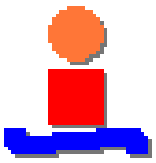


Proyecto Final de Carrera

DISEÑO DE ESTRATEGIAS DE CONTROL PARA UN SISTEMA DE LEVITACIÓN NEUMÁTICA

Autor: D. Fernando-Borja Breña Lajas
Tutor: D. Manuel G. Ortega Linares

Departamento de Ingeniería de Sistemas y Automática
Escuela Superior de Ingenieros
Universidad de Sevilla

Septiembre 2005

A mi familia.
A mis amigos.

Índice

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción.

1.1. Ingeniería de Control.

1.1.1. Conceptos generales.

La *Ingeniería de Control* es una disciplina que tiene como objetivo lograr un nivel deseado de rendimiento para un sistema frente a perturbaciones o incertidumbres en el proceso.

Un *sistema* es un conjunto de elementos o partes organizados con alguna forma de relación que interactúan entre sí y con su ambiente para lograr objetivos comunes, operando con entradas como información, energía, materia u organismos para producir ciertas salidas.

Un *sistema de control* es aquél que influye sobre la salida del sistema controlado para que mantenga un valor deseado bajo ciertas condiciones o para que varíe según la entrada al sistema. Existen dos formas básicas de sistemas de control: en *bucle (o lazo) abierto* y en *bucle (o lazo) cerrado*. Con un sistema en bucle abierto, para producir el valor de salida requerido, la entrada se elige según indique la experiencia con el sistema controlado.

En un sistema en bucle cerrado se tiene una señal de realimentación desde la salida hacia la entrada. La entrada global al sistema de control es el valor requerido (*referencia* o *consigna*) para la variable controlada y la salida es el valor actual de dicha variable. Tras comparar la referencia con la medida del valor de salida y obtener de esta forma el *error*, el *controlador* decide que acción correctora se debe realizar sobre el sistema. El

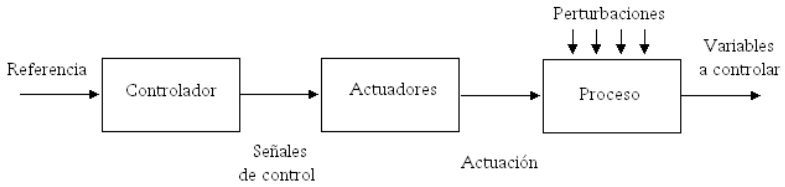


Figura 1.1: Esquema general de un sistema de control en bucle abierto.

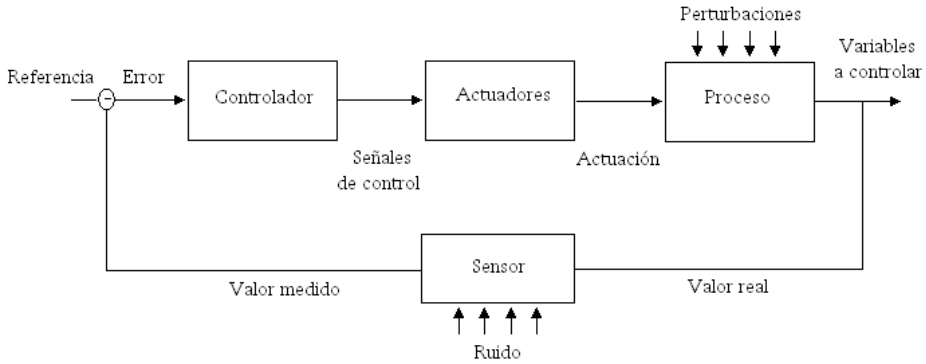


Figura 1.2: Esquema general de un sistema de control en bucle cerrado.

actuador ejecutará la señal de control emitida para producir un cambio en el *proceso* controlado y así conseguir el valor deseado para la variable de salida.

Comparando los resultados que se obtienen con un sistema de control en bucle cerrado (BC) frente a los que se obtienen con un sistema en bucle abierto (BA) se observa principalmente que:

1. El BC consigue mayor exactitud en la igualación de los valores real y requerido para la salida.

2. El BC presenta menor sensibilidad a las perturbaciones y a cambios en las características de los componentes.

3. El BC tiene mayor ganancia.

4. En el BC disminuye la posibilidad de inestabilidad.

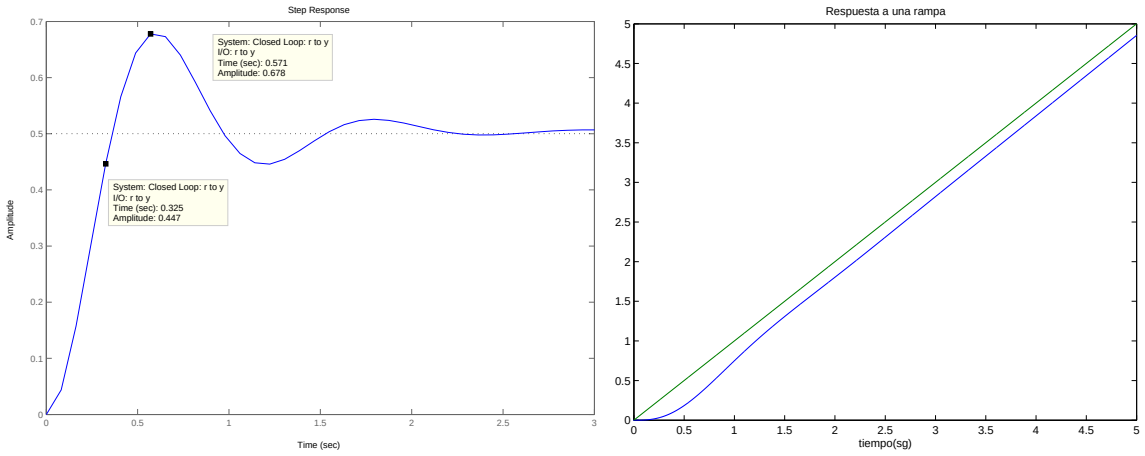
5. El BC es un sistema de control más complejo.

Aunque el control en bucle cerrado presenta algunos inconvenientes, sus ventajas son suficientes para preferir casi siempre este esquema de control.

El *controlador* es un elemento en el sistema en bucle cerrado que tiene como entrada la señal de error y produce una salida que se convierte en entrada del actuador o elemento correctivo (ver figura 1.2). La relación entre la salida y la entrada al controlador se suele denominar *ley de control*. Hay multitud de estrategias para definir la ley de control. Una vez escogida la estrategia de control, al proceso de selección de los mejores valores para el controlador se le llama *sintonización*.

Las *especificaciones de control* son las exigencias que se desean imponer al comportamiento del proceso. Estas exigencias suelen definirse como condiciones temporales que debe cumplir la respuesta final del sistema controlado. Por ejemplo, en la figura 1.3 se muestran las respuestas de un sistema controlado que cumple el siguiente conjunto de especificaciones:

- Estabilidad del sistema.
- Sobreoscilación menor o igual al 30 % en la respuesta a escalón.
- Tiempo de subida menor o igual a 1 segundo en la respuesta a escalón.
- Error de seguimiento en velocidad menor o igual al 0.1



(a) Respuesta a escalón. (b) Seguimiento en velocidad.

Figura 1.3: Ejemplo de respuesta de un sistema controlado.

1.1.2. Historia y actualidad.

El concepto de control por realimentación no es nuevo. El primer lazo de realimentación fue usado en 1774 por James Watt para el control de la velocidad de una máquina de vapor. En 1868, J.C.Maxwell proporcionó el primer análisis matemático

riguroso de un sistema de control realimentado. La aplicación de los bucles de control fue lenta hasta que en 1940 se hicieron comunes los sistemas de transmisión neumática. El control por realimentación es parte del progreso de la segunda revolución industrial y, en la actualidad, el control automático de procesos es un elemento esencial para la industria.

El control automático de procesos es ventajoso porque reduce el coste de dichos procesos y aumenta la productividad, lo que compensa la inversión en el equipo de control. Además, provoca que la mano de obra pasiva deba ser sustituida por otra especializada, lo que redunda en la reducción de fallos en el proceso.

El uso de las computadoras ha permitido la aplicación de ideas de control automático sobre sistemas físicos, que antes eran imposibles de analizar o comprobar.

Por todos estos motivos, la Ingeniería de Control está presente, de una u otra forma, en todos los sistemas modernos de ingeniería.

1.2. Motivación del Proyecto.

La idea de un dispositivo de levitación neumática surgió como propuesta útil para poner en práctica los conocimientos teóricos sobre Ingeniería de Control. La propuesta fue desarrollada posteriormente con motivo de la obtención del Premio Omron 2005 "Iniciación a la investigación e innovación en Automática"por los alumnos F.B. Breña, J.J. Caparrós y J.A. Pérez.

Este dispositivo es una herramienta construida integrando elementos mecánicos y electrónicos y software de controladores que podrían ser incorporados en variados procesos productivos. El empleo de dicho dispositivo ayudará a los usuarios a comprender aspectos generales del proceso de identificación de una planta y del diseño de controladores. Por tanto, este proyecto generaría un producto de indudable valor didáctico que podrá ser utilizado en posteriores prácticas de laboratorio en el Departamento de la Universidad.

El desarrollo de plantas didácticas para realizar prácticas es una herramienta fundamental para el aprendizaje de una rama de la Ingeniería como es el Control Automático. Contar con un dispositivo de control no convencional en el ámbito académico brindará a los estudiantes la posibilidad de desarrollar habilidades en las técnicas del control de procesos industriales.

1.3. Objetivos del proyecto.

1.3.1. Desarrollo previo.

El presente Proyecto tiene su origen en otro anterior cuyo objetivo era la construcción, la identificación y el modelado de un sistema de levitación neumática ([Cap05]). Este sistema, denominado **SISTEMA LEVINEU**, se encuentra en el Laboratorio del Departamento de Ingeniería de Sistemas y Automática. Dicho proyecto se encargó de:

- El estudio teórico del sistema, esto es, los fundamentos físicos, matemáticos y electrónicos necesarios para la viabilidad del proyecto
- La construcción y puesta a punto del sistema en el laboratorio.
- La configuración del hardware y el software necesarios para la adquisición y monitorización de los datos relativos al sistema.
- La identificación de la planta.
- El modelado matemático de la planta.

1.3.2. Nuevos objetivos.

Teniendo como base este proyecto anterior, se ha fijado como objetivo general realizar el control en tiempo real de la altura de un objeto suspendido dentro del flujo de aire y atrapado por él con cierta fuerza.

En este Proyecto se tratarán de obtener controladores para el Sistema Levineu mediante diversas estrategias con la condición común de que la respuesta de dicho sistema cumpla una calidad mínima.

En este Proyecto no se han pretendido alcanzar unas especificaciones de control determinadas, sino que se ha buscado desarrollar una plataforma software a partir de la cual sea posible profundizar en el estudio de diversas estrategias de control. Se hace notar que las especificaciones de control elegidas en un futuro deben tener en cuenta la complejidad del medio aéreo.

Se trabajará con un sistema en bucle cerrado realimentado unitariamente, con la intención de conseguir beneficios tales como mayor estabilidad y flexibilidad, resistencia a pequeños cambios de las variables, mejor consecución de objetivos, etc.

Las estrategias de control que se pretenden analizar son:

- Clásicas: Control PID.
- Robustas: Control H_∞.
- Predictivas: Control Predictivo Generalizado (GPC).
- No lineales: Control Borroso.

Tras estudiar cada una de ellas se compararán los resultados obtenidos y, por último, se propondrán conclusiones y futuras líneas de trabajo.

1.4. Organización de la documentación.

La documentación del proyecto se desglosa en los siguientes ítems:

- **Memoria descriptiva**. La presente memoria resume tanto los planteamientos teóricos adoptados, como los resultados prácticos obtenidos durante el desarrollo del proyecto. En el capítulo 2 se describe el equipo utilizado para llevar a cabo el proyecto. En el capítulo 3 se introduce el denominado Sistema Levineu. En el capítulo 4 se presentan los controladores clásicos diseñados.En el capítulo 5 se diseñan controladores H_∞. En el capítulo 6 se diseña un controlador predictivo generalizado (GPC). En el capítulo 7 se diseña un controlador borroso. En el capítulo 8 se explica el software desarrollado. En el capítulo 9 se comparan los resultados y se extraen conclusiones.
- **Apéndices**. Contiene los siguientes apartados:
  1. Explicación detallada del programa desarrollado en CX-Programmer.
  2. Códigos fuente: Scripts de Matlab.
- **CD-ROM**. En este soporte se incluyen:
  1. Memoria del proyecto y apéndices.
  2. Ficheros de datos resultado de los experimentos.
  3. Programas software desarrollados en Labview, Matlab y CX-Programmer.
  4. Imágenes, fotos, videos.
  5. Otra documentación.

Capítulo 2

Equipo utilizado en el proyecto.

2.1. Hardware.

En primer lugar se debe hacer constar que, en un proyecto como el presente, el 'hardware' ocupa un lugar destacado y, por tanto, a su estudio y preparación se dedicó largo tiempo. Hay muchos componentes involucrados y surgieron muchos problemas en la construcción del sistema. Todo esto es asunto que se detalla profusamente en otro Proyecto Fin de Carrera ([Cap05]).

A continuación se ofrece un listado de los elementos principales del sistema.

Autómata programable Omron CJ1M.

- CPU 160 E/S 5Kpasos 32KW (CJ1MCPU11).
- Fuente de alimentación 100 a 240Vca 5Vcc 2,8A Relé (CJ1WPA202).
- Módulo 2 Salidas analógicas (CJ1WDA021NL).

Sensor de visión Omron F150.

- Unidad procesadora de imagenes V3 NPN (F150C10E3).
- Cámara CCD sin lente (F150S1A).
- Consola de programación (F150KP).
- Cable conexión cámara (F150VS).
- Cable conexión monitor (F150VM).

Variador de frecuencia Omron F7.

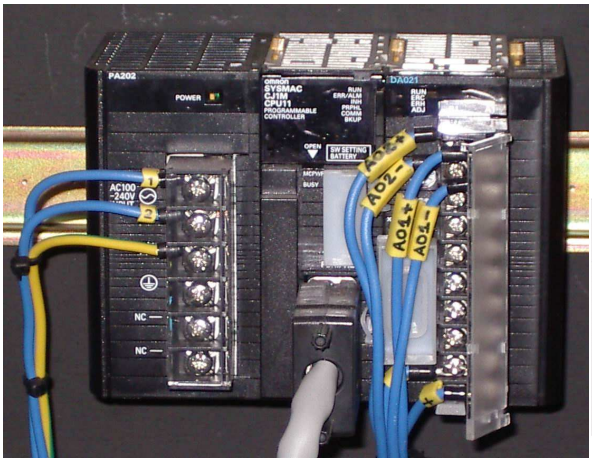
- Modelo CIMR-F7Z40P41B: Trifásico, 400V, 0.4KW, control vectorial.

Soplador Soler&Palau.

- Motor ABB. Modelo 3GVA071001-CSC: Trifásico, 2870rpm, 0,4KW.

PC Pentium 4 a 3.2Ghz. En el PC se ha instalado el siguiente software básico para la aplicación.

- CX-Programmer 5.0.
- Labview 7.1.
- Matlab 6.5.
- Microsoft Visual C++ 6.0.



(a) Autómata Omron CJ1M.

(b) Sensor de visión Omron F150

(c) Variador de frecuencia Omron F7.

(d) Motor soplador ABB.

Figura 2.1: Elementos 'hardware' del sistema.

2.2. Software.

Se emplearon varios paquetes de software de diversas características.

CX-Programmer De Omron Electronics Inc. Se empleó la versión 5.0. CX-Programmer es un entorno gráfico de configuración, desarrollo y depuración de aplicaciones para autómatas Omron. CX-Programmer se ejecuta en Windows y con él se pueden programar todos los tipos de autómatas Omron, desde micro-PLCs hasta las series más potentes como CJ/CS, proporcionando toda la potencia de programación que se necesita, incluso para el desarrollo de sistemas muy complejos y con múltiples dispositivos. Actualmente los lenguajes soportados compatibles con IEC-61131-3 son IL, LD, ST, FB. En este Proyecto se emplearon LD y FB.

El funcionamiento del CX-Programmer está explicado en el manual rápido de uso elaborado para este Proyecto; asimismo, Omron proporciona una profusa documentación (en inglés y español) incluida en el CD adjunto.

Matlab De la firma The Mathworks, Inc. Se empleó la versión 6.5 (release 13). MATLAB es un lenguaje de programación de alto nivel para aplicaciones técnicas y un entorno para el desarrollo de algoritmos, visualización y análisis de datos y cálculo numérico. MATLAB permite resolver problemas de programación técnicos más rápido que con los lenguajes de programación generales, como C, C++ o Fortran. Matlab puede emplearse para muchos tipos de aplicaciones: procesamiento de señal e imagen, comunicaciones, diseño de control, pruebas y medidas, modelado y análisis de sistemas, etc. Esto se consigue mediante módulos llamados 'toolboxes', que son colecciones de funciones Matlab de propósito específico, disponibles separadamente. Otro aspecto interesante es que se puede integrar código Matlab con otros lenguajes y aplicaciones.

LabView De National Instruments Inc. Se empleó la versión 7.1. NI LabView es un entorno de desarrollo gráfico para crear de forma rápida y con bajo coste aplicaciones flexibles y escalables de prueba, medida y control. LabView permite interactuar con las señales del mundo real, analizar datos y compartir resultados y aplicaciones, todo ello a través del PC. LabView facilita el diseño de interfaces gráficas preparadas para todo tipo de usuarios.

WinEdt Shareware. Se empleó la versión 5.3. WinEdt es un editor ASCII versátil y muy potente. Se ejecuta en MS Windows y está muy preparado para la creación de documentos TeX o LaTeX. WinEdt es ampliamente utilizado como una interfaz de escritura y compilación para diversos sistemas, como TeX o HTML. WinEdt proporciona esquemas iluminados de los códigos fuente que pueden ser personalizados según diferentes modos; además, su función de comprobación de ortografía soporta multitud de idiomas, con diccionarios (listas de palabras) disponibles para descargar de internet.

Capítulo 3

Estudio del sistema.

3.1. Descripción del sistema.

Para desarrollar experimentalmente este proyecto se ha empleado un sistema compuesto por un autómata programable (PLC), un motor ventilador, un variador de velocidad, un sensor de visión y un ordenador personal (PC) que constituye el eje central del sistema. La figura 3.1 muestra una vista general del Sistema Levineu.

El proceso que se desarrolla en el Sistema Levineu está esquematizado en la figura 3.2.

La bola es sustentada a cierta altura gracias a una corriente vertical de aire proporcionada por el motor soplador. El sistema de visión calcula la altura de la bola y envía el dato al ordenador y al autómata, a través de aquél. Para cerrar el bucle de control, el autómata indica la frecuencia de salida del variador de velocidad que controla la frecuencia de giro del motor.

El soplador es un motor centrífugo de corriente alterna. La velocidad y el arrastre del flujo de aire varían con la frecuencia de la entrada del motor, determinada por el variador de frecuencia F7 de Omron. El variador se ha programado para que su frecuencia de salida, controlada por el autómata mediante una entrada analógica de 0 a 10 Voltios, varíe entre 20 y 30 Hz, lo cual interesa para controlar la altura de la bola entre 0 y $60cm$, aprovechando el comportamiento casi lineal de ésta respecto a la frecuencia.

El cálculo de la señal de control se realiza en el autómata o en el PC, para lo cual se recibe la altura medida de la bola y la referencia. La comunicación PC-autómata se

Figura 3.1: Vista general de la infraestructura del Sistema Levineu.

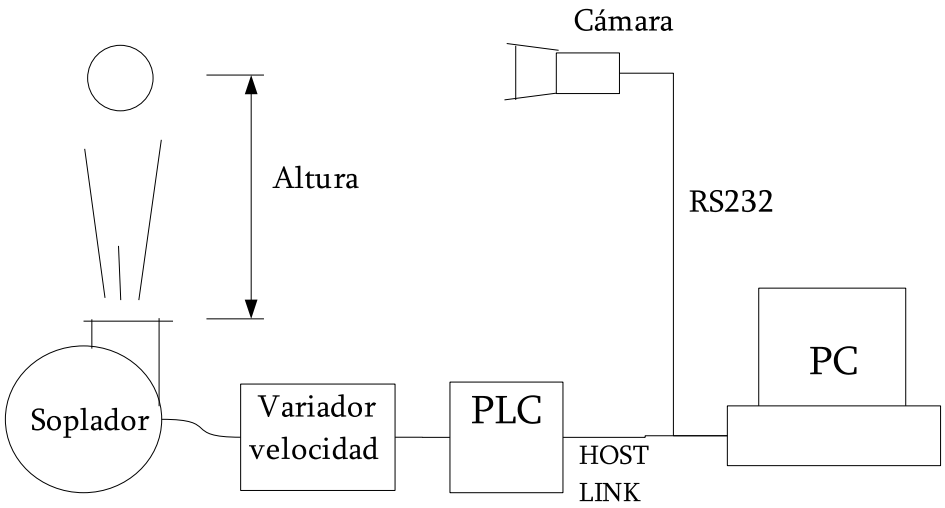


Figura 3.2: Esquema general del Sistema Levineu.

realiza mediante el protocolo Host-Link desarrollado por Omron, que funciona sobre un protocolo RS-232.

Desde el PC se envían los comandos adecuados para solicitar el valor de la altura en cada iteración del bucle de control. El sistema de visión F150 de Omron se ha configurado para que cuando llegue una petición para efectuar una nueva medida, procese la imagen capturada por la cámara y calcule la altura del centro de la pelota respecto a la boca del soplador como el centro de gravedad de la imagen binarizada. Para facilitar este procesamiento se ha dispuesto un fondo negro que contrasta con el color blanco de la bola. Este dato de la altura se le pasa al PC mediante el puerto serie con una serie de comandos entendidos por el controlador.

En el ordenador reside la aplicación que controla el funcionamiento del equipo, desarrollada en LabVIEW. El ordenador se emplea, por tanto, para monitorizar el proceso y configurar el control. Se ha programado una interfaz gráfica en LabView para facilitar estas tareas. La aplicación de control permite elegir entre los distintos controladores para actuar en el equipo, enviando al autómata una trama de configuración para que active la sección correspondiente al control seleccionado. Esta aplicación también se encarga de la recogida y almacenamiento de datos para el análisis posterior, almacenándolos en un fichero que puede ser monitorizado por Matlab. Toda la aplicación se explica en el capítulo 8.

Se han ensayado distintas estrategias de control preparadas de diversa forma:

- **PID**: Se han preparado dos versiones, ambas íntegramente en el autómata. Una utiliza la instrucción PID del juego de instrucciones del autómata; se ha programado con lenguaje estructurado en un bloque específico del autómata.
- H_∞: El controlador se diseña en Matlab pero el cálculo de la señal de control en tiempo real se realiza en el autómata.
- **GPC**: La señal de control se calcula en Matlab en cada iteración y se envía al autómata para que la pase al variador.
- **Borroso**: Programado íntegramente en el autómata.

3.2. Modelado del proceso de levitación.

El problema que se plantea es el modelado de un proceso de levitación neumática en el que un ventilador expulsa un chorro de aire vertical manteniendo en posición de equilibrio una pelota cuyas características son conocidas.

El trabajo de identificación y modelado de la planta está detallado en [Cap05]. Para ello se realizaron una larga serie de desarrollos teóricos, simulaciones y experimentos. Esta sección hará uso de sus resultados.

Para estudiar el comportamiento temporal del sistema hay que evaluar su ecuación dinámica. Se realizó una simulación en bucle abierto (BA) con una entrada de 2.5 V. En la figura 3.3 se observa la simulación junto con la correspondiente salida del sistema real. Se aprecia un comportamiento oscilatorio alrededor del punto de equilibrio nominal, lo que induce a pensar en la existencia de un ciclo límite. Las amplitudes de las representaciones teórica y real varían, aunque no en exceso, lo que se puede explicar por las perturbaciones en el sistema real o por errores de modelado. En cambio, los periodos de las oscilaciones real y simulada son muy parecidos, en torno a 1.8 s.

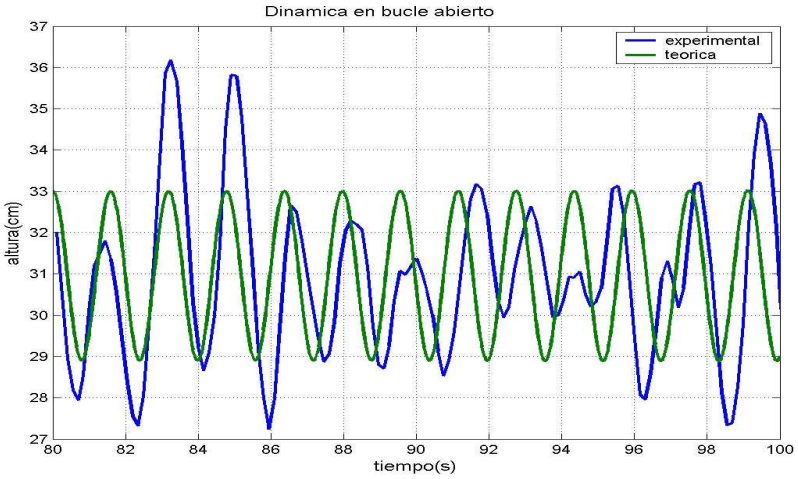


Figura 3.3: Comportamiento dinámico del sistema en bucle abierto.

Por otro lado, los experimentos sobre la planta real dieron como resultado las siguientes funciones de transferencia en distintos puntos de operación.

1. Punto de trabajo 1: tensión de entrada=1V.

$$G_1(s) = \frac{2{,}4899s^2 - 93{,}3089s + 1941{,}1931}{s^3 + 20{,}1682s^2 + 24{,}3028s + 244{,}7105} \tag{3.1}$$

2. Punto de trabajo 2: tensión de entrada=4V.

$$G_2(s) = \frac{1{,}2273s^2 - 74{,}3129s + 1136{,}1254}{s^3 + 16{,}8837s^2 + 26{,}4839s + 274{,}7454} \tag{3.2}$$

3. Punto de trabajo 3: tensión de entrada=7.5V.

$$G_3(s) = \frac{1{,}0000s^2 - 30{,}9559s + 398{,}8621}{s^3 + 27{,}4029s^2 + 80{,}8324s + 401{,}5041} \tag{3.3}$$

3.3. Estudio del sistema sin compensar.

Antes de analizar el control del sistema en secciones posteriores, estudiamos el comportamiento del sistema sin compensar. Para ello, tomaremos como planta $G(s)$ la función de transferencia calculada para el punto de operación 2 (ver sección 3.2). A este punto de operación se le denominará en adelante *punto nominal*. Dicha planta tiene los siguientes polos y ceros:

- Ceros: $c_{1,2} = 30{,}2747 \pm 3{,}0245i$.
- Polos: $p_1 = -16{,}2932; p_{2,3} = -0{,}2953 \pm 4{,}0958i$.
- Ganancia estática: $K_{BA} = 1{,}2273$

El polo real p_1 es de dinámica mucho más rápida que los polos complejos conjugados (c.c.), pues la constante de tiempo de aquél es de orden de magnitud inferior a la de éstos. En cuanto al numerador, es de fase no mínima pues tiene dos ceros c.c. en el semiplano derecho (SPD).

Para predecir la estabilidad se puede construir la *tabla de Routh*. Esta tabla permite determinar el semiplano donde se localizan las raíces de un polinomio. El criterio de Routh-Hurwitz indica que un sistema es estable dada su función de transferencia si, al construir la tabla de Routh de su denominador, no hay variación en el signo de los elementos de la primera columna. Esto significa que el denominador no tendrá polos en el SPD y es, como ya se vio al principio de esta sección, lo que sucede con el sistema en BA sin compensar (tabla 3.1).

$$D_{BA} = s^3 + 16{,}8837s^2 + 26{,}4839s + 274{,}7454 \tag{3.4}$$

En cambio, el criterio predice que el sistema en BC sin compensar será inestable, puesto que no son positivos todos los elementos de la primera columna de su tabla

s^3	1	26.4839	0
s^2	16.8837	274.7454	0
s	10.2111	0	
1	274.7454		

Tabla 3.1: Tabla de Routh para el denominador de la planta en BA.

de Routh (tabla 3.2). Indica asimismo que habrá dos polos con partes reales positivas puesto que hay dos cambios de signo en los elementos de dicha primera columna.

$$D_{BC} = s^3 + 18{,}1110s^2 - 47{,}8290s + 1410{,}8708 \tag{3.5}$$

s^3	1	-47.8290	0
s^2	18.1110	1410.8708	0
s	-125.7303	0	
1	1410.8708		

Tabla 3.2: Tabla de Routh para el denominador de la planta en BC.

Efectivamente, al calcular la función de transferencia del sistema en BC, se obtiene:

$$G_{BC} = \frac{1{,}2273s^2 - 74{,}3129s + 1136{,}1254}{s^3 + 18{,}1110s^2 - 47{,}8290s + 1410{,}8708} \tag{3.6}$$

cuyos polos y ceros son:

- Ceros: $c_{1,2} = 30{,}2750 \pm 3{,}0230i$.
- Polos:$p_1 = -22{,}8925; p_{2,3} = 2{,}3907 \pm 7{,}4776i$.
- Ganancia estática:$K_{BC} = 1{,}2273$

Como puede verse en la figura 3.4, la planta en BC tiene 2 polos inestables, es decir, en el SPD.

Respuesta en el tiempo.

Se comprueban ahora las respuestas del sistema sin compensar frente a diversas entradas (escalón, rampa y ruido aleatorio), tanto en bucle abierto como en bucle cerrado realimentado unitariamente.

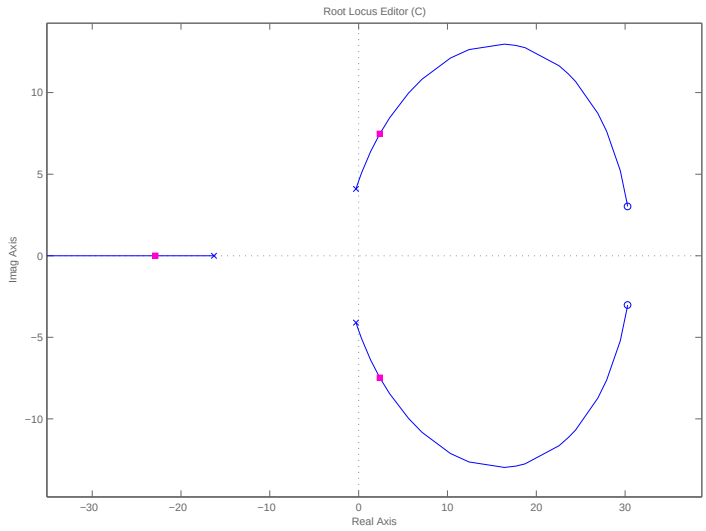
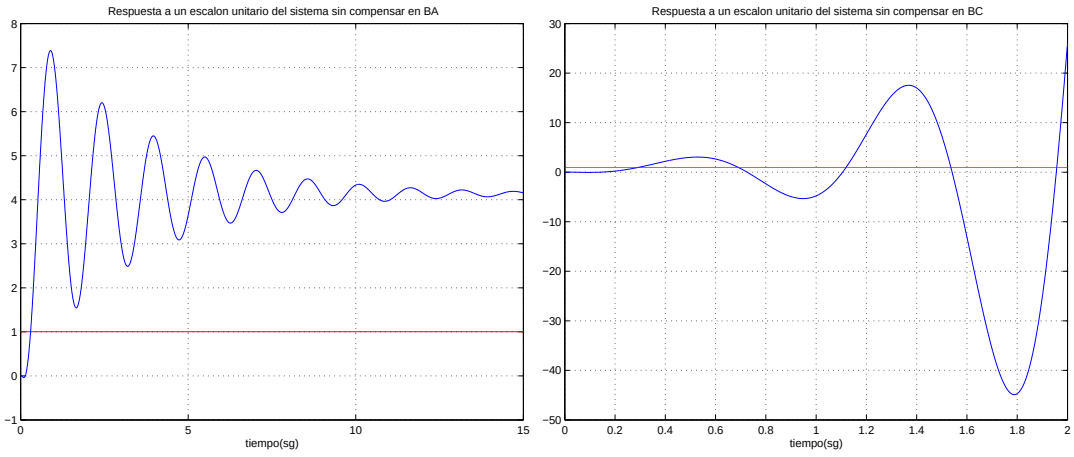
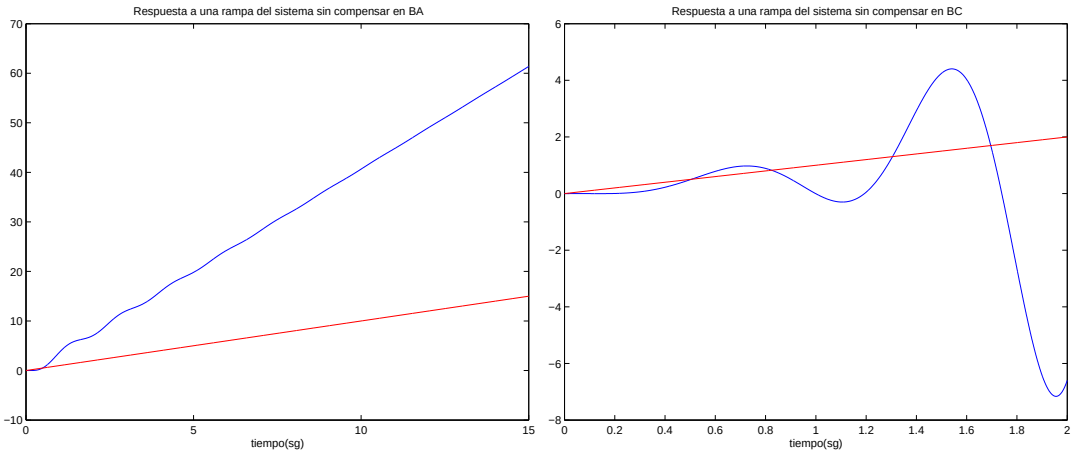


Figura 3.4: Lugar de las raíces del sistema en BC en el punto de operación nominal.



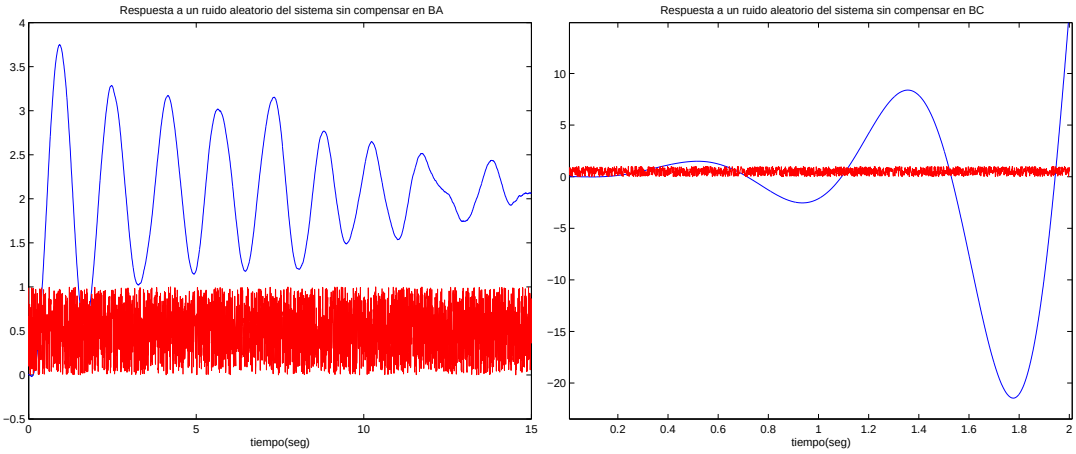
(a) En bucle abierto (b) En bucle cerrado

Figura 3.5: Respuesta a escalón del sistema sin compensar.



(a) En bucle abierto (b) En bucle cerrado

Figura 3.6: Respuesta a rampa del sistema sin compensar.



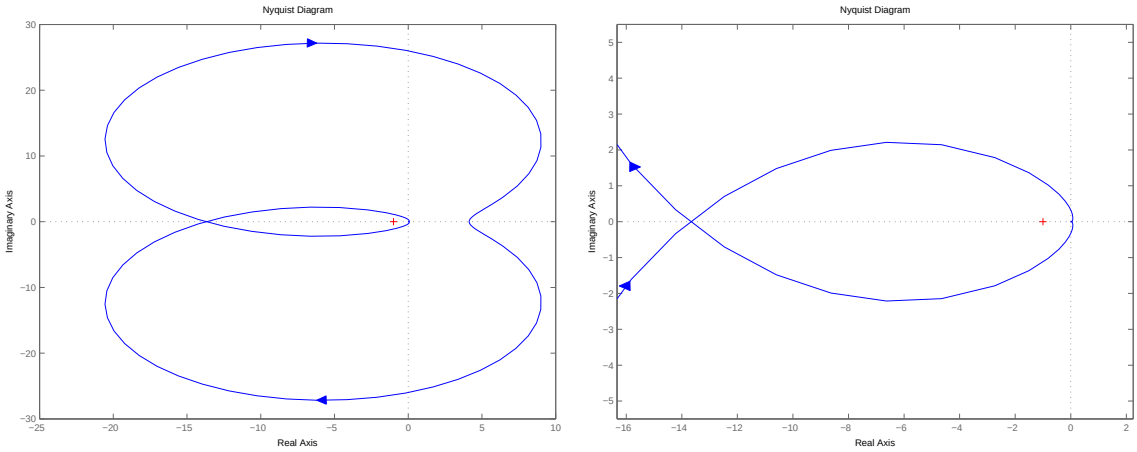
(a) En bucle abierto (b) En bucle cerrado

Figura 3.7: Respuesta a ruido aleatorio del sistema sin compensar.

Respuesta en frecuencia.

El término *respuesta en frecuencia* se define como la respuesta en estado estacionario de un sistema a una entrada senoidal, es decir, la variación entre la magnitud y la fase de la salida en estado estacionario respecto de la entrada. Este análisis se realiza sobre un intervalo de frecuencias. Existen varias técnicas para analizar los datos de la respuesta en frecuencia; a continuación se estudiarán dos de ellas: la de Bode y la de Nyquist.

El *diagrama de Nyquist* es una traza polar de la respuesta en frecuencia del sistema. En 3.8 está dibujado el diagrama de Nyquist del sistema sin compensar.



(a) Vista completa. (b) Detalle en torno al punto $(-1, 0)$

Figura 3.8: Diagrama de Nyquist del sistema sin compensar.

Nyquist enunció además un criterio de estabilidad que reza: "Un sistema en bucle cerrado (BC) será estable si y sólo si, al aplicar la imagen de G(s) en bucle abierto sobre el contorno de Nyquist, el número de veces que esa imagen rodea al punto -1 en sentido antihorario coincide con el número de polos inestables del G(s) de BA". Como se puede observar en la figura 3.8a, la traza rodea una vez al punto -1, mientras que el número de polos inestables del sistema en BA es nulo, por tanto, el sistema no será estable en BC. En la figura 3.8b se tiene un detalle en torno al punto -1 del eje real.

El *margen de ganancia* se define como el factor por el que se puede incrementar la magnitud o ganancia del sistema antes de llegar a hacerlo inestable. Por otro lado, el *margen de fase* se define como el ángulo que se debe añadir a la fase del sistema en bucle abierto para alcanzar 180º cuando su magnitud vale la unidad (0dB). Sus

expresiones matemáticas son

$$M_g|_{dB} = -|G(\omega_{cf})|dB \text{ con } \angle G(\omega_{cf}) = -180^o \tag{3.7}$$

$$M_f = \pi - \angle K \cdot G(\omega_{cg}) \text{ con } |G(\omega_{cg})| = 1 \tag{3.8}$$

Por último, se dibujó el diagrama de Bode del sistema sin compensar (figura 3.9), donde también se aprecia la inestabilidad del sistema al cerrar el lazo con ganancia unitaria, ya que el margen de ganancia es menor que uno (-22.7dB a $\omega = 4{,}6 rad/s$) y el margen de fase es negativo (-57.7º a $\omega = 9{,}1 rad/s$). Se podría conseguir estabilizar el sistema en BC reduciendo mucho la ganancia pero es más conveniente diseñar estrategias de control más complejas.

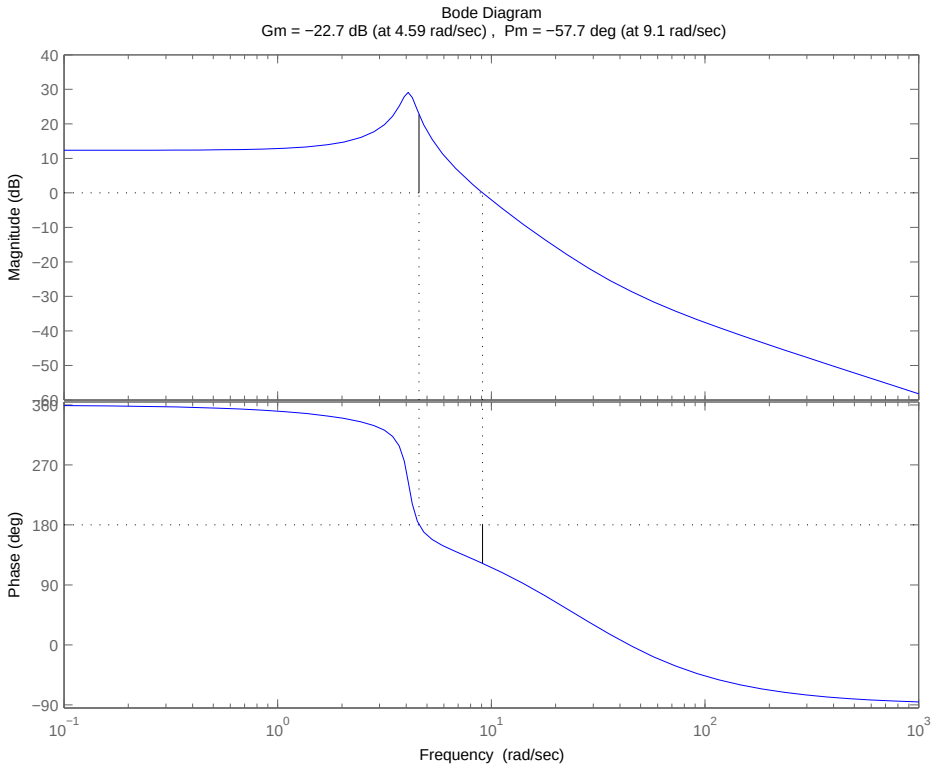


Figura 3.9: Diagrama de Bode del sistema sin compensar.

Capítulo 4

Control clásico.

A partir de ahora se empezarán a diseñar estrategias de control para el Sistema Levineu, es decir, se seleccionaran formas apropiadas del controlador en una configuración de control en bucle cerrado y se determinarán los parámetros idóneos para dicho controlador. En este capítulo se comentarán los esquemas de control clásicos.

4.1. Teoría del control clásico.

El controlador es un elemento en el sistema en bucle cerrado que tiene como entrada la señal de error y produce una salida que se convierte en entrada del actuador o elemento correctivo (ver sección 1.1.1). La relación entre la salida y la entrada al controlador se suele denominar *ley de control*. Clásicamente existen tres formas básicas para dicha ley: *proporcional, integral y derivativo*. En algunas ocasiones es necesario mejorar el desempeño del controlador, lo cual se logra al introducir en el sistema de control elementos adicionales denominados *compensadores*, que producen una alteración en el control llamada *compensación*. En la figura 4.1 se muestra el control en BC con funciones de transferencia en el dominio de Laplace: $C(s)$ es el modelo del controlador, $G(s)$ representa a la planta del sistema, $H(s)$ es el sensor; las señales han sufrido la transformación de Laplace: la referencia $R(s)$, el error $E(s)$, la señal de control $U(s)$ y la salida $Y(s)$.

4.1.1. Leyes de control clásicas.

Las tres formas antes mencionadas para la ley de control pueden aplicarse por separado o conjuntamente, produciendo distintos tipos de controladores. A continuación se expondrá un resumen de cada uno de ellos.

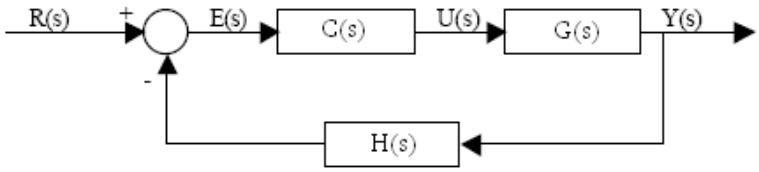


Figura 4.1: Esquema de BC en el dominio de Laplace.

Control proporcional.

Con el control proporcional (control P) la salida del controlador es directamente proporcional a su entrada, que es el error. La función de transferencia para el controlador P es, por tanto

$$u(t) = K_p e(t) \Rightarrow C_P(s) = K_p \tag{4.1}$$

donde el parámetro K_p es una constante llamada *ganancia proporcional*. Esta ganancia constante, sin embargo, tiende a existir sólo sobre cierto rango de errores que se conoce como *banda proporcional* (BP o PB - de 'Proportional Band').

También se suele expresar la salida del controlador P como un porcentaje de la posible salida total. De este modo, un 100 % de cambio en la salida del controlador corresponde a un cambio en el error desde un extremo a otro de la banda proporciona. Así

$$K_p = \frac{100}{BP} \tag{4.2}$$

Control integral.

Con el control integral (control I) la salida del controlador es proporcional a la integral de la señal de error con el tiempo. La función de transferencia para el control I es, por tanto

$$u(t) = K_i \int_0^t e(t)dt \ \Rightarrow C_I(s) = \frac{K_i}{s} \tag{4.3}$$

donde K_i es una constante llamada *ganancia integral* (con unidades de s^{-1}). El control integral introduce un polo en el origen por lo que aumenta el tipo del sistema y se reduce el error en estado estacionario; por contra, este polo añadido hace disminuir la estabilidad relativa del sistema.

Control derivativo.

Con el control derivativo (control D), la salida del controlador es proporcional a la razón de cambio con el tiempo del error. La función de transferencia para el control D es, por tanto

$$u(t) = K_d \frac{de(t)}{dt} \Rightarrow C_D(s) = K_d s \tag{4.4}$$

donde K_d es la *ganancia derivativa* (con unidades de s). El control derivativo introduce un cero y se logra aumentar la rapidez de la respuesta, pero no reduce el error estacionario. No obstante, en la práctica se emplea conjuntamente con otro controlador o se aproxima por un compensador de adelanto (también llamado red de avance).

Control proporcional integral.

La reducción en la estabilidad relativa que se mencionó al explicar el control integral se puede resolver combinándolo con un control proporcional. De esta forma, el control proporcional integral (PI) queda

$$\begin{aligned} u(t) &= K_p e(t) + K_i \int_0^t e(t)dt \Rightarrow \\ C_{PI}(s) &= K_p + \frac{K_i}{s} = K_p \left(1 + \frac{1}{T_i s}\right) = K_p \left(\frac{s + 1/T_i}{s}\right) \end{aligned} \tag{4.5}$$

donde T_i se llama *constante de tiempo integral* y vale $T_i = K_p/K_i$. El control PI añade un polo en el origen y un cero en $s = -(1/T_i)$ por lo que disminuye el error estacionario pero la reducción de estabilidad relativa no es tan fuerte. Además, este control constituye un sistema bipropio e realizable.

Control proporcional derivativo.

El control proporcional derivativo (control PD) tiene la siguiente función de transferencia

$$\begin{aligned} u(t) &= K_p e + K_d \frac{de(t)}{dt} \Rightarrow \\ C_{PD}(s) &= Kp + K_d s = K_p(1 + T_d s) \end{aligned} \tag{4.6}$$

donde T_d se llama *constante de tiempo derivativo* y vale $T_d = K_d/K_p$. El control PD añade un cero en $S = -1/T_d$. No hay cambios en el tipo de sistema y, por tanto,

tampoco en el error estacionario. Para ser realizable necesita un filtro paso baja como sensor de realimentación, porque por sí solo es un sistema impropio. En la práctica se realiza un control PD con la forma $C_{PDreal}(s) = K_p + \frac{K_p s}{s+K_p/K_d}$

Control proporcional integral derivativo.

El controlador proporcional integral derivativo, también llamado *controlador de tres términos*, tiene como función de transferencia la siguiente

$$\begin{aligned} u(t) &= K_p e + K_i \int_0^t e dt + K_d \frac{de(t)}{dt} \Rightarrow \\ C_{PID}(s) &= K_p + \frac{K_i}{s} + K_d s = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \rightarrow \\ C_{PID}(s) &= \frac{K_p(1 + T_i s + T_i T_d s^2)}{T_i s} \end{aligned} \tag{4.7}$$

El control PID ha incrementado el número de ceros en 2 y el número de polos en 1. El factor $1/s$ incrementa el tipo de sistema en 1, por lo que se reduce el error estacionario, pero gracias a los ceros se puede mantener la estabilidad relativa.

El control PID engloba a todos los demás, puesto que poniendo ciertos valores a sus parámetros se pueden conseguir las anteriores leyes de control. Por ejemplo, si hacemos cero T_i o T_d, obtenemos un control PD o PI, respectivamente. Asimismo, si hacemos $K_p = 0$, anulamos la acción proporcional. Por esta razón sólo se diseñarán controles PID como estrategia de control clásica.

4.1.2. Ajuste de los parámetros de los controladores clásicos.

Para conseguir una primera aproximación de los parámetros de los controladores clásicos existen varias técnicas: reglas de Ziegler-Nichols, método analítico, cancelación polo-cero, lugar de las raíces, entre otras muchas. Aquí se presentará la teoría de las dos primeras. Posteriormente el diseñador debe sintonizar los parámetros con mayor precisión, dependiendo del comportamiento requerido.

Reglas de Ziegler-Nichols.

Los *métodos de ajuste de Ziegler-Nichols* constituyen un mecanismo heurístico que permite obtener una primera aproximación para los parámetros del controlador.

El primer método se suele llamar *método de la curva de reacción del proceso*. El procedimiento consiste en abrir el bucle de control (normalmente entre el controlador y el actuador) y obtener la respuesta y del sistema ante una señal de prueba de pequeña amplitud u (normalmente un escalón). Se mide entonces la ganancia estática K, la constante de tiempo τ y el tiempo muerto o atraso τ_d. La pendiente máxima de la curva de salida será $R = K/\tau$. La tabla 4.1 proporciona los criterios recomendados por Ziegler y Nichols para los valores del controlador basándose en los valores de K,τ y τ_d. Este método sólo es válido para sistemas cuyo comportamiento en BA es estable, por lo que efectivamente puede aplicarse al sistema en cuestión.

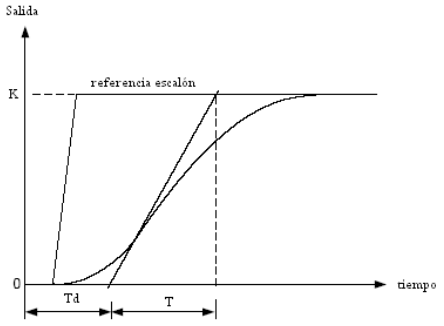


Figura 4.2: Parámetros de la curva de reacción del proceso.

$$K = \frac{y_\infty - y_0}{u_\infty - u_0} \tag{4.8}$$

Modo de control	K_p	T_i	T_d
P	$\frac{\tau}{K\tau_d}$	∞	0
PI	$\frac{0{,}9\tau}{K\tau_d}$	$\frac{\tau_d}{0{,}3}$	0
PID	$\frac{1{,}2\tau}{K\tau_d}$	$2\tau_d$	$0{,}5\tau_d$

Tabla 4.1: Reglas de Ziegler-Nichols a partir de la curva de reacción del proceso.

El segundo método se conoce como el *método de la última ganancia* y se aplica al sistema en bucle cerrado. Teniendo al mínimo las acciones integral y derivativa, se averigua la ganancia crítica K_{crit} con la cual el sistema empieza a oscilar (los polos alcanzan el eje imaginario). Con esta ganancia el sistema adquiere una oscilación permanente con periodo T_{crit} que puede calcularse a través de la frecuencia natural de estos polos puramente imaginarios (ω_{crit} = módulo de los polos). La banda proporcional crítica es $100/K_{crit}$. La tabla 4.2 muestra los criterios de Ziegler-Nichols para establecer los valores de cada controlador basándose en este método.

Modo de control	K_p	T_i	T_d
P	$0{,}5K_{crit}$	∞	0
PI	$0{,}45K_{crit}$	$\frac{T_{crit}}{0{,}2}$	0
PID	$0{,}6K_{crit}$	$0{,}5T_{crit}$	$0{,}125T_{crit}$

Tabla 4.2: Reglas de Ziegler-Nichols a partir de la última ganancia.

Método analítico.

Otra técnica para sintonizar un controlador es el *método analítico en el dominio de la frecuencia*. Se diseñarán los controladores con ayuda de un diagrama de Bode en el que se representará $K_p \cdot G(s)$, es decir, se añadirá previamente una ganancia necesaria para cumplir la exigencia de error en régimen permanente. Posteriormente se colocarán ceros y polos donde convengan para cumplir las especificaciones.

Las especificaciones de control de las que se habló en 1.1.1 se pueden traducir a lugares geométricos en el lugar de las raíces. Así, la condición de sobreoscilación impone una cota inferior a la constante de amortiguamiento δ; esta constante de amortiguamiento impone una cota superior al ángulo α que los polos forman con el semieje real negativo. Por su parte, la condición de tiempo de subida dispone que una cota inferior para la frecuencia natural del sistema ω_n, por lo que este valor será la distancia mínima desde los polos permitidos al origen de coordenadas del plano complejo. A continuación se escriben algunas relaciones interesantes entre estos parámetros del lugar de las raíces:

$$SO = M_p = e^{\frac{\delta\pi}{\sqrt{1-\delta^2}}} \Rightarrow \delta = \sqrt{\frac{\ln^2(SO)}{\ln^2(SO) + \pi^2}} \tag{4.9}$$

$$\alpha = a\cos(\delta) \tag{4.10}$$

$$t_r = \frac{\pi - a\cos(\delta)}{\omega_n\sqrt{1-\delta^2}} \tag{4.11}$$

Para diseñar un controlador PD según el método analítico, se debe colocar un cero a una frecuencia más o menos media década por debajo de la frecuencia de corte ω_c. Esta frecuencia de corte se extrae del Bode del sistema con una ganancia aplicada. Según el método analítico, esta ganancia debe ser añadida previamente a la colocación de ceros y polos, para tener resuelta de antemano la condición de error en régimen permanente. Con el cero que se acaba de incluir sube el valor de ω_c, con lo que se mejora la rapidez del sistema y, además, aumenta el margen de fase (pues se suma la fase del cero), por lo que disminuye la sobreoscilación. La situación del cero es crítica ya que, si se colocara a frecuencias altas, no se aprovecharía la fase que introduce el

cero; en cambio, si se colocara a frecuencias bajas, la ω_c subiría demasiado por lo que el margen de fase podría acabar disminuyendo, ya que el diagrama de Bode de la fase del sistema es descendente a medida que aumenta ω.

Para diseñar un controlador PI según este método, se coloca un polo en el origen, con lo que sube el tipo del sistema en una unidad y se reduce el error en régimen permanente, y, además, se pone un cero más o menos década y media por debajo de la frecuencia de corte.

Por último, para diseñar un controlador PID, se mezclan las instrucciones seguidas para PD y PI, de modo que se pone uno de los ceros media década por debajo de la frecuencia de corte, el otro más o menos década y media por debajo de ω_c.

4.2. Diseño para el Sistema Levineu.

4.2.1. Síntesis del controlador.

El control PID fue programado íntegramente en el autómata mediante el CX-Programmer. Se realizaron dos versiones de este control. La primera versión aprovechaba la instrucción **PID** de la librería del autómata y se calcula en una tarea del programa desarrollado mediante diagrama de escalera (ver sección 8.2). En la segunda versión se programa el PID en un bloque de función del autómata mediante texto estructurado.

Control PID con instrucción de Omron.

La tarea que lleva a cabo el control PID aprovechando la instrucción PID del autómata (instrucción número 190) se denomina *PIDInstr* y se ejecuta si, en el panel de control de LabView, se ha elegido dicho control. Antes de llamar a la instrucción PID, deben ser almacenados sus operandos (ver [Omr04c]).

- Altura: en milímetros.
- Referencia: en milímetros.
- Banda proporcional: calculada como $100/K_p$ y en unidades de 0.1 %.
- Tiempo derivativo: en décimas de segundo.
- Tiempo integral: en décimas de segundo.

- Tiempo de muestreo: en centésimas de segundo.
- Especificador de operación: acción proporcional en dirección inversa; cambios de configuración permitidos en cada periodo de muestreo; salida del 50 % con error nulo; coeficiente del filtro de entrada: $\alpha = 0{,}65$.
- Rango de valores de entrada hasta $03FFhex = 1023dec$; rango de valores de salida hasta $0FFFhex = 4095dec$.
- Límite inferior de la variable manipulada: $0000hex = 0dec$.
- Limite superior de la variable manipulada: $0FA0 = 4000dec$

La salida de la instrucción PID es directamente la señal de control que debe aplicarse al actuador. El cálculo de esta salida se realiza con el control del valor referencia filtrado con dos grados de libertad, que se representa en el siguiente diagrama de bloques (figura 4.3):

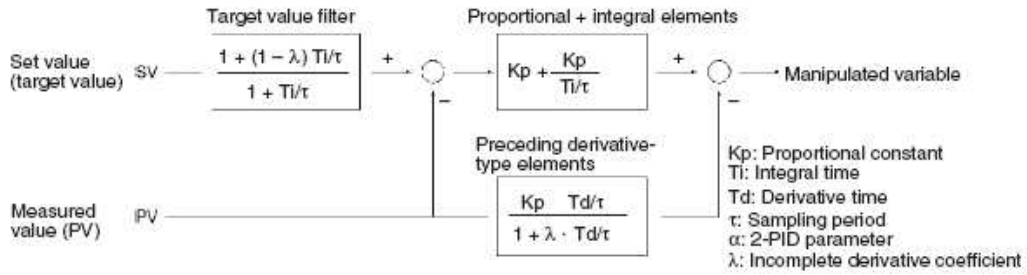


Figura 4.3: Diagrama de bloques para el control PID con filtrado de la referencia con dos grados de libertad.

El momento de la acción del PID está determinada con una combinación del tiempo de muestreo y el tiempo de ejecución de la instrucción, teniendo en cuenta además el tiempo de ciclo del programa. Si el tiempo de muestreo es menor que el tiempo de ciclo, el control PID se ejecuta cada ciclo. Si el tiempo de muestreo es mayor o igual que el tiempo de ciclo, el control PID no es ejecutado en cada ciclo, sino cuando el valor acumulativo de los tiempos entre dos instrucciones PID es mayor o igual al tiempo de muestreo. La porción sobrante del valor acumulativo (o sea, el valor acumulativo menos el tiempo de muestreo) es acarreado hasta el siguiente valor acumulativo.

La acción proporcional de la instrucción PID se muestra en la figura 4.4.

Los resultados obtenidos con este control sobre la planta real se muestran en la sección 4.2.2.

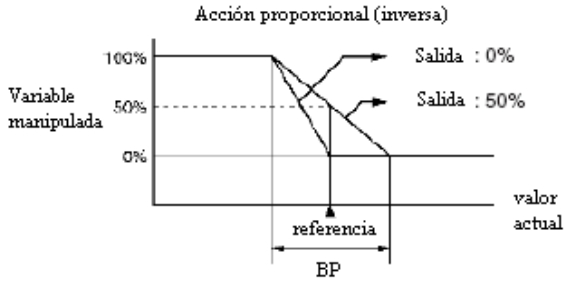


Figura 4.4: Acción proporcional de la instrucción PID del autómata.

Control PID con bloque de función.

La segunda versión para implementar un control PID en el Sistema Levineu consiste en programar un bloque de función en el autómata mediante un lenguaje estructurado propio de Omron (parecido a Pascal). La tarea dentro del programa que lleva a cabo este tipo de control se denomina *PIDBloque* y llama cada tiempo de muestreo al bloque de función *PID*, que calcula la señal de control con la fórmula siguiente:

$$u_k = u_{k-1} + e_k \cdot K_p \left(1 + \frac{T_m}{T_i} + \frac{T_d}{T_m}\right) - e_{k-1} \cdot K_p \left(1 + 2\frac{T_d}{T_m}\right) + e_{k-2} \cdot K_p \frac{T_d}{T_m} \quad (4.12)$$

Esta fórmula 4.12 es proviene de un PID analógico como la ecuación 4.7 discretizado mediante la aproximación Euler II (Euler hacia atrás).

En el Sistema Levineu existen limitaciones en la actuación, pues el variador de velocidad no puede superar los valores mínimo y máximo (0 y 10V, respectivamente) y, por tanto, el motor soplador se encuentra siempre en un rango de frecuencias limitado. Hay tres formas básicas para enfrentarse al problema de la saturación en el actuador:

1. No tener en cuenta las limitaciones en el diseño y truncar posteriormente la señal de control calculada si se excedieron aquéllas.
2. Realizar el diseño de modo que las limitaciones no se alcancen nunca (bastante conservador).
3. Realizar un diseño que tenga en cuenta las restricciones.

En este Proyecto se ha evitado la saturación del actuador truncando la señal de control excesiva, sin perjuicio de que hubiera un diseño que las evitara.

En el caso del control PID se esbozó un integrador 'anti-windup' para tener en cuenta las restricciones en el actuador. El antiwindup se tomó de [Ast90] para un

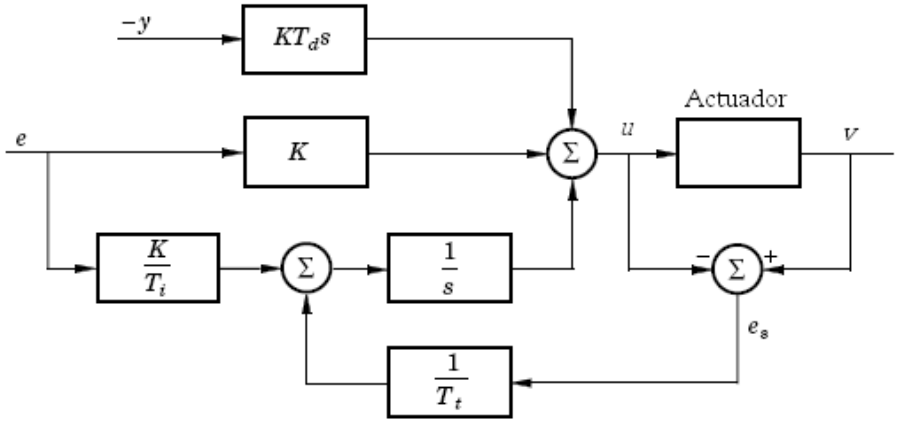


Figura 4.5: Controlador PID con antiwindup.

sistema cuyo señal de control puede ser medida (se considera que la salida instantánea que proporciona el variador es la salida real del actuador). En este sistema se cierra un nuevo lazo de realimentación formado por una señal de error entre la salida del actuador (v) y la salida del controlador (u); este error alimenta al integrador del PID a través de una ganancia $1/T_t$, donde T_t se denomina *tiempo de seguimiento*. El esquema se representa en la figura 4.5 y la fórmula del PID queda como en la ecuación 4.13. Si el actuador no está saturado este bloque añadido no hace nada; si el actuador está saturado el integrador antiwindup añadido intenta que esta señal de error se haga nula. De esta forma, cuando la referencia vuelve a ser alcanzable por el sistema sin saturación, este integrador provoca que la señal de control consiga una recuperación más rápida.

$$u_k = u_{k-1} + e_k \cdot K_p \left(1 + \frac{T_m}{T_i} + \frac{T_d}{T_m}\right) - e_{k-1} \cdot K_p \left(1 + 2\frac{T_d}{T_m}\right) + e_{k-2} \cdot K_p \frac{T_d}{T_m} + (v_{k-1} - u_{k-1}) \cdot \frac{T_m}{T_t} \tag{4.13}$$

4.2.2. Resultados.

En primer lugar se realizó una aproximación a los parámetros del PID mediante las reglas de Ziegler-Nichols. El sistema en BC sin compensar (con ganancia unitaria) no es estable, por lo que no se utilizó el método de la última ganancia de Ziegler-Nichols para calcular los parámetros PID. En cambio, dado que el sistema en BA sí es estable con ganancia unitaria, se empleó el método de la curva de reacción del proceso (se

aprovecha la figura 3.7a). Resultan los siguientes valores:

$$\begin{cases} K = 4{,}2 \\ \tau = 1 \\ \tau_d = 0{,}3 \end{cases} \Rightarrow \begin{cases} K_p = 0{,}95 \\ T_d = 0{,}15 \\ T_i = 0{,}6 \end{cases} \tag{4.14}$$

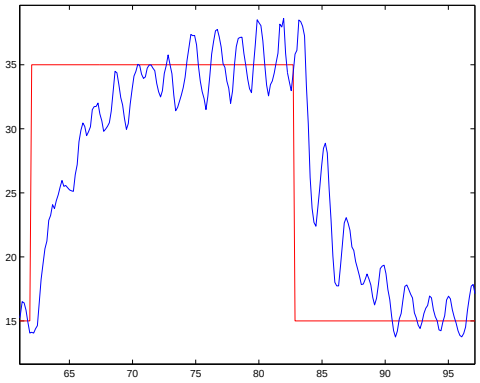
En este apartado se explican los experimentos realizados sobre la planta real. Con el objetivo de poder comparar los resultados, se llevaron a cabo experimentos con los mismos parámetros en las dos versiones. A continuación se detalla la configuración en cada caso y se muestran algunas de las gráficas resultado de los experimentos:

1. Caso 1: $K_p = 0{,}01$, $T_d = 0{,}02$, $T_i = 0{,}20$.
2. Caso 2: $K_p = 0{,}02$, $T_d = 0{,}02$, $T_i = 0{,}15$.
3. Caso 3: $K_p = 0{,}05$, $T_d = 0{,}02$, $T_i = 0{,}20$.
4. Caso 4: $K_p = 0{,}05$, $T_d = 0{,}02$, $T_i = 0{,}70$.
5. Caso 5: $K_p = 0{,}02$, $T_d = 0{,}10$, $T_i = 0{,}20$.
6. Caso 6: $K_p = 0{,}025$, $T_d = 0{,}02$, $T_i = 0{,}30$.

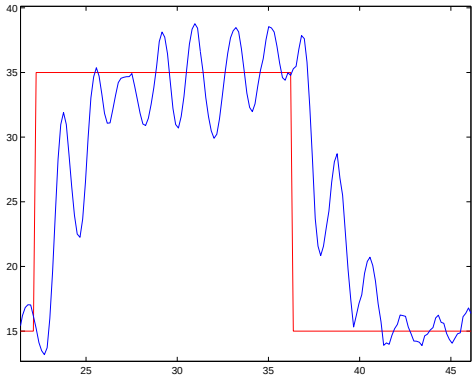
Se observa que al aumentar la ganancia proporcional K_p (o sea, reducir el ancho de la banda proporcional BP), el tiempo de subida es menor, pero se incrementa la sobreoscilación. En el caso 3 se llega a una situación de inestabilidad por haber subido demasiado K_p. Por contra, el sistema se hace más lento para una K_p menor.

Si se aumenta el tiempo derivativo T_d (caso 5), la respuesta del sistema se hace mucho más rápida y se reduce la sobreoscilación cuando hay un cambio de referencia. Sin embargo, perjudica en sistemas de fase no mínima como éste y amplifica demasiado el ruido, lo que se ve en la figura 3.7e cuando se deja la refencia inmóvil. Por este mismo motivo perjudica la recuperación frente a perturbaciones.

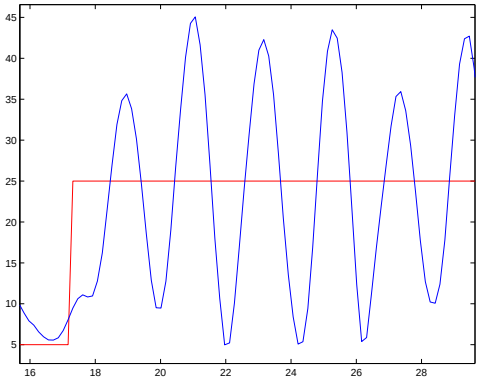
Si se aumenta el tiempo integral T_i el sistema se ralentiza, lo que contrarresta un poco el efecto de una K_p o un T_d excesivos.



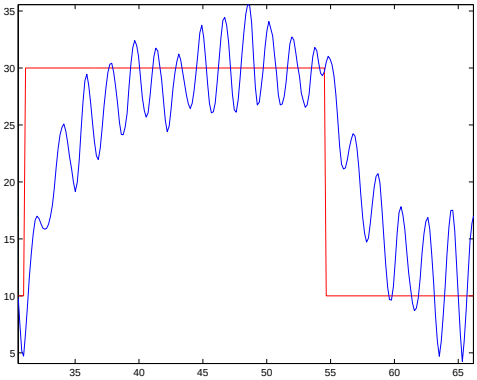
(a) Caso 1.



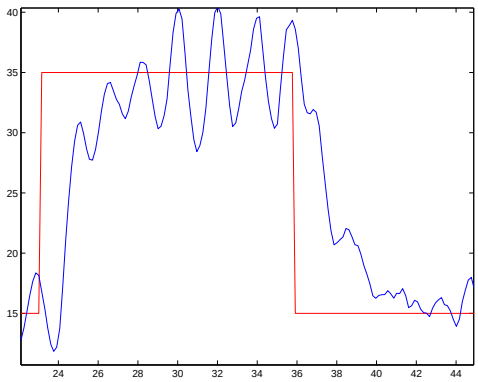
(b) Caso 2.



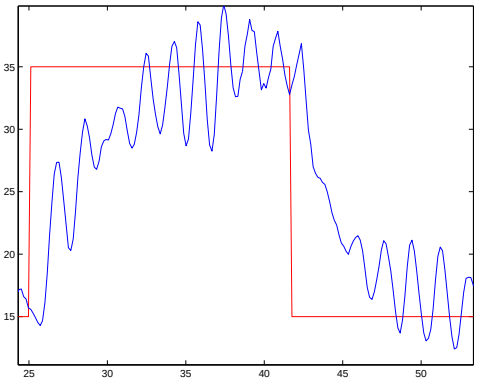
(c) Caso 3.



(d) Caso 4.

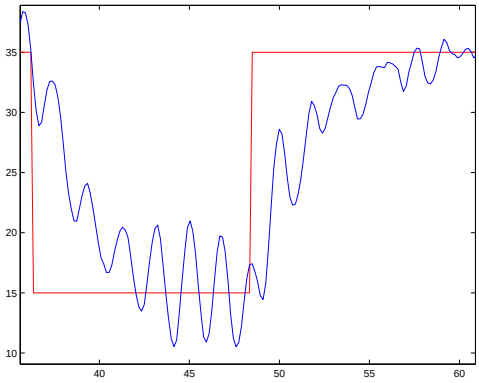


(e) Caso 5.

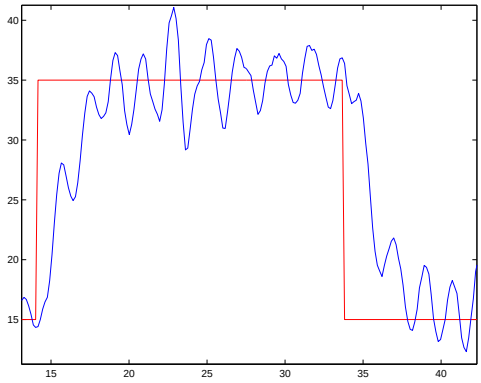


(f) Caso 6.

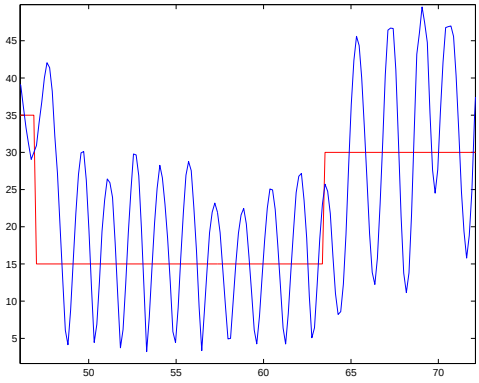
Figura 4.6: Respuesta a escalones de los controles PID con instrucción de librería.



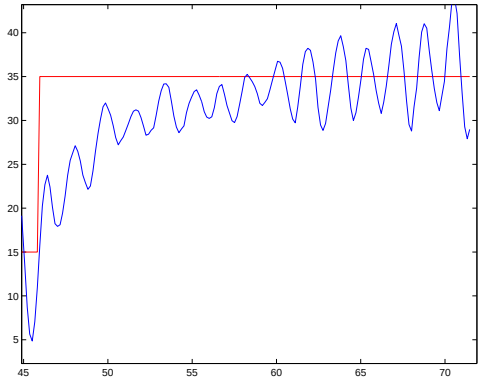
(a) Caso 1.



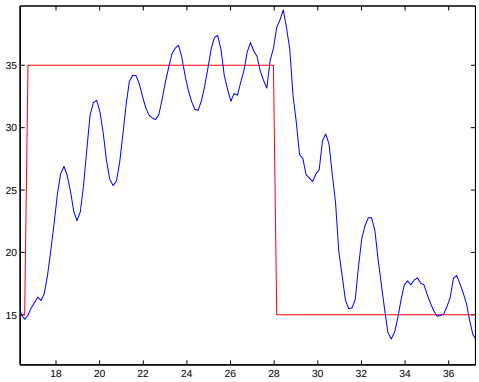
(b) Caso 2.



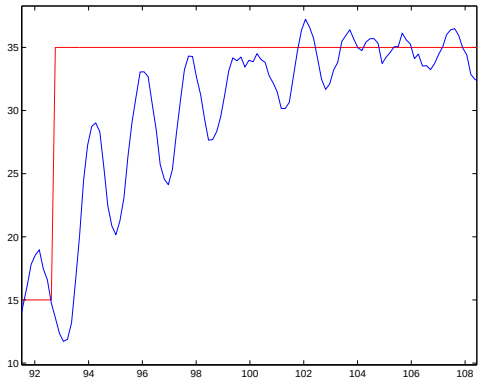
(c) Caso 3.



(d) Caso 4.

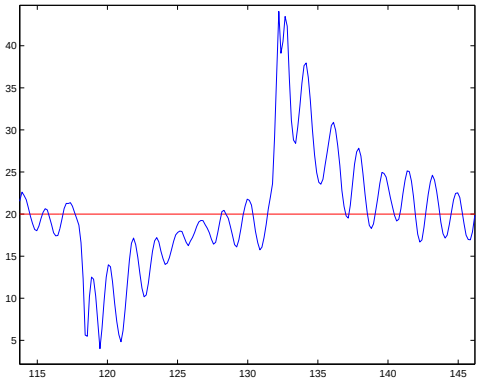


(e) Caso 5.

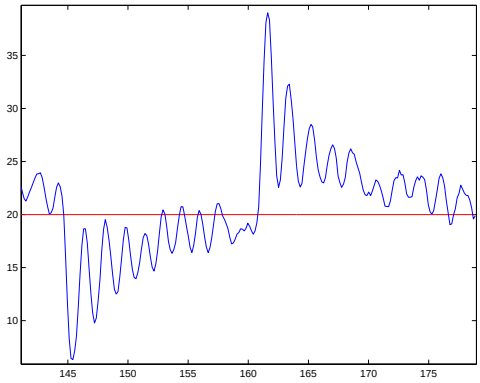


(f) Caso 6.

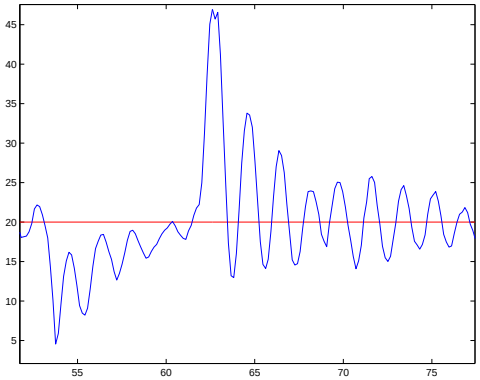
Figura 4.7: Respuesta a escalones de los controles PID con bloque de función.



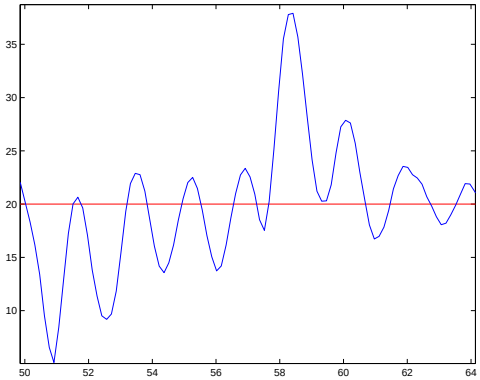
(a) Caso 1 con instrucción.



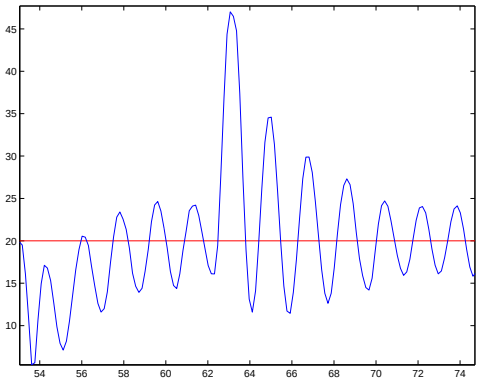
(b) Caso 1 con bloque de función.



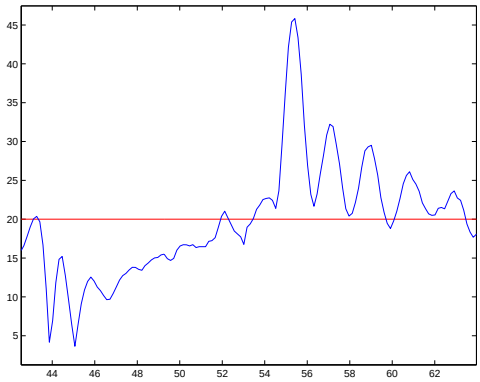
(c) Caso 2 con instrucción.



(d) Caso 2 con bloque de función.



(e) Caso 5 con instrucción.



(f) Caso 5 con bloque de función.

Figura 4.8: Respuesta ante perturbaciones de los controles PID.

Capítulo 5

Control H_∞.

5.1. Teoría del control H_∞.

El estudio de la teoría del control H_∞ es relativamente reciente. En la última década del siglo XX ha ido aumentando la atención hacia este método de control debido a sus características de robustez. Esta teoría está basada en hipótesis más realistas respecto a las restricciones impuestas sobre las señales que aparecen en el esquema de control: no se imponen distribuciones estadísticas específicas, sino que sólo se supone que la energía de dichas señales está limitada.

La formulación del problema de control que se utiliza aquí fue introducida por Doyle ([Doy83][Doy84])y se basa en el esquema representado en la figura 5.1.

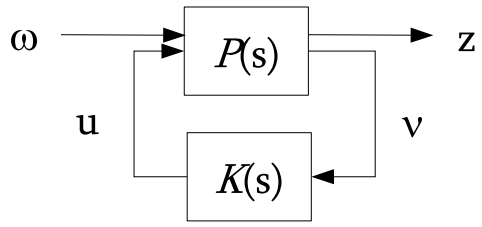


Figura 5.1: Formulación general del problema de control.

En ella se propone una planta generalizada *P(s)* definida como:

$$\begin{bmatrix} z \\ t \end{bmatrix} = P(s) \begin{bmatrix} w \\ u \end{bmatrix} \tag{5.1}$$

donde ω representa un vector de perturbaciones externas a la planta (referencias,

perturbaciones, ruidos, etc.), z representa el vector de señales de error, cuya magnitud (calculada utilizando alguna norma) será indicadora del comportamiento del sistema, v es el vector de señales medibles que alimentará a algún controlador y u será el vector de señales de control que generará un controlador a partir de las señales v.

Por otra parte, sea el controlador *K(s)* que genere señales de control u para la planta generalizada *P(s)* a partir de señales medibles v, esto es,

$$u = K(s)v \tag{5.2}$$

Según esta configuración, la función de transferencia desde las señales perturbadoras externas ω hasta las señales de error z cerrando el bucle con el controlador *K(s)* viene dada por la transformación lineal

$$z = T_{z\omega}\omega = F_l(P(s), K(s))\omega \tag{5.3}$$

El problema de control con esta configuración consiste en calcular un controlador de forma que se atenúe la relación entre una medida de la magnitud del vector de errores, z, frente a una medida de la magnitud del vector de señales perturbadoras, ω. Al nivel de atenuación conseguido se le denominará γ.

Para hallar la magnitud de un vector se ha de utilizar algún índice que proporcione una medida escalar a partir de las componentes del vector. En el control H_∞ el índice que indica la magnitud de los vectores es la *energía* de las señales que componen el vector, y se utiliza el mismo índice tanto para las señales de entrada como para las de salida.

El problema de control óptimo no ha sido resuelto todavía pero existe una solución para el problema subóptimo. En la práctica se halla un controlador H_∞ subóptimo para un nivel de atenuación γ predeterminado y se itera sucesivamente con dicho nivel de atenuación ([Doy89]) hasta un valor suficientemente cercano al óptimo. Este método de síntesis resulta mucho más simple teórica y computacionalmente y es precisamente el que se utiliza para calcular el controlador H_∞ en el paquete de software Matlab.

Para construir la planta generalizada se introducen funciones de ponderación dependiente de la frecuencia que modifican el módulo de algunas señales o de algunas funciones de transferencia. Este segundo enfoque es conocido como *diseño por moldeo de funciones de transferencia*. En este enfoque se pueden moldear bien la función de lazo o bien las funciones de transferencia en bucle cerrado. Aquí se empleará esta última técnica.

Considérese el diagrama de bloques de la figura 5.2 donde ser refleja un sistema general de control por realimentación. En él aparecen señales externas e internas y funciones que las ponderan.

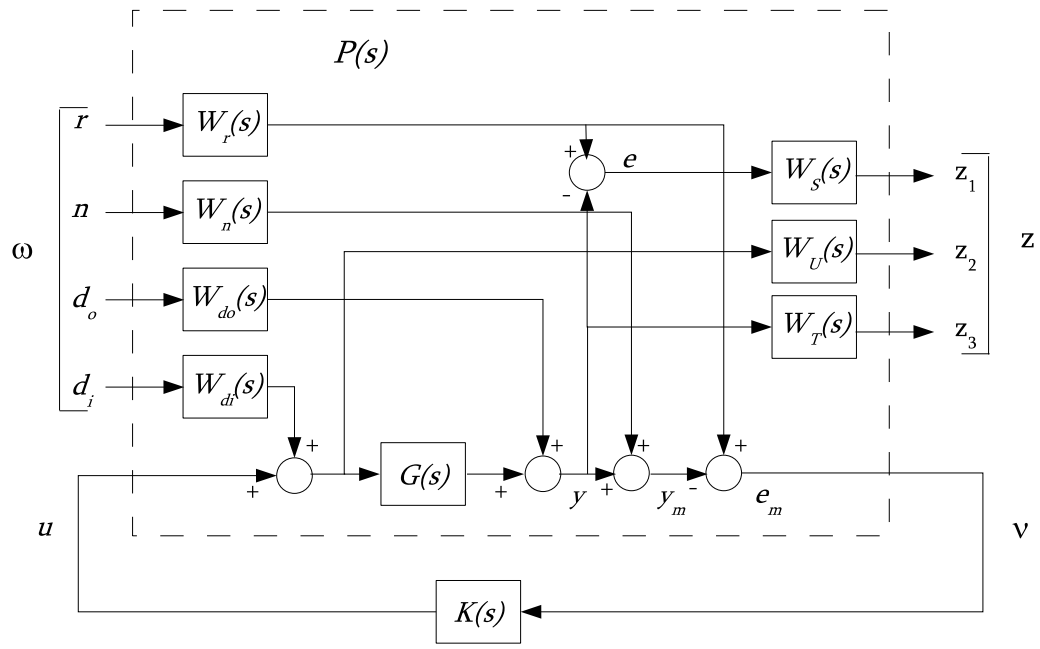


Figura 5.2: Estructura general para problemas de control H_∞ del problema de control.

Si se hacen los términos $W_r(s), W_{do}(s), W_{di}(s)$ y $W_n(s)$ igual a la identidad, entonces se ponderan las distintas funciones de sensibilidad en bucle cerrado mediante una sola función de ponderación en cada término. Al hacer pequeña la norma infinito de $T_{z\omega}(s)$ lo que se consigue es ponderar de forma conjunta las distintas funciones de sensibilidad en bucle cerrado, por lo que este planteamiento recibe el nombre de *enfoque de sensibilidad mixta*. Dependiendo de qué variables consideremos dentro del esquema general, se pueden distinguir casos simples ampliamente conocidos. En este trabajo se empleará la configuración de sensibilidad mixta $S/KS/T$. Estas funciones son la función de sensibilidad a la salida S_o, la función de sensibilidad complementaria a la salida T_o y la función de sensibilidad al control KS_o, con:

$$S_o(s) = \frac{E(s)}{R(s)} = \frac{1}{1 + G(s)K(s)} \tag{5.4}$$

$$T_o(s) = \frac{Y(s)}{R(s)} = \frac{G(s)K(s)}{1 + G(s)K(s)} \tag{5.5}$$

$$K(s)S_o(s) = \frac{U(s)}{R(s)} = \frac{C(s)}{1 + G(s)K(s)} \tag{5.6}$$

En esta configuración se consideran todas las salidas del caso general en función de la referencia (ver figura 5.3), ponderándose simultáneamente las funciones de sensibilidad *S(s)*, sensibilidad al control *K(s)S(s)* y sensibilidad complementaria *T(s)* mediante

las funciones de ponderación $W_S(s), W_U(s)$ (también llamada $W_{KS}(s)$) y $W_T(s)$, respectivamente.

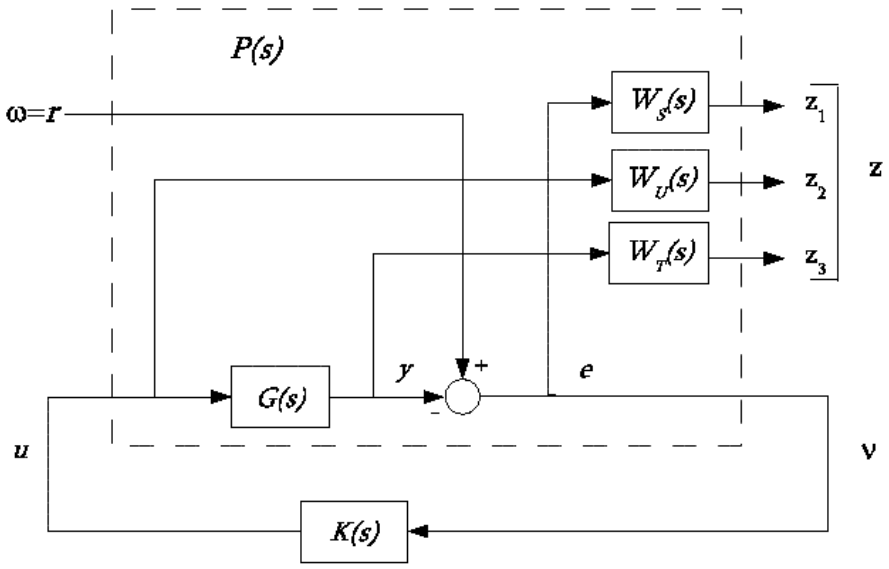


Figura 5.3: Configuración de sensibilidad mixta $S/KS/T$.

La expresión a minimizar en este caso queda como sigue:

$$\|T_{z\omega}(s)\|_\infty = \left\| \begin{array}{c} W_S(s)S_o(s) \\ W_U(s)K(s)S_o(s) \\ W_T(s)T_o(s) \end{array} \right\|_\infty \tag{5.7}$$

El procedimiento para calcular un controlador en esta caso será el siguiente:

1. Elegir la configuración para crear la planta aumentada.
2. Diseñar adecuadamente las funciones de ponderación que intervienen en la planta aumentada.
3. Crear la planta aumentada a partir de las funciones de ponderación y del conocimiento de la planta.
4. Calcular el controlador *K(s)* mediante el algoritmo de síntesis expuesto en [Doy89].

Este procedimiento es sistemático salvo en el diseño de las funciones de ponderación. La elección de estas funciones no es trivial e influye poderosamente en la bondad del

controlador. En este Proyecto se sigue el método expuesto en [Ort01] para simplificar la sintonización de un controlador aceptable.

La síntesis de un controlador robusto está basada en el conocimiento de la incertidumbre del sistema. Por tanto, se toma un modelo nominal de bajo orden y sin retardos, con el objetivo de simplificar el cálculo. Una vez elegido el modelo nominal, se estima la incertidumbre multiplicativa no estructurada a la salida del sistema como

$$E_{oi}(s) = \frac{G_i(s) - G(s)}{G(s)} \tag{5.8}$$

donde G_i representa las funciones de transferencia del sistema en distintos puntos de trabajo. Por definición la incertidumbre multiplicativa indica el porcentaje de desconocimiento que se tiene de la planta en cada frecuencia. Este porcentaje suele aumentar con la frecuencia y siempre habrá una frecuencia a partir de la cual el valor de la incertidumbre multiplicativa supere la unidad, es decir, una frecuencia a partir de la cual el desconocimiento del sistema es total.

Como el tipo de incertidumbre utilizado es la multiplicativa a la salida, la condición de estabilidad robusta asociada a ella puede expresarse como ([Ort01]):

$$\|W_T(s)T_o(s)\|_\infty \equiv \sup_\omega \bar{\sigma}(W_T(j\omega)T_o(j\omega)) \leq 1 \tag{5.9}$$

Se propone escoger una función de transferencia $W_T(s)$ que sea estable, de fase mínima y de forma que para cada frecuencia se cumpla que el módulo sea superior al módulo de la incertidumbre, esto es,

$$|W_T(j\omega)| \geq |E_{o,i}(j\omega))\ \ \forall\omega\ \ \ i = 1, \ldots, q \tag{5.10}$$

Además, dado que $W_T(s)$ pondera a la función de sensibilidad complementaria, que debe tener una ganancia pequeña en alta frecuencia, se diseñará aquélla de forma que su módulo posea un valor alto en alta frecuencia.

La función $W_S(s)$ será empleada para imponer condiciones al comportamiento del sistema. Sin embargo, hay que recordar que será $W_S(s)^{-1}$ la función que moldee a la función de sensibilidad $S(s)$ en forma de cota superior.

Se propone una función de transferencia $W_S(s)$ de la forma

$$W_S(s) = \frac{\alpha s + \omega_S}{s + \beta\omega_S} \tag{5.11}$$

Los parámetros se eligen de la siguiente manera:

- α : ganancia de la función en alta frecuencia. Es deseable que $S(s)$ tenga un valor del pico máximo del módulo en torno a 2 y como $W_S(s)^{-1}$ actúa como su cota superior, un valor apropiado para α debería ser del orden de

$$|W_S^{-1}(s \to \infty)| = \frac{1}{\alpha} = 2 \Rightarrow \alpha = 0{,}5$$

- β : ganancia de la función a baja frecuencia. Hace las veces de cota superior del error en régimen permanente permitido. Sin embargo, por problemas en el cálculo de la planta aumentada, un valor apropiado para este coeficiente puede estar entre 10^{-6} y 10^{-4}.

- ω_S : frecuencia de corte de la función. Marcará la frecuencia de corte mínima que se requiere de la función de sensibilidad. Se puede expresar como

$$\omega_S = 10^{(\kappa-1)}\omega_T \tag{5.12}$$

  donde ω_T es la frecuencia de corte de la función $W_T(s)$ y κ será el parámetro encargado de variar logarítmicamente el valor de ω_S. Cuanto mayor sea κ la respuesta será más rápida y más oscilante; normalmente $\kappa \in [0{,}1]$.

Para la elección del parámetro κ, un buen valor final para sistemas monovariables puede estar entre 0.5 y 1. Esto se justifica con el hecho de que si la función $W_T(s)$ es una cota superior de la incertidumbre, se puede suponer que a partir de la frecuencia ω_T se tiene una incertidumbre multiplicativa superior a la unidad. Por tanto, el desconocimiento del sistema será total partir de esta frecuencia. Con el valor $\kappa = 1$ se está imponiendo que la frecuencia de corte de $W_S(s)$ sea precisamente ω_T, por lo que con valores superiores de κ se estaría pidiendo al controlador que funciones bien en frecuencias donde se desconoce completamente la dirección del sistema.

Por otra parte, la función $W_U(s)$ pondera la sensibilidad al control, así que permite penalizar la señal de control en el rango de frecuencia deseado. De esta forma se disminuye la sobreoscilación de la respuesta temporal del sistema sin afectar mucho a la rapidez. Se propone una $W_U(s) = cte$

Hay diversos parámetros para establecer la bondad del controlador H_∞ diseñado como, por ejemplo, un parámetro denominado γ que indica, entre otras cosas, si el módulo de alguna función de sensibilidad supera al de su función de ponderación asociada en alguna frecuencia, aunque no precisa en cuál.

Finalmente, es importante realizar un escalado del sistema como paso previo al diseño del controlador. Evita problemas numéricos (el número de condición γ depende del escalado del sistema) y permite que todas las señales sean comparables en magnitud.

5.2. Diseño para el Sistema Levineu.

En esta sección se va a exponer la aplicación específica de un control H_∞ al Sistema Levineu. Se presentan posteriormente algunos resultados experimentales obtenidos en la planta real.

5.2.1. Síntesis del controlador.

Gracias a la metodología explicada en la sección 5.1 no es necesario tener unos conocimientos teóricos muy amplios para diseñar un controlador avanzado H_∞; sin embargo, es conveniente tener una información precisa del proceso a controlar. Cuanto mayor sea el conocimiento de la dinámica del sistema real, mejor será la elección del modelo nominal y podremos distinguir en qué puntos de operación funcionará bien el controlador.

Los objetivos que se pretenden conseguir son los siguientes:

1. Garantizar la estabilidad para las zonas de operación de la planta.
2. Conseguir un buen comportamiento para cada uno de estos puntos de operación.

El proceso de síntesis del controlador se formula como un problema de optimización H_∞ que será resuelto de forma subóptima empleando una configuración de sensibilidad mixta $S/KS/T$. Para ello se hallará un controlador estabilizante que minimice en lo posible la norma 5.7.

Modelado del sistema.

Para la elección del modelo nominal que se utilizará en la síntesis del controlador se acudió a los resultados de los experimentos de identificación anotados en la sección 3.2.

Como primera opción se probó un modelo nominal con la misma forma que los modelos en los distintos puntos de funcionamiento y cuyos coeficientes fueron elegidos como la media de los valores estimados en cada punto de funcionamiento. Viendo que la planta identificada en el punto de operación 3 (ver sección 3.2) tenía una magnitud bastante inferior a las otras dos, se decidió no tenerla en cuenta y diseñar un controlador que funcionase mejor cerca de los puntos de operación 1 y 2. El diferente funcionamiento

del sistema en el punto 3 se explica teniendo en cuenta la variabilidad del flujo del aire a lo largo de su trayectoria. No obstante, este punto es el de mayor tensión de entrada (7,5V), por lo que la bola alcanza mayor altura y todavía no se comprende bien el comportamiento del flujo en puntos alejados del origen ([Cap05]). El modelo nominal quedó de la siguiente forma:

$$G_{nom}(s) = \frac{1{,}8586s^2 - 83{,}8109s + 1538{,}6592}{s^3 + 18{,}5259s^2 + 25{,}3934s + 259{,}7276} \tag{5.13}$$

Este modelo nominal produce un diagrama de Bode muy parecido a las plantas identificadas en casi todas las frecuencias, sobre en los puntos de operación 1 y 2, con menor tensión de entrada (ver figura 5.4).

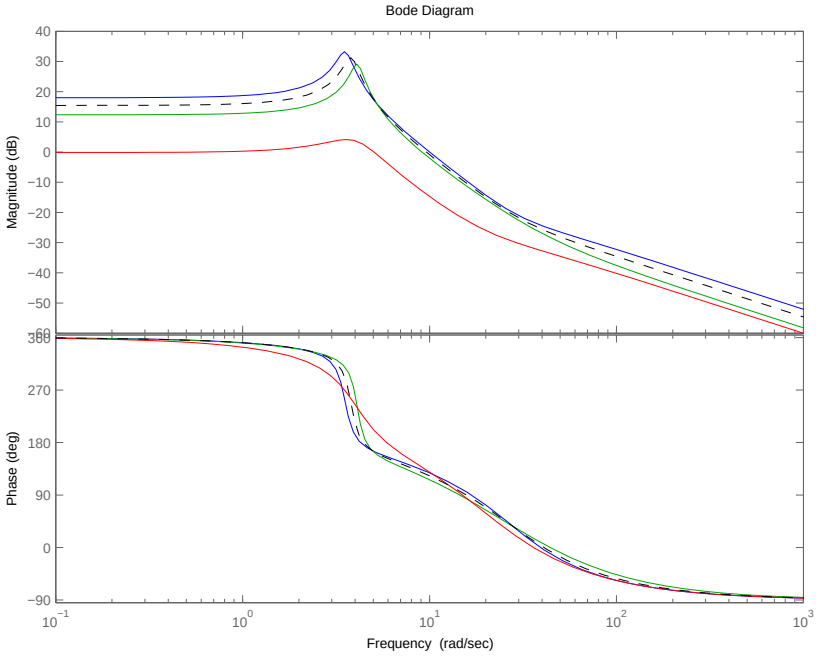


Figura 5.4: Diagrama de Bode de las plantas y el modelo nominal con la media de los coeficientes en los puntos de operación 1 y 2.

Sin embargo, realmente para la síntesis del controlador sólo interesa que el modelo nominal se asemeje a las plantas experimentales en el rango de frecuencias de funcionamiento del sistema. Se calcularon las incertidumbres multiplicativas mediante la definición 5.8 y se muestran en la figura 5.5. Se observa como hay una primera curva de la incertidumbre que supera la unidad (0dB) y por tanto, en frecuencias superiores se desconoce totalmente el comportamiento del sistema. Esta frecuencia límite se encuentra en torno a 3,3rad/s.

Posteriormente, se intentó modelar la planta con una función de transferencia más sencilla, con el objetivo de conseguir un orden menor del controlador final. Se probó por

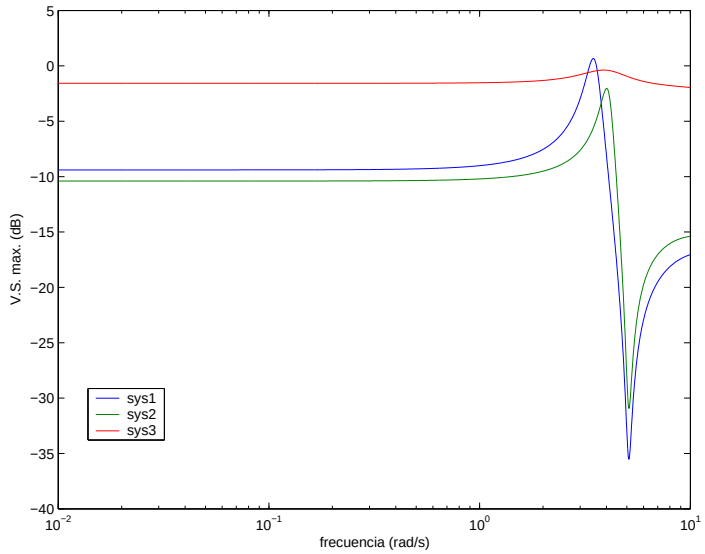


Figura 5.5: Incertidumbres multiplicativas de las plantas identificadas respecto del modelo nominal con la media de los coeficientes en los puntos de operación 1 y 2.

tanto un modelo nominal de la forma

$$G(s) = \frac{K_1(s + K_c\omega_n)}{s^2 + 2\delta\omega_n + \omega_n^2} \tag{5.14}$$

Tras varios experimentos, se escogieron unos parámetros adecuados en el modelo nominal para acercar lo más posible su comportamiento frecuencial al de las plantas experimentales, principalmente en el rango de funcionamiento. De esta forma quedó el siguiente modelo nominal:

$$G(s) = \frac{1 \cdot (s + 6 \cdot 3{,}7)}{s^2 + 2 \cdot 0{,}1 \cdot 3{,}7 + 3{,}7^2} \tag{5.15}$$

En la figura 5.6 se comprueba el gran parecido en el rango de frecuencias de interés entre el comportamiento de la planta nominal y el de las plantas en los puntos de operación a menor tensión. Por otra parte,las incertidumbres multiplicativas se muestran en la figura 5.7. Se observa como la frecuencia límite de conocimiento del sistema se encuentra en torno a 3.3rad/s, igual que en el caso de la figura 5.5.

A continuación, se realizó el escalado de las plantas. Para llevarlo a cabo fue necesario determinar las magnitudes de los máximos cambios de la señal de control u (tensión en V) y la máxima variación permitida para la salida y(altura en cm). En esta aplicación, las plantas fueron escaladas de la siguiente forma:

$$\hat{G}(s) = G(s)\frac{\triangle u_{max}}{\triangle y_{max}} = G(s)\frac{0{,}5}{10} \tag{5.16}$$

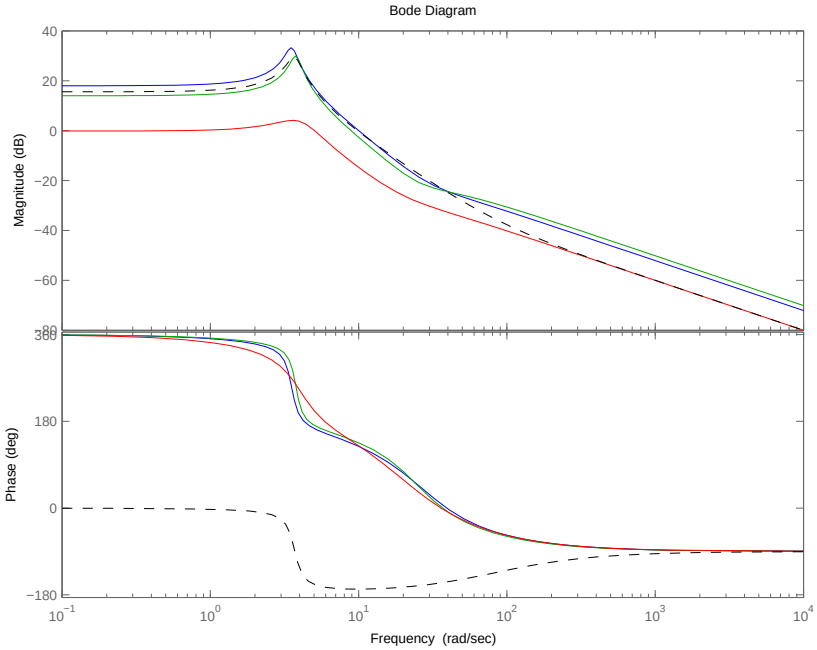


Figura 5.6: Diagrama de Bode de las plantas y el modelo nominal reducido.

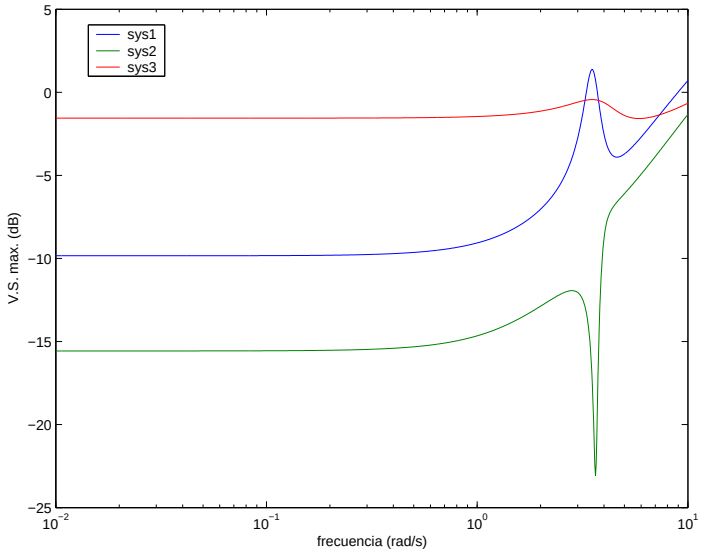


Figura 5.7: Incertidumbres multiplicativas de las plantas identificadas respecto del modelo nominal reducido.

Diseño de las funciones de ponderación.

La segunda fase de la síntesis del controlador H_∞ consiste en el diseño de las funciones de ponderación que moldearán a las funciones de sensibilidad en BC. Es la fase donde el diseñador puede intervenir con mayor intensidad.

Se diseñaron las funciones en continuo. Si se empleara el algoritmo de síntesis en discreto, estas funciones deberían ser discretizadas (por ejemplo, con una transformación bilineal), para poder construir la planta generalizada en tiempo discreto.

Recordando las propuestas de la sección 5.1, se eligieron como funciones de ponderación:

$$\begin{aligned} W_T(s) &= 10^{KWt/20}\frac{\tau s+1}{\tau\cdot 10^{(KWt-Kaf)/20}s+1} = \\ &= 10^{-9{,}5/20}\frac{0{,}7s+1}{0{,}7\cdot 10^{(-9{,}5-40)/20}} = 0{,}335\frac{0{,}7s+1}{0{,}0023s+1} \qquad (5.17) \\ W_S(s) &= \frac{\alpha s+10^{(\kappa-1)}\omega_T}{s+\beta 10^{(\kappa-1)}\omega_T} = \frac{0{,}5s+10^{(\kappa-1)}\cdot 4{,}0183}{s+10^{-4}\cdot 4{,}0183} \qquad (5.18) \\ W_{KS}(s) &= 0{,}1 \qquad (5.19) \end{aligned}$$

El parámetro κ de $W_S(s)$ se empleará para diseñar diferentes controladores. El valor inicial para κ es igual a 0 y se podrá incrementar hasta más o menos la unidad, intentando conseguir la realización más apropiada. Se debe recordar que realmente las funciones de sensibilidad son ponderadas por la inversa de $W_S(s)$ y la de $W_T(s)$.

La función de ponderación W_{KS} se diseñó inicialmente con un valor constante e igual a 1, pero luego se modificó hasta el valor definitivo de 5.19.

En la figura 5.8 se representa el módulo de $W_T(s)$ y se observa que es una cota superior de las incertidumbres multiplicativas en casi todo el rango de frecuencias en el que se conoce el sistema. A partir de $\omega = 2rad/s$ hay una incertidumbre cuyo módulo supera a $|W_T|$; esto se diseña así para imponer unas condiciones un poco más estrictas al controlador. La frecuencia de corte ω_T es igual a $4rad/s$.

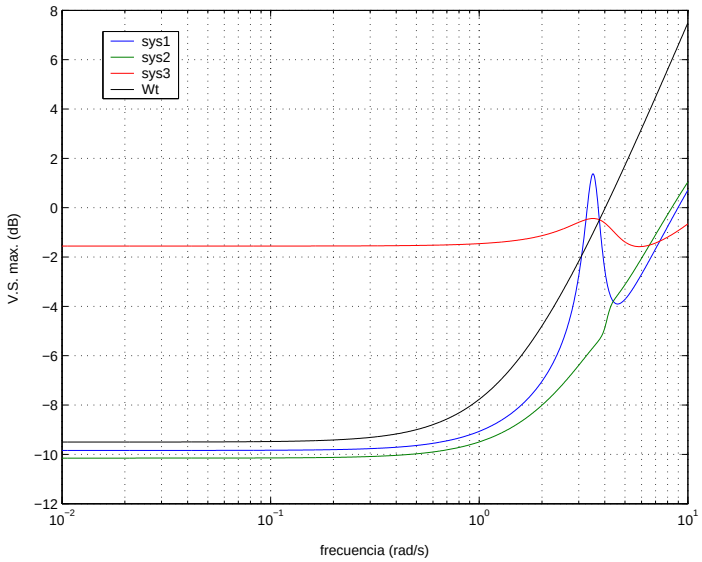


Figura 5.8: $|W_T(s)|$ y las incertidumbres multiplicativas.

Construcción de la planta aumentada y obtención del controlador.

En esta fase se ha empleado un algoritmo de síntesis en el tiempo continuo, según el esquema que se muestra a continuación:

$$\begin{array}{ccccccc} W_T(s) & S/KS/T & & & & bilin & \\ + & \downarrow & & & & \downarrow & \\ \hat{G}(s) & \rightarrow & \hat{P}(s) & \rightarrow & \hat{K}(s) & \rightarrow & \hat{C}(z) \\ + & & & & & & \\ W_S(s) & & & & & & \end{array}$$

Es decir, primero se construye la planta aumentada con el esquema de sensibilidad mixta $S/KS/T$. Luego se calcula el controlador H_∞ en continuo. Se discretiza este controlador con una bilineal de Tustin. Por último, para obtener el controlador real, debe realizarse un desescalado del controlador en discreto ya calculado. El desescalado se realiza mediante la expresión:

$$K(z) = \hat{K}(z)\frac{\triangle u_{max}}{\triangle y_{max}} = \hat{K}(z)\frac{0{,}5}{10} \tag{5.20}$$

$$K(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_N z^{-N}}{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}} \tag{5.21}$$

El controlador programado en el autómata será una ecuación en diferencias del tipo:

$$u_k = \frac{1}{a_0} \cdot (e_k \cdot b_0 + e_{k-1} \cdot b_1 + \ldots + e_{k-N} \cdot b_N - u_{k-1} \cdot a_1 - \ldots - u_{k-N} \cdot a_N) \tag{5.22}$$

Una vez obtenido el controlador, pueden calcularse las funciones de sensibilidad (ya que incluyen al propio controlador). En la figura 5.9 se representan el módulo de la función de sensibilidad complementaria $T(s) = G_{BC}(s)$ y el de su ponderación $W_T^{-1}(s)$.

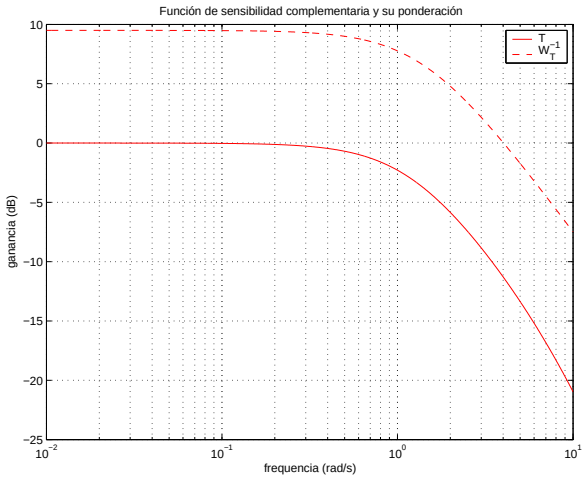


Figura 5.9: Módulo de las funciones $T(s)$ y $W_T^{-1}(s)$.

En la figura 5.10 se representan el módulo de la función de sensibilidad al control $C(s)S(s)$ y el de su ponderación $W_{KS}^{-1}(s)$.

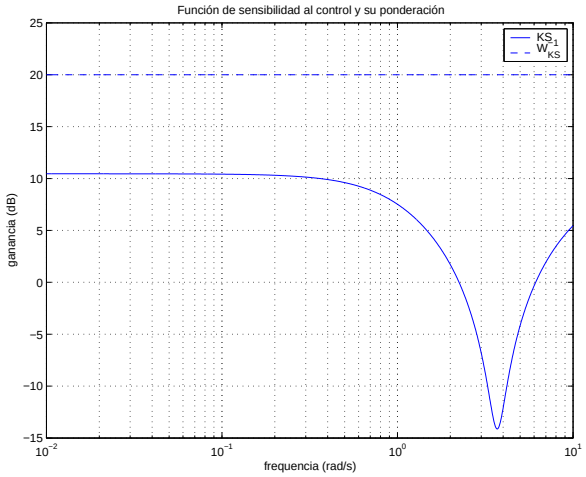
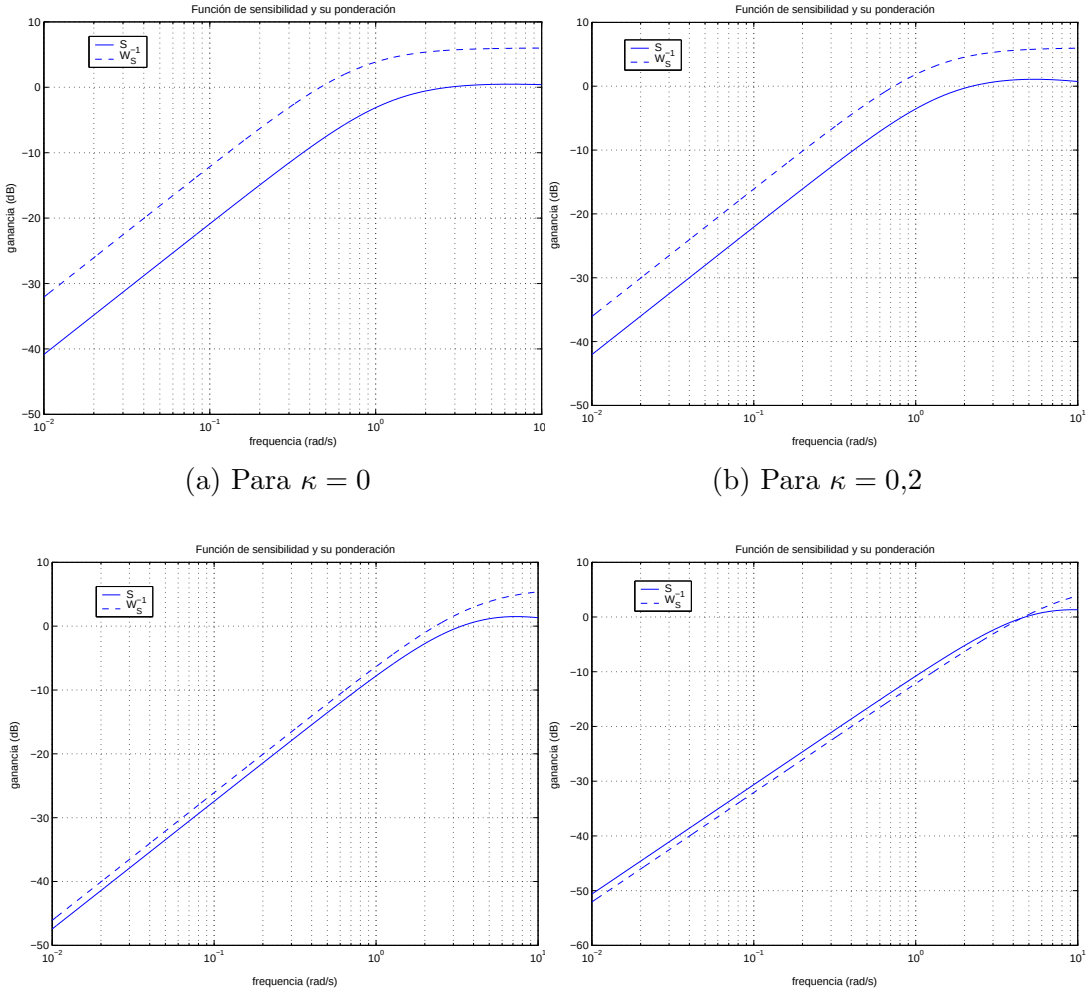


Figura 5.10: Módulo de las funciones $C(s)S(s)$ y $W_{KS}^{-1}(s)$.

A continuación se muestran las funciones $S(s)$ y $W_S^{-1}(s)$ para diferentes valores de κ (figuras 5.11).



(a) Para $\kappa = 0$

(b) Para $\kappa = 0{,}2$

(c) Para $\kappa = 0{,}7$

(d) Para $\kappa = 1$

Figura 5.11: Sensibilidad $S(s)$ y su ponderación $W_S^{-1}(s)$ para diferentes valores de κ.

5.2.2. Resultados.

En los experimentos, se utiliza la herramienta diseñada en LabView/Matlab/CX-Programmer. Los controladores H_∞ se construyen con Matlab y se envían los coeficientes (numerador y denominador) al autómata mediante LabView. El control en tiempo real se ejecuta en el autómata. Al enviar los coeficientes se realiza un truncamiento de los valores por lo que podrían generarse problemas si dicho truncamiento produjera una desviación en algún hipotético polo o cero crítico. Una posible mejora para evitar este problema consiste en enviar al autómata los valores de los polos y ceros, en vez de los coeficientes del controlador.

Como ya se comentó, en el método de diseño aquí empleado se modifica el valor del parámetro κ para obtener controladores con diferentes características. Dado que, en esta aplicación, la incertidumbre del sistema real respecto del modelado nominal supera la unidad a partir de $\omega = 3{,}3 rad/s < \omega_T$, el límite superior de κ no puede ser 1 (como se mencionó en la sección 5.1) sino

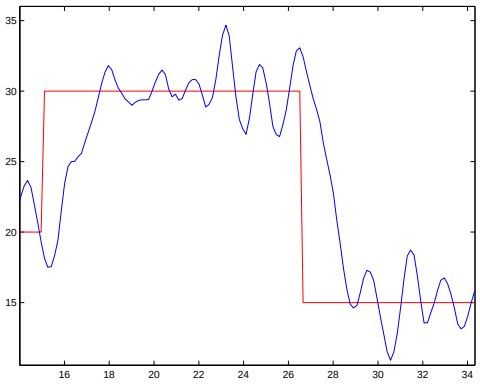
$$\kappa_{max} = 1 + log\left(\frac{\omega_S}{\omega_T}\right) = 1 + log\left(\frac{3{,}3}{4}\right) = 0{,}8076$$

Esto se justifica con el hecho de que con un valor $\kappa > \kappa_{max}$ se está imponiendo que la frecuencia de corte de $W_S(s)$ sea mayor que la frecuencia de corte de la incertidumbre multiplicativa, por lo que se estaría pidiendo al controlador que funciones bien en frecuencias donde se desconoce completamente la dirección del sistema.

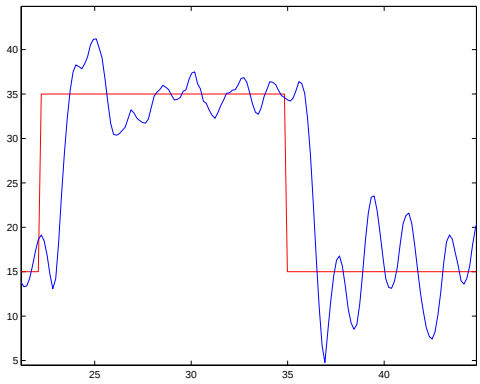
A continuación se exponen resultados de experimentos para diferentes valores de κ: los controladores obtenidos, el parámetro γ conseguido y respuestas ante escalón.

$$\begin{aligned}
\kappa = 0 \rightarrow C_{Caso1}(z) &= \frac{0{,}0935s^4 + 0{,}0289s^3 - 0{,}1233s^2 + 0{,}0204s + 0{,}0792}{s^4 + 0{,}8313s^3 - 1{,}0418s^2 - 0{,}8350s + 0{,}0454} \\
\kappa = 0{,}2 \rightarrow C_{Caso2}(z) &= \frac{0{,}0586s^4 + 0{,}0181s^3 - 0{,}0773s^2 + 0{,}0128s + 0{,}0496}{s^4 - 0{,}1143s^3 - 0{,}8754s^2 + 0{,}0471s - 0{,}0574} \\
\kappa = 0{,}5 \rightarrow C_{Caso3}(z) &= \frac{0{,}1502s^4 + 0{,}0465s^3 - 0{,}1982s^2 + 0{,}0328s + 0{,}1272}{s^4 + 0{,}5904s^3 - 1{,}0023s^2 - 0{,}6100s + 0{,}0220} \\
\kappa = 0{,}7 \rightarrow C_{Caso4}(z) &= \frac{0{,}1349s^4 + 0{,}0417s^3 - 0{,}1780s^2 + 0{,}0294s + 0{,}1143}{s^4 + 0{,}1460s^3 - 0{,}9128s^2 - 0{,}1962s - 0{,}0369} \\
\kappa = 1 \rightarrow C_{Caso5}(z) &= \frac{0{,}2619s^4 + 0{,}0810s^3 - 0{,}3457s^2 + 0{,}0571s + 0{,}2219}{s^4 + 0{,}5953s^3 - 0{,}9605s^2 - 0{,}6170s - 0{,}0176}
\end{aligned}$$

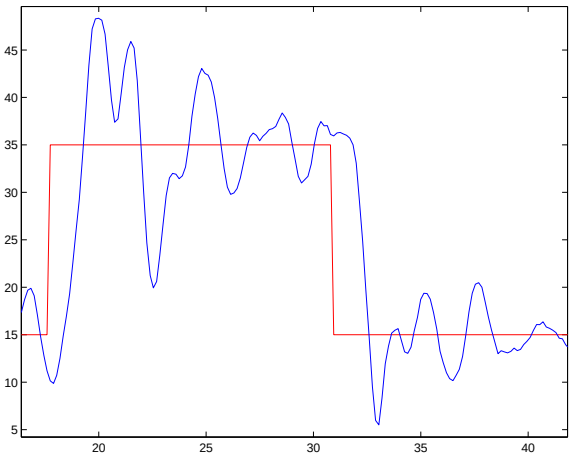
$$Para\ \kappa = 0 \rightarrow \gamma = 0{,}5967$$
$$Para\ \kappa = 0{,}2 \rightarrow \gamma = 0{,}6934$$
$$Para\ \kappa = 0{,}5 \rightarrow \gamma = 0{,}7900$$
$$Para\ \kappa = 0{,}7 \rightarrow \gamma = 0{,}9834$$
$$Para\ \kappa = 1 \rightarrow \gamma = 1{,}2734$$



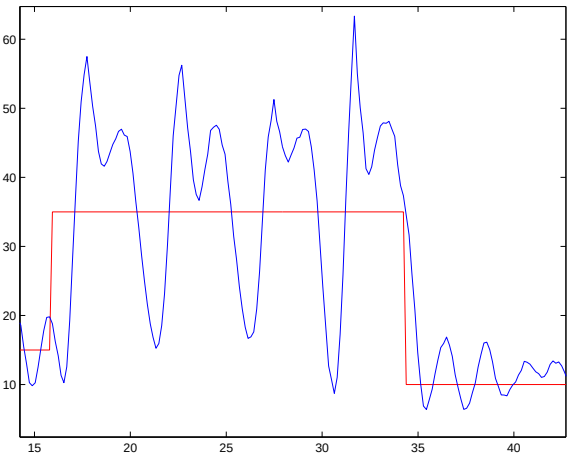
(a) Para $\kappa = 0$



(b) Para $\kappa = 0{,}5$



(c) Para $\kappa = 0{,}7$



(d) Para $\kappa = 1$

Figura 5.12: Respuesta a escalones del control H_∞ para diferentes valores de κ.

Finalmente, viendo los resultados experimentales con la planta real, se eligió un valor para $\kappa = 0{,}2$, es decir, el caso número 2. Las matrices de estado del controlador propuesto se muestran en la figura 5.13.

En las figuras 5.14 se muestra la respuesta del controlador propuesto para el sistema funcionando en torno al punto nominal de operación. El comportamiento observado es

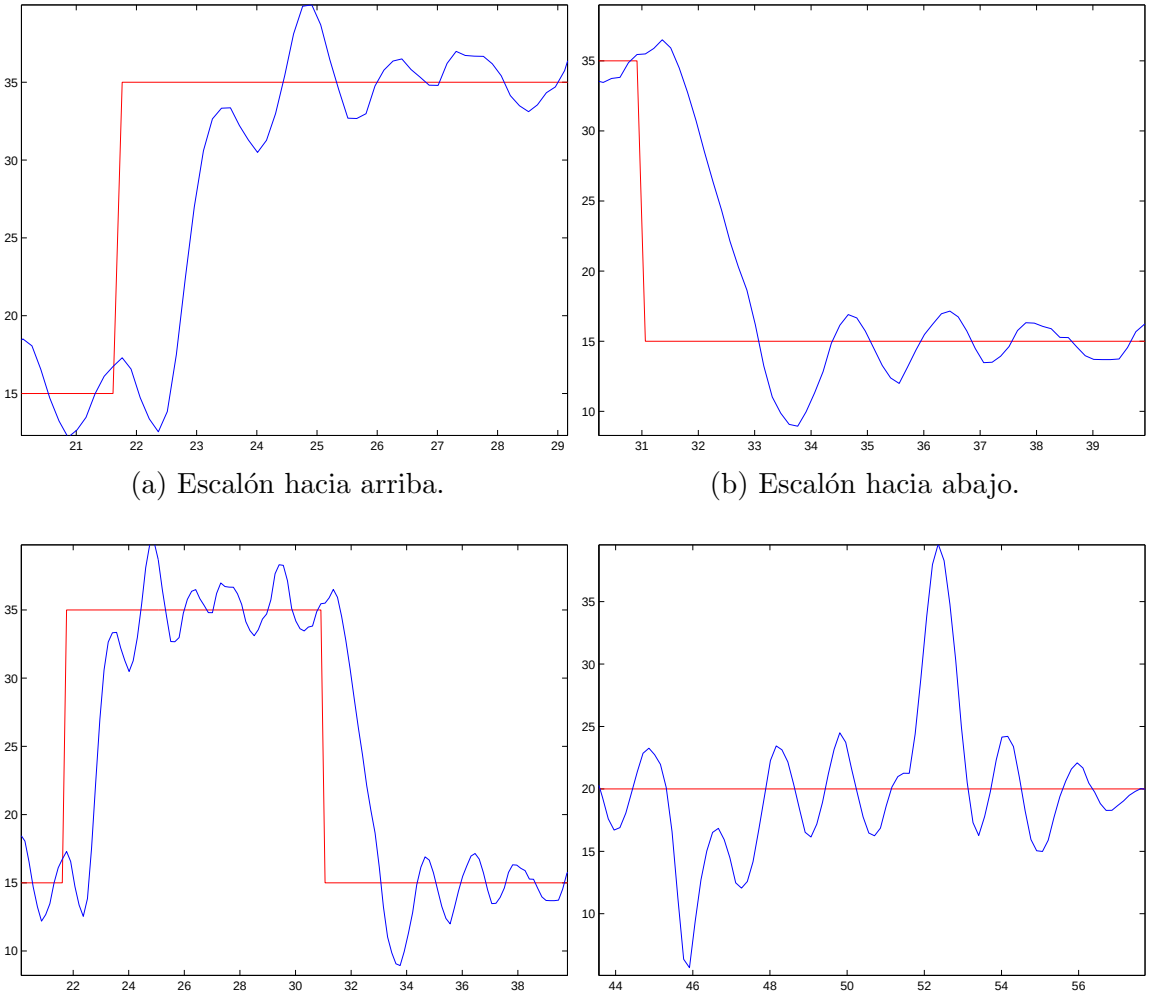
$$A_C = \begin{bmatrix} -25{,}27 & -433{,}1 & 31{,}2 & 68{,}36 \\ 1 & 0 & 0 & 0 \\ -1{,}3 \cdot 10^{-17} & 8{,}9 \cdot 10^{-16} & -6{,}4 \cdot 10^{-5} & 0 \\ 20{,}62 & 1693 & 0 & -426{,}5 \end{bmatrix} ; \quad B_C = \begin{bmatrix} 0 \\ 0 \\ 11{,}50 \\ 0 \end{bmatrix}$$

$$C_C = \begin{bmatrix} -3{,}405 & -58{,}23 & 4{,}332 & 9{,}492 \end{bmatrix} ; \qquad D_C = \begin{bmatrix} 0 \end{bmatrix}$$

Figura 5.13: Matrices de estados del controlador propuesto.

aceptable, con un tiempo de subida de $1{,}5s$, sobreoscilación del 20 %, tiempo de establecimiento de $4s$(figura 5.14a). La respuesta ante perturbaciones también es rápida. El tiempo muerto es de $1s$ y se debe en parte a retardos en la comunicación y el procesamiento de los datos (que consumen casi $0{,}5s$).

En los experimentos los escalones no parten desde 0. Si el actuador estuviera en reposo, hay que tener en cuenta que, al arrancar, necesita abandonar la zona muerta (que abarca hasta una tensión de 1V), por lo que, seguramente, el valor óptimo de κ debería subir ligeramente para contrarrestar este retardo.



(a) Escalón hacia arriba.

(b) Escalón hacia abajo.

(c) Varios escalones.

(d) Ante perturbaciones.

Figura 5.14: Respuesta ante escalón del controlador H_∞ propuesto (con $\kappa = 0{,}2$).

Capítulo 6

Control Predictivo Generalizado (GPC).

6.1. Teoría del Control Predictivo Generalizado.

6.1.1. Control Predictivo Basado en Modelo (MPC).

En la actualidad el objetivo de un sistema de control no consiste sólo en mantener una operación estable del proceso en cuestión, sino en actuar sobre las variables manipuladas de forma que puedan satisfacerse múltiples y cambiantes criterios de funcionamiento (económicos, de seguridad, medioambientales o de calidad) en presencia de cambio en las características de dicho proceso. Las técnicas de Control Predictivo Basado en Modelo (*Model Based Predictive Control*, MPC) constituyen poderosas herramientas para afrontar este reto. El MPC, en su expresión más general, acepta cualquier tipo de modelos, funciones objetivo y restricciones, razón que probablemente explique su éxito en la industria, unida a que es la manera más general de formular el problema de control en el dominio del tiempo.

El MPC es una familia de métodos de diseño de controladores lineales con las siguientes ideas comunes:

- Uso explícito de un modelo para predecir la salida del proceso en futuros instantes de tiempo (horizonte).
- Cálculo de las señales de control minimizando una función objetivo.

- Estrategia deslizante: en cada instante el horizonte se va desplazando hacia el futuro.

- Aplicación de la primera señal de control calculada y desechar el resto, porque se repetirá el cálculo en el siguiente instante de muestreo.

Inicialmente el Control Predictivo se desarrolló según dos líneas de trabajo. Por un lado, surgieron algoritmos que usaban explícitamente un modelo dinámico del proceso para predecir el efecto en la salida de las acciones de control futuras, las cuales eran determinadas minimizando el error predicho sujeto a restricciones de operación. La optimización se repetía en cada instante de muestreo con la información actualizada. Esta línea se hizo popular por su simplicidad y por el uso del modelo de respuesta impulsional o en escalón, más intuitivo y fácilmente identificable. Se diseñaron aplicaciones sobre sistemas multivariables con restricciones. Por otra parte, surgieron estrategias en torno a las ideas de control adaptativo, esencialmente para procesos monovariables formulados con modelos entrada/salida. Se extendieron las ideas del Control de Mínima Varianza y se propuso el Control Predictivo Generalizado (GPC), que es uno de los métodos más empleados en la actualidad.

6.1.2. Formulación del GPC.

El *Control Predictivo Generalizado* (GPC) fue propuesto por Clarke *et al.* en 1987 y se ha extendido tanto en el mundo industrial como en el académico. Tiene un rango de aplicaciones muy amplio, muestra buenas prestaciones en el control y robustez respecto a sobreparametrización o retardos mal conocidos.

La idea básica del GPC es calcular una secuencia de futuras acciones de control de tal forma que minimice una función de coste multipaso. El índice de coste es una función cuadrática que mide el error entre la salida predicha del sistema y una trayectoria de referencia hasta el horizonte de predicción, y también el esfuerzo de control necesario para obtener dicha salida.

Modelo del proceso y de las perturbaciones.

En el GPC se emplea la función de transferencia para modelar el proceso. La mayoría de los procesos de una sola entrada y una sola salida (Single-Input Single-Output, SISO) pueden ser descritos, tras ser linealizados en torno a un determinado punto de trabajo, de la siguiente forma:

$$A(z^{-1})y(t) = z^{-d}B(z^{-1})u(t-1) + \frac{C(z^{-1})e(t)}{D(z^{-1})} \tag{6.1}$$

donde *u(t)* es la señal de control, *y(t)* es la salida del proceso, *e(t)* es un ruido blanco de media cero y d es el tiempo muerto del sistema. A, B y C tienen la siguiente forma:

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{na} z^{-na} \tag{6.2}$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_{nb} z^{-nb} \tag{6.3}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + ... + c_{nc} z^{-nc} \tag{6.4}$$

Este modelo expuesto es conocido como Autorregresivo Integrado de Media Móvil (Controller Auto-Regressive Integrated Moving-Average, CARIMA).

Si para modelar las perturbaciones se escoge un modelo integrado con

$$D(z^{-1}) = 1 - z^{-1} = \Delta \tag{6.5}$$

se consigue un control sin error en permanente. Por simplicidad, se escoge $C(z^{-1}) = 1$, puesto que, si C^{-1} puede ser truncado, se puede absorber en A y B. De este modo, el modelo queda finalmente

$$A(z^{-1})y(t) = z^{-d}B(z^{-1})u(t-1) + \frac{e(t)}{\Delta} \tag{6.6}$$

y es conocido como ARIMA.

Función objetivo.

El algoritmo del GPC consiste en aplicar una secuencia de señales de control que minimice una función objetivo o de costes de la forma:

$$J(N_1, N_2, Nu) = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{Nu} \lambda(j)[\triangle u(t+j-1)]^2 \tag{6.7}$$

donde $\hat{y}(t+j|t)$ es la predicción óptima j pasos hacia delante de la salida del proceso con datos conocidos hasta el instante t, *w(t+j)* es la futura trayectoria de referencia; N_1 y N_2 son los horizontes mínimo y máximo de predicción, Nu es el horizonte de control y δ(j) y λ(j) son las secuencias de ponderación.

La trayectoria de referencia *w(t+j)* no tiene porqué coincidir con la referencia real; de hecho, puede ser una curva suave desde *y(t)* hasta la referencia.

Predicción óptima.

Para minimizar la función de coste hay que obtener previamente la predicción óptima de *y(t+j)* para $j \geq N_1$ y $j \leq N_2$. El desarrollo está expuesto en [Bor96] y parte desde el modelo CARIMA y la ecuación diofántica:

$$1 = E_j(z^{-1})\triangle A + z^{-j}F_j(z^{-j}) = E_j(z^{-1})\tilde{A} + z^{-j}F_j(z^{-j}) \tag{6.8}$$

El conjunto de predicciones óptimas puede ser escrito en forma matricial como:

$$\mathbf{y} = \mathbf{Gu} + \mathbf{F}(z^{-1})y(t) + \mathbf{G'}(z^{-1})\triangle u(t-1) = \mathbf{Gu} + \mathbf{f} \tag{6.9}$$

donde:

$$\mathbf{y} = \begin{bmatrix} \hat{y}(t+d+1|t) \\ \hat{y}(t+d+2|t) \\ \vdots \\ \hat{y}(t+d+N|t) \end{bmatrix}; \qquad \mathbf{u} = \begin{bmatrix} \triangle u(t) \\ \triangle u(t+1) \\ \vdots \\ \triangle u(t+N) \end{bmatrix};$$

$$\mathbf{G} = \begin{bmatrix} g_0 & 0 & \dots & 0 \\ g_1 & g_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g_{N-1} & g_{N-2} & \dots & g_0 \end{bmatrix}; \mathbf{F}(z^{-1}) = \begin{bmatrix} F_{d+1}(z^{-1}) \\ F_{d+2}(z^{-1}) \\ \vdots \\ F_{d+N}(z^{-1}) \end{bmatrix};$$

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

y con $N = N_2 - N_1$. Como hemos estimado un ruido de media cero, no aparece en la predicción óptima. Al tener el proceso un retardo de d períodos de muestreo, la salida será influida por la señal de control después del instante d+1. Por este motivo, se definirán $N_1 = d + 1$, $N_2 = d + N$ y $N_u = N$.

Ley de control.

Una vez obtenido un modelo de predicción y la función de costes, se debe minimizar ésta última para encontrar la ley de control óptima. Sustituyendo 6.9 en 6.7 y minimizando

$$min J(u) \Rightarrow \frac{dJ(u)}{du} = 0; \qquad \frac{d^2J(u)}{du^2} > 0 \tag{6.10}$$

se obtiene, con $\delta = 1$ y siempre que no existan restricciones en la señal de control:

$$\begin{aligned} J(u) &= (\mathbf{Gu} + \mathbf{f} - \omega)^*(\mathbf{Gu} + \mathbf{f} - \omega) + \lambda \mathbf{u}^T\mathbf{u} \ \rightarrow \\ J(u) &= \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{b}\mathbf{u} + \mathbf{f}_0 \ \rightarrow \ \text{min J(u)} \ \rightarrow \end{aligned} \tag{6.11}$$

de donde (6.12)

$$\Delta \mathbf{u} = -\mathbf{H}^{-1}\mathbf{b}^T \tag{6.13}$$

con:

$$\begin{aligned} \mathbf{H} &= 2(\mathbf{G}^T\mathbf{G} + \lambda \mathbf{I}) \\ \mathbf{b} &= 2(\mathbf{f} - \mathbf{w}^T\mathbf{G}) \end{aligned}$$

De esta secuencia de actuaciones u_k se aplica únicamente la primera Δu_0, puesto que en el siguiente período de muestreo, se vuelve a calcular la secuencia que optimiza la función de costes. Para reducir la carga de cálculo se asume que las señales de control serán constantes a partir del intervalo $N_u < N$ y se escoge N_u pequeña.

6.2. Diseño para el sistema Levineu.

6.2.1. Síntesis del controlador.

El controlador GPC que se diseña para el Sistema Levineu es sintetizado en Matlab, al igual que el controlador H_∞, pero, a diferencia de éste, la señal de control en tiempo real también es calculada en el propio Matlab.

Cuando se desea sintetizar un nuevo controlador GPC, porque se ha cambiado la configuración de los parámetros, deben calcularse las matrices que permiten luego definir la ley de control específica.

En primer lugar debe definirse la planta nominal en discreto. Se tomará el mismo modelo nominal en continuo que para el controlador H_∞ (ver sección 5.2). Se discretiza con una bilineal de Tustin de modo que la planta queda:

$$G(z) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{0{,}4742 + 0{,}8160z^{-1} + 0{,}3418z^{-2}}{1 - 1{,}6300z^{-1} + 0{,}9020^{-2}} \tag{6.14}$$

El modelo ARIMA empleado en la teoría del GPC (ver sección 6.1.2) toma, teniendo en cuenta que no se han considerado retrasos, la siguiente expresión:

$$(1 - 1{,}6300z^{-1} + 0{,}9020^{-2})y(t) = (0{,}4742 + 0{,}8160z^{-1} + 0{,}3418z^{-2})u(t-1) + \frac{e(t)}{\Delta} \tag{6.15}$$

A continuación debe definirse la función de costes a minimizar (ver sección 6.1.2). En esta función hay varios parámetros que deben determinarse como parte del diseño del controlador. Por tanto, dada una planta nominal en discreto del sistema, se puede variar el controlador GPC sintetizado jugando con estos parámetros, que son:

- Horizonte de predicción ($hp = N_2 - N_1 = 50$).
- Horizonte de control ($hc = N_u = 50$).
- Coste asociado al error en posición ($\delta = 1$).
- Coste asociado al incremento en la señal de control ($\lambda = 500$)
- Coste asociado al consumo de la acción de control ($\kappa = 0$).

Debe tenerse en cuenta al diseñar que los horizontes deben abarcar el tiempo de establecimiento del sistema, es decir, que multiplicando estos parámetros por el tiempo de muestreo deben resultar un periodo mayor que el de establecimiento. Como en nuestro sistema no importa el consumo, su coste asociado es nulo. Si se deriva la función objetivo 6.7 y se iguala a cero para hallar el mínimo, se observa que más importante que el valor absoluto de los costes δ y λ es la relación entre ellos. Por eso y por motivos de desarrollo matemático, se mantendrá el coste $\delta = 1$ y se variará el coste λ para obtener diferentes controladores.

Tras seleccionar los valores de los parámetros, se calculan las matrices de predicción óptima G, G' y F(ver ecuación 6.9), y luego las matrices de control óptimo H y b.

En este punto ya se han determinado las matrices que permitirán calcular la señal de control en tiempo real. Estas matrices no cambian a lo largo del tiempo si los parámetros antes seleccionados no varían (incluido el tiempo de muestreo). Ahora bien, en cada periodo de muestreo debe recalcularse la secuencia de señales de control para todo el horizonte de control, aunque sólo se aprovechará la primera de ellas. Para calcular la secuencia óptima de la señal de control se desarrolló la fórmula 6.13:

$$\mathbf{u} = -\mathbf{H}^{-1}\mathbf{b}^T = \underbrace{(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})}_{\mathbf{H/2}}\mathbf{G}^T(\omega - \mathbf{f}) \tag{6.16}$$

En la práctica esta fórmula 6.16 se aplica de forma incremental y, además, basta con calcular la primera fila de $\mathbf{H/2}$ porque sólo se aplica el primer elemento de la secuencia óptima de $\mathbf{u}$. Por tanto, con $\mathbf{HG}$ = fila 1 $\{\mathbf{H/2}\}$, queda:

$$\Delta\mathbf{u}(t) = \mathbf{HG}(\omega - \mathbf{f}) = \tag{6.17}$$

$$= \mathbf{HG}\begin{bmatrix} w(t+1) \\ \vdots \\ w(t+hp) \end{bmatrix} - (\mathbf{HG}\cdot\mathbf{G'})\Delta\mathbf{u}(t-1) - (\mathbf{HG}\cdot\mathbf{F})\mathbf{y}(t) \tag{6.18}$$

Y, finalmente,

$$u(t) = \mathbf{HG} \begin{bmatrix} w(t+1) \\ \vdots \\ w(t+hp) \end{bmatrix} - (\mathbf{HG \cdot G'})(u(t-1) - u(t-2)) + u(t-1) - (\mathbf{HG \cdot F})\mathbf{y}(t) \quad (6.19)$$

6.2.2. Resultados.

Para comprobar la validez de la estrategia de control GPC programada, se realizaron varios experimentos sobre la planta real. Estos experimentos consistieron en la aplicación de referencias de tipo escalón y de perturbaciones sobre el flujo de aire, tal y como se hizo en secciones anteriores. Para averiguar el efecto de costes y horizontes en el control GPC, se configuraron diferentes casos:

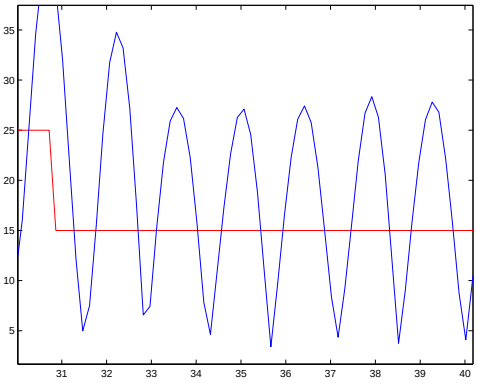
- Caso 1: $hp = 50$, $hc = 50$, $\delta = 1$, $\lambda = 100$.
- Caso 2: $hp = 50$, $hc = 50$, $\delta = 1$, $\lambda = 500$.
- Caso 3: $hp = 50$, $hc = 50$, $\delta = 1$, $\lambda = 1000$.
- Caso 4: $hp = 20$, $hc = 20$, $\delta = 1$, $\lambda = 500$.
- Caso 5: $hp = 20$, $hc = 20$, $\delta = 1$, $\lambda = 1000$.

En las figuras 6.1 se muestran las respuestas de los diferentes controladores ante referencias de tipo escalón hacia arriba y hacia abajo.

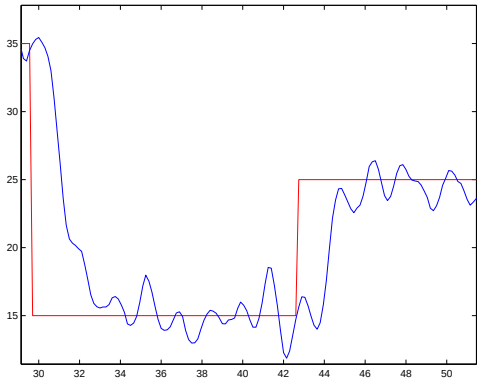
La sobreoscilación es mayor cuanto menor sea λ, puesto que, al penalizar menos el incremento de la señal de control, ésta varía más. Esto se ve claramente en el caso 1 (figura 6.1a). Si se alejan los horizontes la predicción y el control son más precisos, pero aumenta la carga de cálculo, por lo que se debe llegar a un valor de compromiso.

En las figuras 6.2 se representa el comportamiento de los diferentes controladores ante las perturbaciones en el flujo del aire provocadas por tapar con la mano la entrada de aire del soplador. Un horizonte menor perjudica la recuperación frente a perturbaciones. El controlador en el caso 2 es el que mejor se comporta frente a perturbaciones, porque tiene horizontes amplios pero no penaliza en exceso la variación de la señal de control.

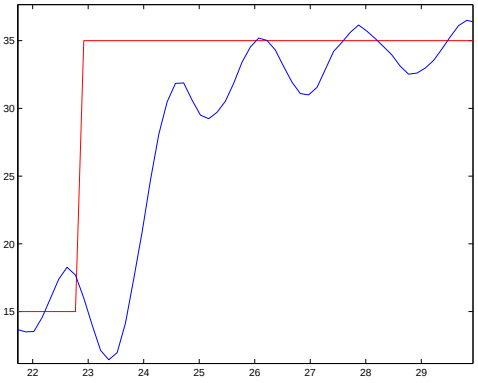
Se escoge finalmente la configuración del caso 2 para dejarla como automática en la aplicación de control local. El comportamiento del sistema controlado es bastante



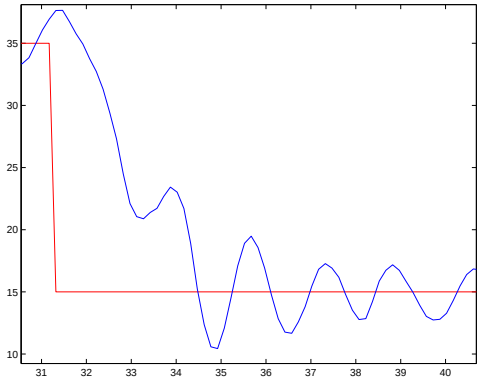
(a) Caso 1.



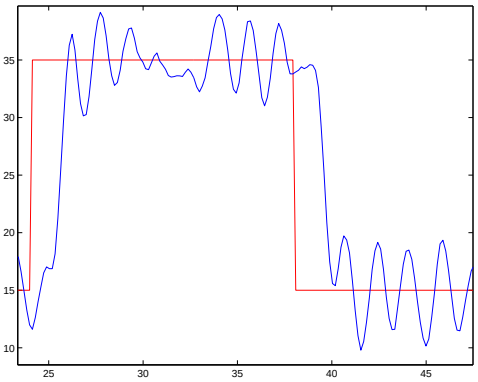
(b) Caso 2.



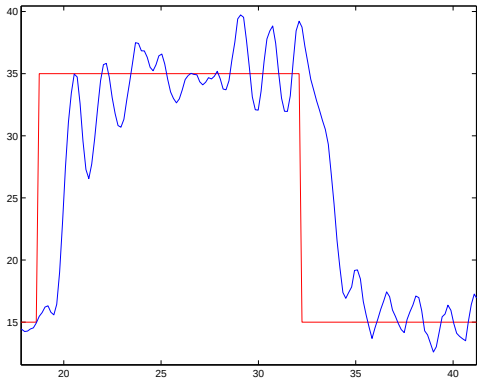
(c) Caso 3.



(d) Caso 3.

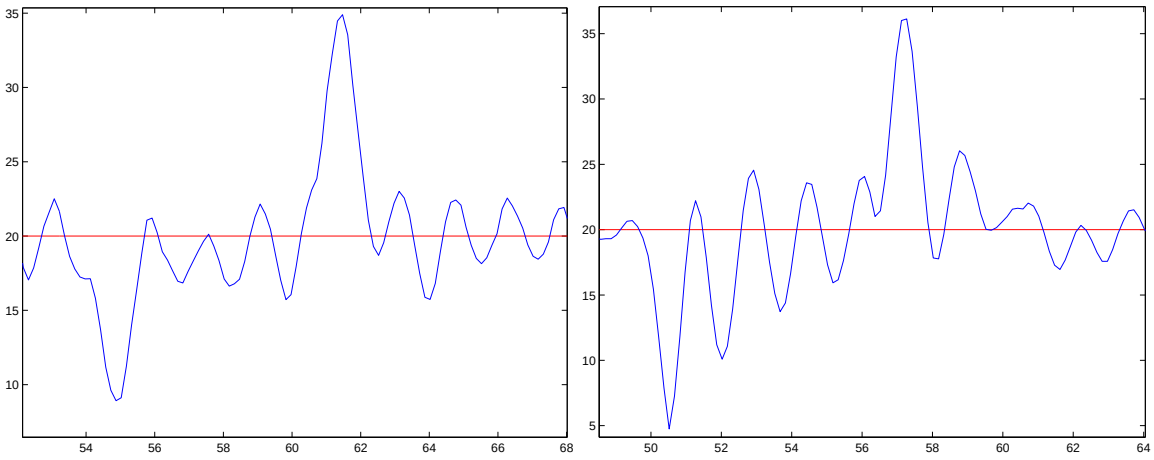


(e) Caso 4.

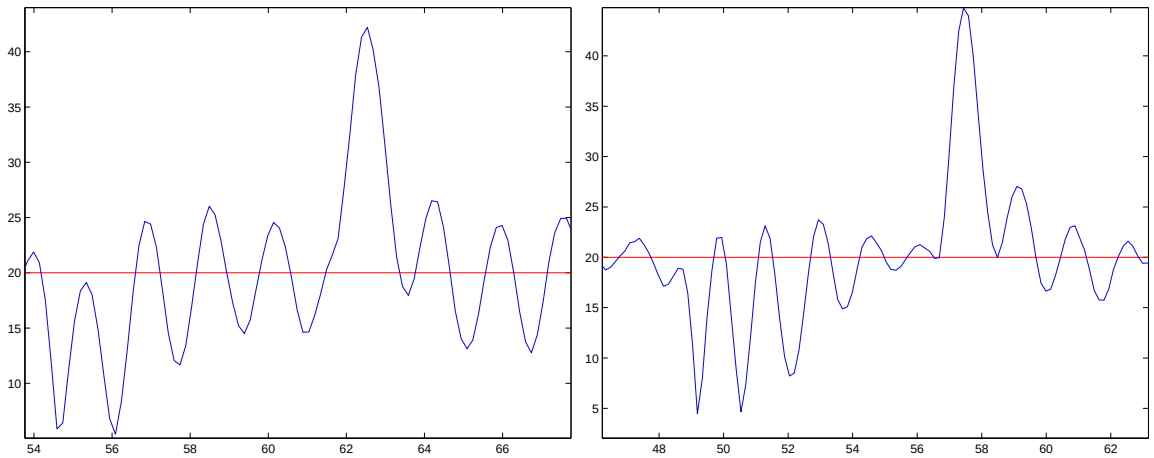


(f) Caso 5.

Figura 6.1: Respuesta ante escalones de los controladores GPC.



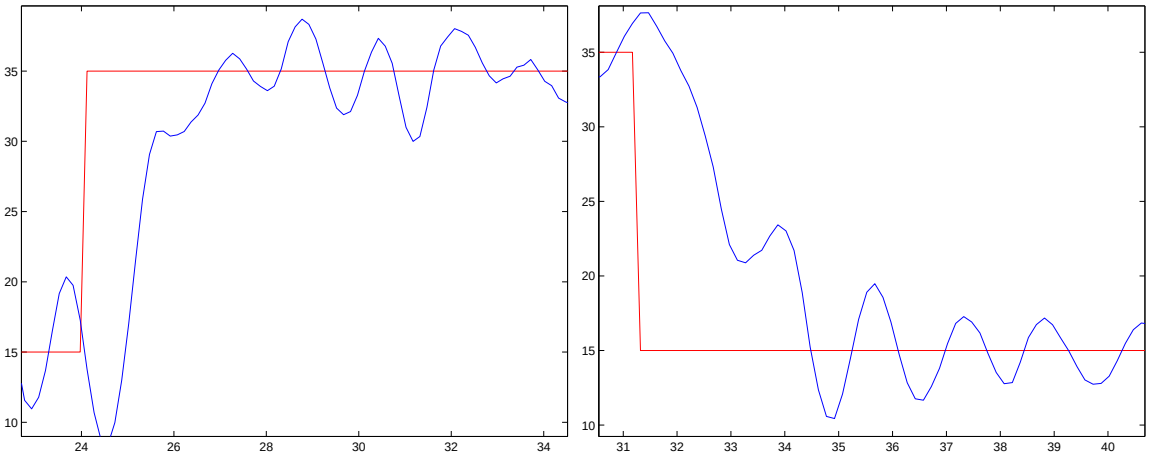
(a) Caso 2. (b) Caso 3.



(c) Caso 4. (d) Caso 5.

Figura 6.2: Respuesta ante perturbaciones de los controladores GPC.

bueno, con una sobreoscilación del 15 %, tiempo de subida de 1,5s y tiempo de establecimiento de 4s(figuras 6.3). La respuesta ante perturbaciones también es rápida como se ve en la figura 6.2a.



(a) Escalón hacia arriba. (b) Escalón hacia abajo.

Figura 6.3: Respuesta ante escalón del control GPC con la configuración del caso 2.

Capítulo 7

Control Borroso.

Muchas clases de objetos no están caracterizadas de forma exacta, es decir, sus límites o contornos no están definidos con precisión o, dicho de otra forma, son borrosos. En el caso de una clase con contornos borrosos, un objeto puede estar en una situación intermedia entre la plena pertenencia y la no pertenencia, o sea, puede tener un cierto grado de pertenencia a dicha clase. Una clase que admite la posibilidad de pertenencia parcial recibe la denominación de *conjunto borroso*.

La borrosidad es relevante en sistemas cuya complejidad, independientemente de la naturaleza del sistema, excede un cierto umbral a partir del cual resulta impracticable hacer afirmaciones precisas y significativas sobre el comportamiento del sistema.

> *En la medida en que las leyes matemáticas se refieren a la realidad, no son ciertas. Y, en la medida en que dichas leyes son ciertas, no se refieren a la realidad.*
>
> Albert Einstein.

> *A medida que la complejidad crece, las afirmaciones precisas pierden significado y las afirmaciones significativas pierden precisión.*
>
> Lofti A. Zadeh.

Este es el caso, precisamente, del sistema de cuyo control se ocupa este Proyecto. La complicada dinámica del aire y la inclusión del sistema de control, con sus retardos en la comunicación de los datos, producen un nivel de complejidad que, a veces, no permite estar seguros de las afirmaciones que se realizan en un momento dado. Por eso, se iniciará un estudio del control borroso del sistema con la misma filosofía que en capítulos anteriores: construir la base para un diseño posterior más profundo.

7.1. Teoría del Control Borroso.

El control convencional está basado en el modelo del proceso expresado en forma de ecuaciones diferenciales (o en diferencias). Por su parte, el control borroso se basa en el conocimiento heurístico del proceso. Algunas ventajas del control borroso son:

- Conceptualmente fácil de entender.
- Flexibilidad y tolerancia a la imprecisión de datos.
- Permite modelar funciones no lineales complejas.
- Descripción a partir del conocimiento y la intuición de expertos.
- Compatibilidad con el control convencional.

Notas sobre control Borroso: [MazyMar],[EspyBer], [Teo03], etc.

7.1.1. Definiciones.

Un *conjunto borroso* es un conjunto sin límites abruptos ni claramente definidos. El conjunto borroso está asociado a un cierto *valor lingüístico*, que es una palabra o etiqueta que puede asignarse a cierta variable x.

El margen de variación de la variable x se llama *universo de discurso*. Los universos de discurso son dominios precisos, normalmente en el conjunto de números reales.

La certeza o certidumbre con que a una variable x se le puede asignar un valor lingüístico i (es decir, que se puede incluir en el conjunto borroso asociado) se indica por la llamada *función de pertenencia* $\mu_i(x)$ ('membership function'). La función de pertenencia mapea la correspondencia entre el universo de discurso y el conjunto de números reales $[0, 1]$.

En las figuras 7.1 y 7.2 se muestran las fórmulas que definen posibles conjuntos borrosos.

La *base de conocimiento* es el conjunto de reglas que permiten inferir una conclusión para el control del proceso. Esta base de conocimiento deber ser completa (todas las posibles entradas del controlador debe tener su conclusión) y consistente (no puede haber conflicto entre conclusiones de diferentes reglas). Estas reglas son de la forma

	Gaussian membership functions
Left	$\mu^L(e) = \begin{cases} 1 & \text{if } e \le c^L \\ \exp\left(-\frac{1}{2}\left(\frac{e-c^L}{\sigma^L}\right)^2\right) & \text{otherwise} \end{cases}$
Centers	$\mu\ (e) = \exp\left(-\frac{1}{2}\left(\frac{e-c}{\sigma}\right)^2\right)$
Right	$\mu^R(e) = \begin{cases} \exp\left(-\frac{1}{2}\left(\frac{e-c^R}{\sigma^R}\right)^2\right) & \text{if } e \le c^R \\ 1 & \text{otherwise} \end{cases}$

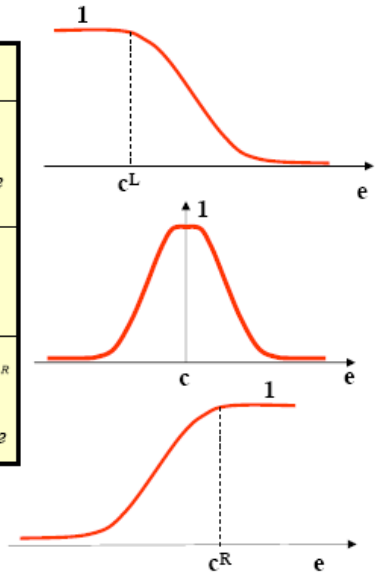


Figura 7.1: Funciones de pertenencia gaussianas.

	Triangular membership functions
Left	$\mu^L(e) = \begin{cases} 1 & \text{if } e \le c^L \\ \max\left\{0,\ 1+\frac{c^L-e}{0.5\cdot\omega^L}\right\} & \text{otherwise} \end{cases}$
Centers	$\mu^C(e) = \begin{cases} \max\left\{0,\ 1+\frac{e-c}{0.5\cdot\omega}\right\} & \text{if } e \le c \\ \max\left\{0,\ 1+\frac{c-e}{0.5\cdot\omega}\right\} & \text{otherwise} \end{cases}$
Right	$\mu^R(e) = \begin{cases} \max\left\{0,\ 1+\frac{e-c^R}{0.5\cdot\omega^R}\right\} & \text{if } e \le c^R \\ 1 & \text{otherwise} \end{cases}$

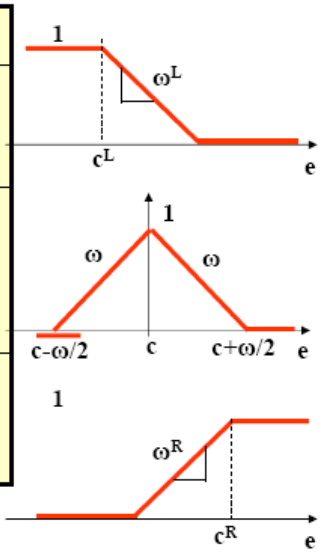


Figura 7.2: Funciones de pertenencia triangulares.

"SI se cumple un conjunto de condiciones ENTONCES se infiere un conjunto de consecuencias". Por tanto, la base de conocimiento consiste en un conjunto de reglas borrosas R_j de la forma:

$$R_j : \quad \text{SI } x_1 \text{ es } A_{1j} \text{ y } x_2 \text{ es } A_{2j} \text{ y } \ldots x_n \text{ es } A_{nj} \tag{7.1}$$
$$\text{ENTONCES } y \text{ es } B_j \tag{7.2}$$

7.1.2. Estructura del controlador borroso.

La lógica borrosa es un procedimiento de análisis con razonamiento aproximado que pretende modelar los modos de razonamiento impreciso del ser humano para tomar decisiones en entornos inciertos y sin precisión. El controlador con lógica borrosa debe automatizar el comportamiento que un experto humano utilizaría para alcanzar el objetivo. El proceso de control borroso se divide a grandes rasgos en tres fases: borrosificación, inferencia y desborrosificación.

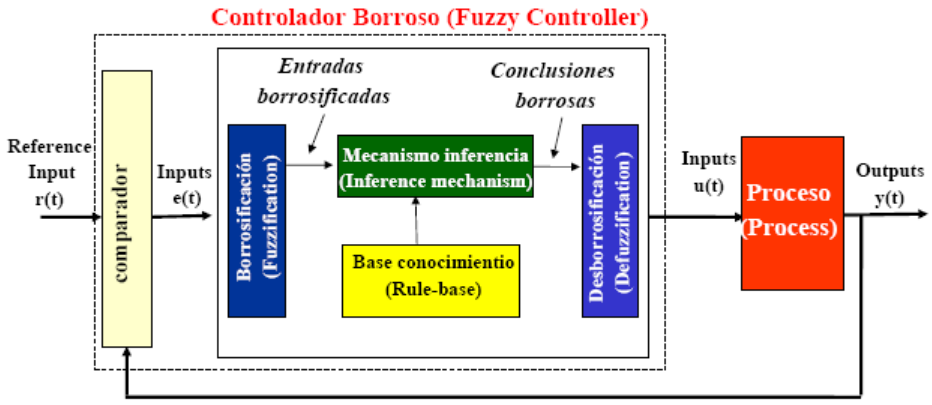


Figura 7.3: Mecanismo general de un control borroso.

Borrosificación.

En la fase de borrosificación ('fuzzyfication'), se determinan los valores lingüísticos que toman las variables de entrada.

Primero hay que definir las variables de entrada y salida y sus universos de discurso. En segundo lugar hay que definir todos los conjuntos borrosos y el valor lingüístico asociado a cada uno. En tercer lugar hay que definir una función de pertenencia para cada conjunto borroso.

El convertidor a borroso realiza la observación de la señal de entrada ordinaria al sistema ($u \in U \subset \mathbb{R}^n$) para incluirla en un conjunto borroso. El convertidor *singleton* es el más cómun cuyas funciones de pertenencia son rectángulos; convierte $x \in U$ en el conjunto borroso A_u en U, con $\mu_{A_u}(u) = 1$ y $\mu_{A_u}(u') = 0$ para todo $u' \in U$ tal que $u \neq u'$.

Inferencia.

El mecanismo de inferencia ('inference') consiste en determinar las relaciones que existen entre las variables de entrada y deducir conclusiones para la variable de salida interpretando la base de conocimiento ('rule-base') en función de dichas entradas.

Primero se determina el alcance de las reglas borrosas relevantes dada la situación de entrada (entrada u_i pertenece a conjunto A_i^m) con el operador:

$$\mu_i(u_1, u_2, \ldots, u_n) = \mu_{A_1^j}(u_1) * \mu_{A_2^k}(u_2) * \cdots * \mu_{A_n^l}(u_n) * \tag{7.3}$$

Luego se establecen las conclusiones en función de las entradas y de la base de conocimiento. Para esto se puede emplear el mecanismo de inferencia de Larsen (7.4) o el de Mandami (7.5).

Sea A_u un conjunto borroso en U, entonces cada regla R_j da lugar a un conjunto borroso $A_u \circ R_j$ en $\mathbb{R}$ a partir de las reglas de composición:

$$\begin{aligned}
\mu_{A_u \circ R_j}(y) &= \sup_{x \in U} \left[\mu_{A_u}(u) * \mu_{A_{1j} \times A_{2j} \times \ldots \times A_{nj} \times \rightarrow B_j}(x_1, x_2, \ldots, x_n, y)\right] \\
&= \sup_{x \in U} \left[\mu_{A_u}(u) * \mu_{A_{1j}}(x_1) * \mu_{A_{2j}}(x_2) * \ldots * \mu_{A_{nj}}(x_n) * \mu_{B_j}\right] \Rightarrow \\
\mu_{A_u \circ R_j}(y) &= \sup_{x \in U} \left[\mu_{A_u}(u) \mu_{A_{1j}}(x_1) \mu_{A_{2j}}(x_2) \ldots \mu_{A_{nj}}(x_n) \mu_{B_j}\right] \qquad (7.4)
\end{aligned}$$

o bien

$$\mu_{A_u \circ R_j}(y) = \sup_{x \in U} min \left[\mu_{A_u}(u) \mu_{A_{1j}}(x_1) \mu_{A_{2j}}(x_2) \ldots \mu_{A_{nj}}(x_n) \mu_{B_j}\right] \tag{7.5}$$

Desborrosificación.

La conversión a escalar ('defuzzyfication') asocia a la función de pertenencia de un conjunto borroso un valor concreto representativo de ese conjunto borroso, es decir, implementa una aplicación de conjuntos borrosos en R a un punto en R. Para realizar esta conversión existen fundamentalmente dos métodos: el de los centros ponderados y el de

los centroides. El método de los centros ponderados tiene la fórmula siguiente, suponiendo que se trata de convertir a escalar el conjunto borroso $A_u \circ R_j (j = 1, 2, \ldots, M)$:

$$y = \frac{\Sigma_{j=1}^{M} \bar{y}_j \mu_{A_u \circ R_j}(\bar{y})}{\Sigma_{j=1}^{M} \mu_{A_u \circ R_j}(\bar{y})} \tag{7.6}$$

en donde $\bar{y}_j$ es el punto de R en el que μ_{B_j} alcanza el valor máximo (normalmente el conjunto B_j está normalizado, por lo que $\mu_{B_j}(\bar{y}_j) = 1$.

7.2. Diseño para el Sistema Levineu.

7.2.1. Síntesis del controlador.

En un principio se analizó la posibilidad de realizar un control borroso con características PID, del cual se habían recibido buenas impresiones de otros diseñadores. Sin embargo, se decidió cambiar de estrategia para poder hacerla más general. Se deja como trabajo futuro la sintonización de un controlador borroso con características PID ([Jan98]).

En este Proyecto se desarrollará un control tipo Mandami con tres conjuntos borrosos por entrada y cinco para la salida. A continuación se explicará el diseño realizado.

Primeramente se define el objetivo que, obviamente, será el control en posición de la altura de una esfera suspendida sobre un flujo de aire.

Después, se definen las entradas (error en altura y derivada del error) y la salida (incremento de tension). Se eligen, asimismo, las variables lingüísticas y sus valores.

- Variables lingüísticas:
  - Error ($e[cm]$). Universo: $[-50, +50]cm$.
  - Derivada del error (de[cm/s]). Universo: $[-30, +30]cm/s$.
  - Incremento de tensión (u[V]). Universo: $[-10, +10]V$.
- Valores lingüísticos posibles:
  - Para las entradas: negativo (N), cero (C), positivo (P).
  - Para la salida: negativo grande (NG), negativo pequeño (NP), cero (CE), positivo pequeño (PP), positivo grande (PG).

A continuación se deduce la base de conocimiento a partir de la experiencia. En este caso, la base de conocimiento se compone de nueve reglas:

1. IF e es N AND de es N THEN u es PG.
2. IF e es N AND de es C THEN u es PP.
3. IF e es N AND de es P THEN u es CE.
4. IF e es C AND de es N THEN u es PP.
5. IF e es C AND de es C THEN u es CE.
6. IF e es C AND de es P THEN u es NP.
7. IF e es P AND de es N THEN u es CE.
8. IF e es P AND de es C THEN u es NP.
9. IF e es P AND de es P THEN u es NG.

A continuación se definen las funciones de pertenencia. Aquí se emplearán funciones de pertenencia triangulares y se definen de la forma expresada en la figura 7.2. Las funciones de pertenencia pueden ser definidas por el usuario mediante el panel de control del sistema.

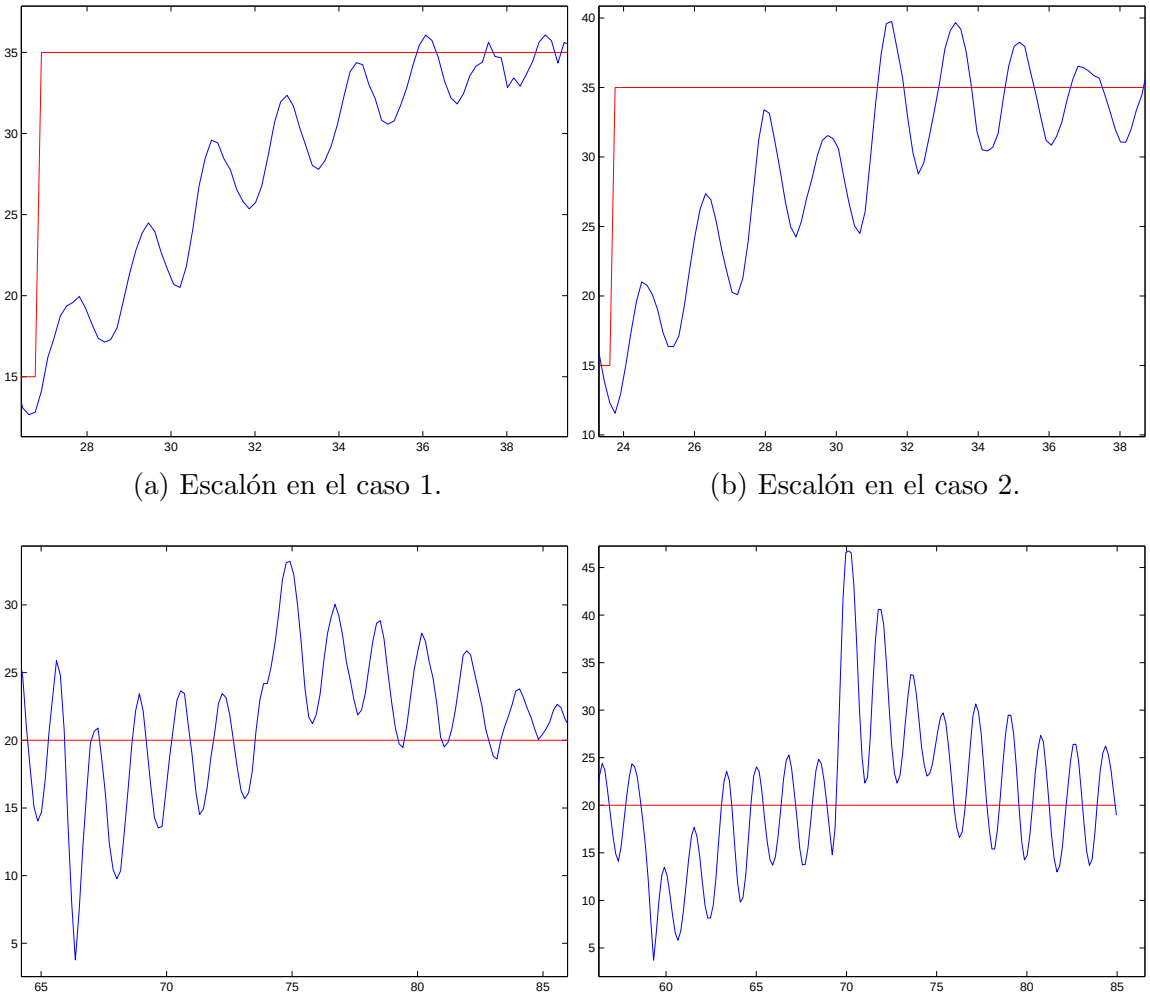
En la fase de borrosificación se llama a una bloque que determina el grado de pertenencia de las entradas a cada una de las funciones $\mu_i(x)$.

En el mecanismo de inferencia primero se determinan las reglas activas (esto es, con $\mu_i(x) \neq 0$) y se ponen a uno los bits asociados. Luego se evalua cada premisa mediante la operación lógica *e AND de* empleando el operador mínimo (inferencia de Mandami). Por último, la obtención de conclusiones se realiza por escalado.

La fase de desborrosificación se lleva a cabo mediante la técnica de centros ponderados.

7.2.2. Resultados.

Desgraciadamente, no se han podido conseguir unos resultados convincentes con la estrategia de control Borroso dada la dificultad de sintonización. No obstante, se presenta algún resultado más o menos aceptable (figura 7.4). Las respuestas a escalón no presenta sobreoscilación, aunque son bastante lentas. Se deja como futura línea de trabajo la sintonización correcta de esta estrategia de control no lineal o bien la renovación del diseño teniendo en cuenta la explicación detallada en [Jan98].



(a) Escalón en el caso 1.

(b) Escalón en el caso 2.

(c) Perturbación en el caso 1.

(d) Perturbación en el caso 2.

Figura 7.4: Respuestas con controladores Borrosos.

Las configuraciones de las funciones de pertenencia se detallan en la tabla 7.1 y las gráficas se muestran en la figura 7.5.

Entradas					
Para e	Caso 1	Caso 2	Para de	Caso 1	Caso 2
cLe	-12	-10	cLde	-15	-15
wLe	20	15	wLde	15	25
cCe	0	0	cCde	0	0
wCe	45	30	wCde	45	30
cRe	12	10	cRde	15	15
wRe	20	15	wRde	15	25

Salida		
Para u	Caso 1	Caso 2
cNGu	-0.5	-0.3
cNPu	-0.1	-0.1
cCEu	0	0
cPPu	0.1	0.1
cPGu	0.5	0.3

Tabla 7.1: Configuración de los parámetros de las funciones de pertenencia para el caso 1 del control Borroso.

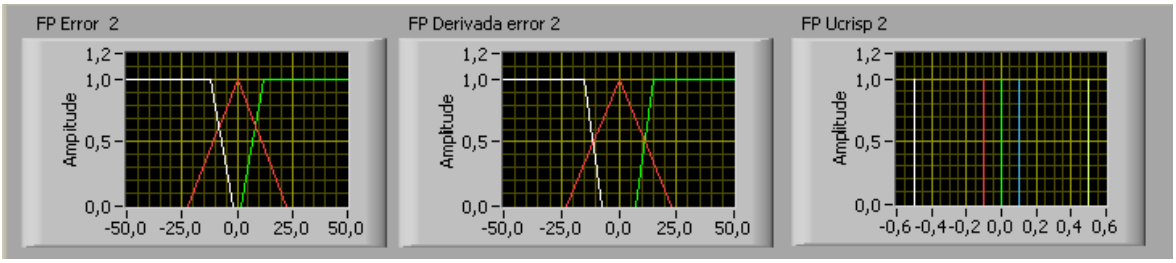


Figura 7.5: Funciones de pertenencia para el caso 1 del control Borroso.

Capítulo 8

Programas desarrollados.

El software presentado en este PFC forma parte del conjunto de programas desarrollado durante el Proyecto Levineu. Para más información se remite al lector a [Cap05] y [Per05].

En este Proyecto se utilizaron principalmente tres potentes aplicaciones de ingeniería: CX–Programmer (software de programación de autómatas Omron), LabView (software de interfaz de experimentos a través del PC), Matlab (lenguaje y entorno interactivo de programación técnica).

8.1. Programas en LabView.

NI LabVIEW es un entorno de desarrollo gráfico para crear de forma rápida y con bajo coste aplicaciones flexibles y escalables de prueba, medida y control. LabView permite interactuar con las señales del mundo real, analizar datos y compartir resultados y aplicaciones, todo ello a través del PC. LabView facilita el diseño de interfaces gráficas preparadas para todo tipo de usuarios.

El programa desarrollado como interfaz de control para el "Sistema Levineu"se denomina *LevineuLocal.vi*. Como todos los archivos LabView consta de dos partes: el panel de control ('Control Panel'), que aparece como interfaz al usuario, y el diagrama de bloques ('Block Diagram'), donde el programador diseña el funcionamiento del proceso.

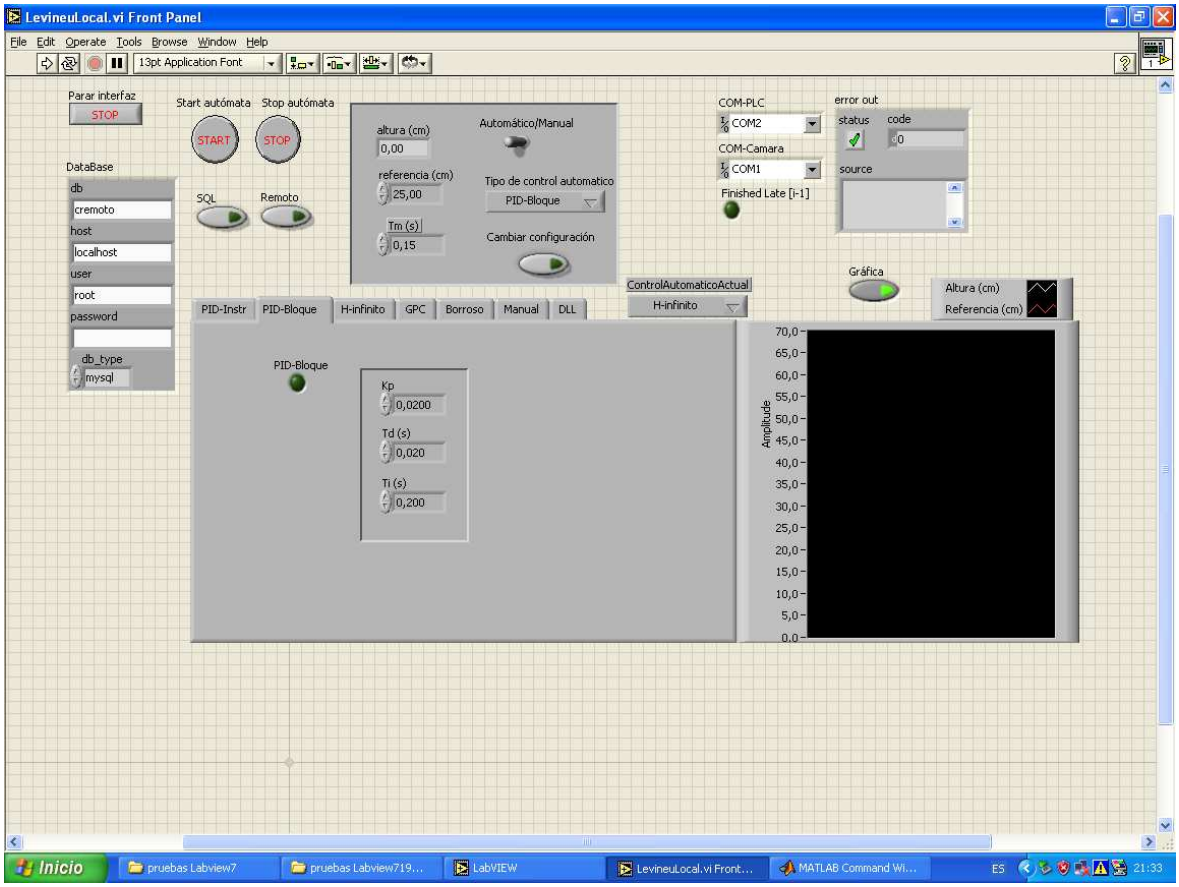


Figura 8.1: Vista inicial del panel de control de la aplicación.

Panel de Control.

El Panel de Control diseñado se ve en la figura 8.1.

En la parte superior izquierda hay situados unos botones que gobiernan el funcionamiento del sistema. El botón denominado *Parar interfaz* detiene el funcionamiento de la interfaz y desactiva previamente el procesamiento del autómata. Los botones *Start autómata* y *Stop autómata* activan y desactivan respectivamente el procesamiento del autómata.

Debajo de los anteriores aparecen dos botones para especificar el proceso del control. El botón *Remoto* apagado indica que el control del sistema se realiza desde el PC local, mientras que si el botón está encendido, el control del sistema es gobernado desde un PC remoto. Por su parte, el botón *SQL* activa o desactiva la actualización de la base de datos. En el caso de utilizar la base de datos (obligatorio si el control del sistema es remoto), el formulario denominado *Database* indica la configuración de dicha base de datos.

En la parte superior derecha se agrupan variables acerca de la comunicación entre los aparatos: se puede elegir entre los puertos serie que hay habilitados (COM1 y COM2). Hay un indicador de error de comunicación (*error out*) y otro que alerta de que se ha superado el tiempo de muestreo seleccionado (*Finished Late [i+1]*).

En la parte superior central se agrupan las variables generales del control. Aquí el sistema indica la altura actual de la bola y se permite al usuario definir la referencia y el tiempo de muestreo (para éste último, el procesamiento del autómata debe estar desactivado). Además se puede seleccionar el modo de control entre Automático y Manual. Dentro del modo Automático, se debe escoger una estrategia de control entre PID-Instr, PID-Bloque, H-infinito, GPC, Borroso y DLL. Para dar el visto bueno y ejecutar un cambio en el tipo de control o en la configuración del tipo ya seleccionado debe pulsarse sobre el botón *Cambiar configuración*.

En el centro del Panel de Control hay un contenedor tipo carpeta clasificadora; en cada pestaña se definen los parámetros de uno de los controles. Cuando es elegido un control, su página correspondiente aparece superpuesta sobre las otras; además, se enciende un LED en dicha página que indica que está activo. Para saber qué control automático está actuando o estaba seleccionado antes de elegir el control Manual se tiene un indicador denominado *ControlAutomáticoActual*. Cuando está seleccionado el control Borroso, en la parte inferior aparecen unas gráficas con las funciones de ponderación.

En el centro a la derecha se dibuja la gráfica de la altura y la referencia a lo largo del tiempo. Esta gráfica se desplaza manteniendo visible los últimos 100 ciclos de muestreo.

Como ya se comentó en la sección 1.3, uno de los objetivos de este Proyecto era preparar una plataforma software que permitiera el estudio y la profundización de las estrategias de control. Este objetivo se cree cumplimentado mediante esta aplicación en LabView, pues permite al usuario participar activamente en el diseño de los controladores.

Existen dos modos de diseño de las estrategias de control: *automático y de usuario*. Cada control tiene una sintonización por defecto, que se ejecutará si está seleccionado el modo automático. A lo largo de este Proyecto se ha indicado qué valores se le ha dado a los parámetros en cada caso. Por otra parte, gracias al panel de control en Labview, el usuario puede cambiar la configuración de los controladores variando algunos parámetros.

En los controles PID, el usuario puede cambiar en cualquier momento la ganancia proporcional, el tiempo derivativo o el tiempo integral (figura 8.2). Tras poner los valores deseados en sus casillas y tener seleccionado tal control, hay que pulsar sobre el botón *Cambiar configuración*.

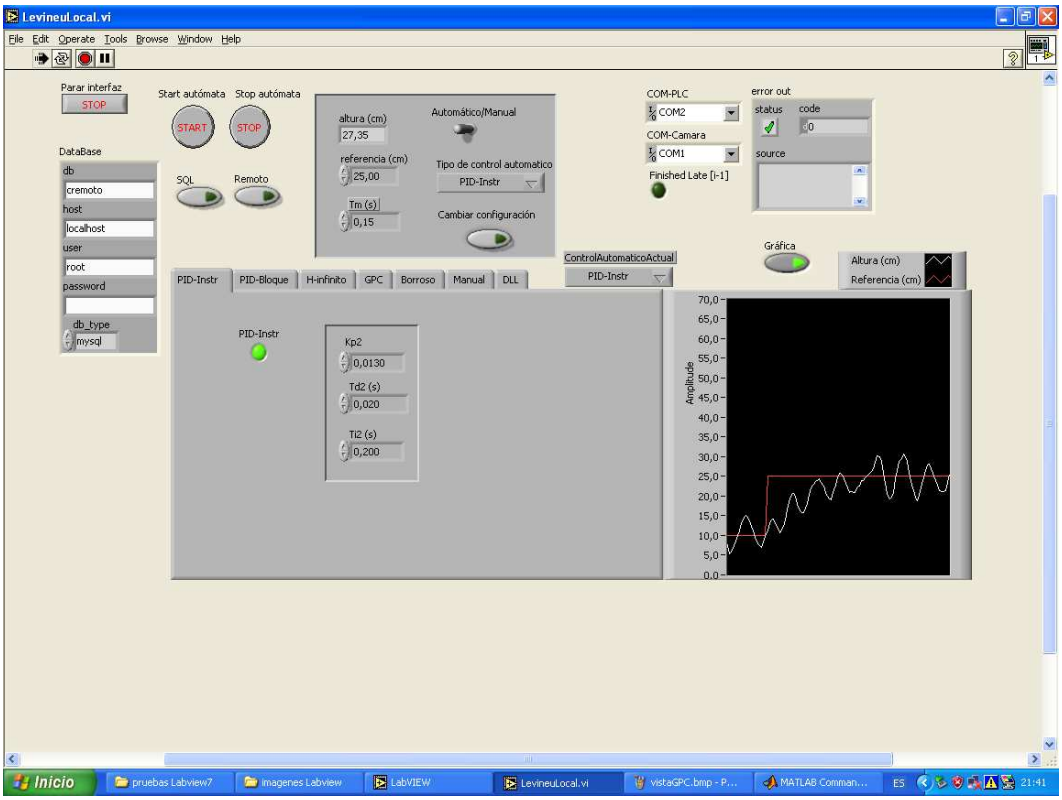


Figura 8.2: Vista del panel de control ejecutándose el control PID con instrucción de librería.

En el control H_∞ hay un indicador llamado *Hinf_Auto/Usu* que puede ponerse a 1 para seleccionar el modo de usuario (ver figura 8.3). En el modo automático, la función de ponderación de la sensibilidad complementaria $W_T(s)$ está predefinida pero el usuario puede modificar la función de ponderación de la sensibilidad $W_S(s)$ variando el parámetro κ. Sin embargo, si está seleccionado el modo de usuario, entonces se puede participar también en el diseño de $W_T(s)$ mediante los parámetros KWt y $tauWt$. Por otra parte, se indica cuál es la función de transferencia en discreto del controlador H_∞ programado.

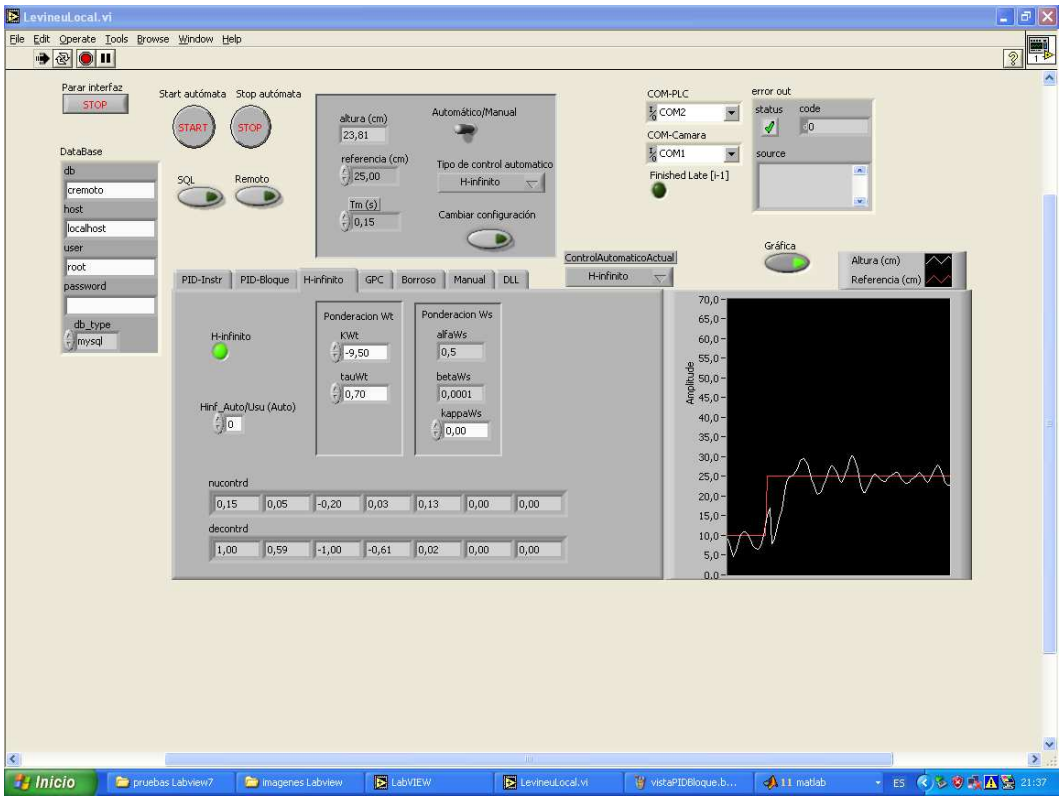


Figura 8.3: Vista del panel de control ejecutándose el control H_∞.

En la pestaña del control GPC está el indicador *GPC_Auto/Usu* con la misma función que para el control H_∞. En este caso, cuando el modo es automático, el controlador GPC está definido como se indicó en la sección 6.2. En cambio, en el modo de usuario pueden elegirse los horizontes y los costes (figura 8.4). En futuras versiones se permitirá también que el usuario cambie la planta nominal en discreto. Además, como el cálculo de la señal de control con GPC se realiza en Matlab, se puede mostrar su resultado en *ukGPC*.

En el control Borroso se permite al usuario definir con completa libertad las funciones de pertenencia de los conjuntos borrosos. Su pestaña en el contenedor tipo carpeta del panel de control se puede ver en la figura 8.5 Para cada una de las en-

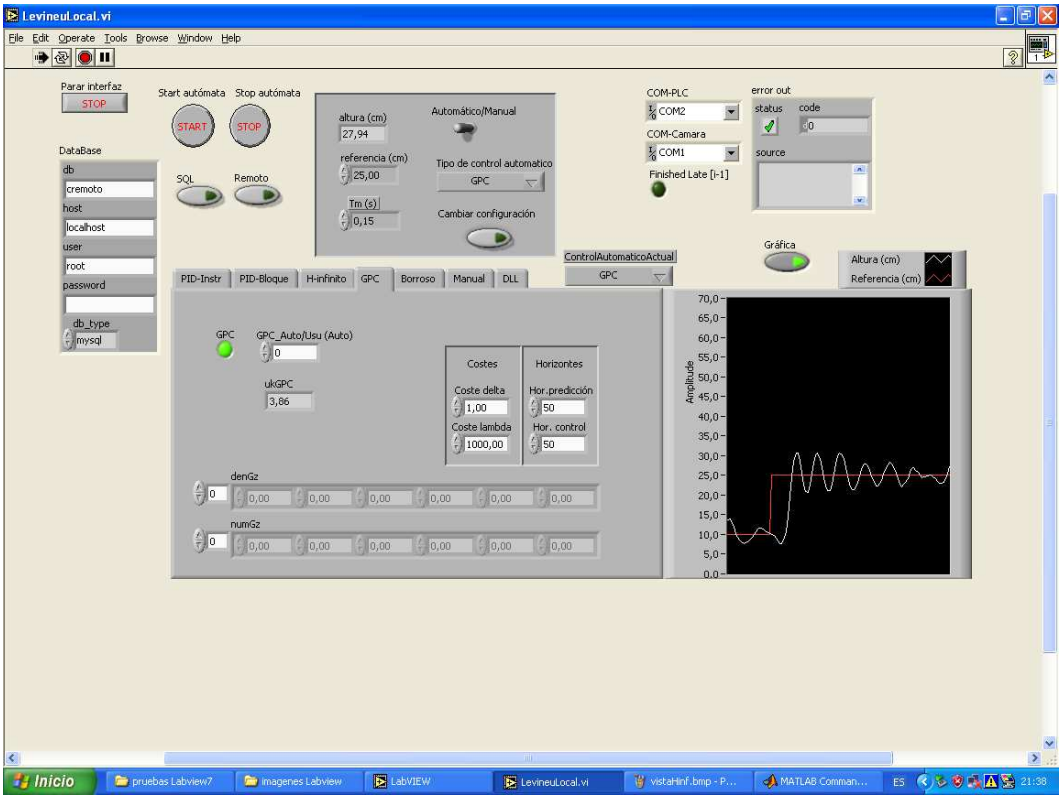


Figura 8.4: Vista del panel de control ejecutándose el control GPC.

tradas (error y derivada del error) hay tres funciones de pertenencia triangulares que se pueden definir mediante dos parámetros: centro (c) y pendiente (w), como se muestra en la figura 7.2. Los nombres de los parámetros relacionados con el error terminan por 'e' y los relacionados con la derivada del error terminan por 'de'. Las funciones de pertenencia tipo LEFT se denotan por la mayúscula 'L', las de tipo CENTER se denotan por la mayúscula 'C' y las de tipo RIGHT, por la mayúscula 'R'. Para la salida (incremento de tensión -'u') existen cinco funciones de pertenencia tipo singleton que se definen por su centro ('c'). Los cinco 'singletones' son: negativo-grande (NG), negativo-pequeño (NP), cero (CE), positivo-pequeño (PP), positivo-grande (PG).

Cuando está ejecutándose el control Borroso las funciones de pertenencia aparecen dibujadas en la parte inferior del panel de control. Por problemas de programación debe pulsarse dos o tres veces en el botón *Cambia configuración* para que se vean reflejados los posibles cambios en estas gráficas.

Por último, en el modo de control manual, el usuario puede seleccionar la tensión de salida del variador con un deslizador (figura 8.6).

La secuencia que debería llevar a cabo un usuario del Sistema Levineu con esta aplicación sería la siguiente:

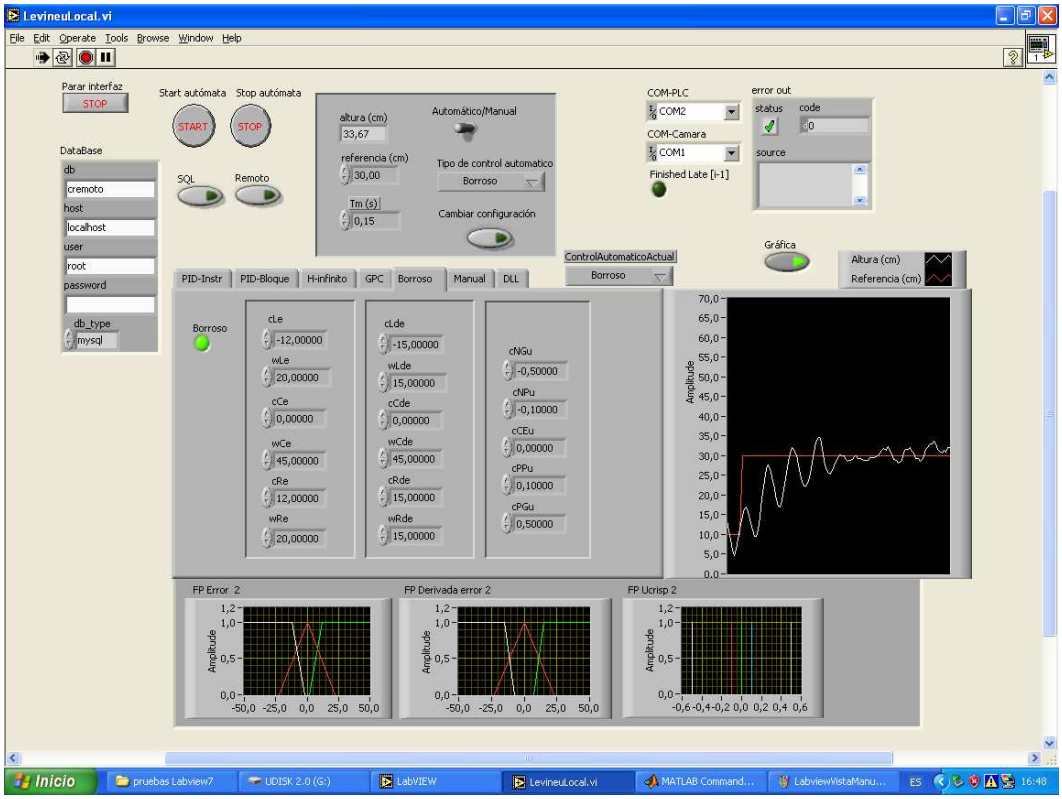


Figura 8.5: Vista del panel de control ejecutándose el control Borroso.

1. Encender los interruptores 'hardware' del Sistema. Encender el PC.

2. Abrir el fichero *LevineuLocal.vi*. Activar el Run del LabView si no ha aparecido en dicho modo.

3. Indicar los puertos serie para la conexión PC-PLC (con el menú *COM-PLC*) y para la conexión PC-Cámara (con el menú *COM-Cámara*).

4. Escoger el origen del control como local desactivando el botón *Remoto*. Habilitar la actualización de la base de datos SQL mediante el botón *SQL*. En caso afirmativo, indicar los parámetros de dicha base de datos.

5. Determinar el modo de control automático o manual con *Automático/Manual*.

6. Seleccionar el tipo de controlador automático con el menú desplegable *Tipo de control automático*.

7. Fijar una referencia para la altura con el recuadro de control *referencia (cm)*.

8. Ajustar los parámetros del controlador elegido en su pestaña correspondiente del contenedor tipo carpeta. Comprobar el modo de diseño en los casos en que haya restricciones en la configuración (H_∞, GPC).

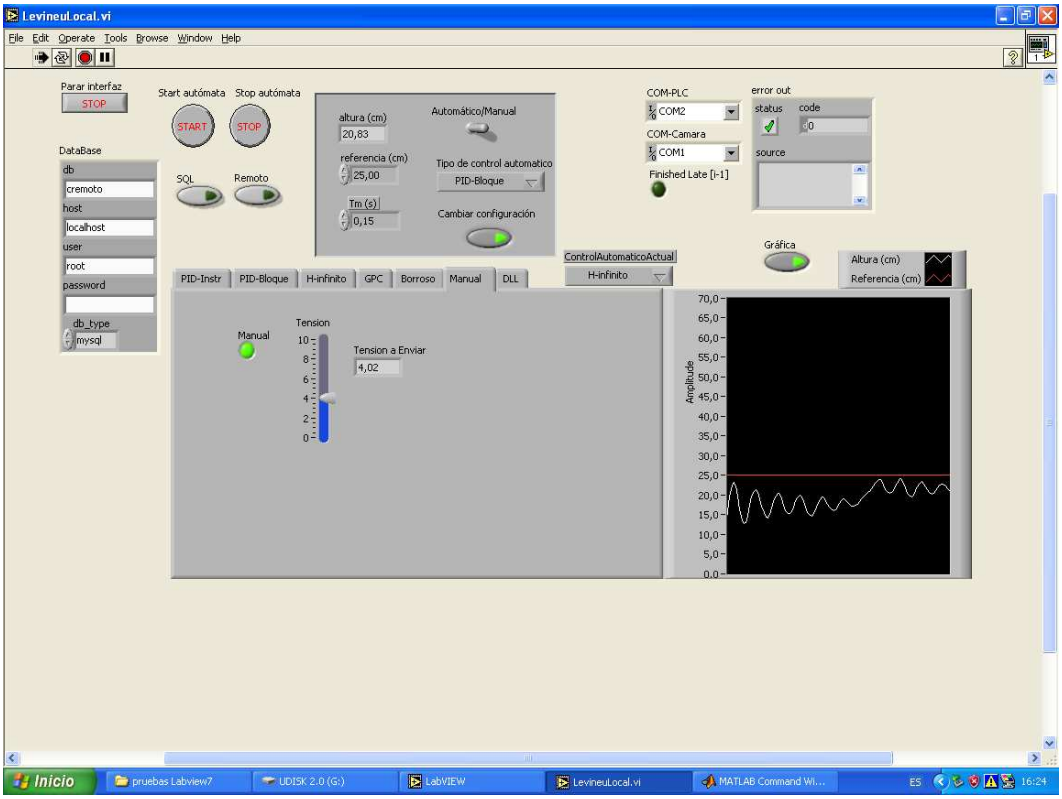


Figura 8.6: Vista del panel de control ejecutándose el control Manual.

9. Si el procesamiento del autómata estaba desactivado, presionar *Start autómata* para dar la orden de inicio. Si el procesamiento del autómata estaba activado pulsar *Cambiar configuración*.

10. Activar la gráfica mediante el botón *Gráfica*.

11. Tras realizar el experimento, pulsar el botón *Stop autómata* para detener el procesamiento del autómata. Aparecerá una ventana para poder guardar el fichero de datos resultado del experimento.

12. Para abandonar la aplicación pulsar el botón *Parar interfaz*.

Diagrama de Bloques.

El Diagrama de Bloques se compone de diversos bucles 'while' anidados, algunos de los cuales incluyen estructuras secuenciales o condicionales.

El bucle superior contiene una estructura secuencial que se ejecuta mientras esté activa la interfaz. En la escena inicial (número 0) hay un bucle en el que se espera a la

activación del procesamiento del sistema de forma local (con *Start autómata*) o remota (atendiendo a la base de datos SQL). En la escena número 1 se configuran los puertos serie que comunicarán el PC local con la cámara y con el PLC. En la escena número 2 se ordena al PLC comenzar el bucle del programa y se inicializan algunas variables.

La escena 4 lleva a cabo el bucle de control propiamente dicho. Contiene un bucle 'while' temporizado que se ejecuta cada tiempo de muestreo hasta que se ordena detener el autómata o la interfaz. Este bucle tiene anidado otra estructura secuencial. En la escena 0 de esta secuencia (a partir de ahora 4-0) se toma el dato *altura* desde la cámara llamando a un bloque 'subvi' denominado *camara.vi*; además, en el caso de haber seleccionado la actualización de la base de datos, se revisa ésta para tomar valores de parámetros si se está en control remoto o escribirlos si se está en control local. En las escenas 4-1, 4-2 y 4-3, se actualiza el tipo de control automático, se prepara el *ControlWord* y se monta la trama 1 con la palabra de control, la altura y la referencia. En la escena 4-4 se monta la trama 2 teniendo en cuenta si el ciclo es de configuración. Cuando es un ciclo de configuración se monta la segunda trama con cualquier tipo de control automático. El proceso es el siguiente: se toman los valores de los parámetros convenientes, se llama a un bloque 'subvi' denominado *Param_hl.vi* para convertir los parámetros a cadenas de 10 caracteres (9 que dan el valor en formato científico más un terminador '00'), y se concatenan las cadenas resultantes. Además, en los casos de control automático H_∞, GPC o Borroso, se llama previamente a bloques 'subvi' que determinan los parámetros a enviar.

Si el ciclo no es de configuración sólo se debe montar la segunda trama si el control es manual, automático con GPC o a través de una DLL. En estos casos en cada ciclo debe formarse la segunda trama con dos parámetros: el tiempo de muestreo y la señal de control.

En la escena 4-5 y 4-6 se montan las tramas Host Link llamando a un bloque 'subvi' denominado *TramaHL.vi*. En la escena 4-7 se envían las tramas Host Link al autómata. En la escena 4-8 se dibuja la gráfica y, en su caso, se actualiza la base de datos.

Cuando se ordena acabar con el proceso de control del sistema, se guardan los datos de tiempo, referencia y altura en un fichero. Posteriormente se ordena al PLC abandonar el bucle del programa de control y se cierran los puertos.

De esta forma se acaba la estructura secuencial principal y se vuelve al principio del bucle general, donde se espera de nuevo la orden de comienzo de un nuevo experimento.

8.2. Programas en CX-Programmer.

CX-Programmer es un entorno gráfico de configuración, desarrollo y depuración de aplicaciones para autómatas Omron. CX-Programmer se ejecuta en Windows y con él se pueden programar todos los tipos de autómatas Omron, desde micro-PLCs hasta las series más potentes como CJ/CS,proporcionando toda la potencia de programación que se necesita, incluso para el desarrollo de sistemas muy complejos y con múltiples dispositivos. Actualmente los lenguajes soportados compatibles con IEC-61131-3 son IL, LD, ST, FB. En este Proyecto se emplearon LD y FB.

El funcionamiento del CX-Programmer está explicado en la profusa documentación que Omron proporciona(en inglés y español) incluida en el CD adjunto ([Omr04a] y otros).

El archivo que guarda el programa diseñado se titula *LevineuLocal.cxp* y consta de un proyecto de trabajo denominado "ProyectoOmron", que incluye un único PLC llamado"Levineu". Este PLC es el modelo CJ1M–CPU11 de la marca Omron Electronics; tiene configuración modular y, además de la CPU–11, posee un módulo de alimentación PA202 y un módulo de salida analógica DA021. Dicho PLC controla el sistema Levineu basándose en la programación que tiene almacenada. Esta programación consiste en una selección de la configuración del autómata, programas de instrucciones llamados tareas ('tasks'), una tabla de entradas y salidas, tablas de símbolos (etiquetas para direcciones de memoria), áreas de memoria y objetos programables llamados bloques de función ('function blocks'). Esta programación es la que se diseña a través del CX-Programmer.

Las posiciones de memoria del PLC pueden ser etiquetadas mediante símbolos globales o locales a una sección. La tabla de símbolos globales está en el apendice A.

La tabla de entrada/salida tiene la siguiente estructura:

- PLC-CPU.
  - Bastidor.[Numeración]
    - Módulo.[Numeración](Analógico-A/Digital-D)(Dirección por unidad) (Número de unidad)(Canal de salida; canal de entrada)

En el caso del Sistema Levineu, la tabla de E/S es la siguiente:

- PLC-CPU: CJ1M-CPU11.
  - Bastidor principal [2000].
    - Módulo: Unidad ASCII SIOU/C200H (A)(1)(0)(Salida:9; Entrada:1).

El control del sistema evoluciona acorde con la serie de programas o tareas desarrollados y posteriormente almacenados en el PLC. Cada uno de estos programas o tareas están compuestos a su vez por una tabla de símbolos locales, diversas secciones de instrucciones y una última sección de fin. Los programas se ejecutan cíclicamente y en cada ciclo se sigue el orden ascendente de su numeración.

La lista de tareas programadas es la siguiente:

1. PrimerPaso (00): encendido del sistema.
2. Inicialización (01): preparación del ciclo.
3. PIDInstr (02): cálculo de un control PID mediante la instrucción del autómata.
4. PIDBloque (03): cálculo de un control PID mediante un bloque de función.
5. Manual (04): control manual de la salida.
6. Borroso (05): cálculo de un control borroso mediante un bloque de función.
7. Hinf (06): cálculo de un control H-infinito mediante un bloque de función.
8. GPC (07): cálculo de un control GPC mediante un bloque de función.
9. ActuacionComun (08): aplicación de la señal de control previamente calculada y actualización de variables.

La explicación detallada de las tareas programadas y las listas de símbolos se encuentra en el apéndice A.

8.3. Programas en Matlab.

Para este Proyecto se han programado los siguientes 'scripts' en Matlab:

- Control clásico: *EspecificacionControl.m, TablaRouth.m, ZieglerNicholsBA.m, ZieglerNicholsBC.m* o *EstudioSinCompensar.m*.
- Control H_∞: *HinfLevineu.m*, *PlantasLevineu.m*, *CalculaSVIncer_SISO.m* y *DibujaFuncionesBC.m*.
- Control GPC: *GPCLevineu.m*, *PlantasLevineu.m*, *DivisionPolinomica.m* y *ExtraeMatrices.m*.
- Control Borroso: *Borroso3entradas.m* y *Borroso5entradas.m*.

En el Sistema Levineu se hace uso de Matlab para sintetizar el controlador H_∞ y para realizar el cálculo de matrices y de la señal de control en GPC. En el caso del control H_∞, se llama desde LabView al 'script' *HinfLevineu.m* tras un cambio de configuración. Este 'script' llama a su vez a los otros tres. En el caso del control GPC, se llama desde LabView al 'script' *GPCLevineu.m* en cada ciclo de muestreo. Este 'script' llama a su vez a los otros tres.

El código fuente de los programas escritos en Matlab se encuentra en B. Se ha empleado entre otros el 'toolbox' "μ-Analysis and Synthesis"([Bal01]).

Capítulo 9

Conclusiones.

En este capítulo final se presentan de forma resumida las contribuciones aportadas por este Proyecto, así como las futuras vías de trabajo que pueden surgir del presente Proyecto.

9.1. Conclusiones del control.

A lo largo de la realización del Proyecto se han puesto de manifiesto las carencias del Sistema Levineu y las dificultades para un control preciso de la altura.

La mayor fuente de problemas es sin duda la dinámica aérea que es, precisamente, lo que se pretende controlar en este Proyecto. El flujo de aire no tiene la misma velocidad y densidad en todos los puntos del eje vertical del movimiento. Cuanto más lejos de la boca del soplador, o sea, a mayor altura, el aire sufre mayor dispersión y su velocidad es menor. Parte de este problema podría solucionarse colocando un cilindro transparente en torno al eje vertical del flujo; sin embargo, no se ha procedido a su colocación porque se consideró que sería alterar el proceso en una situación de laboratorio que lo diferenciaría de la más compleja realidad.

Otro de los problemas es la imprecisión del actuador. El motor soplador de que se dispone está diseñado con otros objetivos industriales y no entra entre sus características la precisión en la frecuencia del giro. Debido a la fricción de las aspas y a la inercia, la aceleración no es instantánea y esto implica que la señal de control que el PLC transmite al variador de velocidad en un momento dado no es la señal alcanzada por el motor actuador en dicho instante. Esta diferencia tan grande entre las diferentes señales (señal calculada en el PLC, señal enviada al variador por la salida analógi-

ca, señal transmitida del variador al soplador y frecuencia real de giro) añade mayor complejidad al control además de la inevitable de la aerodinámica.

Otro problema en el control es el retardo en las comunicaciones. Por un lado, la cámara tarda un mínimo de $60ms$ en calcular el dato de la altura y pasárselo al PC. Luego éste debe transmitírselo al autómata a través del puerto serie lo cual implica unos $60ms$ por cada trama Host Link a una velocidad de 9600baudios (se recuerda que en cada ciclo hay 1 ó 2 tramas Host Link). Por lo tanto, en los controles que se calculan en el autómata se tiene como salida actual un dato de hace $120ms$. Esto perjudica mucho a las estrategias de control programadas en el PLC. Se puede corregir aumentando la velocidad del puerto serie, pero la mejor solución es modificar el esquema de control conectando directamente el PLC con la cámara. Actualmente sólo se dispone de una salida serie en el autómata, que necesariamente debe estar conectada al PC, por lo que, para la última solución, debería adquirirse un módulo de comunicación serie para el PLC.

Ya se comentó al principio de este Proyecto (ver sección 1.3), que no se pretendía profundizar en la sintonización de las estrategias de control, sino preparar el avance posterior en cada una de ellas, mediante la recopilación teórica y la programación de la aplicación software. Por todo ello, no se pueden sacar conclusiones concluyentes de la comparación entre controladores.

No obstante, a partir de los experimentos realizados, se pueden destacar algunas características de las diversas estrategias de control. Conforme aumenta la complejidad, mejora el resultado, aunque a costa de un incremento del coste del equipo de control.

Control PID.

El control PID es la solución más sencilla. Consigue corregir la sobreoscilación y eliminar los errores en régimen permanente. Las reglas de Ziegler-Nichols nos dan una idea aproximada del orden de los parámetros de control, pero para concretar tales magnitudes se debe acudir los esquemas y diagramas que representan las características especiales de la planta (Routh-Hurwitz, Bode, Nyquist, Nichols, lugar de las raíces).

Este controlador tiene que utilizarse con un término derivativo muy bajo debido a la gran amplificación de ruido. Esto provoca que el PID pierda capacidad de sintonización. Sin embargo, colocando un filtro paso baja antes del derivador podría evitarse este problema.

Además, este controlador produce saturaciones en el motor cuando se siguen referencias con variaciones bruscas como es el caso de un escalón. La respuesta ante perturbaciones es manifiestamente mejorable.

Control H_∞.

Se han diseñado controladores H_∞ mediante el planteamiento de sensibilidad mixta. Se ha utilizado una metodología de síntesis casi automatizada, que incluye un diseño de funciones de ponderación muy sencillo pero efectivo. Esta metodología permite aplicar esta técnica de control sin necesidad de conocimientos teóricos elevados, aunque sí es conveniente tener un amplio conocimiento del sistema controlado.

Control GPC.

El control GPC ha dado muy buenos resultados en la planta real. Consigue una error en régimen permanente nulo y su comportamiento es invariable sea cual sea el tipo de referencia. El movimiento de la carga, ante cambios bruscos en la señal de referencia, se realiza de forma suave. El controlador actúa antes de que se produzca un cambio de orientación, para frenar o acelerar antes de que la inercia lo aleje de la trayectoria deseada. Posee un buen comportamiento ante perturbaciones.

Esta estrategia de control podría mejorarse mucho imponiendo restricciones o añadiendo a la función de costes términos relativos a la velocidad o al consumo de la señal de control. También puede variarse ligeramente la técnica para convertirla en otra estrategia de su familia MPC.

Control borroso.

Debido a la falta de tiempo, no ha podido sacarse ninguna conclusión acerca del control borroso. La sintonización es bastante compleja dada la cantidad de parámetros existentes, y actualmente es bastante mejorable. Una aproximación podría hacerse a partir de la explicación en [Jan98] y en [Esc04]. Se deja por tanto la base para poder estudiar mejor en un futuro este control no lineal.

9.2. Contribuciones del Proyecto.

En general se ha programado un software para desarrollar y aplicar diversas estrategias de control sobre un sistema de levitación neumática.

Se ha intentado avanzar en el estudio del control en tiempo real de un sistema no lineal de gran complejidad, que involucra aspectos tan heterogéneos y difíciles como la aerodinámica, la comunicación de datos y la integración de varias aplicaciones comerciales muy potentes.

Se ha desarrollado una amplia variedad de estrategias de control lineales y no lineales. Se ha diseñado una interfaz manejable para configurar todas las técnicas de control implementadas. Se han sentado los cimientos para la profundización posterior en cada una de las estrategias de control.

Se ha investigado la programación de un autómata de marca Omron. Además, se ha preparado un instrumento de valor didáctico que podrá ser utilizado en diversas asignaturas sobre Ingeniería de Control.

9.3. Líneas futuras de trabajo.

Este Proyecto se deja como punto de partida para profundizar en el estudio de los diversos controladores investigados. Así pues, para dar por terminado este Proyecto, se apuntan posibles líneas para continuar el trabajo desarrollado en él.

- Mejora de la sintonización de cada uno de los controladores presentados.
- Aplicación práctica de teorías más complejas en cada uno de las estrategias de control
    1. PID: utilización de métodos de ajuste distintos, introducción de integrador anti-windup.
    2. H_∞: desarrollo de nuevas formas de modelado y de ponderación; síntesis en discreto.
    3. GPC: aplicación de restricciones y nuevos algoritmos predictivos.
    4. Borroso: uso de operadores alternativos y construcción de nuevas funciones de pertenencia.
- Aprovechamiento de la planta construida en el laboratorio para trabajar en problemas relacionados con la saturación.

- Modificación de la configuración del sistema para permitir en lo posible que el autómata sea el eje central del proceso de control y dedicar el ordenador a tareas de monitorización.
- Ampliar los módulos del autómata para permitir la comunicación bidireccional PC-PLC. Durante este Proyecto se ha puesto de manifiesto la necesidad de contar con un módulo de entradas-salidas digitales, un módulo de comunicación serie y/o un módulo de comunicación Ethernet.
- Estudio de los problemas derivados del procesado de datos y de los retardos en la comunicación. Reducción de dichos retardos.

APÉNDICES

Apéndice A

Configuración y programación del autómata.

A.1. Configuración y listas de símbolos.

La tabla de entrada/salida tiene la siguiente estructura:

- PLC-CPU.
  - Bastidor.[Numeración]
    - Módulo.[Numeración](Analógico-A/Digital-D)(Dirección por unidad) (Número de unidad)(Canal de salida; canal de entrada)

En el caso del Sistema Levineu, la tabla de E/S es la siguiente:

- PLC-CPU: CJ1M-CPU11.
  - Bastidor principal [2000].
    - Módulo: Unidad ASCII SIOU/C200H (A)(1)(0)(Salida:9; Entrada:1).

La lista de símbolos globales se despliega en las tablas A.1 a A.5 y los símbolos locales se enumeran en A.6.

Nombre	Tipo de dato	Dirección	Comentario
Start	BOOL	10.00	Activación general
Stop	BOOL	10.01	Parada general
on_off	BOOL	11.00	Indicador de estado general del sistema.
SalidaAnalog	UINT	2001	Salida de tensión (valor escalado y en hexadecimal)
TmREAL	REAL	D30	Variable flotante auxiliar para el tiempo de muestreo.
TmUINT	UINT	D32	Tiempo de muestreo para los temporizadores en entero sin signo.
TmBCD	UINT_BCD	D34	Tiempo de muestreo para los temporizadores en el tipo que se pasa a TIMH.
TensionNom	REAL	D40	Tensión de equilibrio nominal.
u_min	REAL	D42	Tensión de control mínima.
u_max	REAL	D44	Tensión de control máxima.
NumReal1	REAL	D50	Número 1 en flotante.
NumReal10	REAL	D52	Número 10 en flotante.
NumReal100	REAL	D54	Número 100 en flotante.
NumReal400	REAL	D56	Número 400 en flotante.
ukPos	REAL	D60	Tensión de control calculada en la fase de posicionamiento.
errorTension	REAL	D62	Error entre la tensión actual y la nominal al principio del posicionamiento.
contadorPos1	UINT	D64	Contador de posicionamiento hasta alcanzar la tensión nominal.
contadorPos2	UINT	D66	Contador para mantener el sistema un tiempo en la tensión nominal.
ukdef	REAL	D100	Señal de control definitiva que se manda al variador de velocidad.
error	REAL	D110	Error en altura en el ciclo actual.
error1	REAL	D112	Error en altura en el último ciclo.
error2	REAL	D114	Error en altura en el penúltimo ciclo.
SalAnalog_PID	UINT	D251	Salida analógica del PID
ukPID	REAL	D300	Señal de control calculada en PIDBloque para el ciclo actual.
uk1PID	REAL	D302	Señal de control en el ciclo anterior con PIDBloque.
uk2PID	REAL	D304	Señal de control en el penúltimo ciclo con el PIDBloque.
ekPID	REAL	D310	Error actual para el PID.
ek1PID	REAL	D312	Error anterior para el PID.
ek2PID	REAL	D314	Error hace dos ciclos para el PID.

Tabla A.1: Lista de símbolos globales del programa *LevineuLocal.vi* (I).

Nombre	Tipo de dato	Dirección	Comentario
perdida	REAL	D320	Error cometido por el actuador por saturación: diferencia entre la señal de control calculada por el PID-Bloque y la definitiva (todo en el ciclo anterior).
ukMan	REAL	D400	Señal de tensión en control manual. Se aplica directamente sobre la salida en Actuación.
ukBor	REAL	D500	Señal de control calculada con control Borroso.
uk1Bor	REAL	D502	Señal de control en el ciclo anterior con Borroso.
ekBor	REAL	D504	Error en altura en el ciclo actual con Borroso.
ek1Bor	REAL	D506	Error en altura en el ciclo anterior con Borroso.
mu_max_u	REAL	D508	Máximo valor de una función de pertenencia.
de	REAL	D510	Derivada del error en Borroso.
mu_e_N	REAL	D512	Valor de la función de pertenencia 'negativo' del error.
u_de_N	REAL	D514	Valor de la función de pertenencia 'negativo' de la derivada del error.
mu_e_C	REAL	D516	Valor de la función de pertenencia 'cero' del error.
mu_de_C	REAL	D518	Valor de la función de pertenencia 'cero' de la derivada del error.
mu_e_P	REAL	D520	Valor de la función de pertenencia 'positivo' del error.
mu_de_P	REAL	D522	Valor de la función de pertenencia 'positivo' de la derivada del error.
CambioTension	REAL	D524	Valor de la salida del control Borroso.
TotalRegla	REAL	D526	Denominador de la ecuación de desborrosificación.
PonderaRegla	REAL	D528	Numerador de la ecuación de desborrosificación.
Premisa11	REAL	D530	
Premisa12	REAL	D532	
Premisa13	REAL	D534	
Premisa21	REAL	D536	
Premisa22	REAL	D538	
Premisa23	REAL	D540	
Premisa31	REAL	D542	
Premisa32	REAL	D544	
Premisa33	REAL	D546	

Tabla A.2: Lista de símbolos globales del programa *LevineuLocal.vi* (II).

Nombre	Tipo de dato	Dirección	Comentario
ukHoo	REAL	D600	Señal de control en H-infinito.
uk1Hoo	REAL	D602	
uk2Hoo	REAL	D604	
uk3Hoo	REAL	D606	
uk4Hoo	REAL	D608	
uk5Hoo	REAL	D610	
uk6Hoo	REAL	D612	
uk7Hoo	REAL	D614	
ekHoo	REAL	D620	Error en el ciclo actual con H-infinito
ek1Hoo	REAL	D622	
ek2Hoo	REAL	D624	
ek3Hoo	REAL	D626	
ek4Hoo	REAL	D628	
ek5Hoo	REAL	D630	
ek6Hoo	REAL	D632	
ek7Hoo	REAL	D634	
b0Hoo	REAL	D640	
b1Hoo	REAL	D642	
b2Hoo	REAL	D644	
b3Hoo	REAL	D646	
b4Hoo	REAL	D648	
b5Hoo	REAL	D650	
b6Hoo	REAL	D652	
b7Hoo	REAL	D654	
a0Hoo	REAL	D660	
a1Hoo	REAL	D662	
a2Hoo	REAL	D664	
a3Hoo	REAL	D666	
a4Hoo	REAL	D668	
a5Hoo	REAL	D670	
a6Hoo	REAL	D672	
a7Hoo	REAL	D674	
ukGPC	REAL	D710	Señal de control calculada con GPC.
ukGPC1	REAL	D712	Señal de control en el ciclo anterior con GPC.
ukGPC2	REAL	D714	Señal de control en el penúltimo ciclo con GPC.
canalControlWord	WORD	D1001	Canal de recepción de la palabra de control
canalAltura	UINT	D1002	Canal de recepción de la altura mediante HL
canalReferencia	UINT	D1007	Canal de recepción de la referencia mediante HL

Tabla A.3: Lista de símbolos globales del programa *LevineuLocal.vi* (III).

Nombre	Tipo de dato	Dirección	Comentario
canalP1	UINT	D1012	Canal de recepción del parámetro 1 mediante HL
canalP2	UINT	D1017	Canal de recepción del parámetro 2 mediante HL
canalP3	UINT	D1022	Canal de recepción del parámetro 3 mediante HL
canalP4	UINT	D1027	Canal de recepción del parámetro 4 mediante HL
canalP5	UINT	D1032	Canal de recepción del parámetro 5
canalP6	UINT	D1037	Canal de recepción del parámetro 6
canalP7	UINT	D1042	Canal de recepción del parámetro 7
canalP8	UINT	D1047	Canal de recepción del parámetro 8
canalP9	UINT	D1052	Canal de recepción del parámetro 9
canalP10	UINT	D1057	Canal de recepción del parámetro 10
canalP11	UINT	D1062	Canal de recepción del parámetro 11
canalP12	UINT	D1067	Canal de recepción del parámetro 12
canalP13	UINT	D1072	Canal de recepción del parámetro 13
canalP14	UINT	D1077	Canal de recepción del parámetro 14
canalP15	UINT	D1082	Canal de recepción del parámetro 15
canalP16	UINT	D1087	Canal de recepción del parámetro 16
canalP17	UINT	D1092	Canal de recepción del parámetro 17
canalP18	UINT	D1097	Canal de recepción del parámetro 18
altura	REAL	D1202	Altura actual
referencia	REAL	D1204	Referencia de altura
P1	REAL	D1206	Parámetro 1
P2	REAL	D1208	Parámetro 2
P3	REAL	D1210	Parámetro 3
P4	REAL	D1212	Parámetro 4
P5	REAL	D1214	Parámetro 5
P6	REAL	D1216	Parámetro 6
P7	REAL	D1218	Parámetro 7
P8	REAL	D1220	Parámetro 8
P9	REAL	D1222	Parámetro 9
P10	REAL	D1224	Parámetro 10
P11	REAL	D1226	Parámetro 11
P12	REAL	D1228	Parámetro 12
P13	REAL	D1230	Parámetro 13
P14	REAL	D1232	Parámetro 14
P15	REAL	D1234	Parámetro 15
P16	REAL	D1236	Parámetro 16
P17	REAL	D1238	Parámetro 17
P18	REAL	D1240	Parámetro 18

Tabla A.4: Lista de símbolos globales del programa *LevineuLocal.vi* (IV).

Nombre	Tipo de dato	Dirección	Comentario
ControlWord	WORD	W1	Palabra de control
cicloConfig	BOOL	W1.00	Activo cuando el ciclo es de configuración del control
controlManual	BOOL	W1.01	Activo cuando el control es manual
bit1SelControl	BOOL	W1.02	Bit 1 de la selección del control.
bit2SelControl	BOOL	W1.03	Bit 2 de la selección del control.
bit3SelControl	BOOL	W1.04	Bit 3 de la selección del control.
manual_a_auto	BOOL	W4.00	Activo cuando se ha pasado de manual a automático.
CambioControl	BOOL	W4.01	Activo cuando se ha cambiado de control.
Posicionado	BOOL	W4.02	Activo cuando se ha acabado la fase de posicionamiento en el punto de equilibrio nominal.
ControlUltimo	UINT	W5	Número de control en el último ciclo.
ControlActual	UINT	W6	Número del control actual.
on_PIDInstr	BOOL	W7.00	Indicador del estado del control PID basado en la instrucción PID del autómata.
on_PIDBloque	BOOL	W7.01	Indicador del estado del control PID basado en el bloque de función.
on_Manual	BOOL	W7.02	Estado del control manual.
on_Borroso	BOOL	W7.03	Indicador del estado del control Borroso
on_Hinf	BOOL	W7.04	Indicador del estado del control Hinf.
on_GPC	BOOL	W7.05	Estado del control GPC.
errorPIDPropio	BOOL	W8.01	
errorBorroso	BOOL	W8.03	
errorHinf	BOOL	W8.04	
errorGPC	BOOL	W8.05	
bitPremisa11	BOOL	W59.00	
bitPremisa12	BOOL	W59.01	
bitPremisa13	BOOL	W59.02	
bitPremisa21	BOOL	W59.03	
bitPremisa22	BOOL	W59.04	
bitPremisa23	BOOL	W59.05	
bitPremisa31	BOOL	W59.06	
bitPremisa32	BOOL	W59.07	
bitPremisa33	BOOL	W59.08	

Tabla A.5: Lista de símbolos globales del programa *LevineuLocal.vi* (y V).

Nombre	Tipo	Dirección	Tarea	Comentario
cambiosmanauto	INT	W12	Inicialización	Número de cambios de manual a automático.
Habilitación	WORD	2000	Inicialización	Palabra de habilitación de la salida analógica.
Habilitado	BOOL	2000.00	Inicialización	Activo cuando se ha habilitado ya la salida analógica
nuevoTmPos	BOOL	T0001	Inicialización	Flag para calcular cada Tm
numcambios	INT	W10	Inicialización	Número de cambios de control.
on_SelControl	BOOL	W3.00	Inicialización	Activo cuando es un ciclo de configuración del control.
RefPID	UINT	D200	PIDInstr	Referencia para el bloque PID
PBPID	UINT	D201	PIDInstr	Banda proporcional adaptada para instrucción PID.
TiPID	UINT	D202	PIDInstr	Tiempo integral adaptado para instrucción PID.
TdPID	UINT	D203	PIDInstr	Tiempo derivativo adaptado para instrucción PID.
TmPID	UINT	D204	PIDInstr	Tiempo de muestreo adaptado para instrucción PID
OpIpPID	UINT	D205	PIDInstr	Especificador de operación y coeficiente de filtro de entrada
RangoPID	UINT	D206	PIDInstr	Rango de entrada y salida
LimInf	UINT	D207	PIDInstr	Límite inferior de la variable de salida manipulada
LimSup	UINT	D208	PIDInstr	Límite superior de la variable de salida manipulada
EntAnalogPID	UINT	D250	PIDInstr	Entrada analógica al PID
SalAnalogPID	UINT	D251	PIDInstr	Salida analógica del PID
auxREAL	REAL	D252	PIDInstr	Variable flotante auxiliar
auxINT	INT	D254	PIDInstr	Variable auxiliar de 16bits
altINT	INT	D255	PIDInstr	Altura en INT
refINT	INT	D256	PIDInstr	Referencia en INT
PropBand	REAL	D257	PIDInstr	Banda proporcional en flotante (100/Kp en flotante)
TdINTPID	INT	D260	PIDInstr	Tiempo derivativo en entero de 16bit(décimas de segundo)
TiINTPID	INT	D261	PIDInstr	Tiempo integral en entero de 16bits (décimas de segundo)
TmINTPID	INT	D262	PIDInstr	Tm en entero de 16bit
nuevoTm3	BOOL	T0003	PIDBloque	Flag para calcular cada Tm
nuevoTm5	BOOL	T0005	Borroso	Flag para calcular cada Tm
nuevoTm6	BOOL	T0006	Hinf	Flag para calcular cada Tm
nuevoTm7	BOOL	T0007	GPC	Flag para calcular cada Tm
nuevoTm8	BOOL	T0008	ActuacionComun	Flag para calcular cada Tm

Tabla A.6: Lista de símbolos locales del programa *LevineuLocal.vi*.

A.2. Explicación detallada de los programas.

La lista de tareas programadas es la siguiente:

1. PrimerPaso (00): encendido del sistema.
2. Inicialización (01): preparación del ciclo.
3. PIDInstr (02): cálculo de un control PID mediante la instrucción del autómata.
4. PIDBloque (03): cálculo de un control PID mediante un bloque de función.
5. Manual (04): control manual de la salida.
6. Borroso (05): cálculo de un control borroso mediante un bloque de función.
7. Hinf (06): cálculo de un control H-infinito mediante un bloque de función.
8. GPC (07): cálculo de un control GPC mediante un bloque de función.
9. ActuacionComun (08): aplicación de la señal de control previamente calculada y actualización de variables.

Primer Paso.

Al principio se ejecuta el programa *PrimerPaso*, con el número 00. En esta tarea se comprueba el estado (encendido o apagado)del sistema. Con las señales de entrada *Start* y *Stop* se controla el indicador de estado *on_off* a través de la instrucción KEEP, que actúa como un relé de enclavamiento. Cuando *on_off*=1 se permite la ejecución de las tareas *Inicialización* y *ActuaciónComún* con TKON. Tras leer las líneas de entrada *Start* y *Stop*, se resetean con RSET. Como no se dispone de un módulo de entrada-salida digital, en esta aplicación se escribirá directamente sobre el bit *on_off* mediante los botones START y STOP de la interfaz en LabView.

Inicialización.

El siguiente programa se llama *Inicialización* y tiene el número 01.

La primera sección se llama *Activación*. Realiza la inicialización de variables y constantes y habilita la salida analógica. Con FLT se convierte un entero con signo a flotante; con MOV se mueve una palabra. Al final se deshabilitan las tareas de cálculo de la señal de control con TKOF.

Si *on_off* es TRUE pero el bit *habilitado* es FALSE, significa que se acaba de encender el sistema pero que la salida analógica aún no ha sido habilitada. En este caso, se habilita la salida analógica del PLC (módulo DA202) poniendo un 1 en la palabra *Habilitacion*. La salida analógica depende del valor almacenado en la palabra CIO2001 (cuyo símbolo es *SalidaAnalog*).

La segunda sección se llama *SelControl*. Interpreta la palabra de control. Si el ciclo actual es de configuración del control (*ciclo_config* = 1), se activa *on_SelControl*. La selección del control se realiza comprobando el estado de los bits de selección del control de la palabra *controlword*. Se permite la ejecución de la tarea correspondiente al control elegido con TKON. Además se pone el número de la tarea del control elegido en la palabra *ControlActual* y se activa el indicador correspondiente al control. Posteriormente se comprueba si hubo cambio en el control, si se está en una fase de posicionamiento en equilibrio nominal o si hubo paso de manual a automático comparando *ControlActual* y *ControlUltimo*.

La tercera sección se llama *AdaptaParam* y lleva a cabo la adaptación de los parámetros que provienen del LabView. Los parámetros del control han llegado desde el PC mediante HostLink como una cadena de 10 caracteres (9 que dan el valor más un terminador '00'). Estas cadenas de caracteres han sido almacenadas en las posiciones indicadas por los símbolos que empiezan con *canal_* ... Con la instrucción FVAL, se lee cada grupo de nueve caracteres que indica el valor del parámetro en formato científico (exponencial) y se convierte este valor a un número real en coma flotante de precisión simple. Estos números reales ya podrán ser usados posteriormente en cada control y se almacenan en las posiciones determinadas por los símbolos globales *altura*, *referencia*, *P1*, *P2*, ...

La altura y la referencia deben actualizarse continuamente. El resto de parámetros sólo se actualizan en un ciclo de configuración del control, excepto los dos primeros, que también se actualizan cuando el control es manual o GPC. El parámetro P1 es siempre el tiempo de muestreo. El resto de parámetros depende del control seleccionado.

La cuarta sección se llama *PosEquilibrio*. Si se está en una fase de posicionamiento en equilibrio nominal (*Posicionado*=0), se ejecuta el bloque de función *IniciaPos*. Este bloque lleva el objeto hasta una posición nominal de partida cuando hay un cambio de control. Se calcula una curva suave hasta alcanzar la tensión de equilibrio (*TensionNom*) y luego se mantiene un pequeño lapso de tiempo. Esta fase de posicionamiento no se ejecuta con control manual.

Resumiendo: en cada ciclo, el programa *Inicialización* activa la tarea de control cuando haya sido elegida, y prepara los parámetros necesarios. El segundo programa en ejecutarse en el ciclo debe ser, por tanto, el que desarrolle el control escogido.

Control PIDInstr.

El siguiente programa se denomina *PIDInstr* y tiene el número 02. Esta tarea se ejecuta si ha sido elegido el control PID basado en la instrucción PID del autómata, de modo que ha sido activada en la tarea *Inicialización* y, por tanto, *on_PIDInstr* = 1.

Esta tarea se compone de una sola sección denominada *CalculaPIDInstr*. Antes de llamar a la instrucción PID, deben ser almacenados sus operandos en posiciones de memoria adecuadas. En primer lugar se adaptan los parámetros llegados desde el PC. La altura y la referencia se actualizan en todos los ciclos, mientras que las constantes del control (Kp,Ti,Td) se actualizan cuando hay un ciclo de configuración del control (*ciclo_config* = 1).

Todos los datos se encuentran en posiciones etiquetadas con símbolos globales pero están almacenados en tipo REAL y deben ser adaptados a tipo UINT antes de ser pasados a la instrucción PID. La altura y la referencia están almacenadas en centímetros, por lo que son multiplicadas por 10 para ponerse en milímetros (instrucción *F, el símbolo *NumReal10* almacena el número 10 en coma flotante de precisión simple); luego se pasan a enteros con signo mediante FIX (se queda con la parte entera del operando flotante); por último, son pasados a los canales que tomará la instrucción PID (*EntAnalogPID* y *RefPID*).

La ganancia proporcional (Kp) debe ser pasada a banda proporcional ($BP = 100/Kp$) y se almacena en *PropBand* (instrucción /F, *NumReal1* y *NumReal100* indican dónde están los valores 1 y 100 en real); luego se pasa a entero con FIX y se mueve al canal que tomará PID (*PBPID*). El tiempo derivativo y el tiempo integral llegan en segundos pero deben ser pasados a la instrucción PID como el número de veces que contienen el tiempo de muestreo o 100ms (se elige como unidad 100ms). Por tanto hay que dividir entre 0.1s, lo que es igual que multiplicar por 10 (como con la altura). Luego se pasan a entero con FIX y se mueven a los canales que tomará el PID (*TdPID* y *TiPID*). El tiempo de muestreo llega en segundos pero debe pasarse a la instrucción PID en décimas de segundo, por lo que habría que dividirlo entre 0.1 lo que es lo mismo que multiplicar por 10; luego se pasa a entero con FIX y se mueve al canal que tomará PID (*Tm_PID*).

Tras todo esto hay que indicar en cada ciclo de configuración del control otras especificaciones a la instrucción PID.

- Especificador de operación y coeficiente del filtro de entrada: se pone ♯2 en *OpIpPID*.

- Rango de entrada y salida: se pone ♯1294 en *RangoPID*. Rango de entrada hasta $03FFhex = 1023dec$; rango de salida hasta $0FFFhex = 4095dec$.

- Límite inferior de la variable manipulada: se pone $\sharp 0000 = 0dec$ en *LimInf.*

- Limite superior de la variable manipulada: se pone $\sharp 0FA0 = 4000dec$ en *LimSup.*

Luego se llama a la instrucción PID con los siguientes operandos:

- Primer operando: EntAnalogPID, donde se encuentra la altura actual.

- Segundo operando: RefPID, primer canal de los parámetros de control.

- Tercer operando: SalAnalogPID, donde se saca la salida analógica del PID.

Control PIDBloque.

El siguiente programa se denomina *PIDBloque* y tiene el número 03. Esta tarea se ejecuta si ha sido elegido el control PID basado en el bloque de función diseñado por el autor, de modo que ha sido activada en la tarea *Inicialización* y, por tanto, *on_PIDBloque* = 1.

Para calcular el tiempo de muestreo se prepara un temporizador de alta velocidad mediante la instrucción TIMH. TIMH funciona como un temporizador decremental en centésimas de segundo. Un temporizador se activa cuando su condición de ejecución es ON, y se resetea al valor seleccionado cuando la condición de ejecución se pone en OFF. Una vez activado, la función TIMH decrementa el valor actual en unidades de 0,01 segundos. Si la condición de ejecución permanece en ON lo suficiente para que transcurra el tiempo fijado en TIMH, se pondrá a ON la bandera de finalización del temporizador utilizado y permanecerá en dicho estado hasta que se resetee TIMH (es decir, hasta que su condición de ejecución se ponga en OFF). El rango del valor seleccionado es de 0 a 99,99s, pero a TIMH se le debe pasar un valor comprendido entre ♯0000 y 9999 (BCD). La precisión del temporizador es de 0 a 0,01s. El número de temporizador debe estar entre 0000 y 4095.

Cuando se alcanza el tiempo de muestreo (*nuevoTm3=1*) se ejecuta el bloque de función *PID*. Primero se toman los parámetros y se espera a que termine la fase de posicionamiento en equilibrio nominal (*Posicionado*=1). Luego se calcula la señal de control de forma incremental habiendo discretizado con la aproximación Euler II. Existe la posibilidad de emplear un integrador 'antiwindup' (Astrom y Witenmark). La señal de control calculada se pone en *ukPID*.

Manual.

El siguiente programa se denomina *Manual* y tiene el número 04. Esta tarea se ejecuta si ha sido elegido el control en modo manual, de modo que ha sido activada en la tarea *Inicialización* y, por tanto, *on_Manual* = 1.

Esta tarea se compone de una sola sección llamada CalculaManual que toma el parámetro P2 y lo pone en ukMan para aplicarlo directamente a la salida analógica. Se calcula el *error*, restando *referencia* menos *altura* con -F.

Control Borroso.

El siguiente programa se denomina *Borroso* y tiene el número 05. Esta tarea se ejecuta si ha sido elegido el control Borroso, de modo que ha sido activada en la tarea *Inicialización* y, por tanto, *on_Borroso* = 1.

Cuando se alcanza el tiempo de muestreo (*nuevoTm5*=1) y si ya se ha terminado la fase de posicionamiento en equilibrio nominal tras un cambio de control (*Posicionado*=1), se ejecuta el bloque de función *Borroso* donde se ha programado un control borroso Mandami con 3 conjuntos borrosos para las entradas y 5 para la salida. Esta técnica está explicada en la memoria de este Proyecto (capítulo 7).

Primero se definen los límites de las funciones de pertenencia copiando los parámetros P2 a P18 que llegan del LabView. Luego se dan valores a las variables de entrada y salida. Después se realiza la borrosificación comprobando el grado de pertenencia de las entradas a cada uno de los conjuntos borrosos. En la fase de inferencia se determinan las reglas activas (*bitPremisaXX*=1), se cuantifica el grado de cumplimiento de las premisas de las reglas (*PremisaXX*). En *PonderaRegla* se va sumando la influencia de cada 'singleton' de salida según su premisa asociada ($PremisaXX \cdot cXXu$). En *TotalRegla* se obtiene el valor total de las premisas. Para convertir a escalar se realiza una división de *PonderaRegla* entre *TotalRegla* (siempre que este no valga 0). Por último se suma a la señal de control anterior el incremento de tensión *CambioTension* obtenido en este ciclo.

Control H_∞.

El siguiente programa se denomina *Hinf* y tiene el número 06. Esta tarea se ejecuta si ha sido elegido el control H_∞, de modo que ha sido activada en la tarea *Inicialización* y, por tanto, *on_Hinf* = 1.

Cuando se alcanza el tiempo de muestreo (*nuevoTm6*=1) y si ya se ha terminado la fase de posicionamiento en equilibrio nominal tras un cambio de control (*Posicionado*=1), se ejecuta el bloque de función *Hinf*, donde se ha programado un controlador tipo H_∞ cuyos coeficientes han sido calculados con Matlab.

Primero se definen los coeficientes copiando los parámetros P2 a P17 transmitidos al autómata mediante LabView con el protocolo Host Link. Luego, si *Posicionado*=1, se calcula el error en altura y, después, la señal de control con una ecuación en diferencias:

$$\begin{aligned} ekHoo &:= referencia - altura; \\ ukHoo &:= (1/a0) * (ekHoo * b0 + e(k-1) * b1 + \ldots + e(k-N) * bN \\ &\quad -u(k-1) * a1 - \ldots - u(k-N) * aN); \end{aligned} \tag{A.1}$$

Control GPC.

El siguiente programa se denomina *GPC* y tiene el número 07. Esta tarea se ejecuta si ha sido elegido el control GPC, de modo que ha sido activada en la tarea *Inicialización* y, por tanto, $on\_GPC = 1$.

Cuando se alcanza el tiempo de muestreo (*nuevoTm7*=1) y si ya se ha terminado la fase de posicionamiento en equilibrio nominal tras un cambio de control (*Posicionado*=1), se ejecuta el bloque de función *GPC*. La señal de control con esta estrategia se ha calculado en Matlab y se envía al autómata mediante LabView en el parámetro *P2* de la trama Host Link número 2. En este bloque se copia dicho parámetro a *ukGPC* para que sea la salida analógica.

Actuación común.

El último programa se denomina *ActuacionComun* y tiene el número 08. Esta tarea se ejecuta siempre que el sistema esté encendido (on_off=1). Esta tarea termina el ciclo de refresco y después de ella se vuelve a la tarea 00 (*PrimerPaso*).

Cuando se alcanza el tiempo de muestreo (*nuevoTm7*=8) se ejecuta el bloque de función *Actuacion*, donde se manda la señal de control correcta a la salida analógica y se actualizan las variables.

Primero se decide cuál es la señal de control definitiva (*u_def*), es decir, cuál es la etiqueta que contiene la señal que se debe sacar y si hay que sumarle la tensión

nominal (*TensionNom*). Luego se aplica la señal de control, comprobando antes si satura el actuador (satura si $u\_def < u\_min$ o $u\_def > u\_max$). La señal de control viene en un rango de 0 a 10V, pero en la posición *SalidaAnalog* debe tener un rango de 0 a $4000dec$ ($FA0hex$) por lo que se multiplica por 400 (*NumReal400*); además, debe pasarse a entero sin signo con la función *REAL_TO_UINT*. Por último, se actualizan las variables según el control actual. Se toma el valor de SalidaAnalog como señal de control realmente emitida porque hay una discretización al pasar de tipo REAL a UINT; para actualizar correctamente se le debe quitar la tension nominal.

Apéndice B

Código de los programas desarrollados en Matlab.

B.1. Estudio previo.

PlantasLevineu.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       PLANTAS DEL SISTEMA LEVINEU EN VARIOS PUNTOS DE TRABAJO
%       Y DEFINICION DE LA PLANTA NOMINAL.
%
%       Autores: F.B.Breña, J.J.Caparros, J.A.Perez
%       Tutor:   M.G. Ortega
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Gi=[numGi,denGi]
%Gnom=[numGnom,denGnom];
nptostrab=3; %numero de puntos de trabajo

%   Plantas identificadas experimentalmente.
%Punto de trabajo 1: a 1V y magnitud de PRBS=0.5
numG1=[0 2.4899 -93.3089 1941.1931]; denG1=[1 20.1682 24.3028
244.7105];
%Punto de trabajo 2: a 4V y magnitud de PRBS=0.5
%Este punto sera el punto de equilibrio nominal
numG2=[0 1.2273 -74.3129 1136.1254]; denG2=[1 16.8837 26.4839
274.7454];
%Punto de trabajo 3: a 7.5V y magnitud de PRBS=0.5
numG3=[0    1.0000  -30.9559  398.8621]; denG3=[1   27.4029
```

```
80.8324  405.5041];

%   Definicion del modelo nominal.

%Con la media de los coeficientes en los tres puntos de trabajo
%numGnommedia=(numG1+numG2+numG3)/nptostrab;
%denGnommedia=(denG1+denG2+denG3)/nptostrab;
%Produce
%numGnommedia=[0 1.5724 -66.1925 1158.7268];
%denGnommedia=[1 21.4849 43.8730 308.3200];

%Con la media de los coeficientes de los puntos de trabajo 1 y 2
%numGnommedia=(numG1+numG2)/(nptostrab-1);
%denGnommedia=(denG1+denG2)/(nptostrab-1);
%Produce
%numGnommedia=[0 1.8586 -83.8109 1538.6592];
%denGnommedia=[1 18.5259 25.3934 259.7276];

%Con un modelo reducido.
deltanom=0.1; wn=3.7; numGnom=[0 0 1 6*wn^2]; denGnom=[0 1
2*deltanom*wn wn^2];

%Dibujo los bode de las plantas.
if(0)
    figure(1);
    hold on;
    bode(numG1,denG1,'b');      %sale en azul
    bode(numG2,denG2,'g');      %sale en verde
    bode(numG3,denG3,'r');      %sale en rojo
    %bode(numGnommedia,denGnommedia,'c--');    %sale en cyan
    bode(numGnom,denGnom,'k--');  %sale en negro
    hold off;
end
```

EstudioSinCompensar.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       ESTUDIO DEL SISTEMA LEVINEU SIN COMPENSAR
%
%       Autor: F.Borja Breña
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Estudiamos el sistema sin compensar en el punto de operacion 2
%   Punto de trabajo 2: tension de entrada=5V.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Definicion de la planta
numG2=[1.2273 -74.3129 1136.1254]; denG2=[1 16.8837 26.4839
```

```
274.7454]; G2=tf(numG2,denG2);

%   Dibujamos el lugar de las raíces
[cerosG2,polosG2,ganG2]=tf2zp(numG2,denG2) figure(1);
rlocus(numG2,denG2);
%Produce el siguiente resultado.
%cerosG2 =  13.4333 +20.7091i; 13.4333 -20.7091i
%polosG2 =  -26.6490; -0.2980 + 3.7658i; -0.2980 - 3.7658i
%ganG2 =    3.1210

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Respuestas en BA.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t1=[0:0.005:15]';           %defino el vector de tiempos

%   Dibuja la respuesta ante escalon
figure(2) ye=step(G2,t1); plot(t1,1,'r',t1,ye,'b');
title('Respuesta a un escalon unitario del sistema sin compensar
en BA'); xlabel('tiempo(sg)'); grid;

%   Dibuja la respuesta ante rampa
figure(3); ramp=t1; yr=lsim(G2,ramp,t1);
plot(t1,yr,'b',t1,ramp,'r'); title('Respuesta a una rampa del
sistema sin compensar en BA'); xlabel('tiempo(sg)');

%   Dibuja la respuesta ante ruido
noise = rand(size(t1)); figure(4) yn=lsim(G2,noise,t1);
plot(t1,yn,'b',t1,noise,'r'); title('Respuesta a un ruido
aleatorio del sistema sin compensar en BA');
xlabel('tiempo(seg)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Estudio y respuestas en BC.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[numG2bc,denG2bc]=feedback(numG2,denG2,1,1,-1);
G2bc=tf(numG2bc,denG2bc);
[cerosG2bc,polosG2bc,ganG2bc]=tf2zp(numG2bc,denG2bc)

t2=[0:0.001:2]';            %defino el vector de tiempos

%   Dibuja la respuesta ante escalon
figure(5) ye=step(G2bc,t2); plot(t2,ye,'b',t2,1,'r');
title('Respuesta a un escalon unitario del sistema sin compensar
en BC'); xlabel('tiempo(sg)'); grid;

%   Dibuja la respuesta ante rampa
figure(6); ramp=t2; yr=lsim(G2bc,ramp,t2);
plot(t2,yr,'b',t2,ramp,'r'); title('Respuesta a una rampa del
sistema sin compensar en BC'); xlabel('tiempo(sg)');

%   Dibuja la respuesta ante ruido
```

```
figure(7) noise = rand(size(t2)); yn=lsim(G2bc,noise,t2);
plot(t2,yn,'b',t2,noise,'r'); title('Respuesta a un ruido
aleatorio del sistema sin compensar en BC');
xlabel('tiempo(seg)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Respuesta en frecuencia.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   Gráficas de nyquist en la misma figura:
%   1-para frecuencias cercanas al origen
%   2-para rango amplio de frecuencias
figure(8); nyquist(G2);
%w = logspace(1.5,5,10000);
%Njw = polyval(numG2,j*w);
%Djw = polyval(denG2,j*w);
%Gjw = Njw./Djw;
%subplot(121),plot(real(Gjw),imag(Gjw)), title('Nyquist en torno a origen');
%xlabel('Re'), ylabel('Im'), grid;
%w = logspace(-1000,1000,1000000);
%Njw = polyval(numG2,j*w);
%Djw = polyval(denG2,j*w);
%Gjw = Njw./Djw;
%plot(real(Gjw),imag(Gjw)), title('Nyquist para rango amplio de frec');
%xlabel('Re'), ylabel('Im'), grid;
%subplot(122),plot(real(Gjw),imag(Gjw)), title('Nyquist para rango amplio de frec');
%xlabel('Re'), ylabel('Im'), grid;

%   Diagrama de Bode
figure(9); w1 = logspace(-3,4,10000);
%bode(gs,w1);               %no se representa en la memoria
margin (numG2,denG2);       %hace bode y marca márgenes fase y ganancia
```

TablaRouth.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CONSTRUCCION DE LA TABLA DE ROUTH DEL SISTEMA LEVINEU
%
%   Autor: F.Borja Breña.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Definicion del denominador de la planta en BA
denG2=[1 16.8837 26.4839 274.7454];
%Definicion del denominador de la planta en BC
denG2bc=[1 18.1110 -47.8290 1410.8708];

%Definicion del polinomio de partida
a=denG2;        %   Para tabla de Routh del BA
```

```
%a=denG2bc;     %   Para tabla de Routh del BC
n=length(a)-1;  %orden del denominador
coc=n/2;
if n~=2*coc     %para saber si el orden es par
    a=[a 0]
end

%Relleno las dos primeras filas de la tabla
tabla=zeros(n+1,ceil(n/2)+1); fila=1; for i=1:n+1
    tabla(fila,ceil(i/2))=a(i);
    if fila==1
        fila=2;
    else
        fila=1;
    end
end

%Relleno las siguientes filas
for fila=3:n+1
    for col=1:ceil(n/2)
        tabla(fila,col)=((tabla(fila-1,1)*tabla(fila-2,col+1)...
        -tabla(fila-2,1)*tabla(fila-1,col+1)))/tabla(fila-1,1);
    end
end tabla
```

EspecificacionControl.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Calculo de parametros en el lugar de las raices
%   para cumplir las especificaciones
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SO = 0.3;  %30% de sobreoscilacion
delta = sqrt((log(SO)^2)/((pi^2)+(log(SO)^2))),

trise = 2;  %2 segundos de tiempo de subida
alfa = cos(delta); wnmin = (pi - alfa)/(trise * sqrt(1 -
delta^2)),

%Produce como resultados
%delta=0.3579
%wnmin= 1.1807
```

B.2. Control Clásico.

ZieglerNicholsBA.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       CONTROLES CLASICOS CON AJUSTE ZIEGLER-NICHOLS
%       EN BUCLE ABIERTO DEL SISTEMA LEVINEU
%
%       Autor: F.Borja Breña
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Se calculan controles clasicos con reglas Ziegler-Nichols en BA
%   en el punto de trabajo 2: tension de entrada=5V; magnitud de PRBS=0.5
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Definicion de la planta
numGnom=[1.2273 -74.3129 1136.1254]; denGnom=[1 16.8837 26.4839
274.7454]; Gnom=tf(numGnom,denGnom);
%Calculo de param. Z-N
tau=1; retardo=0.3; ganestat=4.2;

%Parametros de los controladores usando reglas de Z-N(en bucle abierto)
%En cualquier caso,estas reglas no nos permiten diseñar un PD

%Controlador P
%KpP = tau/(ganestat*retardo);
%numPZN=KpP;
%denPZN=1;
%numbaP=conv(numPZN,numGnom);
%denbaP=conv(denPZN,denGnom);
%t=[0:0.001:6]';            %defino el vector de tiempos
%figure(1);
%ye=step(numbaP,denbaP,t);
%plot(t,ye,t,1);
%title('Respuesta a un escalon unitario');
%xlabel('tiempo(sg)');

%Controlador PI
%KpPI = 0.9*tau/(ganestat*retardo);
%TiPI = retardo/0.3;
%TdPI = 0;
%ceroPI = 1/TiPI;
%ganPI = KpPI;
%numPIZN = [ganPI ganPI*ceroPI];
%denPIZN = [1 0];
%numbaPI=conv(numPIZN,numGnom);
%denbaPI=conv(denPIZN,denGnom);
%t=[0:0.001:6]';
```

```
%figure(2);
%yf=step(numbaPI,denbaPI,t);
%plot(t,yf,t,1);
%title('Respuesta a un escalon unitario');
%xlabel('tiempo(sg)');

%Controlador PID
KpPID = 1.2*tau/(ganestat*retardo); TiPID = 2*retardo; TdPID =
0.5*retardo; numPIDZN = [KpPID*TdPID*TiPID KpPID*TiPID KpPID];
denPIDZN = [TiPID 0]; cerosPIDZN = roots(numPIDZN) polosPIDZN =
roots(denPIDZN) ganPIDZN = KpPID*TdPID
numbaPID=conv(numPIDZN,numGnom); denbaPID=conv(denPIDZN,denGnom);
t=[0:0.001:6]'; figure(3); yg=step(numbaPID,denbaPID,t);
plot(t,yg,t,1); title('Respuesta a un escalon unitario');
xlabel('tiempo(sg)');
```

B.3. Control H_∞.

HinfLevineu.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%        CONTROLADOR H-INFINITO S/KS/T PARA SISTEMA LEVINEU
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Autor: F.Borja Breña & M.G.Ortega
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Se seguira el metodo propuesto por M.G.Ortega,2001.
%   Emplea el 'mu-Analysis and Synthesis Toolbox' y el 'Control Toolbox'.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear all:
%close all;
clc;
%% Para probar este fichero directamente en Matlab quitar comentarios a
%% auto_usuario (linea 31) y Tm (linea 335)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 1:Calculo de la planta nominal y de las incertidumbres
%   1.1:Realizar pruebas para identificar al sistema en distintos
%   puntos de trabajo.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PlantasLevineu;

%   Si auto_usuario = 0 -> Wt, Ws y G generada automaticamente
%   Si auto_usuario = 1 -> Wt, Ws y G definida por el usuario
%auto_usuario=0;     % para pruebas se definen automaticamente

% Si en un futuro se permite al usuario cambiar la planta nominal.
%if auto_usuario == 1
%    numGnom=canal_numGnom;
%    denGnom=canal_denGnom;
%end

[aGnom, bGnom, cGnom, dGnom]=tf2ss(numGnom, denGnom);

sys_pn=tf( numGnom, denGnom); sys_p1=tf( numG1, denG1); sys_p2=tf(
numG2, denG2); sys_p3=tf( numG3, denG3);

sys0=sys_pn; sys1=sys_p1; sys2=sys_p2; sys3=sys_p3;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   1.2: Normalizar modelos.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
umax=0.5;       %maximo cambio de tension=0.5voltios
altmax=10;      %maxima altura=10cm
Du=[umax]; De=[altmax];

% G=inv(De)*G'*Du  ( G: escalada  ;  G': sin ecalar )
% Para utilizar matrices de estados
% A=A' ; B=B'*Du ; C=inv(De)*C' ; D=inv(De)*D'*Du

aGnomn=aGnom; bGnomn=bGnom*Du; cGnomn=inv(De)*cGnom;
dGnomn=inv(De)*dGnom*Du;

sys0n=inv(De)*sys0*Du;      % n de normalizado
sys1n=inv(De)*sys1*Du; sys2n=inv(De)*sys2*Du;
sys3n=inv(De)*sys3*Du;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   1.3: Cálculo y dibujo de los valores singulares de la incertidumbre multiplicativa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculo de la incertidumbre normalizada respecto a sistema nominal:

w=logspace(-2,1,1000);

SV_inc_sys1n=CalculaSvIncer_SISO(sys0n,sys1n,w);
SV_inc_sys2n=CalculaSvIncer_SISO(sys0n,sys2n,w);
SV_inc_sys3n=CalculaSvIncer_SISO(sys0n,sys3n,w);


if (1)      % Dibuja todas las incertidumbres
   SV_inc_max=[SV_inc_sys1n(:,1) SV_inc_sys2n(:,1) SV_inc_sys3n(:,1) ];
else    % Deja fuera la del punto de operación 1
   SV_inc_max=[SV_inc_sys2n(:,1) SV_inc_sys3n(:,1) ];
end

if (0)
   figure;
   grid;
   semilogx(w,db(SV_inc_max));
   legend('sys1','sys2','sys3');
   xlabel('frecuencia (rad/s)');
   ylabel('V.S. max. (dB)');
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 2: Diseño de las funciones de ponderacion.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Modelo nominal normalizado sin retardos: sys0srn
```

```
% Conversión del modelo nominal a la forma de Toolbox Hoo discreto

Planta_sys=pck(aGnomn,bGnomn,cGnomn,dGnomn);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 2.1: Diseño de las funciones de ponderacion en continuo.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% --------- Diseño de Wt ---------------
%
%                         1/b
% Kaf(db)                  ------------------
%                         /
%                        /
%                       /
%                      /
% k(db) ____________/
%                  1/tau
%
% Wt=k(as+1)/(bs+1)
%
% Wt=KWt(tau*s+1)/((tau*(KWt/Kaf)s+1)

if auto_usuario==1
    %Usuario define Wt
    numWt=[(10^(canal_KWt/20))*[canal_tau 1]];    %con este numerador mod(Wt)>mod(Gi)
    denWt=[canal_tau*10^((canal_KWt-Kaf)/20) 1];  %polo a alta frecuencia para hacer Wt propia
    Wt=[numWt;denWt];
else
    %Si el controlador Hinfinito es el automatico
    %Propongo Wt
    KWt=-9.5;         % Valor en dB buscado en baja frecuencia
    tau=0.7;          % Frecuencia aproximada de corte con 0dB de Wt
    Kaf=40;
    numWt=[(10^(KWt/20))*[tau 1]];
    denWt=[tau*10^((KWt-Kaf)/20) 1];
    Wt=[numWt;denWt];
end

% Calculo de la frecuencia de corte de 0dB de Wt
Wt2=tf(numWt,denWt); [mag,phase]=bode(numWt,denWt,w);
[MgWt,MfWt,wgWt,wfWt]=margin(Wt2);

% Representacion de Wt junto con las incertidumbres
if (1)
   figure;
   semilogx(w,db(SV_inc_max),w,db(mag),'k-');grid;
   legend('sys1','sys2','sys3','Wt');
   xlabel('frecuencia (rad/s)');
   ylabel('V.S. max. (dB)');
end

%% --------- Diseño de Ws ---------------
```

```
if auto_usuario == 1
    %Usuario define Ws
    alfa=0.5;               %function gain at high frequency
    beta=1e-4; %o bien 1e-6 %function gain at low frequency
    numWs=[alfa wfWt*10^(canal_kappa-1)];
    denWs=[1 beta*wfWt*10^(canal_kappa-1)];
    Ws=[numWs;denWs];
else
    %Propongo Ws
    alfa=0.5;               %function gain at high frequency
    beta=1e-4; %o bien 1e-6 %function gain at low frequency
    kappa=0.5;  %varia entre 0 y 1 para provocar cambios logaritmicos en ws
    wt=wfWt;                %crossover frequency of Wt
    ws=wt*10^(kappa-1);     %crossover frequency of Ws
    numWs=[alfa ws];
    denWs=[1 beta*ws];
    Ws=[numWs; denWs];
end

%% --------- Diseño de Wks (Wu o Wr) ---------------

Wu=[0.1;1];


%% CAMBIO DE FORMATO DE LAS FUNCIONES DE PONDERACION AL mu-analysis
     %% Diseño de Wt
     Wt_sys = nd2sys(Wt(1,:),Wt(2,:));

    %% Diseño de Ws
     Ws_sys = nd2sys(Ws(1,:),Ws(2,:));

    %% Diseño de Wks
     Wu_sys = nd2sys(Wu(1,:),Wu(2,:));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 3: Construccion de la planta aumentada segun el tipo de diseño
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 3.1: Eleccion del tipo de diseño del controlador.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% ELECCION DEL TIPO DE DISEN\O:
%%%
%%%    design = 1  =>   S/KS
%%%    design = 2  =>   S/T
%%%    design = 3  =>   S/KS/T
%%%    desing = 4 =>    GS/T

design=3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 3.1: Construccion de la planta aumentada.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%    design = 1  =>   S/KS
if design==1
  systemnames = 'Planta_sys Ws_sys Wu_sys';
  inputvar = '[ref(1); control(1)]';
  outputvar = '[ Ws_sys ; Wu_sys ; ref - Planta_sys ]';
  input_to_Planta_sys = '[ control ]';
  input_to_Ws_sys = '[ ref - Planta_sys ]';
  input_to_Wu_sys = '[ control ]';
  sysoutname = 'Planta_aumentada';
  cleanupsysic = 'yes';
  sysic

%%%    design = 2  =>   S/T
elseif design==2
  systemnames = 'Planta_sys Ws_sys Wt_sys';
  inputvar = '[ref(1); control(1)]';
  outputvar = '[ Ws_sys ; Wt_sys ; ref - Planta_sys ]';
  input_to_Planta_sys = '[ control ]';
  input_to_Ws_sys = '[ ref - Planta_sys ]';
  input_to_Wt_sys = '[ Planta_sys ]';
  sysoutname = 'Planta_aumentada';
  cleanupsysic = 'yes';
  sysic

%%%    design = 3  =>   S/KS/T
elseif design==3
  systemnames = 'Planta_sys Ws_sys Wu_sys Wt_sys';
  inputvar = '[ref(1); control(1)]';
  outputvar = '[ Ws_sys ; Wu_sys ; Wt_sys ; ref - Planta_sys ]';
  input_to_Planta_sys = '[ control ]';
  input_to_Ws_sys = '[ ref - Planta_sys ]';
  input_to_Wu_sys = '[ control ]';
  input_to_Wt_sys = '[ Planta_sys ]';
  sysoutname = 'Planta_aumentada';
  cleanupsysic = 'yes';
  sysic

%%%    desing = 4 =>    GS/T
elseif design==4
  Wd_sys=pck([],[],[],0.02*[1 0;0 1]);
  systemnames = 'Planta_sys Ws_sys Wt_sys Wd_sys';
   % Aquí Wt_sys pondera a T y Ws*G(s) (y no sólo Ws) pondera a S
  inputvar = '[dist(1); ref(1); control(1)]';
  outputvar = '[ Ws_sys ; Wt_sys ; Wd_sys - Planta_sys ]';
  input_to_Planta_sys = '[ control + dist ]';
  input_to_Ws_sys = '[ Planta_sys ]';
  input_to_Wt_sys = '[ control ]';
  input_to_Wd_sys = '[ ref ]';
  sysoutname = 'Planta_aumentada';
  cleanupsysic = 'yes';
  sysic
```

```
  clear Wd_sys;

else
  error('La eleccion no corresponde con ningun tipo de los diseños S/KS, S/T, S/KS/T');

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calcula el controlador Hoo, el bucle cerrado del sistema y dibuja los resultados.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 4: Calculo del controlador Hinf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 4.1: Resolucion del problema en continuo.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Definicion para llamar a Sintesis de Hinf.
Planta_aumentada;   % Planta aumentada que acabamos de calcular
ncont=1;        % No. de entradas del controlador
nmeas=1;        % No. de salidas del controlador
gmax=50;        % gamma maxima (definida en el principio del programa)
gmin=0.1;       % gamma minima
tol=0.1;        % tolerancia
% k = controlador calculado
% N = sistema en bucle cerrado
% gfin = gamma final calculado

%function [k,g,gfin,ax,ay,hamx,hamy] =
%           = hinfsyn(p,nmeas,ncon,gmin,gmax,tol,ricmethd,epr,epp)
[kcontrn,g,gfin] =
hinfsyn(Planta_aumentada,nmeas,ncont,gmin,gmax,tol);

if ( ~exist('kcontrn') | isempty(kcontrn) )
    error(' No se ha podido calcular el controlador Hinf en continuo');
end

if (1) % Dibujar Tzw
   [Ag,Bg,Cg,Dg]=unpck(g);
   sys_g=ss(Ag,Bg,Cg,Dg);
   figure;
   sigma(sys_g);
end

 [acontrn,bcontrn,ccontrn,dcontrn]=unpck(kcontrn);
 sys_K=ss(acontrn,bcontrn,ccontrn,dcontrn);
 sys_G=ss(aGnomn,bGnomn,cGnomn,dGnomn);

 if (1)  % Dibujar funciones en bucle cerrado y sus cotas
    DibujaFuncionesBC(sys_K,sys_G,Wt,Ws,Wu,w);
end
```

```
if (0)
   [acontrnred,bcontrnred,ccontrnred,dcontrnred]=...
                balmr(acontrn,bcontrn,ccontrn,dcontrn,3);
   pause;
else
   acontrnred=acontrn;
   bcontrnred=bcontrn;
   ccontrnred=ccontrn;
   dcontrnred=dcontrn;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 4.2: Discretización del controlador mediante una bilineal de Tustin con un
% tiempo de muestreo Tm segundos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tm=canal_Tm;
%Tm=0.150 ;
[acontrndred,bcontrndred,ccontrndred,dcontrndred]=...
        bilin(acontrnred,bcontrnred,ccontrnred,dcontrnred,1,'Tustin',Tm);
[nucontrnd,decontrnd]=ss2tf(acontrndred,bcontrndred,ccontrndred,dcontrndred);

% SOLO FALTA DESESCALAR EL CONTROLADOR DISCRETO NORMALIZADO
nucontrd = Du/De*nucontrnd; decontrd = decontrnd;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 7: Implementacion en sistema real con control digital C(z)
%   Produce dos vectores en funcion de z, el primer elemento es el
%   de mayor grado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M=7;N=7; if nucontrd(1,1) == 0
    nucontrd2=zeros(1,length(nucontrd)-1);
    for p=2:length(nucontrd)
    nucontrd2(1,p-1)=nucontrd(1,p);
    end
    nucontrd=nucontrd2;
end

while length(nucontrd)<= M
    nucontrd=[nucontrd 0];
end while length(decontrd)<=N
    decontrd=[decontrd 0];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 8: Envio al automata a traves de LabView los valores calculados
%   para el controlador Hinf.
%   Actualmente se envian los coeficientes, pero se podria mejorar enviando
%   los valores de los polos, ceros y la ganancia.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

CalculaSvIncer_SISO.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Este archivo es llamado desde HinfLevineu.m para el control H-infinito
% del Sistema Levineu.
% CalculaSvIncert_SISO calcula los valores singulares de la incertidumbre
% multiplicativa de un sistema SISO.
% Devuelve vector de dimensión 1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function svincert=CalculaSvIncer_SISO(sysn,sys,w)

svincert=[]; [msysn,fsysn]=bode(sysn,w); [msys,fsys]=bode(sys,w);

for k=1:length(w);
   % Gn(jw): matriz de complejos de 2x2
   mGn=[msysn(1,1,k)];
   fGn=pi/180*[fsysn(1,1,k)];
   Gn=mGn.*exp(j*fGn);
   % G(jw): matriz de complejos de 2x2
   mG=[msys(1,1,k) ];
   fG=pi/180*[fsys(1,1,k) ];
   G=mG.*exp(j*fG);
   % Calculo de la incertidumbre multiplicativa en la salida para esa frecuencia
   incG=(G-Gn)*inv(Gn);
   svincert=[svincert; svd(incG)'];
end return;
```

DibujaFuncionesBC.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Este archivo es llamado desde HinfLevineu.m
% para el control H-infinito del Sistema Levineu.
% Dibuja funciones de transferencia en bucle cerrado
% (funciones de senbilidad) y sus respectivas cotas superiores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function DibujaFuncionesBC(sys_K,sys_G,Wt,Ws,Wks,w)
% Función de lazo en bucle abierto
sys_L=sys_G*sys_K;

% Función identidad
sys_I=ss([],[],[],eye(1));

% Función de sensibilidad
sys_S=feedback(sys_I,sys_L,-1);

% Función de sensibilidad complementaria
sys_T=feedback(sys_L,sys_I,-1);
```

```
% Función de sensibilidad al control
sys_KS=sys_K*sys_S;

% Respuesta frecuencial

% Función de sensibilidad
[m_S,f_S]=bode(sys_S,w); m_S=m_S(1,:);

% Función de sensibilidad complementaria
[m_T,f_T]=bode(sys_T,w); m_T=m_T(1,:);

% Función de sensibilidad al control
[m_KS,f_KS]=bode(sys_KS,w); m_KS=m_KS(1,:);

% Funciones de ponderación invertidas
[m_WT,f_WT]=bode(Wt(2,:),Wt(1,:),w);
[m_WS,f_WS]=bode(Ws(2,:),Ws(1,:),w);
[m_WKS,f_WKS]=bode(Wks(2,:),Wks(1,:),w);

% Gráficas
figure; semilogx(w,20*log10(m_S),'b',w,20*log10(m_WS),'b--');
xlabel('frecuencia (rad/s)'); ylabel('ganancia (dB)');
title('Función de sensibilidad y su ponderación'); grid;
legend('S','W_{S}^{-1}');

figure; semilogx(w,20*log10(m_T),'r',w,20*log10(m_WT),'r--');
xlabel('frecuencia (rad/s)'); ylabel('ganancia (dB)');
title('Función de sensibilidad complementaria y su ponderación');
grid; legend('T','W_{T}^{-1}');

% Gráficas
figure; semilogx(w,20*log10(m_KS),'b',w,20*log10(m_WKS),'b--');
xlabel('frecuencia (rad/s)'); ylabel('ganancia (dB)');
title('Función de sensibilidad al control y su ponderación');
grid; legend('KS','W_{KS}^{-1}');
```

B.4. Control GPC.

GPCLevineu.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       CONTROL PREDICTIVO GENERALIZADO PARA EL SISTEMA LEVINEU
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       Autor: Fernando-Borja Breña Lajas
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Se seguira el metodo propuesto por E.F.Camacho y C. Bordons, 1999.
%   Estos scripts son adaptacion de los desarrollados en su PFC
%   por J.A. Yanes, 2004.
%   GPCLevineu es el script principal y llama a PlantasLevineu.m,
%   DivisionPolinomica.m y ExtraeMatrices.m.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%clear;
close all; clc;

%%%%%%%%%%%%%%%%%%%%%%
%   Definiciones para que funcione en las pruebas.
%%%%%%%%%%%%%%%%%%%%%%
%auto_usuario=0;
%canal_ref=25;
%canal_uk1=0;
%canal_altura=7;
%canal_Tm=0.15;
%canal_delta=1;
%canal_lambda=1000;
%canal_hc=50;
%canal_hp=50;
%calcula_matrices=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 1: Definicion de plantas, variables y constantes.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Tras un Cambio de configuracion se deben volver a calcular las matrices del GPC.
if calcula_matrices == 1

if ~exist('iniciado')
    global HG F P Tm uk vector_altura vector_uk
    vector_altura=[0 0 0]';
    vector_uk=[0 0];
```

```
    iniciado=1;
end

% periodos de muestreo
Tm=canal_Tm;

% Estructura Pos ->    A,B,C: polinomios en (1/z) del modelo CARIMA
% Modelo CARIMA: A(1/z)y(t)=z^(-d)*B(1/z)*u(t-1)+C(1/z)*e(t)/D
% D= 1-1/z
if auto_usuario == 0
   %Definicion de la planta nominal en z.
    PlantasLevineu;
    [numGz,denGz]=c2dm(numGnom,denGnom,Tm);
    Pos.A=denGz;
    Pos.B=numGz;
    Pos.C=1;
    %Pos.A=[1 -1.6108 0.8949];    %para Tm=0.15
    %Pos.B=[0 1.0031 0.7017];
else
    Pos.A=[1 -1.6108 0.8949];    %para Tm=0.15
    Pos.B=[0 1.0031 0.7017];
    %Pos.A=canal_denGz;
    %Pos.B=canal_numGz;
    Pos.C=1;
end

% Constantes de la funcion de coste
% Estructura Costes ->   delta,lambda,kappa.
% Estructura Hor ->      pre,con.
if auto_usuario == 0
    Costes.d=1;            % coste asociado a error en posicion
    Costes.l=500;          % coste asociado a incremento de señal de control
    Costes.k=0;             % coste asociado al consumo de accion de control
    Hor.pre=50;            % horizonte de prediccion
    Hor.con=50;            % horizonte de control
else
    Costes.d=canal_delta;   % coste asociado a error en posicion
    Costes.l=canal_lambda;  % coste asociado a incremento de señal de control
    Costes.k=0;             % coste asociado al consumo de accion de control
    Hor.pre=canal_hp;       % horizonte de prediccion
    Hor.con=canal_hc;       % horizonte de control
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 2: Calcula matrices de prediccion optima.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% A*y1 = B*u0 -->CARIMA--> DA*y1 = B*Du0 + C*e1  ( D = 1-1/z )
% A,B,C son polinomios en 1/z  ( a0 + a1*1/z + a2*1/z2 + ... )
% hor es el horizonte de prediccion ( = al horizonte de control )
% si se desea un horiz. de control menor basta con eliminar de G
% las ultimas ( horpre - horcon ) columnas
%
```

```
% [y1 y2 ... yhor]' = G * [Du0 Du1 ... Duhor-1]' + P * [Du-1 Du-2 ...]' +
% + F * [y0 y-1 y-2 ...]'
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hp=Hor.pre; hc=Hor.con; if hc>hp
    hp=hc;
    %disp('error: Horizonte prediccion < Horizonte control');
    %return;
end

DA=conv(Pos.A,[1 -1]);
%Ecuacion diofantica.
[e,f]=DivisionPolinomica(1,DA,hp);
%Division para comprobaciones de grados.
[m,n]=DivisionPolinomica(1,1,hp);
%Obtencion de las matrices G,P y F.
[Gy,Py,Fy]=ExtraeMatrices(e,f,m,n,Pos.B); lg=length(Gy(1,:));
Gy(:,lg-hp+hc+1:lg)=[];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 3: Calculo de la matrices de control optimo.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculo de la matriz fila Hg.
HG=inv(Gy'*Gy+Costes.l*eye(length(Gy(1,:))))*Gy';
% Se quitan las columnas vacias de P y F
Py(:,2:length(Py(1,:)))=[];
Fy(:,length(Pos.A)+1:length(Fy(1,:)))=[]; F=Fy; P=Py;
%   Matriz fila Hg
HG=HG(1,:); %%
% Para versiones anteriores de Matlab que no permiten matrices de mas de 50
% elementos, se divide F en dos submatrices F1 y F2 (deben ser globales).
%   F1=F(1:15,:);
%   F2=F(16:hc,:);
%%

end         %Fin del calculo de matrices.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 4: Calculo de la señal de control optimo.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for p=1:size(F,2)-1
    vector_altura(size(F,2)-p+1,1)=vector_altura(size(F,2)-p,1);
end vector_altura(1,1)=canal_altura;
vector_uk(1,2)=vector_uk(1,1); vector_uk(1,1)=canal_uk1;
vector_ref=canal_ref*ones(hp,1);

uk=HG*vector_ref-HG*P*(vector_uk(1,1)-vector_uk(1,2))+...
    vector_uk(1,1)-(HG*F)*vector_altura;

%Prevengo la saturacion del actuador.
u_min=0; u_max=10;
```

```
if uk<u_min
    uk=u_min;
else
    if uk>u_max
        uk=u_max;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Fase 5: Envio al automata a traves de LabView la señal de control optimo calculada.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DivisionPolinomica.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       DIVISION POLINOMICA CON HORIZONTE
%
%       [cociente resto]=division(dividendo,divisor,hor)
%
%       Este archivo es llamado desde GPCLevineu.m
%       para el control GPC del Sistema Levineu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [cociente,resto]=division(dividendo,divisor,hor)

n=length(dividendo); m=length(divisor); if n<m
    dividendo(n+1:m)=0;
end if n>m
    disp('error: orden(dividendo) > orden(divisor)');
    return;
end cociente=zeros(hor,hor); resto=zeros(hor,m); for f=1:hor
    [q(f) r]=deconv(dividendo,divisor);
    resto(f,:)=r;
    dividendo=[r(2:m) 0];
end resto(:,1)=[]; for f=1:hor
    cociente(f,:)=[q(1:f) zeros(1,hor-f)];
end
```

ExtraeMatrices.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Extrae las matrices de prediccion optima para el GPC.
%
%   Este archivo es llamado desde GPCLevineu.m
%   para el control GPC del Sistema Levineu.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [Gf,P,F]=extrae(e,f,m,n,b)

ge=length(e(1,:))-1; gf=length(f(1,:))-1; gm=length(m(1,:))-1;
gn=length(n(1,:))-1;

if ge~=gm
    disp('error: grado(e) <> grado(m)');
    return;
end
if (gm+gf)<gn
    disp('error: grado(n) > grado(m*f)');
    return;
else
    for i=gn+2:gm+gf+1
        n(:,i)=0;
    end
end
for i=1:ge+1
    E(i,:)=conv(e(i,:),m(i,:));
    F(i,:)=conv(m(i,:),f(i,:))+n(i,:);
end
for i=1:ge+1
    G(i,:)=conv(E(i,:),b);
end gg=length(G(1,:))-1; gb=length(b)-1; if gg~=(ge+gm+gb)
    disp('error: las dimensiones no coinciden');
    return;
end for i=1:ge+1
    P(i,:)=G(i,i+1:i+gg-ge);
    Gf(i,:)=[G(i,i:-1:1) zeros(1,ge+1-i)];
end
```

B.5. Control borroso.

Borroso3entradas.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   SIMULACION DEL CONTROL BORROSO DEL SISTEMA LEVINEU
%
%   Autor: F.B. Breña
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Control tipo Mandami con tres conjuntos borrosos por entrada
%   y cinco para la salida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fase 1: el objetivo es controlar la altura de una pelota.
%Fase 2: se definen entradas (error en altura y derivada del error) y salida (cambiotension)

%Fase 3: se eligen variables, valores y reglas.
%variables:
```

```
%e = error(cm) Universo: -50 - +50cm,
%de = derivada del error (cm/s)
%u = cambiotension (V) Universo: -10 - +10V
%valores:
%para las entradas:N=negativo;C=cero;P=positivo
%para la salida: NG=negativo grande; NP= negativo pequeño; CE= cero;
%                PP=positivo pequeño;PG=positivo grande

%reglas:
%1.IF e es N AND de es N THEN u es PG
%2.IF e es N AND de es C THEN u es PP
%3.IF e es N AND de es P THEN u es CE
%4.IF e es C AND de es N THEN u es PP
%5.IF e es C AND de es C THEN u es CE
%6.IF e es C AND de es P THEN u es NP
%7.IF e es P AND de es N THEN u es CE
%8.IF e es P AND de es C THEN u es NP
%9.IF e es P AND de es P THEN u es NG

%Fase 4: se definen los limites de las funciones de pertenencia

%e=error en altura (cm)
num_val_e = 3; e_N_2 = -25; e_N_3 = 0; e_C_1 = -25; e_C_2 = 0;
e_C_3 = 25; e_P_1 = 0; e_P_2 = 25; e_abs_max = 100;

%de = derivada del error en altura >(cm/s)? cambiar valores numericos si fuera necesario
num_val_de = 3; de_N_2 = -25; de_N_3 = 0; de_C_1 = -25; de_C_2 =
0; de_C_3 = 25; de_P_1 = 0; de_P_2 = 25;

%salida del controlador: u= cambio en la tension
%                        (aumentar la tension provoca mayor velocidad del aire)
u_NG_2 = -5.0; u_NG_3 = -2.5; u_NP_1 = -5.0; u_NP_2 = -2.5; u_NP_3
= 0; u_CE_1 = -2.5; u_CE_2 = 0; u_CE_3 = 2.5; u_PP_1 = 0; u_PP_2 =
2.5; u_PP_3 = 5.0; u_PG_1 = 2.5; u_PG_2 = 5.0; mu_max_u = 1;
tension_max = 10; tension_min = 0;


%Se definen los parametros: salida (y), referencia (ref), error (e), derivada del error (de)
Tm = 0.2; %tiempo de muestreo (s)
y = input('salida anterior:  '); ref = 50; aux = 0; e = y - ref;
de = e - aux;

%Fase 5: Borrosificacion
mu_e_N = left_N(e); mu_e_C = center_C(e); mu_e_P = right_P(e);
mu_de_N = left_N(de); mu_de_C = center_C(de); mu_de_P =
right_P(de);

%Fase 6: Mecanismo de inferencia
%Fase 6.1: Determinacion de reglas activas
%Fase 6.2: Cuantificacion de las premisas de las reglas
regla_on = zeros (num_val_e,num_val_de); premisa = zeros
(num_val_e,num_val_de);
```

```
if (mu_e_N ~= 0) & (mu_de_N ~= 0)
    regla_on(1,1) = 1;
    premisa(1,1) = min (mu_e_N,mu_de_N);
end if (mu_e_N ~= 0) & (mu_de_C ~= 0)
    regla_on(1,2)=1;
    premisa(1,2) = min (mu_e_N,mu_de_C);
end if (mu_e_N ~= 0) & (mu_de_P ~= 0)
    regla_on(1,3)=1;
    premisa(1,3) = min (mu_e_N,mu_de_P);
end if (mu_e_C ~= 0) & (mu_de_N ~= 0)
    regla_on(2,1)=1;
    premisa(2,1) = min (mu_e_C,mu_de_N);
end if (mu_e_C ~= 0) & (mu_de_C ~= 0)
    regla_on(2,2)=1;
    premisa(2,2) = min (mu_e_C,mu_de_C);
end if (mu_e_C ~= 0) & (mu_de_P ~= 0)
    regla_on(2,3)=1;
    premisa(2,3) = min (mu_e_C,mu_de_P);
end if (mu_e_P ~= 0) & (mu_de_N ~= 0)
    regla_on(3,1)=1;
    premisa(3,1) = min (mu_e_P,mu_de_N);
end if (mu_e_P ~= 0) & (mu_de_C ~= 0)
    regla_on(3,2)=1;
    premisa(3,2) = min (mu_e_P,mu_de_C);
end if (mu_e_P ~= 0) & (mu_de_P ~= 0)
    regla_on(3,3)=1;
    premisa(3,3) = min (mu_e_P,mu_de_P);
end

%Fase 7: obtencion de conclusiones por escalado
mu_regla = zeros(num_val_e,num_val_de); mu_accion =
mu_max_u*ones(num_val_e,num_val_de); mu_regla =
mu_accion.*premisa;

%Fase 8: deborrosificacion por centros ponderados
%cada elemento de "centros" es el centro del conjunto borroso que se fuerza
%al activarse la regla i,j
centros=[u_PG_2,u_PP_2,u_CE_2;
         u_PP_2,u_CE_2,u_NP_2;
         u_CE_2,u_NP_2,u_NG_2];
num_aux = zeros (num_val_e,num_val_de); num_aux =
mu_regla.*centros; num_u = sum(sum(num_aux,1),2); den_u =
sum(sum(mu_regla,1),2); if den_u ~= 0
    ucrisp = num_u/den_u
else
    disp('Error:division por cero');
end
```

Bibliografía

[Ast90] Åström,K.J., Wittenmark,B. *Computer-Controlled Systems Theory and Design*. Prentice-Hall. New Jersey, 1990.

[Bal01] Balas,G.J. et al. *μ-Analysis and Synthesis Toolbox, For Use in Matlab*. The Mathworks Inc. 2001.

[Bor96] Bordons Alva, C. *Control predictivo basado en modelo*. Universidad de Sevilla, 1996.

[Bol01] Bolton, W. *Ingeniería de control*. Marcombo-AlfaOmega. México D.F.,2001.

[Cap05] Caparrós Jiménez,J.J. *Construcción, identificación y modelado de un sistema de levitación neumática*. Proyecto Fin de Carrera. Escuela Superior de Ingenieros. Universidad de Sevilla, 2005.

[Cla87a] Clarke, D.W. et al. *Generalized Predictive Control Part. I.The Basic Algorithm*. Automática, vol. 23, no.2, pp. 137-148. 1987.

[Cla87b] Clarke, D.W. et al. *Generalized Predictive Control Part. II. Extensions and Interpretations*. Automática, vol. 23, no.2, pp. 149-160. 1987.

[Cla88a] Clarke, D.W. et al. *Properties of Generalized Predictive Control*. Report No. OUEL 1721/88. Department of Engineering Science. University of Oxford, 1988.

[Doy83] Doyle,J.C. *Synthesis of Robust Controllers and Filters*. Proc.IEEE Conf. on Decision and Control, pages 109-114,1983.

[Doy84] Doyle,J.C. *Lecture Notes on Advances in Multivariable Control*. ONR Honeywell workshop. Minneapolis, 1984.

[Doy89] Doyle,J.C., Glover,K., Khargonekar,P., Francis,B. *State-space Solutions to Standard H_2 and H_∞* Control Problems. IEEE Transactions on Automatic Control,34(8):831-847,1989.

[Esc04] Escaño,J.M., Algarin-Muñoz,D., Ortega,M.G. *Identificación y control de posición de un sistema de levitación neumática*. Jornadas Nacionales de Automática. CEA-IFAC. Ciudad Real, 2004.

[EspyBer] Espinosa,F. y Bergasa,L.M. '*Apuntes sobre control Borroso*. Universidad de Alcalá. Alcalá de Henares.

[Jan98] Jantzen, J. *Tunning of Fuzzy PID Controllers*. Technical University of Denmark, tech. report 98-H 871. Lyngby, 1998.

[MazyMar] Mazo,M. y Marrón,M. *Transparencias sobre control Borroso*. Universidad de Alcalá. Alcalá de Henares.

[Omr03] *Sysmac CS/CJ Programming Consoles Operation Manual*. Omron Electronics. Japan, 2003.

[Omr04a] *Sysmac CS/CJ Programmable Controllers Programming Manual*. Omron Electronics. Japan, 2004.

[Omr04b] *Sysmac CS/CJ Programmable Controllers Operation Manual*. Omron Electronics. Japan, 2004.

[Omr04c] *Sysmac CS/CJ Programmable Controllers Instructions Reference Manual*. Omron Electronics. Japan, 2004.

[Omr04d] *Sysmac CS/CJ Communication Commands Reference Manual*. Omron Electronics. Japan, 2004.

[Ort01] Ortega,M.G. *Aportaciones al control H_∞ de sistemas multivariables*. Tesis doctoral. Escuela Superior de Ingenieros. Universidad de Sevilla, 2001.

[Ort03] Ortega,M.G. y Rubio,F.R. *Systematic design of weighting matrices for the H-infinity mixed sensitivity problem*. Journal of Process Control, 14, pp. 89-98. Elsevier, Inc. 2003.

[Per05] Pérez García de Castro, J.A. *Plataforma de control remoto aplicada a un sistema de levitación neumática*. Proyecto Fin de Carrera. Escuela Superior de Ingenieros. Universidad de Sevilla, 2005.

[Teo03] *Teoría de Sistemas*. Apuntes de cátedra. Escuela Superior de Ingenieros. Universidad de Sevilla, 2003.

[Yan04] Yanes,J.A. *Control Predictivo Lineal de Plataforma*. Proyecto Fin de Carrera. Escuela Superior de Ingenieros. Universidad de Sevilla, 2004.