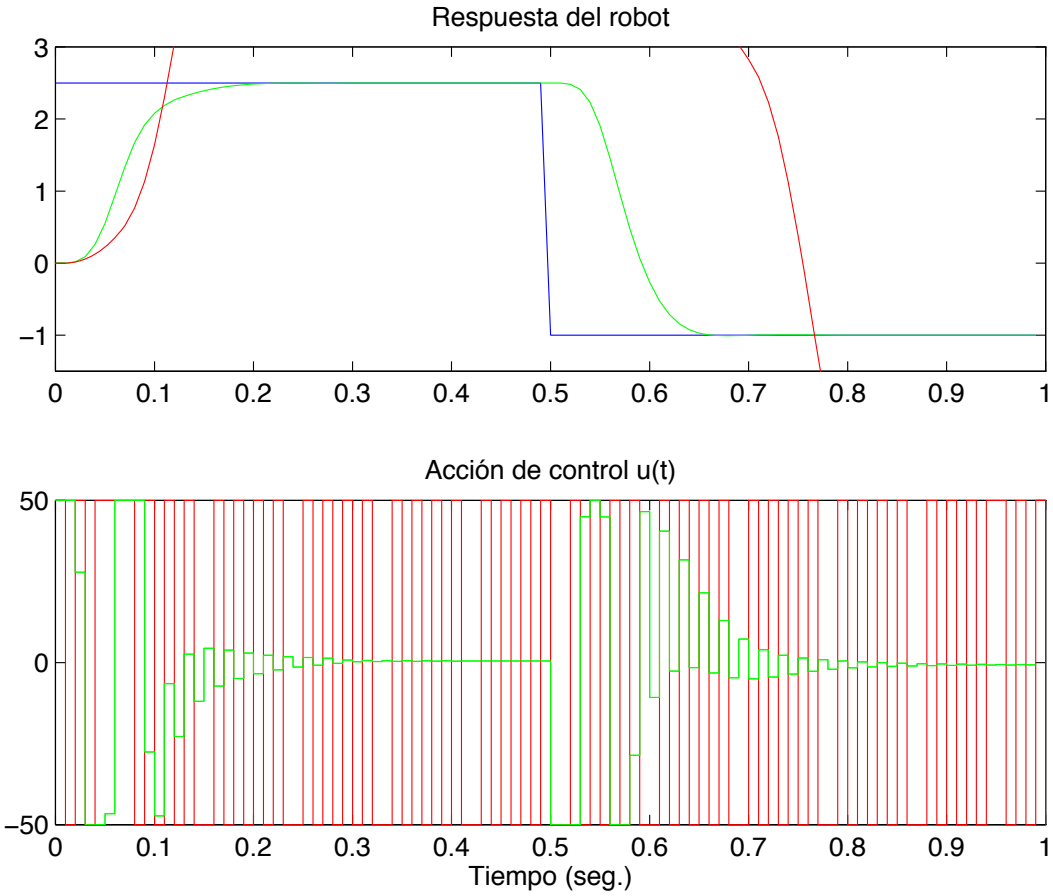


Figura 5.34: Resultados del control predictivo con $N_u = 2$ con un GPC convencional (en rojo) y un control predictivo con optimización heurística (en verde).

5.6 Conclusiones

Este capítulo ha mostrado algunas de las posibilidades que aparecen en el control predictivo por el hecho de utilizar Algoritmos Genéticos como técnica de optimización. Hay que recordar que los beneficios que se obtienen con los Algoritmos Genéticos no son generalizables a cualquier técnica de optimización heurística, la calidad y limitaciones de este tipo de algoritmos ya ha sido estudiado en un capítulo anterior donde se mostraba como la herramienta más potente de entre las analizadas.

En este capítulo se ha utilizado un Algoritmo Genético con codificación real y en algunas partes del mismo se ha comparado con otra técnica de optimización muy extendida como el algoritmo SQP. La cualidad más importante para su aplicación al control predictivo es la capacidad para enfrentarse a una gran variedad de problemas de optimización con resultados razonables tal y como se muestra en este capítulo. Esto permite una mayor libertad en la elección de la función a minimizar, y por tanto una mayor flexibilidad que permite muchas y variadas posibilidades en el campo del control predictivo.

El capítulo se ha centrado en la mejora de las prestaciones que se obtienen al poder utilizar modelos no lineales en los actuadores y en los procesos, e incluir además, restricciones en las variables de salida del proceso. Esto da lugar, a lo largo del capítulo, a tres partes diferenciadas:

1. No linealidades 'bruscas' en los actuadores tales como las saturaciones, zonas muertas, *backlash* e histéresis. Estas no linealidades son comunes en los actuadores y constituyen un problema en el control de procesos, por contra se pueden detectar e identificar, por lo que esta información se puede tener en cuenta en el controlador.

   Para ilustrar la utilización de estos modelos se ha considerado que dichas no linealidades aparecen en los actuadores, aunque se puede generalizar a cualquier tipo de variable interna del modelo.

   Dependiendo del tipo de no linealidad de la que se trate, aparecen distintas alternativas para conseguir introducir la información en el controlador: bien en el modelo del proceso o como restricciones del problema de optimización. En el capítulo se detallan las posibilidades para cada uno de los tipos de no linealidades.

   Se muestra que, salvo en el caso de las saturaciones, es más fácil introducir la información en el modelo de predicción que como restricciones o penalizando la función de coste. En el caso concreto de las saturaciones de las acciones de control es mucho más simple incorporar la información como restricciones del espacio de búsqueda del optimizador.

   Cuando se usan índices cuadráticos para que sea posible aplicar el algoritmo SQP, en todos los casos (salvo la saturación en la acción de control si el resto del modelo es

lineal) el control predictivo con optimización mediante Algoritmos Genéticos consigue buenos resultados, mostrándose superior al algoritmo de optimización SQP en cuanto a la calidad se refiere. Incluso, en algunos de estos ensayos el control predictivo con SQP tiene serias dificultades para conseguir que la respuesta siga la referencia.

Aprovechando las ventajas de flexibilidad por la utilización de Algoritmos Genéticos se han realizado ensayos con el índice modular en lugar del cuadrático. Se llega a la conclusión que el tipo de índice que se debe utilizar depende de las prestaciones que se quieran obtener, en concreto de los indicadores que se vayan a utilizar para evaluar la calidad del control, la ventaja de los Algoritmos es que no plantea, como lo hacen otros métodos, ninguna limitación en cuanto a las funciones de coste a minimizar ni del tipo de modelo que se emplea para diseñar el controlador incrementando así enormemente las posibilidades del MBPC.

2. En el capítulo se muestra además, como se pueden tener en cuenta restricciones en la variable de salida. El tratamiento que se realiza es similar a lo que se hace con las no linealidades 'bruscas' en los actuadores, en el caso de las restricciones físicas del proceso la forma más sencilla de tenerlas en cuenta es incluirlas en el modelo del proceso y en el caso de las restricciones de funcionamiento es más adecuado tenerlas en cuenta mediante funciones de penalización. La ventaja fundamental de la utilización de un Algoritmo Genético para el tratamiento de las restricciones de funcionamiento es que no es necesario el desarrollo de funciones de penalización especiales, ya que aunque el problema de optimización que resulte sea no convexo, el Algoritmo Genético presenta bastantes garantías para resolverlo.

3. La última parte del capítulo muestra la utilización de modelos no lineales en la metodología MBPC gracias a la inclusión de una técnica de optimización heurística como los Algoritmo Genéticos. Esta técnica aporta una mayor sencillez en la estructura del controlador y una mejor utilización del modelo no lineal frente a un controlador lineal o un controlador compuesto por un conjunto de controladores lineales. Además permite simultáneamente: la utilización de un modelo no lineal del proceso, la inclusión de no linealidades en los actuadores y restricciones de funcionamiento

En resumen, uniendo los beneficios del control predictivo (con nuevas posibilidades en los índices de coste) a los que aporta una técnica de optimización heurística como los Algoritmos Genéticos se puede conseguir un control adecuado para un amplio rango de problemas. La limitación de esta técnica vendrá impuesta por los tiempos de cálculo.

Capítulo 6

MBPC con optimización heurística en sistemas MIMO. Aplicación al control climático de un invernadero

Este capítulo muestra como se aplica el control predictivo con optimización heurística a un sistema con múltiples entrada y múltiples salidas (MIMO). Como en cualquier control predictivo, se debe establecer un modelo para realizar las predicciones, una función de coste y un optimizador. Igual que en el control predictivo SISO, la utilización de una técnica de optimización heurística permite un mayor grado de libertad a la hora de establecer el tipo de modelo y la función de coste. Para procesos con fuertes no linealidades estas pueden ser consideradas en el modelo sin necesidad de realizar ninguna simplificación lo que generalmente se traduce en un control de mayor calidad. El problema fundamental sigue siendo el coste computacional. En el caso MIMO se complica más el problema de optimización puesto que aparecen más variables, en cualquier caso si el periodo de muestreo lo permite se puede aplicar una técnica de optimización heurística. Se verá que el control predictivo para un proceso MIMO es una extensión del de un proceso SISO.

La última parte del capítulo corresponde a la aplicación de un control predictivo con técnica de optimización heurística a un problema MIMO no lineal. Se trata del control climático de un invernadero, el modelo presenta una complejidad tal que dificulta la solución mediante controladores obtenidos con modelos lineales o linealizados. La aplicación del control predictivo basado en modelos con optimización heurística está justificado, máxime por ser un proceso suficientemente lento lo que posibilita coste computacionales altos.

6.1 Aplicación a sistemas MIMO

Para aplicar el control predictivo a un sistema MIMO se puede utilizar la misma estructura que para un sistema SISO. El primer paso es definir un modelo para realizar las predicciones, para ello se propone utilizar la estructura que se muestra en la figura 6.1. Donde:

- $Y_u(t) = [y_{u1}(t), y_{u2}(t), \ldots, y_{ur}(t)]^T$, es un vector con variables de salida del modelo del proceso.
- $N(t) = [n_1(t), n_2(t), \ldots, n_r(t)]^T$, es un vector con las señales que provienen del modelo de perturbaciones. Este vector es de la misma dimensión que $Y_u(t)$, a cada $y_{ui}(t)$ le corresponde una $n_i(t)$.
- $Y(t) = [y_1(t), y_2(t), \ldots, y_r(t)]^T$, se obtiene de $Y(t) = Y_u(t) + N(t)$, es decir, $y_i(t) = y_{ui}(t) + n_i(t)$.
- $\xi(t) = [\xi_1(t), \xi_2(t), \ldots, \xi_r(t)]^T$, entradas de los modelos de perturbaciones que se consideran ruidos blancos.
- $U(t) = [u_1(t), u_2(t), \ldots, u_e(t)]^T$, vector de entradas del modelo del proceso.

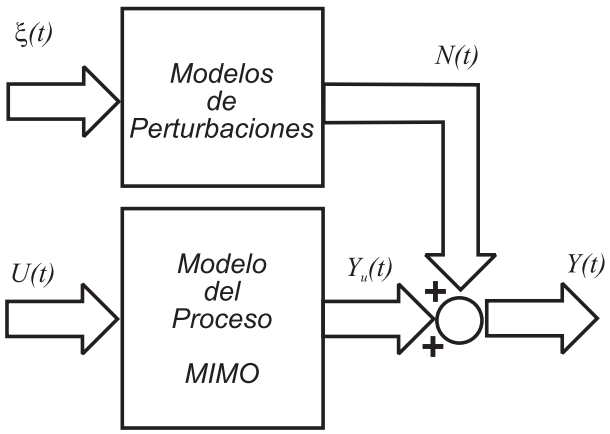


Figura 6.1: Estructura del modelo utilizado para un sistema MIMO.

El modelo del proceso puede ser de cualquier tipo: funciones de transferencia, espacio de estados, etc... basta con que sea capaz de predecir el valor de las variables de salida a partir de la información actual y pasada del proceso[1]. Las predicciones en el instante 't+j' con la información disponible en 't', se obtienen de:

$$Y(t+j|t) = Y_u(t+j|t) + N(t+j|t) \tag{6.1}$$

[1]En este trabajo se utiliza únicamente información entrada/salida del proceso.

Donde $Y_u(t+j|t)$ se obtiene del modelo del proceso y las acciones de control futuras, y $N(t+j|t)$ se obtiene de los modelos de perturbaciones.

Una alternativa para formar un modelo de perturbaciones es mantener una estructura similar a la que tiene el GPC, es decir, incorporar el factor Δ en el denominador para obtener un buen comportamiento en régimen permanente y un polinomio de diseño $T_i(z^{-1})$ en el numerador para conseguir buenas cualidades de robustez y atenuación de perturbaciones. Además, si se dispone de más información se puede incluir, de la misma forma que en el GPC añadiéndola como un polinomio al denominador. En resumen, el modelo de perturbaciones queda:

$$n_i(t) = \frac{T_i(z^{-1})}{\Delta A_i(z^{-1})}\xi_i(t) \tag{6.2}$$

Con este modelo de perturbaciones, la predicción para el instante '$t+j$' de una de las perturbaciones $n_i(t+j|t)$ se obtiene de la misma manera que para un sistema SISO:

$$n_i^f(t) = n_i(t)/T_i(z^{-1}) \tag{6.3}$$

$$\begin{aligned} n_i^f(t+j|t) &= (\Delta A_i(z^{-1}))' n_i^f(t+j|t) = -\hat{a}_{i1} n_i^f(t+j-1|t) - \\ &-\hat{a}_{i2} n_i^f(t+j-2|t) - \ldots - \hat{a}_{i,j-1} n_i^f(t+1|t) - \hat{a}_{ij} n_i^f(t) - \\ &-\hat{a}_{i,j+1} n_i^f(t-1) - \ldots - \hat{a}_{i,n_a+1} n_i^f(t+j-n_a-1) \end{aligned} \tag{6.4}$$

Donde:

$$\begin{aligned} \Delta A_i(z^{-1}) &= 1 + \hat{a}_{i1} z^{-1} + \hat{a}_{i2} z^{-2} + \cdots + \hat{a}_{i,n_a+1} z^{-(n_a+1)} \\ (\Delta A_i(z^{-1}))' &= -\hat{a}_{i1} z^{-1} - \hat{a}_{i2} z^{-2} - \cdots - \hat{a}_{i,n_a+1} z^{-(n_a+1)} \end{aligned}$$

A partir de $n_i^f(t+j|t)$ se obtiene la mejor predicción de la salida del modelo de perturbaciones $n_i(t+j|t)$:

$$n_i(t+j|t) = T_i(z^{-1}) n_i^f(t+j|t) \tag{6.5}$$

En cuanto al índice de coste que se propone:

$$J(U) = \sum_{i=1}^{r} \left(\sum_{j=N_{i1}}^{N_{i2}} \alpha_{ij} \left| y_i(t+j|t) - w_i(t+j) \right| \right) \tag{6.6}$$

Donde:

$$U = [u_1(t), \ldots, u_1(t+N_{1u}-1), \ldots, u_r(t), \ldots, u_r(t+N_{ru}-1)]^T$$

El índice no incluye ningún término referente a las acciones de control pero no existe ninguna dificultad matemática adicional si se añade uno similar al que aparecen en los GPCs. El problema es el ajuste de las ponderaciones de este término.

Se opta por operar con el valor absoluto del error predicho por tratarse de un magnitud menos distorsionada que la que corresponde a los índices cuadráticos.

El problema más importante que aparece es el ajuste de los coeficientes de ponderación (α_{ij}) puesto que en el mismo índice aparece distintos tipos de magnitudes. La opción más lógica pasa por tratar de normalizar todas las variables y sobre las variables normalizadas aplicar las ponderaciones necesarias. La normalización se puede realizar, por ejemplo, mediante:

$$\alpha_{ij} = \frac{1}{|w_i(t+j)|} \tag{6.7}$$

Se trata de relacionar el error predicho con el valor de la referencia (en principio se trata de un punto de funcionamiento) con lo cual se obtiene una evaluación del error respecto de la referencia establecida:

$$\frac{|y_i(t+j|t) - w_i(t+j)|}{|w_i(t+j)|}$$

Si además es necesaria una ponderación adicional (α'_{ij}) se puede añadir a la normalización:

$$\alpha_{ij} = \frac{\alpha'_{ij}}{|w_i(t+j)|} \tag{6.8}$$

Además de la normalización en magnitud es necesaria una normalización en el tiempo salvo que los tiempos de establecimiento de todas las variables que intervienen en el índice de coste sean del mismo orden de magnitud. Las variables con tiempos de establecimiento muy dispares pueden afectar de forma muy diferente la función de coste, por ejemplo, la variable con mayor tiempo de establecimiento es la que más afecta al índice de coste puesto que es la que presenta errores predichos mayores durante más tiempo en el horizonte de predicción. La solución en este caso es más compleja, como alternativa cabe la posibilidad de utilizar los factores α_{ij} para dar más importancia a unas variables u otras.

6.2 Aplicación al control climático del cultivo bajo invernadero

En está sección se detallan los pasos seguidos para la aplicación del control predictivo con optimización heurística al problema de control climático de un cultivo de rosa en invernadero.

El trabajo forma parte de un proyecto CICYT (TAP96-1090-C04-02) de título *Desarrollo de controladores predictivos basados en modelos optimizados mediante algoritmos genéticos para su aplicación en el control en tiempo real de proceso industriales no lineales.* Se trata de un proyecto coordinado en el que colaboran personal de:

- Instituto Valenciano de Investigaciones Agrarias (IVIA).
- Institut National de la Recherche Agronomique (INRA), Francia.
- Grupo de Control Predictivo y Optimización Heurística (CPOH) de la Universidad Politécnica de Valencia.

El proceso en cuestión justifica la utilización de un control predictivo basado en modelos con optimización heurística por varios motivos:

- Se trata de un proceso muy complejo, como se mostrará más adelante presenta no linealidades importantes.
- Existen muchas perturbaciones, varias de ellas medibles, e incertidumbres en el modelo.
- El control que actualmente está funcionando presenta muchas deficiencias.
- Se trata de un proceso que permite periodos de muestreo elevados.

Como motivación adicional se trata de un problema real, el invernadero que se utiliza para los ensayos se encuentra en las instalaciones del IVIA situadas en Moncada (Valencia).

Para abordar el problema, el primer paso es la obtención de un modelo del proceso que, como se detallará posteriormente, se obtiene a partir de principios básicos ([12], [14], [13] y [58]). Este modelo consiste en una representación en espacio de estado, aunque se debe recordar que en ningún caso se supone accesible el estado, el controlador se debe basar en un modelo que utilice exclusivamente datos de entrada salida, esta característica permitiría cambiar por otro tipo de modelo si fuese necesario. El siguiente paso consiste en describir la estructura que se elige en el control predictivo (índice de coste, ajuste de parámetros, algoritmo genético, etc.) y finalmente se muestran los resultados obtenidos comparados con un control con reguladores de tipo PID con prealimentaciones.

6.2.1 Modelo climático de un cultivo de rosas bajo invernadero.

El modelo que se va a obtener vendrán expresado mediante un conjunto de ecuaciones diferenciales que relacionan las variables presentes en el invernadero, se trata de un modelo en espacio de estados. El modelo básicamente se compone de tres ecuaciones diferenciales que corresponden a un balance de masas y dos balances energéticos:

1. Balance másico de vapor de agua en el volumen de aire del invernadero.
2. Balance energético del volumen de aire del invernadero.
3. Balance energético de la masa térmica que incorpora el efecto del suelo del invernadero como elemento almacenador de energía.

Se trata de un sistema multivariable MIMO. Las variables manipulables del proceso son tres:

1. Abertura de la ventana.
2. Nebulización.
3. Calefacción.

Las variables de salida para el control climático son dos:

1. Temperatura interior del invernadero
2. Humedad interior.

Además existen varias variables medibles consideradas como perturbaciones (puesto que no son manipulables). Estás variables se irán enumerando a lo largo del desarrollo del modelo.

Medidas de la humedad.

Dos de las variables que forman parte del proceso son las humedades interior y exterior, esta magnitud se puede expresar de distintas formas: como humedad absoluta, como humedad relativa (HR en %) o como déficit de saturación (D en unidades de presión). Dependiendo de su utilización se expresará de una forma u otra, en cualquier caso se puede pasar de una representación a otra aplicando las expresiones que se detallan en este apartado.

- Humedad absoluta corresponde a:

$$x = \frac{Kg_{H_2O}}{Kg_{aire}} \tag{6.9}$$

- Humedad relativa, se mide en tanto por cien y corresponde a:

$$HR = 100\frac{x}{x_{sat}} \tag{6.10}$$

donde x_{sat} es la humedad absoluta de saturación a la temperatura T que puede ser calculada mediante la expresión:

$$x_{sat} = 0.611\frac{p_{sat}(T)}{P} \tag{6.11}$$

$p_{sat}(T)$ y P son la presión de saturación a la temperatura T y presión absoluta respectivamente medidas en KPa. Si no se mide la presión se suele tomar, en el contexto de un invernadero, la presión de una atmósfera $P = 98.1KPa$. La presión de saturación a una temperatura determinada se calcula con:

$$p_{sat}(T) = 0.61064(1 + 1.4142136sin(5.81776\,10^{-3}T))^{8.827} \tag{6.12}$$

La temperatura se mide en ^oC.

- El Déficit de Saturación D (expresado en (KPa)) se obtiene mediante:

$$D = p_{sat}(T)\left[1 - \frac{HR}{100}\right] \tag{6.13}$$

Balance másico sobre el vapor de agua.

La primera ecuación diferencial del modelo climático se obtiene del balance másico donde intervienen varios flujos (todos se miden en Kg/s):

1. Un flujo de renovación debido a la abertura de la ventana: F_v.
2. La evapotranspiración del cultivo: E.
3. La nebulización: neb.
4. La condensación sobre las paredes del invernadero: C.

Resultando la siguiente ecuación diferencial:

$$\rho v_i \frac{dx_i}{dt} = F_v + E + neb + C \tag{6.14}$$

donde:

ρ	(Kg_{aire}/m^3)	Densidad del aire
x_i	(Kg_{H_2O}/Kg_{aire})	Humedad absoluta interior
v_i	(m^3)	Volumen del invernadero

En realidad esta ecuación sólo es válida mientras $x_i < x_{sat}$ (equivalente a $HR_i < 100\%$), el volumen de aire tiene una capacidad máxima de agua dependiendo de la temperatura a la que se encuentre. Este limitación se debe tener en cuenta en la ecuación diferencial. Los flujos que sean positivos (aportan agua al aire) sólo deben intervenir en la ecuación cuando se cumpla $HR_i < 100\%$. De los flujos que interviene en la ecuación $C \leq 0$, $E > 0$, $neb \geq 0$ y F_v puede tomar cualquier signo, aunque en el caso en que $x_i = x_{sat}$, $F_v \leq 0$. Por tanto la ecuación diferencial se debe modificar según:

$$\rho v_i \frac{dx_i}{dt} = F_v + C_{vapo}(E + neb) + C \tag{6.15}$$

Donde:

$$C_{vapo} = \begin{cases} 1 & x_i < x_{sat} \\ 0 & x_i = x_{sat} \end{cases} \tag{6.16}$$

Para completar la ecuación diferencial se muestran los cálculos de todos los flujos (excepto el de nebulización puesto que se trata de una variable manipulable que vendrá impuesta por el sistema de control).

Flujo de Renovación. El flujo de renovación se obtiene de:

$$F_v = \rho G(x_o - x_i) \tag{6.17}$$

donde:

ρ	(Kg_{aire}/m^3)	Densidad del aire
x_i	(Kg_{H_2O}/Kg_{aire})	Humedad absoluta interior
G	(m^3_{aire}/seg)	Caudal de renovación
x_o	(Kg_{H_2O}/Kg_{aire})	Humedad absoluta exterior

Las expresiones para el cálculo del caudal de aire de renovación G responden a diferentes aproximaciones, de las que podemos destacar, dos de ellas:

$$\left.\begin{array}{l} G(\alpha) = G/(AV) \\ G(\alpha) = a\alpha + G(0) \end{array}\right\} \longrightarrow G = AV(a\alpha + G(0)) \tag{6.18}$$

$$\left.\begin{array}{l} G(\alpha) = G/(AV) \\ G(\alpha) = a'sin(\alpha/2) + G'(0) \end{array}\right\} \longrightarrow G = AV(a'sin(\alpha/2) + G'(0)) \tag{6.19}$$

donde:

A	(m^2)	Área de viento = $L_1 L_2$
V	(m/s)	Velocidad del viento
α	^{o}	Ángulo de abertura de la ventana
a, a'		Constantes para el flujo de renovación
$G(0)$, $G'(0)$		para cada aproximación
G	(m^3_{aire}/seg)	Caudal de renovación

Ambas expresiones no son del todo válidas cuando la velocidad del viento es baja ($V < 1m/s$), en estos casos la principal fuerza que influye en el flujo es el incremento de temperatura entre el interior y el exterior. Por simplicidad, se adopta en este trabajo la primera expresión.

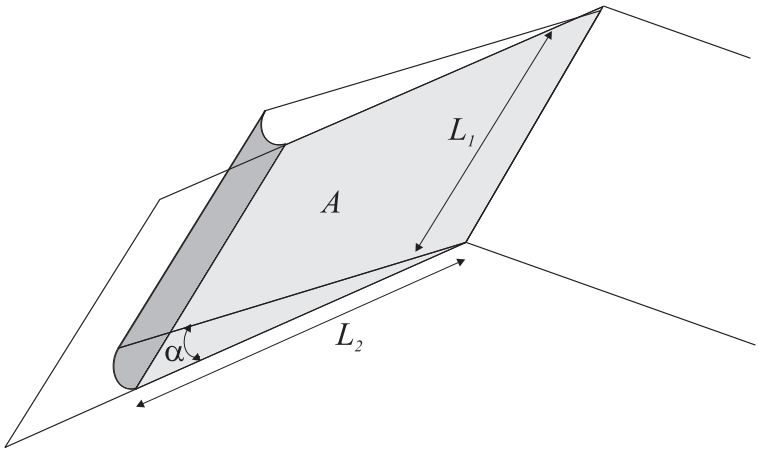


Figura 6.2: Esquema de la ventana de un invernadero.

Evapotranspiración. La evapotranspiración se formula tomando como base la ecuación de Penman-Monteith, según la descripción hecha en [58]:

$$E = \frac{A_i(\Delta Rn + 2L\rho\, c_p\, D_i\, gwb)}{\left(\Delta + \gamma\left(1 + \frac{gwb}{gws}\right)\right)\lambda} \tag{6.20}$$

Donde:

E	(Kg/seg)	Evapotranspiración
A_i	(m^2)	Área del invernadero
Δ	$(KPa/^oC)$	Pendiente de la curva de saturación
Rn	$(J/(m^2 seg))$	Radiación neta
L	$(m^2_{hojas}/m^2_{suelo})$	Índice foliar
ρ	(Kg/m^3)	Densidad del aire
c_p	$(J/(Kg^oK))$	Calor específico del aire
gwb	(m/s)	Conductancia capa límite
D_i	(KPa)	Déficit de saturación
γ	$(KPa/^oC)$	Constante psicrométrica
gws	(m/s)	Conductancia estomática
λ	(J/Kg)	Calor latente de vaporización

Cálculos auxiliares necesarios para la obtención de la evapotranspiración:

1. Calor latente de vaporización

$$\lambda = (3.1468 - 0.002365(T + 273.16))10^6 \tag{6.21}$$

En nuestro caso T será la temperatura interior del invernadero en ^oC.

2. Pendiente de la curva de saturación

$$\Delta = p_{sat}(T + 0.5) - p_{sat}(T - 0.5) \tag{6.22}$$

3. Radiación en el invernadero

$$Ss = \tau So \tag{6.23}$$

donde So $(J/(m^2\, seg))$ es la radiación exterior y τ es el coeficiente de transmisión de calor de las paredes del invernadero.

4. Radiación neta absorbida por las plantas. Cálculo simplificado.

$$Rn = (1 - \exp(-kL))Ss \tag{6.24}$$

donde k representa al coeficiente de extinción de la radiación solar.

5. Cálculo de la conductancia estomática

$$gws = gws_{min} + (gws_{max} - gws_{min}) \left[1 - exp\left(-\frac{Ss}{160}\right)\right] g_D \tag{6.25}$$

$$g_D = \begin{cases} \frac{0.39}{0.029+D_i} & D_i \geq 0.361 \\ 1 & D_i < 0.361 \end{cases}$$

Donde gws_{min} y gws_{max} son las conductancias estomáticas mínima y máxima.

Condensación. Este factor suele modelarse según:

$$C = \begin{cases} g_{con}\rho A_i(x_i - x_p) & x_p > x_i \\ 0 & x_p \leq x_i \end{cases} \tag{6.26}$$

donde, además de las magnitudes y parámetros ya conocidos resulta:

C	$(Kg/seg.)$	Caudal másico de condensación
g_{con}	$(m/seg.)$	Coeficiente de transferencia aire-pared
x_p	(Kg_{H_2O}/Kg_{aire})	Humedad absoluta de saturación a la temperatura de la pared.

Para calcular x_p se utiliza la expresión 6.11 y por tanto es necesaria la temperatura de la pared del invernadero (T_p). Si no se dispone de esta medida se puede estimar mediante la expresión empírica debida a Kittas:

$$T_p = T_i + \frac{(T_i - T_0) + 0.021S_0 - 4.46}{1.32 + 0.0036V^{0.8}} \tag{6.27}$$

Es importante hacer notar que la condensación sólo aparece en el caso de que $x_p < x_i$, pues es el caso en que la humedad del ambiente supera a la correspondiente a la temperatura de la pared. En caso contrario, la condensación (C) debe ser igual a 0.

Balance de Energía en el volumen del invernadero

En el balance energético en el volumen del invernadero aparecen los siguientes flujos (medidos en $Watios$ o J/seg):

1. Radiación solar: $Q_s = A_i \tau So$.

2. Pérdidas por convección y conducción: $Q_{cc} = A_i(Ac + BcV)(Ti - To)$.
3. Pérdidas debidas a la evapotranspiración del cultivo: $Q_e = \lambda E$.
4. Pérdidas debidas al flujo de ventilación provocado por la abertura de la ventana: $Q_v = \rho c_p G(Ti - To)$.
5. Pérdidas por le flujo de nebulización: $Q_n = \lambda neb$.
6. Calefacción: W
7. Intercambio con la masa térmica: $Q_m = A_i h_m(T_m - T_i)$.

Con lo que se obtiene la siguiente ecuación diferencial:

$$v_i \rho c_p \frac{dT_i}{dt} = Q_s - Q_{cc} + Q_m - Q_e - Q_n - Q_v + W \tag{6.28}$$

donde:

v_i	(m^3)	Volumen del invernadero
ρ	(Kg/m^3)	Densidad del aire
T_i	(^oC)	Temperatura interior del invernadero
A_i	(m^2)	Superficie del invernadero
τ		Coeficiente de transmisión del invernadero
So	$(J/(m^2 seg))$	Radiación exterior
Ac		Coeficiente de pérdidas energéticas para invernadero con cobertura simple. (Conducción)
Bc		Coeficiente de pérdidas energéticas para invernadero con cobertura simple. (Convección)
V	(m/s)	Velocidad del viento en el exterior del invernadero
T_o	(^oC)	Temperatura exterior del invernadero
E	(Kg/seg)	Evapotranspiración
λ	(J/Kg)	Calor latente de vaporización
T_m	(^oC)	Temperatura de la masa térmica
h_m	$(W/m^2 seg)$	Conductividad térmica de la masa térmica

Este balance es adecuado siempre que $x_i < x_{sat}$, en el caso en que el volumen de aire este saturado de agua, los términos Q_e y Q_n no deben aparecer puesto que no se produce la evaporación. La ecuación diferencial se debe modificar:

$$v_i \rho c_p \frac{dT_i}{dt} = Q_s - Q_{cc} + Q_m - C_{vapo}(Q_e + Q_n) - Q_v + W \tag{6.29}$$

Donde C_{vapo} corresponde a la expresión 6.16.

Balance de Energía de la Masa Térmica.

El Balance de Energía deberá incorporar tres factores:

1. El calor sensible almacenado durante el día por la masa térmica:

$$Q_{sm} = \alpha_m \tau S_0$$

2. El flujo de calor (positivo o negativo) procedente del invernadero.

$$Q_{mm} = h_m(T_m - T_i)$$

3. El flujo de calor perdido por el fondo del suelo.

$$Q_{fm} = k_a \left(\frac{T_m - T_{ref}}{z_{ref}} \right)$$

Así, la ecuación general de Balance de Energía a nivel de la masa térmica es:

$$C_m \frac{dT_m}{dt} = Q_{sm} - Q_{mm} - Q_{fm} \tag{6.30}$$

donde, además de las magnitudes ya definidas con anterioridad tenemos:

C_m	$(J/m^2\ ^oK)$	Capacidad Calorífica de la Masa Térmica
α_m		Factor de calor absorbido por la Masa Térmica
k_a	(W/m^oK)	Conductividad Térmica media del suelo
T_{ref}	(^oC)	Temperatura a la profundidad z_{ref}
z_{ref}	(m)	Profundidad

Diagrama del modelo.

Resumiendo, el proceso se puede modelar mediante tres ecuaciones de estado y dos ecuaciones de salida. La ecuaciones de estado corresponden a las tres ecuaciones diferenciales descritas:

$$\begin{aligned}
\rho v_i \frac{dx_i}{dt} &= F_v + C_{vapo}(E + neb) + C \\
v_i \rho c_p \frac{dT_i}{dt} &= Q_s - Q_{cc} + Q_m - C_{vapo}(Q_e + Q_n) - Q_v + W \\
C_m \frac{dT_m}{dt} &= Q_{sm} - Q_{mm} - Q_{fm}
\end{aligned}$$

Las ecuaciones de salida serían:

$$\begin{aligned} D_i &= p_{sat}(T_i)\left(1 - \frac{x_i}{x_{sat}}\right) \\ T_i &= T_i \end{aligned}$$

Esquemáticamente se puede ver en la figura 6.3. Aparecen cuatro variables medibles pero no manipulables que se consideran como perturbaciones:

1. Radiación solar: S_o.
2. Velocidad del viento en el exterior del invernadero: V.
3. Temperatura en el exterior del invernadero: T_o.
4. Humedad en el exterior del invernadero, esta variable se mide en humedad relativa HR_o, y para ser utilizada en las ecuaciones diferenciales se transforma a humedad absoluta x_o.

Las variables de entrada manipulables corresponden a:

1. Abertura de la ventana: α.
2. Nebulización: neb.
3. Calefacción: W.

Las variables de salida para el modelo climático corresponden a:

1. Temperatura interior del invernadero: T_i.
2. Humedad exterior que se puede medir con la humedad absoluta x_i, humedad relativa HR_i o déficit de saturación D_i. Se utiliza el déficit de saturación a petición de los usuarios finales.

Para su utilización en el control predictivo el modelo debe ser discretizado para ello existen distintas alternativas con sus ventajas e inconvenientes (Euler, Runge-Kutta,...).

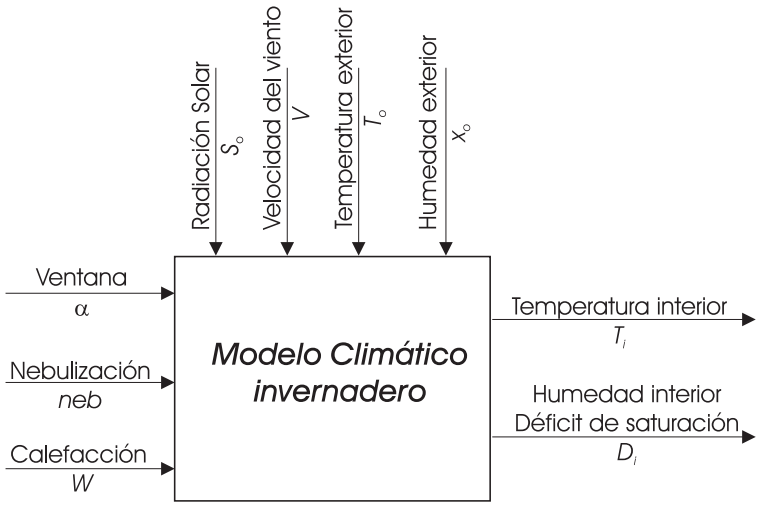


Figura 6.3: Modelo climático del invernadero.

Valores de los Parámetros

Con objeto de simular el comportamiento de un invernadero ejemplo de rosas, se establecen las magnitudes de los parámetros correspondientes. El invernadero que se trata de controlar está situado en la localidad de Moncada en el IVIA (Instituto Valenciano de Investigaciones Agrarias). Algunos de los valores corresponden a constantes físicas conocidas, otras se han podido medir en el propio invernadero y existe unos parámetros que se han tomado de la bibliografía especializada. Estos últimos valores deberían ser actualizados para el invernadero en cuestión. Actualmente se está llevando a cabo este trabajo de identificación de parámetros.

Parámetros Generales

Variable	Valor	Unidades	Descripción
γ	0.066	$KPa/^oC$	Cte. Psicométrica
		$(1KPa = 10mbar)$	
ρ	1.25	Kg/m^3	Densidad del aire
c_p	1003	J/Kg^oK	Calor específico del aire
P	98.1	KPa	Presión atmosférica
		$(1at = 98.1KPa)$	

Parámetros del Cultivo de Rosas.

Variable	Valor	Unidades	Descripción
gws_{max}	0.0155	m/s	Conductancia estomática máxima
gws_{min}	0.0005	m/s	Conductancia estomática mínima
L	2	m^2_{hojas}/m^2_{suelo}	Índice foliar
k	0.64		Coeficiente de extinción de la radiación solar
gwb	0.004	m/s	Conductancia capa límite

Parámetros del Invernadero.

Variable	Valor	Unidades	Descripción
τ	0.6		Coeficiente de transmisión del invernadero
A	51	m^2	Area de viento
α_{max}	15	^{o}	Ángulo máximo de la ventana
a	0.002		Constante para el flujo de renovación
$G(0)$	0.001		Constante para el flujo de renovación
a'	0.202		Constante para el flujo de renovación (segundo modelo)
$G'(0)$	0.0018		Constante para el flujo de renovación (segundo modelo)
A_i	96	m^2	Superficie del invernadero
v_i	358	m^3	Volumen del invernadero
Ac, Bc	6, 0.5		Coeficientes de perdidas energéticas para invernadero con cobertura simple.
C_m	$2 \cdot 10^5$	$J/m^2\ ^oK$	Capacidad Calorífica de la Masa Térmica
α_m	0.1		Factor de calor absorbido por la Masa Térmica
k_a	1.4	$W/m\ ^oK$	Conductividad Térmica media del suelo
T_{ref}	18	^oC	Temperatura a la profundidad z_{ref}
z_{ref}	1	m	Profundidad
g_{con}	$5 \cdot 10^{-3}$	m/seg	Coeficiente de transferencia aire-pared

6.2.2 Validación

Con los valores de las tablas se realizan un ensayo para comparar el modelo obtenido y el comportamiento real del invernaderos. La figura 6.4 muestra los valores de las variables exteriores (radiación, velocidad del viento, temperatura y humedad). La figura 6.5 muestra los valores de temperatura y humedad interiores obtenidos con el modelo (en

color verde) para las acciones de control indicadas. En esta misma figura se muestran los valores reales medidos en el invernadero (color azul) y los valores exteriores (en rojo).

Se puede observar que el modelo no se ajusta exactamente al comportamiento real, pero se puede decir que la estructura del modelo parece adecuada puesto que las formas de las evoluciones del modelo y los datos experimentales, tanto de temperatura como de humedad, son similares[2].

Por tanto este modelo puede servir para ilustrar el proceso de diseño de un control predictivo MIMO no lineal con un optimizador heurístico, por una parte el modelo no discrepa demasiado del comportamiento del invernadero real, y por otra, la complejidad del mismo puede justificar la utilización del control predictivo con optimización heurística.

[2]Actualmente se está llevando a cabo la identificación de los parámetros del modelo, de momento se han utilizado para las simulaciones valores de los parámetros que se encuentran en la bibliografía específica del tema. La estructura del modelo se muestra adecuada y ha sido verificada por los ingenieros agrónomos del INRA que participan en el proyecto.

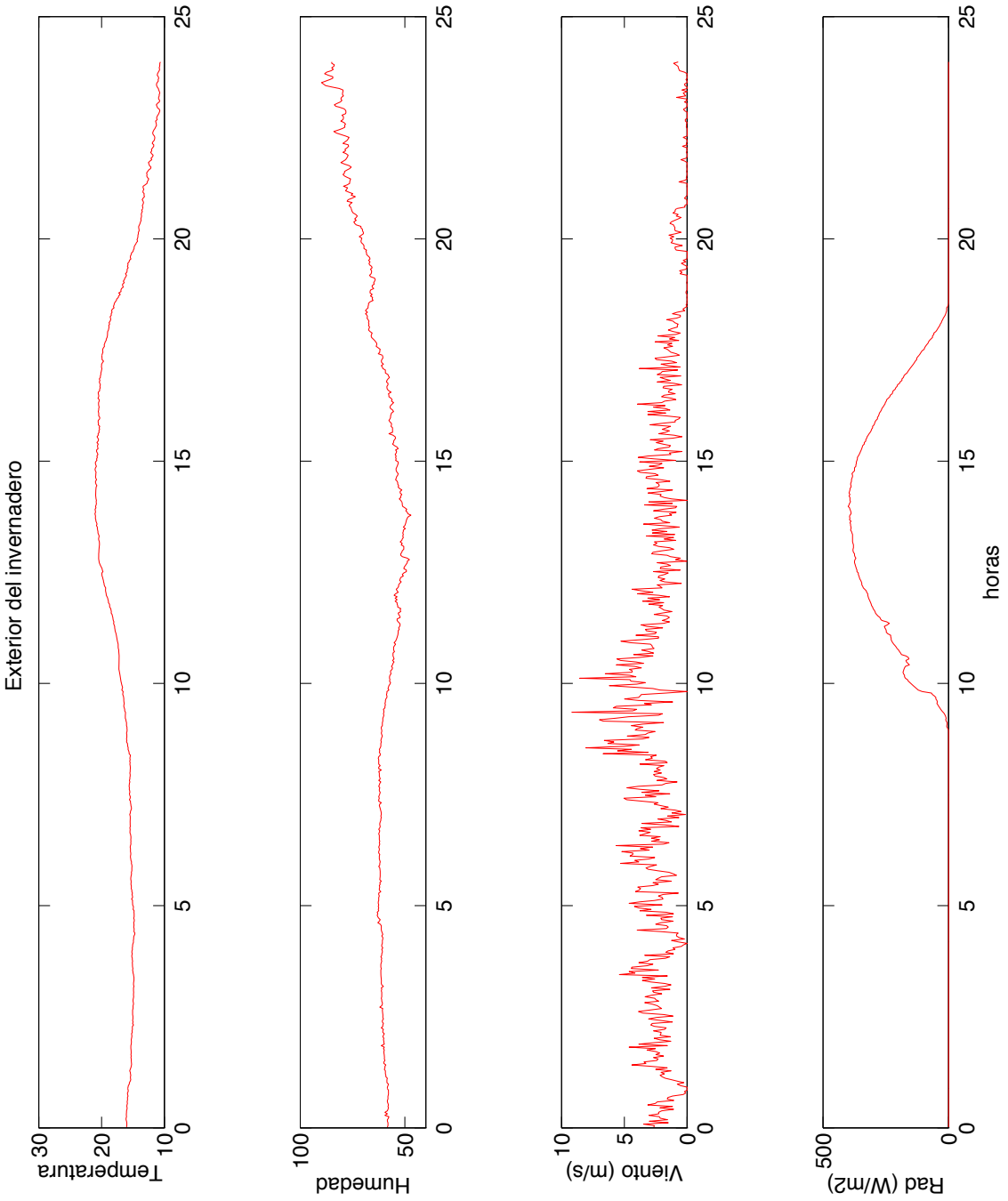


Figura 6.4: Valores exterior medidos para en un día de otoño/invierno (28 de noviembre de 1998). Temperatura (^oC), humedad relativa (%), velocidad del viento (m/s) y radiación solar (W/m^2).

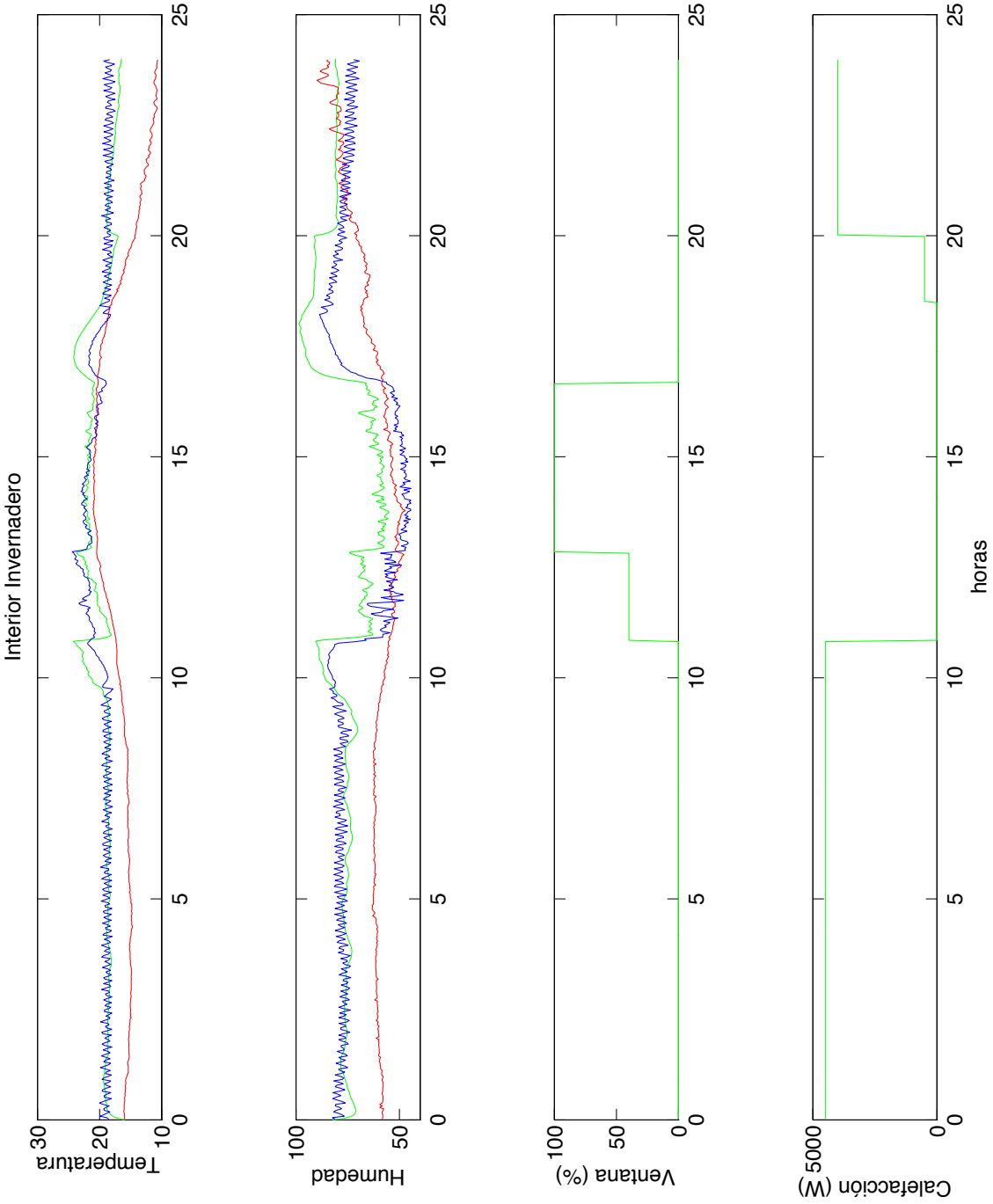


Figura 6.5: Valores de la temperatura y humedad obtenidas del modelo (en verde), valores medidos en el invernadero (en azul) y valores exteriores (en rojo) para las acciones de control de abertura de ventana y calefacción.

6.2.3 Control mediante reguladores PID con prealimentación

En la actualidad el invernadero está controlado de forma manual aunque con la ayuda de reguladores Todo/Nada que vienen con la instalación inicial. Dependiendo de las condiciones climáticas y de la hora, el operador decide que reguladores conectar y que valores se asignan a las referencias. Por ejemplo, si se trata de un día de invierno, el operador conecta el control de temperatura con la calefacción (regulador Todo/Nada) seleccionando la referencia. Utiliza entonces la abertura de la ventana para tratar de mantener la humedad en un rango razonable, esta parte del control es por tanto manual, el operador deja la posición de ventana en un valor que cree conveniente según su experiencia. El tipo de control cambia en verano, el control de temperatura se realiza con un control Todo/Nada de la ventana y el de humedad con otro control Todo/Nada de la nebulización, siempre con intervenciones manuales en ambos casos si el operador considera que el control no es adecuado.

Para poder decidir si es adecuado instalar un control predictivo, se debía primero intentar el control integral del invernadero mediante una estructura más o menos simple pero que ha demostrado su validez en numerosos procesos industriales, son los bucles de control con PID y prealimentación para desacoplar procesos multivariables. La comparación del control predictivo con el control manual realizado hasta el momento no se considera suficientemente objetiva para establecer la necesidad de implantar el control predictivo, por su inexactitud y falta de criterios de diseño.

Este apartado plantea el cálculo y ajuste de la estructura del control PID con prealimentación. El primer paso para un ajuste de los reguladores es disponer de unos modelos lineales. Es complicado obtener unos modelos lineales a partir de ensayos sobre el proceso. El problema básico reside en la imposibilidad de mantener el sistema en un punto de funcionamiento estable, las perturbaciones no son manipulables y varían continuamente. Se ha optado por obtener el modelo lineal por linealización del modelo no lineal calculado.

El punto de funcionamiento elegido corresponde a los valores siguientes:

Punto de funcionamiento						
α	S_o	V	T_o	HR_o	neb	cal
50%	$300W/m^2$	$2.5m/s$	21^oC	55%	0	0
x_i	T_i	T_m	D_i			
0.0107	22.39^oC	24.4^oC	$0.9896KPa$			

A partir de este punto de funcionamiento se aplican variaciones en forma de escalón a cada una de las variables de entrada hasta obtener la matriz de transferencia siguiente (las constantes de tiempo están expresadas en segundos).

$$\begin{pmatrix} D_i \\ T_i \end{pmatrix} = \begin{pmatrix} G_{11}(s) & G_{12}(s) & G_{13}(s) \\ G_{21}(s) & G_{22}(s) & G_{23}(s) \end{pmatrix} \begin{pmatrix} \alpha \\ neb \\ cal \end{pmatrix}$$

$$\begin{pmatrix} D_i \\ T_i \end{pmatrix} = \begin{pmatrix} \frac{0.002272}{170s+1} & \frac{-125.282}{100s+1} & \frac{0.0000352}{90s+1} \\ \frac{-0.0186}{130s+1} & \frac{-505.6}{80s+1} & \frac{0.000235}{100s+1} \end{pmatrix} \begin{pmatrix} \alpha \\ neb \\ cal \end{pmatrix}$$

La matriz de transferencia se ajusta bien al comportamiento del sistema no lineal al menos cerca del punto de funcionamiento. En la figura 6.6 se puede ver una comparación entre el modelo no lineal y el linealizado para tres casos diferentes:

1. Ventana: se aplica un escalón en la abertura de la ventana de un 10% respecto del punto de funcionamiento, las variables de nebulización y calefacción se mantienen en su punto de funcionamiento.

2. Nebulización: se aplica un escalón en la nebulización de 0.005 Kg/s respecto del punto de funcionamiento, las variables de la ventana y calefacción se mantienen en su punto de funcionamiento.

3. Calefacción: se aplica un escalón en la calefacción de 500W respecto del punto de funcionamiento, las variables de la ventana y nebulización se mantienen en su punto de funcionamiento.

El control se va a realizar sin nebulización puesto que corresponde a un día de otoño/invierno[3] con las variables de perturbación que se muestran en la figura 6.4. Por tanto el modelo se puede simplificar:

$$\begin{pmatrix} D_i \\ T_i \end{pmatrix} = \begin{pmatrix} G_{11}(s) & G_{13}(s) \\ G_{21}(s) & G_{23}(s) \end{pmatrix} \begin{pmatrix} \alpha \\ cal \end{pmatrix}$$

Los bucles simples que se plantean son (ver figura 6.7):

1. Uso de la calefacción para el control de la temperatura.

2. Uso de la ventana para el control de la humedad.

Se ajustan un regulador de tipo PI con *antiwindup* para cada uno de los bucles:

$$\begin{aligned} PI_1(s) &= 750\left(1+\frac{1}{170s}\right) \\ PI_2(s) &= 4255\left(1+\frac{1}{100s}\right) \end{aligned}$$

[3]Para condiciones de primavera/verano, el proceso de diseño es idéntico pero se conecta la nebulización y se apaga la calefacción (motivos económicos así lo requieren).

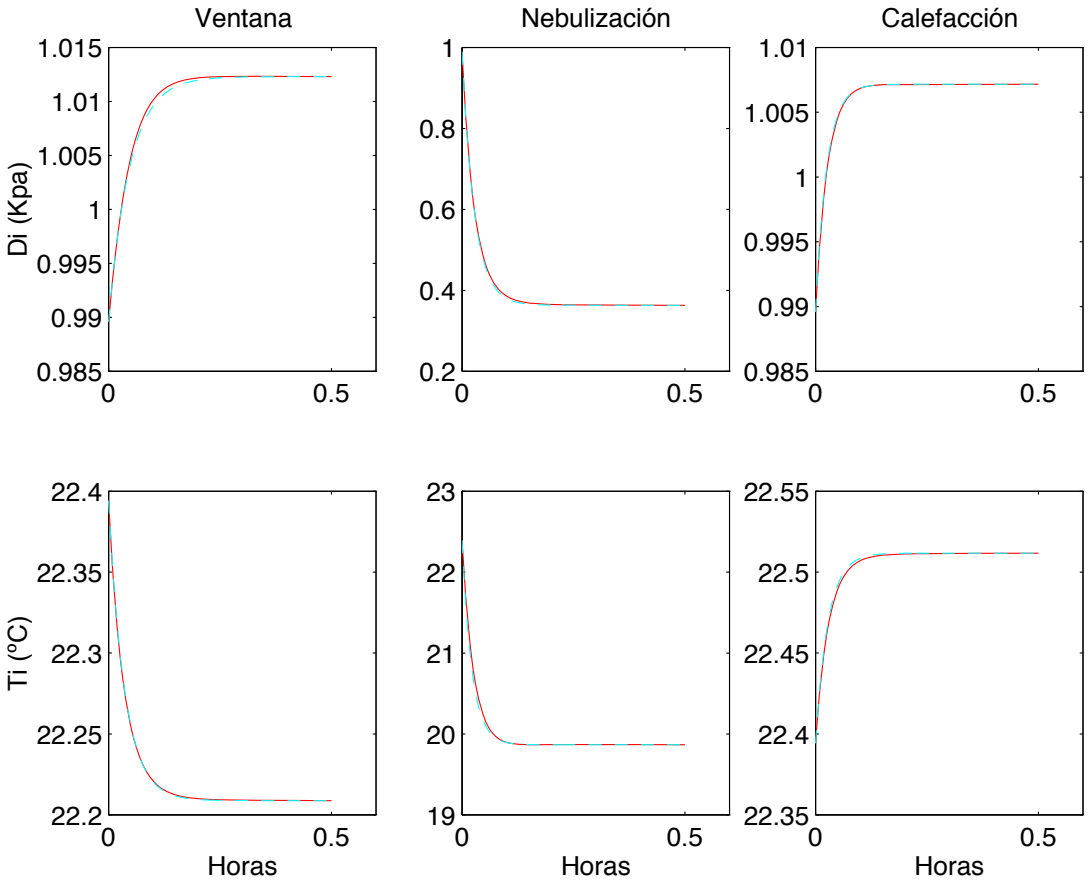


Figura 6.6: Matriz de transferencia. Comparación entre modelo no lineal (en rojo) y modelo lineal (en azul).

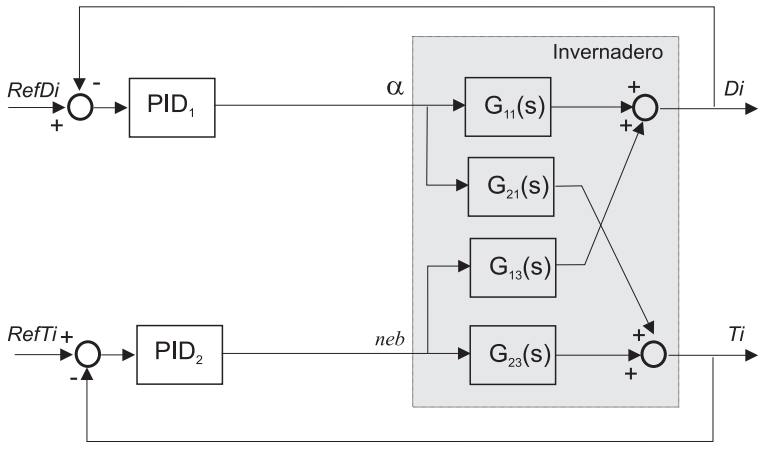


Figura 6.7: Control de temperatura y humedad con dos bucles simples (modelo lineal).

Para desacoplar los dos bucles se añaden prealimentaciones (figura 6.8):

$$\begin{aligned} G_{ff1}(s) &= -0.0155 \\ G_{ff2}(s) &= 79.15 \end{aligned}$$

Con estos controladores se obtienen los resultados que se muestran en la figura 6.9,

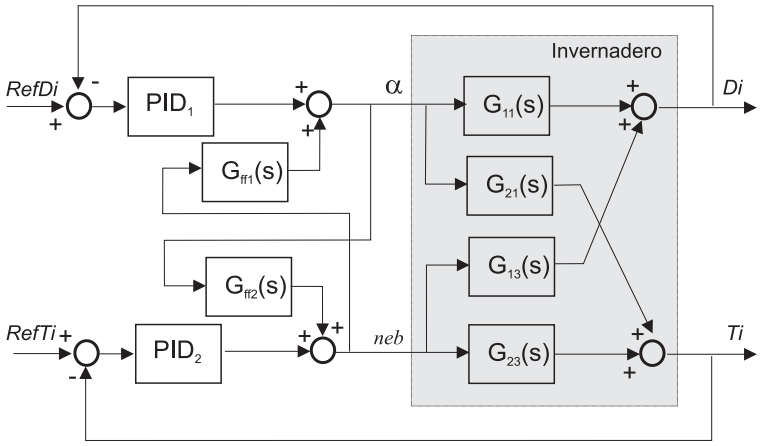


Figura 6.8: Control de temperatura y humedad con dos bucles simples y prealimentación (modelo lineal).

que corresponde al control con los modelos lineales. El control obtenido es aceptable en ambos casos. El siguiente paso es aplicar estos controles al modelo no lineal. Las figuras 6.10 y 6.11 corresponden a los resultados obtenidos para el control en bucle simple y con prealimentación respectivamente. Los datos exteriores corresponden a los de la figura 6.4. Comparado con el control simple, el control con prealimentación tiene más dificultades para mantener la temperatura en un valor aceptable, en ambos casos se mantiene la humedad en un valor cercano a la referencia aunque con un pequeño error.

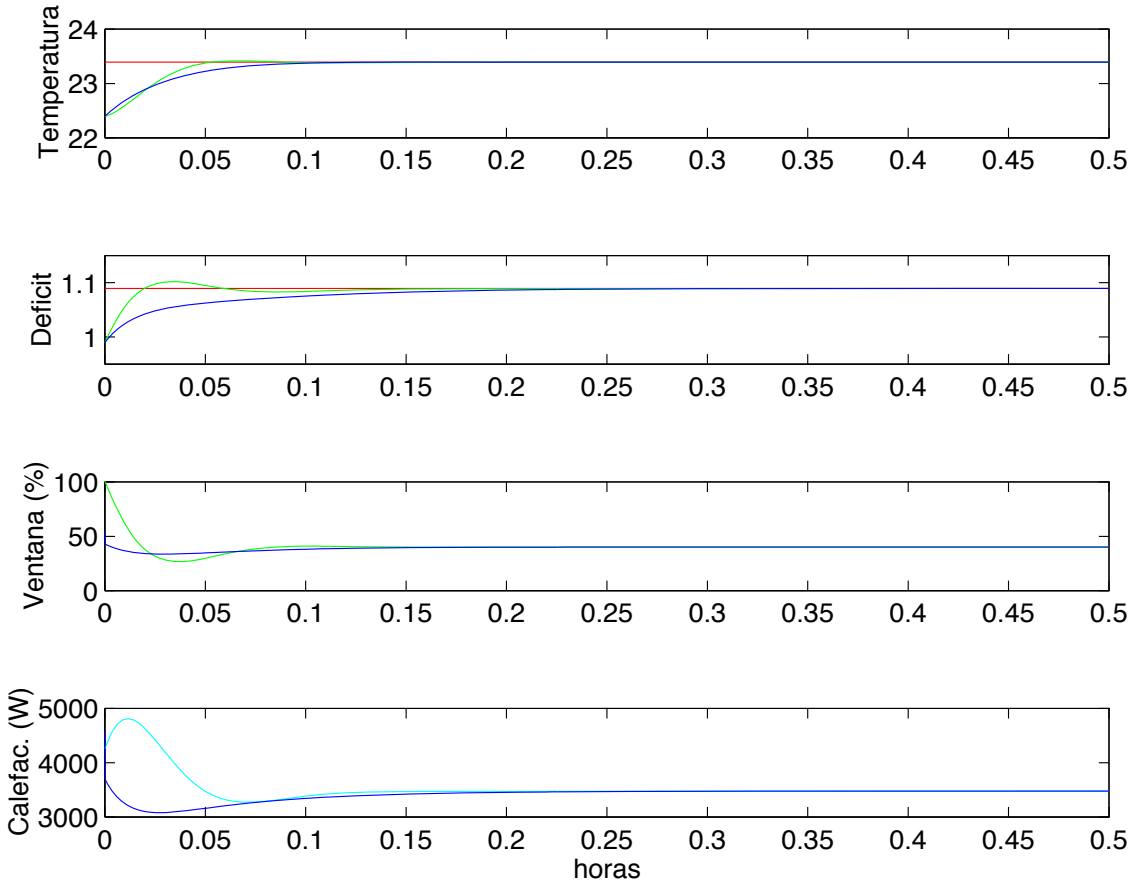


Figura 6.9: Control de temperatura y humedad con modelos lineales. Bucles simples en azul y bucles con prealimentación en verde. Referencias en rojo.

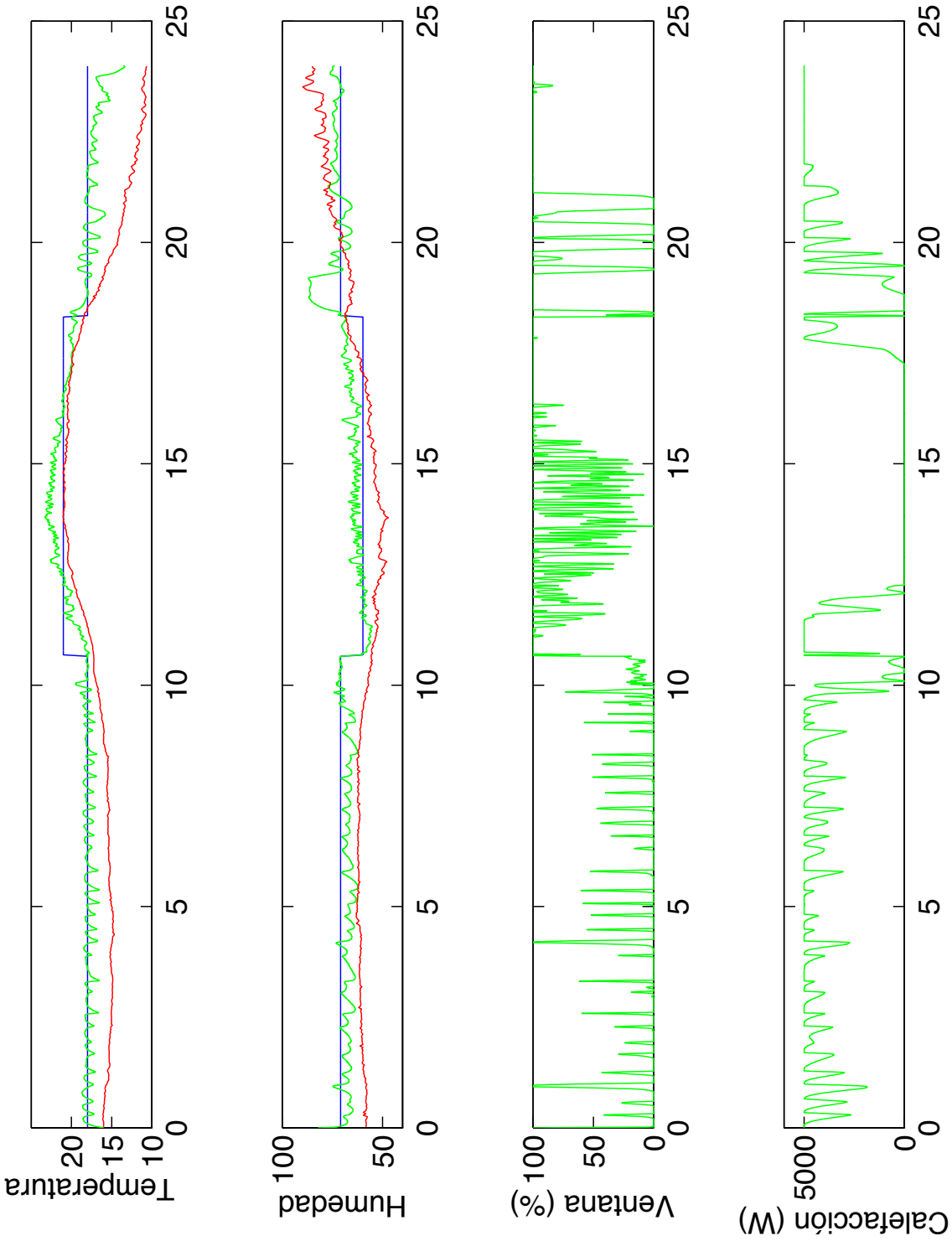


Figura 6.10: Control de temperatura y humedad con bucle simple (modelo no lineal). Variables del invernadero en verde, variables exteriores en rojo y referencias en azul.

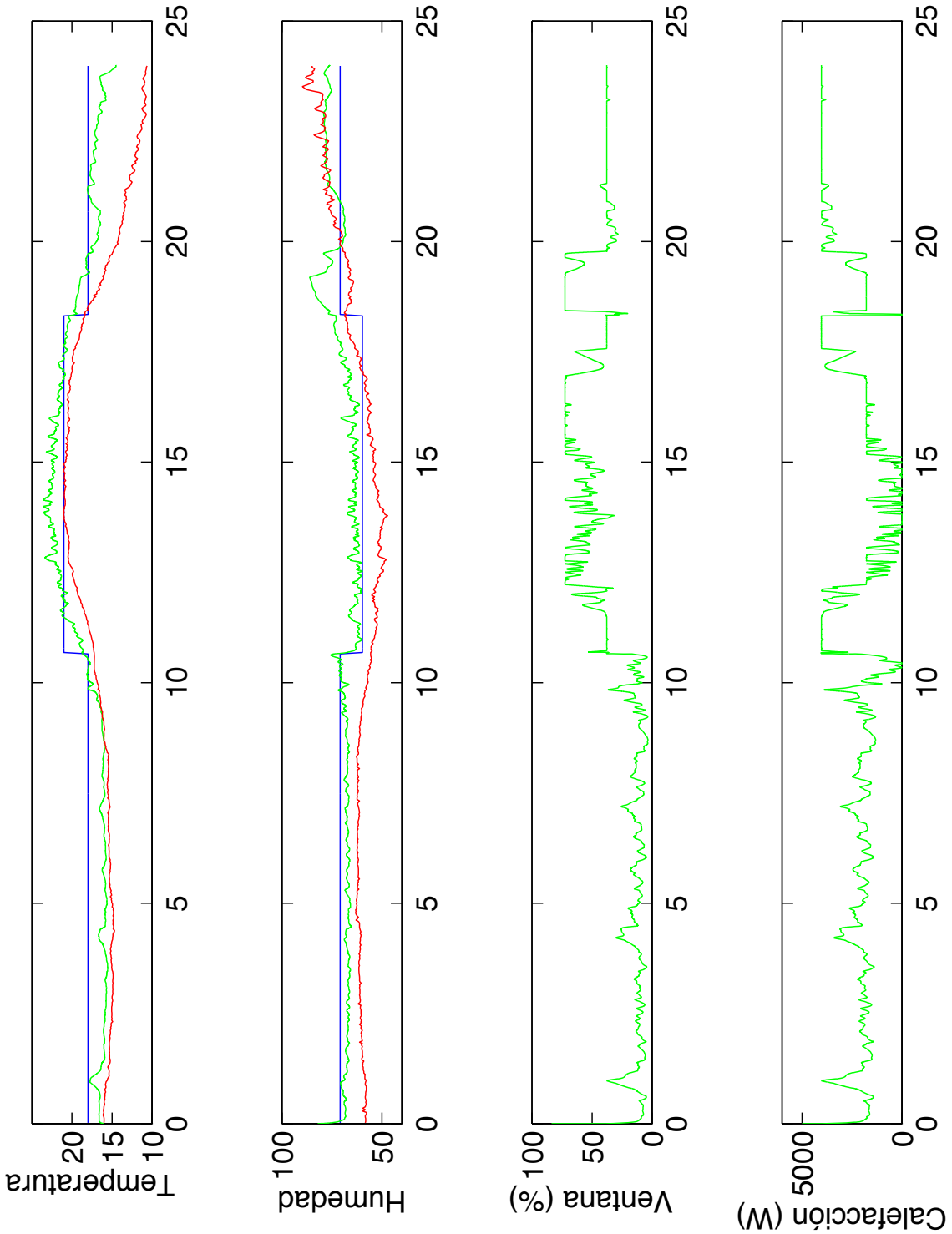


Figura 6.11: Control de temperatura y humedad con bucle simple y prealimentación (modelo no lineal). Variables del invernadero en verde, variables exteriores en rojo y referencias en azul.

6.2.4 Control predictivo con optimización heurística

El modelo utilizado para realizar las predicciones incluye un modelo discretizado no lineal con un modelo de perturbaciones. En este proceso aparecen perturbaciones medibles como son:

1. Radiación solar: S_o.
2. Velocidad del viento en el exterior del invernadero: V.
3. Temperatura en el exterior del invernadero: T_o.
4. Humedad en el exterior del invernadero, este variable se mide en humedad relativa HR_o, y para ser utilizada en las ecuaciones diferenciales se transforma a humedad absoluta x_o.

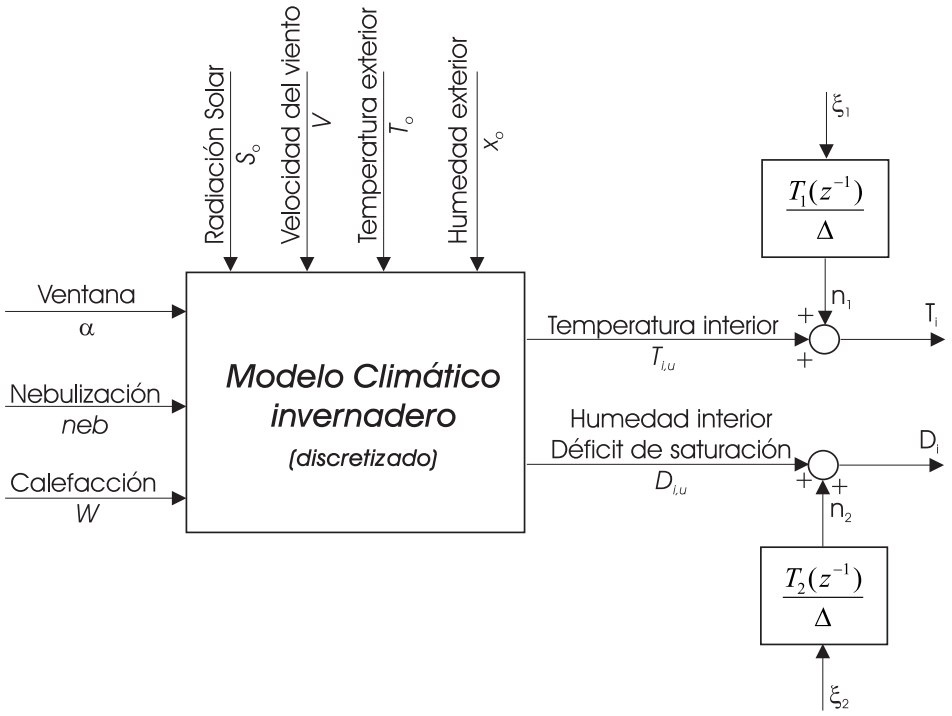


Figura 6.12: Modelo utilizado para el control predictivo.

La relación con las variables de salida es no lineal y ya se tiene en cuenta en las ecuaciones del modelo. Queda por resolver la estimación que se realice de dichas variables a lo largo del horizonte de predicción. Para este ejemplo aproximación se van a tomar

como constantes a lo largo del horizonte de predicción y con el valor que tienen en el instante 't'.

Se añade al modelo unos términos para compensar la dinámica no modelada y posibles problemas con los ruidos de medida. A cada una de las variables de salida se le suma un término del tipo:

$$n_i(t) = \frac{T_i(z^{-1})}{\Delta}\xi_i(t)$$

Las variables $\xi_i(t)$ se consideran ruidos blancos. El factor $1/\Delta$ permite ajustar el régimen permanente incluso con errores de modelado, y el polinomio $T(z^{-1})$ ofrece la posibilidad de mejorar el comportamiento ante ruidos e incrementa la robustez. Para este ejemplo se utiliza $T(z^{-1}) = 1$. El modelo para realizar las predicciones queda como se muestra en la figura 6.12.

La función de coste utilizada es:

$$J = \sum_{j=1}^{N_2} \alpha_{j1} \left| D_i(t+j|t) - RefD_i(t) \right| + \sum_{j=1}^{N_2} \alpha_{j2} \left| T_i(t+j|t) - RefT_i(t) \right|$$

Donde:

$$\begin{aligned} \alpha_{j1} &= \frac{1}{|RefD_i(t)|} \\ \alpha_{j2} &= \frac{1}{|RefT_i(t)|} \end{aligned}$$

Como se ve, en este ejemplo, el índice no tiene en cuenta las referencias futuras, se suponen constantes en todo el horizonte de predicción con el valor en el instante 't'.

Los parámetros del horizonte de predicción y de control se ajustan a: $N_2 = 5$ y $N_u = 1$ respectivamente. El ajuste de $N_2 = 5$ permite considerar todas las perturbaciones constantes a lo largo del horizonte de predicción sin cometer demasiado error. Se puede tomar N_2 mayor pero para conseguir una mejor calidad en las predicciones se deberían modificar las predicciones de las perturbaciones. Con $N_2 = 5$ el horizonte de predicción es de 10 minutos suficiente para recoger buena parte del transitorio de las variables de salida sin que se modifiquen demasiado las perturbaciones.

También se puede tomar un horizonte de control mayor, por ejemplo, $N_u = 2$, en teoría se consigue con ello un control más agresivo, se debería poder conseguir que las variables de salida alcancen las referencias más rapidamente que con $N_u = 1$. En la práctica se observa que los accionadores están trabajando cerca de sus límites de saturación, por tanto, no es de esperar una mejora espectacular en el control si se aumenta el horizonte de control, provocando en cambio un aumento de la complejidad del problema de optimización. Por

tanto se opta por $N_u = 1$ que ya representa un problema suficientemente complejo que justifica la utilización de un control predictivo con optimización heurística.

El optimizador que se utiliza es el Algoritmo Genético con codificación real con las siguientes características:

- Operación de *Ranking* lineal.
- Operador de selección: *Stochastic Universal Sampling.*
- Operador de cruce: Cruce por recombinación lineal con probabilidad de cruce $P_c = 0.7$ y parámetro de recombinación α se evalúa aleatoriamente para cada uno de los individuos sometidos al operador.
- Operador de mutación: mutación orientada, con probabilidad de mutación $P_m = 0.1$.
- $NIND = 25$
- Parámetros de los criterios de finalización $MAXGEN = 15$ y $PRECI = 14$.

Con la selección de $NIND$ y $MAXGEN$ realizada se consigue un coste computacional de 375 evaluaciones de la función objetivo. Esto asegura la ejecución en tiempo real del algoritmo programado en Matlab (se consume casi todo el tiempo del periodo de muestreo). A pesar del bajo número de evaluaciones de la función objetivo se consigue una calidad de la solución suficiente. Es de suponer que si se programa en lenguaje 'C' para su aplicación al invernadero, los resultados de coste computacional serán mucho menores.

Los resultados que se obtienen se muestran en la figura 6.13 donde se muestra también el control que se obtiene con los reguladores PID. Como se ve, el control predictivo consigue una mayor calidad en el control: las variables de salida siguen mejor las referencias y las acciones de control no son tan bruscas.

En estos ensayos, los valores de las variables exteriores son reales y corresponden a los del 28 de noviembre de 1998. El resto de variables se obtienen por simulación del controlador y del proceso todo ello en Matlab.

Las funciones a minimizar en cada instante de muestreo son similares a la que se muestra en la figura 6.14 incluso ocasionalmente, en algunos instantes de muestreo aparecen dos mínimo. Como se puede observar en la figura, la función no es convexa y por tanto no es aplicable un algoritmo SQP. Si se utiliza como optimizador el algoritmo SQP se tiene los resultados de la figura 6.15 donde se ve que no se consigue controlar el proceso.

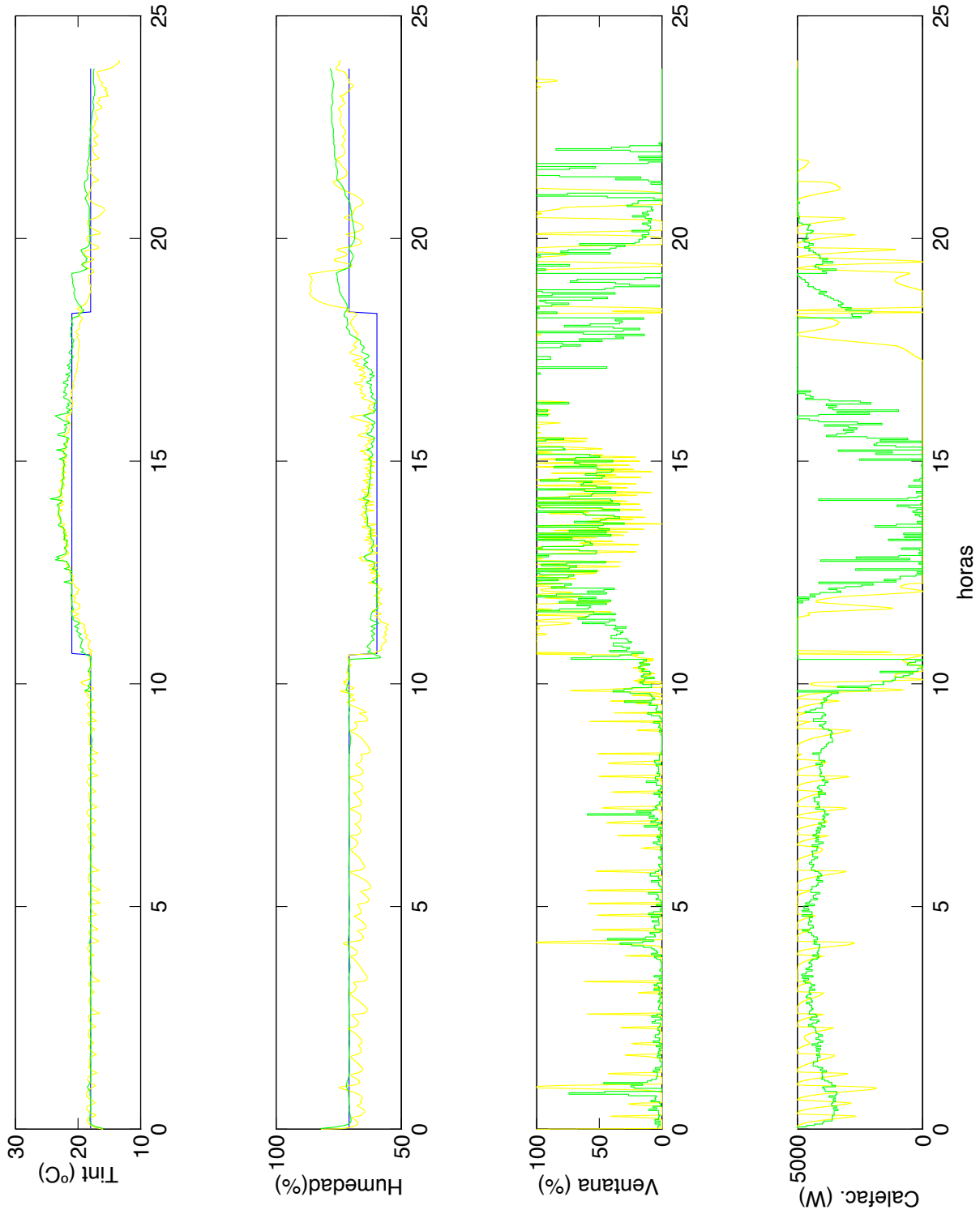


Figura 6.13: Control de temperatura y humedad mediante control predictivo con GA (en verde). Control con PI bucle simple (en amarillo). Referencias en azul.

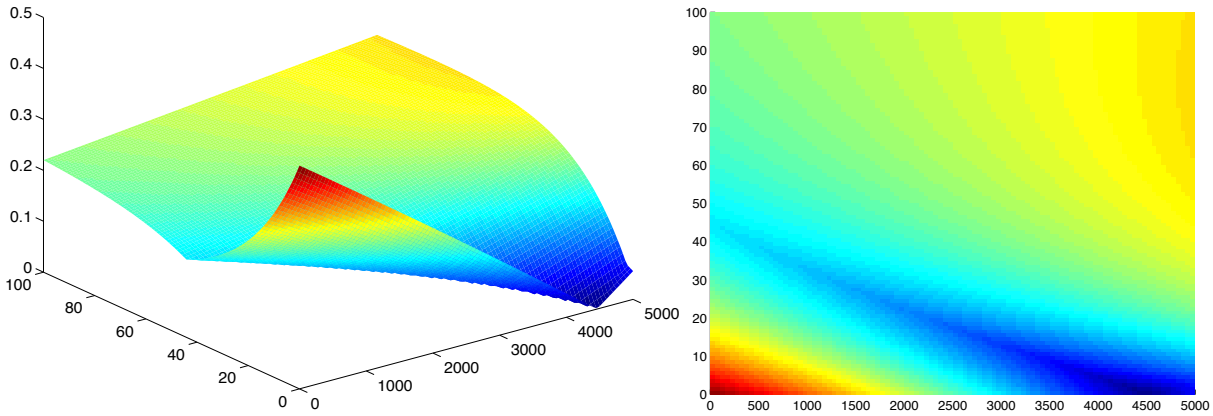


Figura 6.14: Función objetivo resultante en un problema MBPC del control climático de un invernadero para el instante de muestreo 180.

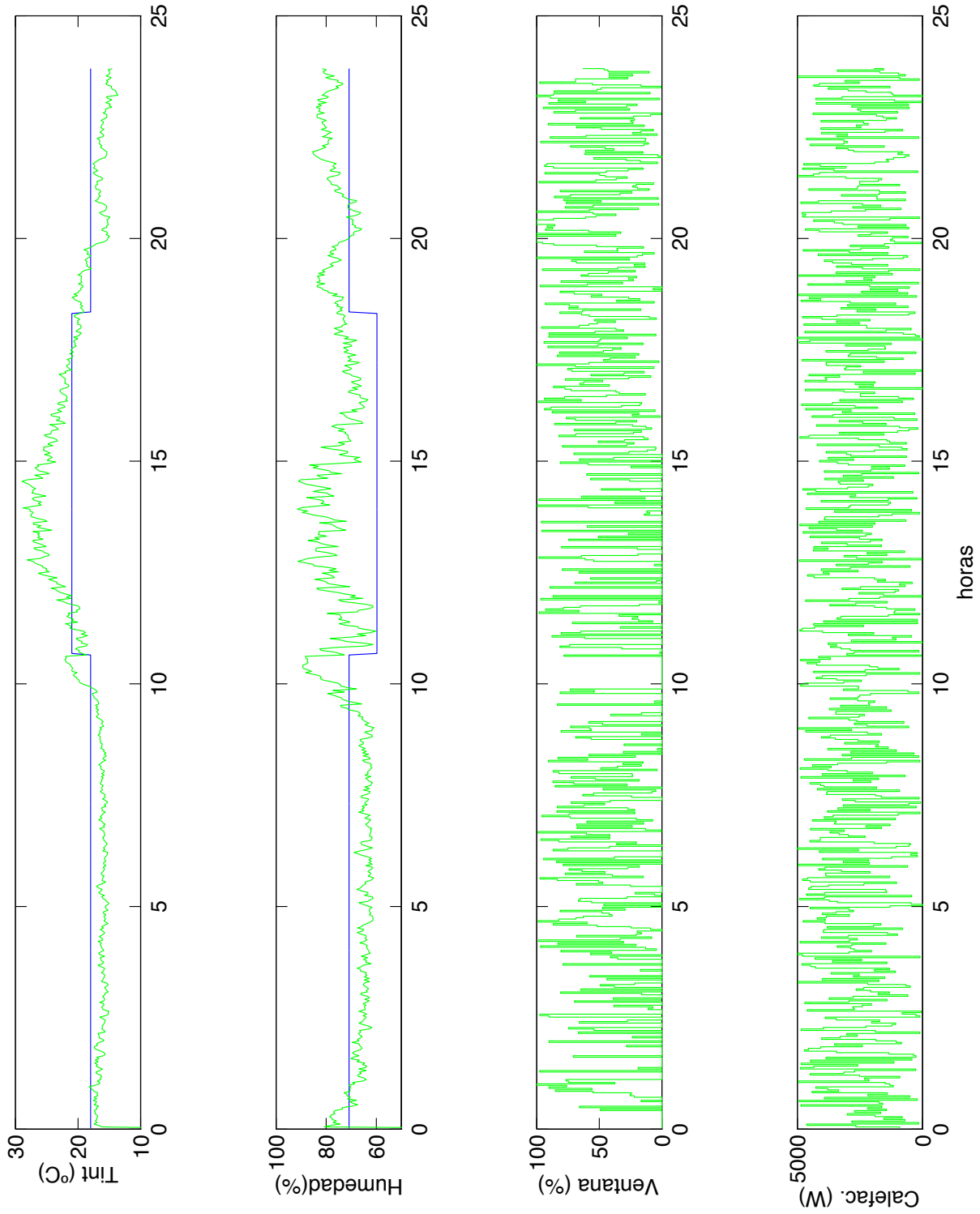


Figura 6.15: Control de temperatura y humedad mediante control predictivo con SQP (en verde). Referencias en azul.

6.3 Conclusiones

En este capítulo se muestra que la propuesta de estructura de control predictivo basado en modelos con optimización heurística se puede extender sin demasiadas dificultades a sistemas MIMO, incluso si se trata de procesos con modelos no lineales.

Se muestra también un ejemplo de aplicación sobre un proceso real: control climático de un invernadero. Se justifica la utilidad de plantear un control predictivo con optimización heurística por la complejidad del modelo no lineal, y la presencia de perturbaciones importantes que dificultan el control con una estructura más simple (PID con prealimentaciones para desacoplar). Los pasos seguidos en el diseño y validación del control predictivo del invernadero han sido:

1. Modelado del proceso a partir de principios fundamentales.
2. Validación del modelo no lineal obtenido con los datos experimentales procedentes del invernadero real (radiación solar, temperaturas interior y exterior, humedad interior y exterior, velocidad del viento).
3. Diseño y validación de un control con PID con y sin prealimentación a partir de una linealización del modelo.
4. Resultados de simulaciones obtenidos del control con PID sobre el modelo no lineal y con entradas de perturbaciones reales.
5. Diseño del control de un MBPC con optimización heurística utilizando el modelo multivariable no lineal.
6. Resultados de simulación obtenidos del MBPC con optimización heurística sobre el modelo no lineal y con entradas de perturbaciones reales.

En cuanto a los costes computacionales, se seleccionan los parámetros del GA (número de individuos y número máximo de generaciones) de forma que se puede aplicar en tiempo real. Se han ajustado de manera que el cálculo de cada acción de control sea inferior al periodo de muestreo. Se debe tener en cuenta que las simulaciones se han realizado en Matlab por lo que es previsible que la versión en lenguaje 'C' (para la implementación del prototipo) todavía sea más rápida.

Capítulo 7

Conclusiones finales y trabajos futuros

En el primer capítulo de esta tesis doctoral se ha presentado una revisión de los conceptos básicos que intervienen en el control predictivo basado en modelos, incluyendo la evolución que ha sufrido hasta el momento actual. El objetivo era poner de manifiesto alguna de las posibles vías de investigación donde aportar alguna contribución en esta área. Parte de la experiencia adquirida en este campo, que ha servido para redactar el capítulo, se ha reflejado en las siguientes contribuciones como coautor [86, 87, 89, 90], éstas han permitido adquirir experiencia práctica en la aplicación de controladores GPC. A partir de este punto se han desarrollado los contenidos del presente trabajo y cuyas conclusiones finales se detallan a continuación.

7.1 Conclusiones finales

Las principales contribuciones que aporta esta tesis son las siguientes:

1. **Evaluación del potencial de las técnicas de optimización heurística: Algoritmos Genéticos y Simulated Annealing**

   Los pasos que se han tenido que seguir para evaluar estas técnicas son:

   (a) Estudio detallado de estas dos técnicas y sus variantes más actuales.

   (b) Propuesta de configuraciones adecuadas para la aplicación en tiempo real (bajo coste computacional y buena calidad). En el caso del *Simulated Annealing* se plantea una implementación novedosa (ASA) en la que se varia la fase de agitación térmica consiguiendo mejores prestaciones que los algoritmos clásico (CSA) y rápido (FSA).

(c) Evaluación comparativa de las implementaciones que se consideran más aptas para su aplicación a un entorno de control predictivo en tiempo real. Se comparan además con las prestaciones obtenidas por dos algoritmos usuales (SQP y QN).

El algoritmo que se perfila como el candidato más adecuado es el GA con codificación real.

(d) Se propone una metodología para el ajuste de los parámetros de un GA con codificación real.

2. **Propuesta de una estructura flexible de control predictivo basado en modelos que permita la incorporación de modelos no lineales, funciones de coste alternativas y varios tipos de optimizadores.** El capítulo cuatro se centra básicamente en este aspecto. Parte de estos resultados también se pueden encontrar en las contribuciones [64, 63, 9].

Los pasos que se han tenido que seguir son:

(a) Plantear una estructura en la que todos los elementos del MBPC (predictor, función de coste y optimizador) sean totalmente independientes.

(b) Adecuar las técnicas de optimización heurística GA y SA al problema de optimización del control predictivo.

(c) Validar la propuesta, replicando los resultados de un GPC convencional.

3. **Propuesta y validación de las alternativas que aparecen con la utilización de la nueva propuesta de controlador predictivo basado en modelos con optimización heurística.**

(a) Grado de libertad adicional, que se obtiene mediante unas redistribuciones alternativas de las acciones de control a lo largo del horizonte de predicción. Este aspecto también se ha publicado en [11].

(b) Posibilidad de utilizar índices de coste alternativos al cuadrático. Se abre la puerta a la utilización de cualquier tipo de índice alternativo para mejorar las prestaciones del controlador.

(c) Incremento de las prestaciones del controlador mediante la incorporación de no linealidades 'bruscas' en el modelo (saturaciones, zonas muertas, *backlash* e histéresis). Parte de estos resultados se han publicado en [64, 9].

(d) Incremento de las prestaciones por la utilización de modelos no lineales representados en espacio de estado. Parte de estos resultados se han publicado en [10, 9].

(e) Inclusión de restricciones de las variables de salida.

4. **Extensión de la estructura de control predictivo basado en modelos con optimización heurística a procesos MIMO no lineales.** Esta extensión se obtiene de forma natural a partir del método desarrollado para procesos SISO.

5. **Aplicación y validación del control predictivo basado en modelos con Algoritmos Genéticos en un proceso real: Control climático de un invernadero.** Se trata del control de un proceso MIMO no lineal. Se aporta un modelo del clima de un invernadero además de mostrar las mejoras de este control frente al que se obtiene con PID y prealimentaciones.

   La aplicación de la estructura propuesta de control predictivo pero con un algoritmo de *Simulated Annealing* para este proceso se muestra en las contribuciones [62, 94].

7.2 Trabajos futuros

Tras la redacción del presente documento y a la vista de los aspectos que han quedado por cubrir, se plantean los posibles trabajos futuros:

1. Implementación en C de los algoritmos estudiados utilizando las rutinas NAG. Obtener algoritmos eficientes para poder aplicar el MBPC con optimización heurística a procesos industriales. Implementación del MBPC con optimización heurística en el control climático de un cultivo de rosas en invernadero.

2. Aprovechando la experiencia conseguida con los GA con codificación real, obtener unas reglas de ajuste de los parámetros del algoritmo ASA propuesto, para facilitar su aplicación al control predictivo.

3. Evaluar las posibles ventajas de utilizar un GA con codificación binaria frente a la codificación real puesto que las acciones de control son cuantificadas (por ejemplo, con los conversores D/A). Esto podría reducir los costes computacionales.

4. Tratar de reducir los tiempos de cálculo del GA.

5. Utilización de otras técnicas de optimización heurística o combinación de varias de ellas para mejorar los costes computacionales. Por ejemplo, utilizar un GA para acercarse al óptimo y un SQP para refinar la solución.

6. Investigar nuevas funciones de coste que permitan mejorar el control de manera que se ajusten mejor a las especificaciones de los usuarios.

7. Análisis de otros modelos de perturbaciones utilizados en sistemas no lineales.

Apéndice A

Validación para procesos lineales

Este apartado pretende únicamente reproducir resultados que ya se han obtenido con un GPC clásico con el afán de validar la propuesta de MBPC con optimización heurística. En este apartado no se muestran la nuevas prestaciones que se pueden aportar con esta técnica.

La simulación es similar a la realizada en [29] que ha sido tradicionalmente referenciada en numerosos trabajos. Se extiende a lo largo de 400 periodos de muestreo, cambiando la función de transferencia del sistema cada 80 periodos de muestreo. La funciones de transferencia utilizadas se muestran en la tabla A.1. El modelo del proceso para las predicciones se obtiene de discretizar el modelo continuo con un retenedor de orden cero y un periodo de muestreo de 1 segundo.

La señal de referencia se repite cada 80 periodos de muestreo, empieza en un valor de 10 durante 20 instantes de muestreo, 25 los siguientes 20 instantes, vuelve a 10 durante 20 instantes y finalmente toma un valor nulo en los últimos 20 instantes de muestreo.

Los parámetros asignados son $N_1 = 1$, $N_2 = 10$, $N_u = 1$, $\lambda(j) = 0$ para la primera simulación (figura A.1) y los mismos parámetros salvo $N_u = 2$ para la segunda simulación (figura A.2). La técnica de optimización consiste en un algoritmo genético con codificación binaria (se podría haber utilizado cualquiera de las técnicas de optimización heurística descritas obteniéndose resultados similares) con las siguientes características:

- 16 bits por parámetro. Longitud del cromosoma para $N_u = 1$ es de $L = 16$ bits y de $L = 32$ bits para $N_u = 2$.
- Número de individuos por generación: 40 para $N_u = 1$ y 100 para $N_u = 2$.
- Número máximo de generaciones (criterio de finalización del algoritmo): 25.

Número de proceso	Proceso	Modelo del proceso
1	$\frac{1}{1+10s+40s^2}$	$\frac{0.0115z^{-1}+0.0106z^{-2}}{1-1.7567z^{-1}+0.7788z^{-2}}$
2	$\frac{e^{-2.7s}}{1+10s+40s^2}$	$\frac{0.0011z^{-3}+0.0159z^{-4}+0.0051z^{-5}}{1-1.7567z^{-1}+0.7788z^{-2}}$
3	$\frac{e^{-2.7s}}{1+10s}$	$\frac{0.0296z^{-3}+0.0656z^{-4}}{1-0.9048z^{-1}}$
4	$\frac{1}{1+10s}$	$\frac{0.0952z^{-1}}{1-0.9048z^{-1}}$
5	$\frac{1}{10s(1+2.5s)}$	$\frac{0.0176z^{-1}+0.0154z^{-2}}{1-1.6703z^{-1}+0.6703z^{-2}}$

Tabla A.1: Funciones de transferencia de los procesos y sus modelos discretos.

- Rango del espacio de búsqueda se ajusta de manera que no se llegue a la saturación. Para $N_u = 1$, $[-100, 100]$ y para $N_u = 2$, $[-400, 420]$.
- Selección mediante *Stochastic Universal Sampling*.
- Cruce por punto simple, $P_c = 0.7$.
- Mutación aleatoria con $P_m = 0.7/L$.

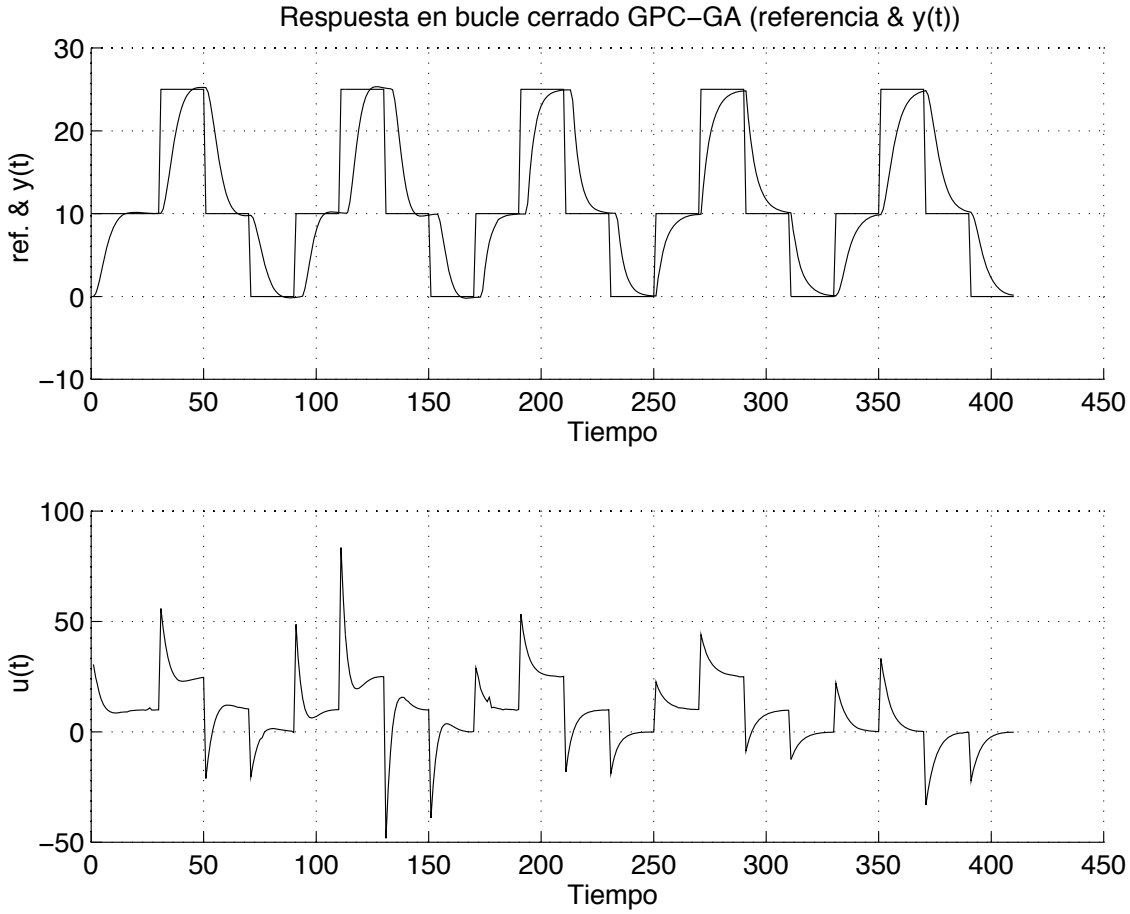


Figura A.1: Controlador GPC con optimización heurística, $Nu = 1$.

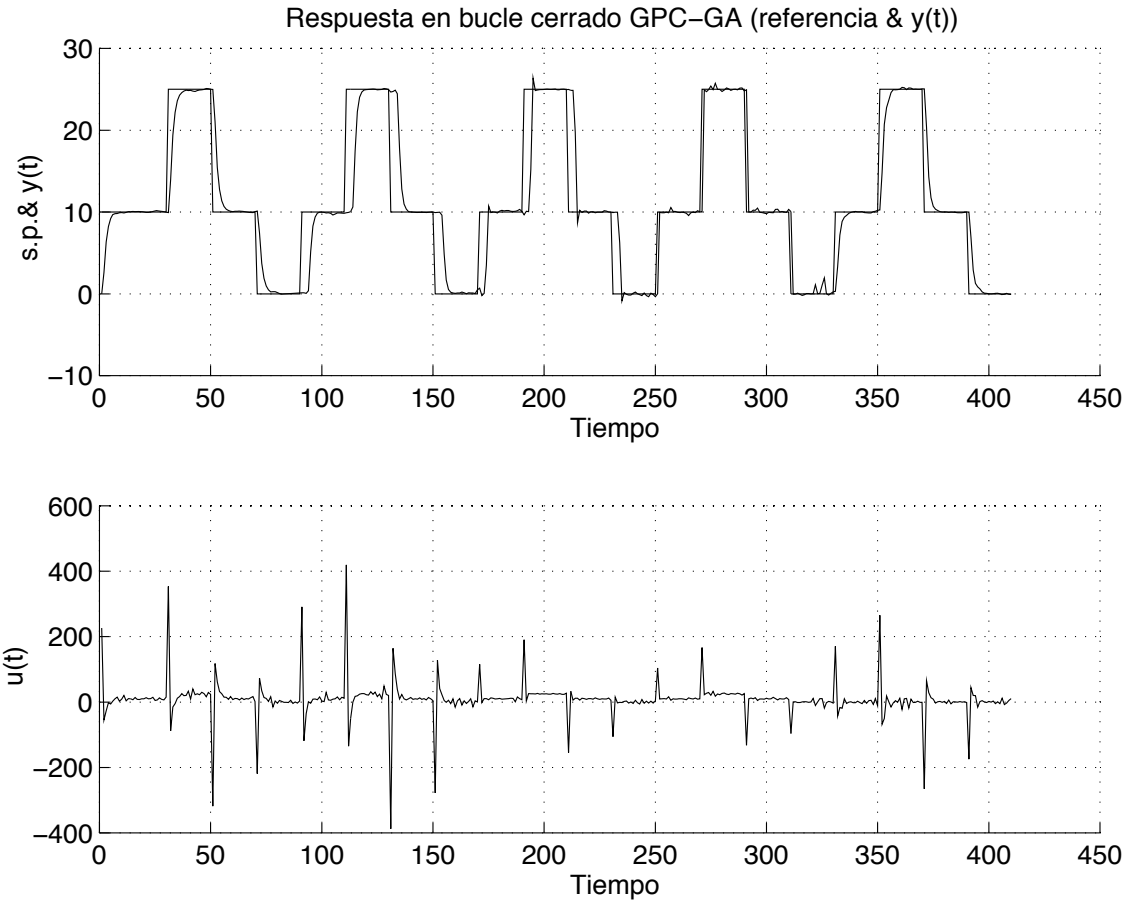


Figura A.2: Controlador GPC con optimización heurística, $Nu = 2$.

Instantes de muestreo	Referencia
1-20	20
21-40	40
41-340	60
341-360	40
361-380	20

Tabla A.2: Referencia aplicada en la simulación.

Los resultados que se obtienen coinciden con los presentados en [29]. En estas simulaciones se ha ajustado el polinomio $T(z^{-1}) = 1$. Para validar el comportamiento de la propuesta (GPC con optimización heurística) con polinomios de diferentes valores se plantean las siguientes simulaciones.

El modelo continuo del proceso es:

$$G(s) = \frac{e^{-2s}}{50s^2 + 15s + 1}$$

Para obtener los modelos discretos para las predicciones (polinomios $A(z^{-1})$ y $B(z^{-1})$) se discretiza el modelo continuo con un retenedor de orden cero y un periodo de muestreo de 1 segundo. Como proceso se utiliza la misma discretización, $A_p(z^{-1}) = A(z^{-1})$ y $B_p(z^{-1}) = B(z^{-1})$), La referencia que se aplica se describe en la tabla A.2.

Las perturbaciones ($d(t)$) que se aplican son de dos tipos generalmente denominadas *dcu* y *dcy* y no aparecen en el modelo que se utiliza para las predicciones (ver expresión 4.3).

- El proceso con las perturbaciones *dcu* corresponden a:

$$A_p(z^{-1})y(t) = B_p(z^{-1})u(t) + d(t)$$

- El proceso con perturbaciones *dcy* corresponde a:

$$A_p(z^{-1})y(t) = B_p(z^{-1})u(t) + A_p(z^{-1})d(t)$$

Los instantes en los que se aplican estas perturbaciones y su valor se muestran en la tabla A.3.

Los parámetros del GPC son, en todas las simulaciones, $N_1 = 1$, $N_2 = 10$ y $N_u = 1$. Se han desarrollado tres simulaciones con distintos polinomios $T(z^{-1})$:

Instantes de muestreo	Tipo de perturbación	Valor de la perturbación
71-100	*dcu*	+3
151-180	*dcy*	+3
211-240	*dcy*	Valor aleatorio entre [-5 5]
270-300	*dcu*	-3

Tabla A.3: Perturbaciones en la simulación.

- *Sim1* Polinomio $T(z^{-1}) = 1$. Esta simulación presentaría los mismos resultados que con el GPC clásico.
- *Sim2* Polinomio $T(z^{-1}) = 1 - 0.8z^{-1}$. Aplica un filtrado sobre las perturbaciones.
- *Sim3* Polinomio $T(z^{-1}) = 1 - z^{-1}$. Filtrado más desfavorable para una perturbación constante puesto que se cancela el factor Δ del modelo de perturbaciones y por tanto el controlador tiene dificultades para ajustar correctamente el régimen permanente.

Se obtienen los mismos resultados que con un GPC analítico (ver [92]).

En todos los resultados anteriores se comprueba que la estructura propuesta es totalmente equivalente a un GPC convencional. Se llega a la conclusión de que es posible utilizar una técnica de optimización heurística para obtener resultados equivalentes a los del GPC, pero evidentemente la propuesta no es computacionalmente competitiva frente al coste del GPC. En cualquier caso, con los ensayos realizados sólo se pretendía una validación para posteriormente mostrar las ventajas que puede aportar. Como ya se ha comentado, la propuesta abre el campo a:

- Otros tipos de modelos.
- Otros tipos de índices.
- Inclusión de restricciones de forma natural.

A cambio el precio que se paga es el incremento del coste computacional que puede invalidar su aplicación en algunos procesos.

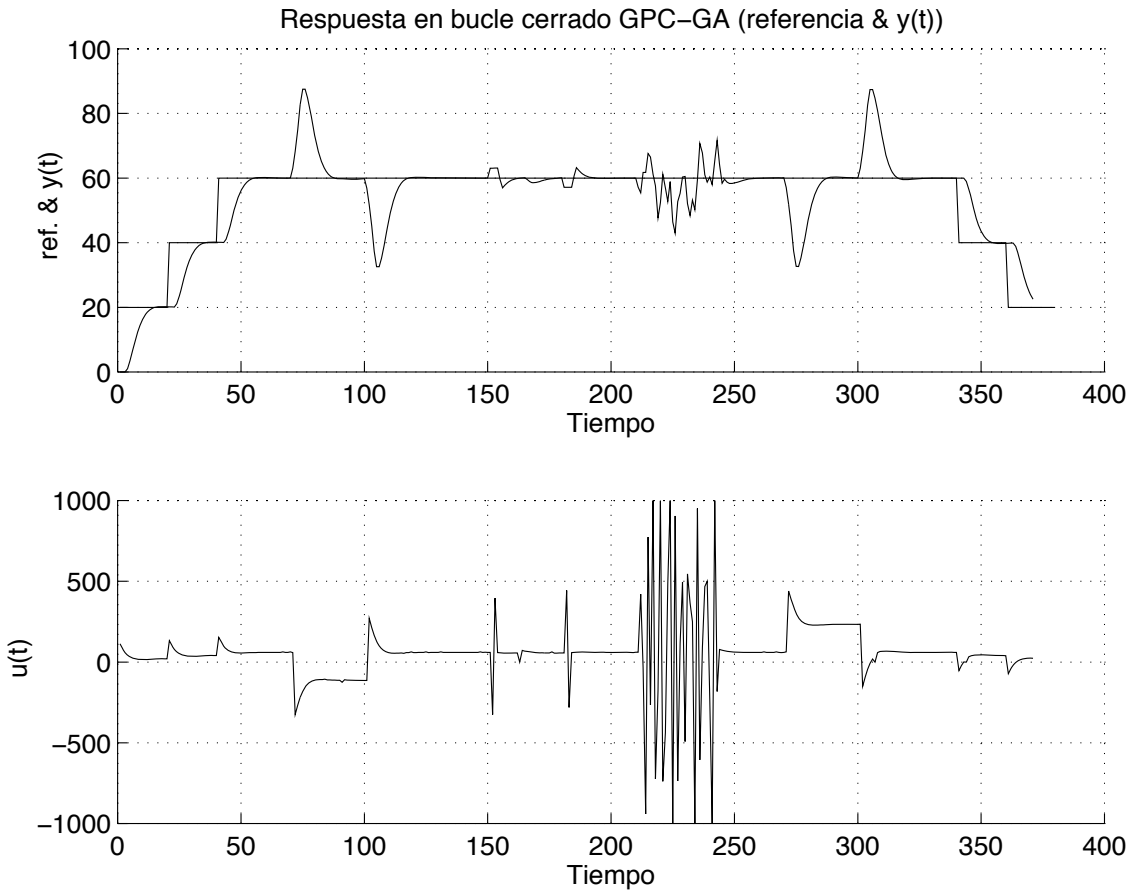


Figura A.3: *Sim1*: Simulación con $T(z^{-1}) = 1$

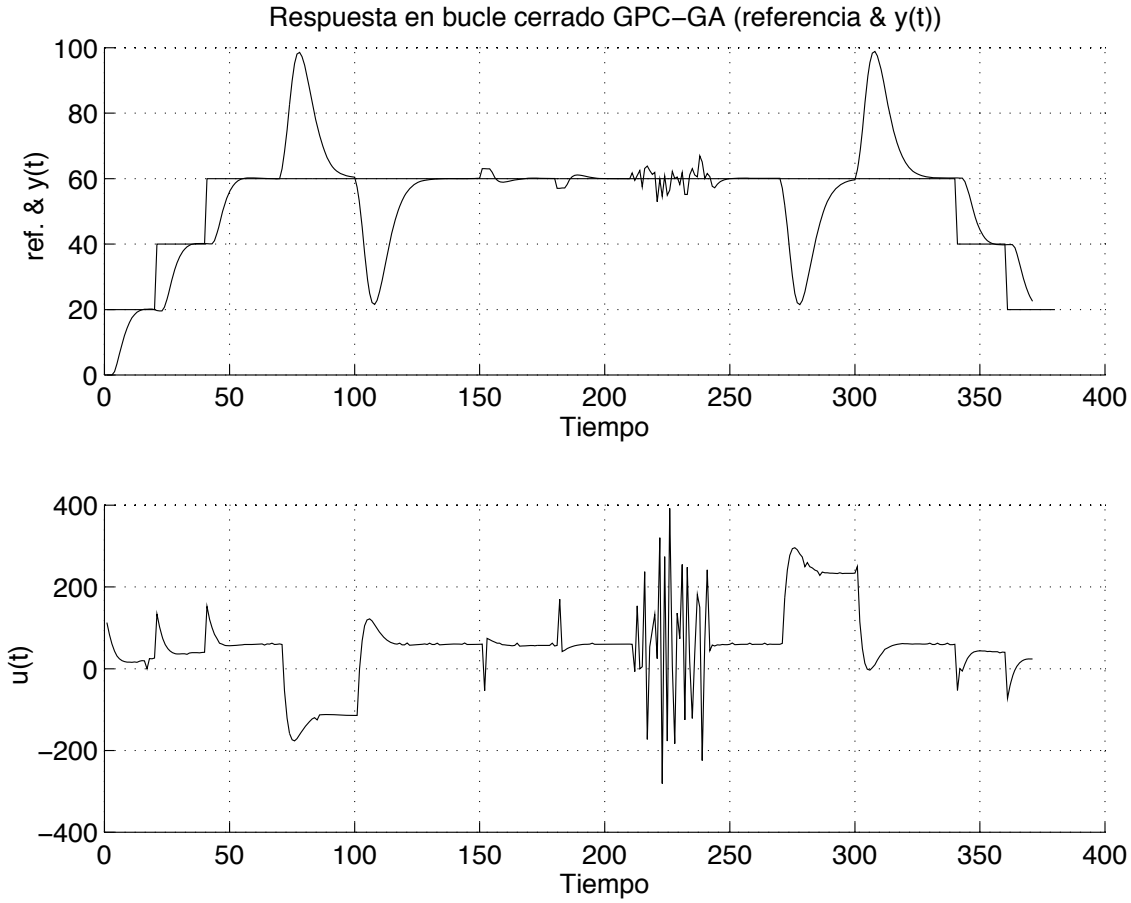


Figura A.4: *Sim2*: Simulación con $T(z^{-1}) = 1 - 0.8z^{-1}$

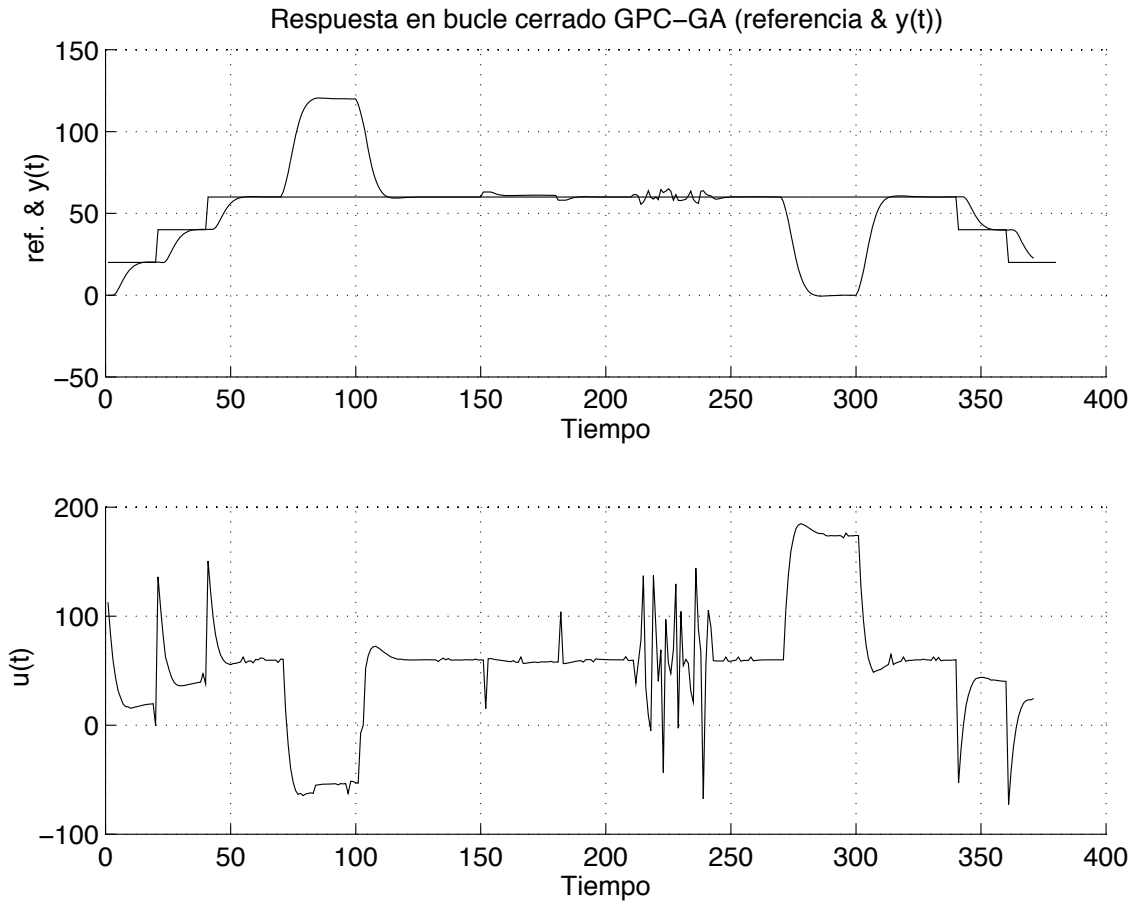


Figura A.5: *Sim3*: Simulación con $T(z^{-1}) = 1 - z^{-1}$

Apéndice B

Funciones de test

F1: Función Modelo Esférico

$$f1(x_1, x_2) = x_1^2 + x_2^2$$

Espacio de búsqueda: $-5.12 \leq x_i \leq 5.12$, $i = 1, 2$ Solución: $\min(f1) = f1(0, 0) = 0$

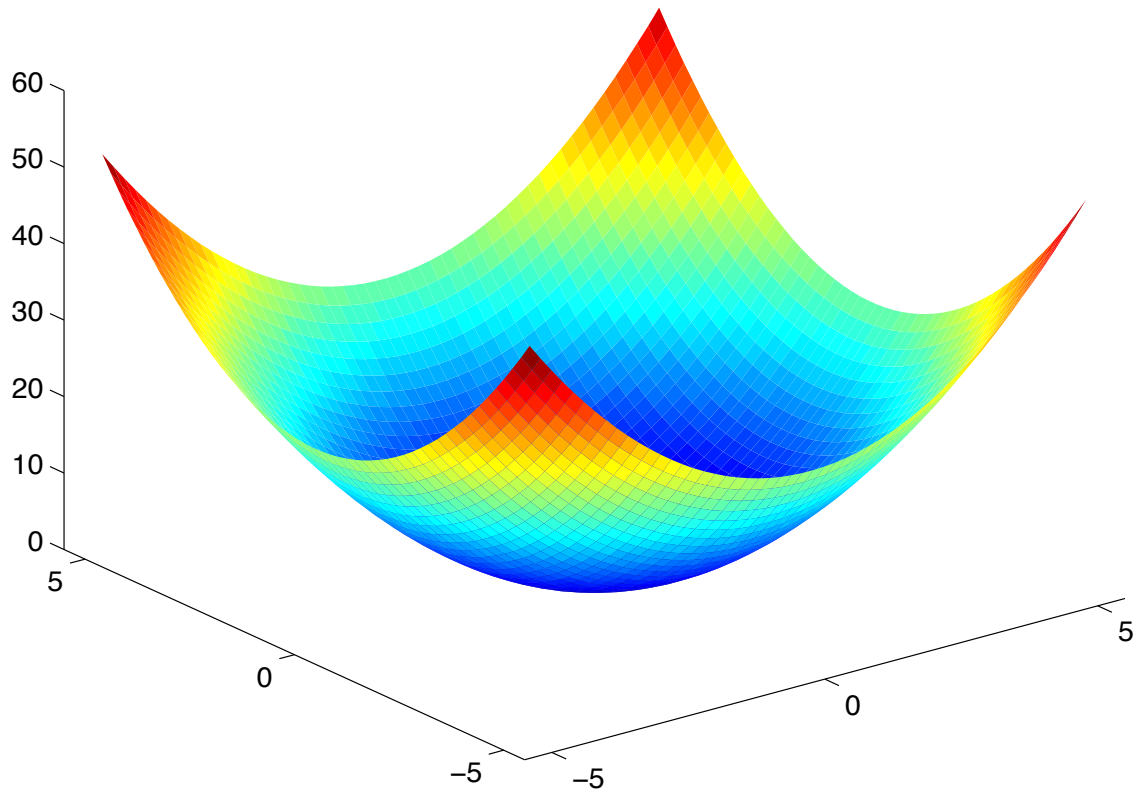


Figura B.1: Función de test F1.

Se trata de una función cuadrática típica tiene un sólo óptimo y no presenten dificultades añadidas.

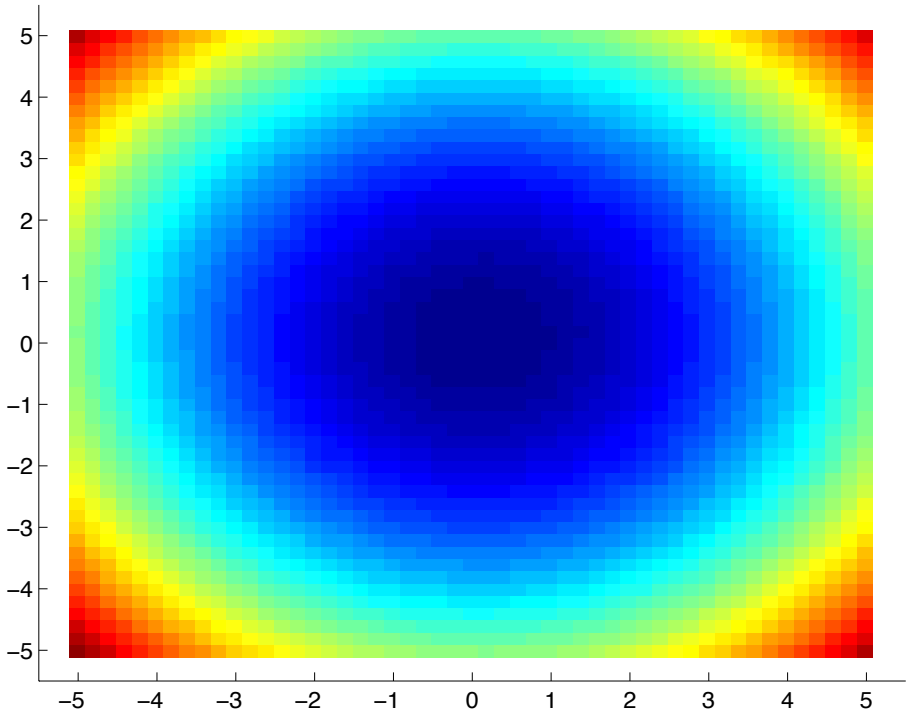


Figura B.2: Función de test F1. Vista superior.

F2: Función de Rosenbrok Generalizada

$$f2(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

Espacio de búsqueda: $-5.12 \leq x_i \leq 5.12$, $i = 1, 2$ Solución: $\min(f2) = f2(1, 1) = 0$

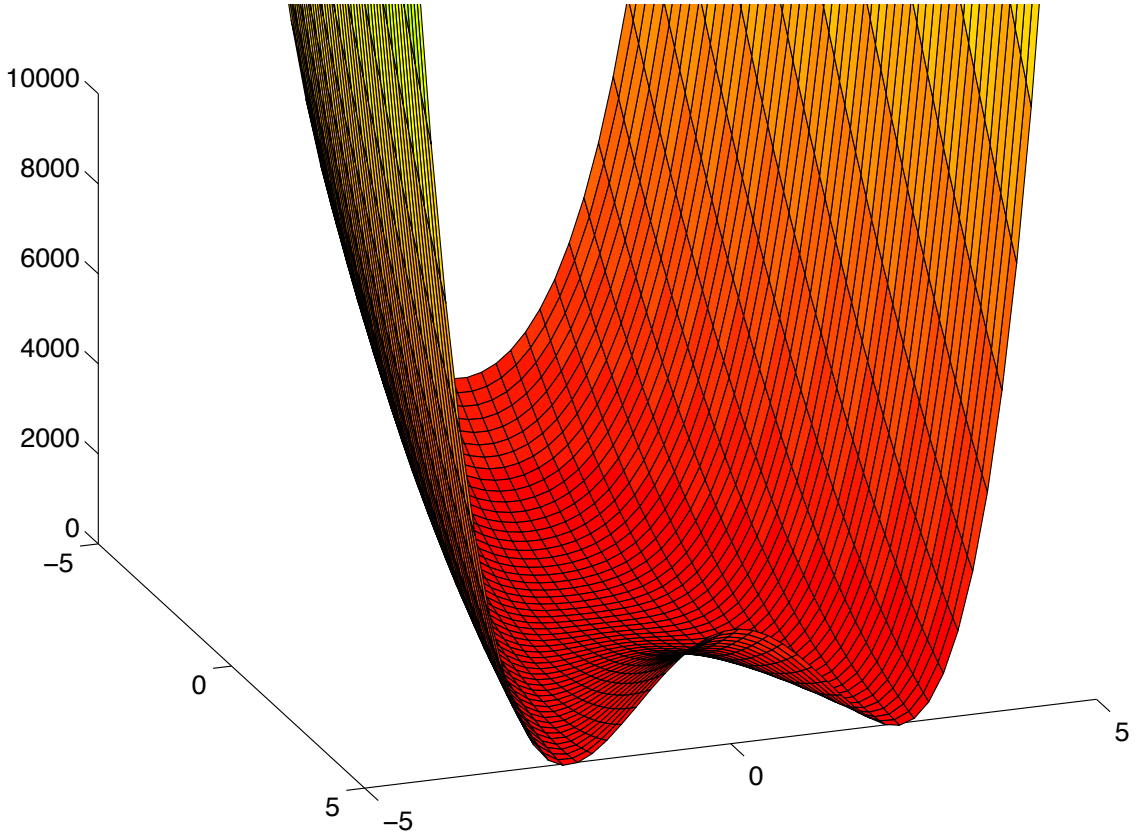


Figura B.3: Función de test F2.

Esta función tiene un único óptimo pero se encuentra en una zona muy plana que puede dificultar su localización.

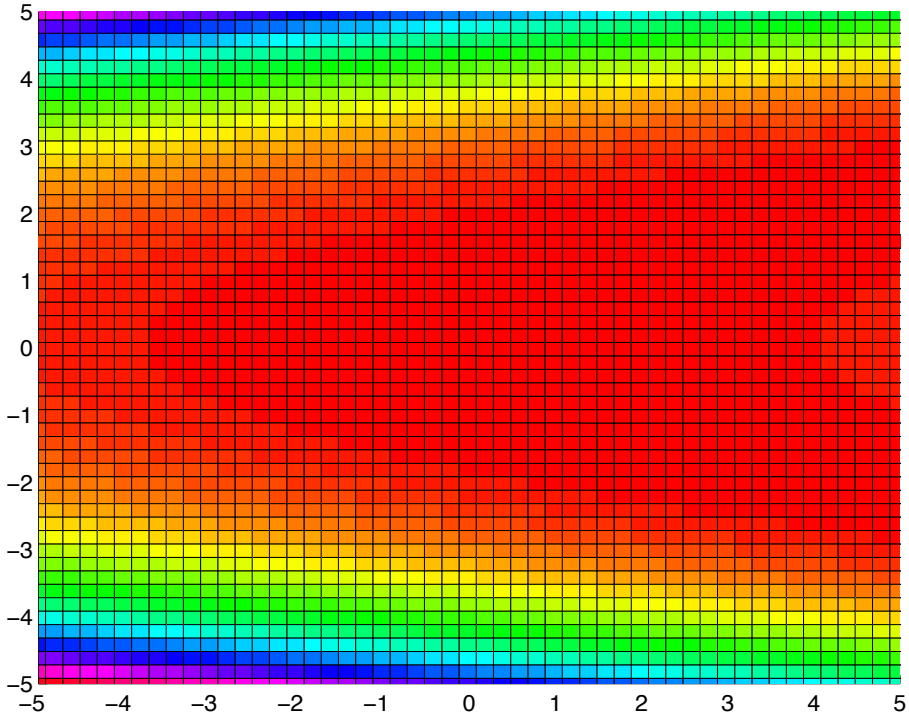


Figura B.4: Función de test F2. Vista superior.

F3: Función Paso

$$f3(x_1, x_2) = 6 \cdot 2 + \lfloor x_1 \rfloor + \lfloor x_2 \rfloor$$

Espacio de búsqueda: $-5.12 \leq x_i \leq 5.12,\ i = 1, 2$

Solución: $\min(f3) = f3([-5.12, -5], [-5.12, -5]) = 0$

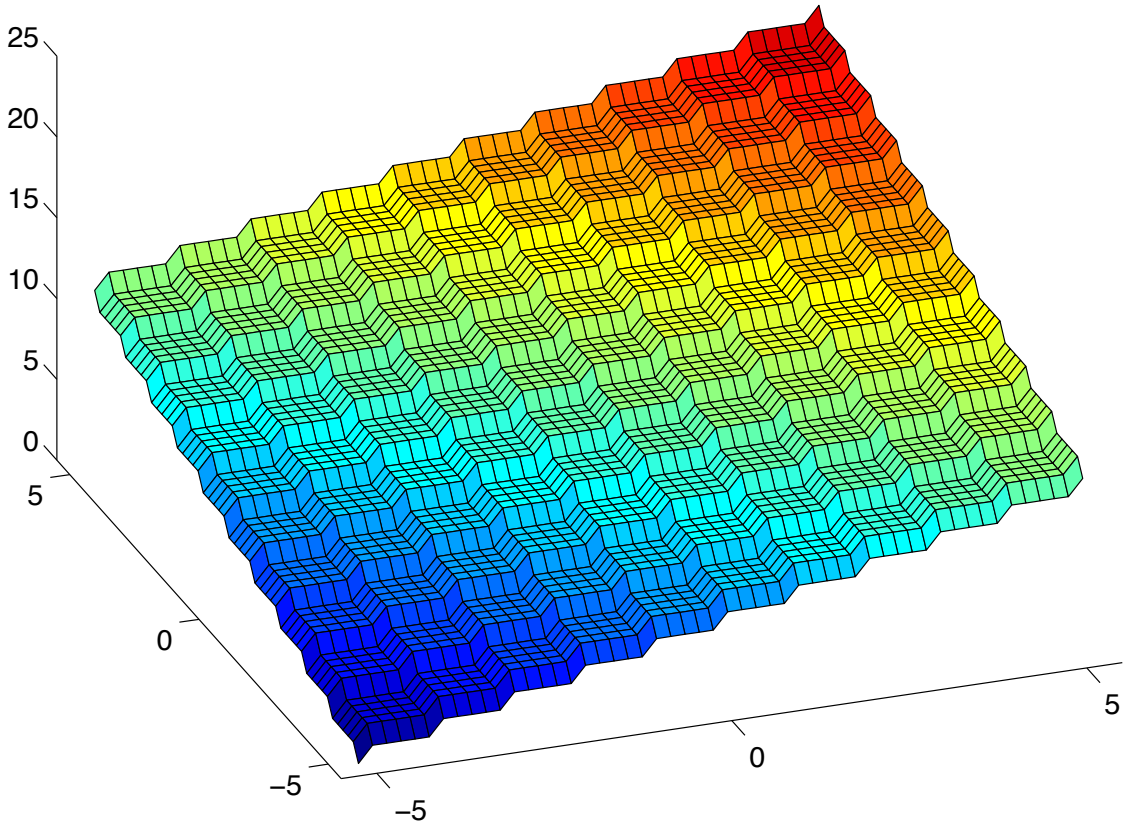


Figura B.5: Función de test F3.

Es una función con infinitos óptimos globales pues se trata de una zona (ocurre lo mismo con los óptimos locales), la dificultad de esta función estriba en que está formada por escalones.

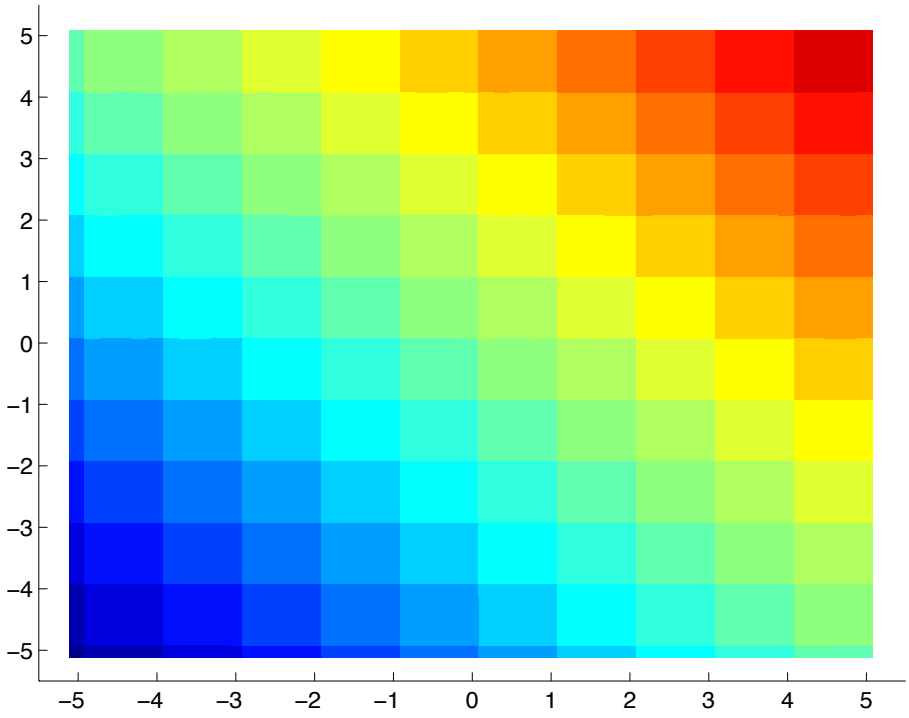


Figura B.6: Función de test F3. Vista superior.

F4: Shekels Foxholes

$$\frac{1}{f4(x_1,x_2)} = \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{c_j + \sum_{i=1}^{2}(x_i - a_{ij})^6}$$

$$(b_1) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 \end{pmatrix}$$

$$(u) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$(a_{ij}) = \begin{pmatrix} b_1 & b_1 & b_1 & b_1 & b_1 \\ -32 \cdot u & -16 \cdot u & 0 \cdot u & 16 \cdot u & 32 \cdot u \end{pmatrix}$$

$$c_j = f5(a_{1j}, a_{2j})$$

Espacio de búsqueda: $-65.536 \leq x_i \leq 65.536,\ i = 1,2$

Solución: $\min(f4) = f4(0,0) \approx 0$

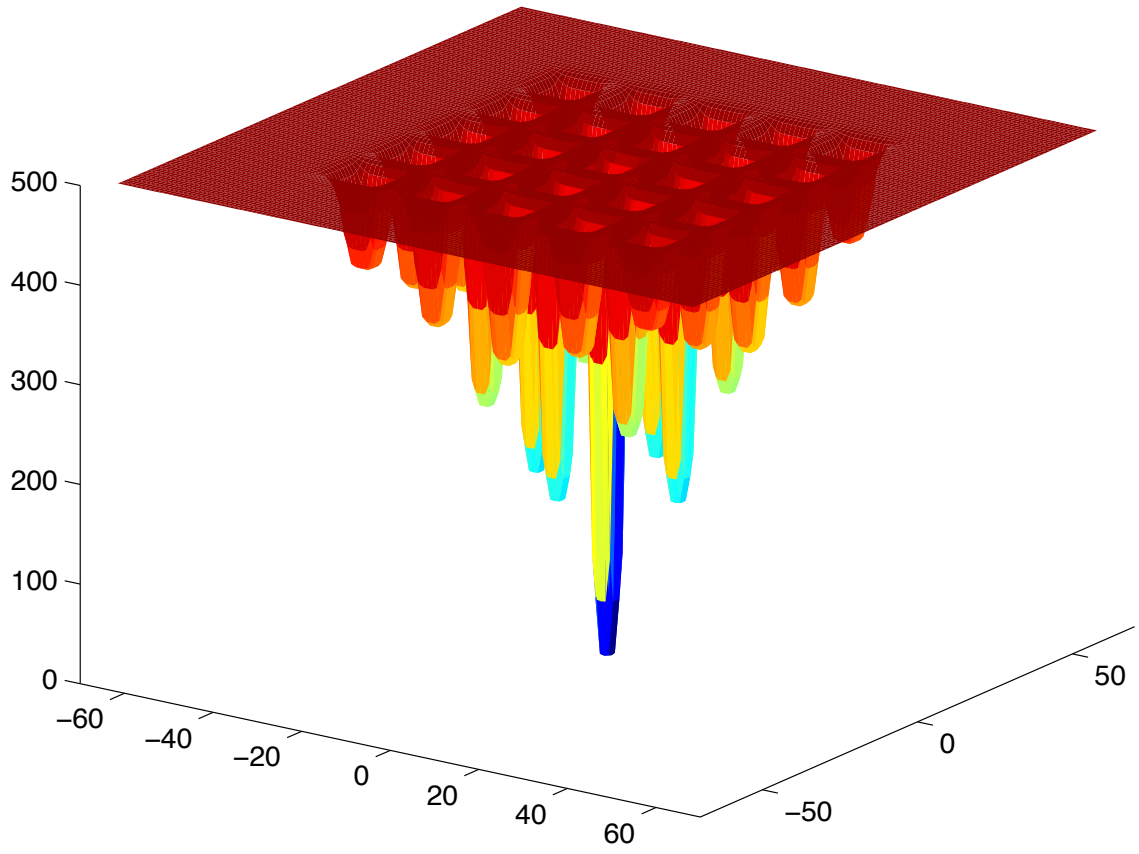


Figura B.7: Función de test F4.

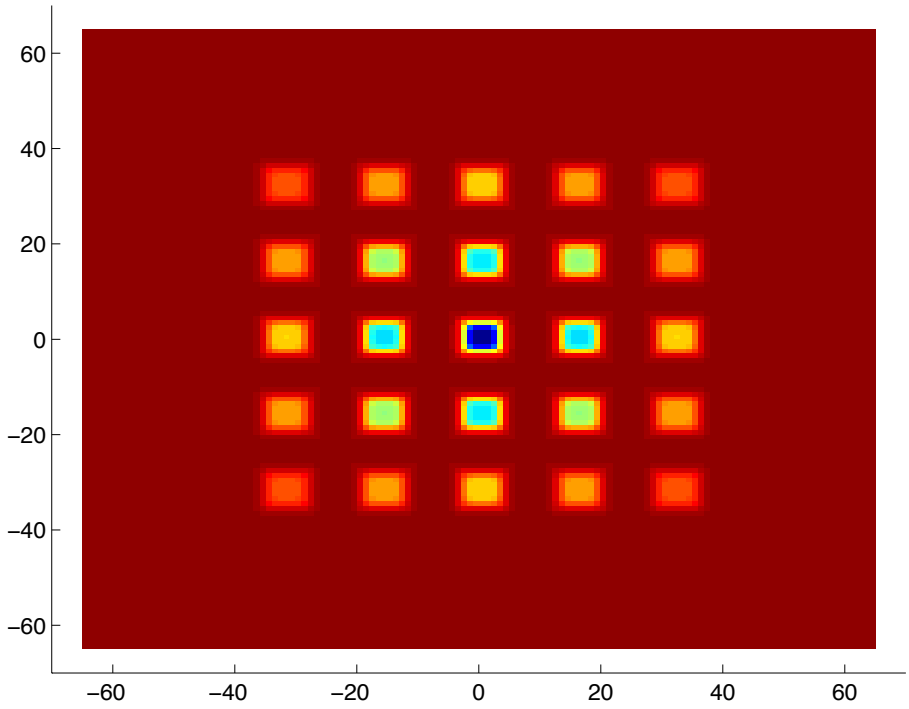


Figura B.8: Función de test F4. Vista superior.

Es una función con un único mínimo global y múltiples minimos locales. Todos los mínimos se encuentran en depresiones muy pronunciadas. En definitiva se trata de una función relativamente plana que cae bruscamente a unos mínimos.

F5: Funcion de Rastrigin Generalizada

$$f5(x_1, x_2) = 2 \cdot 10 + x_1^2 + -10\cos(2\pi \cdot x_1) + x_2^2 - 10\cos(2\pi \cdot x_2)$$

Espacio de búsqueda: $-5.12 \leq x_i \leq 5.12$, $i = 1, 2$

Solución: $\min(f5) = f5(0, 0) = 0$

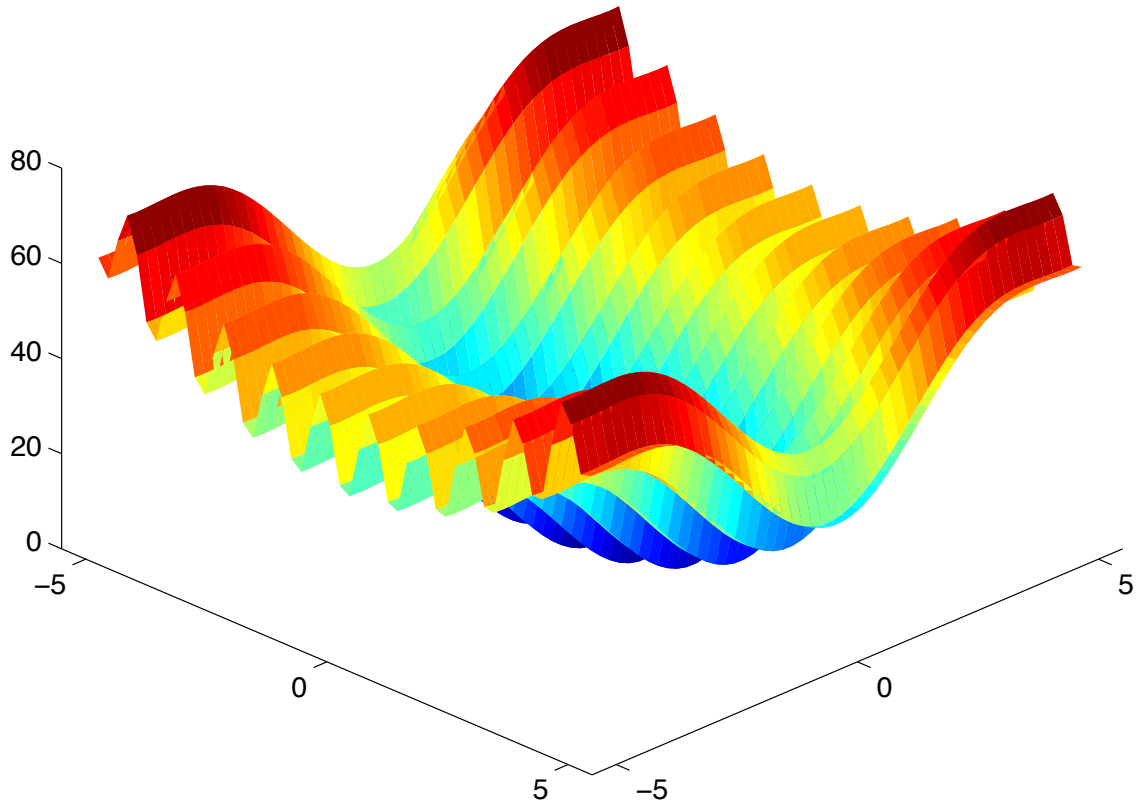


Figura B.9: Función de test F5.

Esta función presenta una forma acanalada y en cada uno de estos canales se encuentra un mínimo todos locales excepto uno que es el mínimo global. Este tipo de función presenta problemas para los algoritmos que se utilizan el gradiente pues, estos algoritmos tienen problemas para salir de un canal para saltar a otro.

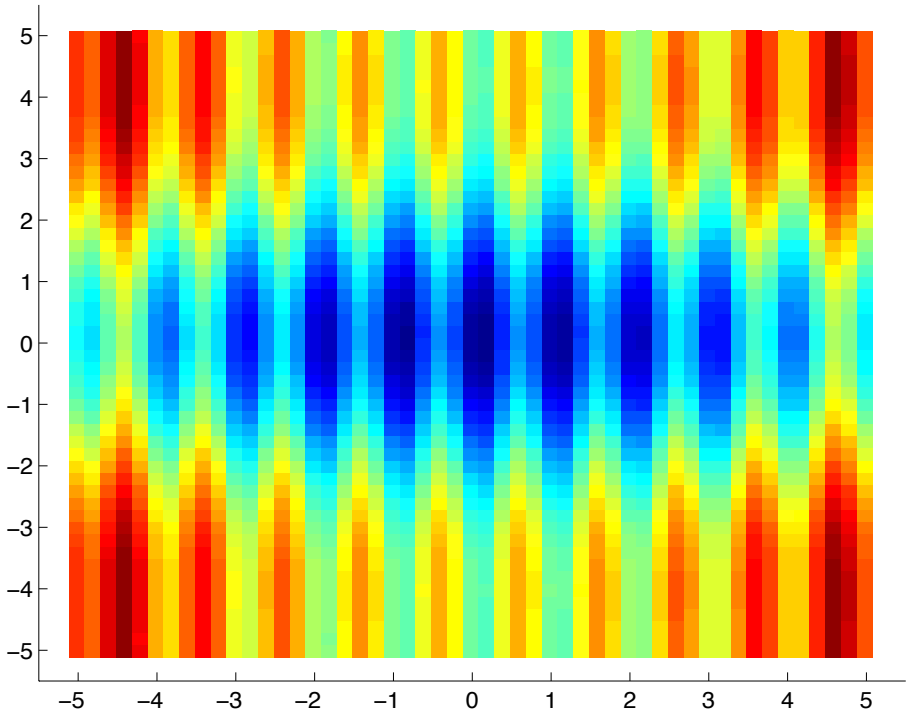


Figura B.10: Función de test F5. Vista superior.

F6: Función de Schwefel

$$f6(x_1, x_2) = -x_1 \cdot \sin(\sqrt{|x_1|}) - x_2 \cdot \sin(\sqrt{|x_2|})$$

Espacio de búsqueda: $-500 \leq x_i \leq 500$, $i = 1, 2$

Solución: $\min(f6) = f6(420.9687, 420.9687) = -837.9658$

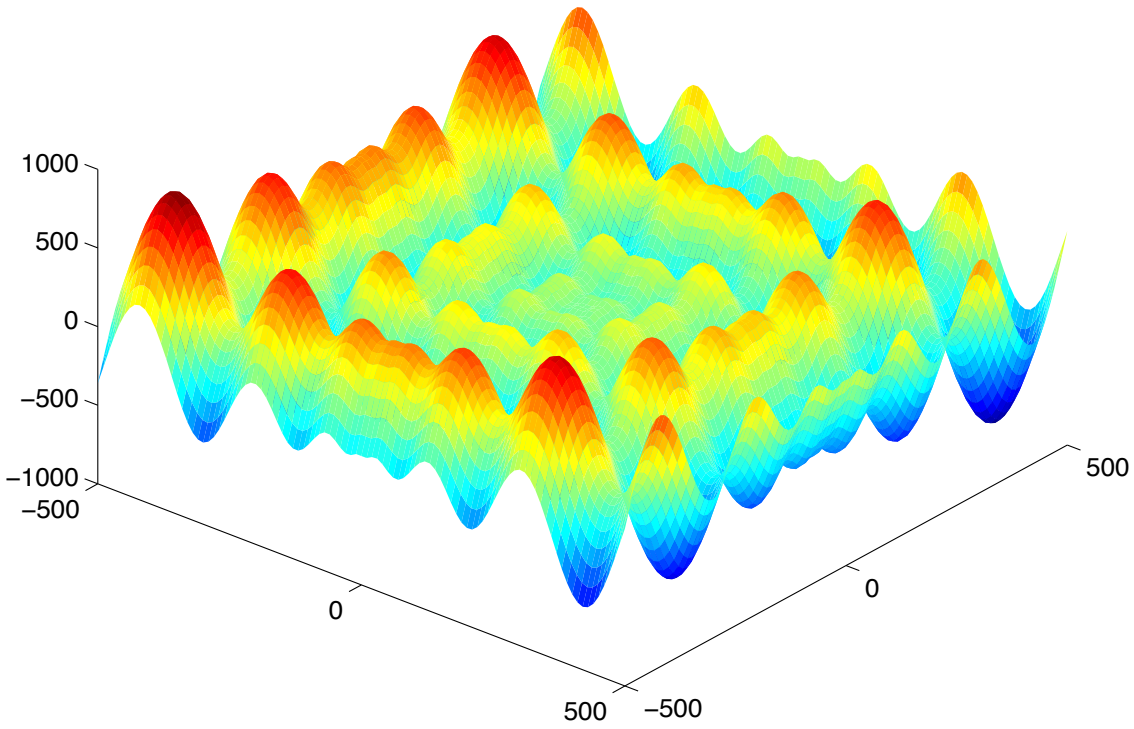


Figura B.11: Función de test F6.

Esta función presenta múltiple mínimos con la particularidad de que existe dos mínimos locales muy similar en valor al mínimo global pero muy alejado del mismo, puntos $(-302.5249, 420.9687)$ y $(420.9687, -302.5249)$.

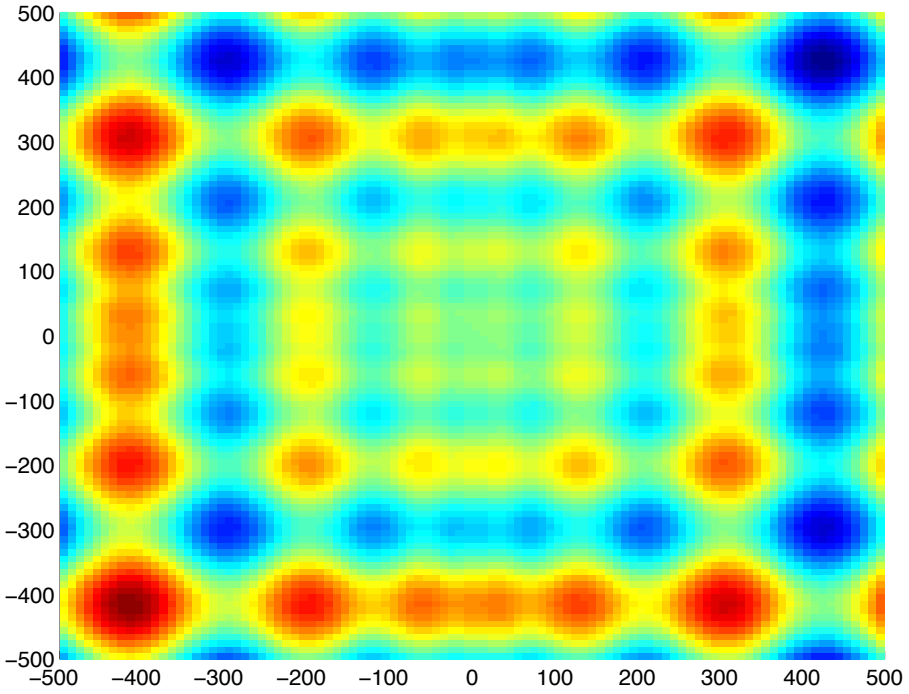


Figura B.12: Función de test F6. Vista superior

F7: Función de Griewank

$$f7(x_1, x_2) = \frac{1}{4000} \cdot \left(x_1^2 + x_2^2\right) - \cos(x_1) \cdot \cos(\frac{x_2}{\sqrt{2}}) + 1$$

Espacio de búsqueda: $-600 \leq x_i \leq 600$, $i = 1, 2$

Solución: $\min(f7) = f7(0, 0) = 0$

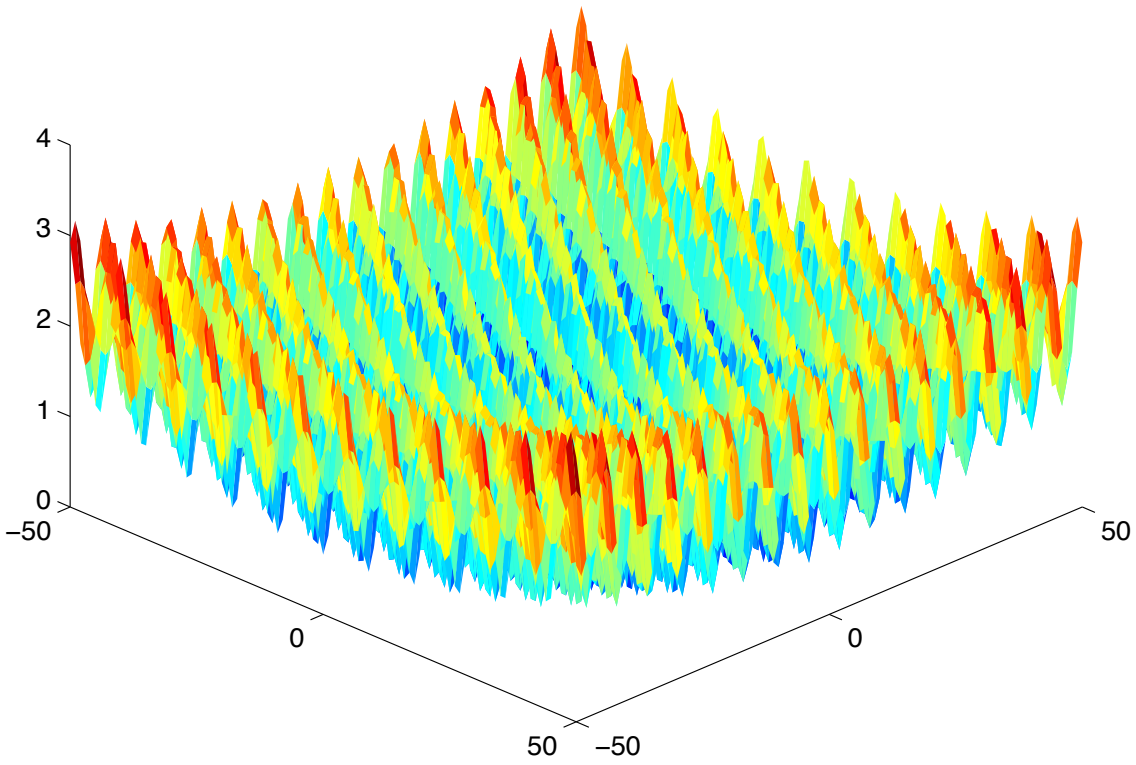


Figura B.13: Función de test F7.

Función numerosos mínimos locales muy cercanos unos de otros.

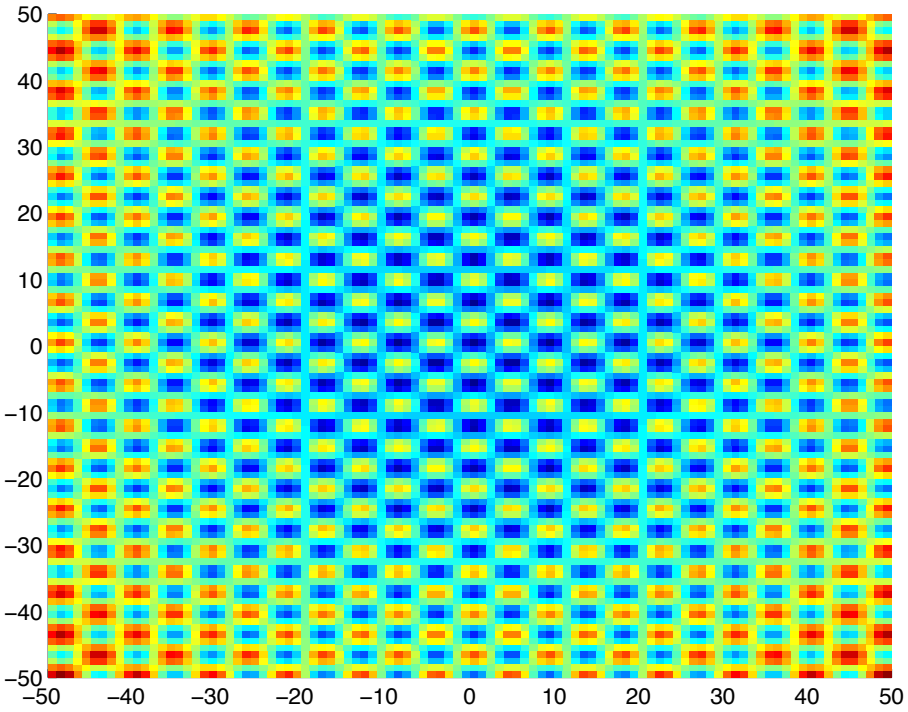


Figura B.14: Función de test F7. Vista superior.

Apéndice C

Tablas de resultados de la evaluación de un GA con codificación real

La figura C.1 muestra un ejemplo utilización de las tablas. En este ejemplo, el rango de individuos que consiguen una solución con una precisión $E(||x - x_o||) < \epsilon$ es $[80, 220]$, dentro de este rango cumple con el criterio de convergencia $F_c \geq 0.9$ el rango $[80, 160]$. Asociada con la información de esta tabla 1, la tabla 2 muestra el coste para los limites de cada rango ($NIND_{min}$ y $NIND_{max}$) y el coste medio del rango en miles de evaluaciones de la función objetivo.

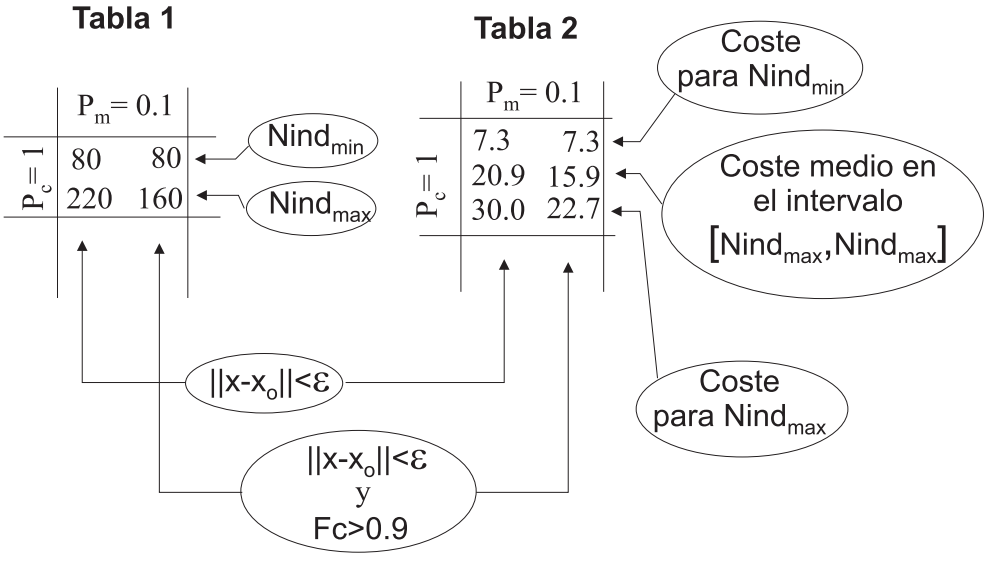


Figura C.1: Utilización de las tabla, ejemplo para la función F2 con los valores $P_m = 0.1$ y $P_c = 1$.

		Prob. de Mutación P_m											
		0		0.05		0.10		0.15	0.20	0.25	0.30	0.35	0.40
Prob. de Cruce P_c	0.0												160 220
	0.1							220 220	220 220	220 220	120 220	140 220	100 220
	0.2								160 220	100 220	100 220	80 220	80 220
	0.3	220 220	220 220	220 220	220 220	220 220		120 220	100 220	100 220	80 220	60 220	80 220
	0.4	160 220	160 220	140 220	140 220	120 220	120 120	140 220	120 220	100 220	60 220	80 220	80 220
	0.5	120 220	120 220	120 220	120 220	120 220	120 140	80 220	100 220	60 220	40 220	60 220	80 220
	0.6	100 220	100 220	100 220	100 220	100 220	100 140	80 220	60 220	60 220	60 220	80 220	120 220
	0.7	140 220	140 220	140 220	140 220	100 220	100 200	80 220	80 220	60 220	60 220	80 220	220 220
	0.8	160 220	160 220	160 220	160 220	100 220	100 140	80 220	80 220	40 220	60 220	100 220	
	0.9	160 220	160 220	100 220	100 220	100 220	100 160	60 220	60 220	40 220	40 220	100 220	
	1.0	80 220	80 220	100 220	100 220	80 220	80 160	60 220	40 220	40 220	40 220	220 220	

Tabla C.1: Función F2. Para cada P_m y P_c se da el rango de número de individuos que cumplen $E(||x - x_o||) < \epsilon$ (1^a columna) y un factor de convergencia mayor F_c (2^a columna).

Prob. de Cruce P_c	Prob. de Mutación P_m										
	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40		
0.0											64.0
											76.5
											88.0
0.1				85.6	88.8	95.9	47.0	53.8	42.5		
				85.6	88.8	95.9	67.4	68.6	62.2		
				85.6	88.8	95.9	86.2	87.7	95.2		
0.2					59.3	35.4	41.4	27.9	29.1		
					69.8	61.3	67.1	53.2	47.3		
					83.5	85.5	99.7	70.3	71.0		
0.3	9.0	9.0	9.6	40.0	55.9	37.1	38.4	31.2	21.7	24.0	
	9.0	9.0	9.6	9.6	40.0	43.6	46.2	62.1	51.2	45.0	47.1
	9.0	9.0	9.6	9.6	40.0	44.2	53.5	90.6	69.8	68.8	81.6
0.4	6.1	6.1	6.2	6.2	13.4	13.4	24.1	33.1	32.8	22.9	23.6	21.8	
	8.3	8.3	7.9	7.9	24.5	13.4	30.9	39.2	58.6	45.5	47.0	45.9	
	8.6	8.6	9.7	9.7	24.7	13.4	35.8	46.8	85.8	74.3	63.5	54.5	
0.5	5.4	5.4	6.0	6.0	13.3	13.3	18.9	40.5	18.4	14.1	20.3	28.1	
	7.3	7.3	8.0	8.0	21.4	14.8	28.5	57.2	46.8	39.5	42.3	43.3	
	9.5	9.5	9.9	9.9	28.4	16.3	36.9	74.1	77.4	58.6	64.3	61.8	
0.6	3.8	3.8	5.0	5.0	9.1	9.1	17.4	24.1	23.9	20.0	18.5	36.0	
	6.7	6.7	8.2	8.2	21.4	13.4	27.0	48.7	41.5	40.7	42.0	50.8	
	9.2	9.2	9.5	9.5	28.4	15.3	37.0	74.1	67.9	69.1	42.9	68.0	
0.7	5.8	5.8	6.5	6.5	11.4	11.4	18.1	26.1	20.8	18.3	22.2	56.6	
	8.5	8.5	8.8	8.8	21.0	19.4	28.7	48.5	41.3	39.9	45.2	56.6	
	8.8	8.8	9.8	9.8	30.6	26.1	40.0	74.6	55.2	71.5	73.4	56.6	
0.8	6.8	6.8	7.9	7.9	13.6	13.6	25.7	22.8	12.3	19.7	36.1		
	8.1	8.1	9.4	9.4	21.1	15.5	51.2	48.4	36.5	43.8	49.0		
	9.3	9.3	10.6	10.6	29.7	18.9	59.3	68.5	62.0	73.7	68.9		
0.9	6.9	6.9	6.7	6.7	14.6	14.6	21.3	16.3	15.9	16.7	34.2		
	8.5	8.5	9.1	9.1	21.8	17.8	48.2	45.5	39.4	39.9	46.2		
	9.2	9.2	12.0	12.0	29.5	26.6	73.6	77.6	68.0	79.5	52.6		
1.0	3.6	3.6	7.5	7.5	7.3	7.3	22.7	13.6	12.9	12.4	71.5		
	7.8	7.8	9.1	9.1	20.9	15.9	42.0	40.7	41.4	39.1	71.5		
	10.8	10.8	12.1	12.1	30.0	22.7	64.1	67.4	73.5	61.0	71.5		

Tabla C.2: Función F2. Coste computacional de los rangos mostrados en la tabla C.1 en miles de evaluaciones de la función objetivo. Se muestra también el valor medio para el rango.

Prob. de Mutación P_m										
	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
Prob. de Cruce P_c 0.0		60 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.1	160 160	20 20 220 20	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.2	100 220	40 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.3	60 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.4	60 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.5	80 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.6	40 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.7	40 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.8	40 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
0.9	40 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220
1.0	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220	20 220

Tabla C.3: Función F3. Para cada P_m y P_c se da el rango de número de individuos que cumplen $E(||x - x_o||) < \epsilon$ (1^a columna) y un factor de convergencia mayor F_c (2^a columna).

Prob. de Mutación P_m										
Prob. de Cruce P_e	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.0		0.9	0.4	0.3	0.2	0.2	0.2	0.2	0.2
		1.2	1.0	0.8	0.7	0.6	0.6	0.6	0.6
		1.8	1.5	1.2	1.1	0.8	0.9	0.9	0.9
0.1	7.6	0.5 0.5	0.4	0.2	0.2	0.2	0.2	0.2	0.2
	7.6	0.8 0.5	0.7	0.6	0.6	0.5	0.5	0.5	0.5
	7.6	0.8 0.5	1.2	0.9	0.9	0.7	0.6	0.7	0.7
0.2	1.1	0.5	0.3	0.2	0.2	0.2	0.2	0.2	0.2
	2.3	0.7	0.6	0.5	0.5	0.5	0.4	0.4	0.4
	1.1	0.9	0.8	0.8	0.9	0.7	0.6	0.6	0.6
0.3	1.0	0.3	0.3	0.2	0.2	0.1	0.2	0.2	0.2
	1.8	0.6	0.5	0.5	0.4	0.4	0.4	0.4	0.4
	1.6	0.8	0.8	0.8	0.6	0.5	0.6	0.6	0.6
0.4	2.7	0.3	0.2	0.3	0.2	0.2	0.1	0.2	0.2
	1.1	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	0.8	0.7	0.6	0.6	0.7	0.5	0.6	0.6	0.7
0.5	0.4	0.4	0.2	0.2	0.2	0.1	0.2	0.1	0.1
	0.6	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3
	0.7	0.6	0.5	0.6	0.6	0.6	0.5	0.5	0.4
0.6	0.5	0.3	0.2	0.2	0.2	0.1	0.2	0.2	0.1
	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.3
	0.6	0.6	0.6	0.6	0.5	0.6	0.6	0.6	0.5
0.7	1.6	0.3	0.2	0.2	0.1	0.1	0.1	0.1	0.1
	0.7	0.4	0.4	0.4	0.3	0.3	0.3	0.4	0.3
	0.6	0.6	0.5	0.6	0.4	0.5	0.6	0.6	0.5
0.8	0.5	0.4	0.2	0.2	0.2	0.2	0.1	0.1	0.1
	0.4	0.4	0.3	0.3	0.3	0.4	0.3	0.3	0.3
	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.6	0.5
0.9	0.1	0.4	0.1	0.1	0.2	0.1	0.1	0.1	0.1
	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	0.5	0.4	0.4	0.6	0.5	0.4	0.5	0.4	0.5
1.0	1.0	0.2	0.1	0.2	0.2	0.2	0.1	0.1	0.2
	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	0.5	0.5	0.5	0.5	0.4	0.5	0.5	0.5	0.5

Tabla C.4: Función F3. Coste computacional de los rangos mostrados en la tabla C.3 en miles de evaluaciones de la función objetivo. Se muestra también el valor medio para el rango.

		Prob. de Mutación P_m																	
		0		0.05		0.10		0.15		0.20		0.25		0.30		0.35		0.40	
Prob. de Cruce P_c	0.0					60 220	60 120	40 220	40 40	40 220		20 220	20 20	20 220		20 220		20 220	
	0.1			140 220	140 220	80 220	80 140	40 220	40 40	20 220	20 20	20 220		20 220		20 220		20 220	
	0.2			140 220	140 220	80 220	80 160	40 220	40 40	20 220	20 20	20 220		20 220		20 220		20 220	
	0.3	180 180	180 180	120 220	120 220	60 220	60 120	40 220	40 60	20 220	20 20	20 220		20 220		20 220		20 220	
	0.4	160 220	160 220	80 220	80 220	60 220	60 120	20 220	20 40	20 220	20 20	20 220		20 220		20 220		40 220	
	0.5	120 220	120 220	100 220	100 220	60 220	60 120	40 220	40 40	20 220	20 20	20 220		20 220		20 220		40 220	
	0.6	120 220	120 220	80 220	80 220	40 220	40 140	20 220	20 40	20 220		20 220		20 220		20 220		40 220	
	0.7	120 220	120 220	60 220	60 220	40 220	40 140	40 220	40 40	20 220		20 220		20 220		20 220		40 220	
	0.8	100 220	100 220	120 220	120 220	60 220	60 120	40 220	40 40	20 220		20 220		20 220		40 220		40 220	
	0.9	80 200	80 200	100 220	100 220	160 220		40 220	40 40	40 220		20 220		20 220		40 220		60 220	
	1.0	100 220	100 220	120 200	120 200	120 220	120 120	80 220		140 200		20 220		20 220		40 220		60 220	

Tabla C.5: Función F4. Para cada P_m y P_c se da el rango de número de individuos que cumplen $E(||x - x_o||) < \epsilon$ (1^a columna) y un factor de convergencia mayor F_c (2^a columna).

Prob. de Cruce P_c	Prob. de Mutación P_m																					
	0		0.05		0.10		0.15		0.20		0.25		0.30		0.35		0.40	
0.0					7.3	7.3	5.3	5.3	5.8		4.4	4.4	5.0		6.1		7.1	
					14.3	10.2	13.5	5.3	12.4		13.3	4.4	14.8		17.7		17.6	
					20.9	13.8	18.6	5.3	18.9		18.4	4.4	19.2		19.3		36.1	
0.1			9.7	9.7	7.7	7.7	7.0	7.0	3.1	3.1	5.7		7.3		5.6		6.7	
			11.1	11.1	30.1	12.7	41.0	7.0	39.6	3.1	42.7		44.7		47.1		36.3	
			11.4	11.4	54.5	17.3	80.8	7.0	68.2	3.1	71.0		87.8		88.6		64.1	
0.2			7.6	7.6	6.1	6.1	4.7	4.7	4.2	4.2	7.6		7.0		5.9		5.4	
			8.4	8.4	36.6	14.8	51.4	4.7	46.0	4.2	48.4		51.0		39.6		34.8	
			9.0	9.0	85.4	20.0	109.1	4.7	106.5	4.2	97.0		94.4		67.9		74.6	
0.3		6.8	5.9	5.9	5.4	5.4	5.5	5.5	3.9	3.9	6.2		5.0		5.9		5.8	
		6.8	7.2	7.2	41.8	10.1	56.4	9.0	54.2	3.9	53.0		39.2		38.6		35.1	
		6.8	8.8	8.8	100.7	17.9	109.7	12.4	109.3	3.9	101.9		63.9		79.8		57.9	
0.4		5.9	3.8	3.8	4.9	4.9	2.5	2.5	3.7	3.7	6.4		7.7		6.6		10.0	
		7.1	6.3	6.3	41.1	10.3	55.3	3.7	56.0	3.7	41.0		33.5		32.5		34.8	
		8.0	9.0	9.0	79.2	20.9	109.6	4.9	109.3	3.7	62.4		55.9		60.5		57.2	
0.5		4.9	4.2	4.2	4.2	4.2	7.2	7.2	3.7	3.7	4.5		6.5		6.3		8.9	
		6.7	6.6	6.6	43.0	11.0	61.7	7.2	46.7	3.7	38.1		35.0		35.7		39.1	
		8.2	8.5	8.5	91.7	23.2	109.4	7.2	93.4	3.7	80.5		61.0		58.8		50.6	
0.6		4.7	4.9	4.9	3.0	3.0	2.2	2.2	4.8		4.5		6.4		6.3		12.1	
		6.3	6.4	6.4	36.9	12.4	56.6	5.4	39.0		32.5		33.8		35.0		38.7	
		8.4	9.3	9.3	91.0	24.0	109.6	8.7	59.2		63.0		60.4		73.6		81.0	
0.7		4.5	3.0	3.0	3.1	3.1	3.8	3.8	4.5		5.4		7.6		5.3		7.9	
		6.3	6.0	6.0	40.6	14.4	56.8	3.8	36.7		37.6		36.4		34.1		36.6	
		8.4	9.5	9.5	98.5	37.1	102.6	3.8	74.2		83.5		57.8		63.7		64.0	
0.8		3.8	5.2	5.2	3.5	3.5	10.1	10.1	6.9		7.7		5.2		11.6		9.6	
		6.1	7.3	7.3	43.2	11.7	47.9	10.1	38.4		34.7		35.0		36.4		36.3	
		8.4	9.3	9.3	97.4	23.1	79.8	10.1	61.4		78.2		74.1		64.8		49.2	
0.9		3.4	4.2	4.2	63.8		8.3	8.3	14.1		7.2		5.3		12.5		19.7	
		5.6	7.1	7.1	81.9		40.2	8.3	40.2		32.9		36.1		39.0		41.5	
		7.7	9.9	9.9	101.4		67.1	8.3	66.7		51.3		72.4		52.5		61.4	
1.0		4.2	5.4	5.4	36.7	36.7	21.5		47.7		5.6		6.2		15.1		17.2	
		6.3	6.7	6.7	64.5	36.7	49.1		51.6		35.3		34.0		37.4		39.0	
		8.8	9.1	9.1	104.0	36.7	82.1		59.8		74.3		58.4		54.8		60.8	

Tabla C.6: Función F4. Coste computacional de los rangos mostrados en la tabla C.5 en miles de evaluaciones de la función objetivo. Se muestra también el valor medio para el rango.

Prob. de Cruce P_c		Prob. de Mutación P_m: 0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
	0.0			120 120 220 160	80 220	40 220	40 220	40 220	20 220	40 220
	0.1		220 220 220 220	120 120 220 180	100 220	40 220	40 220	40 220	20 220	20 220
	0.2		140 140 220 220	120 120 220 160	80 220	40 220	40 220	20 220	20 220	20 220
	0.3	220 220 220 220	100 100 220 220	80 80 220 160	60 220	40 220	40 220	20 220	20 220	20 220
	0.4	120 120 220 220	80 80 220 220	60 60 220 160	60 220	40 220	20 220	40 220	20 220	20 220
	0.5	120 120 220 220	80 80 220 220	80 80 220 180	80 220	60 220	60 220	20 220	40 220	40 220
	0.6	120 120 220 220	80 80 220 220	80 80 220 160	100 220	60 220	20 220	40 220	80 220	100 220
	0.7	120 120 220 220	60 60 220 220	60 60 220 200	100 220	120 220	120 220	60 220	40 220	100 220
	0.8	60 60 220 220	160 160 220 220	80 80 220 140	120 220	80 220	80 220	60 220	60 220	80 220
	0.9	40 40 220 220	80 80 220 220	80 80 220 140	100 220	100 220	40 220	60 220	100 220	60 220
	1.0	80 80 220 220	80 80 220 220	100 100 220 140	140 220	60 220	40 220	20 220	60 220	60 220

Tabla C.7: Función F5. Para cada P_m y P_c se da el rango de número de individuos que cumplen $E(||x - x_o||) < \epsilon$ (1^a columna) y un factor de convergencia mayor F_c (2^a columna).

Prob. de Cruce P_c	Prob. de Mutación P_m 0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.0			14.3 14.3	13.9	7.8	6.7	8.0	6.4	8.2
			17.9 15.4	16.8	15.2	15.5	16.2	15.0	19.5
			22.4 17.0	27.7	19.2	17.6	18.0	20.1	25.1
0.1		12.2 12.2	13.0 13.0	27.0	10.1	17.4	18.1	7.0	7.4
		12.2 12.2	26.0 19.6	39.5	35.1	34.1	43.5	46.5	39.4
		12.2 12.2	44.1 29.7	57.4	51.3	57.5	83.3	75.2	58.5
0.2		6.5 6.5	11.6 11.6	18.8	10.0	12.4	6.6	7.7	8.1
		7.8 7.8	16.5 16.4	26.8	22.5	27.0	31.2	39.4	34.1
		8.6 8.6	18.3 21.3	23.1	21.3	29.3	48.3	75.6	61.6
0.3	8.4 8.4	4.2 4.2	5.3 5.3	12.7	13.7	13.8	6.5	6.3	6.6
	8.4 8.4	6.5 6.5	11.3 8.8	14.9	16.0	19.3	36.0	37.3	35.4
	8.4 8.4	8.7 8.7	14.6 10.5	17.5	20.4	25.4	70.4	70.9	70.0
0.4	4.1 4.1	3.3 3.3	3.9 3.9	7.8	9.1	7.9	11.4	6.6	4.4
	6.2 6.2	6.0 6.0	9.6 7.6	11.9	13.7	21.4	41.6	36.5	37.6
	7.3 7.3	8.2 8.2	15.0 11.4	16.9	20.0	29.5	65.3	65.5	70.5
0.5	4.0 4.0	3.1 3.1	4.6 4.6	7.9	9.0	20.6	6.7	13.6	12.1
	5.9 5.9	5.9 5.9	10.0 8.9	12.4	14.3	43.7	37.9	41.1	37.0
	7.3 7.3	8.9 8.9	14.6 12.1	16.4	20.5	70.7	75.5	75.8	58.5
0.6	4.2 4.2	3.3 3.3	4.6 4.6	7.9	9.8	6.7	13.2	25.9	23.5
	5.7 5.7	5.8 5.8	9.5 7.5	13.3	16.1	35.3	37.2	45.7	40.3
	7.4 7.4	8.3 8.3	14.4 10.7	17.0	22.0	66.2	60.9	66.8	46.4
0.7	4.3 4.3	2.8 2.8	3.3 3.3	8.4	26.6	29.9	16.8	9.8	27.2
	5.8 5.8	5.4 5.4	8.9 8.2	12.9	25.6	49.1	44.3	38.0	47.7
	7.8 7.8	8.4 8.4	14.1 12.8	17.1	27.7	62.6	67.9	61.2	57.5
0.8	2.2 2.2	6.1 6.1	4.8 4.8	9.5	25.4	23.1	14.2	16.3	20.3
	4.7 4.7	7.3 7.3	9.7 7.0	15.7	38.2	46.0	40.5	41.1	42.6
	7.3 7.3	8.5 8.5	13.9 8.9	17.2	48.2	81.2	72.0	51.3	64.8
0.9	1.5 1.5	3.1 3.1	4.6 4.6	8.7	31.2	11.7	14.1	27.9	15.8
	4.4 4.4	5.6 5.6	9.7 6.8	13.3	41.4	35.0	39.3	43.7	39.6
	7.4 7.4	8.3 8.3	14.5 8.9	17.7	54.2	64.0	51.7	73.9	66.1
1.0	2.7 2.7	3.2 3.2	6.4 6.4	13.7	12.7	9.2	6.8	16.3	16.7
	5.0 5.0	5.8 5.8	10.6 8.0	16.7	42.0	37.7	33.0	38.1	41.8
	7.2 7.2	8.7 8.7	14.3 9.9	18.7	67.7	63.4	47.4	50.9	70.6

Tabla C.8: Función F5. Coste computacional de los rangos mostrados en la tabla C.7 en miles de evaluaciones de la función objetivo. Se muestra también el valor medio para el rango.

Prob. de Cruce P_c	Prob. de Mutación P_m: 0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.0			120 220	80 220	60 220	40 220	20 220	40 220	40 220
0.1		180 180 200 200	80 80 220 80	80 220	40 220	40 220	40 220	40 220	40 220
0.2		140 140 200 200	80 80 220 80	60 220	40 220	40 220	40 220	40 220	40 220
0.3		120 120 220 220	100 220	80 220	40 220	40 220	40 220	40 220	60 220
0.4	160 160 220 220	140 140 220 220	140 220	60 220	80 220	60 220	40 220	60 220	80 220
0.5	180 180 180 180	120 120 220 220	100 100 220 100	120 200	100 220	100 220	140 220	80 220	220 220
0.6									
0.7									
0.8									
0.9									
1.0									

Tabla C.9: Función F6. Para cada P_m y P_c se da el rango de número de individuos que cumplen $E(||x - x_o||) < \epsilon$ (1^a columna) y un factor de convergencia mayor F_c (2^a columna).

Prob. de Cruce P_c	Prob. de Mutación P_m: 0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.0			8.6 14.2 15.1	9.0 11.7 14.3	7.8 9.3 13.1	5.9 8.7 13.3	4.2 8.4 12.6	4.4 8.4 12.4	5.3 9.1 12.2
0.1		9.4 9.4 9.5 9.5 9.5 9.5	6.9 6.9 8.7 6.9 9.5 6.9	6.9 7.5 10.1	7.1 7.7 9.3	5.0 6.8 9.6	4.0 7.2 9.8	3.3 7.6 10.5	4.3 7.7 11.7
0.2		7.5 7.5 7.9 7.9 7.8 7.8	5.4 5.4 7.3 5.4 9.3 5.4	4.3 6.7 8.5	5.1 6.5 8.8	4.2 6.4 8.5	5.1 6.6 9.5	3.4 6.9 10.0	4.8 7.8 11.9
0.3		5.1 5.1 6.7 6.7 7.7 7.7	7.0 7.1 7.6	5.1 6.6 7.8	4.7 6.3 8.6	6.4 6.7 8.8	6.7 6.8 9.4	5.8 7.3 10.1	6.2 8.5 11.4
0.4	5.7 5.7 6.9 6.9 7.5 7.5	4.9 4.9 6.1 6.1 6.8 6.8	5.2 7.1 10.4	5.4 7.9 8.2	8.9 7.2 7.9	5.7 7.0 9.7	7.3 8.3 10.2	6.9 8.9 10.3	6.4 10.2 12.9
0.5	5.7 5.7 5.7 5.7 5.7 5.7	5.2 5.2 6.9 6.9 8.3 8.3	4.9 4.9 10.3 4.9 9.6 4.9	14.6 10.1 11.2	10.8 12.8 10.1	15.5 14.6 18.7	9.4 12.0 10.5	7.4 13.8 27.6	21.6 21.6 21.6
0.6									
0.7									
0.8									
0.9									
1.0									

Tabla C.10: Función F6. Coste computacional de los rangos mostrados en la tabla C.9 en miles de evaluaciones de la función objetivo. Se muestra también el valor medio para el rango.

Prob. de Mutación P_m																					
Prob. de Cruce P_c	0		0.05		0.10		0.15		0.20		0.25		0.30		0.35		0.40				
0.0																					
0.1																					
0.2																					
0.3													220		80						
													220		80						
0.4											220				200		140				
											220				200		220				
0.5											160		180		80		60				
											220		220		220		220				
0.6					220				180		160		60		100		140				
					220				200		220		220		220		220				
0.7			200	200	200		100		140		80		80		100		120				
			200	200	200		220		220		220		220		220		220				
0.8			220	220	180		140		60		60		100		120		80				
			220	220	220		220		220		220		220		220		220				
0.9	180	180	160	160	180		80		80		80		80		80		60				
	220	220	220	220	220		220		220		220		220		220		220				
1.0	160	160	160	160	100		60		40		60		100		120		120				
	220	220	220	220	220		220		220		220		220		220		220				

Tabla C.11: Función F7. Para cada P_m y P_c se da el rango de número de individuos que cumplen $E(||x - x_o||) < 1$ (1^a columna) y un factor de convergencia mayor F_c (2^a columna).

Prob. de Cruce P_c	Prob. de Mutación P_m									
	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.0									
0.1									
0.2									
0.3							59.2	28.5	
							59.2	28.5	
							59.2	28.5	
0.4						43.6		39.3	41.5
						43.6		39.3	52.4
						43.6		39.3	71.4
0.5						33.3	43.6	20.0	14.9
						41.4	48.6	36.7	38.4
						46.1	48.4	33.7	59.8
0.6			25.6		23.8	43.1	19.9	25.1	42.1
			25.6		35.5	50.4	37.0	37.7	48.2
			25.6		47.2	51.1	49.5	51.3	64.2
0.7		12.1 12.1	28.0	10.3	41.6	24.4	24.4	29.9	35.6
		12.1 12.1	28.0	19.5	49.1	43.3	38.1	42.7	50.6
		12.1 12.1	28.0	32.3	71.2	62.0	52.3	68.0	75.6
0.8		12.5 12.5	24.8	31.7	12.3	19.1	22.3	31.5	20.9
		12.5 12.5	21.4	32.5	36.8	38.1	42.5	47.4	33.4
		12.5 12.5	22.5	43.8	67.7	64.0	60.1	77.9	44.0
0.9	9.2 9.2	10.9 10.9	18.9	12.0	24.8	16.4	26.8	24.7	21.0
	10.1 10.1	11.3 11.3	23.3	36.6	38.6	39.4	43.5	35.9	35.7
	11.0 11.0	12.0 12.0	28.8	66.0	50.2	53.5	73.5	48.9	46.2
1.0	10.0 10.0	13.4 13.4	13.8	14.0	8.1	16.2	29.2	29.6	32.1
	11.1 11.1	14.9 14.9	24.8	35.7	31.8	36.8	44.6	41.3	48.2
	12.8 12.8	15.8 15.8	39.3	71.0	46.3	69.6	54.8	51.3	50.4

Tabla C.12: Función F7. Coste computacional de los rangos mostrados en la tabla C.11 en miles de evaluaciones de la función objetivo. Se muestra también el valor medio para el rango.

Apéndice D

Resultados del control con no linealidades en los actuadores

Saturación en la acción de control

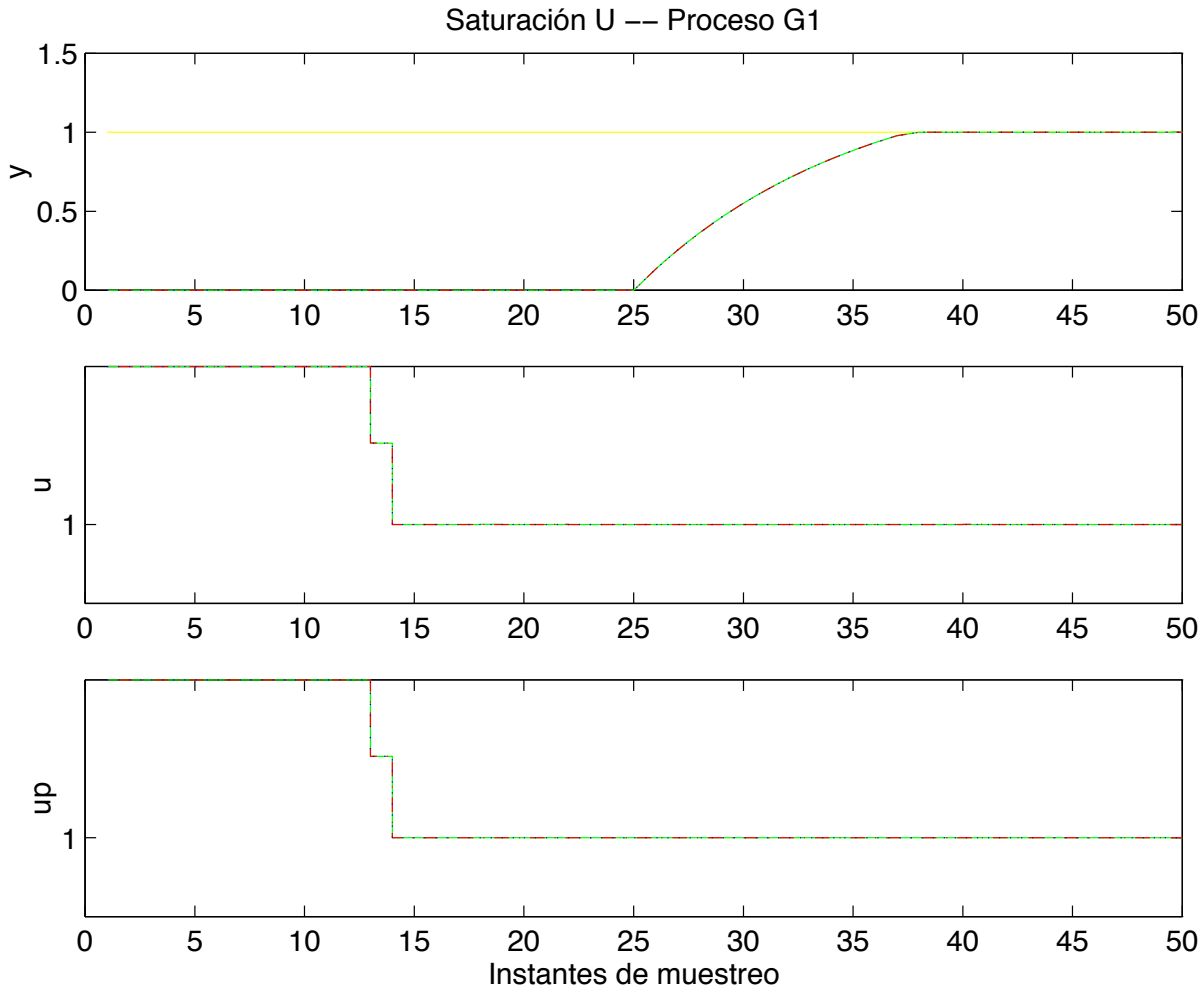


Figura D.1: Control del proceso $G1$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

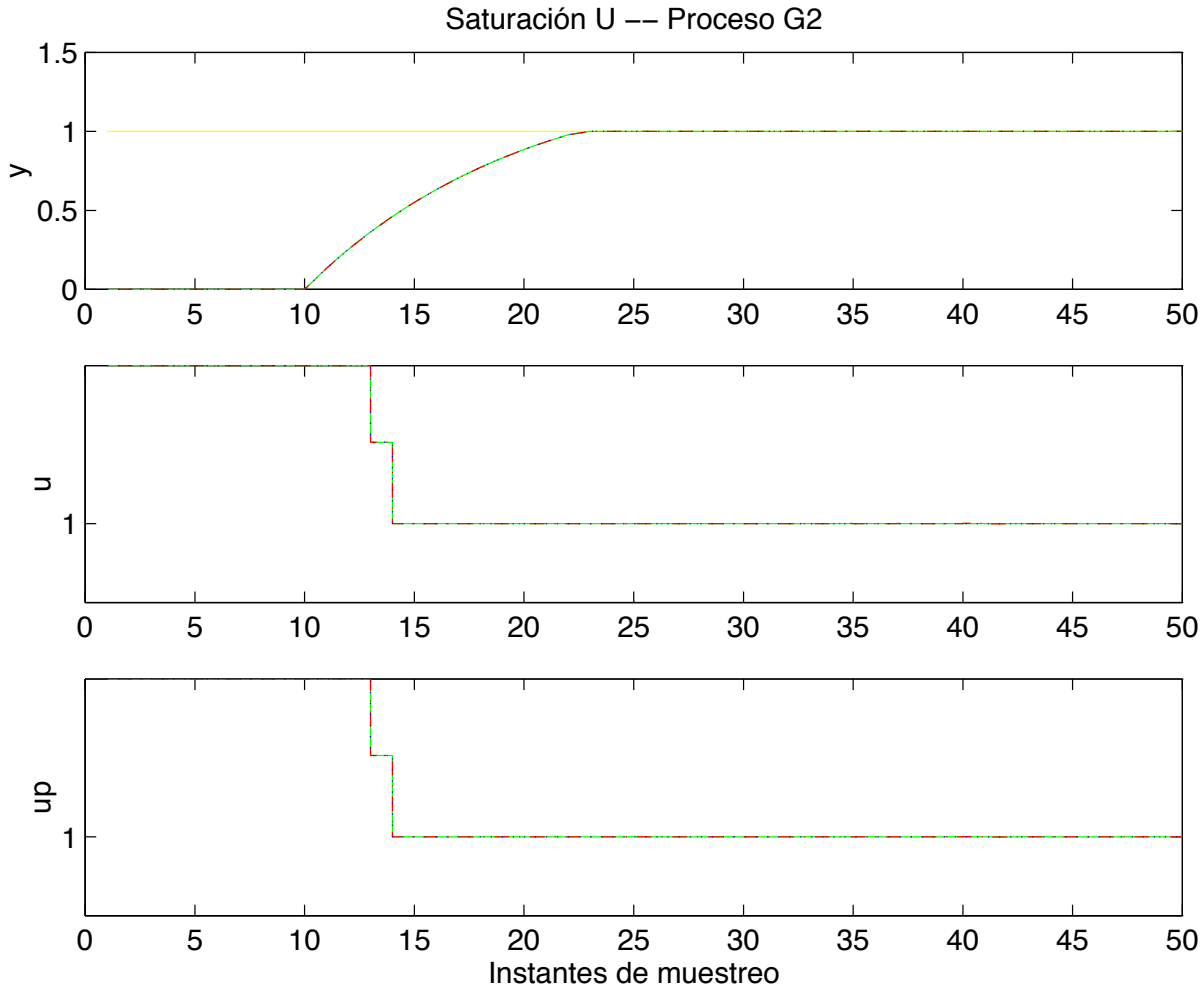


Figura D.2: Control del proceso $G2$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

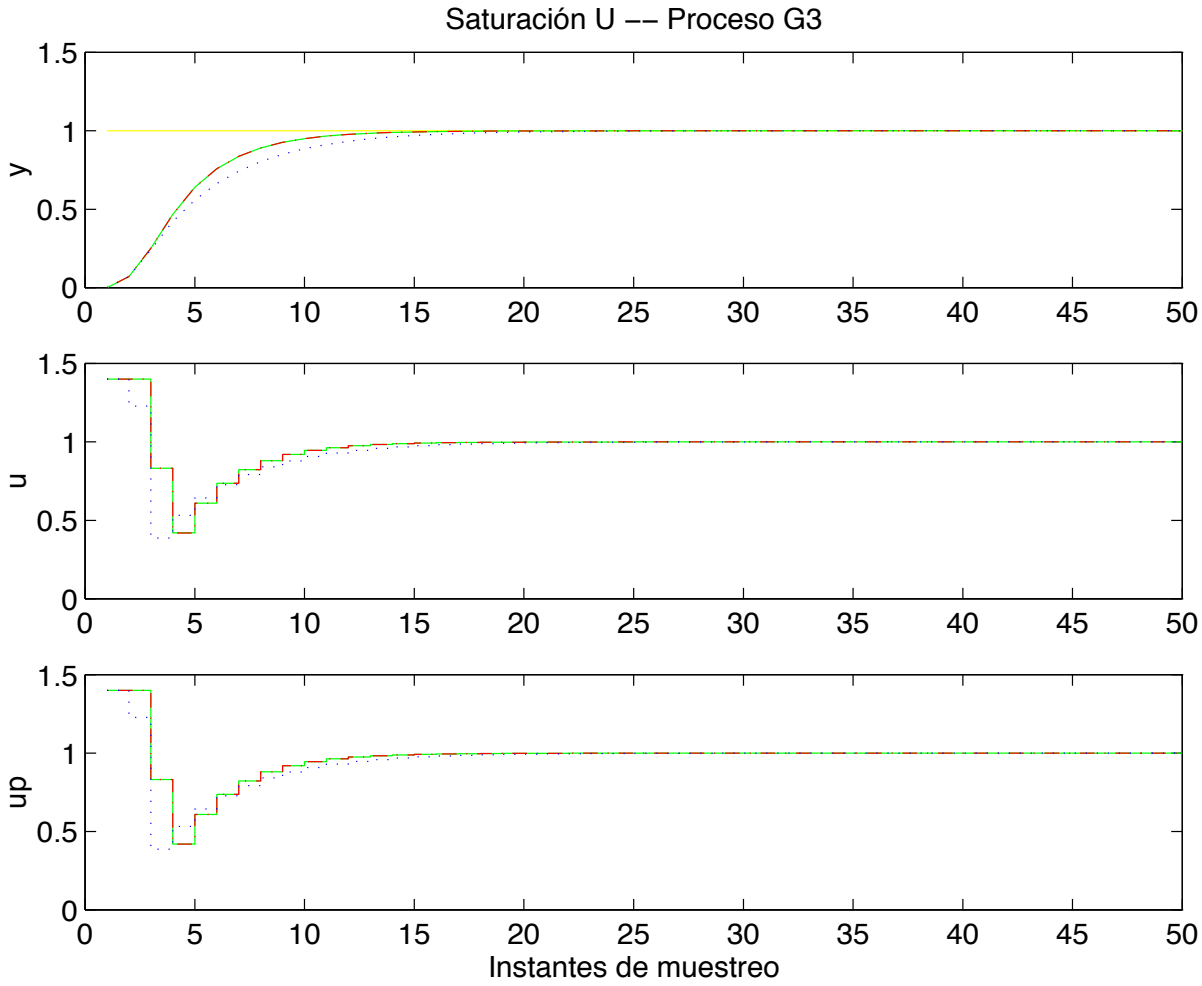


Figura D.3: Control del proceso $G3$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

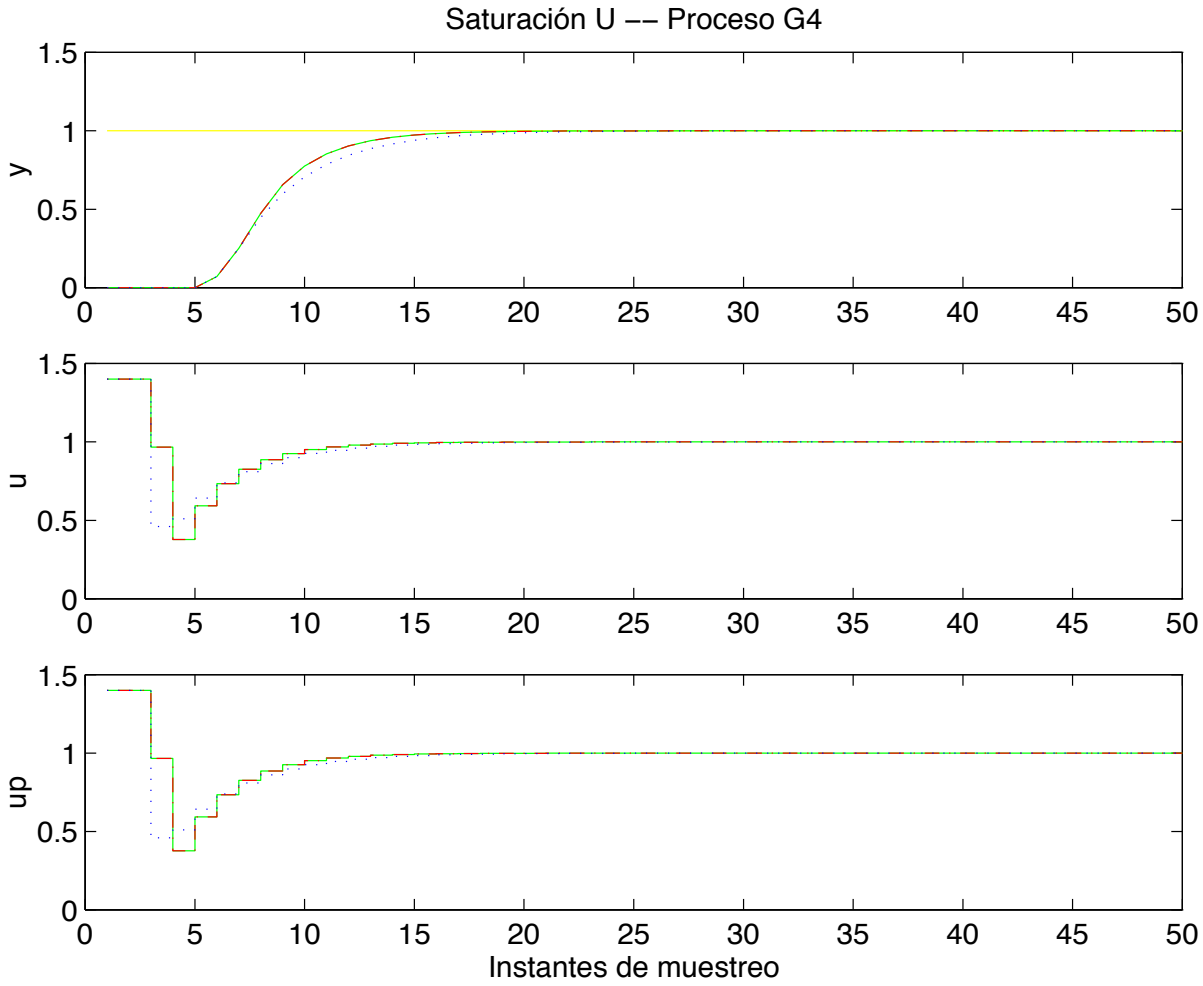


Figura D.4: Control del proceso $G4$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

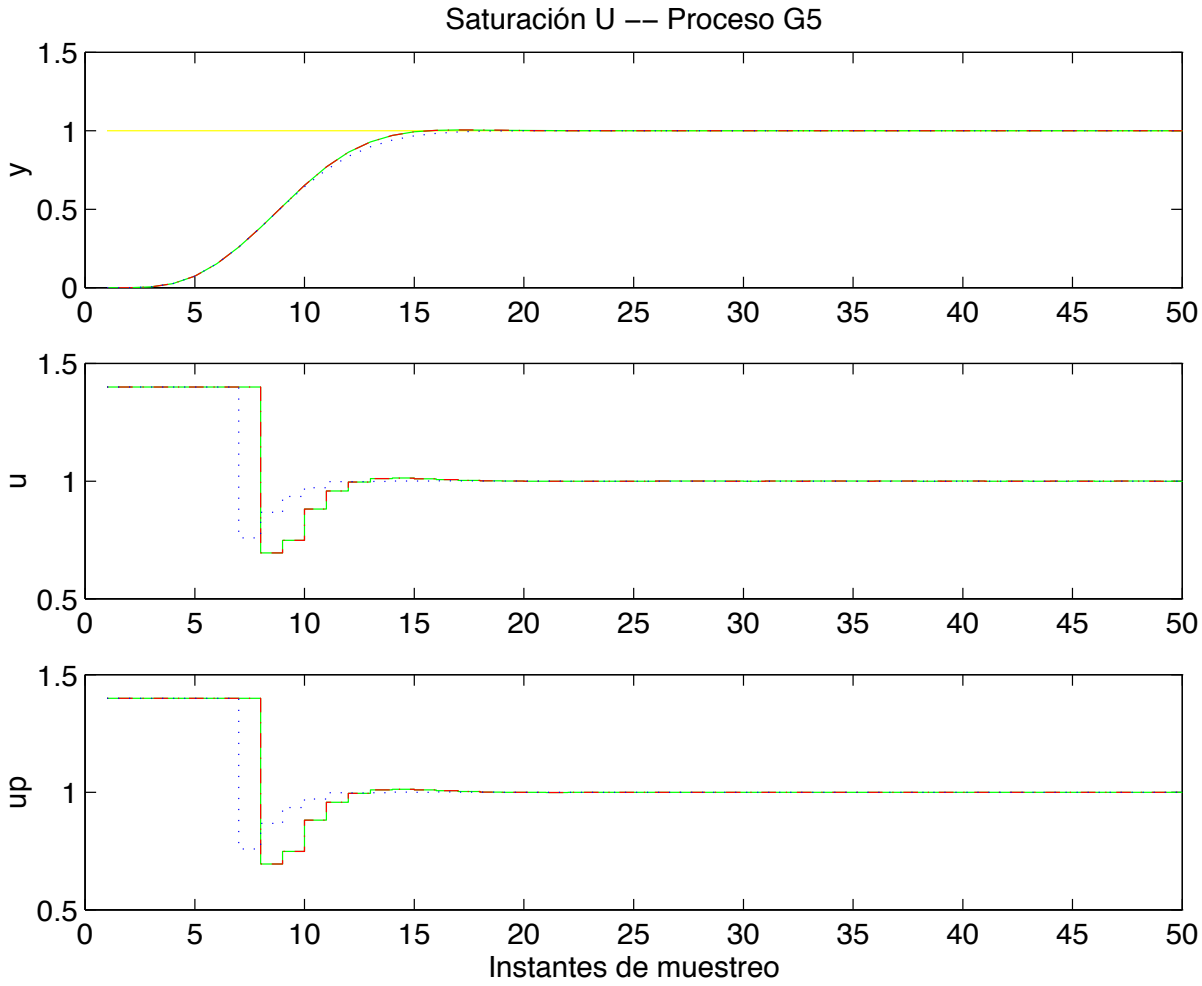


Figura D.5: Control del proceso $G5$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

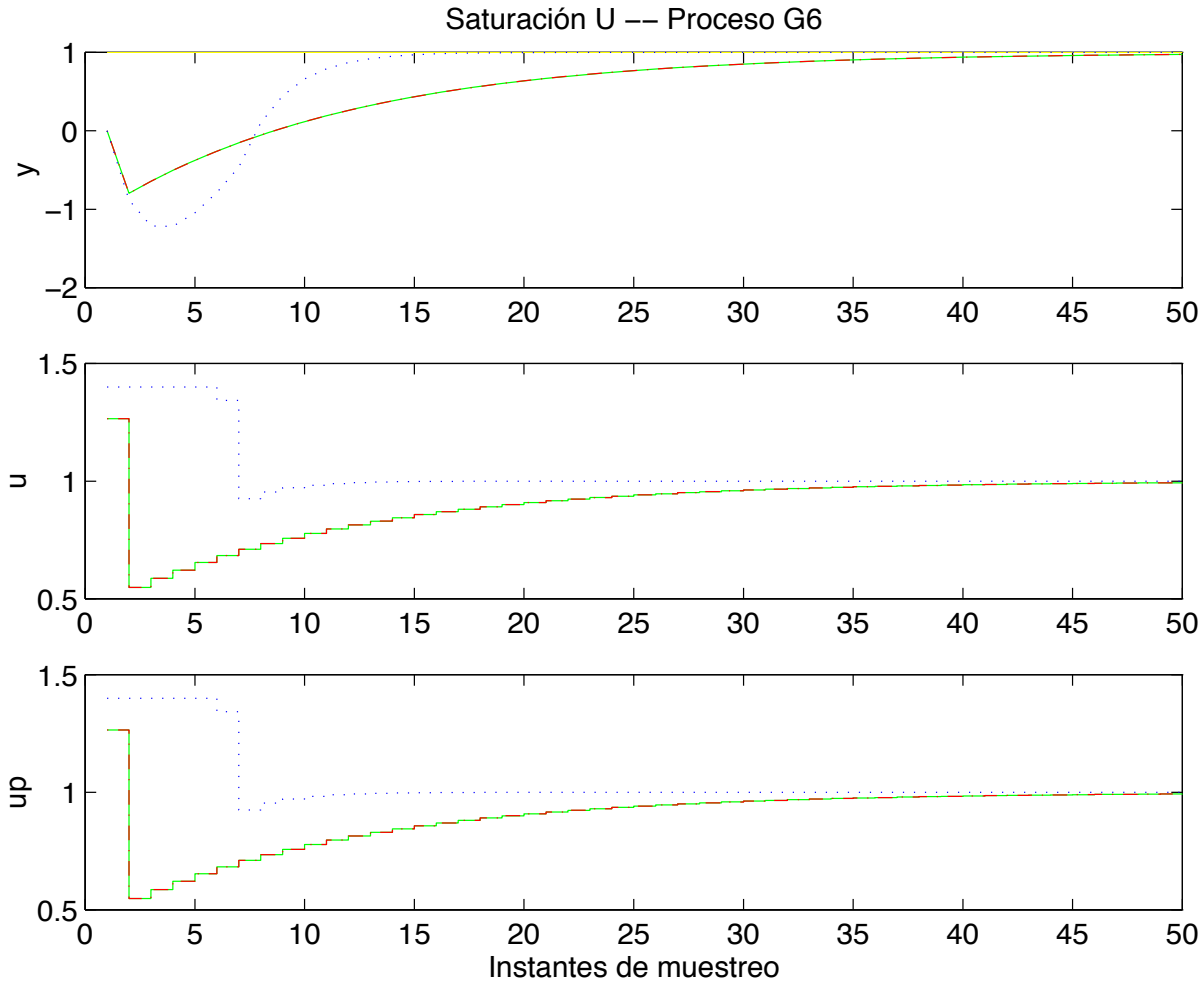


Figura D.6: Control del proceso $G6$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

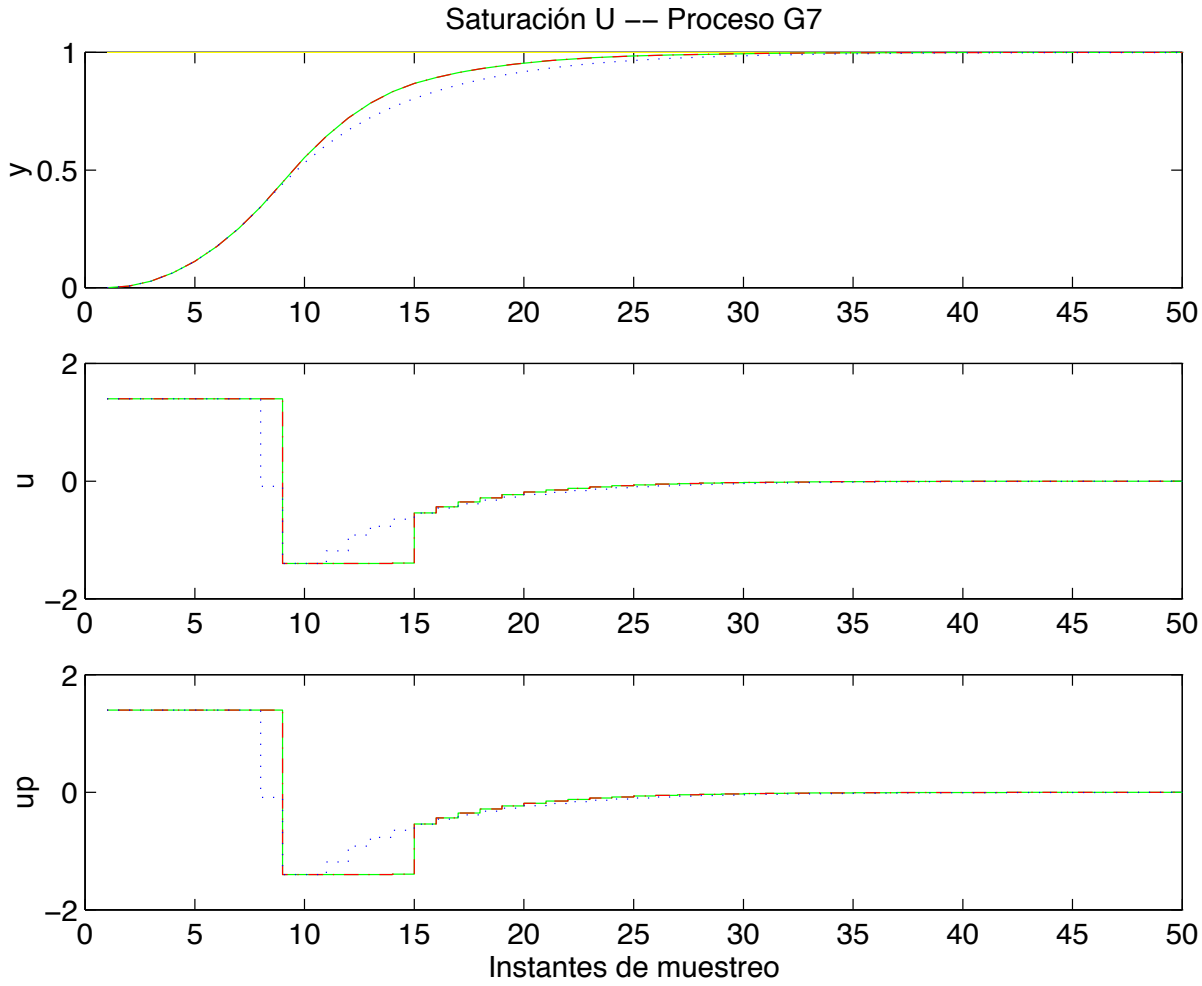


Figura D.7: Control del proceso $G7$ con saturación en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Saturación del incremento de la acción de control

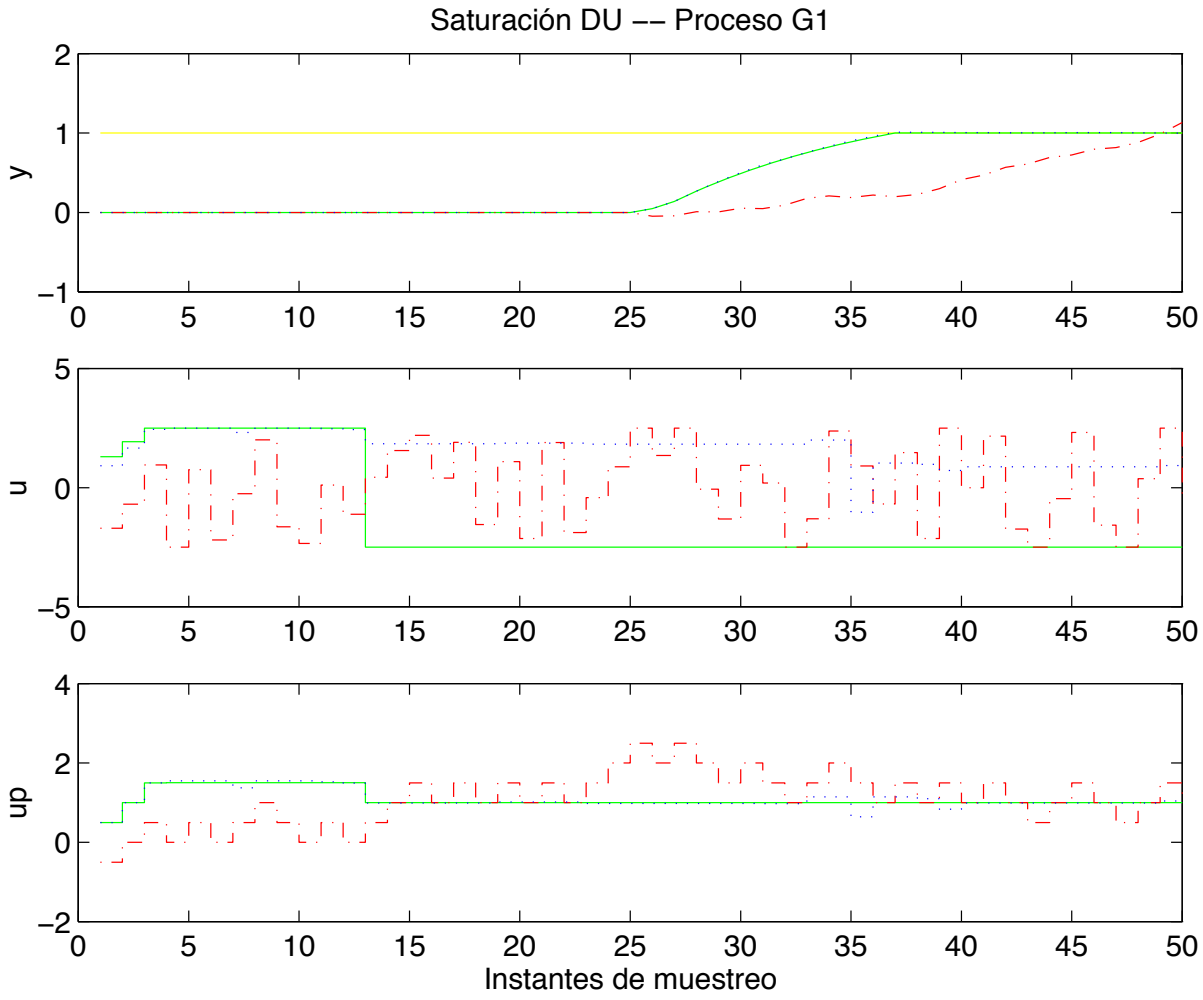


Figura D.8: Control del proceso $G1$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

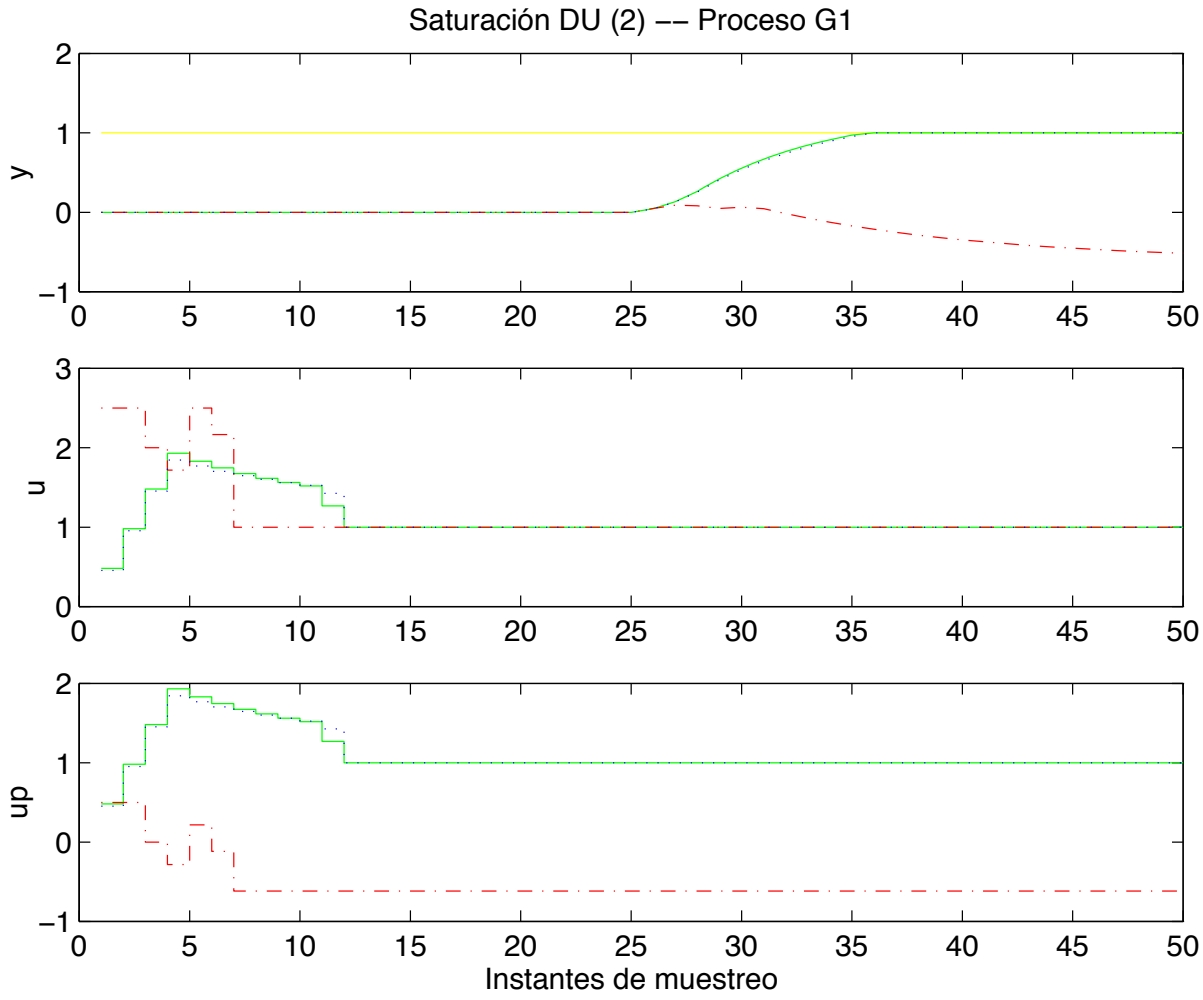


Figura D.9: Control del proceso $G1$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

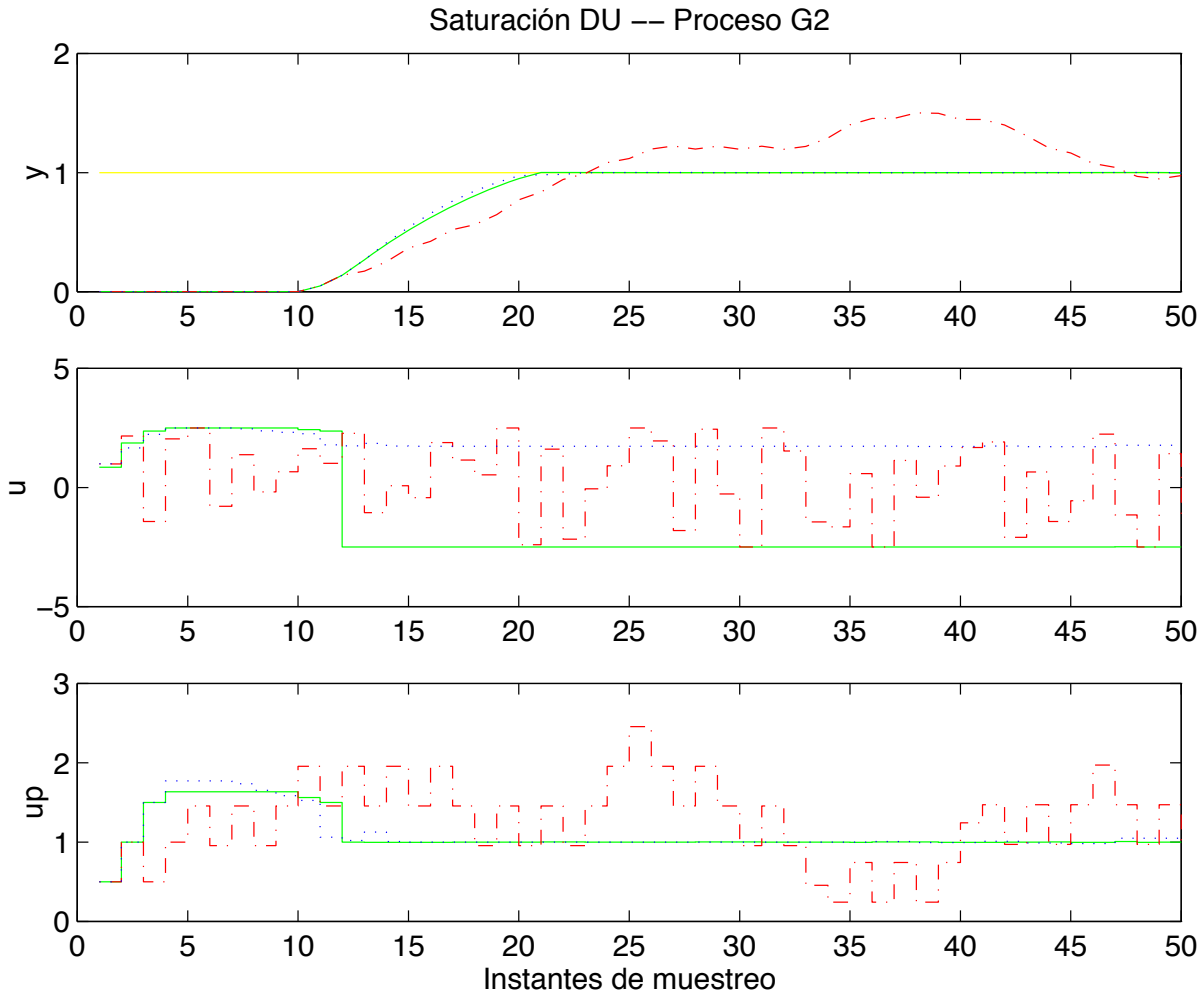


Figura D.10: Control del proceso $G2$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

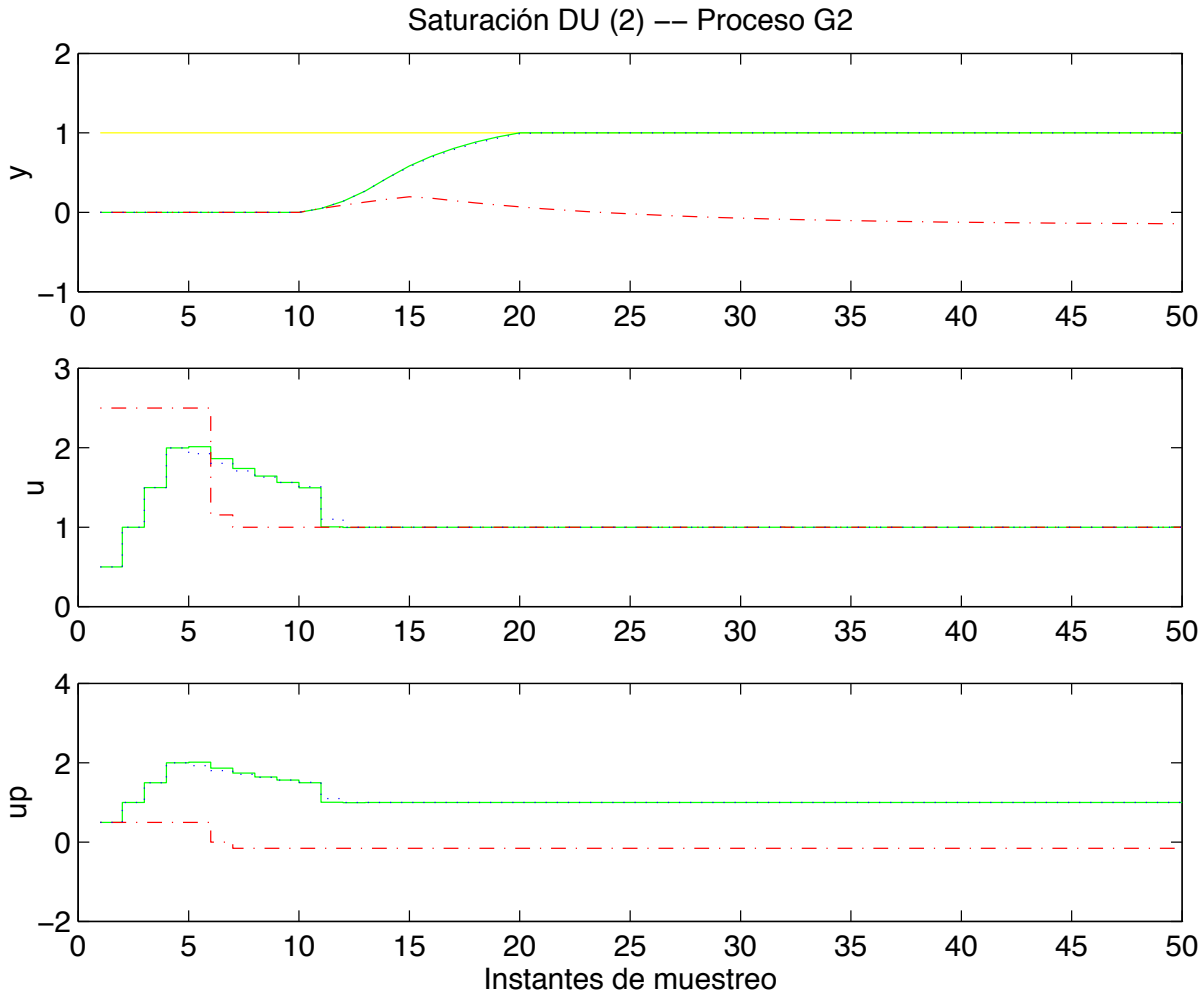


Figura D.11: Control del proceso $G2$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

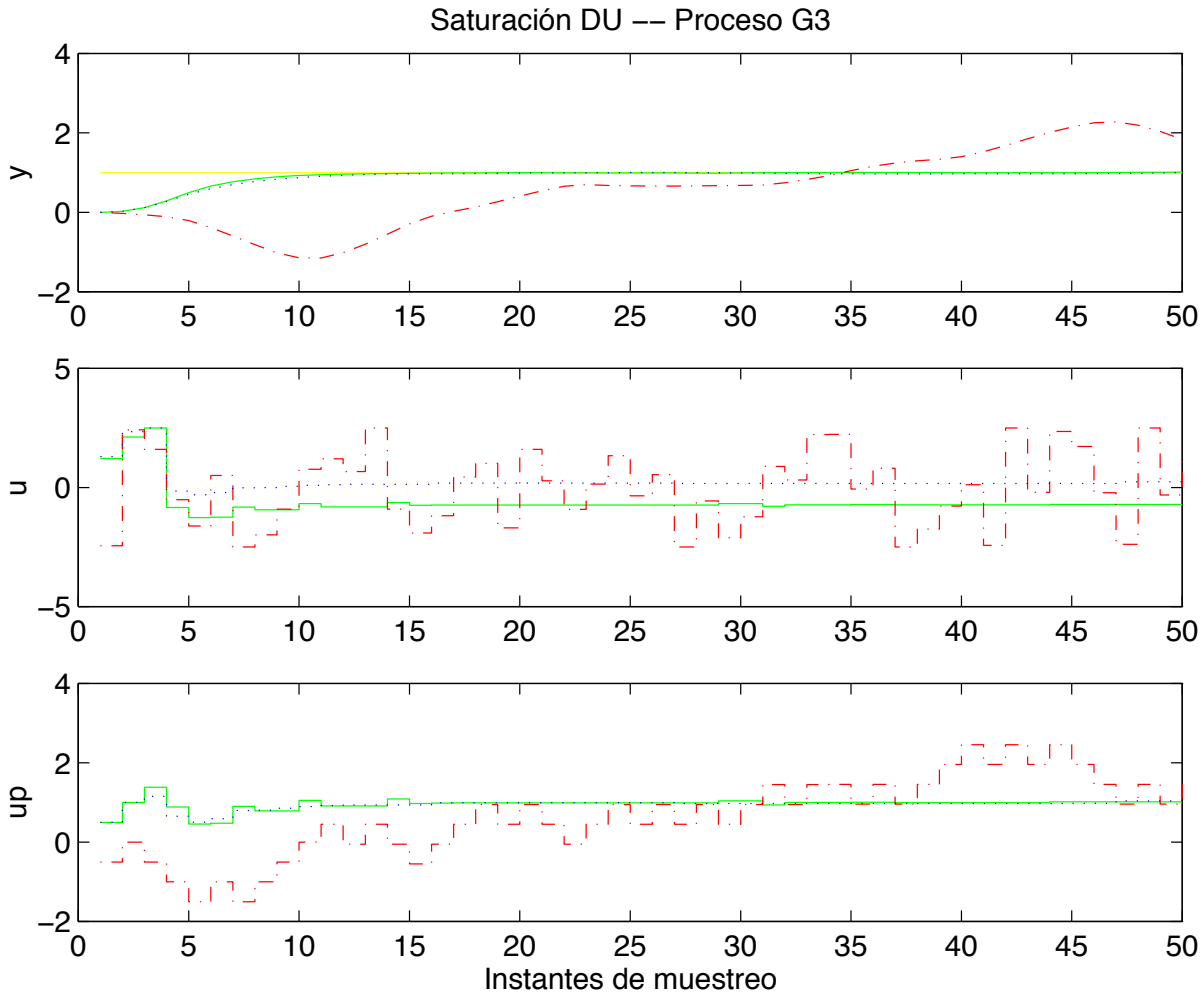


Figura D.12: Control del proceso $G3$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

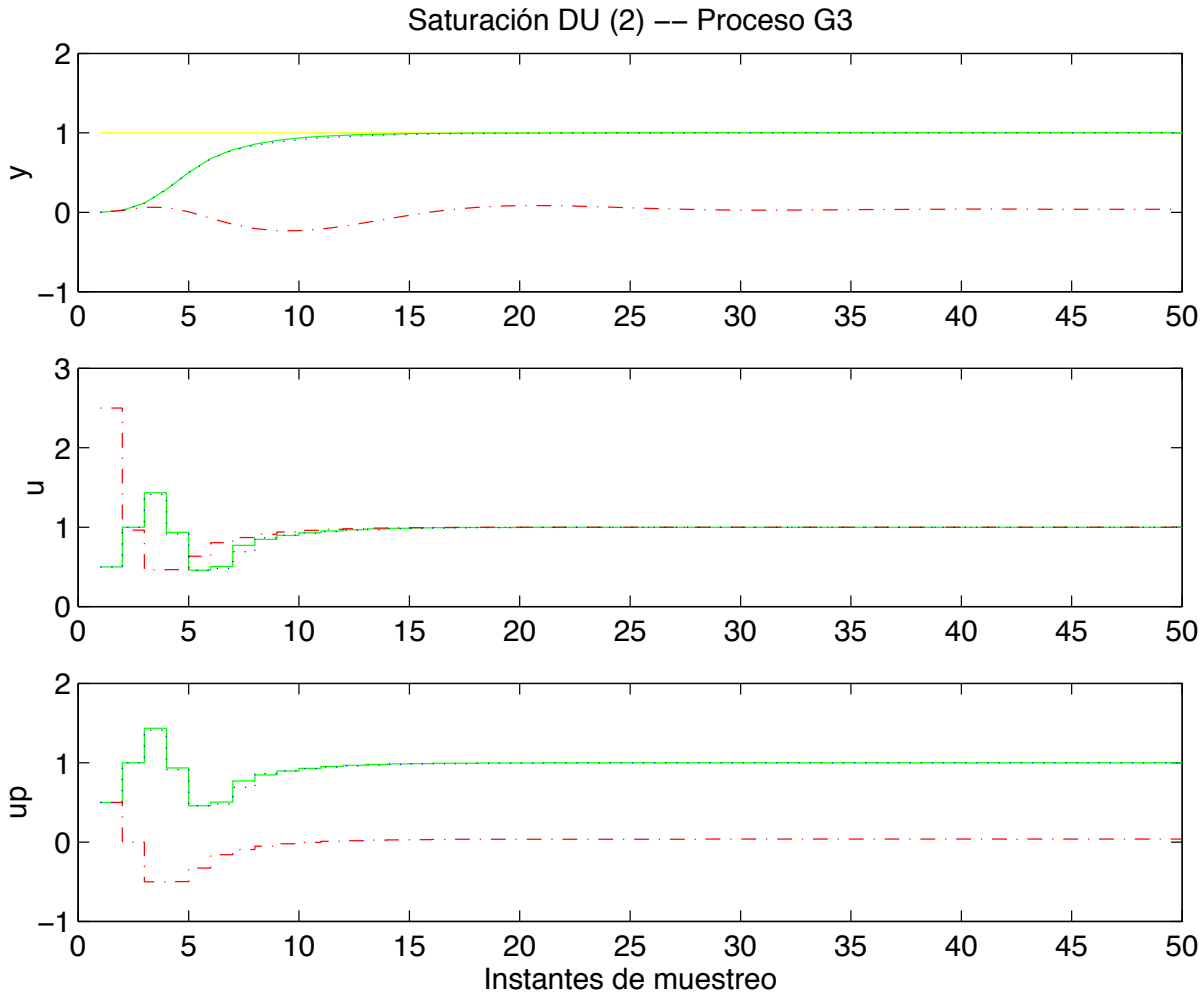


Figura D.13: Control del proceso $G3$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

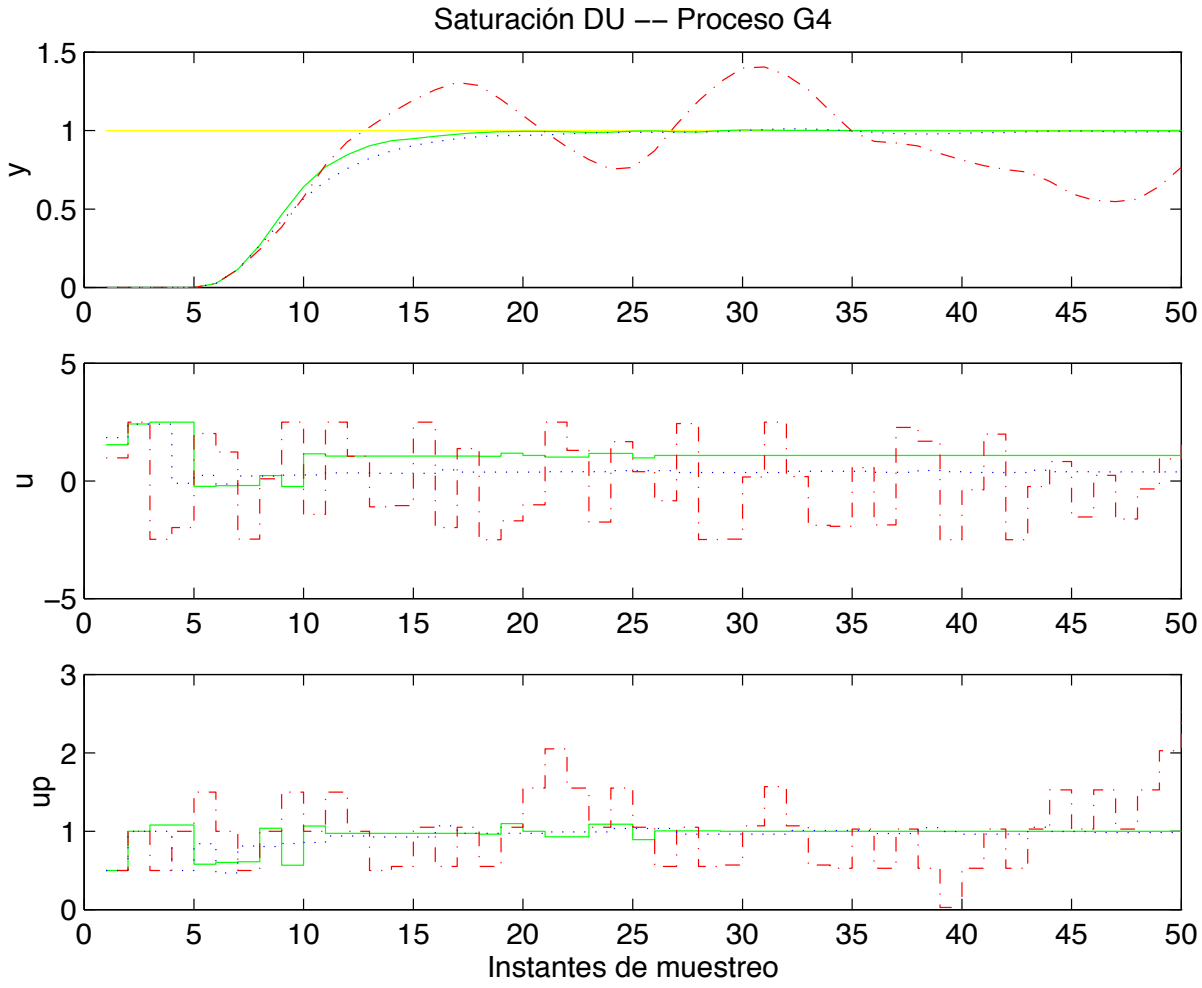


Figura D.14: Control del proceso $G4$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

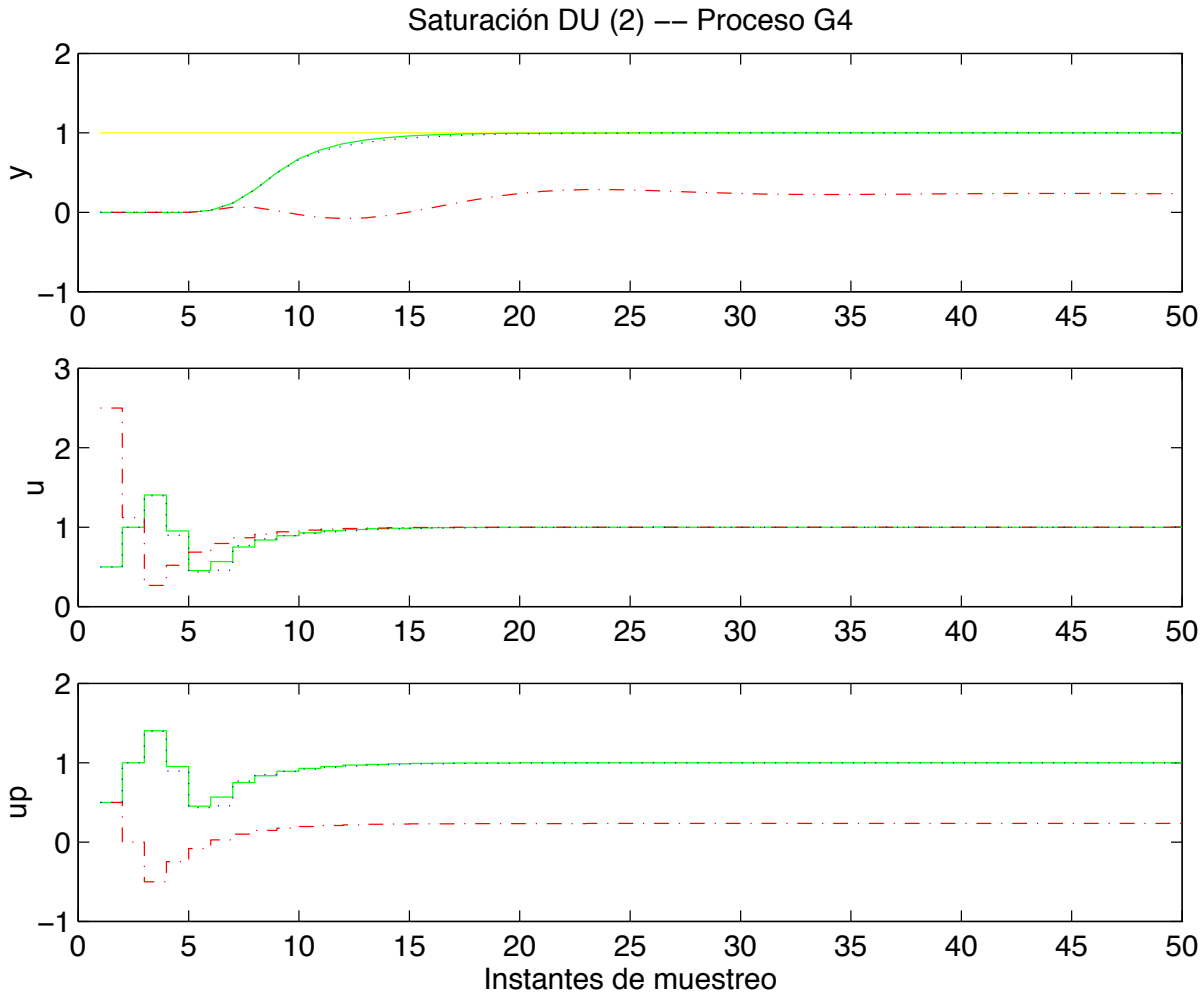


Figura D.15: Control del proceso $G4$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

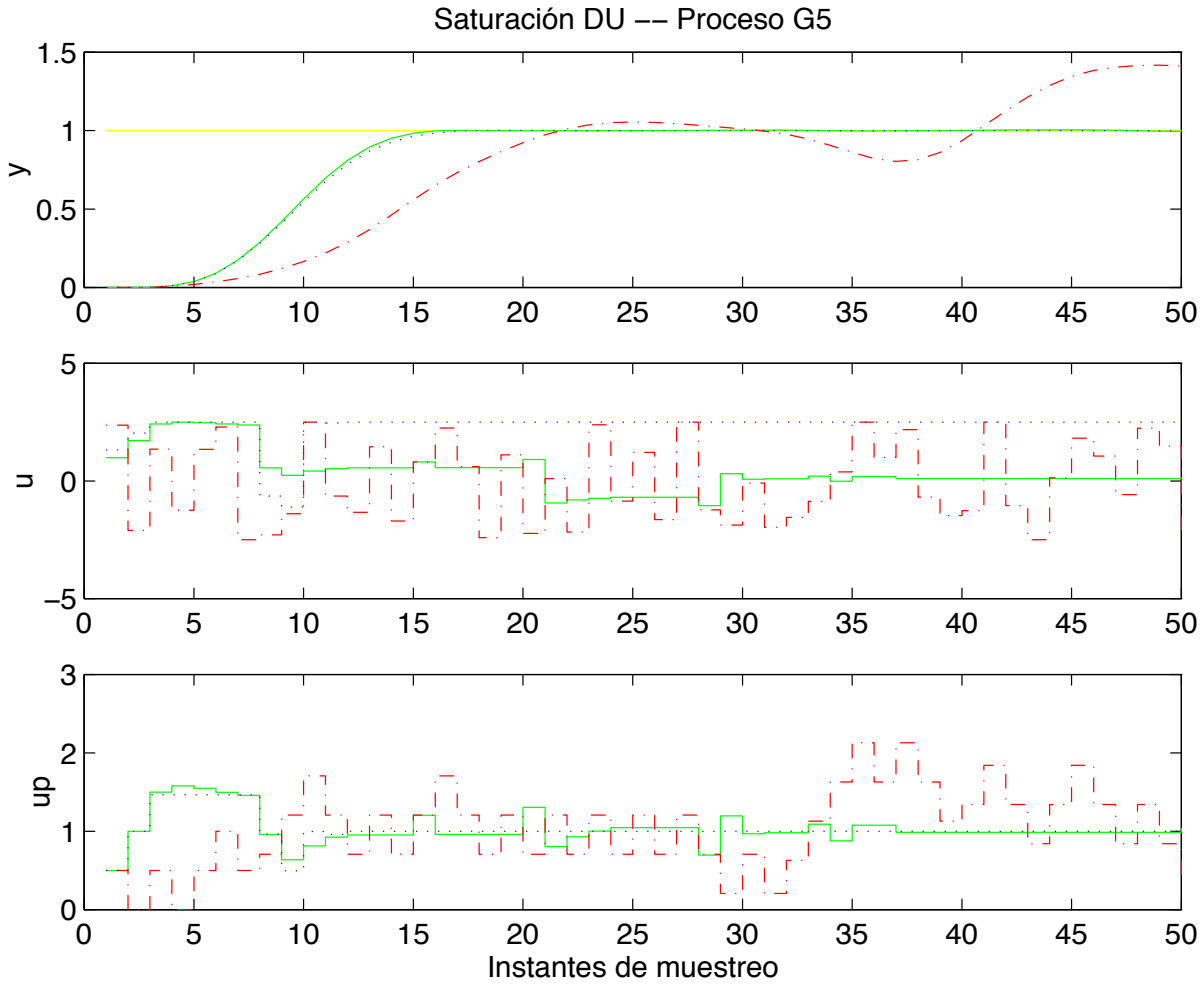


Figura D.16: Control del proceso $G5$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

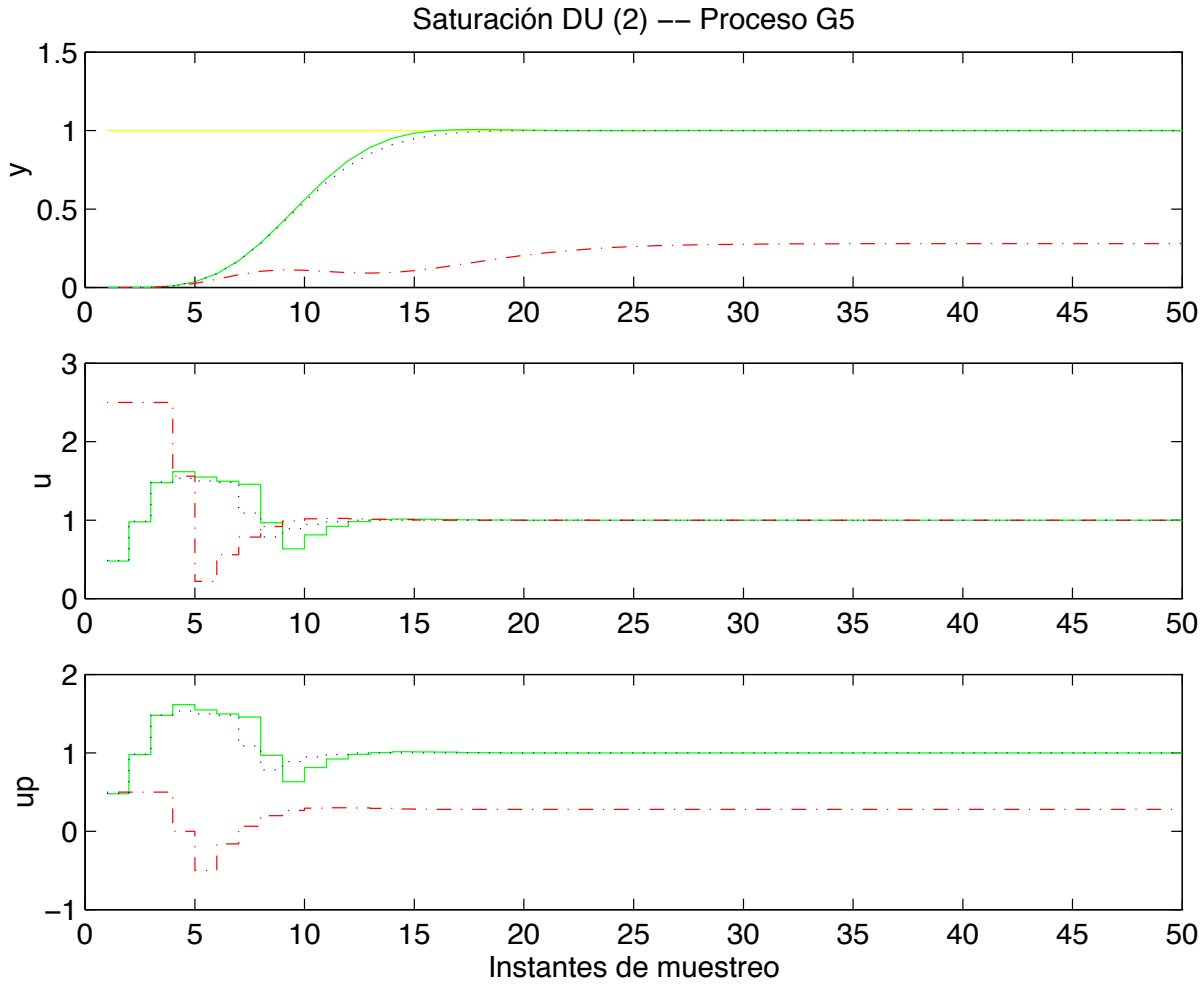


Figura D.17: Control del proceso $G5$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

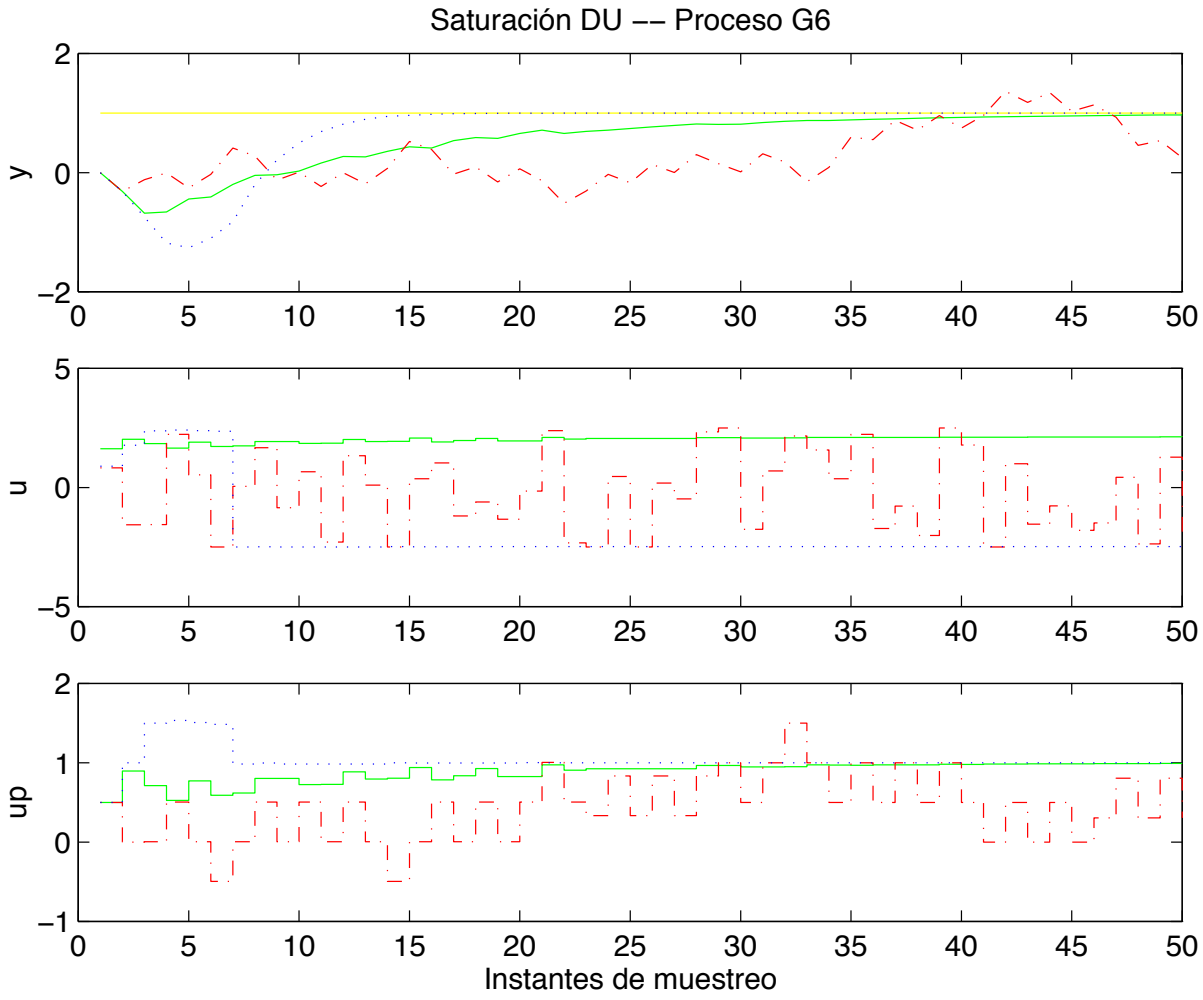


Figura D.18: Control del proceso $G6$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

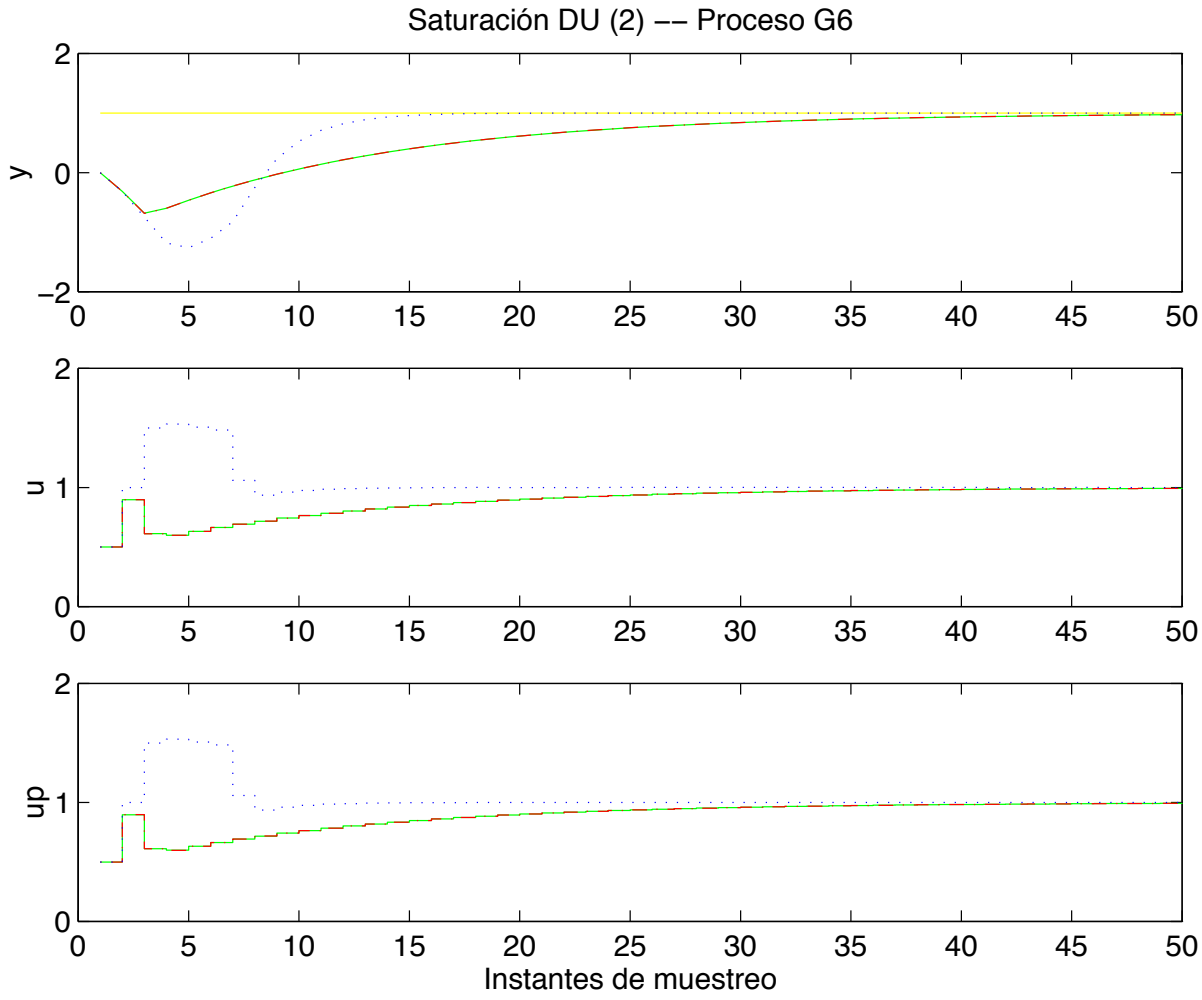


Figura D.19: Control del proceso $G6$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

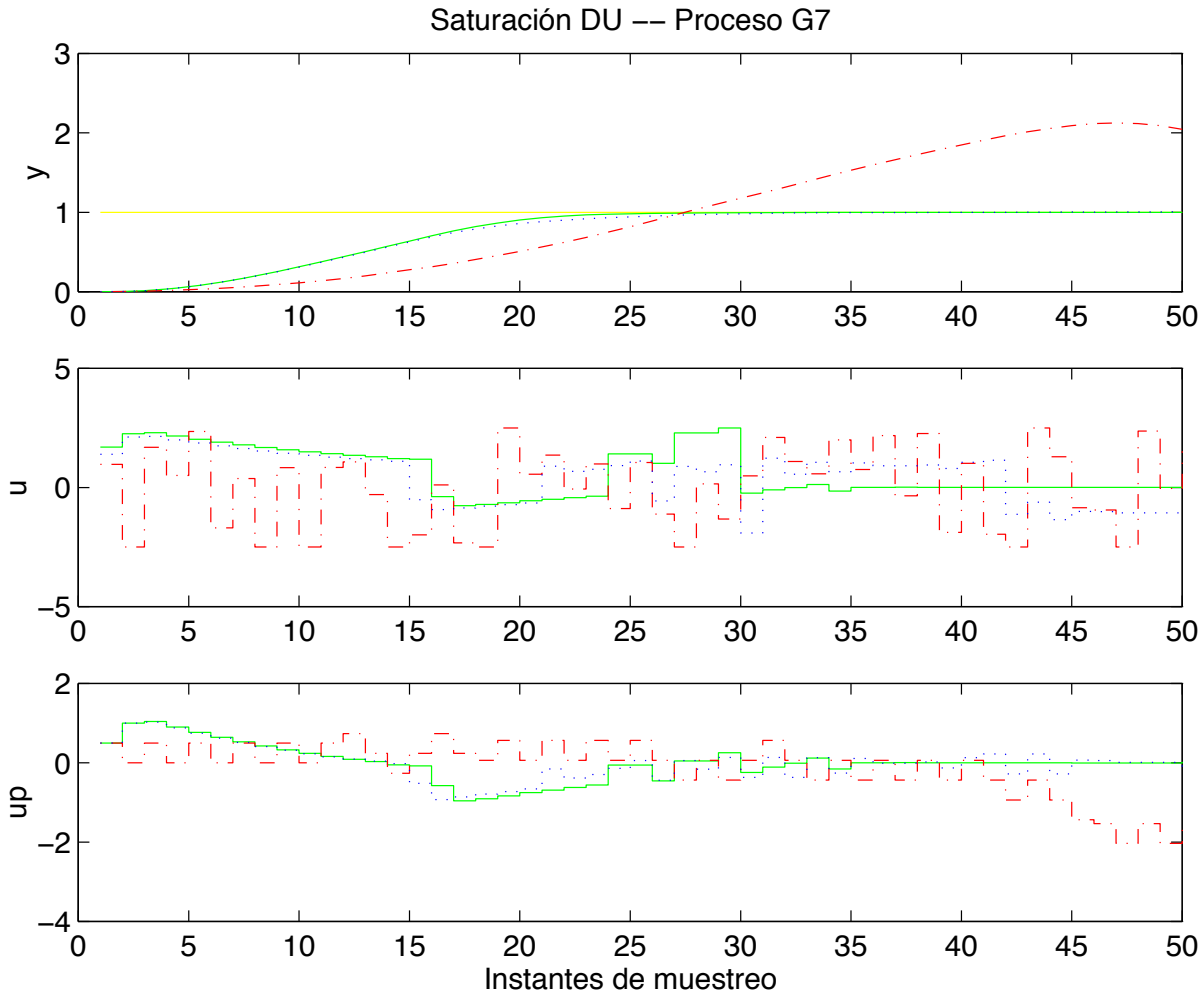


Figura D.20: Control del proceso $G7$ con saturación del incremento de la acción de control. Inclusión de la no linealidad en el modelo de predicción. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

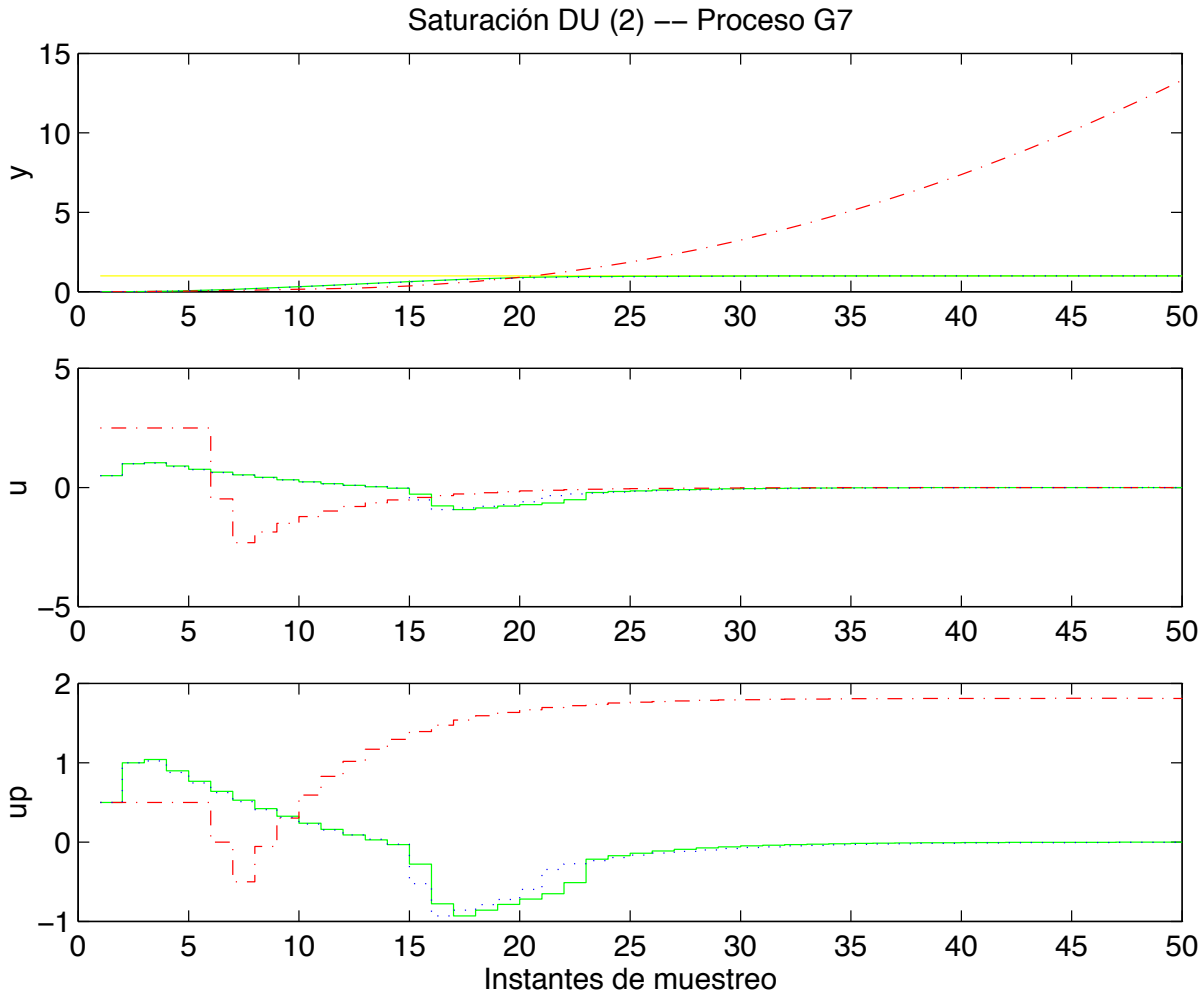


Figura D.21: Control del proceso $G7$ con saturación del incremento de la acción de control. Inclusión de la no linealidad penalizando la función de coste. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Zona Muerta de tipo I

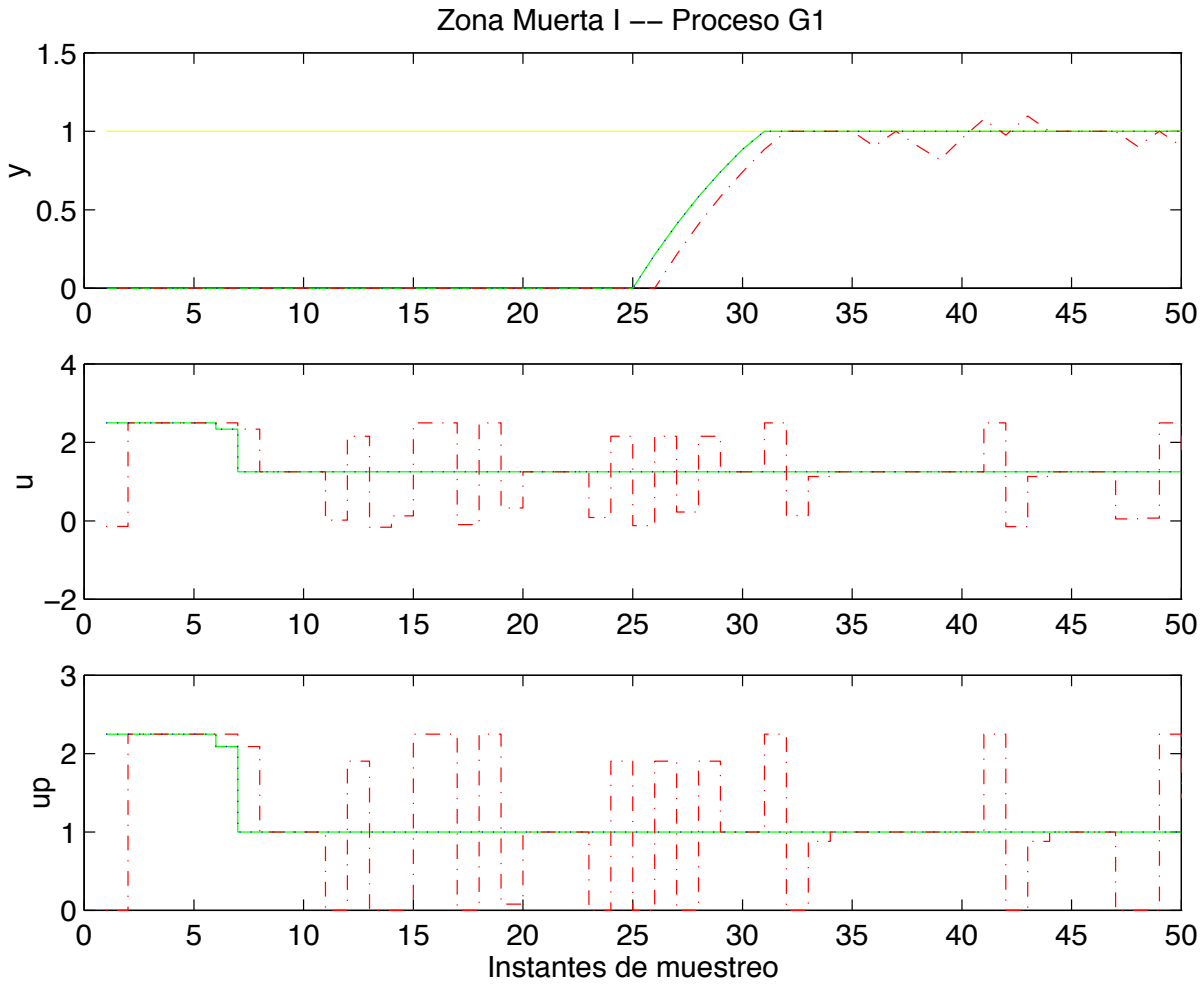


Figura D.22: Control del proceso $G1$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

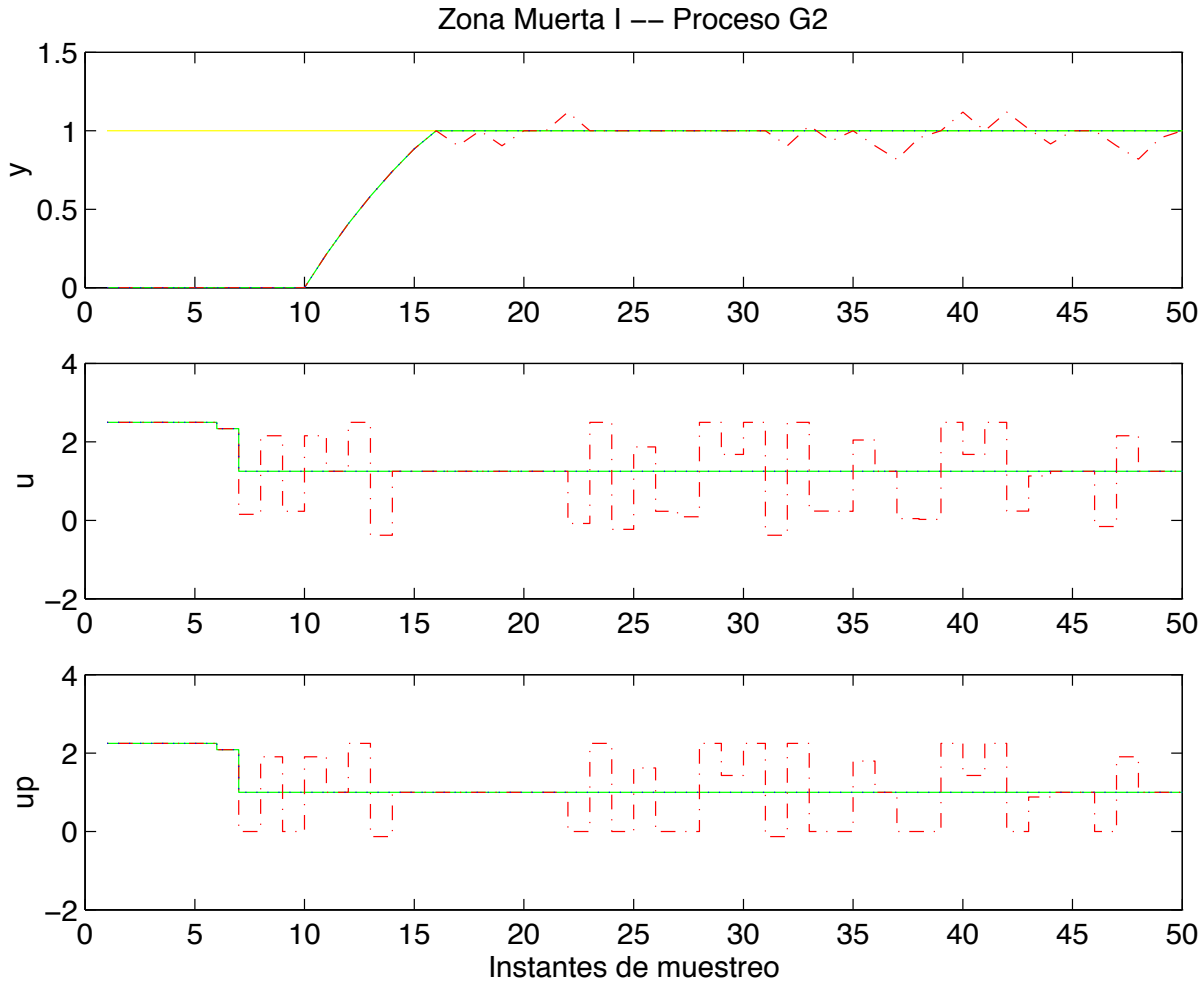


Figura D.23: Control del proceso $G2$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

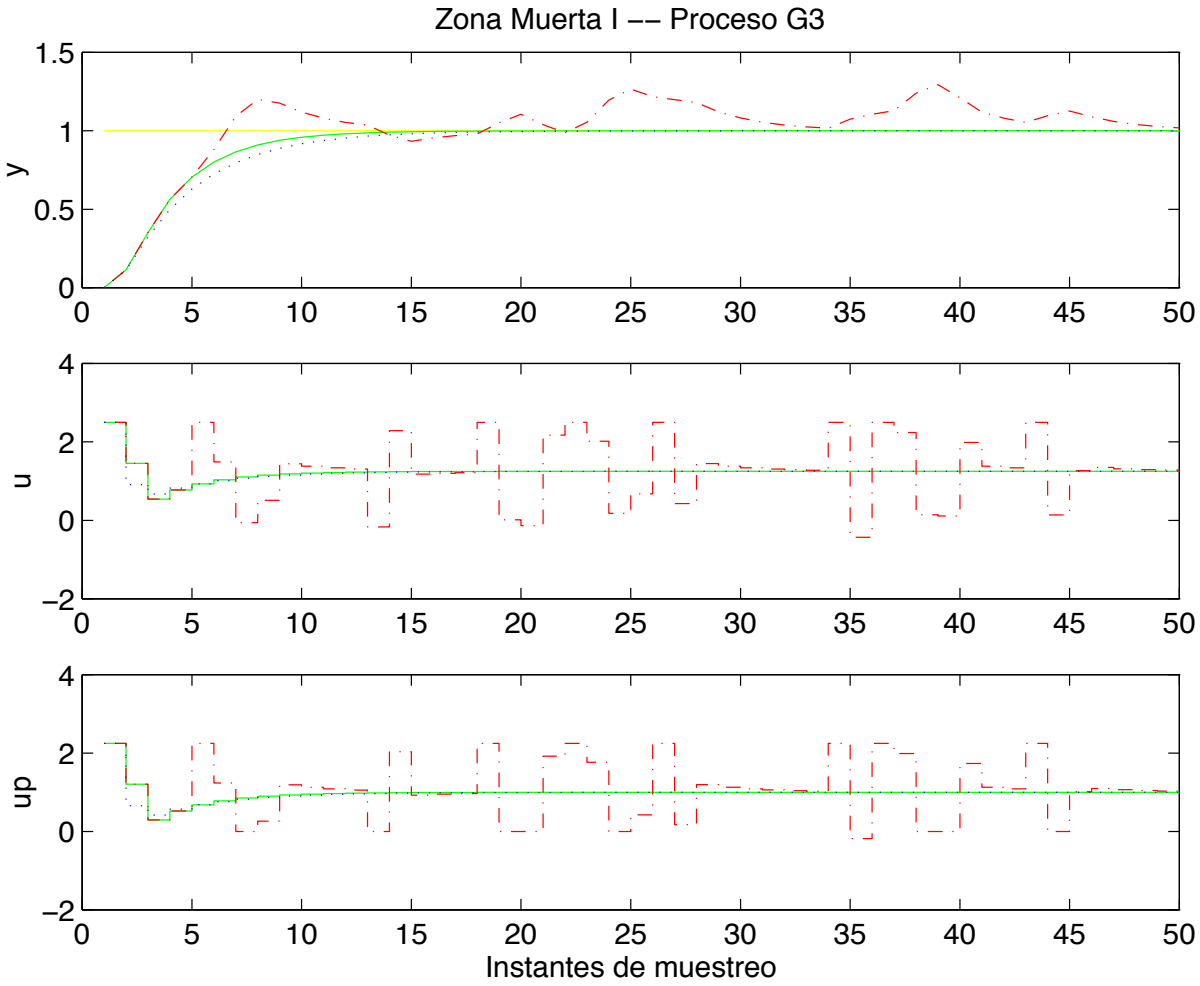


Figura D.24: Control del proceso $G3$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

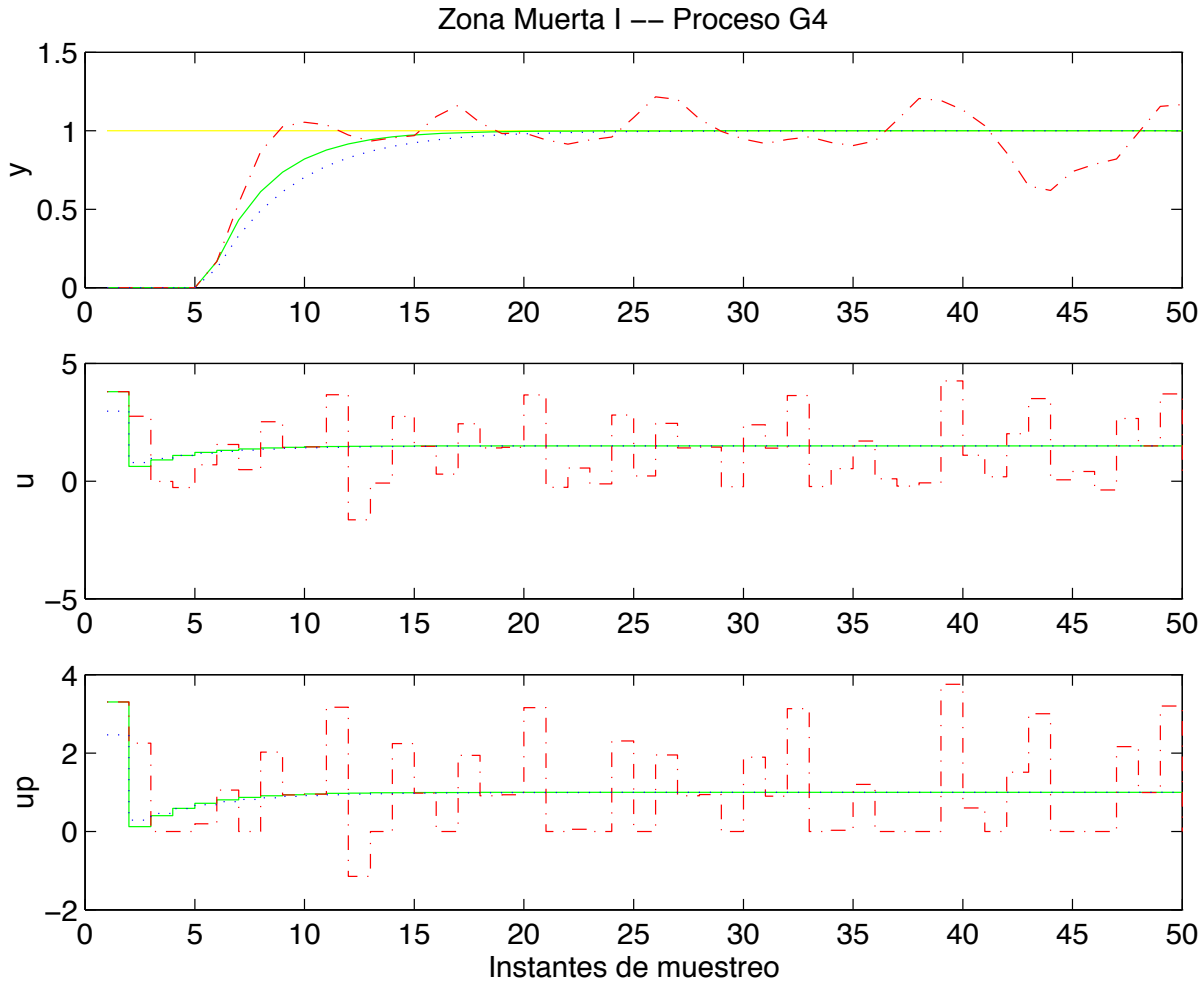


Figura D.25: Control del proceso $G4$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

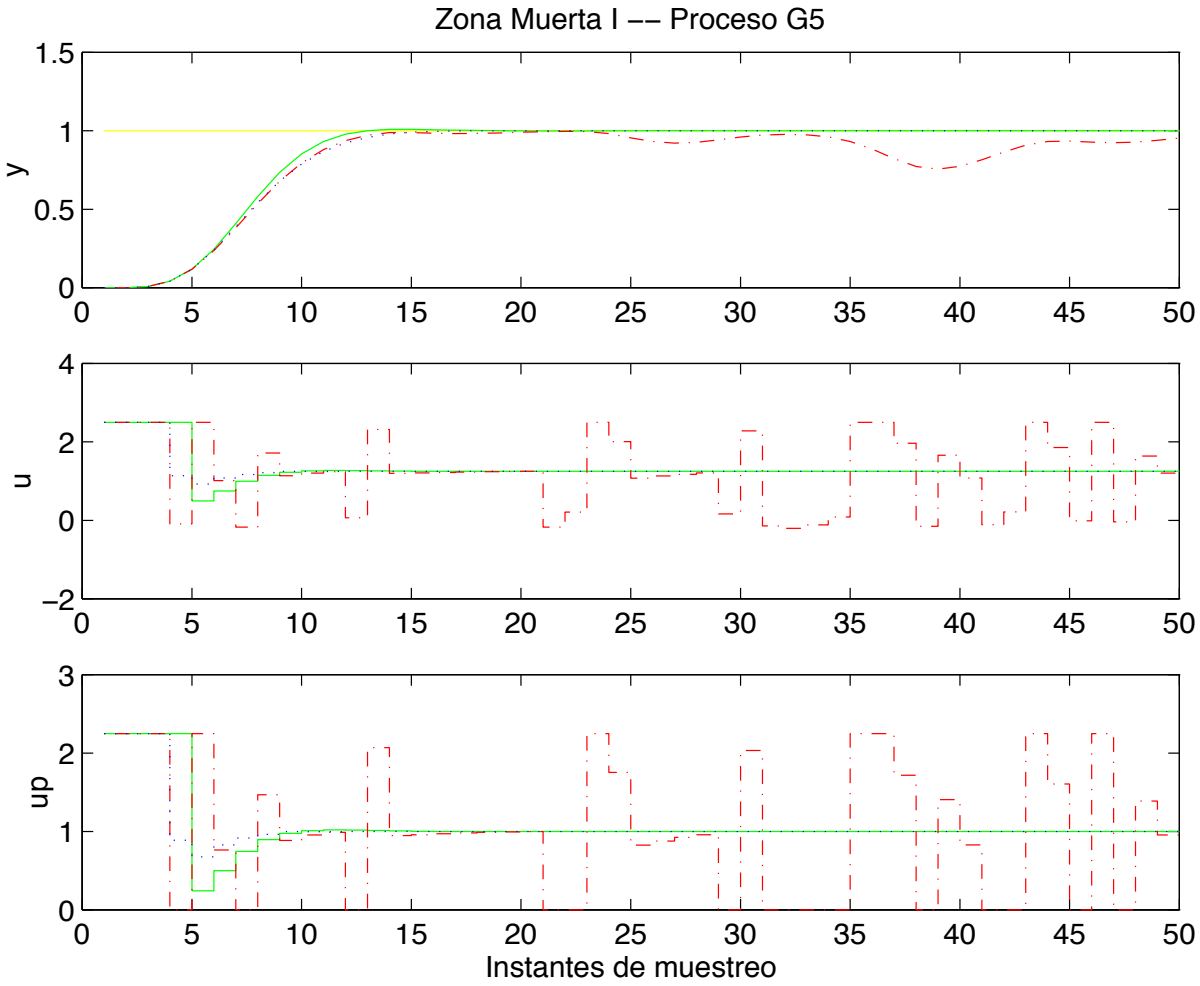


Figura D.26: Control del proceso $G5$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

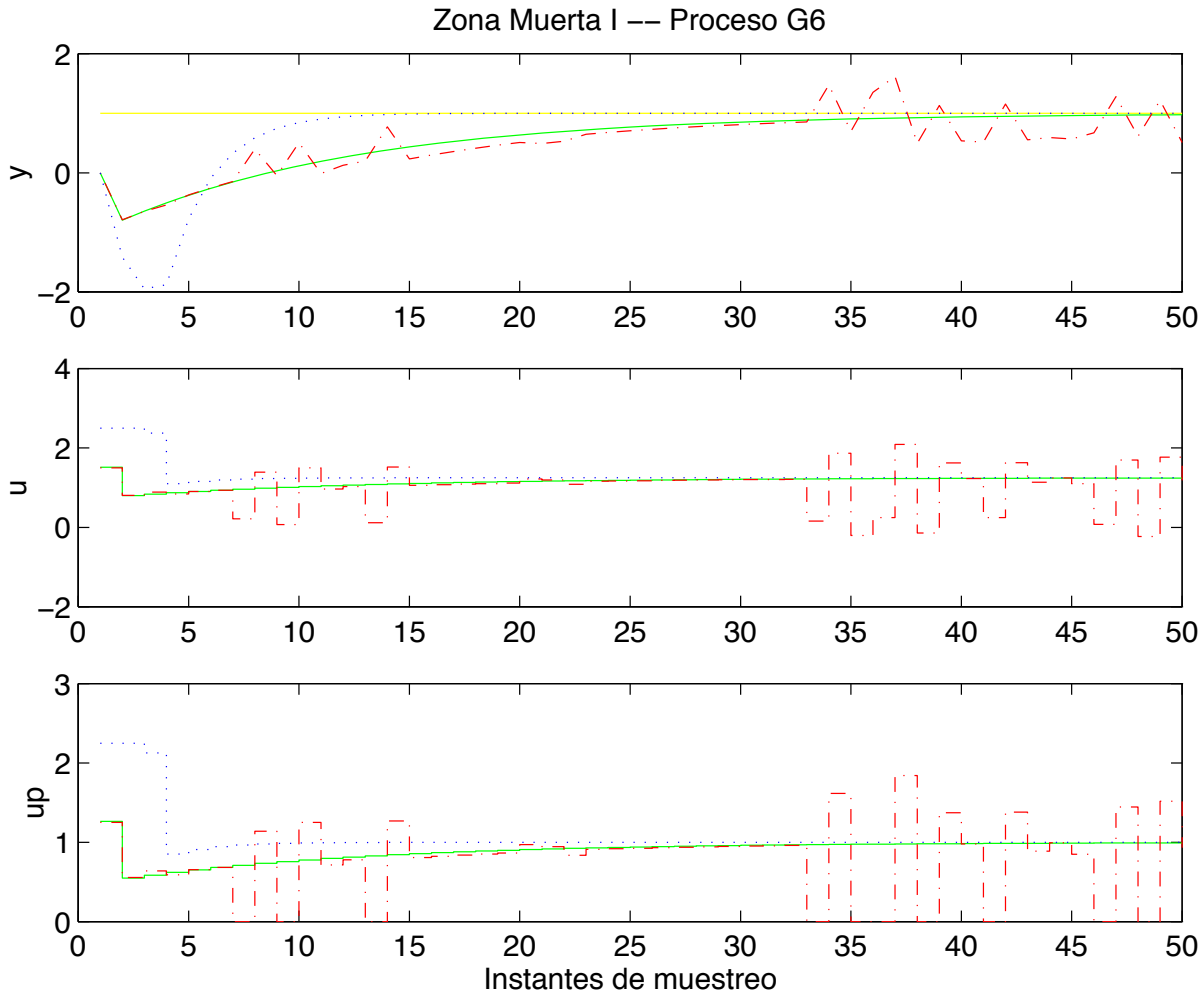


Figura D.27: Control del proceso $G6$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

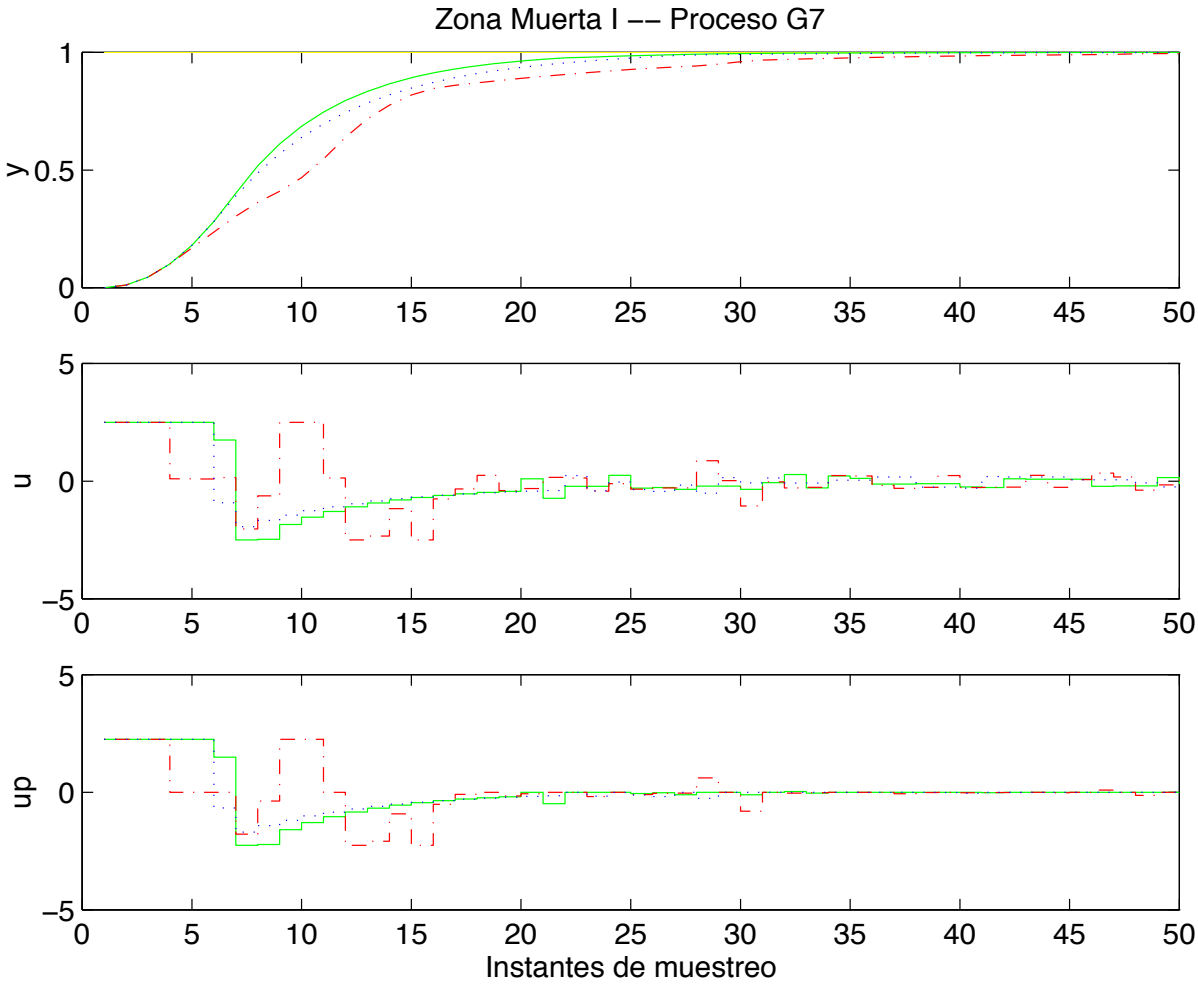


Figura D.28: Control del proceso $G7$ con zona muerta de tipo I en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Zona Muerta de tipo II

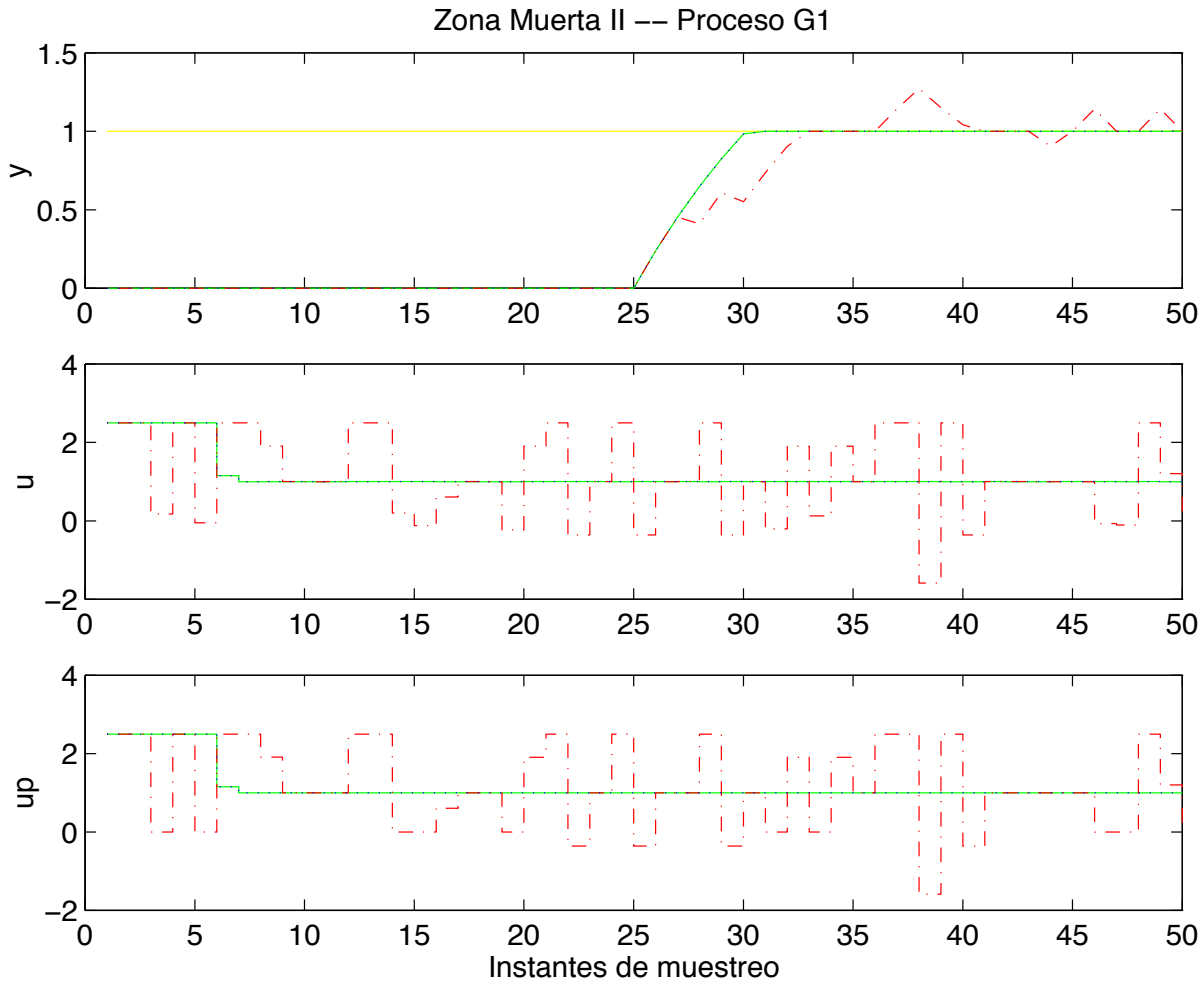


Figura D.29: Control del proceso $G1$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

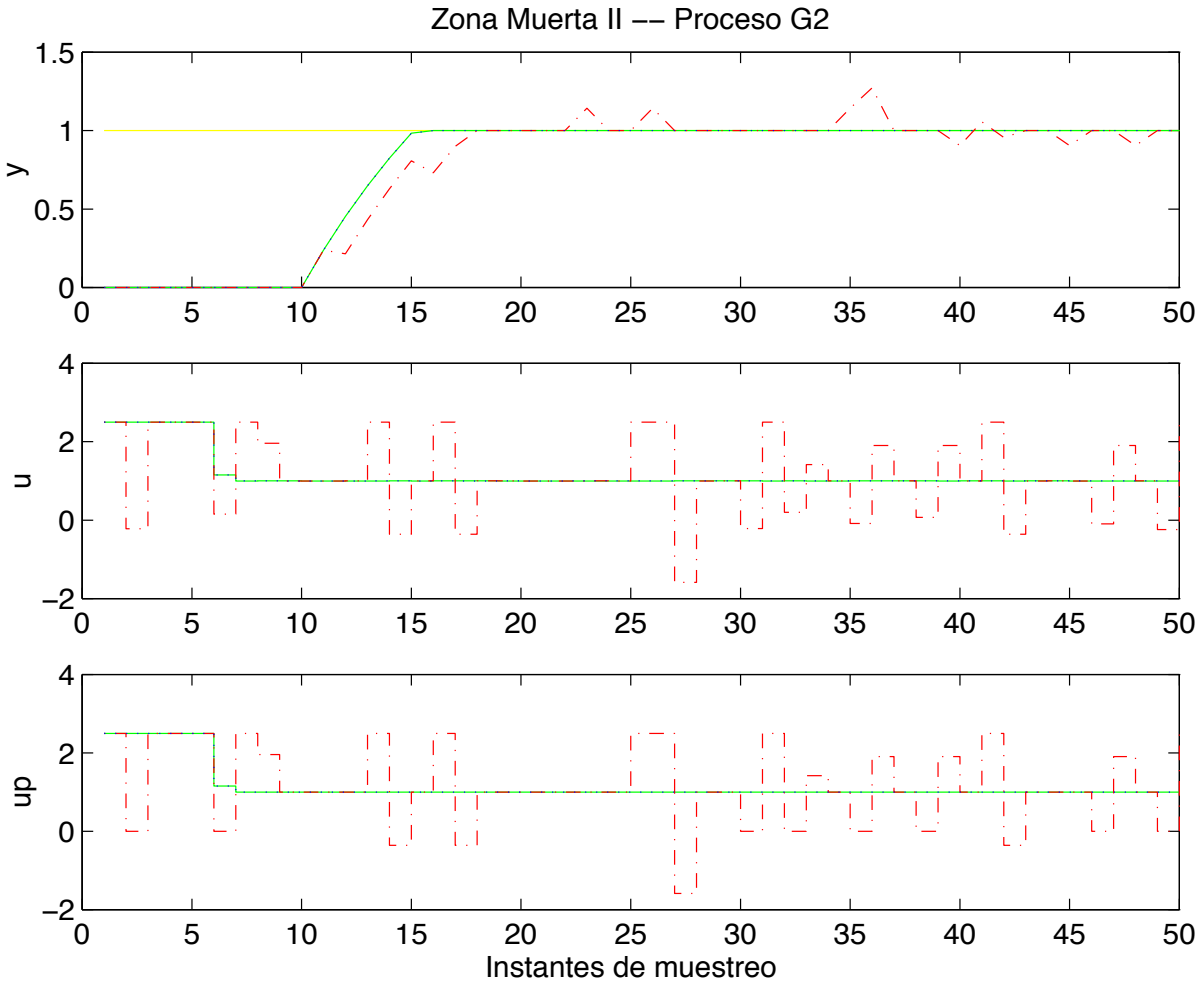


Figura D.30: Control del proceso $G2$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

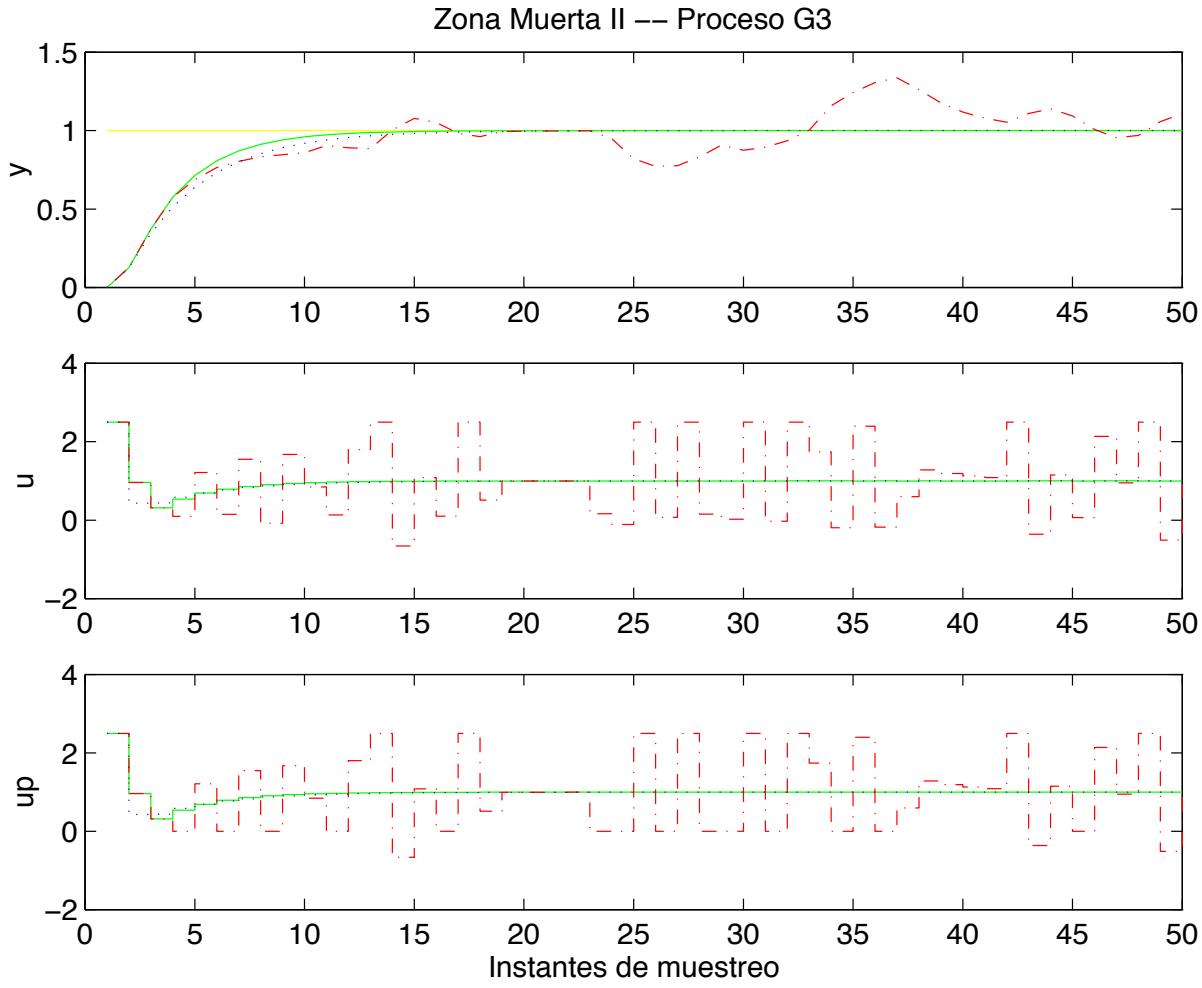


Figura D.31: Control del proceso $G3$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

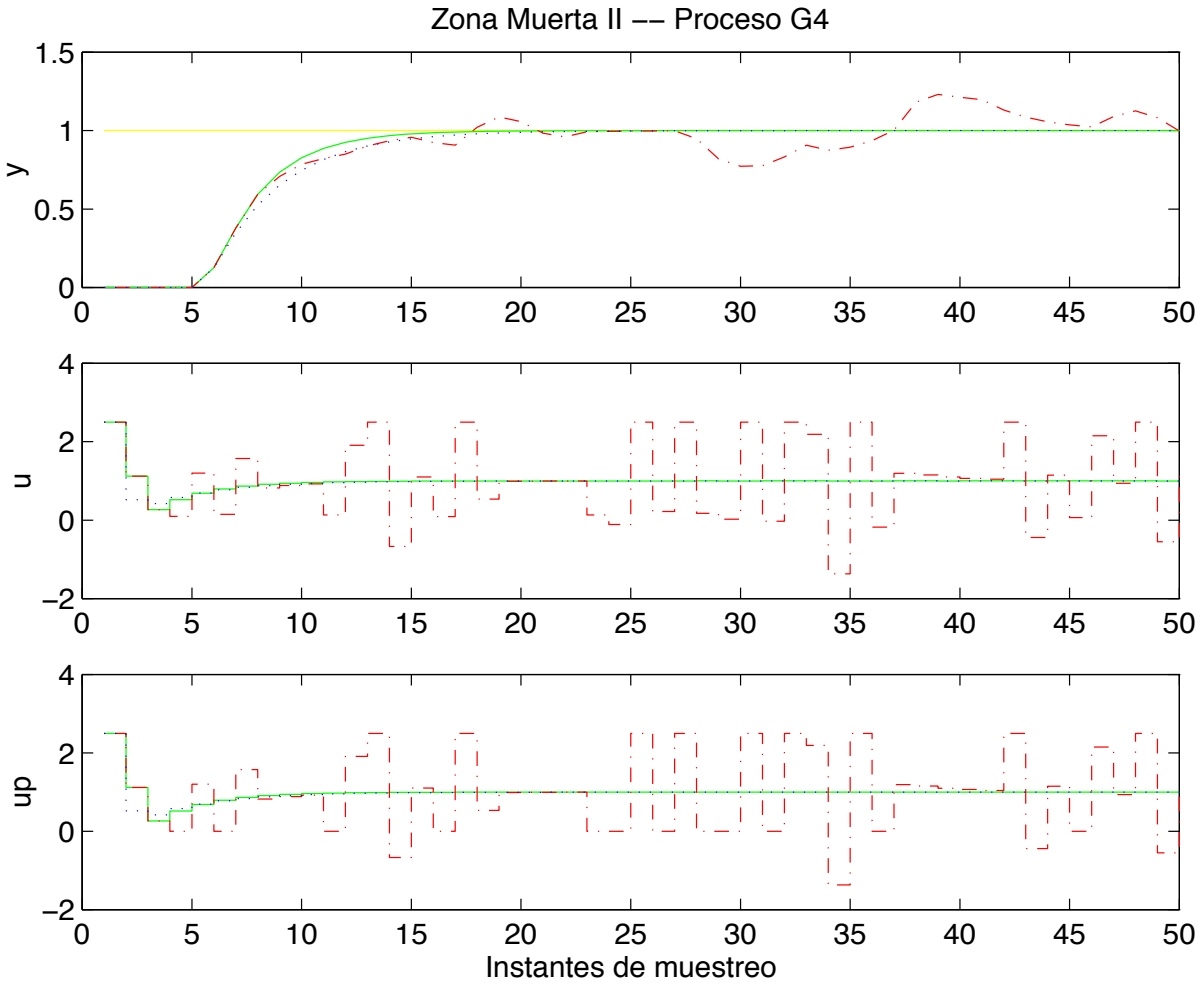


Figura D.32: Control del proceso $G4$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

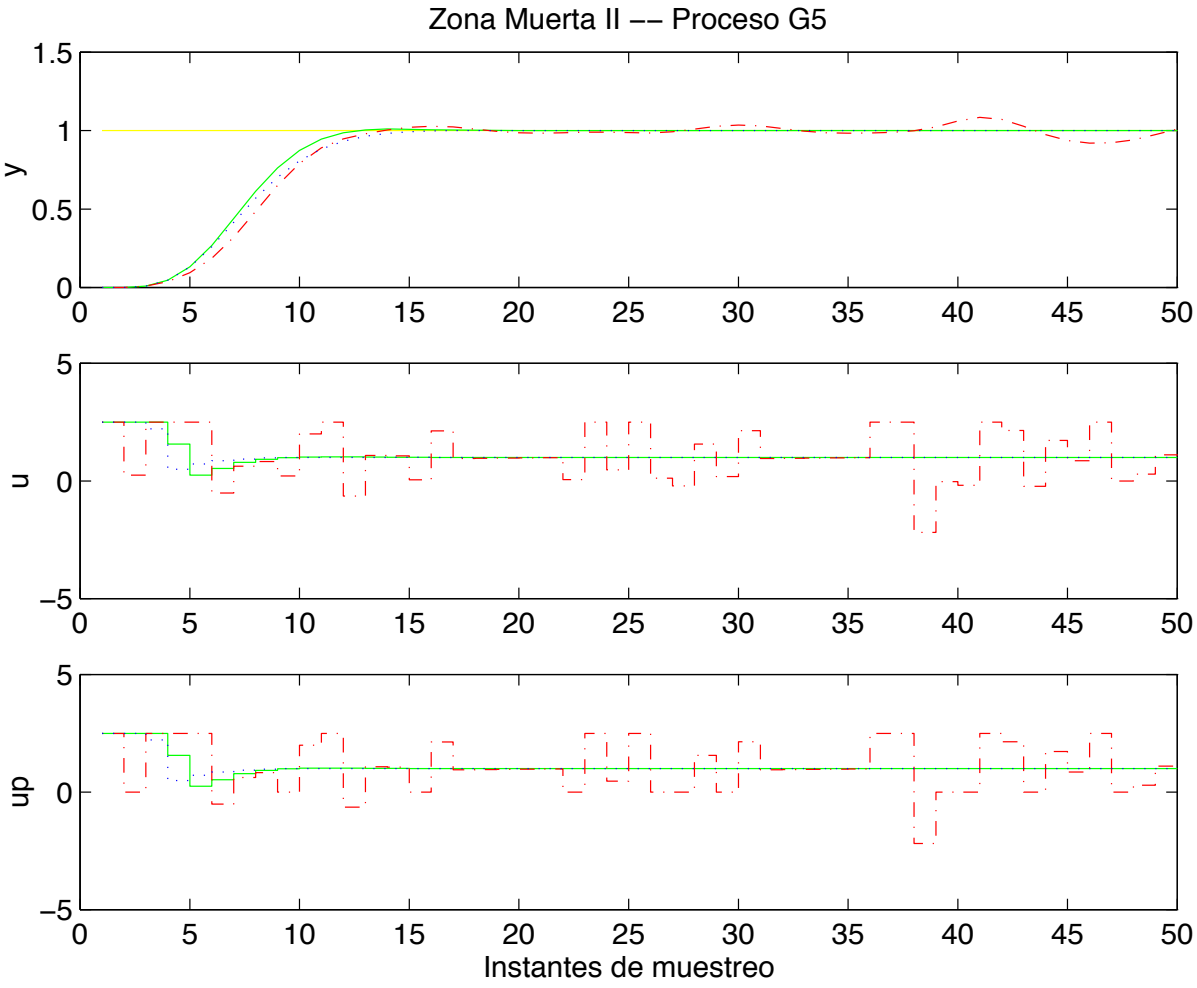


Figura D.33: Control del proceso $G5$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

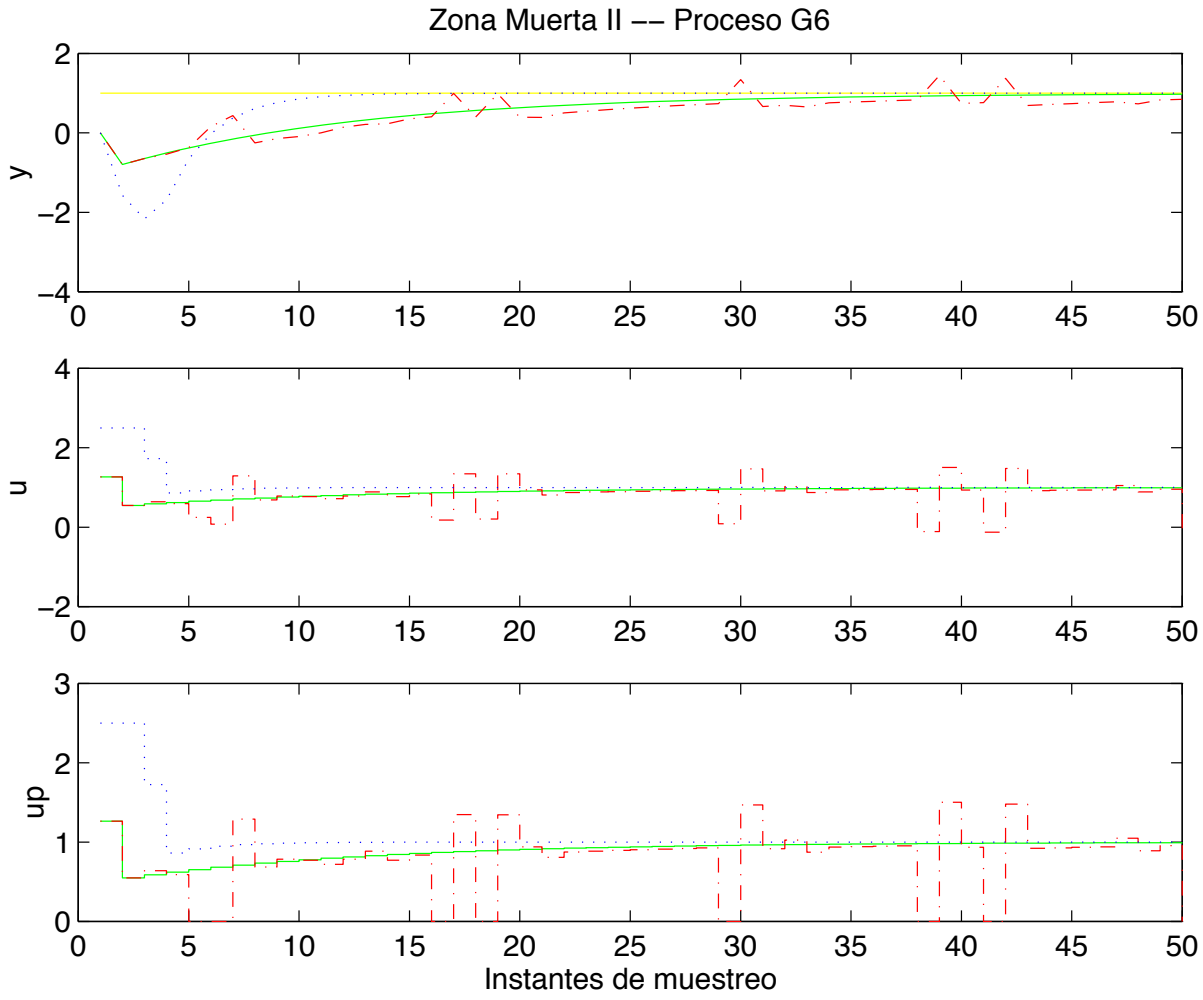


Figura D.34: Control del proceso $G6$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

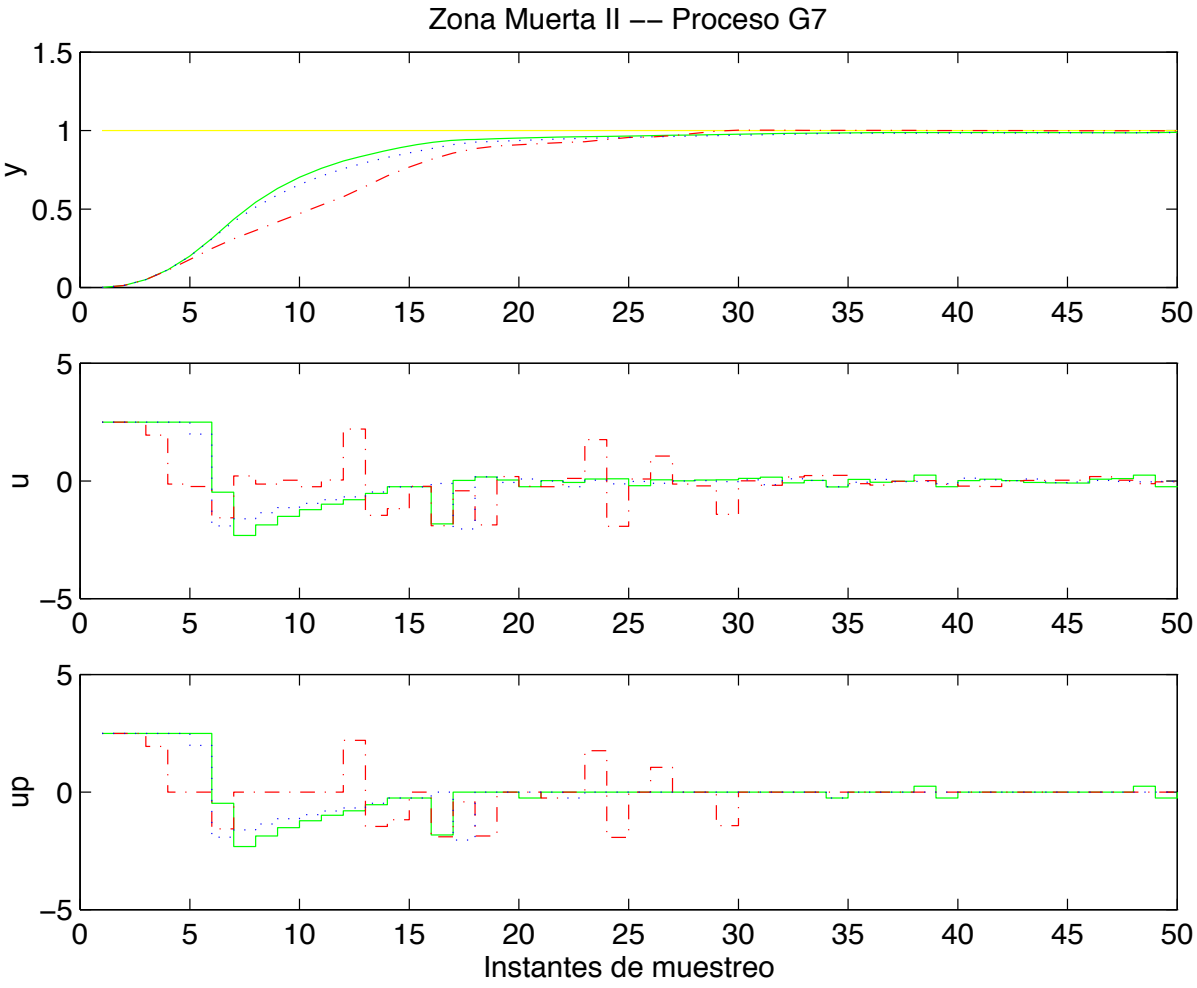


Figura D.35: Control del proceso $G7$ con zona muerta de tipo II en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Backlash

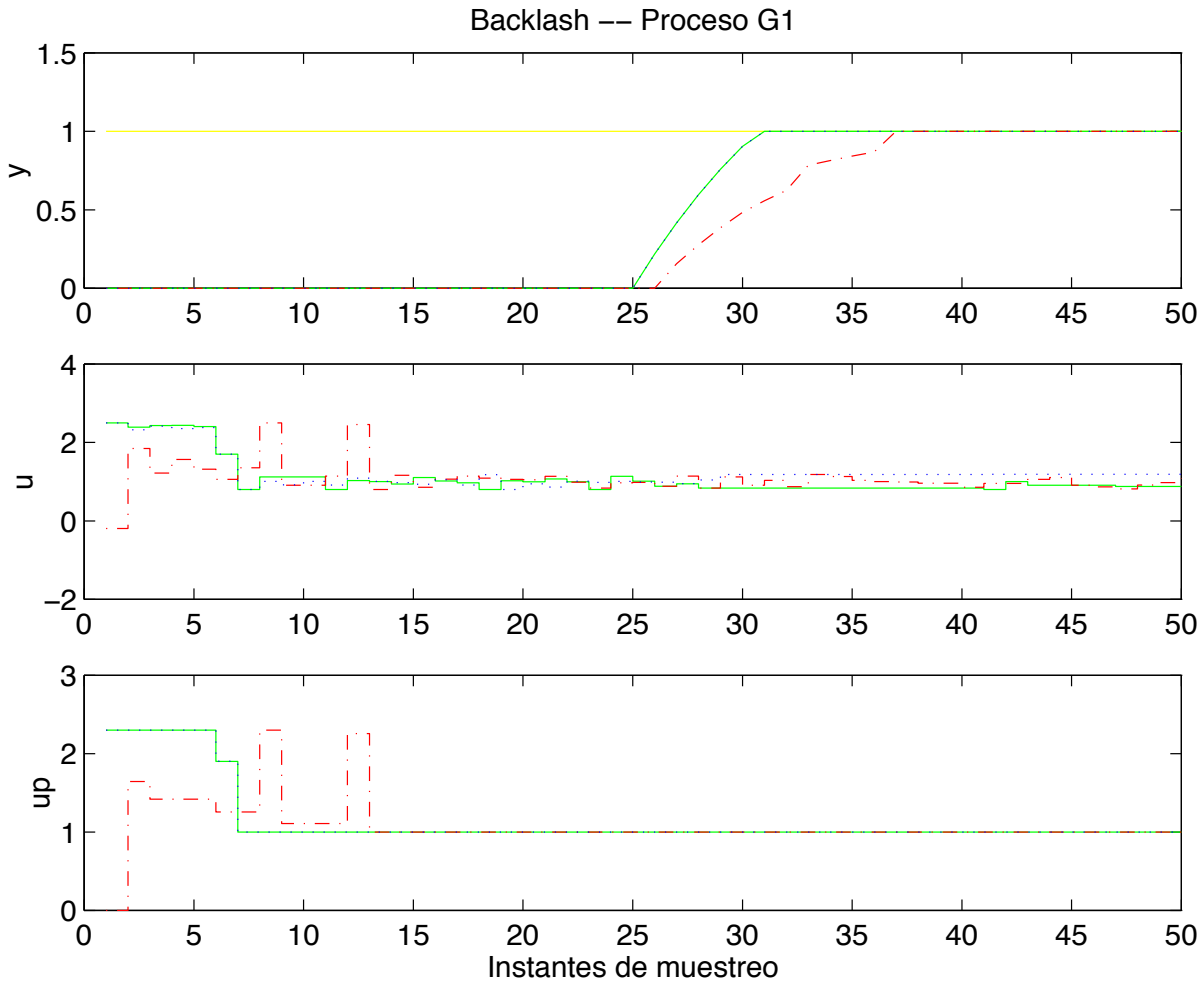


Figura D.36: Control del proceso $G1$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

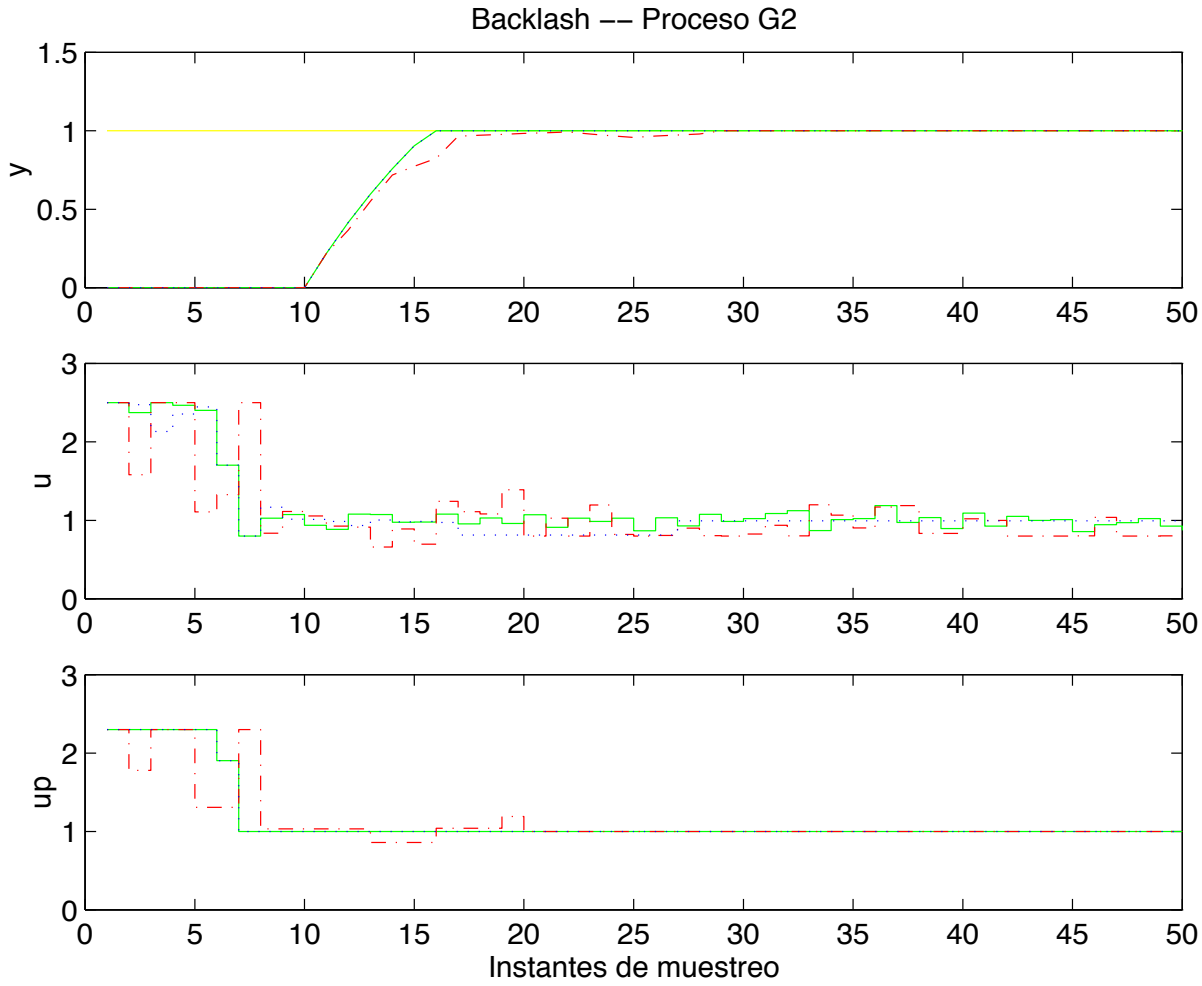


Figura D.37: Control del proceso $G2$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

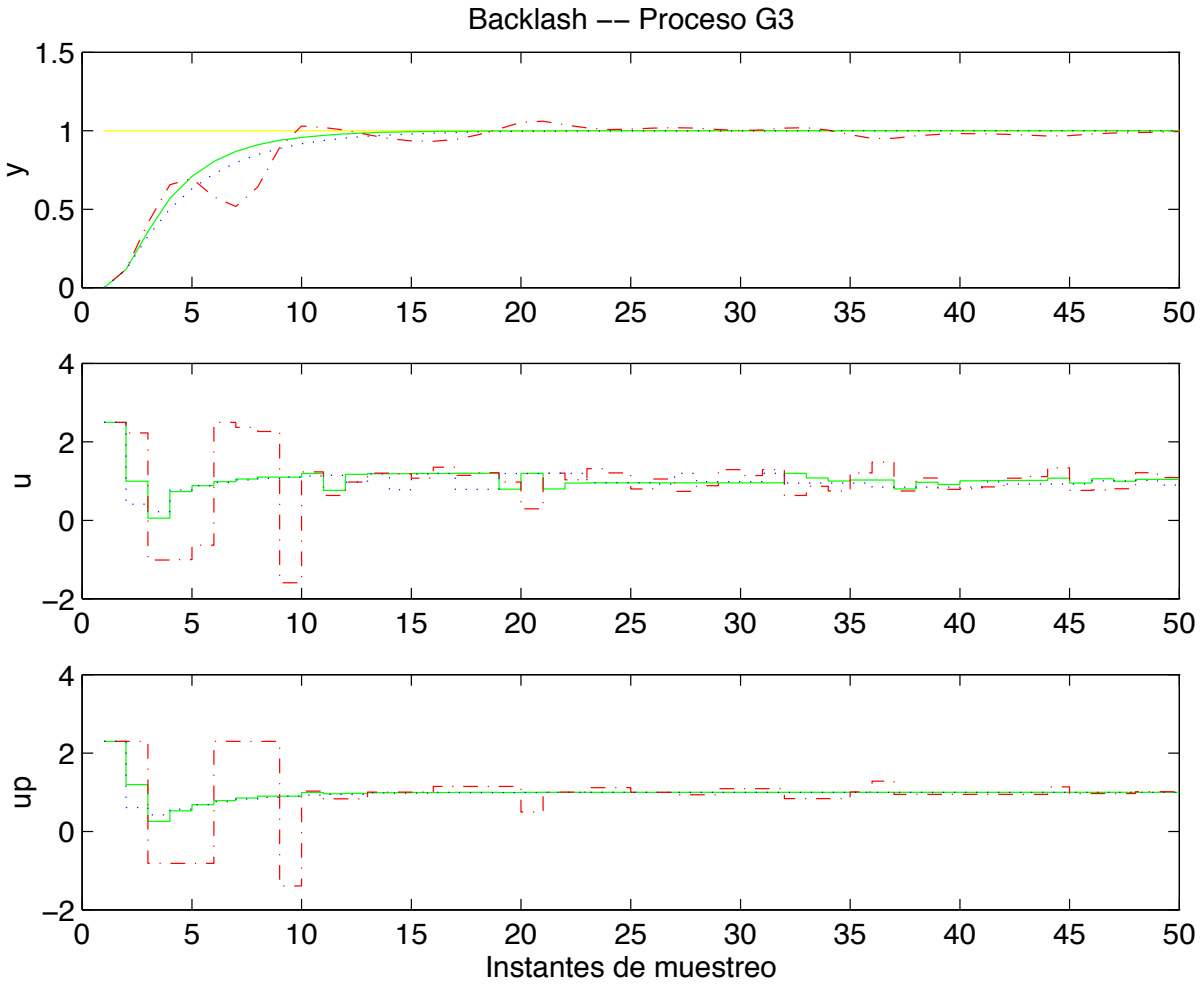


Figura D.38: Control del proceso $G3$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

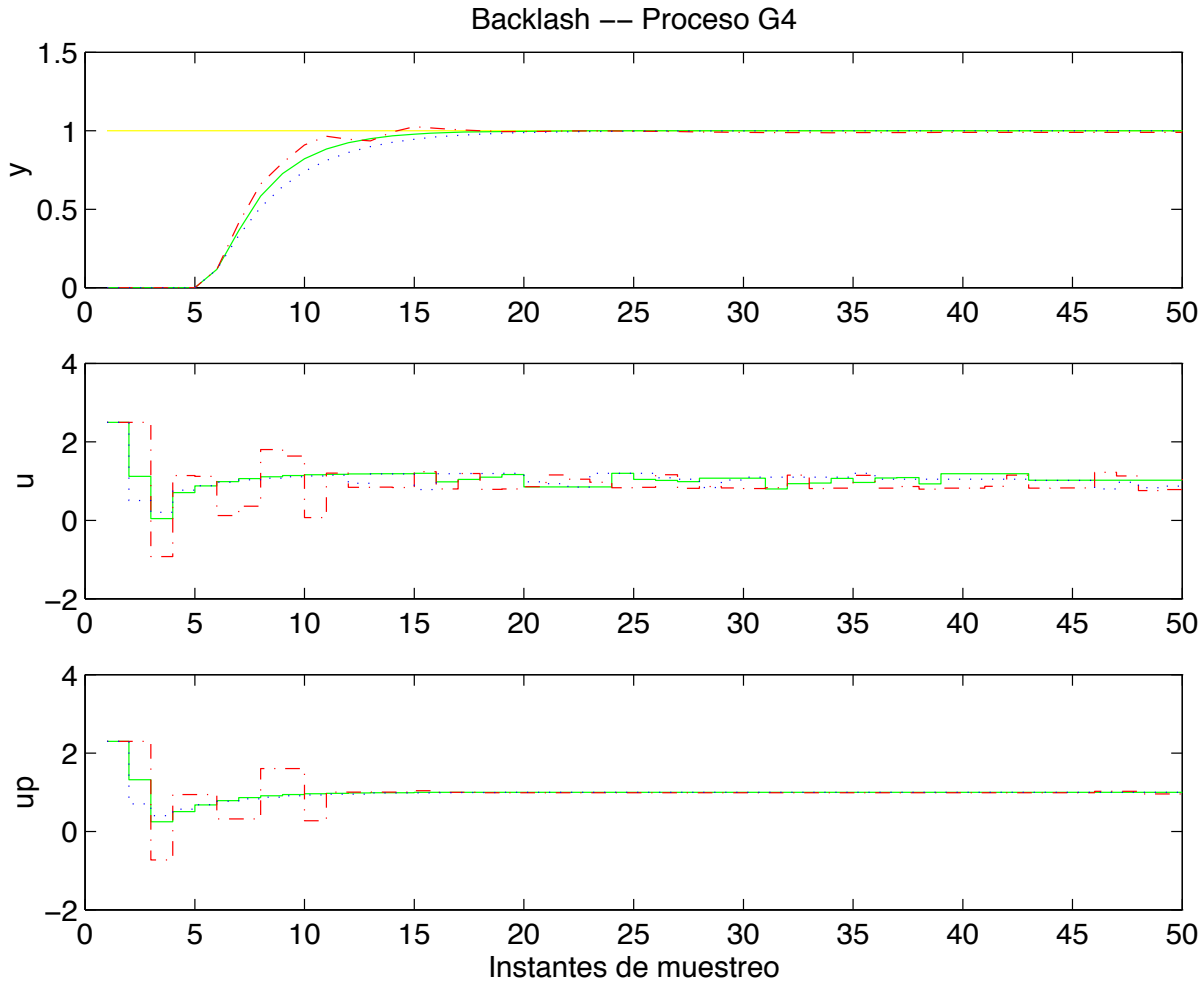


Figura D.39: Control del proceso $G4$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

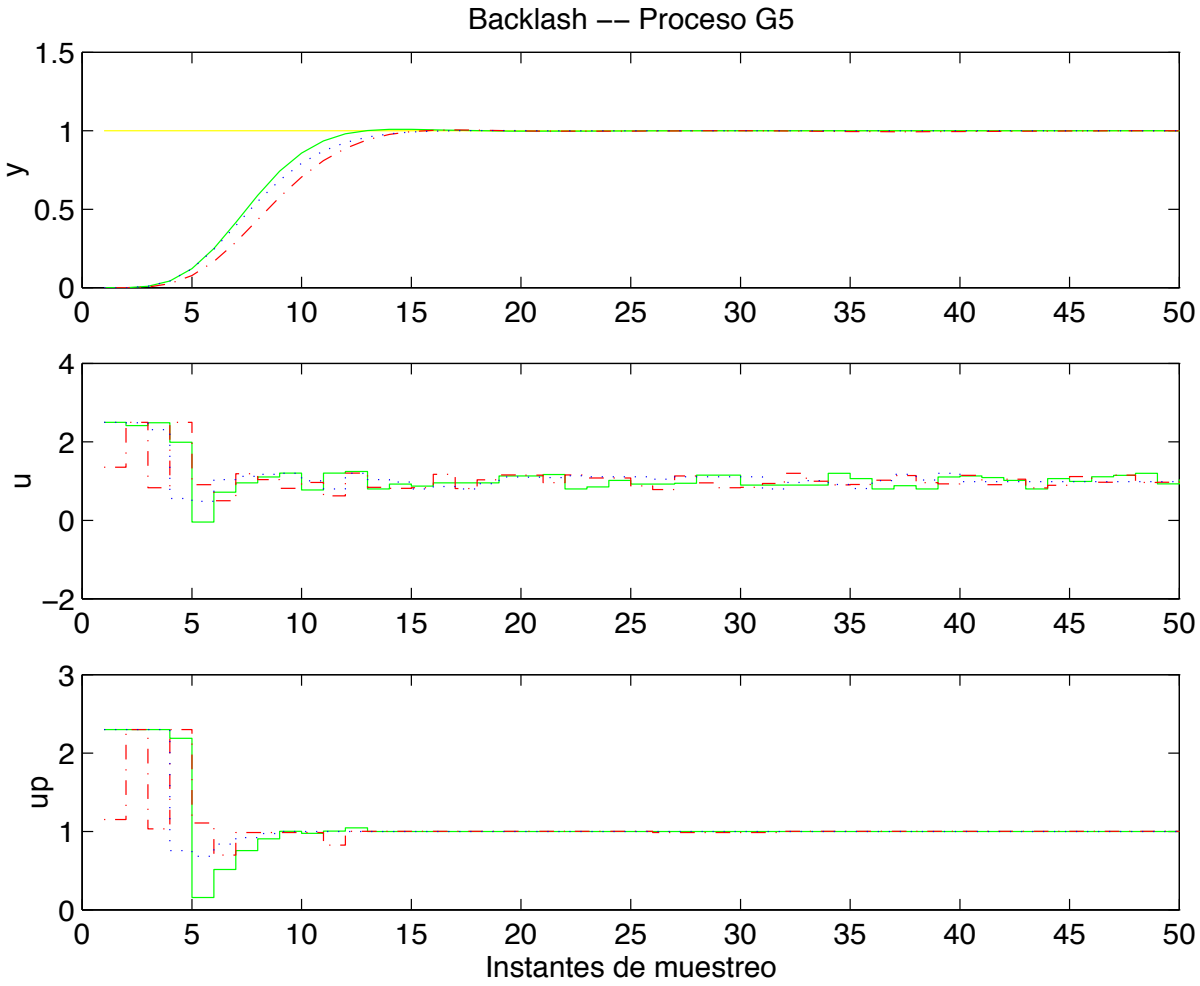


Figura D.40: Control del proceso $G5$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

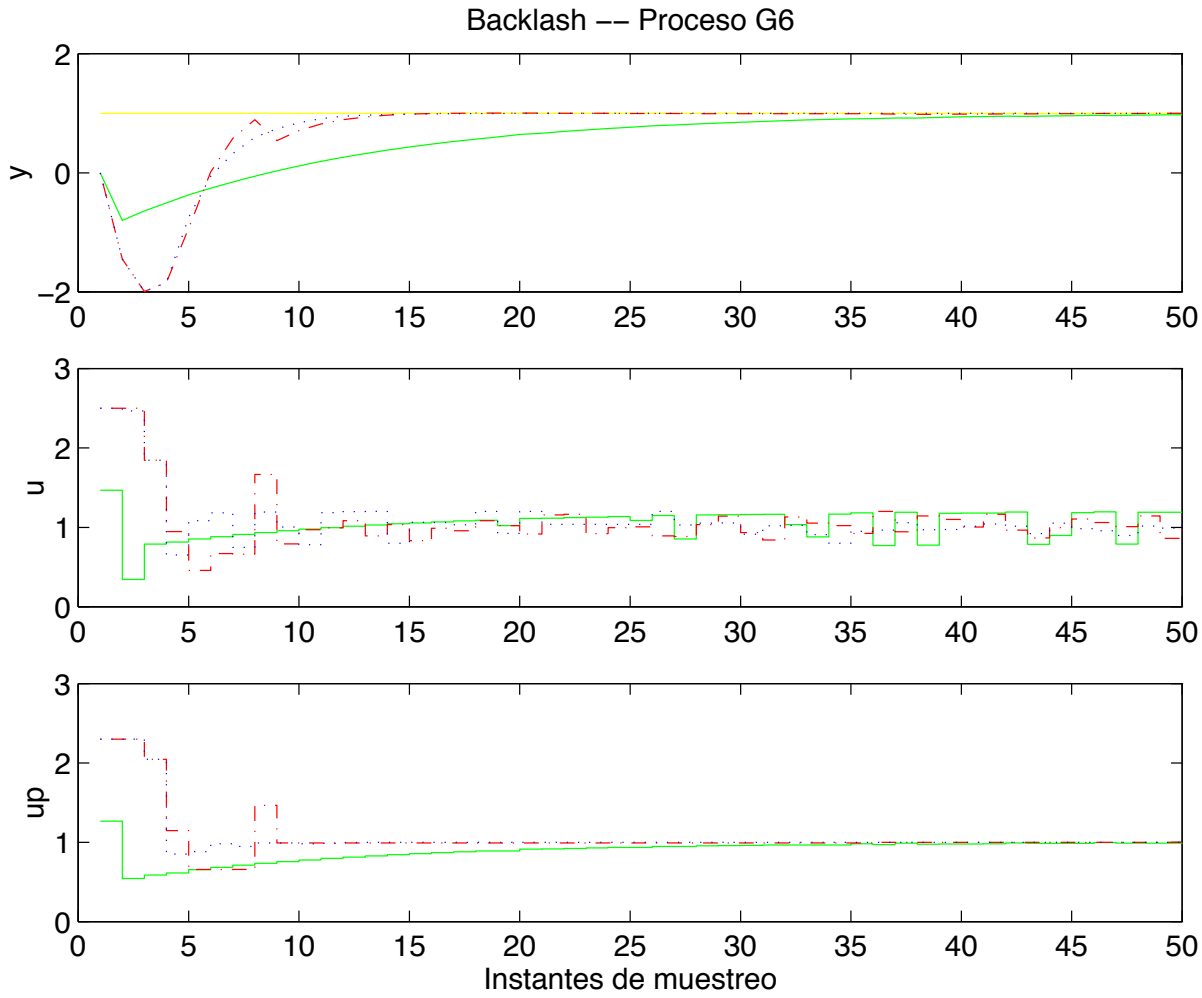


Figura D.41: Control del proceso $G6$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

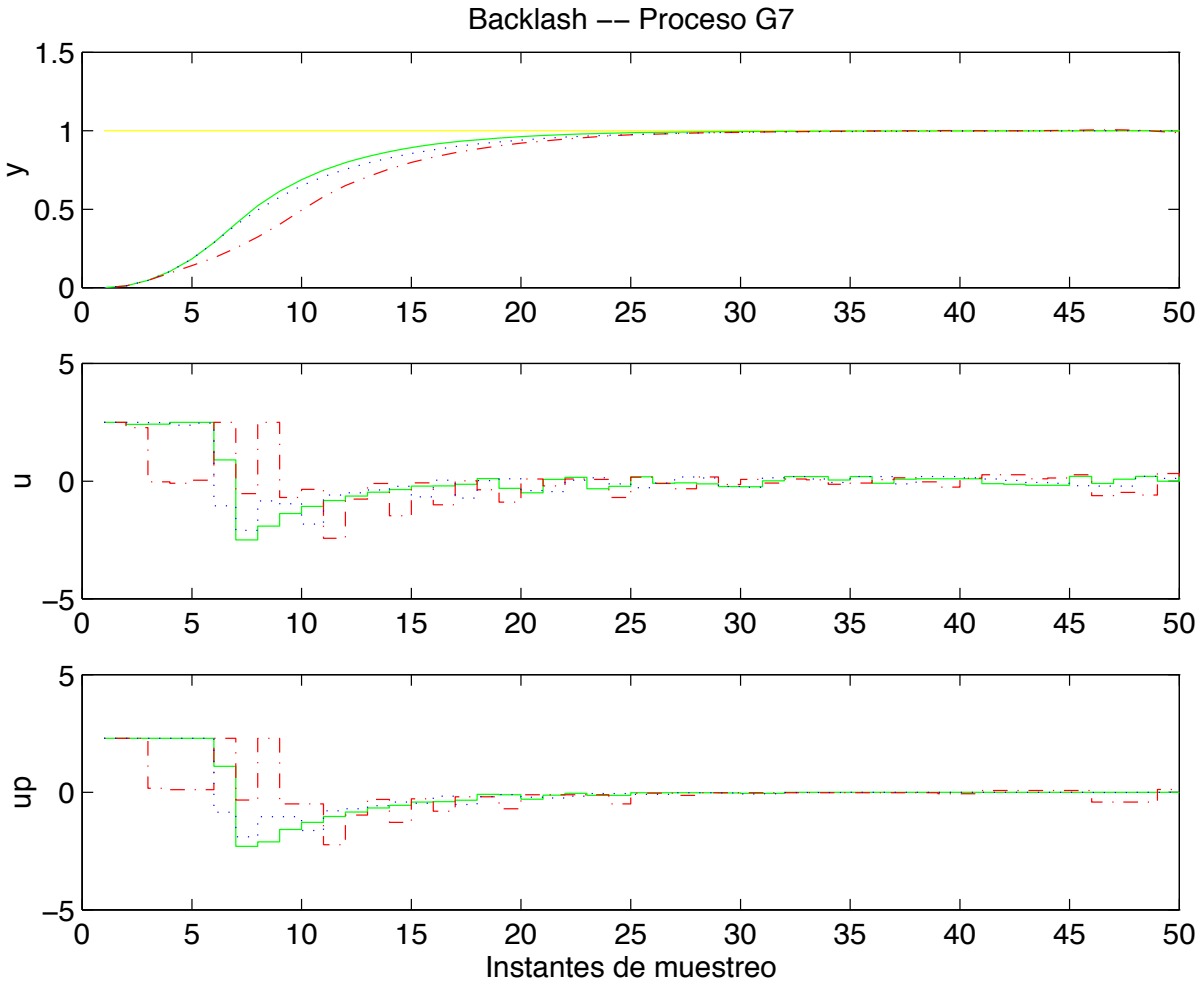


Figura D.42: Control del proceso $G7$ con backlash en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Histéresis

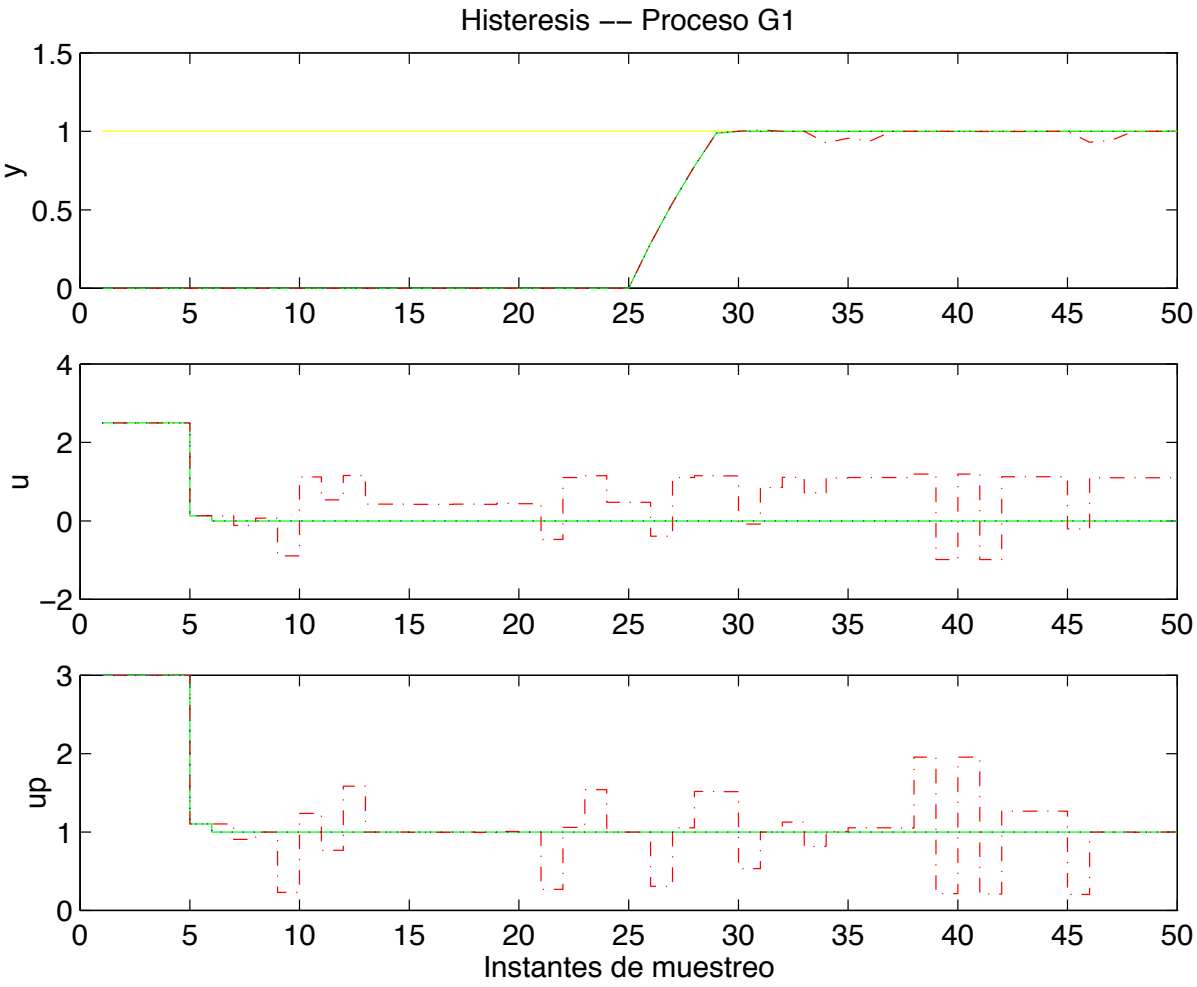


Figura D.43: Control del proceso $G1$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

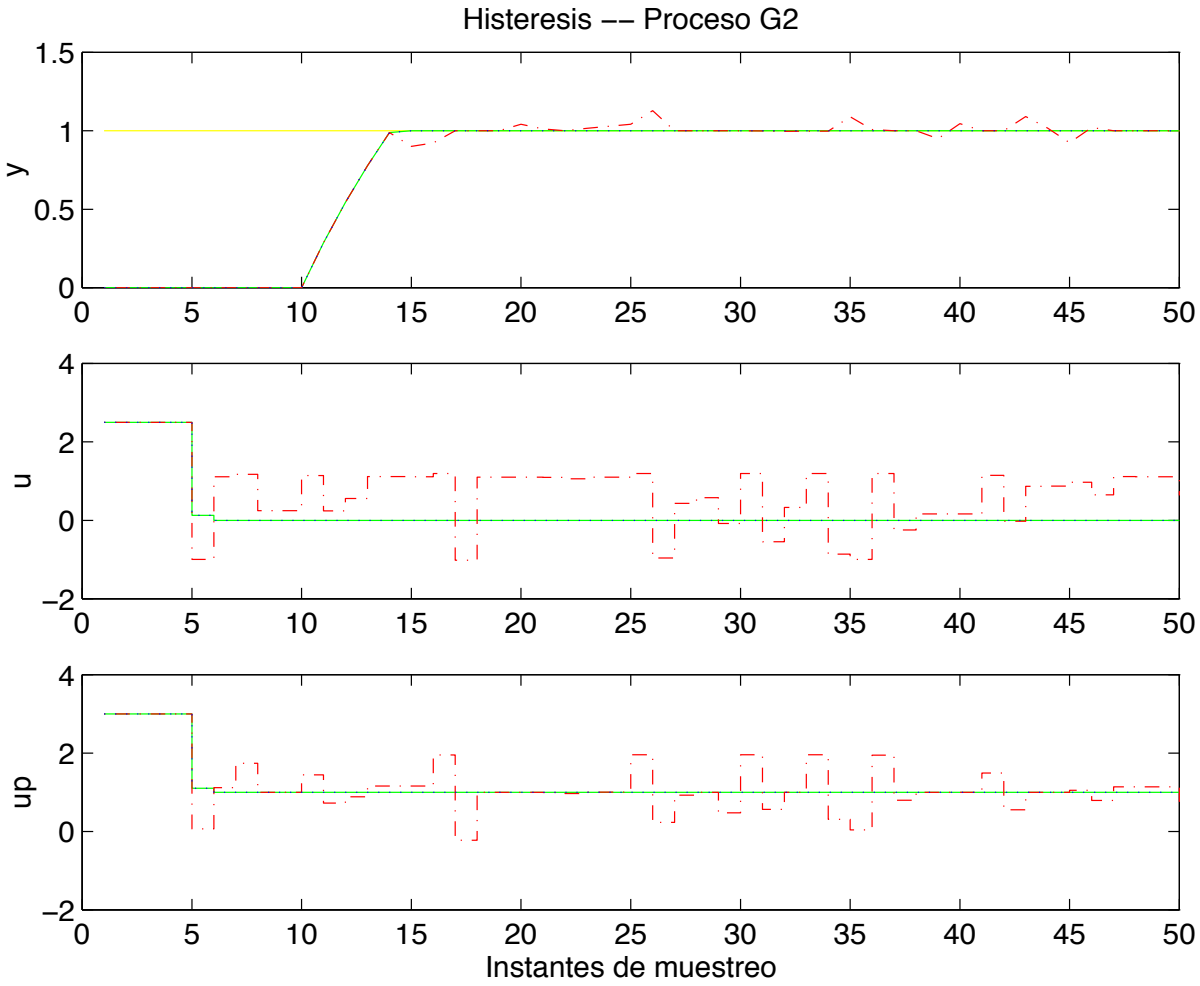


Figura D.44: Control del proceso $G2$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

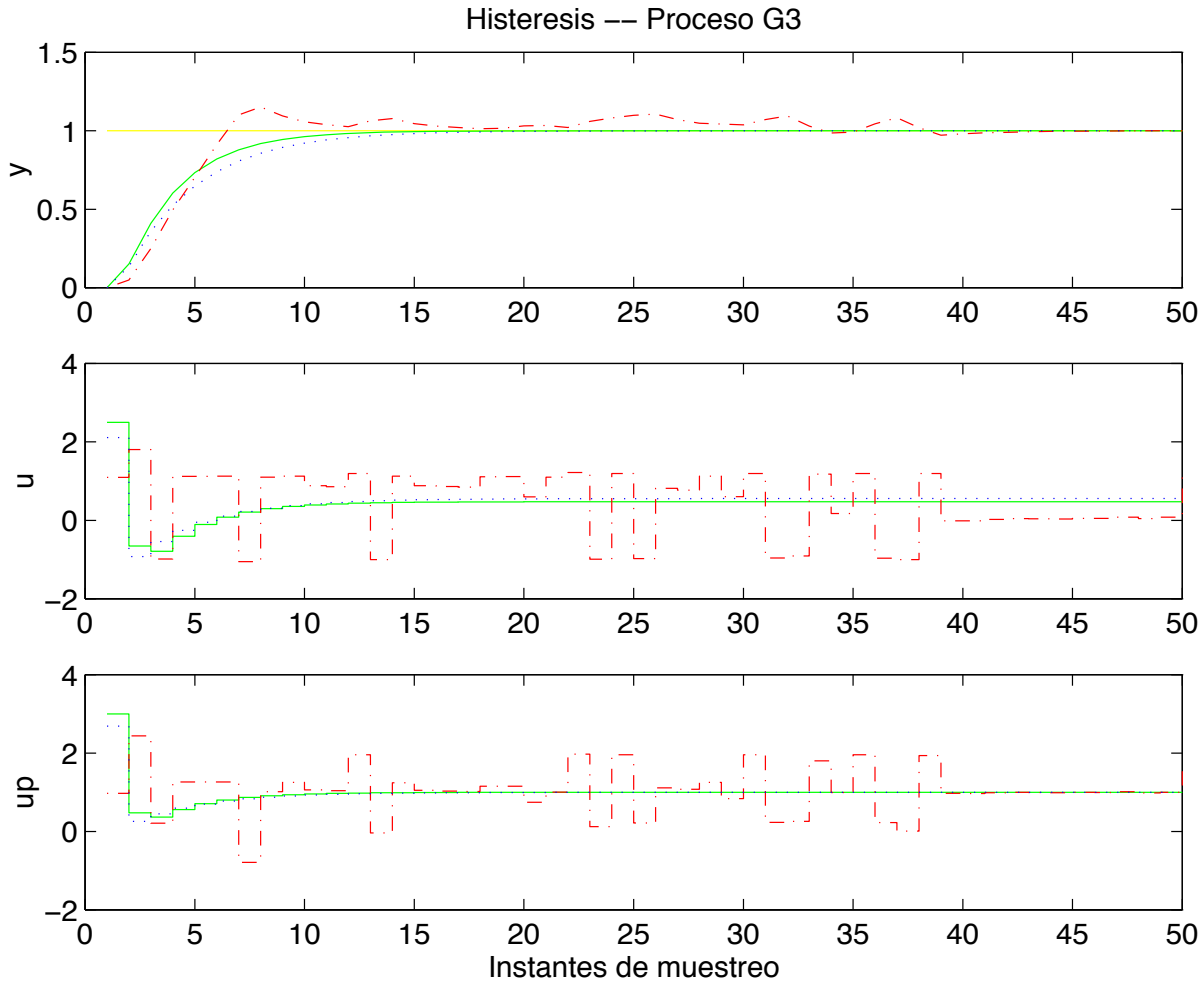


Figura D.45: Control del proceso $G3$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

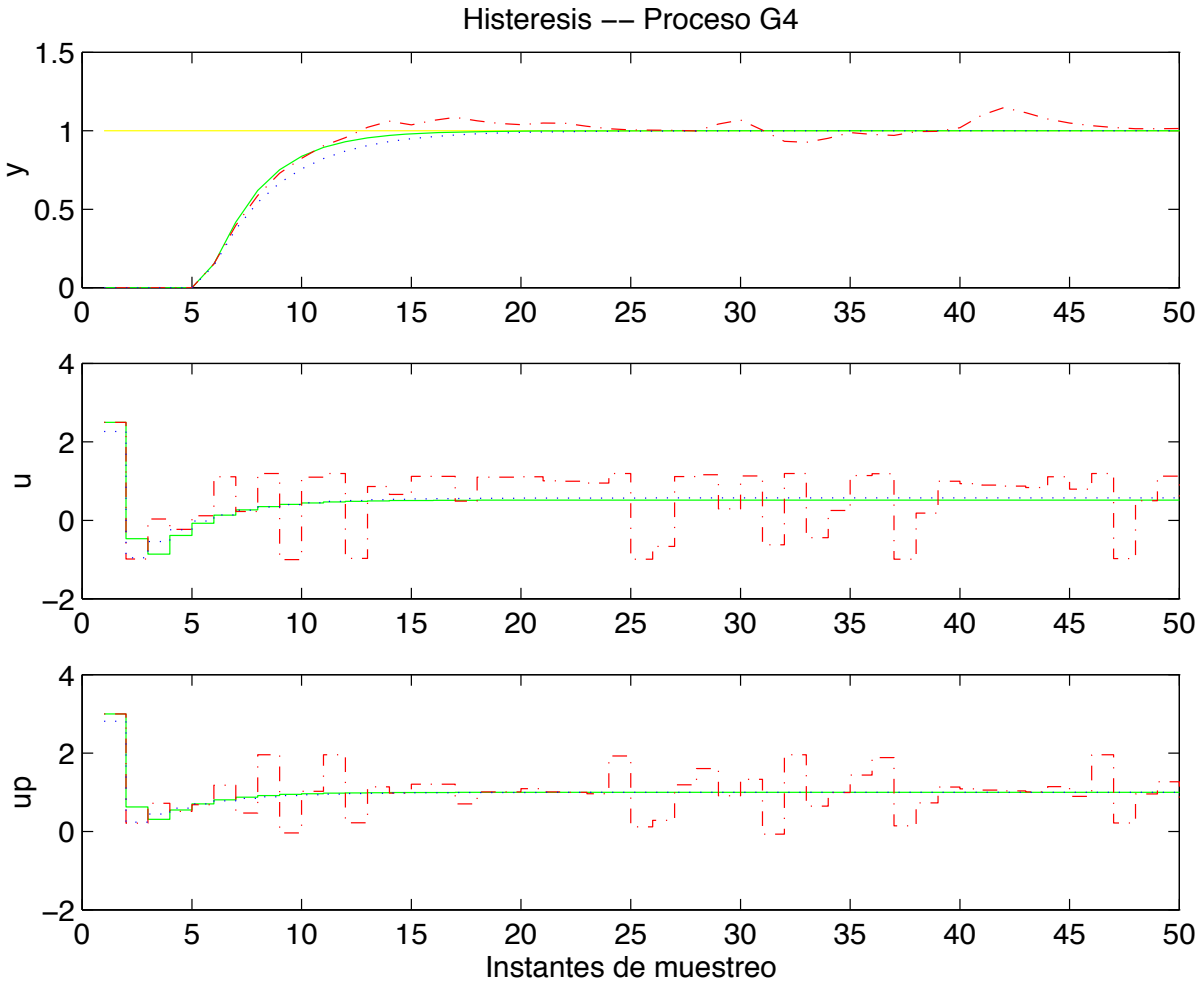


Figura D.46: Control del proceso $G4$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

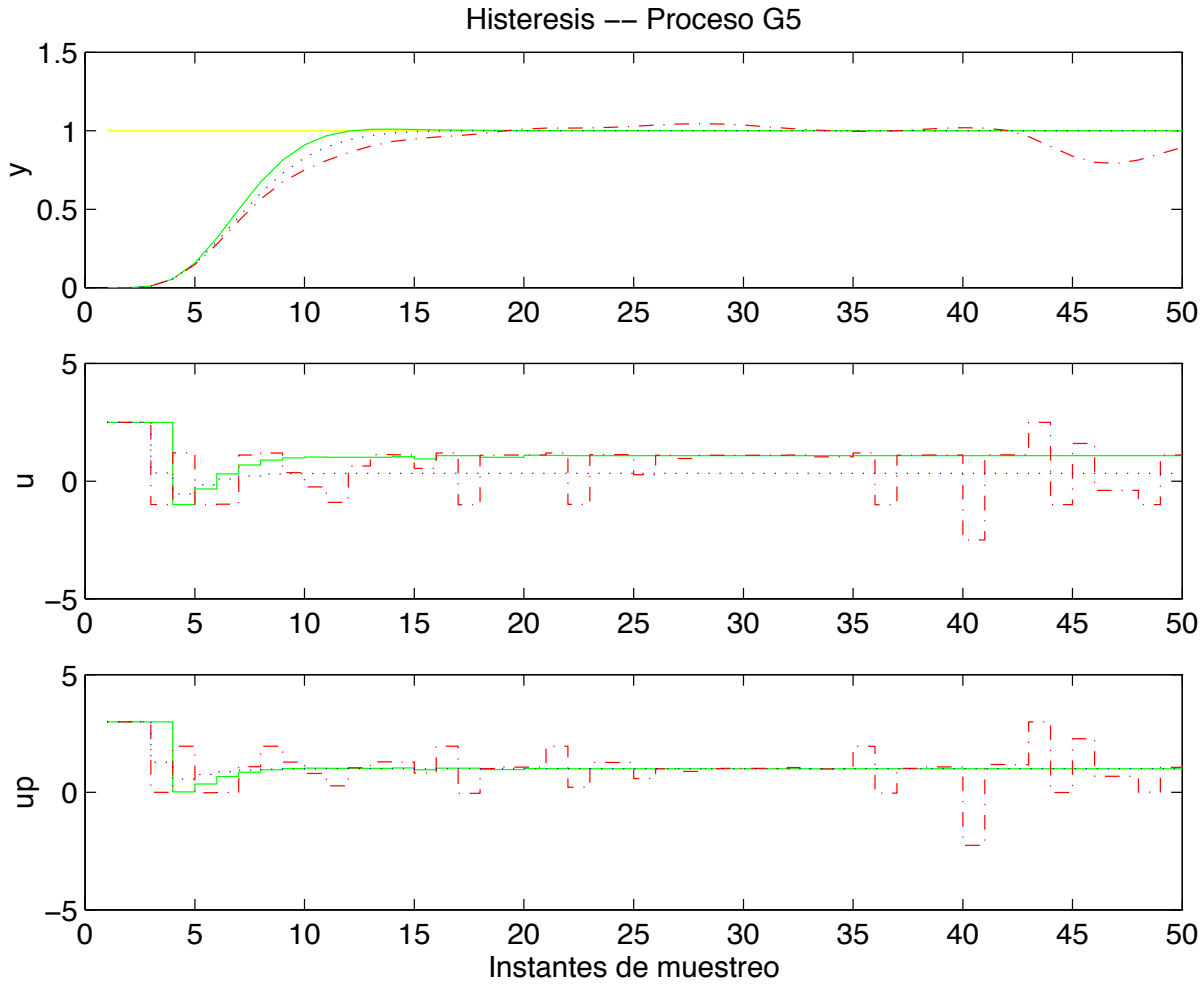


Figura D.47: Control del proceso $G5$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

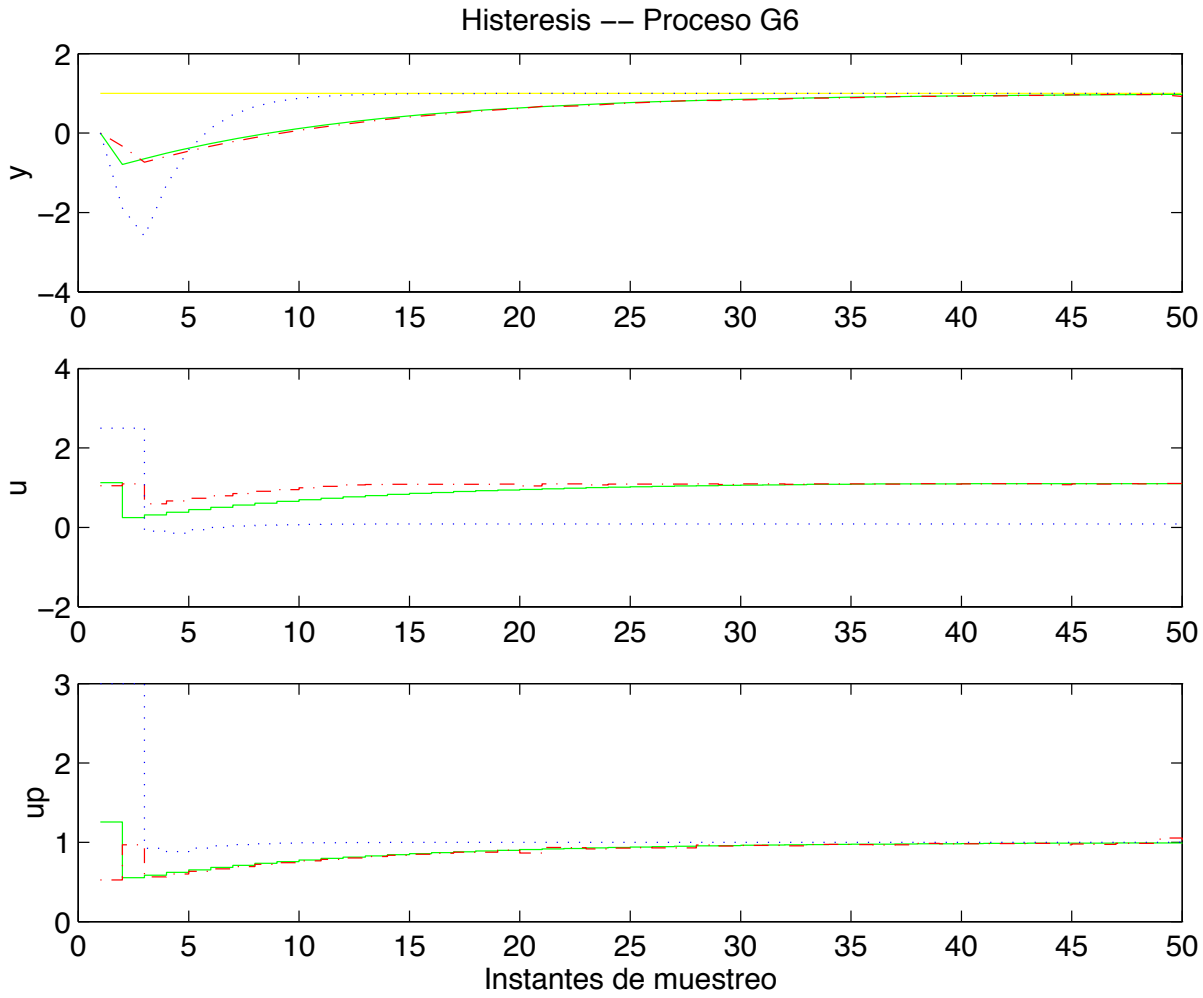


Figura D.48: Control del proceso $G6$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

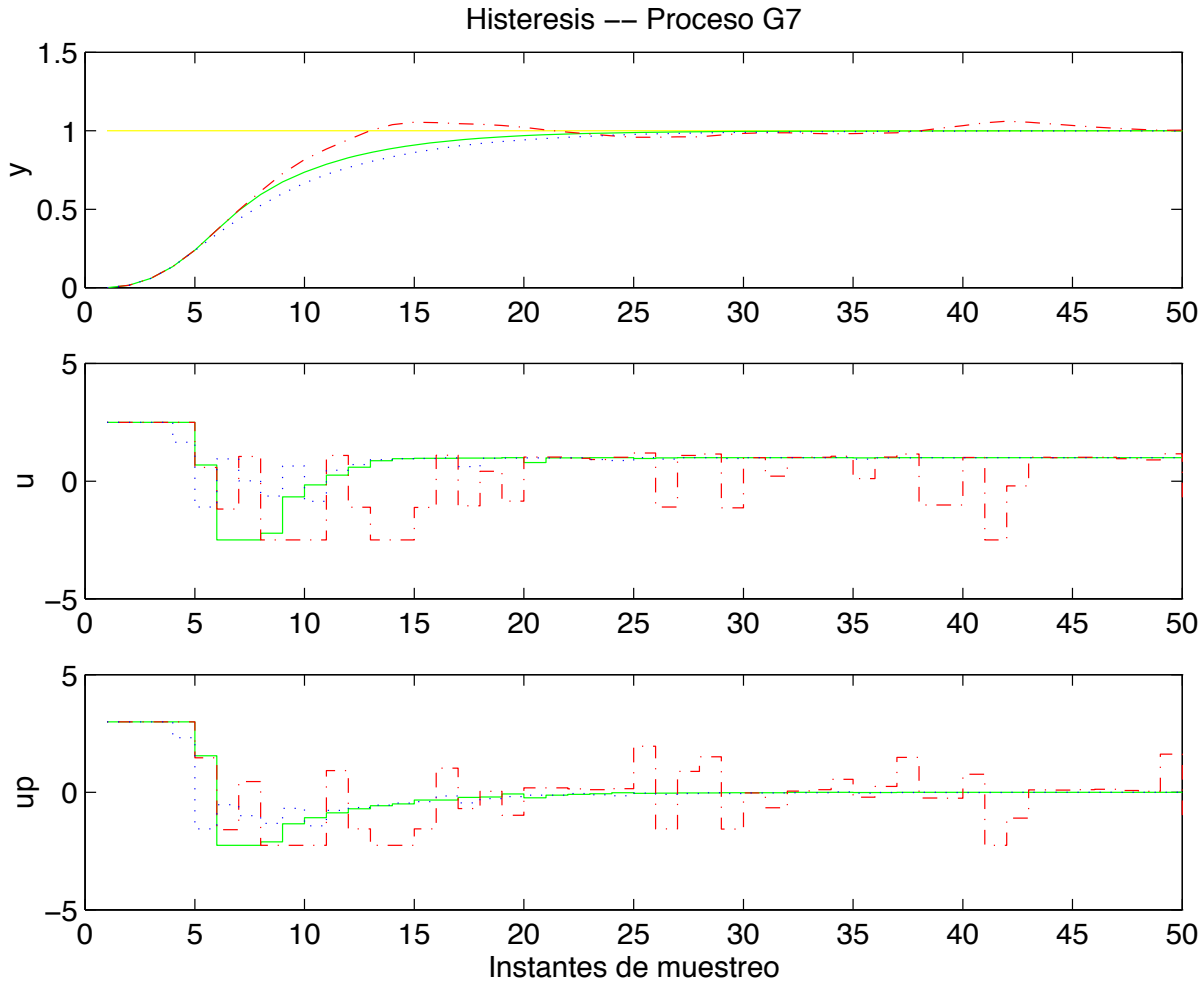


Figura D.49: Control del proceso $G7$ con histéresis en la acción de control. Índice cuadrático y GA (en verde), índice cuadrático y SQP (en rojo), índice modular y GA (en azul).

Bibliografía

[1] P. Albertos and R. Ortega. On generalized predictive control: Two alternative formulations. *Automatica*, 25(5):753–755, 1989.

[2] K.J. Aström and B. Wittenmark. *Adaptive control.* Addison-Wesley, 1989.

[3] J. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proc. Second International Conf.*, 1987.

[4] J.G. Balchen, C. Ljungquist, and S. Strand. State space model predictive control of a multistage electrometalurgical process. In *Model based process control. Proceedings of the IFAC Workshop*, Atlanta, Georgia, USA, June 1988.

[5] R. S. Barrl, B. L Golden, J. P. Kelly, M .G. C Rsende, and W. R. Stewat. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1995.

[6] D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[7] D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993.

[8] Lorenz T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In *International Symposium on Nonlinear Model Predictive Control: Assessment and Future Directions*, Ascona, June 1998.

[9] F.X. Blasco. Controlador predictivo optimizado con algoritmos genéticos (GAGPC). Extensiones. Technical Report , Dept. Ingeniería de Sistemas Computadores y Automática. U. Politécnica de Valencia., 1995.

[10] F.X. Blasco, M. Martínez, J. Senent, and J. Sanchis. Generalized predictive control using genetic algorithms (GAGPC). An application to control of non-linear process with model uncertainty. In Springer, editor, *Methodology and tools in Knowledge-based systems.* 1998.

[11] F.X. Blasco, J.S. Senent, and M. Martínez. Control predictivo generalizado mediante algoritmos genéticos con redistribución de la acción de control. In *Seminario anual de automática y electrónica industrial, SAAEI'96*, pages 615–619, Septiembre 1996.

[12] T. Boulard and A. Baille. A simple greenhouse climate control model incorporating effects of ventilation and evaporative cooling. *Agricultural and Forest Meteorology*, (65):145–157, 1993.

[13] T. Boulard and A. Baille. Modelling of air exchange rate in a greenhouse equipped with continous roof vents. *Journal of Agricultural Engineering Research*, (61):37–48, 1995.

[14] T. Boulard and B. Draoui. Natural ventilation of greenhouse with continous roof vents: Measurements and data analysis. *Journal of Agricultural Engineering Research*, (61):27–36, 1995.

[15] E.F. Camacho. Constrained generalized predictive control. *IEEE Transactions on Automatic Control*, 38(2):327–332, February 1993.

[16] E.F. Camacho and C. Bourdons. *Model Predictive Control in the Process Industry.* Springer, 1995.

[17] E. Cantú-Paz. A summary of research on parallel genetic algorithms. Technical Report 95007, Illinois Genetic Algorithms Laboratory. IlliGAL, July 1995.

[18] R.A. Caruana and J.D. Schaffer. Representation and hidden bias: Gray vs. binary coding. In *6th Int. Conf Machine Learning*, pages 153–161, 1988.

[19] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Accepted for Automatica*, 1997.

[20] H. Chen and F. Allgöwer. A computationally attractive nonlinear predictive control scheme with guaranteed stability for stable systems. *Accepted for Journal of Process Control*, 1998.

[21] S. Chen, S.A. Billings, and P.M. Grant. Non-linear system identification using neural networks. *Int. J. Control*, 51(6):1191–1214, 1990.

[22] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca. Genetic algorithm toolbox. user's guide. Technical report, Dept. Automatic Control and Systems Engineering.

[23] C.M. Chow and D.W. Clarke. Actuator nonlinearities in predictive control. In D. W. Clarke Editor, editor, *Advances in Model-Based Predictive Control*, pages 245–260. Oxford University Press, 1994.

[24] C.M. Chow, A. Kuznetsov, and D.W. Clarke. Application of generalized predictive control to the benchmark paper machine. Technical Report OUEL 2017/94, Department of Engineering Science, April 1994.

[25] D. W. Clarke. Advances in model-based predictive control. In D. W. Clarke Editor, editor, *Advances in Model-Based Predictive Control*, pages 3–21. Oxford University Press, 1994.

[26] D.W. Clarke. Application of generalized predictive control to industrial processes. *IEEE Control Systems Magazine*, 138:49–55, April 1988.

[27] D.W. Clarke. Adaptive generalized predictive controller. Technical Report OUEL 1879/91, Department of Engineering Science, 1991.

[28] D.W. Clarke and C. Mohtadi. Properties of generalized predictive control. *Automatica*, 25(6):859–875, 1989.

[29] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized Predictive Control-Part I. *Automatica*, 23(2):137–148, 1987.

[30] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized Predictive Control-Part II. Extensions and Interpretations. *Automatica*, 23(2):149–160, 1987.

[31] C.R. Cutler and R.B. Hawkins. Constrained multivariable control of a hydrocracker reactor. 1987.

[32] C.R. Cutler and D.L. Ramaker. Dynamic matrix control - a computer control algorithm. In *Proceedings of the joint automatic control conference (JACC)*, San Francisco, CA, 1980.

[33] A. Pérez de Madrid Pablo. *Aplicación de técnicas de programación dinámica a control predictivo basado en modelos.* PhD thesis, Universidad Nacional de Educación a Distancia. Facultad de Ciencias, 1995.

[34] H. Demircioglu and D.W. Clarke. GPC with guaranteed stability properties. *IEEE Proceedings-D*, 139(4):371–379, July 1992.

[35] H. Demircioglu and D.W. Clarke. Generalised predictive control with end-point state weighting. *IEEE Proceedings-D*, 140(4):275–282, July 1993.

[36] A. Díaz, F. Glover, H. M. Ghaziri, J.L. González, M. Laguna, P. Moscato, and F. T. Tseng. *Optimización Heurística y Redes Neuronales.* Paraninfo, 1996.

[37] J.B. Froisy. Model predictive control: past, present and future. *ISA Transactions*, 33:235–243, 1994.

[38] C.E. Garcia and A.M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). *Chemical Eng. Commun.*, 46:73–87, 1986.

[39] C.E. Garcia, D.M. Prett, and M. Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25(3):355–348, 1989.

[40] D.E. Goldberg. *Genetic Algorithms in search, optimization and machine learning.* Addison-Wesley, 1989.

[41] D.E. Goldberg. A comparative analysis of selection schemes used in genetic algorithms. In Gregory Rawlins, editor, *Foundations of Genetic Algorihms*, pages 69–93. Morgan Kaufmann Publishers, 1991.

[42] Vipin Gopal and Lorenz T. Biegler. Large scale inequality constrained optimization and control. *IEEE Control Systems*, pages 59–68, December 1998.

[43] P. Gray, W. Hart, L Painton, C. Phillips, M. Trahan, and J. Wagner. A survey of global optimization methods. Technical report, Sandia National Laboratories. Albuquerque., 1997.

[44] P. Grosdidier, B. Froisy, and M. Hammann. The IDCOM–M controller. In *Model based process control. Proceedings of the IFAC Workshop*, Atlanta, Georgia, USA, June 1988.

[45] F. Herrera, J.L. Verdegay, and M. Lozano. Bioinformática: Algoritmos genéticos. Technical report, Dept. Ciencias de la computación e I.A., Universidad de Granada , 1993.

[46] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms I. Fundamentals.* Springer-Verlag, 1993.

[47] J.H. Holland. *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press, 1975.

[48] K. A. De Jong and W. M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Intelligence Journal*, 5(1):1–26, 1992.

[49] R.M.C. De Keyser. Basic principles of model based predictive control. *1st. European Control Conference*, 1(1):1753–1758, 1991.

[50] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[51] B. Kouvaritakis, J. a. Rossiter, and A. O. T. Chang. Stable generalised predictive control: an algorithm with guaranteed stability. *IEE Proceedings-D*, 139(4):349–362, July 1992.

[52] Alexander G. Kuznetsov. Nonlinear optimization toolbox. Technical Report OUEL 1936/92, Department of Engineering Science, 1992.

[53] Alexander G. Kuznetsov and David W. Clarke. Application of constrained GPC for improving performance of controlled plants. In D. W. Clarke Editor, editor, *Advances in Model-Based Predictive Control*, pages 318–329. Oxford University Press, 1994.

[54] P.L Lee. Direct use of nonlinear models for process control. In *Proc. 4th International conference on chemical process control*, 1991.

[55] J.M. Lemos and E. Mosca. A multipredictor based LQ self-tuning controller. In *IFAC Symp. on identification and system parameter estimation*, York, UK, 1985.

[56] D.A. Linkens and M. Mahfouf. Genralized predictive control (gpc) in clinical anaesthesia. In D. W. Clarke Editor, editor, *Advances in Model-Based Predictive Control*, pages 402–414. Oxford University Press, 1994.

[57] G.P. Liu, V. Kadirkamanathan, and S.A. Billings. Predictive control for non-linear systems using neural networks. *Int. J. Control*, 71(6):1119–1132, 1998.

[58] González Real (Baille M). Manejo integrado de la nebulización y ventilación. *Report Interno Proyecto CICYT*, 1996.

[59] B.R. Maner, F.J. Doyle, B.A. Ogunnaike, and R.K. Pearson. Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order volterra models. *Automatica*, 32(9):1285–1301, 1996.

[60] T. E. Marlin. *Process Control. Designing Processes and Control Systems for Dynamic Performance*. Mc Graw-Hill, 1995.

[61] G.D. Martin, J.M. Cadwell, and T.E. Ayral. Predictive control applications for the petroleum refining industry. *Petroleum refining Conf.*, 1986.

[62] M. Martínez, J. Senent, J. Sanchis, and F.X. Blasco. Control predictivo multivariable de un invernadero optimizado mediante simulated annealing (SA). In *XVII Jornadas de Automática*, Septiembre 1997.

[63] M. Martínez, J.S. Senent, and F.X. Blasco. A comparative study of classical vs genetic algorithm optimization applied in GPC controller. *IFAC World congress*, 1996.

[64] M. Martínez, J.S. Senent, and F.X. Blasco. Generalized predictive control using genetic algorithms (GAGPC). *Engineering applications of artificial intelligence*, 11(3):355–368, 1998.

[65] David Megías, Javer Serrano, and César DePrada. Aspecto de robustez en el GPC. In *Workshop sobre el estado y perspectivas del control predictivo*, Valladolid, Septiembre 1997.

[66] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1996.

[67] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. on Automatic Control*, 1993.

[68] M. Morari, C.E. Garcia, and D.M. Prett. Model predictive control: Theory and practice. In *Model based process control. Proceedings of the IFAC Workshop*, Atlanta, Georgia, USA, June 1988.

[69] M. Morari and J. H. Lee. Model predictive control: past, present and future. In *PSE'97-ESCAPE-7 symposium*, Trondheim, Norway, 1997.

[70] E. Mosca. *Optimal, predictive, and adptive control.* Prentice-Hall Information and System Sciences Series, 1995.

[71] H. Mullenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary computation*, 1(1):25–49, 1993.

[72] K.R. Muske and J.B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.

[73] Numerical Algorithms Group NAG. NAG Fortran Library. Introductory guide. Mark 18. Technical report, 1997.

[74] E.P. Nahas, M.A. Henson, and D.E. Seborg. Nonlinear internal model control strategy for neural network models. *Computer Che. Engng*, 16(12):1039–1057, 1992.

[75] C Onnen, R. Balbuska, U. Kaymak, J.M Sousa, H.B. Verbruggen, and R. Isermann. Genetic algorithms for optimization in predictive control. *Control Engineering Practice*, 5(10):1363–1372, 1997.

[76] A. W. Ordys and David W. Clarke. A state-space description for gpc controllers. *Int. J. Systems Sci.*, 24:1727–1744, 1993.

[77] C. L. Phillips and J. M. Parr. *Signals, systems and transforms.* Prentice Hall, 1995.

[78] János Pinter. Continuous global optimization software: A brief review. *Optima, the Newsletter of the Mathematical Programming Society*, (52), December 1996.

[79] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C: The art of scientific computing. 2nd edition.* Cambridge university press, 1995.

[80] S.J. Qin. An overview of industrial model predictive control technology. In *Proceedings AIchE symposium serie 316*, volume 93, pages 232–256, 1996.

[81] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29(5):1251–1274, 1993.

[82] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.

[83] N.L Ricker, T. Subrahmanian, and T. Sim. Case studies of model predictive control in pulp and paper production. In *Model based process control. Proceedings of the IFAC Workshop*, Atlanta, Georgia, USA, June 1988.

[84] J.A. Rossiter, B. Kouvaritakis, and R.M. Dunnett. Application of generalized predictive control to a boiler turbine unit for electricity generation. *IEE Proceedings-D*, 138:59–67, 1991.

[85] Reuven Y. Rubinstein. *Monte Carlo optimization, simulation and sensitivity of queuing networks.* Wiley, 1986.

[86] J.V. Salcedo, F.X. Blasco, and M. Martínez. Herramienta de desarrollo y análisis de GPC's con aplicaciones industriales. In *Seminario anual de automática y electrónica industrial, SAAEI'97*, Septiembre 1997.

[87] J.V. Salcedo, F.X. Blasco, M. Martínez, and J. Senent. Control predictivo generalizado experimental de proceso inestables. In *Seminario anual de automática y electrónica industrial, SAAEI'98*, September 1998.

[88] J. Sanchis. Diseño de GPC para procesos MIMO. Análisis de las restricciones en los actuadores y aplicación a procesos reales. Technical Report :Doctorado, 6 créditos, Departmento de Ingeniería de Sistemas y Automática, Septiembre 1997.

[89] J. Sanchis, M. Martínez, F.X. Blasco, and J. Senent. Experimentación en control predictivo basado en modelos. prácticas para la enseñanza de controladores predictivos generalizados (GPC's) en la ETSII. In *XVII Jornadas de Automática*, Septiembre 1997.

[90] J. Sanchis, J. Senent, F.X. Blasco, and M. Martínez. Teaching generalized predictive control: Lab practices for control engineers. In *International conference on Control'98*, September 1998.

[91] P.O.M. Scokaert. *Constrained predictive control.* PhD thesis, University of Oxford. Department of Engineering Science, Oxford, UK, 1994.

[92] J.S. Senent. Controlador predictivo generalizado. optimización mediante algoritmos genéticos (GAGPC). Technical Report , Dept. de Ingeniería de Sistemas Computadores y Automática, Universidad Politécnica de Valencia, 1995.

[93] J.S. Senent. *Control predictivo no lineal con horizonte de predicción casi infinitos.* PhD thesis, Universidad Politécnica de Valencia, Valencia, 1998.

[94] J.S. Senent, M. Martínez, F.X. Blasco, and J. Sanchis. MIMO predictive control of temperature and humidity inside a greenhouse using simulated annealing (SA) optimizer of a multicriteria index. In Springer, editor, *Tasks and methods in applied artificial intelligence.* 1998.

[95] George Stephanopoulos. *Chemical process control: An introduction to theory and practice.* Prentice Hall International, 1984.

[96] G. Syswerda. Uniform crossover in genetic algorithms. In J. David Shaffer, editor, *Proc. 3rd International Conf. on Genetic Algorithms.* Kaufmann Publishing, 1989.

[97] H. SZU and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122(3,4):157–162, 1987.

[98] H.A. Taha. *Operations research. An introduction.* Prentice Hall International, 1992.

[99] Gang Tao and Petar V. Kokotovic. *Adaptative control of systems with actuator and sensor nonlinearities.* A Wiley–Interscience publication, 1996.

[100] The Mathworks, Inc. *Optimization toolbox user's guide.* The Mathworks, Inc., 1997.

[101] T.T.C. Tsang and David W. Clarke. Generalized predictive control with input constraints. *IEE Proceedings.*, 135:451–460, November 1988.

[102] Li-Xin Wang. *Adaptative fuzzy systems and control design and stability analysis.* PTR Prentice Hall, 1994.

[103] D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Dept. of Comp. Science. Colorado State University, 1993.

[104] Deniz Yuret and Michael de la Maza. Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *2nd Turkish symposium on artificial intelligence and artificial neural networks*, pages 254–260, 1993.

[105] J. M. Zamarreño. *Identificación y control predictivo basado en modelos mediante red neuronal en espacio de estados.* PhD thesis, Universidad de Valladolid, Valladolid, 1996.