UNIVERSIDAD SIMON BOLIVAR

Sede del Litoral

Departamento de Tecnología Industrial

CONTROL-DESIGN:
UNA HERRAMIENTA PARA EL DISEÑO DE CONTROLADORES

Trabajo de Ascenso
presentado ante la Universidad Simón Bolivar por el profesor:
GUSTAVO ADOLFO SANCHEZ HURTADO

Como requisito para optar a la categoría de:
AGREGADO

Diciembre, 2005

AGRADECIMIENTOS

Quisiera manifestar mi más profundo agradecimiento a las personas quienes directa o indirectamente colaboraron para realizar el presente trabajo de ascenso.

En primer lugar al Prof. Orlando Reyes, por sus ideas, comentarios y asesorías. En segundo lugar a todos los profesores del Departamento de Tecnología Industrial de la Universidad Simón Bolivar, Sede del Litoral, quienes constantemente me animaron a terminar el trabajo a tiempo.

Por último, agradezco a mis padres, a mis hermanos, a mi esposa y a mis hijos por su paciencia y comprensión.

RESUMEN

En este trabajo se presenta una nueva herramienta computacional interactiva llamada CONTROL-DESIGN, la cual permite diseñar controladores para sistemas lineales de manera sencilla pero eficiente. CONTROL-DESIGN consiste básicamente en una interfaz gráfica, realizada mediante el "toolbox" GUIDE de MATLAB®. Puede ser adaptada para resolver una gran cantidad de problemas de diseño realizando modificaciones menores al código fuente, lo cual permite su utilización con fines de investigación y docencia, requiriendo además menos tiempo y esfuerzo con respecto a herramientas actuales. Debido a que el problema de optimización planteado es no-convexo, se utilizan técnicas directas de optimización no-lineal, las cuales no implican derivaciones de la función objetivo. El trabajo, desde un punto de vista más amplio, se enmarca dentro de una linea de investigación centrada en el desarrollo de programas "inteligentes" para el diseño de controladores, utilizando diversos métodos de optimización.

Palabras Clave: Interfaces gráficas, diseño de controladores, optimización.

INDICE GENERAL

1 INTRODUCCION **2**

2 FUNDAMENTOS TEORICOS **6**

2.1 Señales y Sistemas 6

2.2 Topologías de Control 15

2.3 Parametrización de Controladores en $\mathfrak{M}[RH_\infty]$ 20

2.4 Problemas de Optimización Multi-objetivos 26

2.5 Técnicas directas de optimización 27

2.6 GUIDE (Graphic User Interface Design Environment) 34

3 PLANTEAMIENTO DEL PROBLEMA DE DISEÑO **38**

4 CONTROL-DESIGN: ESTRUCTURA y FUNCIONAMIENTO **47**

4.1 Estructura 52

4.2 Funcionamiento: Ejemplos de diseño. 57

4.2.1 Control de un sistema de dos masas y resorte 57

4.2.2 Control de un sistema SISO de orden 4 con acción integral 69

4.2.3 Problema de control $\mathcal{H}_2$ 77

4.3 Información práctica sobre CONTROL-DESIGN 79

5 CONCLUSIONES **80**

5.1 Contribuciones Principales 80

5.2 Lineas futuras de investigación 81

BIBLIOGRAFIA **82**

CAPITULO 1

INTRODUCCION

Diseñar un sistema de control que cumpla con determinados requerimientos puede ser un problema difícil de resolver, cuando no imposible. A menudo esta actividad tiene más de arte e intuición que de método científico.

Por este motivo, numerosos investigadores han desarrollado herramientas computacionales con el fin de resolver estos problemas de la manera más sencilla y eficiente posible. Actualmente estan disponibles una gran cantidad de programas de tipo CACSD (*Computer Aided Control System Design*), los cuales permiten atacar problemas de mayor o menor complejidad: multivariables, con variables discretas, distribuidos, con saturación de actuadores, variantes en el tiempo, susceptibles a diversas fallas, etc.

El término CACSD se refiere en general a una "*tecnología que incluye una gran variedad de herramientas y plataformas computacionales para el diseño de sistemas de control*"(Grubel, 2005). Dicha tecnología se vincula de manera estrecha con diversas disciplinas tales como: teoría de control, modelos matemáticos, métodos numéricos, computación simbólica, optimización, procesamiento de datos, ingeniería de software, interfaces hombre-máquina, etc.

Si repasamos brevemente la historia de las últimas décadas del siglo XX, es posible constatar que ya en 1984 la conocida institución IEEE ("*Institute of Electrical and Electronics Engineer*") publicaba un número especial sobre CACSD en el cual se insistía sobre su gran potencial de evolución (IEEE, 1984). En particular se discutían en esta

publicación aspectos relacionados con arquitectura, integración a diversas plataformas, lenguajes y algoritmos de optimización, muchos de los cuales siguen teniendo vigencia en la actualidad.

Tres años antes, en 1981, se presentó el programa DELIGHT.MIMO (Nye, 1981): un software interactivo para el diseño de controladores multivariables.

En 1983, apareció el artículo "*Controller design for linear multivariable feed-back systems with stable plants, using optimization with inequality constraints*" (Gustafson and Desoer, 1983), en el cual se plantea la posibilidad de utilizar la parametrización de controladores de Youla para reformular el problema de búsqueda de controladores.

Otro de los primeros y más reconocidos trabajos relacionados con CACSD lo constituye (Boyd et al., 1988), en donde se comienza a atacar el problema de un lenguaje formal para codificar los requerimientos y plantear los problemas de diseño. En este mismo año se presentó PROCEED (Chawla et al., 1988): un sistema experto para el diseño de controladores multivariables.

Más recientemente, en (Gu. et al., 1994) se presentó MODCONS: un conjunto de programas de diseño de controladores con múltiples objetivos, con una interfaz gráfica o GUI (Graphic User Interface) bastante primitiva.

Una de las interfaces de diseño más conocidas es sin lugar a dudas SISOTOOL de MATLAB®. Sus limitaciones se presentan en cuanto al planteamiento de requerimientos y el tratamiento de problemas multivariables. En realidad el diseñador debe "entonar" el controlador basandose en diagramas de Bode o Nyquist.

Otra plataforma bastante conocida que permite el diseño de controladores es SLICOT (Benner et al., 1997). Sin embargo es poco interactiva, por cuanto el usuario debe programar una gran cantidad de instrucciones. Se trata en realidad de una plataforma similar en cuanto a su filosofía al programa MATLAB, con la diferencia de ser una distribución de libre uso.

Otro ejemplo lo constituye FRTOOL (DeKeyser, 2001): una GUI que permite diseñar

controladores utilizando técnicas clásicas frecuenciales. El diseñador debe, al igual que en SISOTOOL, seleccionar la posición de los polos/ceros y la ganancia del controlador.

En la Universidad Simón Bolívar se han presentado trabajos de ascenso relacionados con CACSD (Alfonzo, 1992).

En el transcurso de esta investigación se pudo comprobar que curiosamente la literatura revisada no es rica en análisis comparativos de distintas GUIs para el diseño de controladores. Aparentemente los diseñadores se han inclinado por una de las siguientes opciones:

- No utilizar ninguna interfaz. En este caso se trabaja directamente mediante "comandos" con el "prompt" de MATLAB®, por ejemplo.
- Desarrollar una interfaz gráfica adaptada a sus propias necesidades.

La primera opción es la preferida por diseñadores expertos. La segunda no representaría ningun inconveniente si fuera posible utilizar la misma interfaz para múltiples aplicaciones, tomando en cuenta el tiempo y el esfuerzo que debe invertirse en esta tarea. Sin embargo, cualquier diseñador experimentado puede confirmar que en la práctica esto es sumamente dificil. En general, la interfaz no puede ser desarrollada de manera independiente de los métodos y del problema que se desea resover. En consecuencia, es necesario desarrollar una nueva interfaz *ad hoc* para cada nuevo problema.

Ante estas circunstancias, el objetivo principal del presente trabajo es presentar una nueva herramienta computacional llamada CONTROL-DESIGN, la cual permite llevar a cabo los diseños de manera sencilla pero eficiente, por lo cual se adapta perfectamente a los fines de la investigación y la docencia. Dicha interfaz puede ser adaptada a cualquier problema de diseño realizando modificaciones menores al código fuente. De esta forma se requiere menos tiempo y esfuerzo en la fase de diseño con respecto a herramientas actuales.

CONTROL-DESIGN fue creada para ejecutarse sobre la plataforma MATLAB®, buscando un equilibrio entre la capacidad de la GUI (es decir, lo que la misma es capaz de

lograr, la clase de problemas que pueden abordarse) y su sencillez o facilidad de interacción. Es importante insistir en que una "buena" GUI es fundamental para realizar pruebas, análisis, comparaciones entre distintos métodos de búsqueda, etc.

El presente informe está dividido en 5 Capítulos. En el Capítulo 2 se exponen los fundamentos teóricos, definiciones, teoremas y algoritmos que serviran de base para la descripción de la interfaz. El lector familiarizado con la teoría de sistemas lineales, la parametrización de controladores estabilizantes y las técnicas directas de optimización puede omitirlo. En el Capítulo 3 se realiza un planteamiento detallado del problema matemático que se intenta resolver. En el Capítulo 4 se describe la estructura y funcionamiento de CONTROL-DESIGN con el fin de que cualquier interesado pueda utilizar la presente versión como "plantilla" para desarrollar su propia interfaz. Finalmente, en el Capítulo 5 se presentan algunas conclusiones que se derivan de los resultados obtenidos.

CAPITULO 2

FUNDAMENTOS TEORICOS

Antes de comenzar a describir la estructura y el funcionamiento de CONTROL-DESIGN, es conveniente recordar algunos resultados de la teoría moderna de control, necesarios para la formulación matemática del problema de optimización.

2.1 Señales y Sistemas

Espacio $H_2^{p\times m}$ y Norma $\mathcal{H}_2$

Notación: Se denota $L_2^{p\times m}(-\infty,\infty)$ al Espacio Lineal compuesto por las señales $f:\mathcal{R}\rightarrow\mathfrak{C}^{p\times m}$ continuas y determinísticas, tales que:

$$\int_{-\infty}^{+\infty} traza[f^*(t)f(t)]dt<\infty \tag{2.1}$$

Theorem 1 *La función* $\|\cdot\|_2 : L_2^{p\times m}(-\infty,\infty)\rightarrow\mathcal{R}_0^+$ *tal que*

$$\|f\|_2=\sqrt{\int_{-\infty}^{+\infty} traza[f^*(t)f(t)]dt} \tag{2.2}$$

$\forall f \in L_2^{p\times m}$ *define una norma sobre* $L_2^{p\times m}(-\infty,\infty)$.

Nota: La norma $\|f\|_2$, $f \in L_2^{p\times m}(-\infty,\infty)$, puede ser interpretada como la "Energía" total contenida en f. Para que esta cantidad sea finita, debe cumplirse

$$\lim_{t\to\infty} traza[f^*(t)f(t)] = 0 \tag{2.3}$$

Notación: Se denota $\hat{L}_2^{p\times m}(j\mathcal{R})$ al Espacio de Hilbert compuesto por las funciones matriciales complejas $\hat{f} : j\mathcal{R} \to \mathfrak{C}^{p\times m}$, dotado del producto interno

$$\left\langle \hat{f}_1, \hat{f}_2 \right\rangle = \int_{-\infty}^{+\infty} traza[\hat{f}_1^*(j\omega)\hat{f}_2(j\omega)]d\omega \tag{2.4}$$

Definition 2 *El operador lineal acotado* $\Gamma : L_2^{p\times m}(-\infty,\infty) \to \hat{L}_2^{p\times m}(j\mathcal{R})$ *definido por*

$$\Gamma(f) = \hat{f}(j\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt \tag{2.5}$$

se denomina Operador Transformada de Fourier.

Nota: Γ es un operador unitario e invertible, esto es, Γ define un isomorfismo entre $L_2^{p\times m}(-\infty,\infty)$ y $\hat{L}_2^{p\times m}(j\mathcal{R})$.

Notación: Se denota $\mathfrak{C}^+$ al semi-plano complejo abierto derecho,

$$\mathfrak{C}^+ \equiv \{c \in \mathfrak{C}; Re(c) > 0\}$$

Se denota $\overline{\mathfrak{C}}^+$ al semi-plano complejo cerrado derecho, esto es

$$\overline{\mathfrak{C}}^+ \equiv \{c \in \mathfrak{C}; Re(c) \geq 0\}$$

Notación: Se denota $H_2^{p\times m}$al espacio de funciones matriciales complejas

$$\hat{f} : \overline{\mathfrak{C}}^+ \to \mathfrak{C}^{p\times m}$$

tales que:

- $\hat{f}$ es analítica en $\overline{\mathfrak{C}}^{+}$.
- Para cualquier $\omega \in \mathcal{R}$, excepto en un conjunto finito $\{\omega_1, \omega_2, ...\omega_N\}$, se cumple:

$$\hat{f}(j\omega) = \lim_{\sigma \to 0^+} \hat{f}(\sigma + j\omega) \tag{2.6}$$

- La siguiente cantidad es finita

$$\sup_{\sigma \geq 0} \int_{-\infty}^{+\infty} traza[\hat{f}^*(\sigma + j\omega)\hat{f}(\sigma + j\omega)]d\omega \tag{2.7}$$

Theorem 3 *La función* $\|\cdot\|_2 : H_2^{p\times m} \to \mathcal{R}_0^+$ *tal que*

$$\left\|\hat{f}\right\|_2 = \sqrt{\sup_{\sigma \geq 0} \frac{1}{2\pi} \int_{-\infty}^{+\infty} traza[\hat{f}^*(\sigma + j\omega)\hat{f}(\sigma + j\omega)]d\omega} \tag{2.8}$$

$\forall f \in H_2^{p\times m}$ *, define una norma sobre* $H_2^{p\times m}$*. En este trabajo, esta norma en particular será denotada* $\mathcal{H}_2$*.*

Una función matricial compleja $\hat{f} : \overline{\mathfrak{C}}^{+} \to \mathfrak{C}^{p\times m}$ se denomina racional, si cada uno de sus elementos $\hat{f}_{ij}(s)$ es de la forma:

$$\hat{f}_{ij}(s) = \frac{b_{nu}s^{nu} + b_{nu-1}s^{nu-1} + .. + b_1 s + b_0}{s^{nd} + a_{nd-1}s^{nu-1} + .. + a_1 s + a_0} \tag{2.9}$$

Si $\forall \hat{f}_{ij}(s)$ todos los coeficientes son reales, $\hat{f}$ es llamada real racional. El conjunto de funciones matriciales $\hat{f} : \overline{\mathfrak{C}}^{+} \to \mathfrak{C}^{p\times m}$ reales racionales se denota $\mathcal{R}(s)^{p\times m}$. Si $\forall \hat{f}_{ij}(s)$, $nd \geq nu$, $\hat{f}$ es llamada propia. Si $\forall \hat{f}_{ij}(s)$, $nd > nu$, $\hat{f}$ es llamada estrictamente propia

Theorem 4 *Una función real racional* $\hat{f} : \overline{\mathfrak{C}}^{+} \to \mathfrak{C}^{p\times m}$ *pertenece a* $H_2^{p\times m}$ *si y solo si:*

- $\hat{f}$ *es estrictamente propia.*
- $\hat{f}$ *no posee polos en* $\overline{\mathfrak{C}}^{+}$.

Notación: Se denota $RH_2^{p\times m}$ al sub-espacio de funciones matriciales reales racionales pertenecientes a $H_2^{p\times m}$.

Definition 5 *El operador lineal acotado* $\Lambda : L_2^{p\times m}[0,\infty) \rightarrow H_2^{p\times m}$ *definido por:*

$$\Lambda(f) = \hat{f}(s) = \int_0^{+\infty} f(t)e^{-st}dt \tag{2.10}$$

se denomina Operador Transformada de Laplace.

Notas:

- Λ define un isomorfismo entre $L_2^{p\times m}[0,\infty)$ y $H_2^{p\times m}$.
- $L_2^{p\times m}[0,\infty) \subset L_2^{p\times m}(-\infty,\infty) \Rightarrow H_2^{p\times m} \subset \hat{L}_2^{p\times m}(j\mathcal{R})$

Definition 6 *El operador retardo* $S_\tau : L_2^{p\times m}(-\infty,\infty) \rightarrow L_2^{p\times m}(-\infty,\infty)$ *viene definido por*

$$y = S_\tau u \;\Leftrightarrow y(t) = u(t-\tau) \tag{2.11}$$

Ejemplo: En la Figura 2-1 se muestra un ejemplo de la acción del operador retardo.

Definition 7 *Un operador* $G : L_2^{p\times m}(-\infty,\infty) \rightarrow L_2^{p\times m}(-\infty,\infty)$ *se dice invariante en el tiempo si y solo si:*

$$GS_\tau = S_\tau G \tag{2.12}$$

para todo $\tau \geq 0$.

Definition 8 *El operador de truncado* $P_\tau : L_2^{p\times m}(-\infty,\infty) \rightarrow L_2^{p\times m}(-\infty,\infty)$ *viene definido por*

$$y = P_\tau u \Leftrightarrow y(t) = \begin{cases} u(t)\ ,\ t \leq \tau \\ 0\ ,\ t > \tau \end{cases} \tag{2.13}$$

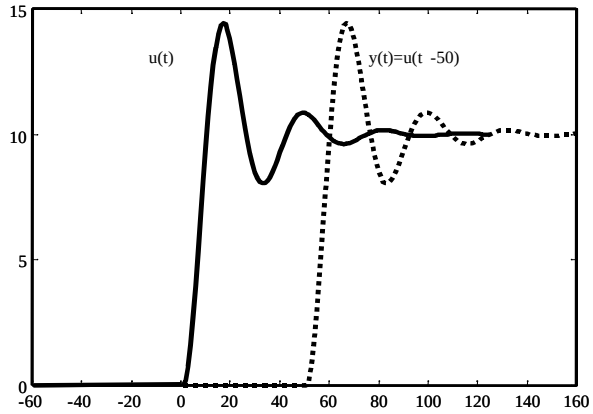

Figura 2-1: Ejemplo de la acción del operador retardo.

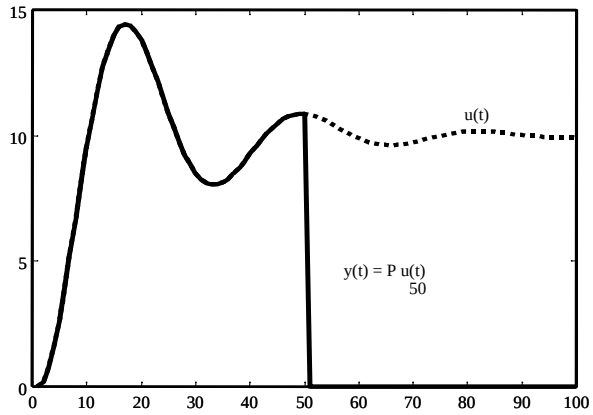

Figura 2-2: Ejemplo de la acción del operador truncado.

Ejemplo: En la Figura 2-2 se muestra un ejemplo de la acción del operador de truncado.

Definition 9 *El operador* $G : L_2^{p\times m}(-\infty,\infty) \rightarrow L_2^{p\times m}(-\infty,\infty)$ *se denomina causal si y solo si:*

$$P_\tau G P_\tau = P_\tau G \tag{2.14}$$

para todo $\tau \in \mathcal{R}$

Nota: Para un operador causal, la salida y antes de un tiempo τ no se ve afectada por la entrada u posterior a dicho tiempo.

Theorem 10 *Sea el operador* $G : L_2^{p\times m}(-\infty,\infty) \rightarrow L_2^{p\times m}(-\infty,\infty)$ *invariante en el tiempo.* G *es causal si y solo si para* $\tau = 0$

$$P_\tau G P_\tau = P_\tau G \Leftrightarrow P_0 G P_0 = P_0 G \tag{2.15}$$

Corollary 11 *Un operador lineal* $G : L_2^{p\times m}(-\infty,\infty) \rightarrow L_2^{p\times m}(-\infty,\infty)$ *invariante en el tiempo es causal si y solo si:*

$$u \in L_2^{p\times m}[0,\infty) \Rightarrow Gu \in L_2^{p\times m}[0,\infty) \tag{2.16}$$

Espacio $H_\infty^{p\times m}$ y Norma $\mathcal{H}_\infty$

Notación: Se denota $L_\infty^{p\times m}(-\infty,\infty)$ al Espacio Lineal de señales continuas y determinísticas, compuesto por las funciones $f:\mathcal{R}_0^+\rightarrow\mathfrak{C}^{p\times m}$ tales que:

$$\underset{t\in(-\infty,\infty)}{ess\sup}\ \sqrt{traza[f^*(t)f(t)]}<\infty \tag{2.17}$$

donde $ess\sup\sqrt{traza[f^*(t)f(t)]}<\eta$ equivale a $\sqrt{traza[f^*(t)f(t)]}<\eta$ excepto en un conjunto de medida cero, es decir, para un conjunto finito de instantes $\{t_i;i=1,2,...,N\}$.

Theorem 12 *La función* $\|\cdot\|_\infty:L_\infty^{p\times m}(-\infty,\infty)\rightarrow\mathcal{R}_0^+$ *tal que*

$$\|f\|_\infty=\underset{t\in(-\infty,\infty)}{ess\sup}\ \sqrt{traza[f^*(t)f(t)]} \tag{2.18}$$

$\forall f\in L_\infty^{p\times m}$ *es una norma sobre* $L_\infty^{p\times m}(-\infty,\infty)$

Nota: La norma $\|f\|_\infty$ contiene información correspondiente al "Módulo Máximo" de la señal f para $t\in(-\infty,\infty)$.

Notación: Se denota $H_\infty^{p\times m}$al Espacio de funciones matriciales $\hat{G}:\overline{\mathfrak{C}}^+\rightarrow\mathfrak{C}^{p\times m}$ tales que:

- $\hat{G}$ es analítica en $\overline{\mathfrak{C}}^+$.
- Para cualquier $\omega\in\mathcal{R}$, excepto en un conjunto finito $\{\omega_1,\omega_2,...\omega_N\}$, se cumple:

$$\hat{G}(j\omega)=\lim_{\sigma\rightarrow0^+}\hat{G}(\sigma+j\omega) \tag{2.19}$$

- La siguiente cantidad es finita:

$$\underset{s\in\overline{\mathfrak{C}}^+}{ess\sup}\ \overline{\sigma}[\hat{G}(s)] \tag{2.20}$$

donde $\overline{\sigma}$ representa el máximo valor singular de la matriz $\hat{G}(s)$, es decir,

$$\overline{\sigma}[\hat{G}(s)]=\max_i\sqrt{\lambda_i[\hat{G}(s)^*\hat{G}(s)]} \tag{2.21}$$

con,

$$\lambda_i[\hat{G}(s)^*\hat{G}(s)] \equiv \text{i-ésimo autovalor de } \hat{G}(s)^*\hat{G}(s)$$

Theorem 13 *La función* $\|\cdot\|_\infty : H_\infty^{p\times m} \to \mathcal{R}_0^+$ *tal que*

$$\left\|\hat{G}\right\|_\infty = \sup_{\omega\in\mathcal{R}} \overline{\sigma}[\hat{G}(j\omega)] \tag{2.22}$$

$\forall \hat{G} \in H_\infty^{p\times m}$ *es una norma sobre* $H_\infty^{p\times m}$*. En este trabajo, esta norma en particular será denotada* $\mathcal{H}_\infty$*.*

Ejemplo: Suponga $u \in H_2^{n_u\times 1}$. Entonces, $\hat{G} \in H_\infty^{n_y\times n_u}$ se comporta como un operador lineal multiplicativo de $H_2^{n_u\times 1}$, esto es, $y = \hat{G}u \in H_2^{n_y\times 1}$. El operador $\hat{G}$ es comunmente llamado "matriz de transferencia" de $H_2^{n_u\times 1}$ a $H_2^{n_y\times 1}$.

Notas:

- Utilizando el operador Transformada de Laplace, la matriz $\hat{G} : \overline{\mathfrak{C}}^+ \to \mathfrak{C}^{p\times m}$ tal que $\hat{G} \in H_\infty^{p\times m}$ define un operador lineal acotado y causal G de $L_2^{m\times 1}[0,\infty) \to L_2^{p\times 1}[0,\infty)$ mediante
$$G = \Lambda^{-1}\hat{G}\Lambda \tag{2.23}$$

- Una matriz $\hat{G} \in \mathcal{R}(s)^{p\times m}$, pertenece a $H_\infty^{p\times m}$ si y solo si $\hat{G}$ es propia y no posee *polos* en $\overline{\mathfrak{C}}^+$.El conjunto de funciones matriciales racionales pertenecientes a $H_\infty^{p\times m}$ se denota $RH_\infty^{p\times m}$.

Sistemas LTI

Existen varias formas de representar un sistema lineal, causal, invariante en el tiempo (LTI) y estable, como el representado en la Figura 2-3 :

- Mediante un operador lineal acotado G de $L_2^{m\times 1}[0,\infty) \to L_2^{p\times 1}[0,\infty)$.
- Mediante una función matricial racional $\hat{G}$ perteneciente a $RH_\infty^{p\times m}$
- Mediante un par de ecuaciones de estado,

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{2.24}$$

donde,

- u, de dimensión m, es el vector de señales de entrada.
- y, de dimensión p, es el vector de señales de salida.
- x, de dimensión n, es el vector de estados del sistema.
- $A \in \mathcal{R}^{n\times n}$, $B \in \mathcal{R}^{n\times m}$, $C \in \mathcal{R}^{p\times n}$y $D \in \mathcal{R}^{p\times m}$

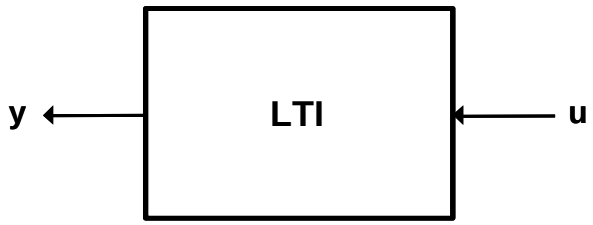

Figura 2-3: Ejemplo de Sistema LTI

En el último caso, la función matricial racional $\hat{G} \in RH_\infty^{p\times m}$ asociada al sistema viene dada por,

$$\hat{G} = C(sI - A)^{-1}B + D \tag{2.25}$$

Recuerde igualmente que,

- El par (A, B) se dice controlable si y solo si la matriz de controlabilidad,

$$\begin{pmatrix} B & AB & \cdots & A^{n-1}B \end{pmatrix} \tag{2.26}$$

es de rango n.

- El par (A, C) se dice observable si y solo si la matriz de observabilidad,

$$\begin{pmatrix} C \\ CA \\ . \\ . \\ CA^{n-1} \end{pmatrix} \tag{2.27}$$

  es de rango n.

- La representación en espacios de estado en (2.24) se dice *mínima* si (A, B) es controlable y (A, C) es observable.

2.2 Topologías de Control

Una de las principales decisiones que debe tomar el diseñador de un controlador automático, es seleccionar la *topología* de control más adecuada tomando en cuenta de una parte los requerimientos del usuario y de la otra la disponibilidad y características de actuadores y sensores.

Ejemplo: En la Figura 2-4 se muestra la topología correspondiente a un regulador de un grado de libertad a lazo cerrado.

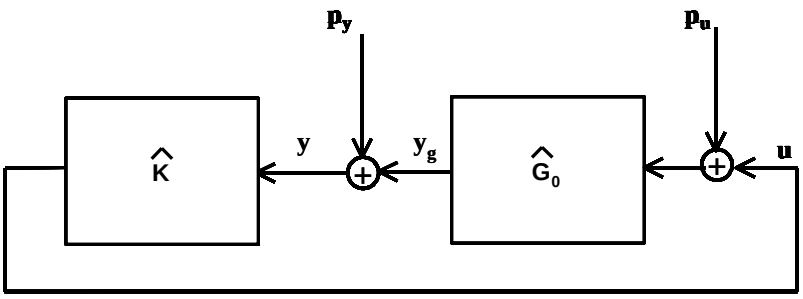

Figura 2-4: Regulador de un grado de libertad a lazo cerrado.

Para este ejemplo,

- y_g , de dimensión n_y, es el vector de salidas del sistema "sin perturbación".
- y , de dimensión n_y, es el vector de salidas medidas perturbadas (sensores físicos).
- u , de dimensión n_u, es el vector de entradas manipulables (actuadores).
- p_u , de dimensión n_u, es una perturbación o ruido que actúa de forma aditiva a la entrada del sistema.
- p_y , de dimensión n_y, es una perturbación que actúa de forma aditiva a la salida del sistema.
- $\hat{G}_0 \in \mathcal{R}(s)^{n_y \times n_u}$, es el modelo lineal del sistema.
- $\hat{K} \in \mathcal{R}(s)^{n_u \times n_y}$, es el modelo lineal del controlador.

Suponga que se desea regular el vector de salidas y_g, tratando de minimizar la energía necesaria (proporcional a u) ante las perturbaciones p_u y p_y. Mediante algunas sencillas modificaciones, es posible redibujar este esquema tal como se muestra en la Figura 2-5, donde:

- w , de dimensión n_w, es el vector de entradas exógenas (ruidos, perturbaciones, referencias, etc).
- z , de dimensión n_z, es el vector de salidas que se desean controlar (errores, señales de control , etc).
- u, de dimensión n_u , es el vector de señales manipulables.
- y, de dimensión n_y , es el vector de salidas medidas.
- $\hat{G} \in \mathcal{R}(s)^{(n_w+n_u)\times(n_z+n_y)}$, es el modelo lineal *generalizado* de la planta.
- $\hat{K} \in \mathcal{R}(s)^{n_u \times n_y}$, es el modelo lineal del controlador.

Así, es posible escribir el siguiente conjunto de ecuaciones:

$$y_g = z_1 = \hat{G}_0(w_1 + u) \tag{2.28}$$

$$z_2 = u \tag{2.29}$$

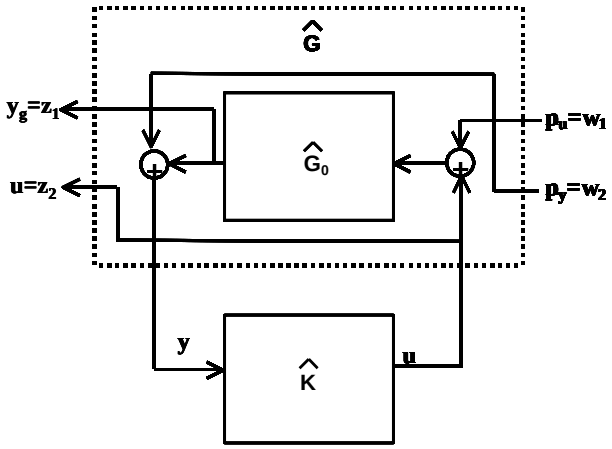

Figura 2-5: Otra forma de representar un regulador de un grado de libertada lazo cerrado.

$$y = w_2 + \hat{G}_0(w_1 + u) \tag{2.30}$$

o directamente en forma matricial:

$$\begin{pmatrix} z_1 \\ z_2 \\ y \end{pmatrix} = \begin{pmatrix} \hat{G}_0 & 0 & \hat{G}_0 \\ 0 & 0 & 1 \\ \hat{G}_0 & 1 & \hat{G}_0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ u \end{pmatrix} \tag{2.31}$$

Note que $\hat{G}$ puede expresarse como:

$$\hat{G} = \begin{pmatrix} \hat{G}_{zw} & \hat{G}_{zu} \\ \hat{G}_{yw} & \hat{G}_{yu} \end{pmatrix} \tag{2.32}$$

donde:

- $\hat{G}_{zw}$ es la función de transferencia de $w \rightarrow z$.
- $\hat{G}_{zu}$ es la función de transferencia de $u \rightarrow z$.
- $\hat{G}_{yw}$ es la función de transferencia de $w \rightarrow y$.
- $\hat{G}_{yu}$ es la función de transferencia de $u \rightarrow y$.

con,

$$\hat{G}_{zw} = \begin{pmatrix} \hat{G}_0 & 0 \\ 0 & 0 \end{pmatrix} \tag{2.33}$$

$$\hat{G}_{zu} = \begin{pmatrix} \hat{G}_0 \\ 1 \end{pmatrix} \tag{2.34}$$

$$\hat{G}_{yw} = \begin{pmatrix} \hat{G}_0 & 1 \end{pmatrix} \tag{2.35}$$

$$\hat{G}_{yu} = \left(\hat{G}_0\right) \tag{2.36}$$

De manera general, cualquier configuración o topología de control mediante realimentación, puede ser representada como se muestra en la Figura 2-6, donde $\hat{G}$ es el modelo *generalizado* de la planta, el cual usualmente puede contener factores arbitrarios de ponderación de las entradas y salidas, introducidos por el diseñador.

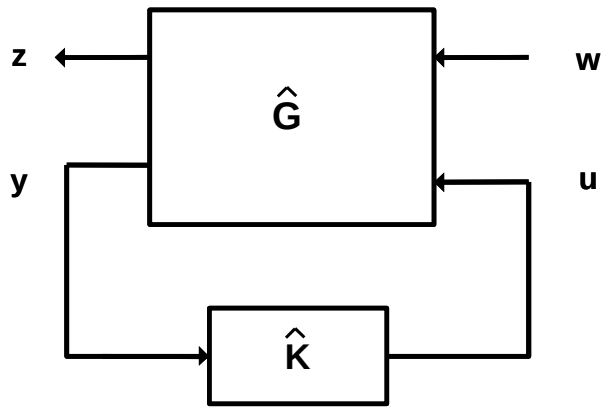

Figura 2-6: Planta Generalizada a Lazo Cerrado

Ahora bien, para el sistema realimentado de la Figura 2-6, es posible escribir la siguiente ecuación de realimentación:

$$u = \hat{K}y \tag{2.37}$$

Un cálculo sencillo permite expresar, a partir de la Ecuación 2.37, la relación a lazo

cerrado entre el vector de salidas controladas z y el vector de perturbaciones w:

$$z = \left[\hat{G}_{zw} + \hat{G}_{zu}\hat{K}(I_{n_y \times n_u} - \hat{G}_{yu}\hat{K})^{-1}\hat{G}_{yw}\right] w \tag{2.38}$$

La Ecuación 2.38 solo tiene sentido siempre y cuando exista $\omega \in \mathcal{R}$ tal que,

$$det[I_{n_y \times n_u} - \hat{G}_{yu}(j\omega)\hat{K}(j\omega)] \neq 0 \tag{2.39}$$

Si la Ecuación 2.39 se cumple, se dice que la conexión de $\hat{G}$ y $\hat{K}$ está *bien expuesta* (*well-posed*). Por otra parte, la expresión entre corchetes de la Ecuación 2.38 se conoce como LFT (Linear Fractional Transformation) inferior y puede denotarse como,

$$\underline{S}(\hat{G}, \hat{K}) = \hat{G}_{zw} + \hat{G}_{zu}\hat{K}(I_{n_y \times n_u} - \hat{G}_{yu}\hat{K})^{-1}\hat{G}_{yw}$$

de forma que a lazo cerrado se tiene,

$$z = \underline{S}(\hat{G}, \hat{K})w = \hat{G}_{zw_{cl}}w \tag{2.40}$$

2.3 Parametrización de Controladores en $\mathfrak{M}[RH_\infty]$

En esta sección, suponga que la matriz de transferencia $\hat{G}$ de la interconexión mostrada en la Figura 2-6 caracterizada por $\underline{S}(\hat{G},\hat{K})$, se encuentre asociada con una representación *mínima* en espacio de estado,

$$\begin{cases} \dot{x} = Ax + B_1 w + B_2 u \\ z = C_1 x + D_{11} w + D_{12} u \\ y = C_2 x + D_{21} w + D_{22} u \end{cases} \tag{2.41}$$

Suponga a su vez que el controlador $\hat{K}$ esté asociado con una representación *mínima* en espacio de estado,

$$\begin{cases} \dot{x}_K = A_K x_K + B_K y \\ u = C_K x_K + D_K y \end{cases} \tag{2.42}$$

Entonces, $\underline{S}(\hat{G},\hat{K})$ estará asociada con las siguientes ecuaciones de estado:

$$\begin{cases} \begin{pmatrix} \dot{x} \\ \dot{x}_K \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & A_K \end{pmatrix} \begin{pmatrix} x \\ x_K \end{pmatrix} + \begin{pmatrix} B_2 & 0 \\ 0 & B_K \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} + \begin{pmatrix} B_1 \\ 0 \end{pmatrix} w \\ z = \begin{pmatrix} C_1 & 0 \end{pmatrix} \begin{pmatrix} x \\ x_K \end{pmatrix} + \begin{pmatrix} D_{12} & 0 \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} + D_{11} w \end{cases} \tag{2.43}$$

donde,

$$\begin{pmatrix} I & -D_K \\ -D_{22} & I \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} 0 & C_K \\ C_2 & 0 \end{pmatrix} \begin{pmatrix} x \\ x_K \end{pmatrix} + \begin{pmatrix} 0 \\ D_{21} \end{pmatrix} w \tag{2.44}$$

Definition 14 *Se dice que la interconexión $\underline{S}(\hat{G},\hat{K})$ es internamente estable si y solo si, cualquiera que sean las condiciones iniciales del sistema $x(0) \neq 0$ y $x_K(0) \neq 0$ se tiene,*

$$\lim_{t\to\infty} x(t) = 0 \tag{2.45}$$

$$\lim_{t\to\infty} x_K(t) = 0 \tag{2.46}$$

para $w = 0$. *Igualmente, se dice que* $\underline{S}(\hat{G}, \hat{K})$ *es exponencialmente estable si* $\exists c_1, c_2 > 0$ *tales que:*

$$\|x(t)\| \leq c_1 e^{-c_2 t} \|x(0)\| \tag{2.47}$$

$$\|x_K(t)\| \leq c_1 e^{-c_2 t} \|x_K(0)\| \tag{2.48}$$

para $w = 0$.

Theorem 15 *La interconexión* $\underline{S}(\hat{G}, \hat{K})$ *es internamente estable si y solo si se cumple de manera simultanea:*

- *La matriz,*

$$I - D_{22} D_K \tag{2.49}$$

  es invertible, o de forma equivalente $\underline{S}(\hat{G}, \hat{K})$ *bien expuesta.*

- *Todos los autovalores de la matriz*

$$A_{cl} = \begin{pmatrix} A & 0 \\ 0 & A_K \end{pmatrix} \tag{2.50}$$

  tienen parte real estrictamente negativa, es decir, A_{cl} *es Hurwitz.*

El requisito fundamental a la hora de diseñar un controlador automático es la estabilidad interna del sistema a lazo cerrado. De las Ecuaciones 2.49 y 2.50, se puede inferir que no es evidente, en el caso general, seleccionar $\hat{K}$ de forma que $\underline{S}(\hat{G}, \hat{K})$ sea internamente estable.

No obstante, existe un resultado fundamental dentro de la Teoría de Control de Sistemas Lineales (Youla et al., 1976) que permite *parametrizar* el espacio de *todos* los controladores lineales estabilizantes. Antes de presentar este resultado, es necesario recordar algunas definiciones y teoremas fundamentales.

Notación: Se denota $\mathfrak{M}[\mathcal{R}(s)]$ al conjunto de funciones matriciales complejas racionales de *todas* las dimensiones posibles. Se denota $\mathfrak{M}[RH_\infty]$ al conjunto de funciones matriciales complejas de *todas* las dimensiones posibles, pertenecientes a RH_∞.

Definition 16 *Sean* $M, N \in \mathfrak{M}[\mathcal{R}(s)]$, *de dimensiones adecuadas. Suponga* $D \in \mathfrak{M}[\mathcal{R}(s)]$ *matriz cuadrada. Se dice que:*

- *D es un divisor derecho (resp. izquierdo) de M si y solo si,* $\exists Q \in \mathfrak{M}[\mathcal{R}(s)]$ *tal que* $M = QD$ *(resp.* $M = DQ$*).*

- *D se denomina Máximo Común Divisor Derecho (resp. Máximo Común Divisor Izquierdo) si y solo si:*

  i) *D es un divisor derecho (resp. izquierdo) de M y N.*

  ii) *Todo* $A \in \mathfrak{M}[\mathcal{R}(s)]$ *divisor derecho (resp. izquierdo) de M y N es igualmente divisor derecho (resp. izquierdo) de D.*

Definition 17 *Sean* $M, N \in \mathfrak{M}[\mathcal{R}(s)]$*. M y N se denominan coprimas derechas (resp. izquierdas) si y solo si, todo Máximo Común Divisor Derecho (resp. Máximo Comun Divisor Izquierdo) D y su inverso* D^{-1} *son estables y propios.*

Theorem 18 *(Identidad de Bezout) Las matrices* $M, N \in \mathfrak{M}[\mathcal{R}(s)]$ *son coprimas derechas (resp. izquierdas) si y solo si* $\exists X, Y \in \mathfrak{M}[\mathcal{R}(s)]$ *tales que* $XM + YN = I$ *(resp.* $MX + NY = I$*).*

Definition 19 *Sea* $\hat{G}_{yu} \in \mathfrak{M}[\mathcal{R}(s)]$*. Se dice que* $\hat{G}_{yu}$ *admite una doble factorización coprima sobre* $\mathfrak{M}(RH_\infty)$ *si existen:*

- $N_r, D_r \in \mathfrak{M}(RH_\infty)$ *tales que:*

$$\hat{G}_{yu} = N_r D_r^{-1} \tag{2.51}$$

- N_l,$D_l \in \mathfrak{M}(RH_\infty)$ *tales que:*

$$\hat{G}_{yu} = D_l^{-1} N_l \tag{2.52}$$

Además, si $\hat{G}_{yu}$ *admite una doble factorización coprima sobre* $\mathfrak{M}(RH_\infty)$ *siempre existen* $X_r, Y_r, X_l, Y_l \in \mathfrak{M}(RH_\infty)$ *tales que:*

$$\begin{pmatrix} Y_r & X_r \\ -N_l & D_l \end{pmatrix} \begin{pmatrix} D_r & -X_l \\ N_r & Y_l \end{pmatrix} = I \tag{2.53}$$

La Ecuación 2.53 se conoce como "Identidad Generalizada de Bezout" sobre $\mathfrak{M}(RH_\infty)$

Ejemplo: Sea $\hat{G}_{yu} \in \mathcal{R}(s)^{2\times 2}$,

$$\hat{G}_{yu} = \begin{pmatrix} \frac{4}{s^2+3s-18} & \frac{s+1}{s^2+3s-18} \\ \frac{10s+20}{s^2+3s-18} & \frac{50}{s^2+3s-18} \end{pmatrix}$$

Mediante algunas multiplicaciones es posible demostrar que $\hat{G}_{yu}$ admite una doble factorización coprima sobre $\mathfrak{M}(RH_\infty)$ con,

$$N_r = \begin{pmatrix} 0 & \frac{1}{s+4} \\ \frac{10}{s+3} & 0 \end{pmatrix}$$

$$D_r = \begin{pmatrix} \frac{s+1}{s+3} & \frac{-5}{s+4} \\ \frac{-4}{s+3} & \frac{s+2}{s+4} \end{pmatrix}$$

$$N_l = N_r$$

$$D_l = \begin{pmatrix} \frac{s+2}{s+4} & \frac{-0.4}{s+4} \\ \frac{-50}{s+3} & \frac{s+1}{s+3} \end{pmatrix}$$

$$X_r = \begin{pmatrix} \frac{20s+70}{s^2+7s+12} & \frac{2.4s+7.6}{s^2+7s+12} \\ \frac{24s+92}{s^2+7s+12} & \frac{1.6s+5.6}{s^2+7s+12} \end{pmatrix}$$

$$Y_r = \begin{pmatrix} \frac{s+5}{s+3} & \frac{5}{s+4} \\ \frac{4}{s+3} & \frac{s+6}{s+4} \end{pmatrix}$$

$$X_l = Xr$$

$$Y_l = \begin{pmatrix} \frac{s+6}{s+4} & \frac{0.4}{s+4} \\ \frac{50}{s+3} & \frac{s+5}{s+3} \end{pmatrix}$$

Consideremos nuevamente el problema de seleccionar $\hat{K}$ de forma que la interconexión $\underline{S}(\hat{G}, \hat{K})$ de la Figura 2-6 sea internamente estable. El siguiente teorema (Youla et al., 1976) permite *parametrizar* el conjunto de controladores tales que $\underline{S}(\hat{G}, \hat{K})$ sea internamente estable.

Theorem 20 *Sea la interconexión mostrada en la Figura 2-6, representada por* $\underline{S}(\hat{G}, \hat{K})$, *con*

$$\hat{G} = \begin{pmatrix} \hat{G}_{zw} & \hat{G}_{zu} \\ \hat{G}_{yw} & \hat{G}_{yu} \end{pmatrix}$$

Suponga que $\hat{G}_{yu}$ *admite una doble factorización coprima sobre* $\mathfrak{M}(RH_\infty)$ *como la mostrada en (2.53).*

Entonces, $\underline{S}(\hat{G}, \hat{K})$ *es internamente estable, si y solo si:*

$$\hat{K} = (D_r Q - X_l)(N_r Q + Y_l)^{-1} \tag{2.54}$$

o igualmente,

$$\hat{K} = (Y_r + QN_l)^{-1}(QD_l - X_r) \tag{2.55}$$

con,

$$\det(N_r Q + Y_l) \neq 0$$

$$\det(Y_r + QN_l) \neq 0$$

$$Q \in \mathfrak{M}(RH_\infty)$$

Por otra parte, el controlador correspondiente a $Q = 0$, *viene expresado por,*

$$K_0 = -X_l Y_l^{-1} = -Y_r^{-1} X_r \tag{2.56}$$

Además $\underline{S}(\hat{G}, \hat{K})$ *puede escribirse como,*

$$\underline{S}(\hat{G}, \hat{K}) = R + UQV \tag{2.57}$$

con,

$$R = \hat{G}_{zw} - \hat{G}_{zu} X_l D_l \hat{G}_{yw} \in \mathfrak{M}(RH_\infty) \tag{2.58}$$

$$U = \hat{G}_{zu} D_r \in \mathfrak{M}(RH_\infty) \tag{2.59}$$

$$V = D_l \hat{G}_{yw} \in \mathfrak{M}(RH_\infty) \tag{2.60}$$

Nota: $\underline{S}(\hat{G}, \hat{K})$ puede escribirse como una *función afin* en $Q \in \mathfrak{M}(RH_\infty)$.

A pesar de lo complejo que pudiera parecer la determinación de una doble factorización coprima para $\hat{G}_{yu}$, a continuación se presenta un resultado que permite calcular dicha factorización de forma inmediata.

Theorem 21 *(Vidsayagar, 1984) Suponga que $\hat{G}_{yu}$ admita la siguiente representación en espacio de estado*

$$\left(\begin{array}{c|c} A^{yu} & B^{yu} \\ \hline C^{yu} & D^{yu} \end{array}\right)$$

Suponga (A^{yu}, B^{yu}) controlable y (A^{yu}, C^{yu}). Sean F y L tales que $A + B^{yu}F$ y $A + LC^{yu}$ sean Hurwitz. Entonces $\hat{G}_{yu}$ admite una doble factorización coprima caracterizada por:

$$\left(\begin{array}{cc} D_r & -X_l \\ N_r & Y_l \end{array}\right) = \left(\begin{array}{c|cc} A^{yu} + B^{yu}F & B^{yu} & -L \\ \hline F & I & 0 \\ C^{yu} + D^{yu}F & D^{yu} & I \end{array}\right) \tag{2.61}$$

$$\left(\begin{array}{cc} Y_l & X_r \\ -N_l & D_l \end{array}\right) = \left(\begin{array}{c|cc} A^{yu} + LC^{yu} & -B^{yu} - LD^{yu} & L \\ \hline F & I & 0 \\ C^{yu} & -D^{yu} & I \end{array}\right) \tag{2.62}$$

2.4 Problemas de Optimización Multi-objetivos

Muchos problemas de ingeniería pueden ser formulados de la siguiente forma:

$$\min_{\theta \in \Theta} \|\Phi(\theta)\| \tag{2.63}$$

con,

$$\Phi = \begin{pmatrix} \Phi_1(\theta) \\ * \\ * \\ \Phi_s(\theta) \end{pmatrix}$$

Las $\Phi_i : \Theta \rightarrow \mathcal{R}_0^+$, $i = 1, 2, ..., s$ representan funcionales (lineales o no lineales) a minimizar. El vector θ representa los parámetros de diseño y Θ denota el conjunto de valores posibles para θ (espacio de búsqueda).

De esta forma, el problema planteado en (2.63) se denomina *Problema de Optimización Multi-objetivos* (**POM**), formulado en este caso *sin restricciones*. Muy a menudo, la reducción del valor de una funcional Φ_i conduce al incremento en otra(s). Se dice entonces que las funcionales se encuentran *en conflicto*. Es facil entonces comprender que no existe en general una solución única para un **POM**, es decir, un único θ^* que *minimice* simultáneamente *todas* las funcionales $\Phi_i : \Theta \rightarrow \mathcal{R}_0^+$, $i = 1, 2, ..., s$.

Numerosos métodos han sido propuestos para abordar los **POM**, como por ejemplo en (Zakian and Al-Naib, 1973), (Giesy, 1978) o (Gembicki and Haimes, 1975). A contiuación se presenta el enfoque utilizado en el presente trabajo.

Problema Min-Max (PMM).

Es posible reducir el problema de optimización multi-objetivo (2.63) a un problema escalar de tipo *Min-Max*, como el planteado en (Whidborne et al., 1997), de la siguiente forma:

$$\min_{\theta\in\Theta}\left\{\max_{i=1,2,..s}\left(\frac{\Phi_i(\theta)-\varepsilon_i}{\varepsilon_i}\right)_{\varepsilon_i>0}\right\} \tag{2.64}$$

donde los $\varepsilon_i > 0$ son números reales que expresan los valores máximos tolerables que pueden tomar las diferentes funcionales en conflicto Φ_i para $i = 1, 2, ..., s$.

2.5 Técnicas directas de optimización

Una vez planteado el problema multi-objetivos como en (2.64), la siguiente etapa consiste en seleccionar una técnica de optimización para intentar resolverlo. Esta selección depende de múltiples factores, tales como la complejidad del problema, los recursos disponibles, el tiempo de cálculo estimado, etc.

Dentro de la amplia gama de técnicas disponibles, en esta investigación nos interesan particularmente las llamadas heurísticas *directas* de optimización (Hooke and Jeeves, 1961), las cuales se caracterizan porque durante el proceso de búsqueda solo se utilizan evaluaciones de las funcionales, sin hacer intervenir ningun tipo de derivación (Lagarias et al., 1998). Entre las ventajas que presenta la aplicación de este tipo de técnicas es posible citar las siguientes:

1. Pueden aplicarse a problemas lineales o no lineales.
2. Proporcionan resultados aceptables.
3. Son sencillas, desde el punto de vista de su implementación.
4. Pueden utilizarse para conseguir "buenos" puntos de inicio para técnicas más sofisticadas.

Básicamente, todas las técnicas directas se basan en un proceso de búsqueda que puede ser resumido como se muestra a continuación:

1. Se seleccionan diferentes posibles soluciones y se calcula sus correspondientes funcionales de mérito.
2. Se determina la "mejor" solución, en función del cálculo realizado.
3. Se repite hasta que se haya conseguido una solución "aceptable" o hasta cierto número de iteraciones.

Note que al finalizar la aplicación de esta estrategia de búsqueda, la última "mejor" solución es la que ha cumplido en mayor grado con los requerimientos del problema. Como puede apreciarse, el proceso de búsqueda es en esencia bastante sencillo. Sin embargo, la experiencia ha demostrado que puede ser bastante eficaz a la hora de conseguir buenas soluciones en casos reales.

A continuación se describen brevemente los dos algoritmos de optimización utilizados en CONTROL-DESIGN, a saber, el "Moving Boundaries Process" (MBP) y el "Guided Evolutionary Simulated Annealing" (GESA).

Algoritmo "Moving Boundaries Process " (MBP).

Este algoritmo fue propuesto en (Rosenbrock, 1960) para resolver el problema de minimización de la famosa "*Función Banana*", la cual se muestra en la figura 2-7:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Al igual que en el caso del algoritmo "Nelder-Mead", el MBP se basa en realizar movimientos exploratorios alrededor de un punto de inicio. La búsqueda se mantiene hasta que se haya conseguido por lo menos un "exito" y un "fracaso" en cada dirección seleccionada. A partir este momento, el algoritmo intenta lograr que los próximos movimientos

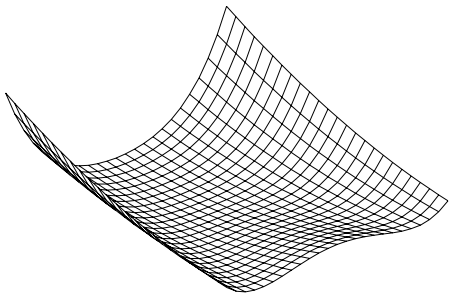

Figura 2-7: Función "Banana" de Rosenbrock

se produzcan en la dirección de máxima variación y en direcciones ortogonales a ésta. Un resumen del agoritmo se muestra a continuación.

Algoritmo "Moving Boundaries Process " (MBP)

1. Inicializar $\theta = \theta_0$, con $n = \dim(\theta)$
2. Inicializar matriz de movimiento $V = V_0$, de dimensiones $n \times n$.
3. Inicializar vector de longitudes de movimiento $e = e_0$, de dimensión n.
4. Inicializar vector de suma de direcciones exitosas $d = d_0$, de dimiensión n.
5. Inicializar $rot = 0$.
6. Inicializar $Niter = 1$.
7. Inicializar $i = 1$.
8. Mientras $[\Phi_k(\theta) > \varepsilon_k$ para $k = 1, .., s]$ y $[abs(max(e)) > \delta\ ]$ y $[Niter < Nitermax]$ hacer:

   $\theta_{trial} = \theta + e_i V(:, i)$

   Si $\Phi_k(\theta_{trial}) < \max_k (\Phi_k(\theta), \varepsilon_k)$ entonces,

succes(i) = 1

si no,

succes(i) = 0

$Niter = Niter + 1$

Si succes(i) = 0 y succes(i-1)=0 entonces, % exito seguido de fracaso

$e_i = \beta e_i$

rot =1; % Rotación

$ortogonalizar(v, d)$

Si succes(i) = 1 entonces,

$e_i = \alpha e_i$

$d_i = d_i + e_i$

$\theta = \theta_{trial}$

9. Fin de Mientras.

El código utilizado en CONTROL-DESIGN fue adaptado del creado en (Gu. et al., 1994) para el toolbox MODCONS y puesto a la disposición del público en la página web:

http : //www.eee.kcl.ac.uk/mecheng/jfw/modcons.html

Guided Evolutionary Simulated Annealing (GESA).

La estrategia de búsqueda Simulated Annealing (SA) fue inicialmente propuesta an principios de la decada de los 80 (Kirpatrick et al., 1983). Se basa en una analogía con un proceso metalúrgico simple, el cual consiste en calentar un material hasta que alcance cierta temperatura y posteriormente enfriarlo con una lentitud tal que al finalizar el proceso posea una estructura de mínima energía (equilibrio térmico).

El algoritmo parte de una solución inicial x0 y de una temperatura T0. En cada iteración se genera un candidato, el cual se acepta como nueva solución si su función de mérito asociada es mejor que la actual. En caso contrario se acepta de todas formas con una probabilidad que disminuye con la temperatura T al final de cada iteración. Esto implica que inicialmente casi todos los movimientos son aceptados, explorándose aleatoriamente sobre todo el espacio de soluciones. Progresivamente la temperatura va disminuyendo y la probabilidad de aceptar nuevas soluciones decrece. Al final del algoritmo, sólo los movimientos que mejoran efectivamente la función objetivo seran aceptados. El algoritmo SA ha demostrado ser bastante eficaz para conseguir buenas soluciones de problemas complejos. Sin embargo, todos los autores coinciden en que el mayor inconveniente de esta estrategia es que el tiempo de cálculo y los recursos computacionales que requiere pueden resultar excesivos.

Los algoritmos evolutivos y genéticos son técnicas que se basan en los principios de evolución natural como método de búsqueda para resolver problemas complejos. Su gran ventaja frente a los métodos convencionales es que presentan una alta eficiencia para una gran variedad de problemas. Mientras que los métodos tradicionales alcanzan una eficiencia mayor para sólo un tipo específico de problemas. La diferencia fundamental entre los algoritmos genéticos y evolutivos es que los primeros utilizan preferiblemente el cruce entre individuos como mecanismo de variación y los segundos por el contrario se basan en mutaciones.

El funcionamiento básico de un algoritmo evolutivo o genético puede ser resumido como sigue a continuación:

- Se genera, de forma aleatoria, una población inicial de soluciones potenciales del problema.
- Se desarrolla un proceso iterativo que transforma dicha población mediante:
  - Evaluación de las soluciones que forman la población.
  - Selección y reproducción de un conjunto de soluciones basándose en su conveniencia o adecuación.

- Recombinación de estas soluciones por medio de operadores de variación para formar una nueva población.

En general un algoritmo evolutivo o genético debe contener los siguientes elementos (Jimenez and Sanchez, 2002):

- Una representación de las soluciones potenciales al problema.
- Una forma de crear una población inicial de soluciones potenciales.
- Una "función de mérito" capaz de medir la bondad de cualquier solución.
- Un conjunto de operadores de variación.
- Un conjunto de parámetros que guían la evolución del algoritmo (tamaño de la población, número de iteraciones, probabilidades, etc).

El algoritmo GESA, propuesto en (Yip, 1993), se plantea como un intento de mejorar el tiempo de cálculo del algoritmo SA mediante la integración de varios conceptos de computación evolutiva. En efecto, este algoritmo puede ser considerado como varios procesos de SA funcionando en paralelo y compitiendo entre sí de dos formas distintas. En la primera, los hijos de una misma familia compiten entre si y contra su padre. En el segundo, las familias compiten entre si por el número de hijos que seran generados en la próxima generación. El algoritmo básico se muestra a continuación.

Algoritmo "Guided Evolutionary Simulated Annealing" (GESA)

1. Inicializar la temperatura T_0^{GESA}.
2. Seleccionar Np padres.
3. Generar Nh hijos de cada padre.
4. Encuentre el mejor hijo de cada padre.

5. Seleccionar los padres para la próxima generación. Por cada familia, se acepta el mejor hijo como el padre de la próxima generación si:

$$\Phi_1 < \Phi_2$$

o,

$$e^{-\frac{\Phi_1 - \Phi_2}{T^{GESA}}} > \rho \tag{2.65}$$

donde,

- Φ_1 es el valor de la función objetivo del mejor hijo.
- Φ_2 es el valor de la función objetivo del padre.
- T^{GESA} es el valor del coeficiente de temperatura.
- ρ es un valor real aleatorio uniformemente distribuido en $[0, 1]$

6. Encuentre el número de hijos que seran generados en la próxima generación.

7. Decremente el coeficiente de temperatura, esto es:

$$T^{GESA} \leftarrow \alpha_{GESA} T^{GESA}$$

con $\alpha_{GESA} \in (0, 1)$, coeficiente de enfriamiento.

8. Repita desde el paso 3 hasta que se haya conseguido una solución aceptable o se hayan alcanzado un cierto número de iteraciones.

Esde hacer notar que el código correspondiente a este algoritmo fue escrito y probado por el autor del presente trabajo, tomando como base el artículo (Yip, 1993).

2.6 GUIDE (Graphic User Interface Design Environment)

El toolbox GUIDE de MATLAB® permite desarrollar interfaces gráficas de manera sencilla pero eficiente: el programador puede concentrarse en la definición de los modos de funcionamiento y los programas que son ejecutados desde la interfaz.

El proceso de diseño de una interfaz gráfica debe comenzar con un análisis conceptual de las necesidades. El resultado de este primer paso puede ser una descripción en prosa del funcionamiento esperado. A partir de este documento, se elabora un primer esbozo, tal como se muestra en la figura 2-8.

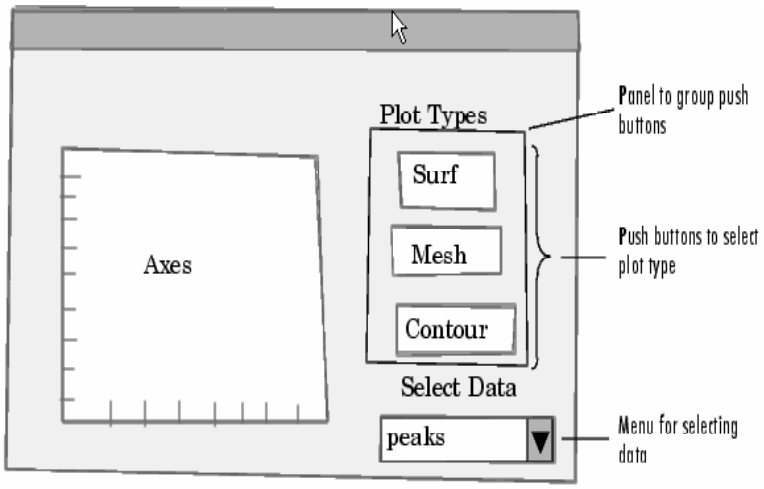

Figura 2-8: Primer esbozo de una interfaz gráfica

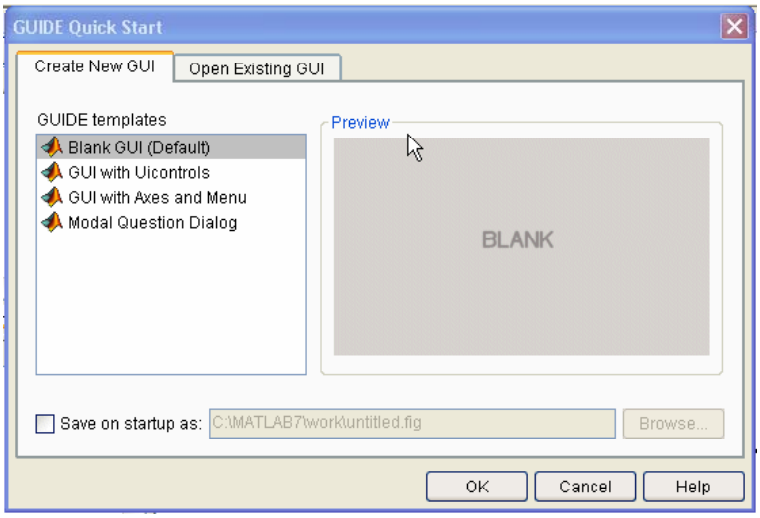

Figura 2-9: Menú de inicio del toolbox GUIDE

En general una GUI está compuesta por botones, campos de texto, barras deslizantes, gráficos, controles, etc. Cada uno de estos elementos se conoce con el nombre de "objeto". En el contexto del toolbox GUIDE, cada interfaz está relacionada con dos archivos:

- Un archivo de extensión FIG: el cual contiene la descripción de los objetos gráficos que conforman la interfaz.
- Un archivo de extensión M: el cual contiene las "callback" (funciones que se ejecutan cuando se selecciona algun objeto gráfico).

Cuando se ejecuta el comando "guide" en MATLAB® aparece la caja de diálogo mostrada en la figura 2-9.

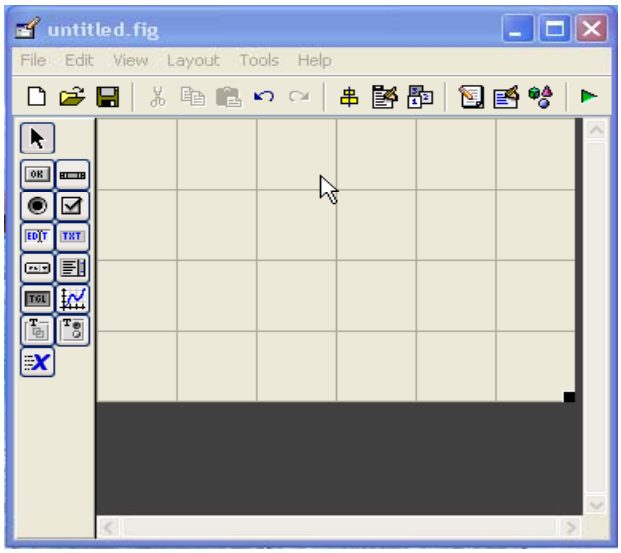

Figura 2-10: Una interfaz gráfica "en blanco"

Como puede observarse, es posible crear una nueva GUI a partir de modelos pre-establecidos (templates) o abrir una GUI previamente guardada. Supongamos que se decide crear una GUI en "blanco" (sin ningun objeto). Cuando se selecciona esta opción aparece la ventana mostrada en la figura 2-10.

En la parte izquierda se aprecian los diferentes objetos que pueden utilizarse para el diseño de la GUI. La parte central corresponde al área de diseño.

Es importante entender que lo principal de este toolbox es su "filosofía" la cual se basa en dos elementos esenciales:

- Las "callback", es decir, las funciones que se ejecutan cuando el usuario ejerce una acción en la interfaz.
- Las estructuras de datos ("handles") en las que se almacena todas las variables que utiliza la interfaz.

La plataforma MATLAB ofrece numerosos ejemplos de diseño de interfaces gráficas y

responde a las preguntas más frecuentes.

De esta forma concluye el presente capítulo dedicado a recordar algunas nociones básicas que serán necesarias para explicar el próximo capítulo. Se recomienda al lector interesado revisar la bibliografía recomendada para cualquier detalle con respecto a las definiciones, teoremas, algoritmos y programas aquí presentados.

CAPITULO 3

PLANTEAMIENTO DEL PROBLEMA DE DISEÑO

El problema de diseño de controladores será formulado matemáticamente en este trabajo de la siguiente forma:

Dada la interconexión mostrada en la Figura 2-6, caracterizado por la matriz de transferencia $\underline{S}(\hat{G}, \hat{K}) \in \mathcal{R}(s)^{n_z \times n_w}$ *tal que:*

$$z = \underline{S}(\hat{G}, \hat{K})w \tag{3.1}$$

donde:

- $\hat{G} \in \mathcal{R}(s)^{(n_u+n_w)\times(n_z+n_y)}$ *es el modelo lineal generalizado de la planta a lazo abierto.*
- $\hat{K} \in \mathcal{R}(s)^{n_u \times n_y}$ *es el modelo lineal del controlador.*
- w *es el vector de entradas exógenas.*
- z *es el vector de salidas que se desean controlar .*

- *u es el vector de señales manipulables.*
- *y es el vector de salidas medidas.*

Encuentre $\hat{K}^* \in \mathcal{R}(s)^{n_u \times n_y}$ *tal que*:

$$\hat{K}^* = \arg\left\{ \min_{\hat{K} \in \mathcal{R}(s)^{n_u \times n_y}} \left\{ \max_{i=1,2,..s} \left(\frac{\Phi_i(\hat{K}) - \varepsilon_i}{\varepsilon_i} \right)_{\varepsilon_i > 0} \right\} \right\} \tag{3.2}$$

sujeto a $\underline{S}(\hat{G}, \hat{K})$ *internamente estable*

Una de las etapas más importantes durante el diseño de controladores, es justamente la elección adecuada de las $\Phi_i[\underline{S}(\hat{G}, \hat{K})]$, $i = 1, 2, ..., s$. Mediante esta formulación es posible "codificar" una gran cantidad de requerimientos, tanto desde el punto de vista temporal como frecuencial. Algunos ejemplos sencillos podrían ser, en el caso escalar:

- Para minimizar el error en estado estacionario ante una entrada en forma de escalón, es posible utilizar:

  $$\Phi_1[\underline{S}(\hat{G}, \hat{K})] = \left| s_{ref} - \lim_{t \to \infty} s(t) \right|$$

  donde s(t) es la respuesta al escalón y s_{ref} es el nivel deseado. Note que de manera similar es posible hacer "tracking" de entradas más complejas tales como rampas, señales triangulares, etc.

- Para limitar los "picos" del régimen transitorio ante una entrada en forma de escalón:

  $$\Phi_2[\underline{S}(\hat{G}, \hat{K})] = \max |s(t)|$$

  donde s(t) es la respuesta al escalón.

- Para minimizar alguna norma de la transferencia a lazo cerrado:

  $$\begin{aligned} \Phi_3[\underline{S}(\hat{G}, \hat{K})] &= \left\| \underline{S}(\hat{G}, \hat{K}) \right\|_p \\ p &\in \{1, 2, \infty\} \end{aligned}$$

Hasta los momentos, el problema de diseño ha sido formulado con una restricción fundamental, como lo es la estabilidad interna de la matriz de transferencia a lazo cerrado $\underline{S}(\hat{G}, \hat{K})$. Es en este punto donde el resultado clásico sobre la parametrización de controladores estabilizantes de Youla, presentada en el Capítulo 2 (p. 25), demuestra toda su importancia. En efecto, recuerde que la matriz de transferencia a lazo cerrado $\underline{S}(\hat{G}, \hat{K})$ tal que,

$$\underline{S}(\hat{G}, \hat{K}) = \hat{G}_{zw} + \hat{G}_{zu}\hat{K}(I_{n_y \times n_u} - \hat{G}_{yu}\hat{K})^{-1}\hat{G}_{yw}$$

es internamente estable si y solo si:

- $\hat{G}_{yu}$ admite una doble factorización coprima,

  $$\hat{G}_{yu} = N_r D_r^{-1} = D_l^{-1} N_l$$

  con $N_r \in \mathfrak{M}(RH_\infty)$, $D_r \in \mathfrak{M}(RH_\infty)$ y además existen $X_r, Y_r, X_l, Y_l \in \mathfrak{M}(RH_\infty)$ tales que:

  $$\begin{pmatrix} Y_r & X_r \\ -N_l & D_l \end{pmatrix} \begin{pmatrix} D_r & -X_l \\ N_r & Y_l \end{pmatrix} = I$$

- $\hat{K} \in \mathcal{R}(s)^{n_y \times n_u}$es tal que:

  $$\begin{aligned} \hat{K} &= (D_r Q - X_l)(N_r Q + Y_l)^{-1} \\ &= (Y_r + Q N_l)^{-1}(Q D_l - X_r) \end{aligned}$$

  con,

  $$\det(N_r Q + Y_l) \neq 0$$

  $$\det(Y_r + Q N_l) \neq 0$$

  y

  $$Q \in RH_\infty^{n_{qu} \times n_{qy}}$$

  En este caso, $\underline{S}(\hat{G}, \hat{K})$ puede además escribirse como,

  $$\underline{S}[\hat{G}, \hat{K}(Q)] = R + UQV \tag{3.3}$$

con,

$$R = \hat{G}_{zw} - \hat{G}_{zu} X_l D_l \hat{G}_{yw} \tag{3.4}$$

$$U = G_{zu} D_r \tag{3.5}$$

$$V = D_l G_{yw} \tag{3.6}$$

y de manera inmediata, las matrices de transferencia a lazo cerrado de $w \rightarrow y$ y de $w \rightarrow u$, denotadas $\hat{G}_{z_1 w_{cl}}$ y $\hat{G}_{z_2 w_{cl}}$ respectivamente, son tales que:

$$\hat{G}_{z_1 w_{cl}}(Q) = R_{z_1 w} + U_{z_1 w} Q V_{z_1 w} \tag{3.7}$$

$$\hat{G}_{z_2 w_{cl}}(Q) = R_{z_2 w} + U_{z_2 w} Q V_{z_2 w} \tag{3.8}$$

donde las matrices $R_{z_1 w}, R_{z_2 w}, U_{z_1 w}, U_{z_2 w}, V_{z_1 w}$ y $V_{z_2 w} \in \mathfrak{M}(RH_\infty)$ dependen de la factorización coprima de $\hat{G}_{yu}$ seleccionada.

En términos del parámetro de Youla, el problema de diseño puede ser reformulado *sin ninguna restricción* tal como sigue.

$$Q^* = \arg \left\{ \min_{Q \in RH_\infty^{n_{gu} \times n_{qy}}} \left\{ \max_{i=1,2,..s} \left(\frac{\Phi_i(Q) - \varepsilon_i}{\varepsilon_i} \right)_{\varepsilon_i > 0} \right\} \right\} \tag{3.9}$$

Nota: Los $\varepsilon_i > 0$ son números reales seleccionados por el diseñador para expresar los valores máximos tolerables que pueden tomar las funcionales.

La principal ventaja de la reformulación que se acaba de exponer es la desaparición de la restricción de estabilidad interna, puesto que todo $Q \in RH_\infty^{n_{gu} \times n_{qy}}$ produce un controlador $\hat{K}(Q)$ el cual estabiliza el modelo lineal de la planta. Ahora bien, si se quiere implementar cualquier estrategia de búsqueda computacional, es necesario *codificar* la variable Q para

que pueda ser representada, de forma más o menos aproximada, en un equipo de computación (por ejemplo, mediante un arreglo de parámetros en formato de *coma flotante).* A continuación se presenta la representación propuesta en esta investigación.

Una de las representaciones más utilizadas de $Q \in RH_\infty^{n_{qu} \times n_{qy}}$, propuesta por ejemplo en (Hu et al., 1995) y (Boyd et al., 1988), consiste en definir una aproximación de Q mediante una matriz $Q_{cvx}^{n_q}$ tal que:

$$Q_{cvx}^{n_q} = \sum_{i=1}^{N_Q} \nu_i Q_i \simeq Q \tag{3.10}$$

con,

- $\{Q_i\}$, $i = 1, 2, ...N_Q$, conjunto de elementos arbitrarios de $RH_\infty^{n_{qu} \times n_{qy}}$, *ortogonales* entre si.
- $n_q = deg(Q_{cvx}^{n_q})$
- $\nu \in \mathcal{R}^{N_Q}$, vector de "coordenadas" de $Q_{cvx}^{n_q}$, representando a $Q \in RH_\infty^{n_{qu} \times n_{qy}}$.

La gran ventaja de este enfoque es que, si una funcional $\Phi(Q)$ es convexa, entonces la funcional $\Phi(\nu)$ sigue siendo convexa. En este caso es posible utilizar los eficientes algoritmos de optimización convexa disponibles (Polak et al., 1984). Sin embargo, mediante esta representación se limita el espacio de búsqueda al espacio generado por los Q_i. Es decir que el diseñador debe, mediante ensayo y error, seleccionar las matrices Q_i más *adecuadas* para cada aplicación, lo cual puede evidentemente ser bastante engorroso. En este sentido, varios investigadores han trabajado para facilitar la selección de dichas Q_i , como por ejemplo en (Clement, 2001),(Hindi et al., 1998).

El método que se adopta en CONTROL-DESIGN fue propuesto originalmente en (Sanchez, 2003), para un caso bien específico: el control de un reactor químico CSTR de dos entradas y dos salidas. Aquí se presenta el método de manera general, es decir para cualquier número de entradas y salidas. En primer lugar, se introduce el operador *vec* tal como sigue:

Notación: Se denota Θ el conjunto de vectores θ tales que:

- $\theta \in \mathcal{R}^{n_\theta}, \theta_{\min,i} < \theta_i < \theta_{\max,i}\ i = 1, 2, ..., n_\theta$ con

$$\begin{aligned} n_\theta &= n_{_q}(1 + n_{qu} + n_{yu}) + n_{qu} \times n_{qy} \\ n_q &= n_r + 2n_i \\ n_q, n_r, n_i &\in \mathcal{Z}_0^+ \end{aligned}$$

- θ es tal que el vector p^{n_q} formado por:

$$p^{n_q} = \begin{pmatrix} p_1 \\ p_2 \\ . \\ . \\ p_{nr} \\ p_{nr+1} \\ p_{nr+2} \\ p_{nr+3} \\ p_{nr+4} \\ . \\ . \\ p_{nr+2ni-1} \\ p_{nr+2ni} \end{pmatrix} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ . \\ . \\ \theta_{nr} \\ \theta_{nr+1} + j\theta_{nr+2} \\ \theta_{nr+1} - j\theta_{nr+2} \\ \theta_{nr+3} + j\theta_{nr+4} \\ \theta_{nr+3} - j\theta_{nr+4} \\ . \\ . \\ \theta_{nr+2ni-1} + j\theta_{nr+2ni} \\ \theta_{nr+2ni-1} - j\theta_{nr+2ni} \end{pmatrix} \tag{3.11}$$

cumple con:

- $p_i^{n_q} \in \mathfrak{C}^-$ (semi-plano complejo abierto izquierdo), para $i = 1, 2, ..., n_{_q}$, esto es, $\theta_1 < 0, \theta_2 < 0$,...,$\theta_{n_r} < 0, \theta_{n_r+1} < 0, \theta_{n_r+3} < 0, ..., \theta_{n_r+2n_i-1} < 0$
- $Im(p_i^{n_q}) = 0$, para $i = 1, 2, ..., n_r$
- $\overline{p_i^{n_q}} = p_{i+1}^{n_q}$ para $i = n_r + 1, n_r + 3, ..., n_r + 2n_i - 1$

Considerando la anterior notación, sea $vec : \Theta^{n_q,nr,ni} \rightarrow RH_\infty^{n_{qu}\times n_{qy}}$ el operador tal que a todo vector $\theta \in \Theta$ le asigna una matriz $Q_\theta^{n_q,n_r,n_i} = vec(\theta) \in RH_\infty^{n_{qu}\times n_{qy}}$ tal que:

$$\begin{aligned} Q_\theta^{n_q,n_r,n_i} &= vec(\theta) = C_Q(sI - A_Q)^{-1}B_Q + D_Q \\ &= C_Q(sI - A_{0Q} + B_{0Q}K_p)^{-1}B_Q + D_Q \end{aligned} \tag{3.12}$$

con,

$$A_Q = A_{0Q} - B_{0Q}K_p \tag{3.13}$$

y en la cual:

- $A_{0Q} \in \mathcal{R}^{n_q\times n_q}, B_{0Q} \in \mathcal{R}^{n_q\times n_{qu}}$ son matrices arbitrarias seleccionadas de forma que el par (A_{0Q}, B_{0Q}) sea controlable, es decir:

$$rank\begin{pmatrix} B_{0Q} & A_{0Q}B_{0Q} & \cdots & A_{0Q}^{n_q-1}B_{0Q} \end{pmatrix} = n_q \tag{3.14}$$

- K_p es determinado de forma que:

$$eig(A_Q) = eig(A_{0Q} - B_{0Q}K_p) = p^{n_q} \tag{3.15}$$

- $B_Q \in \mathcal{R}^{n_q\times n_{qu}}$, $C_Q \in \mathcal{R}^{n_{qy}\times n_q}$, $D_Q \in \mathcal{R}^{n_{qy}\times n_{qu}}$ son tales tales que:

$$B_Q = \begin{pmatrix} \theta_{n_q+1} & \theta_{2n_q+1} & \cdot & \cdot & \theta_{n_{qu}\times n_q+1} \\ \theta_{n_q+2} & \theta_{2n_q+2} & \cdot & \cdot & \theta_{n_{qu}\times n_q+2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \theta_{2n_q} & \theta_{3n_q} & \cdot & \cdot & \theta_{n_{qu}\times n_q+n_q} \end{pmatrix}$$

$$C_Q = \begin{pmatrix} \theta_{(n_{qu}+1)\times n_q+1} & \cdot & \cdot & \theta_{(n_{qu}+1)\times n_q+n_q} \\ \theta_{(n_{qu}+2)\times n_q+1} & \cdot & \cdot & \theta_{(n_{qu}+2)\times n_q+n_q} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \theta_{(n_{qu}+n_{qy})\times n_q+1} & \cdot & \cdot & \theta_{(n_{qu}+n_{qy}+1)\times n_q} \end{pmatrix}$$

$$D_Q = \begin{pmatrix} \theta_{n_{d0}+1} & \cdot & \cdot & \theta_{n_{d0}+(n_{qu}-1)n_{qy}+1} \\ \theta_{n_{d0}+2} & \cdot & \cdot & \theta_{n_{d0}+(n_{qu}-1)n_{qy}+2} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \theta_{n_{d0}+n_{qy}} & \cdot & \cdot & \theta_{n_{d0}+n_{qy}\times n_{qu}} \end{pmatrix}$$

$$\text{con } n_{d0} = (n_{qu} + n_{qy} + 1) \times n_q$$

Notas:

- Los vectores $\theta_{\min,i} < \theta_i < \theta_{\max,i}$ delimitan el espacio de búsqueda y deben ser fijados por el diseñador.

- Igualmente es necesario especificar el número de polos reales n_r , el número de pares de polos imaginarios n_i (con $n_q = n_r + 2n_i$) y el par (A_{0Q}, B_{0Q}) , cuya única restricción es que sea completamente controlable. En la presente versión de CONTROL-DESIGN se utiliza la siguiente estructura:

$$A_{0Q} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -a_{nq} \\ 1 & 0 & 0 & 0 & 0 & -a_{nq-1} \\ 0 & 1 & 0 & 0 & 0 & \bullet \\ 0 & 0 & 1 & 0 & 0 & \bullet \\ 0 & 0 & 0 & 1 & 0 & \bullet \\ 0 & 0 & 0 & 0 & 1 & -a_1 \end{pmatrix} \tag{3.16}$$

$$B_{0Q} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.17}$$

la cual viene asociada a la ecuación característica:

$$s^n + a_1 s^{n-1} + ... + a_{n-1} s + a_n = 0 \tag{3.18}$$

- Cada $\theta \in \Theta^{n_q,nr,ni}$ representa una matriz única $Q_\theta^{n_q,nr,ni} \in RH_\infty^{n_{qu} \times n_{qy}}$. Sin embargo, una matriz *arbitraria* $Q^{n_q,nr,ni} \in RH_\infty^{n_{qu} \times n_{qy}}$ puede ser representada por una infinidad de *realizaciones equivalentes*, es decir por una infinidad de vectores $\theta \in \Theta^{n_q,nr,ni}$ tales que

$$vec(\theta) = Q_\theta^{n_q,nr,ni} = Q^{n_q,nr,ni}$$

Habiendo introducido este operador, el problema de optimización puede replantearse en los siguientes términos:

$$\theta^* = \arg \left\{ \min_{\theta \in \Theta^{n_q,n_r,n_i}} \left\{ \max_{i=1,2,..s} \left(\frac{\Phi_i[vec(\theta)] - \varepsilon_i}{\varepsilon_i} \right)_{\varepsilon_i > 0} \right\} \right\} \tag{3.19}$$

La gran ventaja del método propuesto es que la representación computacional de $Q_\theta^{n_q,nr,ni} \in RH_\infty^{n_{qu} \times n_{qy}}$ mediante el vector $\theta \in \Theta^{n_q,nr,ni}$ requiere de menos información *a priori* de parte del diseñador. Particularmente, no se necesita especificar las matrices Q_i que intervienen en la Ecuación (3.10). Por otra parte, la gran desventaja es que el problema resultante es en general *no convexo,* con gran cantidad de mínimos locales. De igual forma la dimensión del vector de parámetros $\theta \in \Theta^{n_q,n_r,n_i}$ resulta elevada, en el caso de órdenes n_q elevados.

CAPITULO 4

CONTROL-DESIGN: ESTRUCTURA y FUNCIONAMIENTO

El proceso típico de diseño de un sistema de control puede representarse de manera simplificada como se muestra en la figura 4-1.

A lo largo del mismo, el uso de herramientas computacionales es fundamental en todas sus etapas. En particular, las interfaces gráficas ofrecen la posibilidad de interactuar con el computador de manera sencilla y eficiente.

Desde la aparición de los sistemas operativos basados en interfaces gráficas, la gran mayoría de los usuarios de software prefieren los programas basados en interfaces gráficas, en lugar de los tradicionales basados en "comandos".

En particular, los usuarios de CACSD exigen programas cada vez más eficientes (que aporten solución a problemas reales) pero por otra parte más fáciles de utilizar (que

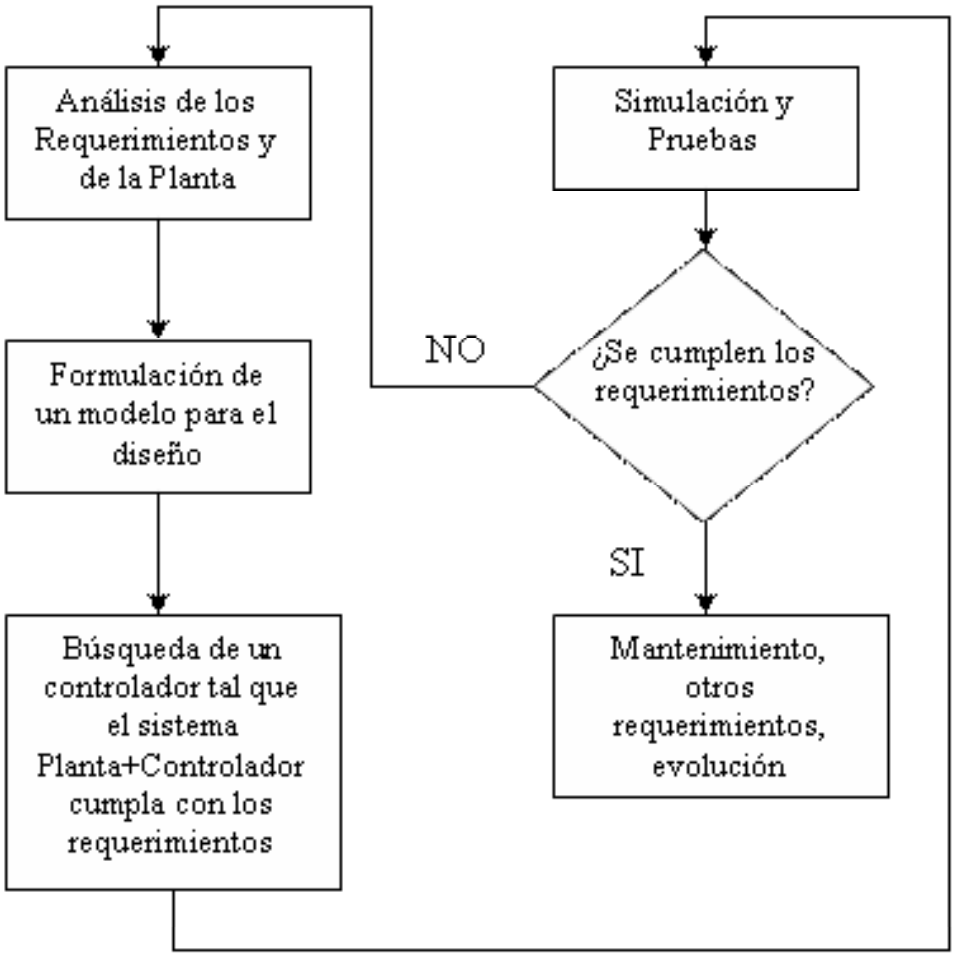

Figura 4-1: Proceso de diseño

permitan plantear y resolver los problemas de manera sencilla).

Sin embargo, al examinar el estado del arte actual, es evidente que todavía falta mucho camino por recorrer en este sentido. Las soluciones derivadas de los programas de CACSD se encuentran a menudo con numerosas dificultades para funcionar adecuadamente con la instrumentación disponible. Muchas veces los requerimientos teóricos de diseño no se cumplen como consecuencia de situaciones no previstas en el modelo.

Por otra parte es importante contar con programas fáciles de utilizar, que requieran un tiempo mínimo de aprendizaje. Y esto motivado no solamente por el ahorro de tiempo y esfuerzo: un programa complicado puede auyentar a sus posibles usuarios, aun cuando sus potencialidades sean superiores con respecto a otros.

En el origen de esta investigación se plantearon entonces las siguientes interrogantes:

- ¿Qué interfaces gráficas estan disponibles en la actualidad para el diseño de controladores?
- ¿Cuáles son sus ventajas y desventajas?
- ¿Para qué tipo de problemas se adaptan?
- ¿Para qué tipo de diseñadores se concibieron?

Es interesante notar que el diseñador actual de sistemas de control se encuentra en una posición contraria a la que caracterizó la época del llamado "control clásico". En ese entonces eran pocas las plataformas computacionales disponibles para el usuario comun. Por el contrario, en la actualidad la cantidad de opciones, métodos y herramientas disponibles es tan abrumadora que muy fácilmente puede confundir.

En consecuencia, es imprescindible comprender que los problemas esenciales que enfrenta cualquier diseñador de sistemas de control son los siguientes:

- Necesita expresar de manera sencilla y precisa el problema de diseño: tanto los requerimientos como el modelo.

- Necesita modificar los parámetros de los algoritmos de búsqueda.
- Necesita guardar los datos, comparar los diseños, realizar comentarios, etc; de la manera más cómoda posible.

En función de estas necesidades, se propuso crear una herramienta que cumpliera además con los siguientes requerimientos:

- Disminución de la intervención del usuario.
- Adaptación a diseñadores inexpertos, con conocimientos básicos de control
- Que sirva como herramienta didáctica en un curso de Sistemas de Control.

La interfaz se desarrolló mediante el toolbox "GUIDE" ("*Graphic User Interface Design Environment*") de MATLAB®. Se tomaron en cuenta aspectos como la facilidad de aprendizaje, la facilidad de uso, necesidades de los diseñadores, etc (ver 4-2), siguiendo las recomendaciones de Cortes (1997):

- El programa debe estar centrado en las necesidades del usuario. Se debe comprender sus modelos mentales, su forma de pensar.
- El programa debe tomar en cuenta los "eventos principales", es decir los servicios más comunes que un usuario pudiera necesitar. Se debe realizar un análisis crítico con respecto a cómo se realiza el diseño de manera "manual", para decidir cuales aspectos deben ser atacados.
- Las figuras e imágenes deben tener un sentido facil de reconocer por el usuario.

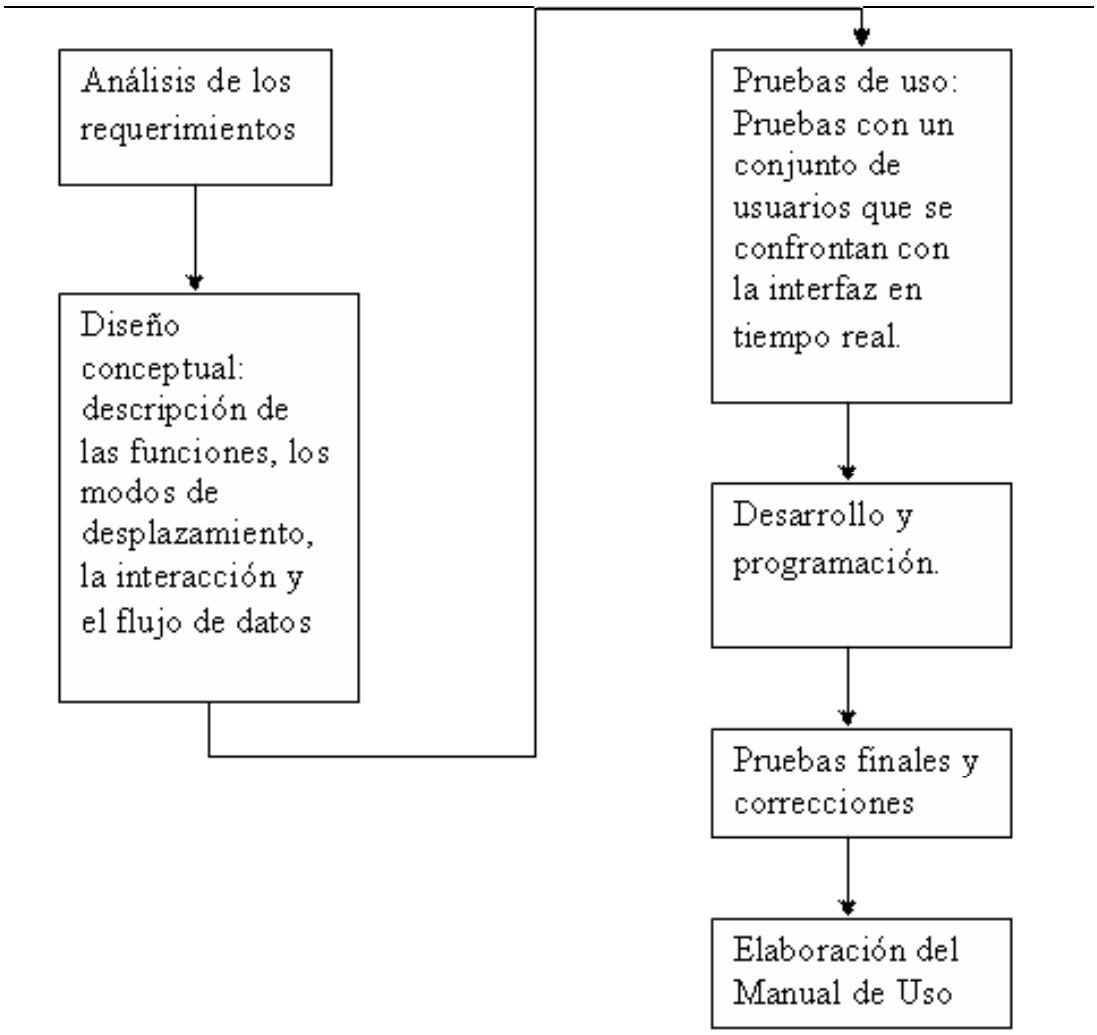

Figura 4-2: Metodología para el diseño de la interfaz

4.1 Estructura

La estructura de CONTROL-DESIGN, tal como se muestra en la figura 4-3, toma en cuenta el equilibrio entre la capacidad de la interfaz y su sencillez.

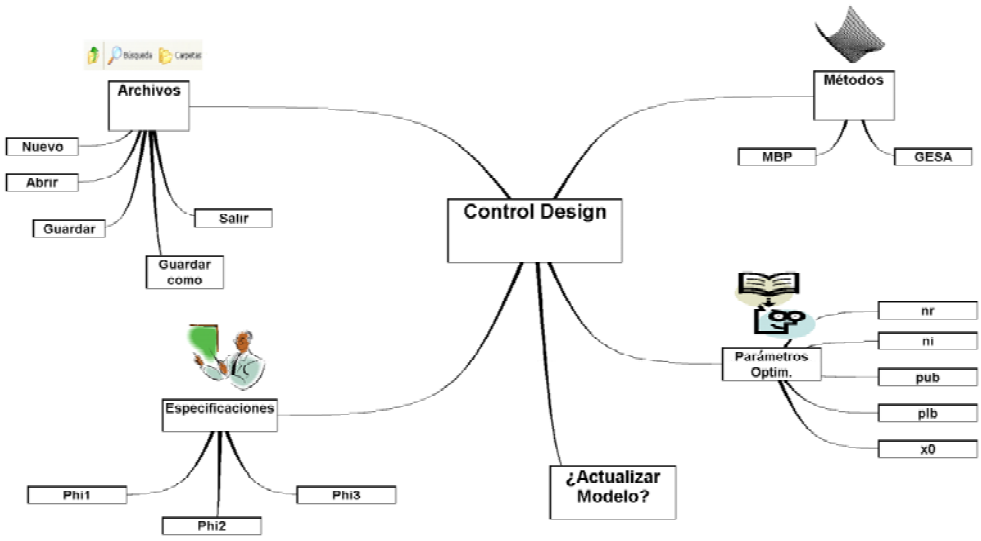

Figura 4-3: Estructura de CONTROL-DESIGN

A continuación se explica las funciones que realiza cada módulo. La vista principal se muestra en la figura 4-4.

- **Archivos**: en este menú se realizan todas las operaciones relacionadas con crear un nuevo diseño, abrir uno previamente guardado, guardar las modificaciones realizadas al diseño en curso o guardar uno con otro nombre.
- **Método**s: en este menú se selecciona el método de optimización que se desea utilizar. En la versión actual solo hay dos métodos disponibles: MBP y GESA.
- **Actualizar Modelo**: En un principio se propuso incluir un editor de matrices en CONTROL-DESIGN. Sin embargo, luego de varios intentos se prefirió utilizar el

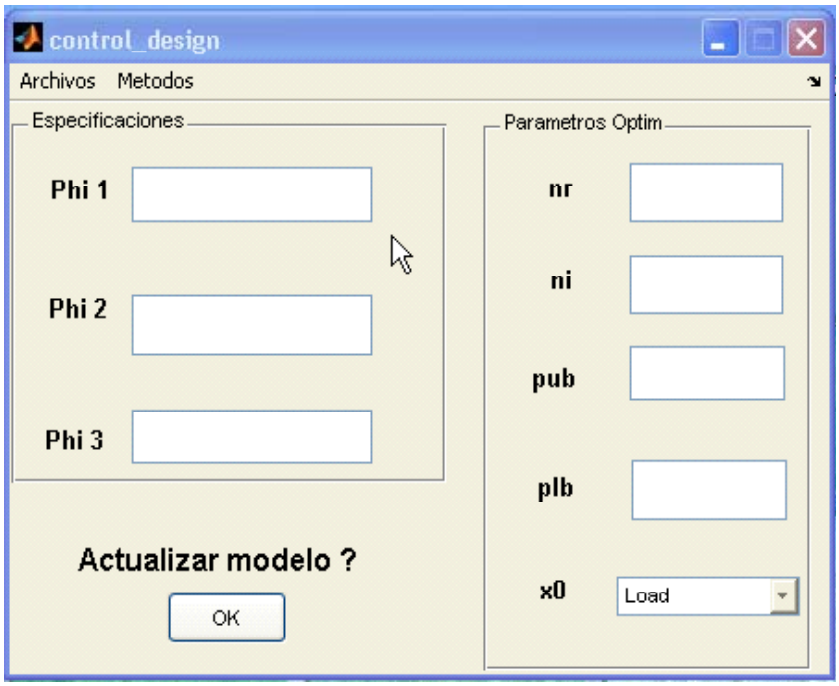

Figura 4-4: Vista principal de la interfaz

editor de matrices incluido en la versión n°7 de MATLAB® (ver figura 4-5). Hay que resaltar que en dicha versión, el editor de matrices ha sido mejorado considerablemente. El diseñador debe entonces "crear" un nuevo diseño utilizando el menú **Archivos**, luego editar las matrices con el editor de MATLAB® y posteriormente "importar" las matrices desde el "Workspace" mediante el botón "**Actualizar Modelo**".

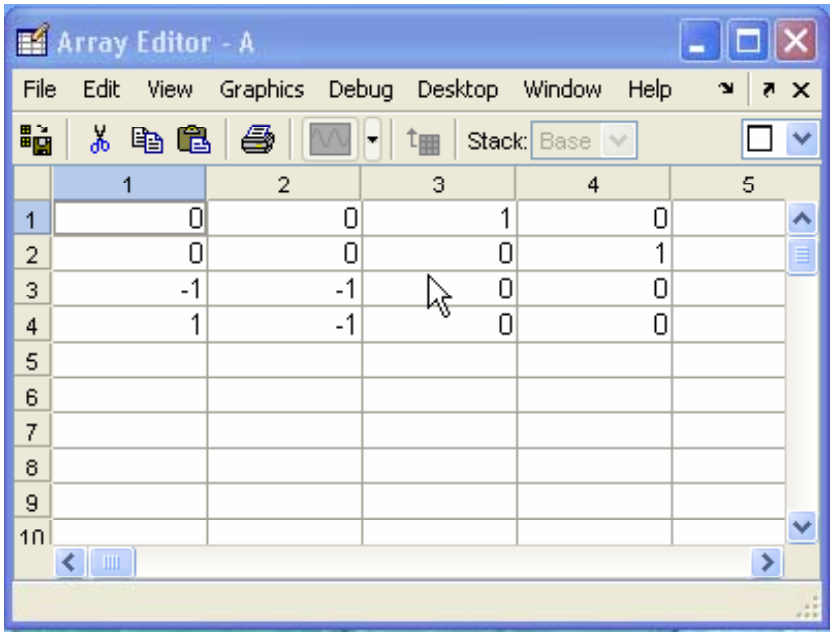

Figura 4-5: Editor de Matrices de MATLAB v.7

- **Especificaciones**: En este módulo se especifican los requerimientos de diseño. En la presente versión de CONTROL-DESIGN el módulo se ha simplificado a solo tres especificaciones denotadas Phi1 Phi2 y Phi3. Debido a que se trabaja con modelos SISO (Single Input Single Output) no existe ambigüedad en el significado de especificaciones tales como Settling Time, Overshoot (puesto que se refieren a la única salida) y Umax (puesto que se refiere a la única entrada de control). Para posteriores versiones queda pendiente plantear un método sencillo para trabajar con especificaciones de este estilo en el caso de sistemas multivariables.

- **Parámetros Optim**: En este módulo se especifican las variables de la representación del parámetro de Youla n_r y n_i. De igual forma se especifican los límites

del espacio de búsqueda *pub* (parameter upper bound), *plb* (parameter lower bound). Estos parámetros son súmamente importantes ya que definen la estructura de polos y los límites del espacio de búsqueda, los cuales pueden influenciar dramáticamente el performance de la optimización. Debido a que no se tienen indicios *a priori* de cómo debe realizarse esta selección, en próximas versiones de esta herramienta se intentará incluir métodos que permitan determinar estos parámetros automáticamente, es decir, sin la intervención del diseñador.

- **x0** : Este menú permite seleccionar la forma en que se desea inicializar el algoritmo de optimización, mediante tres opciones:
  - **Load**: la cual permite cargar un vector de inicio previamente guardado en el directorio corriente, mediante el comando "save x0 x"
  - **Random**: la cual permite seleccionar un punto de inicio aleatorio, seleccionado por el programa.
  - **Hold**: la cual permite mantener el mismo punto de inicio utilizado en la última optimización.

El procedimiento general para atacar un problema de diseño utilizando CONTROL-DESIGN se muestra mediante el diagrama de flujo de la figura 4-6.

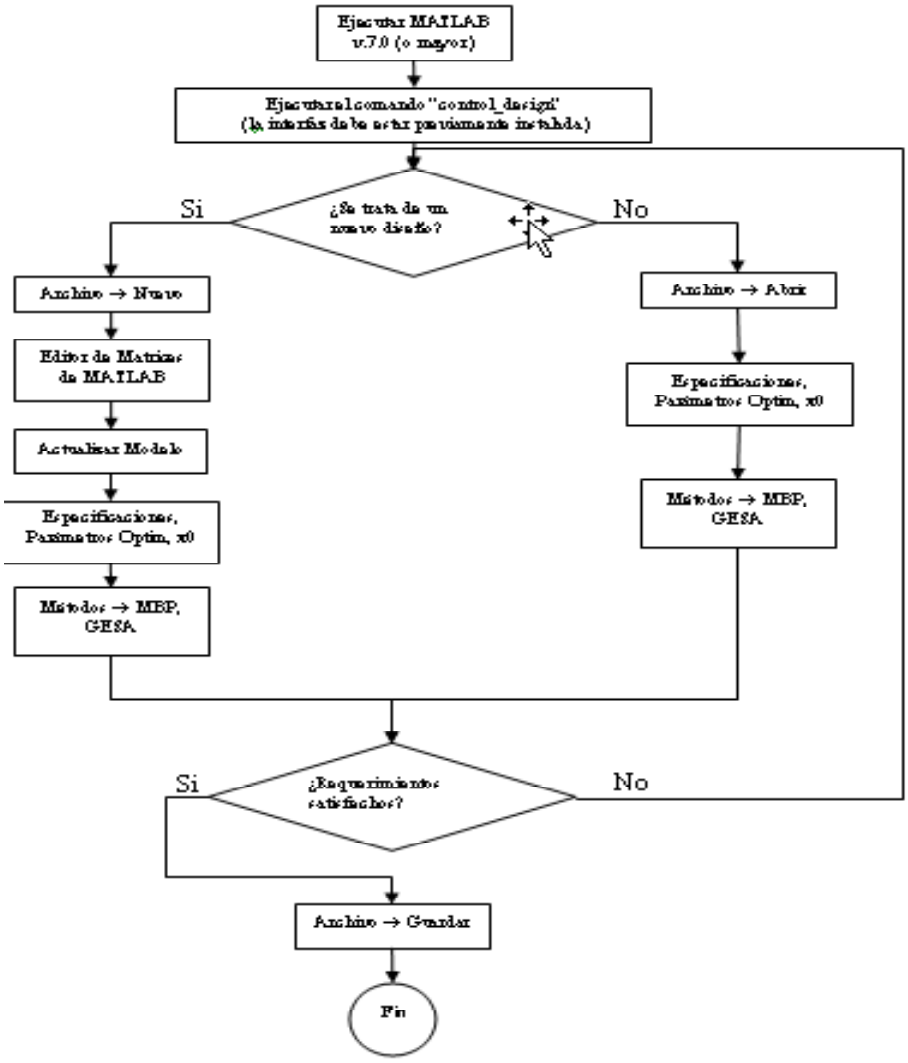

Figura 4-6: Procedimiento de diseño con CONTROL-DESIGN

4.2 Funcionamiento: Ejemplos de diseño.

La mejor manera de explicar el funcionamiento básico de CONTROL-DESIGN es directamente mediante ejemplos de diseño. A continuación se presentan algunos bastante sencillos, destinados a ilustrar en la práctica el funcionamiento de la interfaz gráfica.

4.2.1 Control de un sistema de dos masas y resorte

El problema que se plantea a continuación consiste en una adaptación de los "*benchmark problems*" propuestos en (Wie and Bernstein, 1992). Considere el sistema de dos masas y resorte mostrado en la figura 4-7.

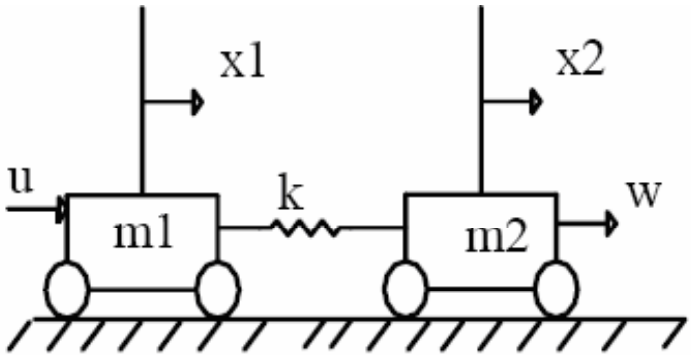

Figura 4-7: Sistema de 2 masas y resorte

Se desea controlar la posición $y = x_2$ de la masa 2 ante perturbaciones representadas por la fuerza w. La señal de control u actúa sobre la masa 1. Las ecuaciones dinámicas del sistema son:

$$m_1 \ddot{x}_1 = u + k(x_2 - x_1) \tag{4.1}$$

$$m_2 \ddot{x}_2 = w - k(x_2 - x_1) \tag{4.2}$$

Las ecuaciones de estado del sistema a lazo abierto se escriben:

$$
\begin{aligned}
\dot{x} &= \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-k}{m_1} & \frac{k}{m_1} & 0 & 0 \\ \frac{k}{m_2} & \frac{-k}{m_2} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_2} \end{bmatrix} w + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_1} \\ 0 \end{bmatrix} u \\
z &= \begin{bmatrix} y \\ u \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} w + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \\
y &= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + [0]\, w + [0]\, u
\end{aligned}
$$

Suponga que los requerimientos que se consideran para este problema son los siguientes:

- Para un impulso en w, se requiere un "*settling time*" menor que 15s. Para esto se toma:

$$\Phi_1 = \max_{t>15} |y(t)| < 0.1$$

- Para un impulso en w, se requiere un pico máximo de salida menor que 3, esto es:

$$\Phi_2 = \max_{t} |y(t)| < 3$$

- Para una entrada en forma de impulso en w, se requiere un pico máximo del esfuerzo de control menor que 5:

$$\Phi_2 = \max_{t} |u(t)| < 5$$

Las matrices de estado, en valores numéricos, con $k = 1, m_1 = 1, m_2 = 1$ son las siguientes:

$$
\begin{aligned}
A &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \\
B_1 &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
B_2 &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\
C_1 &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
C_2 &= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\
D_{11} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
D_{12} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
D_{21} &= 0 \\
D_{22} &= 0
\end{aligned}
$$

Para comenzar el diseño utilizando CONTROL-DESIGN, se debe ir al menú Archivo y luego seleccionar la opción Nuevo, como se muestra en la figura 4-8.

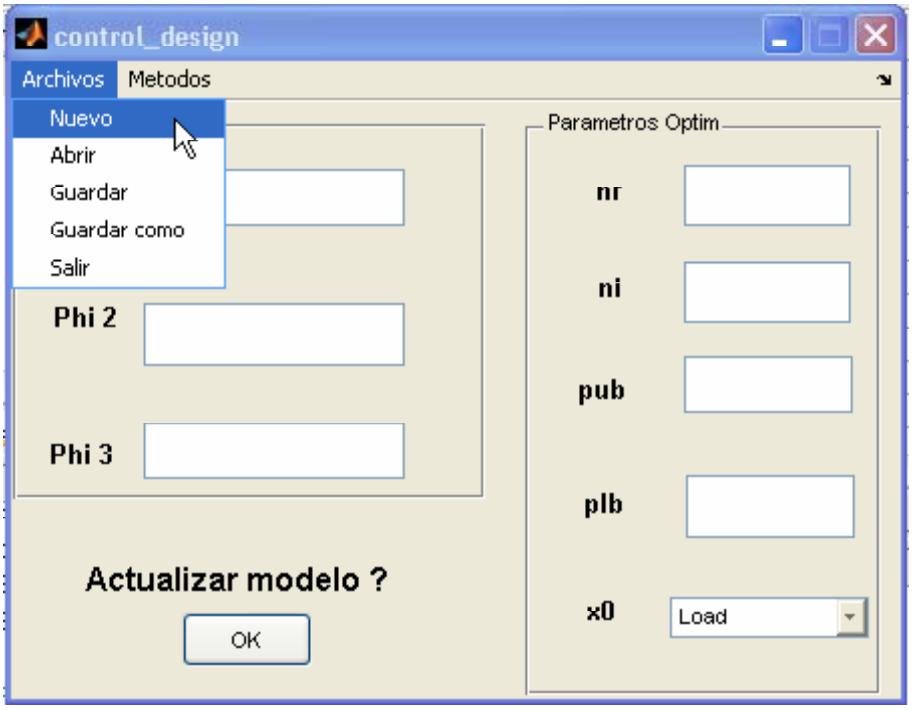

Figura 4-8: Creación de un nuevo modelo utilizando CONTROL-DESIGN

Inmediatamente se crean en el Workspace las variables correspondientes, con el valor inicial 0 para todas ellas, como se muestra en la figura 4-9.

Una vez que se han editado todas las matrices de estado, es necesario actualizar el modelo, utilizando el botón "**Actualizar Modelo**" que aparece en la interfaz gráfica, como se muestra en la figura 4-10.

En realidad, **cada vez que se desee realizar una modificación en las matrices de estado** (por ejemplo para incluir un factor de ponderación de la salida o corregir un error de edición en las matrices) **es necesario actualizar el modelo**, **para que los cambios puedan ser tomados en cuenta por el programa**.

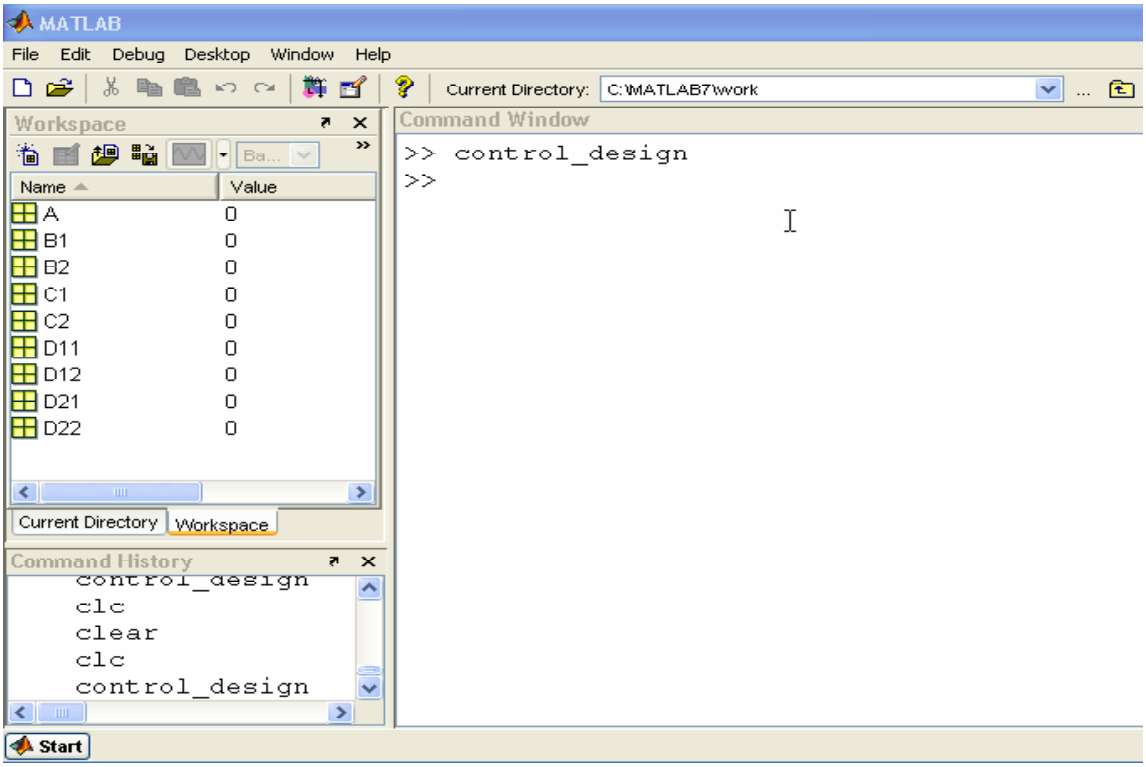

Figura 4-9: Workspace de MATLAB luego de la creación de un nuevo diseño

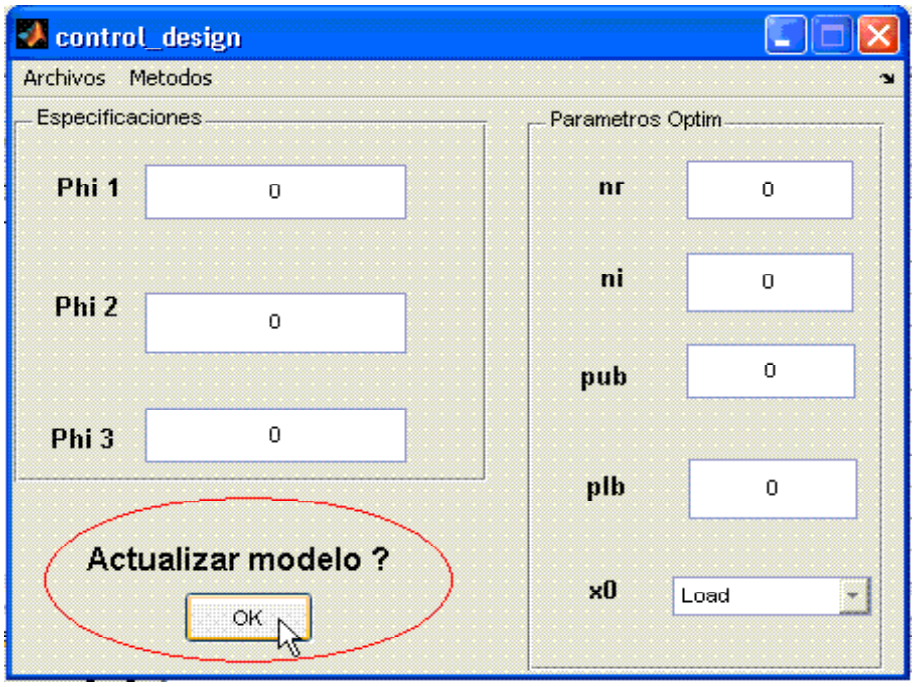

Figura 4-10: Botón Actualizar Modelo

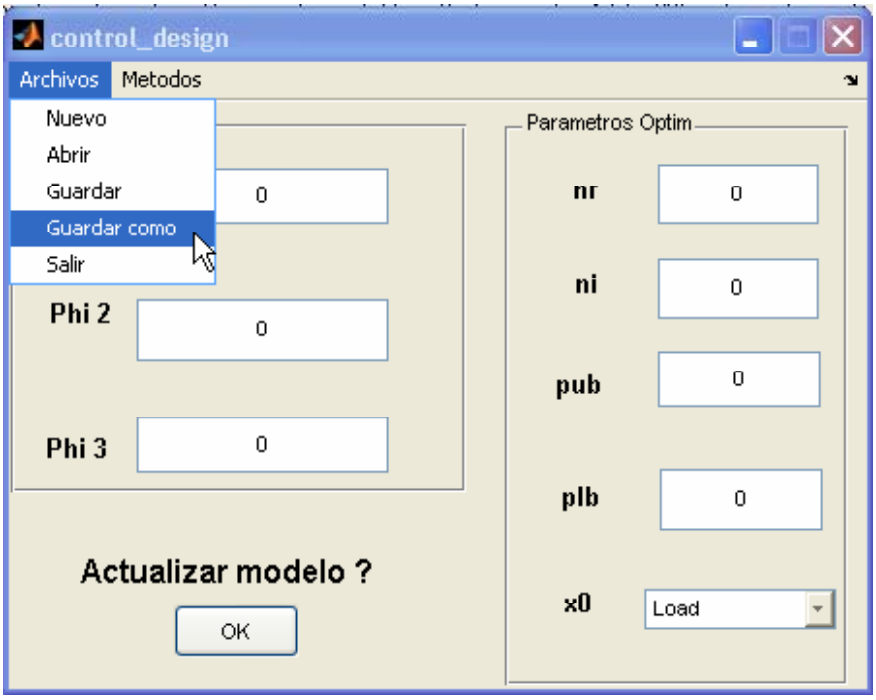

Figura 4-11: Comando "Guardar como"

Una vez actualizado el modelo, el siguiente consiste en guardar el diseño para prevenir cualquier pérdida de información. Para esto se utiliza el menú archivo y el comando "Guardar como" (figura 4-11)

Mediante este comando es posible guardar los datos del diseño en cualquier espacio de memoria disponible en la computadora. La próxima etapa consiste en introducir los valores máximos permisibles para las funcionales de diseño (los ε_i) en las casillas señaladas como Phi 1, 2 y 3. En el caso del ejemplo que estamos analizando el vector de requerimientos sería:

$$\varepsilon = \begin{bmatrix} 0.1 \\ 3 \\ 5 \end{bmatrix} \tag{4.3}$$

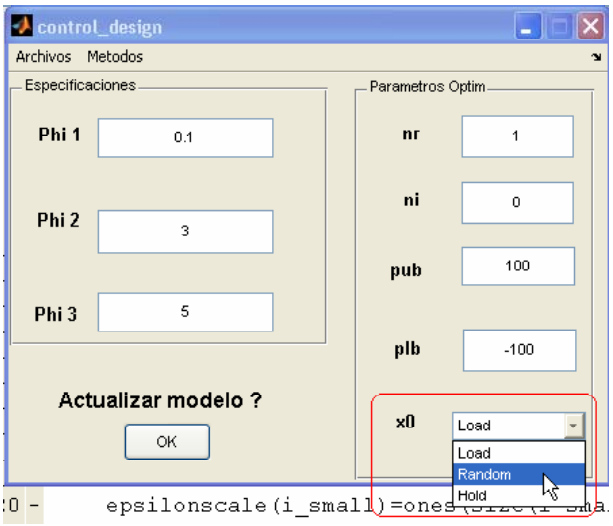

Figura 4-12: Parámetros de Optimización

Luego, se debe seleccionar los parámetros de optimización, a saber:

- El número de polos reales n_r , el número de pares de polos imaginarios n_i (con $n_q = n_r + 2n_i$)
- Los vectores $\theta_{\min}$, $\theta_{\max}$ que limitan el espacio de búsqueda. En la presente versión este espacio se define como un "hipercubo" cuyos límites corresponden a las variables *pub* (parameter upper bound) y *plb* (parameter lower bound) que aparecen en la interfaz gráfica (ver figura 4-12).

Suponga que en este ejemplo planteamos $n_r = 1, n_i = 0$ con

$$pub = -plb = 100 \tag{4.4}$$

y deseamos iniciar la optimización en un punto aleatorio. En la figura 4-12 se muestra esta configuración.

El próximo paso consiste en seleccionar el algoritmo de optimización que se desea utilizar, utilizando el menú "**Métodos**". En la figura 4-13 se muestra esta selección, en

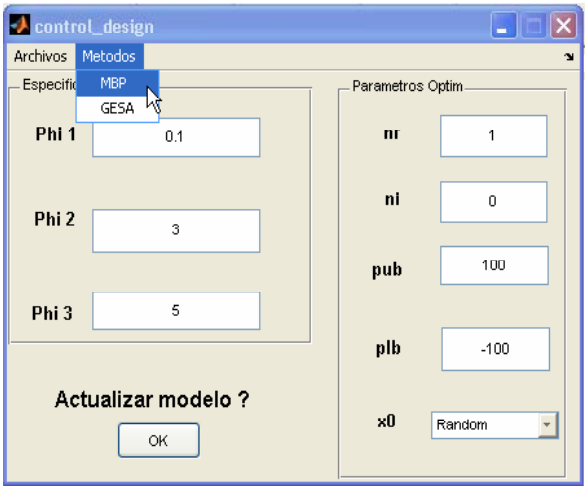

Figura 4-13: Selección del algoritmo MBP

el caso del algoritmo MBP.

A continuación comienza a ejecutarse el algoritmo de optimización, el cual puede ser monitoreado de dos formas. La primera consiste en observar la evolución mediante la información que aparece en la ventana principal de MATLAB, la cual consiste en el número de iteración, los valores actuales de cada funcional y las acciones que realiza el algoritmo.

En el ejemplo mostrado en la figura 4-14 es posible observar que los valores de las funcionales 1 y 2 son menores que los requeridos. Sin embargo el valor de Phi 3 (máximo del esfuerzo de control) es en ese momento mayor que el deseado. En la figura 4-15 se muestra la parte final de la ejecución del algoritmo, el cual se detiene en la iteración 181 debido a que no logra conseguir disminuciones en Phi 3. Note que el mensaje final del algoritmo es "*Inequality not solved*".

Otra forma de monitorear la evolución del algoritmo es mediante la figura que muestra gráficamente el valor relativo de las tres funcionales (4-16).

Finalmente se muestra la respuesta al impulso a lazo cerrado de $y(t)$ y $u(t)$, en los cuales se puede evaluar directamente el cumplimiento de los requerimientos de diseño

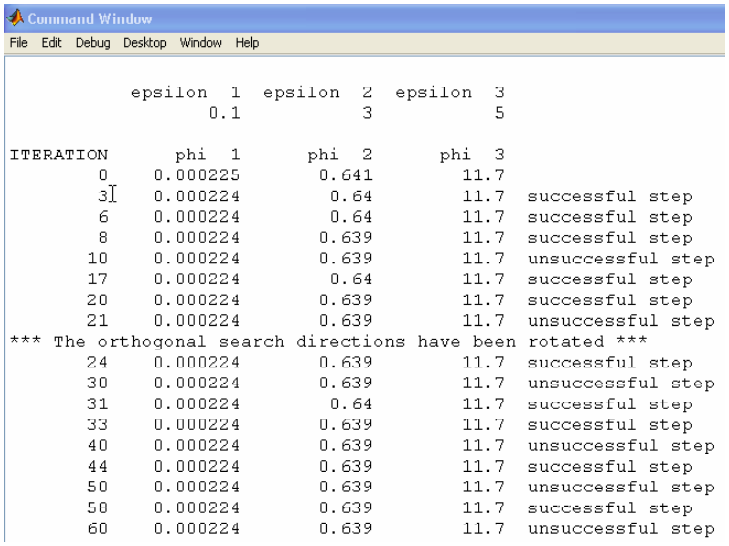

Figura 4-14: Inicio del algoritmo MBP

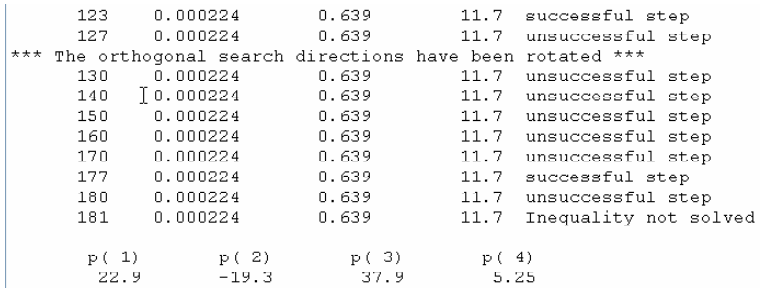

Figura 4-15: Final del algoritmo MBP

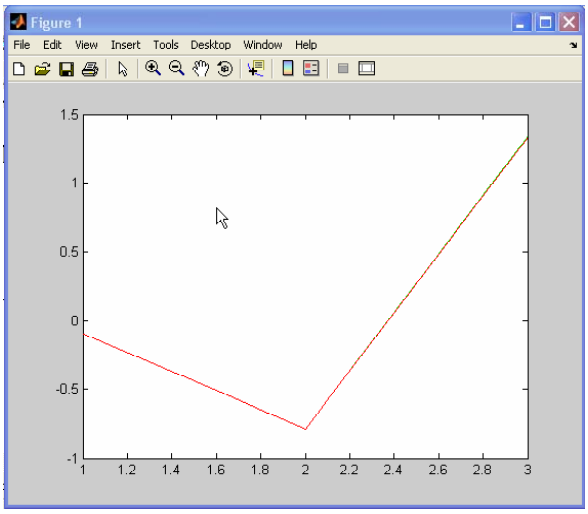

Figura 4-16: Gráfica del valor relativo de las tres funcionales.

(4-22).

Si en lugar del algoritmo MBP se selecciona la opción GESA, el resultado es como se muestra en la figura 4-18.

Note que se muestra la información relativa a la temperatura, el mejor valor conseguido para la función objetivo ("best value"), el número de padres activos y el lapso en minutos transcurridos desde el inicio. En la figura 4-19 se observa la etapa final de la ejecución de GESA. Note que en este caso el algoritmo es capaz de encontrar una solución al problema de diseño.

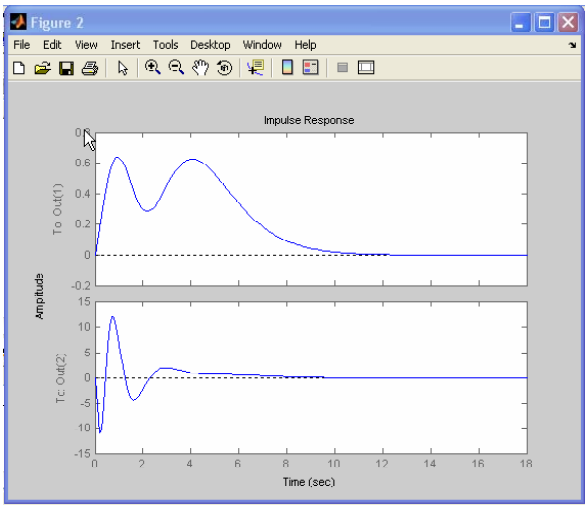

Figura 4-17: Respuesta al impulso a lazo cerrado de y(t) y u(t)

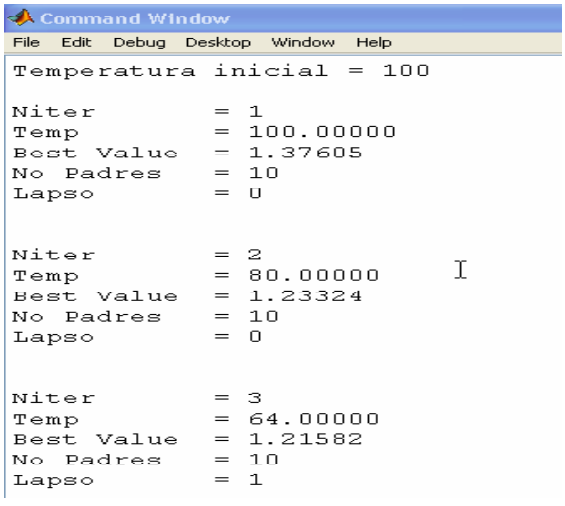

Figura 4-18: Inicio del algoritmo GESA

```
Niter        = 15
Temp         = 4.39805
Best Value   = 0.79630
No Padres    = 10
Lapso        = 1
                          I
Niter        = 16
Temp         = 3.51844
Best Value   = 0.79630
No Padres    = 10
Lapso        = 1


Niter        = 17
Temp         = 2.81475
Best Value   = -0.14851
No Padres    = 10
Lapso        = 1

Optimization terminated successfully ...
13 states removed.
```

Figura 4-19: Fin del algoritmo GESA

4.2.2 Control de un sistema SISO de orden 4 con acción integral

El problema que se considera a continuación es una adaptación del propuesto por J.F Whidborne para demostrar la eficiencia del "Toolbox" **MODCONS** (Gu. et al., 1994).

Considere la topología de control mostrada en la Figura 4-20, donde,

- r es la "señal de referencia".
- yg es la salida de la planta.
- $G(s) \in \mathcal{R}(s)^{1\times 1}$, es el modelo lineal de la planta a controlar, tal que:

$$G(s) = \frac{0.2s^2 + 0.3s + 1}{s^4 + 1.1s^3 + 1.3s^2 + s} \tag{4.5}$$

- $K \in \mathcal{R}(s)^{1\times 1}$, es el modelo lineal del controlador.

Se desea diseñar un controlador lineal $K(s)$ que estabilice internamente este lazo de control y tal que la respuesta $yg_s(t)$ para $r(t) = 1, t \geq 0$, cumpla los siguientes requerimientos:

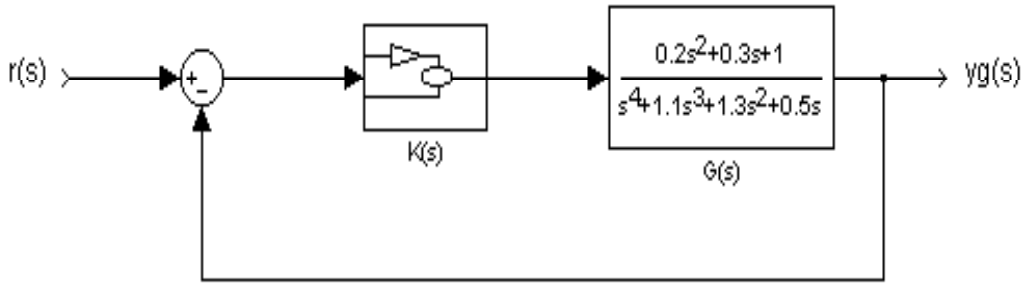

Figura 4-20: Ejemplo de diseño utilizando el método de representación del parámetro de Youla propuesto.

- La salida $yg_s(t)$ no debe sobrepasar un máximo de 1.1, esto es

$$\Phi_1 = \max_t yg_s(t) < 1.1$$

- El tiempo de establecimiento ("settling time"), denotado Φ_2 y definido como el tiempo a partir del cual se cumple,

$$0.95yg_s(\infty) < yg_s(t) < 1.05yg_s(\infty)$$

  debe ser menor que $4seg$, esto es,

$$\Phi_2 < 4$$

- La Banda Pasante a $-3dB$ del sistema a lazo cerrado, denotada Φ_3, debe ser menor que $10rad/\sec$, esto es

$$\Phi_3 < 10$$

Con el fin de aplicar el método propuesto, note que el lazo de control de la figura 4-20 puede ser redibujado como se muestra en la figura 4-21.

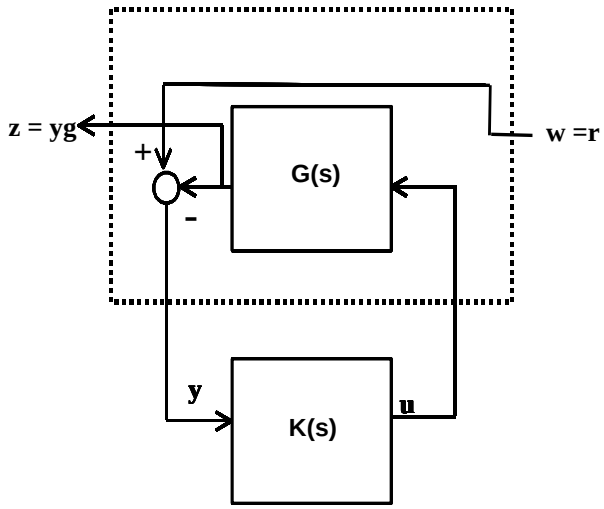

Figura 4-21: Otra forma de representar el lazo de control

Además es posible escribir:

$$\begin{pmatrix} z \\ y \end{pmatrix} = \begin{pmatrix} 0 & G \\ 1 & -G \end{pmatrix} \begin{pmatrix} w \\ u \end{pmatrix} \tag{4.6}$$

$$= \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} \begin{pmatrix} w \\ u \end{pmatrix} \tag{4.7}$$

Por otra parte, la representación en espacio de estado de la planta a lazo abierto mostrada en la Fig. 4-21 se escribe

$$\begin{cases} \dot{x} = Ax + B_1 w + B_2 u \\ z = C_1 x + D_{11} w + D_{12} u \\ y = C_2 x + D_{21} w + D_{22} u \end{cases} \tag{4.8}$$

con,

$$A = \begin{pmatrix} -1.3 & -0.7 & -0.25 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$B_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$C_1 = \begin{pmatrix} 0 & 0.1 & 0.15 & 0.5 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 & -0.1 & -0.15 & -0.5 \end{pmatrix}$$

$$D_{11} = D_{12} = 0$$

$$D_{21} = 1$$

$$D_{22} = 0$$

Para determinar la factorización coprima, los vectores F y L tales que $A + B_2F$ y $A + LC_2$ sean *Hurwitz*, son en valores numéricos:

$$F = (-8.7000 - 16.8000 - 24.7500 - 12.0000)$$

y

$$L = \begin{pmatrix} -29.0083 \\ 32.8333 \\ 46.3333 \\ -3.0667 \end{pmatrix}$$

Estos valores fueron determinados mediante el comando $place()$ de MATLAB de forma que

$$eig(A + B_2F) = eig(A + LC_2) = \begin{pmatrix} -4 \\ -3 \\ -2 \\ -1 \end{pmatrix}$$

donde $eig(A + B_2F)$ se refiere al vector de auto-valores de $A + B_2F$.

Luego, las funciones que intervienen en la factorización coprima de $G_{22} = -G$ son:

$$N_r = \frac{-0.2s^2 - 0.3s - 1}{s^4 + 10s^3 + 35s^2 + 50s + 24}$$

$$D_r = \frac{s^4 + 1.3s^3 + 1.4s^2 + 0.5s}{s^4 + 10s^3 + 35s^2 + 50s + 24}$$

$$N_l = N_r$$

$$D_l = D_r$$

$$X_r = \frac{-1409s^3 - 2253s^2 - 1831s - 576}{s^4 + 10s^3 + 35s^2 + 50s + 24}$$

$$Y_r = \frac{s^4 + 18.7s^3 + 144.3s^2 + 303.9s + 793.2}{s^4 + 10s^3 + 35s^2 + 50s + 24}$$

$$X_l = Xr$$

$$Y_l = Yr$$

El problema de optimización multi-objetivo puede plantearse en este caso como:

Resuelva,

$$\min_{Q \in RH_\infty} \left\{ \max_{i=1,2,3} \left(\frac{\Phi_i(Q) - \varepsilon_i}{\varepsilon_i} \right) \right\} \tag{4.9}$$

En este caso se establece $n_q = 2$, $nr = 0, ni = 1$ y además

$$A_{0Q} = \begin{pmatrix} -1 & 2 \\ 0 & -1 \end{pmatrix}$$

$$B_{0Q} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

de donde,

$$Q_\theta^{2,0,1} = vec(\theta) = C_Q(sI - A_Q)^{-1}B_Q + D_Q$$

$$p^{n_Q=2} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} \theta_1 + j\theta_2 \\ \theta_1 - j\theta_2 \end{pmatrix}$$

$$B_Q = \begin{pmatrix} \theta_3 \\ \theta_4 \end{pmatrix} \tag{4.10}$$

$$C_Q = \begin{pmatrix} \theta_5 & \theta_6 \end{pmatrix} \tag{4.11}$$

$$D_Q = \theta_7 \tag{4.12}$$

En la Figura 4-22 se muestra las respuestas al escalón unitario que resultan de los controladores diseñados al seleccionar los métodos **MBP** y **GESA** mediante la interfaz gráfica. En la Figura 4-23 se muestran los diagramas de Bode de amplitud y fase correspondientes.

Como punto de comparación, note que los valores que se obtienen para el caso trivial

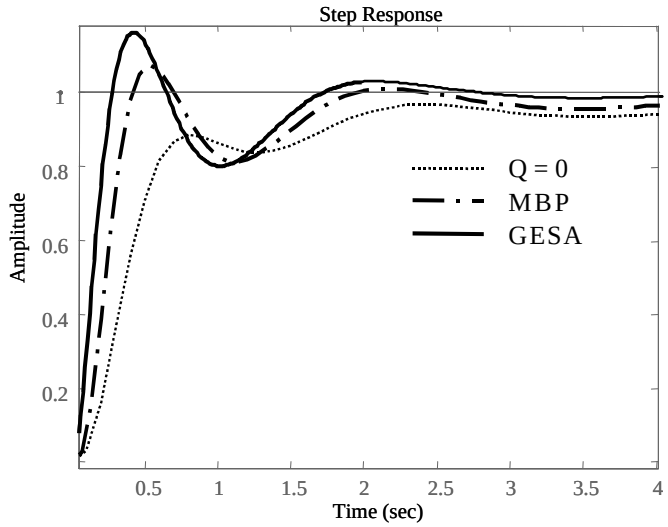

Figura 4-22: Respuestas a un escalón unitario que resultan luego de aplicar las técnicas **MBP** y **GESA**. La línea de puntos corresponde al caso trivial $Q = 0$.

$Q = 0$ son los siguientes:

$$\begin{aligned} \Phi_1 &= 0.95 \\ \Phi_2 &= 10 \\ \Phi_3 &= 1.92 \end{aligned}$$

Como puede apreciarse en las figuras 4-22 y 4-23, en los casos de los algoritmos **MBP** y **GESA** se llega sin mayores dificultades a soluciones que cumplen con los requerimientos del problema, caracterizadas por

$$\begin{aligned} \Phi_1^{MBP} &= 1.1379 \\ \Phi_2^{MBP} &= 1.7262 \\ \Phi_3^{MBP} &= 1.8157 \end{aligned}$$

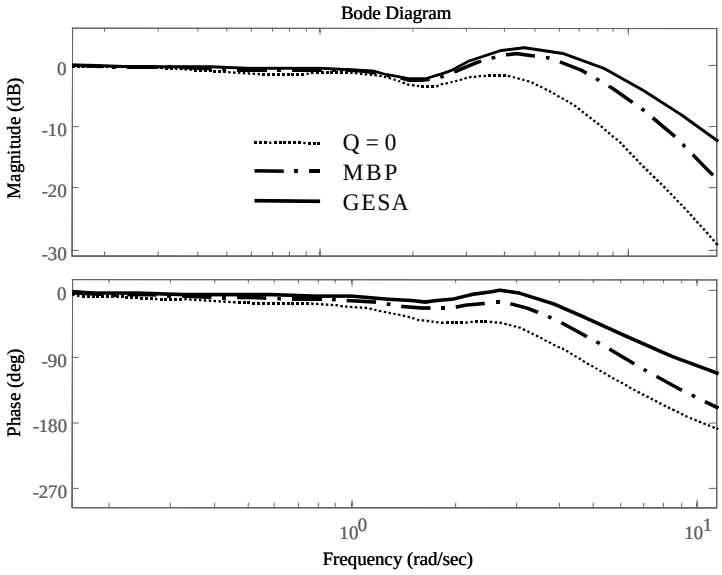

Figura 4-23: Diagramas de Bode que resultan luego de aplicar las técnicas **MBP** y **GESA**. La línea de puntos corresponde al caso trivial $Q = 0$.

y

$$\begin{aligned} \Phi_1^{GESA} &= 1.1615 \\ \Phi_2^{GESA} &= 1.5288 \\ \Phi_3^{GESA} &= 1.9577 \end{aligned}$$

Note que en ambos casos el diseño podría ser mejorado de ser necesario, puesto que se tiene un amplio margen con respecto al requerimiento de banda pasante.

4.2.3 Problema de control $\mathcal{H}_2$

En este problema nos proponemos minimizar la norma $\mathcal{H}_2$ de dos funciones de transferencia de interés $\hat{G}_{z_1 w_{cl}}$ y $\hat{G}_{z2w_{cl}}$. El modelo de la planta fue tomado de (Hu et al., 1995). Considere la siguiente topología de control mostrada en la figura 4-24, en la cual:

$$P = \frac{-s + 10}{s^2 - 0.5s + 1}$$

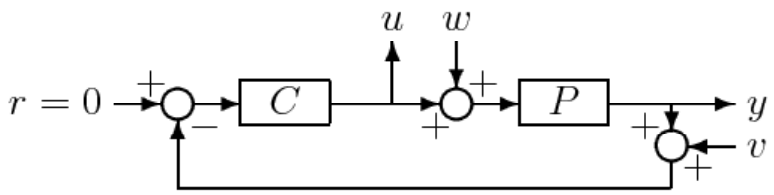

Figura 4-24: Topología de control para el ejemplo de control H2

Para simplificar suponga $v = 0$. $\hat{G}_{z_1 w_{cl}}$ corresponde a la transferencia de w hacia y. por su parte $\hat{G}_{z_2 w_{cl}}$ corresponde a la transferencia de w hacia u. Las matrices de estado, en valores numéricos, son las siguientes :

$$\begin{aligned}
A &= \begin{bmatrix} 0 & -0.5 \\ 2 & 0.5 \end{bmatrix} \\
B_1 &= \begin{bmatrix} 1.25 \\ -0.25 \end{bmatrix} \\
B_2 &= \begin{bmatrix} 1.25 \\ -0.25 \end{bmatrix} \\
C_1 &= \begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix} \\
C_2 &= \begin{bmatrix} 0 & 4 \end{bmatrix}
\end{aligned}$$

	epsilon 1	epsilon 2	epsilon 3	
	5	5	1	
ITERATION	phi 1	phi 2	phi 3	
0	3.29e+003	323	0	
5	3.2e+003	314	0	successful step
6	3.04e+003	299	0	successful step
7	2.89e+003	284	0	successful step
8	2.89e+003	284	0	successful step
9	2.69e+003	263	0	successful step
10	2.27e+003	222	0	successful step
11	1.91e+003	187	0	successful step
12	1.91e+003	187	0	successful step
13	1.6e+003	155	0	successful step
14	710	69	0	successful step
15	324	31.4	0	successful step
16	315	30.6	0	successful step
17	227	21.6	0	successful step

Figura 4-25: Evolución de las funcionales para el ejemplo 2

$$
\begin{aligned}
D_{11} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
D_{12} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
D_{21} &= 0 \\
D_{22} &= 0
\end{aligned}
$$

Se desea diseñar un controlador lineal que estabilice internamente este lazo de control y tal que:

$$\Phi_1 = \left\| \hat{G}_{z_1 w_{cl}} \right\|_2 < 5$$

$$\Phi_2 = \left\| \hat{G}_{z_1 w_{cl}} \right\|_2 < 5$$

Después de introducir los datos en CONTROL-DESIGN, programar las funcionales Phi 1 y 2 se consigue la evolución mostrada en la figura 4-25 cuando se utiliza el algoritmo MBP. La parte final de esta evolución se muestra en la figura 4-26. Note que el algoritmo no tiene ningun problema para conseguir una solución al problema de diseño.

```
       66        5.06        1.58           0  successful step
       70        5.06        1.58           0  unsuccessful step
       71        5.05        1.57           0  successful step
       72        4.92        1.62           0  successful step

result =

  Inequality solved

       72        4.92        1.62           0  Inequality solved

       p( 1)       p( 2)       p( 3)       p( 4)
      -0.507       0.374       0.573       -1.79
7 states removed.
```

Figura 4-26: Evolución de las funcionales para el ejemplo 2. Parte final.

4.3 Información práctica sobre CONTROL-DESIGN

Los archivos necesarios para ejecutar y eventualmente modificar la interfaz gráfica se encuentran disponibles libremente en INTERNET, en la dirección:

http://prof.usb.ve/gsanchez/design.html

CAPITULO 5

CONCLUSIONES

5.1 Contribuciones Principales

A partir de los resultados presentados, se demuestra que las ventajas de la herramienta diseñada CONTROL-DESIGN son las siguientes:

- Permite resolver problemas de diseño de controladores de baja complejidad (escalares, con pocos requerimientos).
- Es interactiva.
- Permitir al diseñador modificar los parámetros de búsqueda.
- Es amigable para el usuario.
- El tiempo para aprender a utilizarla es reducido.

5.2 Lineas futuras de investigación

- Se debe avanzar en las siguientes direcciones: mejora de las interfaces y de los métodos de búsqueda, disminución de la intervención del usuario, etc.
- Desarrollo de métodos para construcción del vector de decisión, los cuales permitan, al mismo tiempo, la optimización parámetrica y estructural.
- Consideración del problema de robustez de diseño y problemas multivariables.
- Consideración del problema de optimización del orden de los controladores, de tal forma que éstos puedan ser implementados cómodamente en problemas prácticos.
- Consideración del problema de modelaje e identificación de la planta dentro del "Toolbox" de diseño, aplicando igualmente técnicas de optimización inteligente.

BIBLIOGRAFIA

Alfonzo, M. (1992). Proposicić́n de un curso de laboratorio de instrumentacić́n y control por computador. *Trabajo de Ascenso para optar a la categorąa de Agregado. Universidad Simć́n Bolą́var.*

Benner, Mehrmann, Sima, Huffel, V., and Varga (1997). Slicot: A subroutine library in systems and control theory. *European Community BRITE-EURAM III Thematic Networks Programme NICONET - Working Group on Software. ESAT Katholieke Universiteit Leuven.*

Boyd, S., Balakrishan, V., Barrat, C., Khraishi, N., Li, X., Meyer, D., and Norman, S. (March, 1988). "A new CAD method and associated architectures for linear controllers". *IEEE Transactions on Automatic Control, Vol AC-33, No.3.*

Chawla, Ragade, and Deshpande (1988). Proceed: an expert system for multivariate process control systems design. *Proceedings of the First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Tullahoma, Tennessee, United States.*

Clement, B. (2001). *"Synthese multiobjectifs et sequencement de gains: Application au pilotage d'un lanceur spatial.* PhD thesis, Service Automatique de SUPELEC - Gif sur Yvette, France.

Gembicki, F. W. and Haimes, Y. Y. (1975). "Approach to performance and sensitivity multiobjective optimization". *IEEE Transactions on Automatic Control, December*, pages 769–771.

Giesy, D. (1978). "Calculation of Pareto-Optimal solutions to multiple-objective problems using threshold of acceptability constraints". *IEEE Transactions on Automatic Control, Vol. AC-23, No-6*, pages 1114–1115.

Grubel, G. (2005). Scope of computer-aided control system design. *[On-line]. Consultado en: http://www.laas.fr/cacsd/scope.html. 27 de abril de 2005.*

Gu., D., Whidborne, J., and Postlethwaite, I. (1994). "MODCONS- a MATLAB Toolbox for multi-objective control system design". *Technical Report 94-26 Leicester University Eng. Dept. Leicester, U.K.*

Gustafson, C. and Desoer, C. (1983). "Controller design for linear multivariable feeback systems with stable plants, using optimization with inequality constraints". *International Journal of Control. Vol. 37. Nø 5*, pages 881–907.

Hindi, H. A., Hassibi, B., and Boyd, S. P. (1998). "Multiobjective H2/Hinf optimal control via finite dimensional q-parametrization and linear matrix inequalities". *Proceedings of the American Control Conference, June*, pages 3244–3249.

Hooke, R. and Jeeves, T. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8:212–219.

Hu, Z., Salcudean, E., and Loewen, P. (June, 1995). "Numerical solution of the multiple objective control system design problem for SISO systems". *Proceedings of the American Control Conference ACC-95 Seattle, Washington, USA*, pages 1458,1462.

IEEE (1984). "Special issue on CACSD systems". *Proceedings of the IEEE*, 72(12).

Jimenez, F. and Sanchez, G. (2002). Computacion evolutiva. Technical report, Universidad de Murcia. Departamento de Ingenieria de la Informacion y las Comunicaciones.

Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). "Convergence properties of the Nelder-Mead simplex method in low dimensions". *SIAM J. Optim*, 9(1):112–147.

Nye, W. (1981). Delight: An optimization based computer-aided design system. *Proceedings of the IEEE ISCAS Chicago, IL*.

Polak, E., Mayne, D. Q., and Stimler, D. M. (1984). "Control system design via semiinfinite optimization: A review". *Proceedings of the IEEE, Vol.72, No.12*, pages 1777–1793.

Rosenbrock, H. (1960). "An automatic method for finding the greast or least value of a function". *Computer Journal*, pages 175–184.

Sanchez, G. (2003). Diseno de un controlador con multiples objetivos para un reactor quimico mediante tecnicas de inteligencia artificial. *Trabajo de Grado presentado ante la USB para optar al titulo de Magister*.

Vidsayagar, M. (1984). "Control systems synthesis: A factorization approach". *M.I.T Press*.

Whidborne, J., Gu, D., and Postlethwaite, I. (1997). "Simulated annealing for multiobjective control system design". *IEE Proc.- Control Theory Appl. Vol 144. N0.6 November*, pages 582–588.

Wie, B. and Bernstein, D. (1992). Benchmark problems for robust control design. *American Institute of Aero. and Astro.*, 15(5):1057–1059.

Yip, P. (1993). *The role of regional guidance in optimization: The Guided Evolutionary Simulated Annealing approach*. PhD thesis, Department of Electrical Engineering and Applied Physics, Case Western Reserve University.

Youla, D., Jabr, H., and Bongiorno, J. (1976). "Modern Wiener-Hopf design of optimal controllers - part II : The multivariable case". *IEEE Transactions on Automatic Control, Vol. AC-21, No. 3*, pages 319–338.

Zakian, V. and Al-Naib, V. (1973). "Design of dynamical and control systems by the Method of Inequalities". *Proceedings of the Institution of Electrical Engineers 120(11)*, pages 1421–1427.