UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ETS de Ingeniería Informática
Departamento de Informática y Automática

TESIS DOCTORAL

Desarrollo e implementación de una metodología para el diseño de sistemas de control mediante algoritmos evolutivos multiobjetivo

Manuel Parrilla Sánchez
Licenciado en CC. Físicas

MADRID, 2006

Dpt. Informática y Automática.
ETS de Ingeniería Informática. UNED

Desarrollo e implementación de una metodología para el diseño de sistemas de control mediante algoritmos evolutivos multiobjetivo

Manuel Parrilla Sánchez
Licenciado en CC. Físicas
por la
Universidad Nacional de Educación a Distancia.

Director: Dr. Joaquín Aranda Almansa.

A mi hija Rocío,
la siguiente generación

Agradecimientos

El ser humano es un ser social. Su comportamiento está fuertemente condicionado por las interacciones con los demás miembros de la sociedad de la que forma parte. Las relaciones con otras personas influyen en cada individuo en mayor o menor medida, contribuyendo de forma positiva o negativa en sus acciones.

Por esto, quiero desde estas líneas dar las gracias a todas las personas que han tenido en mí una influencia positiva, haciendo posible que llegara a redactar esta tesis. Por problemas de espacio y de memoria no puedo enumerarlos a todos, por lo que si alguien se sintiera excluido le pido perdón desde aquí.

En primer lugar, agradecer a mi director de tesis Dr. Joaquín Aranda Almansa su orientación y apoyo desde el primer momento en que llegué por el Departamento un poco despistado y mostrando mis inquietudes. Gracias por dirigirme por la senda correcta y empujarme, en el buen sentido de la palabra, cuando me encontraba atascado.

Quiero también dar las gracias a Sebastián Dormido Canto por la ayuda que me prestó en lo referente a la utilización del *clúster* de ordenadores del Departamento, sus valiosos comentarios y toda la información que me proporcionó sobre la materia. Sin su ayuda la tarea habría resultado más complicada.

También quiero agradecer a Rafael López, del Centro de Supercomputación de la Universidad Complutense de Madrid por las facilidades y el apoyo ofrecidos para el uso de sus instalaciones. En la vida te encuentras con personas con una actitud positiva y favorecedora de las cosas, y con otras que, por el contrario, ponen problemas a todo, entorpeciendo e imposibilitando. Gracias Rafael por pertenecer al primer grupo.

Además quiero agradecer al Ejército del Aire su apoyo, que hizo posible mi presencia en Denver para presentar una parte de este trabajo en un ciclo de conferencias del AIAA. En particular quiero dar las gracias a mi jefe el coronel D. Fernando Lens Astray (hoy general) por la receptividad mostrada cuando le expuse el tema.

También quiero dar las gracias a mis compañeros de trabajo, por soportarme. En

particular quiero mencionar a Ignacio Redondo Rodríguez, por su gran vitalidad contagiosa de la que me he beneficiado en ocasiones. Gracias Ignacio por apoyarme y animarme.

Tengo también un recuerdo entrañable de un compañero del Instituto: Javier Cantero, muy dotado para las matemáticas, que me ayudó a comprenderlas en una época en que me encontraba un poco desorientado. Donde estés, gracias.

Pero en la vida no todo es trabajo y estudio, también están los sentimientos. El ser humano necesita también unos lazos afectivos que le den estabilidad emocional. Frente a la vorágine de intereses en la que nos movemos, ¡qué importante es a veces una muestra de cariño desinteresado! Este cariño lo recibo de mi familia:

- Mi esposa Juana, que sin compartir mis inquietudes me soporta.
- Mi hija Rocío, que pertenece a la nueva generación en la cadena evolutiva y con cierta probabilidad de mejora, aunque esto último me resulta irrelevante: me basta con su presencia.
- Mi madre y mis hermanos. Quiero hacer una mención especial a mi padre ya fallecido, que no va a poder compartir este momento.

A todos ellos les doy las gracias por su cariño y por estar ahí cuando los necesito, a la vez que les pido perdón por mis ausencias.

Por último, aunque no por ello menos importante, dar las gracias a la comunidad del *software libre*, por el enorme trabajo que realiza y por crear, de forma desinteresada, todas las magníficas herramientas de las que me he beneficiado durante el desarrollo y escritura de esta tesis.

Prólogo

El diseño de sistemas de control multivariable robustos es una tarea compleja, que requiere un gran conocimiento de la materia por parte del diseñador. Además, en muchas ocasiones habrá que recurrir a procedimientos iterativos, de prueba y error, en los que será muy importante su experiencia.

Por otro lado, a principios de la década de los setenta, John Holland desarrolló los algoritmos genéticos, que son técnicas computacionales basadas en la teoría de la evolución, capaces de resolver complejos problemas de optimización. Posteriormente, se desarrollaron otras técnicas que, aunque basadas también en la teoría de la evolución, presentaban algunas variantes respecto a los algoritmos genéticos. Todas estas técnicas se engloban dentro de lo que son los algoritmos evolutivos, o estrategias evolutivas.

Se debe tener en cuenta también, que la mayoría de los problemas de optimización son problemas multiobjetivo, es decir, existen múltiples objetivos que deben satisfacerse de forma simultánea, muchos de ellos contrapuestos, por lo que la mejora de uno de ellos traerá consigo el empeoramiento de algún otro, debiéndose finalmente llegar a una solución de compromiso en la que todos los objetivos lleguen a cumplirse en un cierto grado.

Los algoritmos evolutivos, por sus características propias, se han mostrado como un método muy eficaz para resolver problemas de optimización multiobjetivo. Los problemas de control pueden encuadrarse dentro de los problemas de optimización multiobjetivo, siendo por tanto candidatos para ser resueltos por medio de algoritmos evolutivos. Además, los algoritmos evolutivos poseen características que los hacen muy atractivos como método para resolver problemas de control, pues permiten trabajar desde un punto de vista más cercano al usuario, utilizar de forma simultánea el dominio temporal y el de la frecuencia, y considerar cualquier parámetro de interés en la sintonía del controlador, resultando por tanto un método muy flexible.

Todo lo expuesto motivó a orientar esta tesis hacia el estudio de la aplicabilidad de los algoritmos evolutivos a los problemas de control, concretándose en el diseño de

controladores para sistemas aeronáuticos y navales. El resultado es una metodología de amplia aplicación al diseño de controladores.

Durante el desarrollo de este trabajo, se aplicó la metodología al diseño de tres controladores que se comentarán a continuación, durante la explicación de la estructura de la tesis. Finalmente, debido al alto coste computacional que supone aplicar la metodología a problemas complejos, se llevó a cabo un estudio sobre la paralelización del algoritmo.

La tesis se estructura en los siguientes capítulos:

Capítulo 1: Se repasan los algoritmos evolutivos y su aplicación al control, incluyéndose un estudio sobre el estado del arte en la materia.

Capítulo 2: Se revisan los términos, conceptos y métodos que se utilizan en control multiobjetivo mediante algoritmos evolutivos. Se desarrolla además un método propio para llevar a cabo la ordenación de la población en función de las aptitudes de sus cromosomas. Este método será empleado en los capítulos siguientes, en la resolución de los distintos problemas de control abordados.

Capítulo 3: En este capítulo se presentan, en primer lugar, los dos problemas de control a los que se aplicará la metodología, exponiéndose sus especificaciones de diseño. Posteriormente, se presenta la metodología utilizada, con especial consideración para la definición de la función de evaluación de cada uno de los problemas. Se propone además la utilización de una función de densidad de probabilidad especial para la obtención de los números aleatorios propios del método, realizándose pruebas para demostrar su conveniencia; los resultados de la pruebas se encuentran en los apéndices.

Capítulo 4: Se aplica la metodología al diseño de un controlador LQ. En este caso, los algoritmos evolutivos sustituirán al procedimiento de prueba y error, propio de las técnicas de diseño LQ para la determinación de las matrices de peso Q y R.

Capítulo 5: En este capítulo se utiliza la metodología desarrollada para sintonizar un controlador con una estructura (orden) determinada. A diferencia de lo realizado en el capítulo anterior, ahora se realiza la sintonía directa del controlador, sin el apoyo de ningún otro método. La validez de la metodología y del proceso de diseño vuelve a confirmarse con la resolución de este problema, correspondiente a un autopiloto de un avión comercial.

Capítulo 6: Se incrementa la complejidad del problema a resolver, haciendo ahora que sea el algoritmo evolutivo el que obtenga primero la estructura (orden) del controlador y luego la sintonice. En este caso se aplica el método al segundo de los problemas tratados, que corresponde a la reducción del índice de mareo en un buque de alta velocidad. Dado el alto coste computacional que supone, el programa se paralelizó y se ejecutó en un supercomputador con múltiples procesadores.

Capítulo 7: Debido a la alta carga computacional que puede suponer la aplicación de la metodología propuesta, en especial cuando aumenta la complejidad del problema a resolver, se realiza en este último capítulo un estudio sobre su paralelización, con objeto de poder acelerar los procesos. Se repasan, en primer lugar, las distintas arquitecturas que pueden utilizarse para la ejecución del algoritmo, particularizando finalmente el estudio al caso de un *clúster* de PCs.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Utilización de algoritmos genéticos y evolutivos en control. Estado del arte

1.1. Introducción.

A la hora de diseñar un sistema de control mediante métodos clásicos, se presentan una serie de dificultades como son el esfuerzo y el tiempo requeridos para el aprendizaje, la implementación y la aplicación de dichos métodos. Además, en ocasiones habrá que recurrir a procesos iterativos en los que la experiencia del diseñador será muy importante.

Con la aplicación de los algoritmos evolutivos al diseño de sistemas de control, se pretende soslayar los inconvenientes mencionados, dotando al ingeniero de control de herramientas más sencillas. Los algoritmos evolutivos permiten el diseño de sistemas de control con múltiples especificaciones (funcionamiento, restricciones, robustez, ...), considerándose todas ellas sin tener que recurrir a técnicas matemáticas complejas, y obteniéndose resultados muy aceptables. Uno de los puntos clave en el diseño del algoritmo, será el diseño de la *función de evaluación*, que se encargará de determinar cuáles son las soluciones más adecuadas de entre todas las disponibles.

Otra gran ventaja que presentan los algoritmos evolutivos, con respecto a otros métodos utilizados en el diseño de controladores, es la de poder tratar con cualquier tipo de función de evaluación a la hora de determinar la idoneidad de las distintas

soluciones potenciales al problema, pudiéndose por tanto diseñar una función que trate directamente con las especificaciones de diseño. Esto contrasta con lo que ocurre con otros métodos, en los que las especificaciones de diseño serán satisfechas de forma indirecta, tras la manipulación de otra serie de variables con las que no presentan, en muchos casos, una relación directa (considérense, por ejemplo, métodos en el dominio de la frecuencia, como QFT, cuando las especificaciones de diseño estén dadas en el dominio temporal).

La función de evaluación será implementada por el diseñador, pudiendo incluir en ella las variables que desee, sin limitaciones impuestas por el método. Se podrán incorporar de forma simultánea variables propias del dominio temporal y del dominio de la frecuencia, y si se incluye alguna medida de la robustez, se obtendrá como resultado un método de control robusto.

Los algoritmos evolutivos constituyen por tanto un método sencillo y flexible en la resolución de problemas de optimización, entre los que se encuentran los problemas de control.

En los siguientes apartados del presente capítulo se repasarán los algoritmos evolutivos y genéticos, y se verán ejemplos de la utilización de este tipo de algoritmos por diversos autores, dentro del ámbito del control.

1.1.1. Terminología.

La terminología empleada por los algoritmos evolutivos, procede de la genética y la evolución natural. Entre los términos más empleados se encuentran los siguientes:

Cromosoma Constituye una posible solución al problema, codificada de una forma adecuada.

Gen Son los elementos de los que están compuestos los cromosomas.

Alelo Valor adoptado por un gen determinado dentro del cromosoma.

Locus Posición de un gen determinado dentro del cromosoma.

Población Conjunto de cromosomas que representan soluciones potenciales al problema, y que son tratadas de forma simultánea por el algoritmo.

Generación Conjunto de cromosomas que componen la población en un instante t determinado.

Aptitud También conocida como *adecuación*. Es una medida de lo bueno que es un cromosoma como solución al problema a resolver.

Individuo Es empleado en ocasiones representando el mismo concepto que el término *cromosoma*. Algunos autores incluyen, junto a la codificación de la solución al problema, información relativa a su adecuación [57].

Función de evaluación o adecuación Función encargada de medir la aptitud de los distintos cromosomas.

Operadores genéticos Se encargarán de realizar transformaciones oportunas en los cromosomas padres, para obtener la descendencia. Los más empleados son los operadores de *mutación* y *cruce*.

1.2. Algoritmos genéticos, estrategias evolutivas y algoritmos evolutivos.

En los problemas de optimización, se trata de encontrar la mejor solución o *solución óptima*, en un *espacio de búsqueda* o *espacio de soluciones potenciales*. Cuando el problema no puede resolverse de forma analítica, pero el espacio de búsqueda es pequeño, es decir, el número de soluciones potenciales no es muy grande, un método de búsqueda exhaustiva en el que se consideren y prueben todas las posibles soluciones puede ser suficiente. Sin embargo, en espacios de búsqueda grandes los métodos exhaustivos dejan de ser manejables. En este caso, cobran fuerza una serie de métodos de búsqueda especializados, entre los que se encuentran los algoritmos genéticos, las estrategias evolutivas y los algoritmos evolutivos. Estos métodos tienen de particular que dirigirán la búsqueda hacia zonas prometedoras del espacio de soluciones potenciales, en las que hay más posibilidades de encontrar una solución óptima.

A continuación se dará un breve repaso a estos métodos, que tienen en común el estar inspirados en la teoría de la evolución natural, o en el principio de supervivencia del más apto.

1.2.1. Algoritmos genéticos.

Los algoritmos genéticos son técnicas de optimización estocástica que se aplican a la resolución de problemas complejos. Fueron creados por John Holland [63], con la

intención de imitar el proceso de evolución natural, mediante el cual las especies han ido adaptándose al medio en el que viven.

Las características de la evolución natural, llevaron a Holland a plantearse la posibilidad de crear un algoritmo que, imitando dichas características e introducido en un ordenador, diera lugar a una técnica que permitiera la resolución de problemas complejos en la forma en la que lo haría la naturaleza por medio de la evolución.

Para la resolución de problemas mediante algoritmos genéticos, en primer lugar hay que codificar las soluciones candidatas en forma de *cromosomas*, manteniéndose en todo instante t una *población* de individuos o cromosomas $P(t) = \{x_1^t, x_2^t, \cdots, x_N^t\}$.

Los cromosomas $(x_1^t, x_2^t, \cdots, x_N^t)$, serán evaluados para comprobar lo buenos que son como solución al problema, obteniéndose de dicha evaluación una medida de su *aptitud*. Para ello se utilizará una *función de evaluación*, *función de aptitud*, o *función de adecuación*, que será la encargada de medir la idoneidad de cada cromosoma, representando por tanto la conexión entre el algoritmo genético y el problema a resolver. La función de evaluación juega el mismo papel en los algoritmos genéticos que el que juega el entorno en la evolución natural. La interacción de un individuo con su entorno proporciona una medida de su idoneidad, lo mismo ocurre con la interacción de un cromosoma con la función de evaluación.

A partir de la aptitud, se llevará a cabo un *proceso de selección* mediante el cual se consigue que los cromosomas más aptos tengan más posibilidades de reproducirse y transmitir sus características beneficiosas a las nuevas generaciones. Por el contrario, los cromosomas menos aptos tendrán una menor probabilidad de reproducirse, con lo que sus características se irán perdiendo de generación en generación.

Sobre los cromosomas seleccionados se aplicarán una serie de *operadores*, para originar los descendientes. Los operadores habituales son el operador de *cruce* y el operador de *mutación*. Mediante el primero, los cromosomas padres transmiten a sus hijos parte de sus características, mientras que el segundo introduce diversidad.

Se espera que tras repetir el proceso anterior un número finito de veces, se produzca la convergencia hacia una *solución óptima* al problema. Se deberá establecer por tanto una condición de finalización que permita determinar si el algoritmo debe continuar o concluir.

Los pasos generales de un algoritmo genético pueden resumirse en:

1. Creación, de forma aleatoria, de la población inicial de cromosomas.

2. Evaluación de la aptitud de cada cromosoma de la población.
3. Si se cumplen las condiciones de finalización, se detiene el algoritmo y se toma el mejor cromosoma como solución al problema. Si no, el proceso continúa.
4. Selección de las parejas de cromosomas que formarán el conjunto de padres de la nueva generación.
5. Aplicación de los operadores de cruce y mutación para obtener una nueva generación de cromosomas hijos.
6. Se vuelve al paso 2.

Los algoritmos genéticos desarrollados por John Holland, presentan además las siguientes características:

1. Los cromosomas están codificados en forma de cadenas de números binarios: 0's y 1's.
2. Utilizan únicamente los operadores de mutación y cruce binarios.
3. Disponen de una justificación teórica de su funcionamiento: el *teorema de los esquemas*.

1.2.2. Estrategias evolutivas.

Las estrategias evolutivas constituyen también un método que imita los principios de la evolución natural, para resolver problemas de optimización de parámetros. Fueron desarrolladas en Alemania, en la década de los 60, por dos estudiantes de la Technical University of Berlin: Ingo Rechenberg y Hans-Paul Schwefel [114]. En estas estrategias, se comenzó utilizando como representación de las posibles soluciones al problema cadenas de números reales y la mutación como único operador.

En un principio, el algoritmo consistía en una población de un único individuo sobre el que se aplicaba el operador de mutación. El individuo estaba representado por un par de vectores de números reales $\boldsymbol{v} = (\boldsymbol{x}, \boldsymbol{\sigma})$, donde el primer vector, $\boldsymbol{x}$, representaba un punto del espacio de búsqueda, y el segundo vector, $\boldsymbol{\sigma}$, representaba las desviaciones estándar de los elementos del primero. Las mutaciones se realizaban sustituyendo $\boldsymbol{x}$ por medio de la expresión:

$$\boldsymbol{x^{t+1}} = \boldsymbol{x^t} + N(\mathbf{0}, \boldsymbol{\sigma})$$

donde $N(\mathbf{0}, \boldsymbol{\sigma})$ es un vector de números aleatorios que se ajustan a distribuciones normales de media cero y desviaciones estándar especificadas en el vector $\boldsymbol{\sigma}$ (se basaba en la observación biológica de que los cambios pequeños ocurren con más frecuencia que los grandes). El descendiente reemplazaba al padre si tenía una aptitud mejor y satisfacía todas las restricciones.

Posteriormente, se probó a incrementar el tamaño de la población, lo que permitió añadir alguna nueva característica:

- Los individuos de la población poseían la misma probabilidad de tener descendencia.
- Posibilidad de introducir un operador de recombinación (llamado "cruce" en la comunidad de los algoritmos genéticos) que a partir de dos padres (seleccionados aleatoriamente) produjera un descendiente.
- El operador de mutación permanecía sin cambios.
- El individuo más débil era sustituido por el descendiente.

En este tipo de algoritmos se han empleado distintas formas de generar y seleccionar los descendientes que formarán parte de la nueva población. Una de estas formas se nombra mediante la notación $(\mu+1)-ES$, y en ella se origina un descendiente a partir de μ individuos. La extensión natural a la forma anterior es la que se conoce como $(\mu+\lambda)-ES$, donde μ individuos producen λ descendientes; de los $\mu+\lambda$ resultantes se eliminan λ para mantener una población de μ individuos.

Otra forma es la conocida como $(\mu,\lambda)-ES$, en la cual los μ individuos de la población producen λ descendientes ($\lambda > \mu$) y por un proceso selectivo se eligen a los μ individuos que formarán la nueva población, a partir únicamente de los λ descendientes. Mediante la utilización de esta última estrategia, la vida de un individuo queda limitada a una única generación.

1.2.3. Comparación entre los algoritmos genéticos y las estrategias evolutivas.

La diferencia básica entre los algoritmos genéticos y las estrategias evolutivas, está en sus dominios de aplicación. Las estrategias evolutivas surgieron como un método de optimización numérica, mientras que los algoritmos genéticos fueron formulados como un método de búsqueda de propósito general.

La mayor similitud entre ambos métodos estriba en que los dos mantienen poblaciones de soluciones potenciales y hacen uso del principio de la supervivencia del más apto, presente en la teoría de la evolución natural. Sin embargo, existen diferencias entre estas aproximaciones:

1. La forma de representar los individuos es diferente. Como ya se ha mencionado, los algoritmos genéticos clásicos utilizan vectores de números binarios, mientras que en las estrategias evolutivas se emplean vectores de números reales.

2. También hay diferencias en los procedimientos de selección. Por un lado el orden en que ocurren el proceso de selección y la aplicación de los distintos operadores: en los algoritmos genéticos el proceso de selección es previo a la aplicación de los operadores genéticos, mientras que en las estrategias evolutivas ambos procesos ocurren en el orden inverso. Por otro lado, el proceso de selección es aleatorio en los algoritmos genéticos, dependiendo las probabilidades de selección de cada individuo de su aptitud, o de su posición en el *ranking*, mientras que es determinístico en las estrategias evolutivas: se seleccionan los μ mejores de $\mu + \lambda$ (para $(\mu + \lambda) - ES$) o de λ (para $(\mu, \lambda) - ES$).

3. Los parámetros de la reproducción (probabilidades de cruce y de mutación) permanecen constantes en los algoritmos genéticos durante el proceso evolutivo, mientras que cambian con el tiempo en las estrategias evolutivas.

1.2.4. Algoritmos evolutivos.

En el apartado anterior, se vio que los algoritmos genéticos y las estrategias evolutivas, tenían marcadas diferencias, sin embargo pronto aparecieron investigadores que empezaron a combinar las características de uno y otro en la solución de sus problemas. De gran utilidad ha resultado, por ejemplo, la utilización de la codificación de los cromosomas en forma de vectores de números reales, en lo que eran los algoritmos genéticos.

La codificación binaria utilizada por los algoritmos desarrollados por Holland, presenta algunos inconvenientes cuando se aplica a problemas numéricos multidimensionales de alta precisión. Por ejemplo, para 100 variables tomando valores dentro del intervalo [-500, 500] y con una precisión de 6 decimales, los cromosomas con codificación binaria necesitarían 3000 genes. Para problemas con estas características, los algoritmos genéticos dejan de comportarse de la forma deseada.

La utilización de otro tipo de codificación, empleando alfabetos más amplios, se ha mostrado sin embargo ideal para trabajar con este tipo de problemas. En una implementación con números reales, cada cromosoma será un vector con tantos números reales como elementos tenga el vector solución. La precisión en este tipo de codificación dependerá de la máquina sobre la que se implemente, pero normalmente será mucho mayor que la de la representación binaria. Evidentemente, se podría ampliar la precisión de la representación binaria, añadiendo más bits, pero a costa de ralentizar considerablemente el algoritmo.

Además, el hecho de emplear una codificación que se acerca más al problema a resolver, facilita el diseño de operadores especiales, que incorporen un conocimiento específico de dicho problema.

El teorema de los esquemas (que será expuesto más adelante), válido para los algoritmos genéticos, no es aplicable a los algoritmos evolutivos, sin embargo, los resultados experimentales obtenidos mediante su utilización superan en muchos casos a los obtenidos por los algoritmos genéticos. Ya en [33] y [87] se pueden encontrar pruebas comparativas entre la codificación binaria y la de números reales, llegándose a la conclusión de que para determinados problemas, este último tipo de codificación presenta mejores resultados, siendo el algoritmo más rápido y proporcionando una mayor precisión, especialmente en dominios grandes, donde la codificación binaria requeriría representaciones prohibitivamente largas.

Para referirse a estos algoritmos más generales, con una codificación de los cromosomas no restringida a los valores binarios y un conjunto más amplio de operadores, diversos autores han utilizado una variada terminología, por ejemplo: Janikow y Michalewicz [68] utilizaron la expresión *algoritmos genéticos especializados*, el propio Michalewicz en trabajos posteriores utilizó los términos *algoritmos genéticos no estándares* [85] y *algoritmos genéticos modificados* [86]. Finalmente, se comenzó a utilizar la expresión de *algoritmos evolutivos* para referirse a ellos [54].

En el presente trabajo, se empleará la denominación de algoritmos evolutivos para este tipo de algoritmos más generales y se considerarán a los algoritmos genéticos, encuadrados dentro de los evolutivos como un caso particular.

1.3. Algoritmos genéticos y evolutivos. Fase de diseño.

A la hora de diseñar un algoritmo evolutivo, se deberán seguir los siguientes pasos básicos:

1. Elegir una representación adecuada para las soluciones del problema. Será necesario algún conocimiento de la naturaleza del problema a resolver.
2. Crear una población inicial de posibles soluciones. Lo habitual es obtener la población inicial de forma aleatoria. Habrá que tener cuidado con los valores asignados cuando tengan que satisfacerse un conjunto de restricciones.
3. Elegir un método de selección.
4. Diseñar la función de evaluación. Requerirá conocimientos específicos del problema a resolver, por lo que puede ser la tarea más complicada si el algoritmo está siendo diseñado por un programador general, sin conocimientos específicos sobre la materia.
5. Elegir los operadores a utilizar. Dependerá de la representación elegida para los cromosomas.
6. Decidir los valores que tomarán los distintos parámetros del algoritmo, como el tamaño de la población y probabilidades de aplicación de los distintos operadores.

En las secciones que siguen se profundizará un poco en cada uno de los pasos anteriores y se comentarán otros temas de interés. No se pretende ser exhaustivo, por existir ya abundante bibliografía sobre la materia (como manuales de referencia se pueden consultar [57], [87], [124], y [44], y también, en castellano, [66]). Únicamente se pretende mostrar la base sobre la que se asentará la metodología de trabajo desarrollada.

1.3.1. La población inicial. Los cromosomas.

El primer paso del algoritmo evolutivo será la creación de los cromosomas que compondrán la población inicial. Ésta estará formada por un conjunto de N cromosomas que representarán posibles soluciones al problema, codificadas de una forma útil para el algoritmo $P(0) = \{x_1^o, \cdots, x_N^o\}$. Lo habitual es generar los cromosomas de la población inicial de forma aleatoria, es decir, creando un conjunto de vectores cuyos elementos serán números obtenidos aleatoriamente.

El tamaño de la población es un parámetro importante. Un tamaño demasiado pequeño traerá consigo una convergencia demasiado rápida y que ésta se produzca hacia un óptimo local, mientras que un tamaño demasiado grande requerirá un enorme coste computacional.

Como ya se ha comentado, los algoritmos genéticos tradicionales utilizan cadenas de bits como forma de representación estándar. Los cromosomas deberán contener la información correspondiente a todas las incógnitas del problema, teniendo asociado cada una de dichas incógnitas un número de genes del cromosoma, dependiendo dicho número de la precisión deseada.

Ejemplo 1.1 Supóngase que se quieren determinar los valores de las variables x_1 y x_2 que den lugar a un mínimo en la función f definida a continuación:

$$f(x_1, x_2) = -cos(x_1) \cdot cos(x_2) \cdot e^{-(x_1-\pi)^2(x_2-\pi)^2}$$

donde $-50 \leq x_i \leq 50$ para $i = 1, 2$.

Puesto que en la codificación empleada por los algoritmos genéticos cada gen es un bit, un posible cromosoma sería el siguiente:

$$(1\,0\,0\,1\,0\,1\,1\,1\,0\,1\,|\,0\,0\,0\,1\,1\,1\,1\,1\,0\,1)$$

donde los 10 primeros bits corresponderían a x_1 y los 10 últimos a x_2. Con el número de bits empleado, cada variable podrá tomar 2^{10} valores diferentes entre los límites establecidos en el enunciado. Por tanto, el intervalo $[-50, 50]$ se discretizará en 1024 valores y la máxima precisión será 100/1023 (el primer valor de los 1024 es el cero). Con esto, los números codificados en binario del posible cromosoma anterior serían equivalentes a $(9.1398, -37.781)$. Si fuera necesaria una mayor precisión, habría que incrementar el número de bits de los cromosomas, con el consiguiente aumento en el coste computacional.

En el caso de un algoritmo evolutivo, se podrían emplear como cromosomas vectores con números reales como elementos, que tendrían una precisión bastante mayor que 100/1023 sin necesidad de incrementar el tamaño de los cromosomas.

1.3.2. La función de evaluación, función de aptitud o función de adecuación.

Como ya se ha comentado, la *función de evaluación* juega el mismo papel que el entorno en la evolución natural. Mediante la función de evaluación se obtiene una medida de lo adecuadas que son las distintas soluciones al problema. Esta medida se conoce como *aptitud*.

En el diseño de la función de evaluación se requerirán unos conocimientos específicos del problema a resolver. Una ventaja de los algoritmos evolutivos, respecto a otros métodos de optimización, es que pueden emplear cualquier tipo de función de evaluación, no necesariamente analítica. En la aplicación a los problemas de control, permitirá tratar directamente con las especificaciones de diseño.

La función de evaluación será la función a optimizar por medio del algoritmo. Los algoritmos evolutivos se han mostrado eficaces en la optimización de funciones complejas, con gran número de óptimos locales, debido a su capacidad de trabajar con varias soluciones potenciales de forma simultánea. La figura 1.1 representa la función correspondiente al ejemplo 1.1, y en ella se puede apreciar la naturaleza del problema.

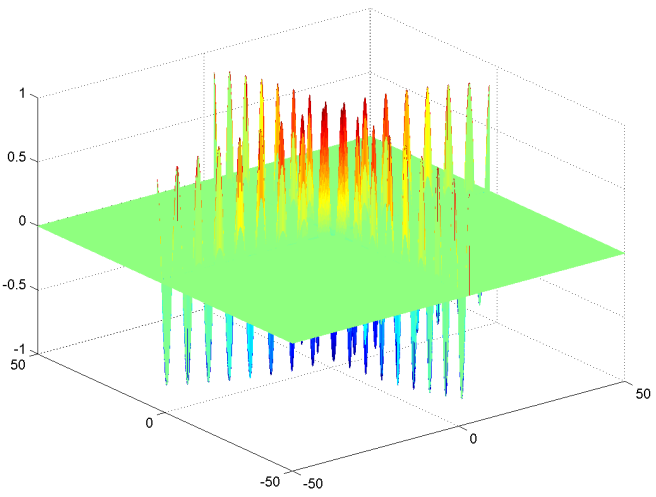


Figura 1.1: Representación de la función a optimizar en el ejemplo 1.1.

1.3.3. Manejo de restricciones.

Un tema importante dentro de un algoritmo evolutivo, es la forma de tratar las violaciones de las restricciones del problema. Dentro del espacio de búsqueda o de soluciones potenciales, se puede distinguir entre individuos *factibles* que son aquellos que satisfacen las restricciones, e individuos *no factibles*, que son los que no las satisfacen. Michalewicz [88] y Coello [25] han realizado revisiones sobre el tema. A continuación se indican las principales formas de tratar el problema:

- No permitir la presencia de cromosomas *no factibles*, eliminando de la población a aquellos cromosomas que no satisfagan las restricciones. Tiene el inconveniente de eliminar posibles caminos para la convergencia, no permitiendo la presencia en la población de individuos que, si bien no cumplen las restricciones, pueden contener genes con características beneficiosas que serían trasmitidas a la descendencia durante la fase de reproducción. Además, en algunos casos puede ocurrir que el algoritmo no sea capaz de obtener soluciones válidas al problema, pues si éstas presentan una probabilidad de ocurrencia baja, no se dispondrá de una forma de hacer converger las soluciones hacia regiones favorables del espacio de búsqueda.

- Permitir la presencia en la población de los cromosomas que infrinjan las restricciones, pero asignándoles una penalización en la función de evaluación. Este método soluciona los problemas mencionados en el punto anterior, presentando a cambio un incremento del coste computacional, ya que se puede perder mucho tiempo en evaluar soluciones no válidas al problema. La forma habitual de aplicar la penalización será mediante la inclusión de un término aditivo en la función de evaluación. Así la función de evaluación $f(\boldsymbol{x})$ original, se transformaría en otra función de la forma

  $$F(\boldsymbol{x}) = f(\boldsymbol{x}) + Q(\boldsymbol{x})$$

  donde el término $Q(\boldsymbol{x})$ representa la penalización.

- Diseñar esquemas de representación y operadores específicos, de forma que todas las posibles soluciones al problema que se generen sean *factibles*. El problema de este método es que no es de aplicación general. El tipo de representación y los operadores dependerán del problema a resolver. Se deberá comenzar generando una población inicial de soluciones *factibles*.

- Convertir la solución no válida en otra que sí lo sea, por medio de un algoritmo reparador. Este método también puede requerir un alto coste computacional y

será fuertemente dependiente del problema particular a resolver. El buen funcionamiento de este método depende de encontrar una heurística de reparación adecuada. Habrá problemas en los que no resulte viable la reparación de soluciones *no factibles*.

- Dentro de los algoritmos evolutivos multiobjetivo, que se estudiarán en el capítulo 2, las restricciones pueden ser tratadas como objetivos adicionales. Ésta es la técnica empleada en los trabajos de la presente tesis. En [121] se añadieron dos nuevos objetivos relacionados con las restricciones: uno basado en una función de penalización y el otro indicando el número de restricciones que han sido violadas.

1.3.4. La selección de padres.

La selección de padres es fundamental dentro del proceso evolutivo, eligiéndose en ella los cromosomas que van a reproducirse y transmitir sus características a la nueva generación. Existen distintas técnicas de selección, mediante las cuales aquellos miembros de la población más aptos tendrán una probabilidad mayor de ser seleccionados y reproducirse. Aparte de los manuales de referencia [57], [87] y [44], mencionados en la sección 1.3, puede ampliarse la información sobre los métodos de selección en Wieczorek y Czech [125], y en Motoki [90].

A la hora de diseñar el procedimiento de selección hay que llegar a una situación de compromiso entre lo que se llama la *presión selectiva* y la *diversidad*. Una presión selectiva alta significa que los mejores cromosomas de la población tienen una probabilidad muy alta de tener descendientes, mientras que para el resto de los cromosomas la probabilidad de descendencia es muy baja. Esto da lugar a que, en pocas generaciones, la población se vuelva muy homogénea y con poca diversidad, dando lugar en muchos casos a una convergencia hacia óptimos locales alejados del mejor óptimo global.

A continuación se verán los dos métodos de selección más empleados.

1.3.4.1. Método de la ruleta.

El método de la ruleta consiste básicamente en lo siguiente:

1. Se suman los índices de aptitud de todos los miembros de la población, llamando al resultado *aptitud total*.
2. Se genera n, un número real aleatorio comprendido entre 0 y la aptitud total.

3. Se selecciona el primer miembro de la población cuya aptitud, añadida a las aptitudes de los miembros precedentes de la población sea mayor o igual que n.

4. Se repite el procedimiento desde el paso 2, hasta obtener el número de padres deseados.

Si bien se trata de una técnica aleatoria, la probabilidad que tiene cada miembro de la población de ser elegido es directamente proporcional a su índice de aptitud. Al aplicar este procedimiento durante varias generaciones, se irán eliminando las características de los cromosomas menos aptos, permaneciendo el material genético de aquellos miembros de la población con una aptitud mayor.

Se ilustrará este método con un ejemplo. En la tabla 1.1 se muestra el índice de aptitud de diez cromosomas y la suma acumulada con los índices de aptitud de los cromosomas precedentes, a lo que se llamará *suma parcial*.

Cromosoma	1	2	3	4	5	6	7	8	9	10
Índice de aptitud	8.3	2.4	17.1	7	2.3	12.1	11.4	7.2	3.1	7.3
Suma parcial	8.3	10.7	27.8	34.8	37.1	49.2	60.6	67.8	70.9	78.2

Tabla 1.1: Índice de aptitud y suma parcial de las aptitudes de 10 cromosomas.

En la tabla 1.2 se muestran los cromosomas que serían elegidos por el método de la ruleta, para siete números obtenidos de forma aleatoria entre 0 y 78.2.

Número aleatorio	23.5	50.1	75.4	12.9	2.3	27.1	57.3
Cromosoma elegido	3	7	10	3	1	3	7

Tabla 1.2: Cromosomas seleccionados.

Como puede verse, el cromosoma número 3, que era el más apto, fue seleccionado en tres ocasiones y el número 7 que era el tercero en aptitud fue seleccionado en dos. Sin embargo, el carácter aleatorio del método hizo que el cromosoma número 6 no fuese seleccionado, aún siendo el segundo más apto.

La selección de padres proporcional a la aptitud, presenta problemas cuando hay gran disparidad entre las aptitudes de los miembros de la población. Así por ejemplo, si hay un cromosoma con una aptitud muy buena —en un caso extremo podría ser incluso de distinto orden de magnitud— la probabilidad de selección del resto de los cromosomas de la población sería remota, existiendo una gran presión selectiva que

originaría una rápida convergencia de la población hacia las características de este *supercromosoma*, que por otro lado no será necesariamente el óptimo global buscado.

Para solucionar este problema, se podría modificar la función de evaluación original, de forma que diera lugar a resultados más homogéneos y evitando de esta forma el dominio ejercido por unos pocos cromosomas.

Otra forma de solucionar este problema es trabajar, en lugar de con la aptitud, con un valor asignado en función de la posición que ocupa cada cromosoma en la población (método basado en el *ranking*), una vez ordenada ésta en función de las aptitudes.

1.3.4.2. El método del torneo o concurso.

Este método tiene una implementación más simple que el de la ruleta y al no estar basado directamente en las aptitudes, no presenta los problemas que se han comentado de aquél. El procedimiento es el siguiente:

1. Se elige de forma aleatoria un número de concursantes de entre los cromosomas que componen la población.
2. Se selecciona como padre al más apto de todos los concursantes.
3. Se vuelve al paso 1 hasta completar la población de padres.

Se ilustra esta técnica con un ejemplo.

> **Ejemplo 1.2** Sea 2 el número de concursantes a escoger para cada torneo. Si se quieren seleccionar 10 padres de una población de 20 cromosomas, harían falta 10 concursos o torneos con 2 participantes en cada uno, elegidos de forma aleatoria. En la tabla 1.3 se resume el proceso.

El número de concursantes es un parámetro importante a tener en cuenta al aplicar esta técnica, pues un aumento en el número de concursantes hace aumentar la probabilidad de que los mejores cromosomas sean seleccionados, con el consiguiente incremento de la presión selectiva. Lo más habitual es utilizar 2 concursantes por torneo.

Una ventaja añadida de este método, compartida por otros métodos basados en el *ranking*, es que al no tratar directamente con la aptitud, ésta no tiene por que ser necesariamente un escalar, pudiendo ser un vector (esta característica puede ser aprovechada en problemas de optimización multiobjetivo).

Concursantes seleccionados aleatoriamente	Padres seleccionados
(9, 3)	3
(12, 6)	6
(10, 10)	10
(20, 3)	3
(4, 7)	4
(13, 3)	3
(14, 13)	13
(17, 5)	5
(10, 8)	8
(5, 1)	1

Tabla 1.3: Método de torneo o concurso.

1.3.5. Operadores genéticos.

Los operadores genéticos son los encargados de obtener los descendientes a partir de los cromosomas padres. Generalmente, cada operador genético tendrá asignado un parámetro con el valor de la probabilidad de ser utilizado. Suponiendo que p_c es la probabilidad de un determinado operador de cruce y N el tamaño de la población, el número esperado de cromosomas que sufrirán cruce será $p_c \cdot N$. Para cada cromosoma de la población se generará un número aleatorio $0 \leq r \leq 1$, si $r < p_c$ el cromosoma sufrirá cruce.

Los operadores genéticos pueden aplicarse sobre los cromosomas de forma conjunta (aplicando primero el operador de cruce y luego, sobre los cromosomas resultantes, el operador de mutación), o de forma independiente (decidiendo de alguna forma qué operador aplicar sobre un cromosoma y aplicar únicamente ese). La aplicación por separado presenta alguna ventaja: por un lado, el procedimiento será más simple, al no tener que aplicar el operador de mutación a los cromosomas sobre los que ya se ha aplicado el de cruce. Por otro lado y como ventaja fundamental, es más fácil ampliar la lista de operadores genéticos, sobre todo cuando no se emplea la codificación binaria de los cromosomas, pudiéndose añadir sin dificultad nuevos operadores dependientes del problema.

Como ya se ha mencionado anteriormente, los operadores habituales son las mutaciones y los cruces, existiendo múltiples variantes de ambos en el caso de los algoritmos evolutivos. Para muchos investigadores, la mutación ha sido meramente una forma de introducir diversidad en la población, mientras que el cruce era el auténtico caballo de batalla, una técnica que actúa rápidamente para combinar lo que es bueno en la

población inicial, y que continúa funcionando mientras el algoritmo se ejecuta, como menciona Davis en [33].

Sea S el conjunto de todas las posibles soluciones al problema, las mutaciones serán aplicaciones de la forma $m : S \to S$, es decir, que a partir de modificaciones en un único cromosoma se obtendrá un nuevo cromosoma descendiente. Los cruces, por su parte, serán aplicaciones de la forma $c : S \times S \times \cdots \times S \to S$, obteniéndose en este caso un nuevo descendiente, a partir de la recombinación de las características de varios individuos (normalmente dos).

Se verá a continuación un amplio conjunto de operadores.

1.3.5.1. Cruce en un punto.

El operador de cruce en un punto es aplicable tanto a cromosomas con codificación binaria, como a aquellos codificados en forma de vectores de números reales. Trabaja de la siguiente forma:

1. Se seleccionan dos padres de la población.
2. Se obtiene, de forma aleatoria, un número entero comprendido entre 1 y la longitud del cromosoma menos 1.
3. Se intercambian los genes de los cromosomas padres situados en las posiciones siguientes al número aleatorio obtenido en el paso anterior. Los cromosomas obtenidos serán los cromosomas hijos.

Se muestra a continuación un sencillo ejemplo:

1. Se seleccionan los siguientes cromosomas padres:

$C_1 =$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
$C_2 =$	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}

2. Se obtiene el número 5 de forma aleatoria.
3. Los cromosomas hijos mantendrán los 5 primeros genes de los padres, intercambiando los 7 últimos, siendo los cromosomas resultantes:

$H_1 =$	a_1	a_2	a_3	a_4	a_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}
$H_2 =$	b_1	b_2	b_3	b_4	b_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}

1.3.5.2. Cruce uniforme.

El operador de cruce uniforme también es aplicable tanto a cromosomas con codificación binaria, como a aquellos codificados en forma de vectores de números reales. Trabaja de la siguiente forma:

1. Se seleccionan dos padres de la población.
2. Se obtienen, de forma aleatoria, varios números enteros entre 1 y la longitud del cromosoma.
3. Se intercambian los genes de los cromosomas padres situados en las posiciones de los números aleatorios obtenidos en el paso anterior. Los cromosomas obtenidos serán los cromosomas hijos.

Se muestra a continuación un sencillo ejemplo:

1. Se seleccionan los siguientes cromosomas padres:

$C_1 =$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
$C_2 =$	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}

2. Se obtienen de forma aleatoria los siguientes números: 3, 5, 7 y 8.
3. En los cromosomas hijos se intercambiarán dichos genes de los padres, siendo los cromosomas resultantes:

$H_1 =$	a_1	a_2	b_3	a_4	b_5	a_6	b_7	b_8	a_9	a_{10}	a_{11}	a_{12}
$H_2 =$	b_1	b_2	a_3	b_4	a_5	b_6	a_7	a_8	b_9	b_{10}	b_{11}	b_{12}

1.3.5.3. Cruce aritmético.

El operador de cruce aritmético es típico en los programas que utilizan cromosomas codificados en forma de vectores de números reales, y no es aplicable a las cadenas de bits. Funciona de la siguiente forma:

Sean C_1 y C_2 dos vectores de números reales —cromosomas— y a un número real tal que $a < 1$. Los cromosomas hijos se obtendrán a partir de los padres de la siguiente forma:

$$H_1 = a \cdot C_1 + (1 - a) \cdot C_2$$

Para obtener un segundo hijo se puede hacer:

$$H_2 = (1 - a) \cdot C_1 + a \cdot C_2$$

En un espacio de búsqueda convexo, este operador asegura que si los cromosomas padres pertenecen a dicho espacio, los cromosomas hijos también lo harán. Por otro lado, se deduce fácilmente que para valores de a próximos a 1 o próximos a 0, lo que hace el algoritmo es explorar las vecindades de los cromosomas padres.

Con la introducción de este operador en el algoritmo surge la necesidad de asignar un valor al parámetro a. Para determinar un valor adecuado para este parámetro habría que realizar un estudio sobre la forma en que afecta su valor al algoritmo. Una posible alternativa a esto es hacer que a sea un número aleatorio comprendido entre 0 y 1, obtenido cada vez que se vaya a aplicar el operador.

1.3.5.4. Cruce heurístico.

El operador de cruce heurístico, al igual que el de cruce aritmético, es sólo aplicable a los algoritmos con cromosomas codificados en forma de vectores de números reales. Funciona de la siguiente forma:

Sean C_1 y C_2 dos vectores de números reales —cromosomas— y r un número aleatorio comprendido entre 0 y 1. Los cromosomas hijos se obtendrán a partir de los padres de la siguiente forma:

$$H = r \cdot (C_2 - C_1) + C_2$$

Siendo C_2 al menos igual de apto que C_1.

1.3.5.5. Mutación aleatoria.

Ya se ha comentado que durante el funcionamiento del algoritmo, se debe mantener una situación de compromiso entre la presión selectiva y la diversidad de la población. La mutación es una de las formas de aportar diversidad a la población, permitiendo la exploración de nuevas regiones del espacio de búsqueda.

Este operador es aplicable tanto a cadenas de bits como a vectores de números reales. En el caso de la codificación binaria el proceso consiste en reemplazar algunos de los genes de forma aleatoria haciéndolos igual a 1 si su valor original era 0 y viceversa. Por otro lado, en el caso de la codificación en forma de números reales, el procedimiento consiste en seleccionar de forma aleatoria los genes a cambiar y asignarles un nuevo valor también aleatoriamente.

La probabilidad de que se produzca mutación suele establecerse como bastante menor que la probabilidad de que se produzca cruce.

1.3.5.6. Mutación no uniforme.

El operador de mutación no uniforme es sólo aplicable a los algoritmos con cromosomas codificados en forma de vectores de números reales. Mediante este operador se persigue el ajuste fino de las soluciones al problema. Se trata de una mutación aditiva, mediante la cual al valor de algún gen del cromosoma padre se le suma o resta un valor previamente determinado. Dicho valor determinado dependerá del número de generaciones producidas hasta ese momento.

Sean $C_1 = \{x_1, \cdots, x_m\}$ un cromosoma y t el número de generaciones producidas. Se selecciona x_k para esta mutación; el resultado de aplicar el presente operador será $H = \{x_1, \cdots, x'_k, \cdots, x_m\}$, donde

$$x'_k = \begin{cases} x_k + \Delta(t, UB - x_k) & \text{si un dígito obtenido aleatoriamente es 0} \\ x_k - \Delta(t, x_k - LB) & \text{si un dígito obtenido aleatoriamente es 1} \end{cases}$$

y LB y UB son los límites inferior y superior del dominio de la variable a_k. La función $\Delta(t, y)$ devuelve un valor en el rango $[0, y]$, tal que la probabilidad de que $\Delta(t, y)$ se aproxime a cero aumenta cuando t aumenta. Esta propiedad hace que el operador busque en el espacio de búsqueda: uniformemente al principio y localmente en etapas

posteriores del algoritmo. Una posible función es la indicada en Michalewicz [87]:

$$\Delta(t, y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b \tag{1.1}$$

siendo r un número real aleatorio comprendido entre 0 y 1, T el número total de generaciones a obtener, y b un parámetro del sistema que determina el grado de dependencia con el número de generaciones.

1.3.6. La nueva generación.

Una de las características interesantes de los algoritmos evolutivos en la resolución de problemas de optimización, es la de trabajar simultáneamente con varias soluciones, lo que permite explorar a la vez zonas diferentes del espacio de soluciones potenciales, y evitar que el algoritmo converja hacia un óptimo local. Sin embargo, la limitación de tener que trabajar con poblaciones finitas atenúa el efecto anterior, por lo que podría llegar a producirse la convergencia comentada hacia un óptimo local.

Para evitar este fenómeno se deben potenciar medidas que favorezcan la diversidad en la nueva población. Ya se ha hablado de estas medidas en los apartados correspondientes a la selección de padres y al operador de mutación aleatoria. Una medida que se ha mostrado eficaz, mejorando la respuesta de los algoritmos genéticos como optimizadores globales, es la de incluir en cada nueva generación un pequeño número de inmigrantes totalmente aleatorios. Éste es el método propuesto en [60].

Por otro lado, será también importante al crear una nueva generación, la determinación de los cromosomas que deben morir y no pasar a formar parte de la nueva población. Cuando ha tenido lugar la reproducción, y se ha creado una nueva generación, surge la necesidad de eliminar los cromosomas que sobran en función del tamaño de la población deseado. Existen varias formas de hacerlo y no se puede decir, de forma general, que una sea mejor que las demás, pues esto va a depender de las características del problema.

Una forma típica de selección es la conocida como $(\mu + \lambda)$ en la que μ padres y λ descendientes, compiten por sobrevivir pero sólo los μ mejores pasan a formar parte de la nueva población (equivalente a la $(\mu + \lambda) - ES$ que se vio para las estrategias evolutivas). Esta técnica presenta una gran presión selectiva, por lo que en general, no será adecuada en problemas con gran número de óptimos locales. Se dice que es *elitista*, por haber cromosomas que permanecen en la población generación tras generación.

Otra forma de selección es la conocida como *generacional* o (μ, λ) (equivalente a la $(\mu, \lambda) - ES$ que se vio para las estrategias evolutivas). En ella los descendientes, una vez producidos, sustituyen completamente a los padres. Esta técnica presenta una menor presión selectiva y muestra un comportamiento mejor en el caso de múltiples óptimos locales. Se puede hacer elitista si se mantiene al mejor de los cromosomas de generación en generación, evitando la pérdida de buenas soluciones al problema que, de otro modo, podrían no volver a obtenerse. Una solución intermedia sería guardar estas buenas soluciones, pero fuera de la población, de forma que no afecten al comportamiento del algoritmo.

También será importante evitar la creación de múltiples copias de un cromosoma en la población. Las múltiples copias de *supercromosomas* en la población, dan lugar a un incremento no deseado de la presión selectiva.

1.3.7. Condiciones de finalización del algoritmo.

Las condiciones de finalización determinan cuándo debe concluir la búsqueda por parte del algoritmo. En muchas ocasiones, las condiciones se reducen a un número de generaciones, en cuyo caso el algoritmo concluirá tras haberse alcanzado dicho número.

Otros métodos tratan de comprobar si se ha producido la convergencia hacia la solución óptima, finalizando el algoritmo cuando las posibilidades de que se produzca una mejora sean muy pequeñas. Estos métodos requerirán más recursos computacionales.

También puede emplearse un método interactivo, en el que el usuario compruebe la evolución del algoritmo tras un número de generaciones y decida si debe seguir ejecutándose por haber posibilidad de nuevas mejoras. Este método requerirá almacenar el estado del algoritmo, para que éste pueda continuar desde el punto en el que se produjo la parada. Se empleó ampliamente durante el desarrollo de la presente tesis.

1.3.8. Asignación de valores a los parámetros del algoritmo.

Algunos autores han investigado la influencia de ciertos parámetros en el funcionamiento del algoritmo, habiéndose estudiado cómo afectan el tamaño de la población y las probabilidades de ocurrencia de los distintos operadores.

De Jong [35] realizó un estudio en el que buscaba valores para los parámetros de un algoritmo genético tradicional, que fuera de aplicación en un conjunto de problemas

de optimización numérica. Encontró de forma manual una serie de valores para los parámetros del cruce en un punto y mutación binaria, manteniendo el tamaño de la población constante en 100 individuos.

En 1986, John Grefenstette [59] utilizó un algoritmo genético para encontrar buenos valores para estos parámetros, utilizando de nuevo una población de 100 individuos. Los parámetros que obtuvo mejoraban ligeramente a los obtenidos por De Jong en su estudio anterior.

En 1989 David Schaffer, Lawrance Eshelman, Richard Caruana y Rajarshi Das [112] llevaron a cabo una búsqueda exhaustiva para los valores de los parámetros. Tras más de 12 meses de tiempo de CPU en sus pruebas, pudieron demostrar que los valores óptimos de los parámetros varían de un problema a otro. Las pruebas las llevaron a cabo para una población de 100 individuos, representación binara, y utilizando como operadores los habituales de cruce en un punto y mutación binaria.

También en 1989, Davis [32] desarrolló un método de búsqueda en la que los valores de los parámetros se iban adaptando durante la ejecución del algoritmo. Para realizar las modificaciones de los parámetros, se tenía en cuenta la cantidad de cromosomas de la población que habían dado lugar a una mejora a partir del uso de un determinado operador. El procedimiento era aplicado hacia atrás, de forma que los padres y antecesores anteriores (cada vez en menor medida) eran también recompensados, pues podían haber dado lugar a cromosomas beneficiosos, sin que se produjeran mejoras directas.

De los trabajos mencionados se desprende que no parecen existir valores de aplicación general, sino que éstos son dependientes del problema a resolver. Aún así, coinciden en utilizar tamaños de población en torno a los 100 cromosomas, valores altos de las probabilidades de cruce y valores bajos de las probabilidades de mutación.

El tema de los valores de los parámetros sigue siendo un área de investigación activa, dando lugar más recientemente a trabajos como los de Jonsson y Malec [69], Fogel *et al* [52], y Pongcharoen *et al* [105].

1.4. El teorema de los esquemas.

La justificación teórica del funcionamiento de los algoritmos genéticos, recae en la noción de esquema. Los esquemas se construyen tras añadir el carácter comodín $*$ al alfabeto de los genes (0 y 1). Un esquema representa a todas las cadenas que tienen, todos los bits que son distintos de $*$, iguales. Por ejemplo, las cadenas 1 0 1 0 0 0 1 1 y

01100111 están representadas por el esquema $**100*11$ y el esquema $********$ representa a todos los esquemas de longitud 8. Cada esquema representa a 2^r cadenas, donde r es el número de caracteres comodín presentes. Por otro lado, cada cadena de longitud m está representada por 2^m esquemas diferentes, resultado de dejar en cada posición a su valor inicial, o sustituirlo por el carácter comodín $*$.

Hay dos importantes propiedades de los esquemas: el *orden* y la *longitud definida*, sobre las que se formula el *Teorema de los Esquemas* (se puede encontrar más información sobre este teorema en Michalewicz [87], Goldberg y Sastry [58], y Haijun *et al* [62]).

Definición 1.1 *El orden del esquema S (expresado como $o(S)$) es el número de caracteres con valor 0 o 1, que aparecen en la cadena que compone el esquema, es decir, el número de caracteres del esquema que son diferentes del comodín.*

Ejemplo 1.3 Los siguientes esquemas:

$$\begin{array}{lcccccccc} S_1 = & * & * & 0 & 1 & 0 & 1 & * & 0 \\ S_2 = & 1 & * & 0 & 1 & 1 & * & * & * \\ S_3 = & 1 & 1 & 0 & 1 & 0 & 1 & 1 & * \end{array}$$

tienen los órdenes: $o(S_1) = 5, o(S_2) = 4, o(S_3) = 7$. El esquema S_3 es el más específico, y el S_2 es el más general.

La noción de orden es útil para el cálculo de la probabilidades de supervivencia de los esquemas durante las mutaciones.

Definición 1.2 *La longitud definida de un esquema S (expresada como $\delta(S)$) es la distancia entre el primero y el último de los caracteres no comodines.*

En los esquemas del ejemplo anterior tendríamos

$$\delta(S_1) = 8 - 3 = 5; \delta(S_2) = 5 - 1 = 4; \delta(S_3) = 7 - 1 = 6$$

La noción de longitud definida es útil para el cálculo de las probabilidades de supervivencia de los esquemas durante los cruces.

Otra propiedad de los esquemas, es su aptitud en el instante t, que se expresará como $eval(S, t)$; se define como la aptitud media de todos los cromosomas de la población

que se ajustan al esquema S. Suponiendo que en el instante t hay p cromosomas en la población que se ajustan al esquema S, se tendría:

$$eval(S,t) = \sum_{j=1}^{p} \frac{eval(x_j)}{p} \tag{1.2}$$

Durante el proceso de selección, un cromosoma x_i tendrá una probabilidad $p_i = eval(x_i)/F(t)$ de ser seleccionado, siendo $F(t) = \sum_{i=1}^{p} eval(x_i)$, la aptitud total del conjunto de la población en el instante t.

Llamando $\xi(S,t)$ al número de cromosomas que se ajustan al esquema S en el instante t, y $\overline{F(t)} = F(t)/PopSize$ a la aptitud media de la población, el número esperado de cromosomas que se ajusten al esquema S en el instante $t+1$, $\xi(S,t+1)$ vendrá dado por la expresión:

$$\xi(S,t+1) = \xi(S,t) \cdot eval(S,t)/\overline{F(t)} \tag{1.3}$$

Si la aptitud media del esquema es mayor que la aptitud media de la población, lo esperado será que el número de cromosomas que se ajusten a S en $t+1$ sea mayor al de cromosomas que se ajustan a S en t.

Pero la selección no es la única que interviene en el proceso evolutivo. A ésta habrá que añadirle la fase reproductiva, en la que se aplicarán los operadores genéticos: mutación y cruce.

La longitud definida de un esquema juega un papel significativo en la probabilidad de supervivencia de dicho esquema. Durante el cruce se elegirá de forma aleatoria una posición de entre las $m-1$ posibles. La probabilidad de que se produzca la destrucción del esquema en un cruce, vendrá dada por la expresión:

$$p_d(S) = \frac{\delta(S)}{m-1} \tag{1.4}$$

y la probabilidad de supervivencia al cruce, por tanto, será:

$$p_s(S) = 1 - \frac{\delta(S)}{m-1} \tag{1.5}$$

Si se tiene en cuenta la probabilidad de producirse cruce, p_c, nos quedará:

$$p_s(S) = 1 - p_c \cdot \frac{\delta(S)}{m-1} \tag{1.6}$$

y como puede darse el caso de que un cruce no rompa el esquema, porque los genes intercambiados tengan el mismo valor, habrá que sustituir el símbolo *igual* por el de *mayor o igual*, quedando finalmente:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1} \tag{1.7}$$

Vemos que longitudes definidas pequeñas, favorecen la probabilidad de supervivencia de un esquema.

El siguiente operador a considerar es el de mutación. Durante la aplicación de este operador se cambiará el valor de un gen del cromosoma. Evidentemente, para que el esquema sobreviva, no podrá cambiar ninguno de los valores fijos del esquema. El número de genes importantes para la supervivencia, será igual al *orden* del esquema.

Sea p_m la probabilidad que tiene un gen de mutar, la probabilidad de sobrevivir será entonces $1 - p_m$. La probabilidad de supervivencia de un esquema S será, por tanto:

$$p_s(S) = (1 - p_m)^{o(S)} \tag{1.8}$$

Puesto que p_m suele ser pequeña, la expresión anterior se puede aproximar por

$$p_s(S) \approx 1 - o(S) \cdot p_m \tag{1.9}$$

Combinando los efectos de la selección, cruce y mutación, se llega a la siguiente ecuación:

$$\xi(S, t+1) \geq \xi(S, t) \cdot \frac{eval(S, t)}{\overline{F(t)}} \cdot [1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m] \tag{1.10}$$

que representa el número esperado de cromosomas en la siguiente generación, que se ajustan al esquema S, en función del número actual de cromosomas, la aptitud relativa del esquema, su *orden* y su *longitud definida*. La expresión anterior lleva a concluir lo siguiente:

Corolario 1.1 (Teorema de los Esquemas) *Los esquemas con aptitud por encima de la media, de longitud definida corta y orden bajo, tenderán a incrementarse en generaciones sucesivas de un algoritmo genético.*

1.5. Aplicaciones en control. Estado del arte.

La eficiencia de los algoritmos evolutivos en la solución de problemas de optimización, llevó pronto a considerarlos como un método alternativo para resolver problemas de sistemas de control. Los algoritmos evolutivos, presentan una serie de ventajas respecto a métodos clásicos que, si bien son efectivos en determinadas situaciones, presentan limitaciones de aplicabilidad. Algunos de estos métodos son:

- **Programación lineal**. De aplicación únicamente en problemas con funciones lineales.
- **Métodos de optimización no lineal basados en el gradiente**. Aplicable a problemas con funciones no lineales. Las funciones tienen que ser continuas y diferenciables, al menos en la vecindad del punto óptimo. Los métodos basados en el gradiente también funcionan con problemas lineales, aunque la programación lineal será un método mucho más eficiente en este caso.
- **Búsqueda exhaustiva**. De aplicación en aquellos casos en que el número de posibles soluciones al problema es limitado.

Frente a estos métodos, los algoritmos evolutivos presentan la ventaja de ser independientes del tipo de función a optimizar y poderse aplicar en el caso de un número ilimitado de posibles soluciones al problema.

Los ejemplos de utilización de los algoritmos evolutivos en control, se pueden clasificar en dos grupos: por un lado estarían las aplicaciones de naturaleza *off-line*, que son la mayoría, en las que los parámetros son determinados con anterioridad a la puesta en funcionamiento del sistema, y por otro aquellas de naturaleza *on-line*, en las que los parámetros requeridos son determinados durante el funcionamiento del sistema a controlar. Los métodos *on-line* presentan dos problemas fundamentales: por un lado, el alto coste computacional que suelen requerir, que hace difícil que los parámetros estén disponibles cuando se requieran, y por otro lado, y más importante si cabe, es el hecho de que la naturaleza estocástica de este tipo de algoritmos podría dar lugar a que la mejor solución obtenida no cumpliera con unos mínimos de bondad, llevando a producir

incluso daños en el sistema. Estos inconvenientes han llevado a que casi la totalidad de trabajos realizados se hayan enfocado a métodos *off-line*, quedando los métodos *on-line* únicamente para trabajos de investigación pura.

Dentro de los sistemas de control, los algoritmos evolutivos se han utilizado ampliamente en aplicaciones que incluyen el diseño de controladores, la identificación de modelos, análisis de estabilidad robusta y diagnosis de fallos. Fleming y Purshouse realizaron un estudio en el que están recogidos algunos de estos trabajos [50].

1.5.1. Diseño de controladores.

En el diseño de controladores, los algoritmos evolutivos se han empleado principalmente para obtener los parámetros del controlador. También se han combinado con controladores borrosos y neuronales para formar un esquema de control inteligente.

A principios de los 90, se comenzó a investigar sobre el uso de los algoritmos evolutivos como medio para sintonizar controladores PID. Oliveira *et al* [94] utilizaron un algoritmo genético estándar para obtener unas estimas iniciales de los parámetros del controlador PID. Aplicaron su metodología a una variedad de sistemas lineales invariantes en el tiempo.

Wang y Kwok [123] adaptaron un algoritmo genético a la sintonía de un controlador PID. Hicieron hincapié en los beneficios que aportaba la flexibilidad en la elección de la función de coste, y aludieron al concepto de optimalidad de Pareto, que aportaba la potencialidad necesaria para tratar de forma simultánea con múltiples objetivos (esto se tratará con más extensión en el siguiente capítulo).

Vlachos *et al* [122] aplicaron un algoritmo genético a la sintonía de controladores PI descentralizados para procesos multivariables. El comportamiento del controlador fue definido en el dominio temporal, para las respuestas del sistema de lazo cerrado ante la introducción de unas señales de referencia.

Onnen *et al* [95] aplicaron los algoritmos genéticos a la determinación de una secuencia de control en un sistema de control predictivo basado en modelo. Se prestó especial atención a sistemas no lineales con restricciones en la entrada. Se desarrollaron unos operadores y una codificación que previnieran la generación de soluciones no factibles.

Aranda *et al* [5] utilizaron los algoritmos evolutivos para realizar la sintonía completa del controlador de vuelo de un avión. Se incluyeron medidas de robustez en la

función de evaluación, obteniéndose con ello controladores robustos.

Más recientemente, Dessalegne y Nicklow [39] han aplicado los algoritmos evolutivos al diseño de un sistema de control óptimo para minimizar las fluctuaciones del nivel de agua en el canal de Illinois. Ouyang *et al* [99] utilizaron un controlador de aprendizaje evolutivo de tipo PD robusto para el seguimiento de trayectorias en un sistema no lineal y variable en el tiempo. Rocha *et al* [108] emplearon un algoritmo evolutivo para obtener control óptimo en un proceso de fermentación de un producto bio-farmacéutico.

Una aproximación diferente al diseño de controladores mediante algoritmos genéticos, es la aplicación de la metodología de una forma indirecta. El algoritmo genético manipula los parámetros de un proceso de diseño de controladores ya establecido. En un diseño LQG, se puede emplear un algoritmo evolutivo para buscar los valores de las matrices de peso que mejor cumplan las especificaciones de diseño. En [6] Aranda *et al* aplicaron esta metodología al diseño de un controlador LQ para la estabilización y autopiloto de un avión.

También han sido aplicados con éxito en el campo del control H-infinito. Chen y Cheng [21] propusieron un controlador H-infinito con una estructura especifica. El algoritmo genético se utilizó para buscar buenos valores para los parámetros del controlador dentro del dominio admisible.

Los algoritmos genéticos multiobjetivo (MOGA), que serán vistos con más detalle en el capítulo siguiente, también han sido empleados con éxito en una variedad de problemas. En ellos se establecen una serie de objetivos, bien en el dominio temporal o en el de la frecuencia, resultando una función objetivo de tipo vectorial. En uno de estos estudios, Fonseca y Fleming [54], aplicaron un MOGA a la optimización del regulador de velocidad de la turbina de baja presión de un motor Rolls-Royce Pegasus. Andrés-Toro *et al* [3], por su parte, utilizaron un MOGA para controlar un proceso de fermentación de cerveza.

Las investigaciones también se han dirigido hacia los llamados *sistemas de control inteligente*. Las técnicas más populares son el *control borroso* y el *control neuronal*. En control borroso, puede emplearse un algoritmo genético para generar la base de reglas y para ajustar los parámetros de la función de pertenencia asociada. En control neuronal, se puede emplear un algoritmo genético para determinar los valores de los pesos. También se han mostrado capaces de optimizar la topología de una red neuronal. Se presentan a continuación algunos ejemplos de utilización.

Ichikawa y Sawa [65] sustituyeron directamente un controlador convencional por

una red neuronal. Los pesos fueron obtenidos por medio de un algoritmo genético. Cada individuo de la población representaba una distribución de pesos para la red neuronal.

Dentro de la aplicación de los algoritmos genéticos al control borroso, se distinguen dos categorías: sintonía de la función de pertenencia, y elección de la base de reglas. Los partidarios de la primera aproximación suelen argüir que la forma de las reglas se puede conocer a priori, siendo la obtención de la función de pertenencia asociada lo que presenta una mayor dificultad. Además, el uso de una base de reglas estática reduce de forma significativa la complejidad computacional, siendo ésta una causa importante de la popularidad de esta aproximación.

Tzes *et al* [119] aplicaron un algoritmo genético a la sintonía *off-line* de funciones de pertenencia gausianas, desarrollando un modelo borroso que describía la fricción en un motor de corriente continua.

Los métodos evolutivos, también se han aplicado en la generación de reglas de control, en situaciones donde un conjunto de reglas razonables no es evidente. Matsuura *et al* [84] utilizaron un algoritmo genético para obtener un control óptimo de evaluación sensorial de un proceso de molido de sake. El algoritmo genético aprendía reglas por un mecanismo de inferencia borroso, que subsecuentemente generaba la trayectoria de referencia para un controlador PI basado en evaluación sensorial. Varsek *et al* [120] también emplearon los algoritmos genéticos para desarrollar la base de reglas, en la solución del clásico problema de control de un péndulo invertido.

Más recientemente, Stirrup y Chipperfield [116] utilizaron un algoritmo genético multiobjetivo para sintonizar un controlador PI borroso, para el control de una planta de energía solar.

1.5.2. Identificación de sistemas.

Muchas aplicaciones de control requieren un modelo matemático del proceso a controlar. En ocasiones se podrá obtener un modelo preciso de forma analítica, a partir del conocimiento físico del sistema. Ésta será a menudo la aproximación apropiada para sistemas lineales, determinísticos, invariantes en el tiempo, de una entrada y una salida, de cuyos procesos exista un conocimiento suficiente. Sin embargo, la mayoría de sistemas del mundo real no encajan en esta categoría. En particular, serán a menudo no lineales y el conocimiento de sus procesos será pobre.

Existen técnicas bien desarrolladas para la estimación de parámetros de modelos lineales, y modelos no lineales pero lineales en los parámetros. Sin embargo, técnicas para la selección de la estructura del modelo, y la estimación de parámetros para modelos no lineales en los parámetros son actualmente materia de estudio e investigación.

La aplicación de los algoritmos evolutivos a la identificación de sistemas, se ha dirigido a dos problemas distintos:

- Selección de la estructura adecuada para el modelo.
- Estimación de los parámetros del modelo.

En 1992, Kristinsson y Dumont [71] aplicaron los algoritmos genéticos a la identificación de sistemas continuos y discretos. El algoritmo era utilizado para identificar de forma directa los polos y ceros, o para obtener los valores de parámetros físicos. La función de coste utilizada era el error entre la salida estimada y la real, para el par entrada-salida actual y 30 muestras anteriores. Kristinsson y Dumont informaron de resultados comparables o mejores que los ofrecidos por técnicas bien conocidas, requiriendo para ello un alto gasto computacional.

Uno de los problemas centrales en la identificación de un sistema, es la elección de los términos de entrada, salida y retardo que formarán parte del modelo. Los algoritmos genéticos proporcionan un método sencillo de búsqueda de términos que contribuyen de forma significativa en la salida del proceso.

Fonseca *et al* [53] utilizaron un algoritmo genético para seleccionar los términos de un modelo no lineal NARMAX (Nonlinear AutoRegressive Moving Average eXogenous) [75]. Cada gen del cromosoma estaba asociado con un término particular. Fonseca *et al* mejoraron la eficacia del operador de mutación haciendo depender la probabilidad de mutación de un gen, de la calidad del término asociado, medida como la varianza del residuo del modelo cuando el término es desechado. Los parámetros de cada modelo pueden ser estimados utilizando métodos estándar de mínimos cuadrados, al ser los modelos NARMAX lineales en los parámetros.

Luh y Wu [80], por su parte, utilizaron un algoritmo genético para la identificación de modelos NARX (NonLinear AutoRegressive eXogenous).

Como ya se ha comentado, en los casos donde el modelo a identificar es lineal en los parámetros, se pueden emplear técnicas estándar de mínimos cuadrados para obtener buenas estimas de los parámetros del modelo. En otras circunstancias, tales como para

modelos racionales no lineales, otras técnicas pueden proporcionar resultados superiores. Así, se han investigado metodologías basadas en algoritmos evolutivos como soluciones potenciales.

Choi *et al* [22], utilizaron estrategias evolutivas para identificar los parámetros de un modelo de fricción. Se utilizó una estructura predefinida para el modelo (basándose en los resultados existentes en la literatura). Los resultados se utilizaron en un sistema de control de compensación de la fricción.

Billings y Mao [16] aplicaron los algoritmos genéticos a la identificación de modelos racionales no lineales. Codificaron tanto la información de la estructura como la de los parámetros en cada individuo, para facilitar la optimización simultánea de ambos elementos. Los modelos racionales no son lineales en los parámetros y, por tanto, los parámetros no pueden ser estimados de forma precisa por métodos estándar. El modelo puede ser manipulado para conseguir que sea lineal en los parámetros, pero esto introduce severos problemas de ruido. Los autores obtuvieron buenos resultados para sistemas pequeños, pero no obtenían la convergencia para sistemas más complejos.

En trabajos más recientes, Pan *et al* [102] utilizaron un algoritmo genético para seguir las variaciones temporales de los retardos en un proceso de identificación *on-line*. Juang [70] utilizó un algoritmo genético junto con una red neuronal recurrente para la identificación de un sistema no lineal. Kumon *et al* [72] utilizaron un algoritmo genético en el diseño de un esquema de control adaptivo indirecto (IAC), el cual identifica el modelo y actualiza de forma automática los controladores; el algoritmo genético fue utilizado en la parte de identificación.

Aranda *et al* por su parte, utilizaron en [8] y [10] una técnica basada en algoritmos genéticos para obtener la estructura del modelo y los valores iniciales de los parámetros, obteniéndose el modelo final mediante un algoritmo de mínimos cuadrados no lineales con restricciones.

1.5.3. Análisis de sistemas.

Los algoritmos genéticos se han utilizado también en el diseño eficiente de sistemas de control robusto. En el análisis de la robustez, se evalúa la probabilidad de que un sistema en lazo cerrado exhiba un comportamiento inaceptable, ante una posible variación de parámetros. Se puede emplear un algoritmo genético para manipular una población de puntos en el espacio de diseño (cada uno de ellos correspondiendo a un compensador diferente). Marrison y Stengel emplearon esta técnica en el diseño de un

sistema de control robusto [83].

El diseño de sistemas robustos requiere una gran potencia de cálculo, por lo que se han considerado métodos que mejoren la velocidad de ejecución del algoritmo. Schubert y Stengel emplearon con este fin, una arquitectura de ordenadores en paralelo [113]. Un problema añadido que surge en la paralelización de este tipo de algoritmos, es la variabilidad del tiempo requerido para evaluar cada uno de los sistemas, que hace que las cargas de trabajo enviadas a cada proceso sean muy diferentes. Esto puede solucionarse mediante una distribución dinámica de las tareas.

Fadali *et al* por su parte [48], realizaron el análisis de estabilidad robusta de un sistema discreto, por medio de una búsqueda evolutiva de los polos del sistema que caen fuera del círculo unidad.

Más recientemente, Ge *et al* [56], utilizaron un procedimiento evolutivo en las primeras etapas de análisis y diseño de un sistema de parachoques para un automóvil.

1.5.4. Diagnosis de fallos.

La demanda general de sistemas más fiables y seguros, ha llevado a la aparición de los sistemas de diagnosis de fallos. Las tareas de un sistema de diagnosis de fallos, pueden separarse en tres áreas, concretamente:

- *Detección* de la presencia de un fallo.
- *Aislamiento* del fallo.
- *Identificación* o clasificación del fallo.

Una faceta particular en este campo de investigación, es el control tolerante a fallos, en el cual se puede generar una acción de control adecuada en la presencia de ciertas condiciones de fallo.

Una aproximación muy empleada consiste en la generación y análisis de *residuos*. Éstos son señales que reflejan las inconsistencias entre la operación nominal y la operación en presencia de fallos. Comúnmente se utilizarán observadores basados en el modelo, para la generación de dichos residuos. La competición entre los efectos de los fallos y las incertidumbres en el modelo (y las perturbaciones) representa el principal desafío para esta técnica. Los fallos grandes y repentinos tendrán un gran efecto so-

bre el residuo, mayor que los efectos debidos a las incertidumbres. Sin embargo, fallos incipientes pueden tener una respuesta más pequeña que la debida a las incertidumbres.

Patton *et al* formularon un sistema de aislamiento y detección de fallos como un problema de optimización multiobjetivo, en el cual la tarea era maximizar los efectos de los fallos sobre el residuo, mientras que se minimizaban los efectos de las incertidumbres [104]. Formularon una función de coste global y la optimizaron por medio de un algoritmo genético. La aproximación fue aplicada a la detección de fallos en sensores en un sistema de control de vuelo.

También se han aplicado los algoritmos genéticos con otros métodos no basados en el concepto de residuo. Marcu formuló la diagnosis de fallos de procesos basada en modelos, como un problema de clasificación y selección [82]. Se utilizó un algoritmo evolutivo multiobjetivo tanto para el aprendizaje *off-line* de regiones correspondientes a fallos conocidos y a condiciones libres de fallo, como para la identificación *on-line* de los coeficientes del proceso.

Painton y Campbell desarrollaron una técnica para mejorar la fiabilidad global del sistema, por medio de elecciones a nivel de componentes [101]. Para cada posible componente, se definía una probabilidad de fallo y un coste. Luego se utilizaba un algoritmo genético para buscar la elección de componentes que llevaba al sistema más fiable, en términos de tiempo medio antes del fallo, sin sobrepasar un coste límite establecido a priori. El algoritmo genético se mostró mucho más eficaz que el método alternativo: una búsqueda exhaustiva.

Más recientemente, Zhang *et al* [128] emplearon los algoritmos genéticos para optimizar la fiabilidad de un sistema; Echtle y Eusgeld [43] por su parte, utilizaron un algoritmo genético en el diseño de un sistema con tolerancia de fallos.

1.5.5. Aplicaciones *on-line*.

Ya se han comentado los problemas que presenta la aplicación *on-line* de los algoritmos evolutivos a los sistemas de control, lo que hace que haya que utilizarlos con precaución. Hay que tener en cuenta que para cada instante de tiempo se requerirá una señal de control, y que las acciones del mejor individuo de la población, en un algoritmo evolutivo, pueden tener graves consecuencias sobre el proceso. Esto será inaceptable en la mayoría de las aplicaciones.

En una aplicación en tiempo real, habrá un tiempo limitado para la obtención de

la siguiente señal de control a aplicar. Con la potencia de cálculo de los ordenadores actuales, un algoritmo evolutivo tendrá problemas en la mayoría de los casos, para converger hacia un valor adecuado.

En el caso de un controlador, el algoritmo evolutivo tendrá que proporcionar una señal de control aceptable para cada instante de control. Si la evolución sólo ha ocurrido durante un número pequeño de generaciones, el comportamiento de la población puede ser pobre aún.

Hay tres aproximaciones al uso de los algoritmos evolutivos *on-line* en control [78]:

- Utilizar un modelo del proceso.
- Utilizar directamente el proceso.
- Permitir la sintonía restringida de un controlador existente.

Lennon y Passino aplicaron algoritmos genéticos *on-line* en un sistema llamado de *control adaptivo de referencia de modelo genético* [74]. El algoritmo fue aplicado de forma particular a un sistema de frenos, tratando de asegurar que los frenos de un vehículo de motor se comportaban de forma consistente ante las acciones del conductor. El problema incluía condiciones de operación variables con el tiempo, debido a las variaciones en la temperatura de los frenos. Se utilizó un algoritmo genético para encontrar la ganancia de un controlador existente de retardo-adelanto. El mejor individuo era seleccionado para controlar la planta. Las aptitudes estaban basadas en la precisión futura, empleando para su obtención un modelo del proceso de frenado. Los parámetros de este modelo también se obtuvieron por medio de algoritmos genéticos. Como medida de seguridad, se incorporaron a la población plantas y controladores fijos, que serían utilizados en el peor de los casos.

Chang y Sim desarrollaron un sistema para reducir el consumo de energía de un tren [19]. Se trataba de un problema complejo con tres objetivos (puntualidad, calidad de viaje y consumo de energía) y muchas variables, incluyendo la distancia entre estaciones, y las condiciones de operación actuales (horario del tren, carga de pasajeros y voltaje en las vías). Antes de que el tren saliera hacia su estación de destino, un algoritmo genético obtenía una tabla de control del coste que era referenciada por el tren en tiempo real.

Pocas aplicaciones se han desarrollado mediante algoritmos genéticos *on-line* para sistemas reales de control. Uno de ellos fue implementado por Ahmad *et al* para la regulación de la temperatura en un sistema calefactor mediante la sintonía de los parámetros de un controlador PI [1]. Los objetivos eran conseguir la temperatura deseada

tan rápido como fuera posible y minimizar el sobreimpulso. Existe un conflicto entre estos dos requerimientos ya que la mejora en uno de ellos trae consigo el empeoramiento del otro. Se empleó por ello como función de coste la suma ponderada del error en la regulación y el cambio en la acción de control.

Además del diseño de controladores, también la identificación de sistemas y la diagnosis de fallos son dos áreas de activa investigación. En este sentido puede verse el trabajo de Kristinsson y Dumont sobre identificación de sistemas *on-line* [71].

Más recientemente, dentro de las aplicaciones *on-line*, Cuppertino *et al* en [28], [29] y [30], han aplicado los algoritmos genéticos al diseño de distintos tipos de controladores para motores de corriente continua. Barrido y Dadios [13] por su parte, utilizaron un algoritmo genético para determinar la posición de un robot en movimiento dentro de un campo de fútbol. Teng *et al* [117] a su vez, emplearon un algoritmo genético en la autosintonía *on-line* de los parámetros de un controlador PID, en un sistema de control de nivel de líquido. Y Naeem *et al* [91] utilizaron los algoritmos genéticos para el diseño de un control predictivo basado en modelo *on-line* para un vehículo submarino. Y dentro del campo de la identificación de sistemas, Chow y Rad [23] utilizaron los algoritmos genéticos en el diseño e implementación de un método de identificación *on-line* borroso; el comportamiento del algoritmo de identificación fue evaluado mediante la simulación de un sistema no lineal.

1.6. Conclusiones.

Los algoritmos evolutivos han demostrado ser un método eficiente en el diseño de sistemas de control. Frente a los métodos clásicos de diseño presentan algunas ventajas como son la mayor sencillez de aplicación y permitir trabajar en el dominio del usuario, manejando conceptos en el dominio temporal que le resultan más intuitivos.

Entendiendo el proceso de búsqueda del controlador como un problema de optimización, el hecho de que los algoritmos evolutivos trabajen simultáneamente con múltiples soluciones al problema facilita la convergencia hacia un óptimo global. Además, como se verá en el siguiente tema, el trabajo con múltiples soluciones de forma simultánea resulta positivo en la solución de problemas de optimización multiobjetivo.

Dentro de los algoritmos evolutivos utilizados en el diseño de sistemas de control, se puede diferenciar entre los que trabajan *on-line* y los que lo hacen *off-line*. Los primeros apenas se han utilizado en sistemas reales dados los problemas que pueden presentar a

la hora de obtener una solución satisfactoria en un tiempo determinado. Los últimos sí han sido utilizados ampliamente.

Los continuos progresos en la tecnología de los ordenadores, permitirá explotar de forma más eficiente el potencial de la metodología de los algoritmos evolutivos. En general se requiere una gran cantidad de cálculos para obtener una buena solución al problema. La disponibilidad de ordenadores con una relación prestaciones/coste cada vez mayor, junto con los avances en arquitecturas en paralelo supondrán un gran beneficio para los algoritmos evolutivos.

Capítulo 2

Optimización multiobjetivo mediante algoritmos evolutivos

2.1. Introducción.

Un gran número de problemas en ciencia e ingeniería, requieren la optimización simultánea de múltiples funciones objetivo. Habrá que optimizar por tanto una función de la forma $f : S \rightarrow T$, donde $S \subset \mathbb{R}^n$ y $T \subset \mathbb{R}^k$; pero el problema está en que normalmente no existe un elemento de S que produzca un óptimo de forma simultánea para cada una de las k funciones objetivo. Esto es debido a un conflicto entre objetivos, que hace que la mejora de uno de ellos dé lugar a un empeoramiento de otro. Habrá que llegar por tanto a una situación de compromiso, en la que todos los objetivos sean satisfechos en un grado aceptable, desde el punto de vista de diseño.

A diferencia de los problemas de optimización con un único objetivo, el concepto de *óptimo* es relativo y la toma de decisión sobre cuál es la mejor solución al problema deberá incorporarse al algoritmo.

En términos matemáticos, el problema de optimización multiobjetivo, puede establecerse de la siguiente forma:

Encontrar un vector $\boldsymbol{x^*} = [x_1^*, x_2^*, \cdots, x_n^*]^T$, que satisfaga las m restricciones:

$$g_i(\boldsymbol{x}) \geq 0 \qquad i = 1, 2, \cdots, m \tag{2.1}$$

y las p restricciones:

$$h_i(\boldsymbol{x}) = 0 \qquad\qquad i = 1, 2, \cdots, p \tag{2.2}$$

y optimice la función vectorial

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_k(\boldsymbol{x})]^T$$

donde $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$ es el vector de variables de decisión.

En otras palabras, se desea determinar la solución particular $x_1^*, x_2^*, \cdots, x_n^*$, del conjunto S formado por todos los valores que satisfacen (2.1) y (2.2), que dé lugar a los valores óptimos para todas las funciones objetivo. Pero normalmente no existe una solución que optimice de forma simultánea todas las funciones objetivo.

A finales de la década de los 60 del siglo pasado, Rosemberg consideró la posibilidad de utilizar técnicas evolutivas, para tratar este tipo de problemas [109]. Desde entonces ha sido un área muy activa de investigación, en la que los algoritmos evolutivos han mostrado un buen comportamiento en la solución de problemas de optimización multiobjetivo; prueba de ello son las conferencias internacionales que se celebran periódicamente [27], y los tutoriales y libros que han ido apareciendo sobre este tema [129], [26], [34]. Esta materia es ahora conocida como Optimización Multi-Objetivo Evolutiva (de forma abreviada EMOO, a partir de los términos en inglés *Evolutionary Multi-Objective Optimization*).

Para que se produzca la evolución en este tipo de algoritmos, es preciso evaluar cada una de las soluciones (cromosomas) de la población con objeto de ver su aptitud y comparar los resultados de estas evaluaciones, de forma que aquellas soluciones más idóneas tengan más probabilidad de reproducirse y transmitir sus características a la descendencia. Pero cómo debe realizarse la comparación cuando se trate con aptitudes vectoriales, con múltiples objetivos satisfechos en mayor o menor medida.

El problema es, por tanto, decidir de alguna forma qué cromosomas son más aptos, con objeto de poder llevar a cabo la selección de los padres que deberán reproducirse y transmitir sus características a los cromosomas de la nueva generación. En los apartados siguientes se repasarán muchos de los métodos utilizados para tratar este problema (se puede encontrar un estudio más exhaustivo en [24]). Estos métodos se pueden clasificar en dos grupos: aquellos basados en el concepto de *dominancia de Pareto*, y otros métodos o técnicas no basados en dicho concepto. Además se propone un método propio en

el apartado 2.5, el *método de las prioridades variables para optimización multiobjetivo evolutiva*, que se utilizará posteriormente en los problemas tratados en la presente tesis.

2.2. Métodos basados en el concepto de dominancia de Pareto.

El concepto de *dominancia de Pareto* fue formulado en el siglo XIX por Vilfredo Pareto [103] y supone el origen de las investigaciones en la materia de optimización multiobjetivo. Se expondrá a continuación dicho concepto restringido al caso de un problema de minimización, la extensión al caso de un problema de maximización es trivial.

Definición 2.1 (Dominancia de Pareto) *Dado un vector* $\boldsymbol{u} = (u_1, \cdots, u_k)$*, se dice que domina a otro vector* $\boldsymbol{v} = (v_1, \cdots, v_k)$ *si y sólo si:*

$$\forall i \in \{1, \cdots, k\},\ u_i \leq v_i \quad y \ \exists i_0 \in \{1, \cdots, k\} \mid u_{i_0} < v_{i_0}$$

Definición 2.2 (Optimalidad de Pareto) *Una solución* $\boldsymbol{x}^*$ *se dice que es Pareto-óptima si y sólo si no existe otro vector* $\boldsymbol{x}$ *tal que* $\boldsymbol{v} = f(\boldsymbol{x}) = (v_1, \cdots, v_k)$ *domine a* $\boldsymbol{u} = f(\boldsymbol{x}^*) = (u_1, ..., u_k)$*.*

En otras palabras, la definición anterior dice que el punto $\boldsymbol{x}^*$ es un óptimo de Pareto si no existe un vector $\boldsymbol{x}$ que haga mejorar alguno de los objetivos —respecto a los valores obtenidos para $\boldsymbol{x}^*$— sin que empeore de forma simultánea alguno de los otros objetivos. En general, la solución en el sentido de Pareto al problema de optimización multiobjetivo no será única: la solución estará formada por el conjunto de todos los vectores no-dominados, a los que se conoce con el nombre de *conjunto de no-dominados*, o *frente de Pareto.*

En la figura 2.1 se representa, con trazo grueso, el Frente de Pareto de una función con 2 objetivos. El área sombreada **T** representa la imagen de dicha función objetivo. Se puede observar que no existe ningún punto perteneciente a **T** que mejore en el sentido de Pareto, a algún punto del Frente: eligiendo un punto de **T** de forma arbitraria, por ejemplo p_3, se puede trazar la vertical hasta obtener el punto de corte con el Frente de Pareto, en este caso p_1 ; dicho punto de corte siempre tendrá el mismo valor de f_1 y un valor mejor de f_2. También se puede observar que para 2 puntos cualesquiera del

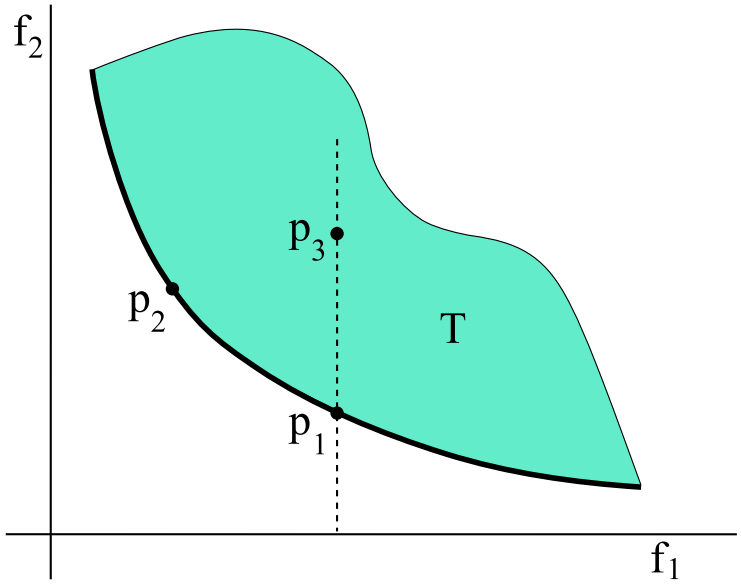


Figura 2.1: Frente de Pareto para una función con dos objetivos.

Frente de Pareto, nunca habrá uno que mejore de forma simultánea los dos objetivos, respecto al otro punto. Cogiendo por ejemplo los puntos p_1 y p_2, se observa que para p_1 mejora f_2, pero a costa de empeorar f_1.

Goldberg propuso por primera vez la idea de asignar las aptitudes de los distintos cromosomas de una población, basándose en el concepto de dominancia de Pareto [57]. Su idea era hacer evolucionar a la población hacia el Frente de Pareto, en un problema de optimización multiobjetivo. Para llevar a cabo la ordenación de la población, en primer lugar se buscan todos aquellos cromosomas que son *no-dominados* por el resto de la población, y se les asigna el orden 1. A continuación se vuelve a repetir el procedimiento anterior, pero sin considerar aquellos cromosomas a los que ya les ha asignado el orden. A los nuevos cromosomas encontrados como *no-dominados* se les asigna el orden 2. El procedimiento continúa hasta que la población está totalmente ordenada.

Además propuso un mecanismo de *nichos*, en el que los cromosomas que se encontraban muy próximos entre sí, eran penalizados con objeto de favorecer la diversidad y evitar la convergencia prematura.

2.2.1. MOGA.

Fonseca y Fleming por su parte, propusieron un algoritmo en el que la posición ocupada por un cromosoma de la población, estaba determinada por el número de cromosomas por los que era dominado [54]. Este algoritmo es conocido como MOGA (a partir de los términos en inglés: *Multi-Objective Genetic Algorithm*).

Considerando, por ejemplo, un cromosoma x_i de la generación t, que es dominado por $p_i^{(t)}$ cromosomas de su generación, su puesto en la población, una vez ordenada, vendría dado por la expresión:

$$\text{puesto}(x_i, t) = 1 + p_i^{(t)}$$

A todos los individuos *no-dominados* se les asigna el puesto 1. Una vez ordenada la población por puestos, se asignan las aptitudes de los distintos cromosomas en la siguiente forma:

1. Se ordena la población según los puestos obtenidos en el procedimiento anterior.
2. Se asignan las aptitudes a los cromosomas de la población, interpolando desde el mejor hasta el peor, de acuerdo con alguna función, normalmente lineal o exponencial, pero no necesariamente.
3. Se promedian las aptitudes asignadas a los cromosomas con el mismo puesto, de forma que todos ellos sean seleccionados con la misma probabilidad.

Para evitar la convergencia prematura, Fonseca y Fleming utilizaron también un mecanismo de nichos, pero utilizando los valores de la función de objetivos en lugar de los valores de los parámetros de los cromosomas.

2.2.2. NSGA.

Basándose en las ideas expuestas por Goldberg [57], mencionadas anteriormente en este capítulo, Srinivas y Deb desarrollaron un nuevo algoritmo conocido como NSGA (a partir de los términos en inglés: *Non-dominated Sorting Genetic Algorithm*) [115]. Los pasos del algoritmo son los siguientes:

1. Se buscan todos aquellos cromosomas que son *no-dominados* por el resto de la

población, y se les asigna el orden 1. Estos cromosomas serán conocidos como *primer frente de no-dominados*.

2. Se asigna un mismo valor de aptitud a cada uno de los cromosomas anteriores, con objeto de dar la misma probabilidad de reproducción a todos ellos.

3. Se modifica la aptitud de cada individuo, dividiendo su valor original por una cantidad proporcional al número de individuos existente en su proximidades, con objeto de favorecer la diversidad en la población.

4. Los individuos componentes del *primer frente de no-dominados* son ignorados temporalmente y con el resto de la población se vuelven a buscar aquellos individuos no-dominados. Al nuevo grupo de no-dominados se les asigna el orden 2 y se les denominará como *segundo frente de no-dominados*.

5. A cada uno de los individuos del *segundo frente de no-dominados* se les asigna una misma aptitud, que será menor que el valor mínimo de las aptitudes del *primer frente*.

6. Se modifican las aptitudes de los individuos del *segundo frente de no-dominados* en la misma forma que se hizo con los del primero.

7. Se repiten los procedimientos anteriores, hasta tener asignadas las aptitudes de todos los individuos de la población.

2.2.3. NPGA.

Horn y Nafpliotis propusieron un esquema de selección por torneo basado en la dominancia de Pareto [64], en lugar de la ordenación de no-dominados y un proceso de selección basado en el *ranking*. En este método, conocido como NPGA (a partir de los términos en inglés: *Niched Pareto Genetic Algorithm*), se seleccionan de forma aleatoria un número específico de individuos, que forman el *conjunto de comparación* para cada proceso de selección. Luego se eligen dos individuos de la población, también de forma aleatoria, y se comparan con los miembros del *conjunto de comparación* en cuanto a la dominancia en el sentido de Pareto respecto a las funciones objetivo. Si uno de ellos es no-dominado y el otro dominado, entonces el individuo no-dominado es seleccionado. Por el contrario, si ambos son no-dominados, o dominados, se realiza un recuento de nicho, simplemente contando el número de puntos en la población que se encuentran a menos de una cierta distancia de un individuo (σ_{share}). Será seleccionado el individuo para el que se obtenga un recuento menor. El número de individuos que compongan el

conjunto de comparación, será en este método un parámetro de gran influencia en el resultado final.

Más recientemente, Erickson *et al* han utilizado el método NPGA en el diseño de sistemas de gestión de la calidad de aguas subterráneas [45].

2.3. Métodos basados en la combinación de objetivos.

Estos métodos combinan de alguna forma los distintos objetivos, aportando al algoritmo evolutivo unas aptitudes escalares que le permiten realizar la ordenación de la población con rapidez. Han sido bastante utilizados y en lo que sigue se repasarán algunos de los más conocidos.

2.3.1. Método de la suma ponderada.

La suma ponderada es un método tradicional, ya utilizado por Zadeh en 1963 [127], que consiste en obtener la aptitud de cada cromosoma, mediante la suma de los valores de cada uno de las funciones objetivo, multiplicados por un coeficiente de peso. El problema de optimización multiobjetivo se transforma así en otro de optimización escalar, que para el caso de la minimización será de la forma

$$\text{mín} \sum_{i=1}^{k} w_i f_i(\boldsymbol{x})$$

donde $w_i \geq 0$ es el coeficiente de peso correspondiente al objetivo i. Los coeficientes de peso permitirán establecer la importancia relativa de los distintos objetivos y normalmente se escogen de forma que

$$\sum_{i=1}^{k} w_i = 1$$

El principal problema de este método, está precisamente en la adecuada elección de los valores de los coeficientes de peso, pues el resultado del problema de optimización es fuertemente dependiente de estos parámetros, y el diseñador tendrá que basarse en su intuición para elegir los valores más adecuados.

2.3.2. Método de la programación de metas.

Charnes y Cooper [20], y por su parte Ijiri [67], desarrollaron este método para un modelo lineal. En él se establecen a priori unas metas que deberán cumplir los objetivos. Estos valores se incorporan al problema como restricciones adicionales. El problema se reducirá a minimizar la suma de los valores absolutos de las desviaciones de los distintos objetivos a sus metas respectivas, es decir

$$\text{mín} \sum_{i=1}^{k} |f_i(\boldsymbol{x}) - M_i| , \qquad \text{sujeto a } \boldsymbol{x} \in S$$

donde M_i representa la meta del i-ésimo objetivo, y S es la región de soluciones factibles. El problema es que se están sumando valores que en general corresponden a propiedades que no pueden relacionarse entre sí directamente.

Con objeto de solucionar el problema anteior, se puede acudir a una formulación más general del método. En dicha formulación, el problema consistiría en minimizar la suma ponderada de las p-ésimas potencias de las desviaciones, es decir

$$\text{mín} \sum_{i=1}^{k} w_i \, |f_i(\boldsymbol{x}) - M_i|^p , \qquad \text{sujeto a } \boldsymbol{x} \in S$$

El problema de este método más general vuelve a ser la determinación de los coeficientes de peso, como ocurría con el método de la suma ponderada.

2.3.3. Método de las restricciones-ϵ.

Este método fue propuesto por Ritzel *et al* como una forma sencilla de resolver problemas de optimización multiobjetivo, por medio de algoritmos genéticos [107].

El método consiste en minimizar uno de los objetivos (el preferido o primario) y restringir el resto de objetivos, haciéndoles permanecer dentro de unos valores deseados, ϵ_1. El método puede formularse de la siguiente forma:

1. Encontrar el mínimo de la función objetivo r-ésima, es decir, encontrar $\boldsymbol{x}^*$ tal que

$$f_r(\boldsymbol{x}^*) = \min_{x \in S} f_r(\boldsymbol{x})$$

sujeto a restricciones adicionales de la forma

$$f_i(\boldsymbol{x}) \leq \epsilon_i \qquad \text{para } i = 1, 2, \ldots, k \text{ e } i \neq r$$

donde ϵ_i son los valores que no deben ser sobrepasados por las funciones objetivo.

2. Repetir el paso 1 para diferentes valores de ϵ_i. La búsqueda concluye cuando se encuentra una solución satisfactoria.

Puede ser necesario repetir el procedimiento anterior para diferentes índices r. Si se eligen límites ϵ_i demasiado bajos, puede ocurrir que no se encuentren soluciones adecuadas al problema; en este caso habrá que relajar el límite establecido para alguno de los objetivos.

Las debilidades de este método son el gran consumo de tiempo durante su ejecución, y el fallo, en ocasiones, a la hora de encontrar una solución apropiada.

2.4. Otros métodos no basados en la dominancia de Pareto.

Para evitar los problemas presentados por los métodos basados en la combinación de objetivos, diversos autores han buscado técnicas alternativas basadas en una manipulación apropiada de la población o en un uso especial de los objetivos. En lo que sigue se repasarán las más conocidas.

2.4.1. VEGA.

David Schaffer desarrolló el método conocido como VEGA (a partir de los términos en inglés: *Vector Evaluated Genetic Algorithm*) [111], en el que en cada generación, durante la selección, se creaban un número k de subpoblaciones, cada una de ellas relacionada con cada uno de los k objetivos. Estas subpoblaciones estarían formadas por N/k cromosomas, donde N representa el tamaño total de la población.

Las subpoblaciones eran mezcladas posteriormente, para obtener la población de N cromosomas sobre la que se aplicarían los operadores de mutación y cruce en la forma habitual.

Schaffer se dio cuenta de que aparecía un problema que en genética se conoce como

especiación. Este problema se produce debido a la selección de individuos que tienen un comportamiento excelente respecto a uno de los objetivos, pero que dejan bastante que desear en el resto. A su vez y de forma simultánea, individuos con un comportamiento aceptable para todos los objetivos, corren el riesgo de ser eliminados de la población, al no comportarse de forma excepcional para ninguno de dichos objetivos y ser considerados menos aptos por el algoritmo. La especiación es un problema indeseable, pues es contraria al objetivo de encontrar una solución de compromiso entre todos los objetivos. Schaffer propuso algunas heurísticas para tratar este problema, como articular algunas preferencias durante el proceso de selección, o favorecer el emparejamiento entre especies distintas, en lugar de utilizar el emparejamiento aleatorio propio de los algoritmos genéticos tradicionales.

2.4.2. Método de ordenación lexicográfica.

En el método de ordenación lexicográfica [15], los objetivos son ordenados por el diseñador en función de su importancia. La solución óptima $\boldsymbol{x^*}$ se obtiene minimizando las funciones objetivo, empezando con la más importante y continuando según el orden de importancia previamente establecido. Suponiendo que los objetivos estén ordenados de forma que $f_1(\boldsymbol{x})$ y $f_k(\boldsymbol{x})$ denoten al más importante y al de menor importancia respectivamente, el primer paso será

$$\text{mín } f_1(\boldsymbol{x})$$

sujeto a las restricciones

$$g_j(\boldsymbol{x}) \leq 0; \qquad j = 1, 2, \cdots, m$$

obteniéndose x_1^* y $f_1^* = f_1(x_1^*)$

El segundo paso será

$$\text{mín } f_2(\boldsymbol{x})$$

sujeto a las restricciones

$$g_j(\boldsymbol{x}) \leq 0; \qquad j = 1, 2, \cdots, m$$
$$f_1(\boldsymbol{x}) = f_1^*$$

obteniéndose x_2^* y $f_2^* = f_2(x_2^*)$

Este procedimiento se repetirá para todos los objetivos. Para el k-ésimo paso será

$$\text{mín } f_k(\boldsymbol{x})$$

sujeto a las restricciones

$$g_j(\boldsymbol{x}) \leq 0; \qquad j = 1, 2, \cdots, m$$
$$f_l(\boldsymbol{x}) = f_l^*; \qquad l = 1, 2, \cdots, i-1$$

obteniéndose x_k^* y $f_k^* = f_k(x_k^*)$

La solución obtenida finalmente, $\boldsymbol{x^*} = (x_1^*, \cdots, x_k^*)$, se tomará como solución al problema.

El inconveniente de este método es que puede llevar a soluciones muy buenas en relación al objetivo preferido, pero malas respecto al resto.

2.4.3. Método de min-max ponderado.

Fue desarrollado por Osyczka [96, 97, 98], Rao [106], y Tseng y Lu [118]. Una aplicación más reciente del método puede encontrarse en [77]. Este método compara las desviaciones relativas de los mínimos, obtenidos de forma separada. Considérese el objetivo i-ésimo, para el cual la desviación relativa puede calcularse en la forma

$$z_i'(\boldsymbol{x}) = \frac{|f_i(\boldsymbol{x}) - f_i^o|}{|f_i^o|} \tag{2.3}$$

o también

$$z_i''(\boldsymbol{x}) = \frac{|f_i(\boldsymbol{x}) - f_i^o|}{|f_i(\boldsymbol{x})|} \tag{2.4}$$

donde en (2.3) y (2.4) se ha asumido que para todo $i \in I$ y para todo $\boldsymbol{x} \in S$, los denominadores son distintos de cero. Siendo I el conjunto de los índices de los objetivos y S el conjunto de las soluciones factibles.

Sea $\boldsymbol{z}(\boldsymbol{x}) = [z_1(\boldsymbol{x}), \cdots, z_i(\boldsymbol{x}), \cdots, z_k(\boldsymbol{x})]^T$ un vector cuyos elementos son los incrementos relativos definidos en $\mathbb{R}^k$, obtenidos a partir de

$$\forall i \in I \qquad z_i(\boldsymbol{x}) = \text{máx}\left\{z_i'(\boldsymbol{x}), z_i''(\boldsymbol{x})\right\}$$

Un punto $\boldsymbol{x^*} \in S$ es óptimo en sentido min-max, si para todo $\boldsymbol{x} \in S$ se satisface la siguiente fórmula de recurrencia:

1. $$v_1(\boldsymbol{x^*}) = \min_{x \in S} \max_{i \in I} \{z_i(\boldsymbol{x})\}$$

y entonces $I_1 = \{i_1\}$, donde i_1 es el índice para el cual el valor de $z_i(\boldsymbol{x})$ es máximo. Si hay un conjunto de soluciones $x_1 \subset S$ que satisfagan el paso 1, entonces

2. $$v_2(\boldsymbol{x^*}) = \min_{x \in x_1} \max_{i \in I, i \notin I_1} \{z_i(\boldsymbol{x})\}$$

y entonces $I_2 = \{i_1, i_2\}$, donde i_2 es el índice para el cual el valor de $z_i(\boldsymbol{x})$ es máximo en este paso.

$$\vdots \qquad \vdots$$

Si hay un conjunto de soluciones $x_{k-1} \subset S$ que satisfagan el paso $k-1$, entonces

k. $$v_k(\boldsymbol{x^*}) = \min x \in x_{k-1} \max_{i \in I, i \notin I_{k-1}} \{z_i(\boldsymbol{x})\}$$

y entonces $I_k = \{I_{k-1}, i_k\}$, donde i_k es el índice para el cual el valor de $z_i(\boldsymbol{x})$ es máximo en este paso.

2.5. Método de prioridades variables para optimización multiobjetivo evolutiva.

El método que se propone a continuación, está inspirado en algunos de los métodos descritos anteriormente en este capítulo, y ha sido utilizado de forma extensiva en los trabajos de investigación recogidos en la presente tesis. Va a permitir la ordenación completa de la población, y será de aplicabilidad a conjuntos de objetivos que tengan en un principio la misma prioridad. Será durante la fase de ejecución del algoritmo cuando se irán asignando prioridades a los distintos objetivos, en función de lo apartados que se encuentren de sus metas: aquellos objetivos que se encuentren más apartados de sus metas, pasarán a tener la máxima prioridad.

El método estará incluido en un algoritmo evolutivo, y será el responsable de realizar la ordenación de la población previa al proceso de selección. La ordenación se llevará a cabo a partir de los resultados obtenidos por la función de *evaluación* o de *aptitud* para los distintos objetivos de todos los cromosomas. A continuación se describen las características propias del proceso de ordenación.

La función a optimizar será de la forma $f : S \rightarrow T$, donde $S \subset \mathbb{R}^n$ y $T \subset \mathbb{R}^k$. La imagen de f estará formada por todos los vectores $\boldsymbol{f}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_k(\boldsymbol{x})\}$ tales que $\boldsymbol{x} \in S$, cuyos elementos son los valores correspondientes a cada uno de los k objetivos. Como función de evaluación del algoritmo evolutivo se tomará otra función $\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}))$, con objeto de normalizar los k valores devueltos por f. Mediante la normalización se consigue que los distintos objetivos sean comparables entre sí.

Para realizar la normalización, habrá que establecer previamente un vector con las metas deseadas para los distintos objetivos (en los problemas de control, estarán basadas en las especificaciones de diseño). Sea este vector $\boldsymbol{o} = \{o_1, \cdots, o_k\} \in T$, para un cromosoma $\boldsymbol{x}$ los valores normalizados serán

$$g_i(\boldsymbol{f}(\boldsymbol{x})) = \frac{o_i - f_i(\boldsymbol{x})}{o_i}$$

si se pretende maximizar el valor correspondiente al objetivo i con $o_i > 0$, o minimizarlo con $o_i < 0$, o

$$g_i(\boldsymbol{f}(\boldsymbol{x})) = \frac{f_i(\boldsymbol{x}) - o_i}{o_i}$$

cuando lo que se pretende es minimizar el valor correspondiente al objetivo i, con $o_i > 0$, o maximizarlo con $o_i < 0$.

En lo que sigue se considerará un problema de minimización, en el que las metas se alcanzarán cuando $g_i(\boldsymbol{x}) = 0$, y se mejorarán cuando $g_i(\boldsymbol{x}) < 0$. La extensión a un caso de maximización es trivial.

La ordenación de la población se hace por un procedimiento similar al utilizado en el método lexicográfico, es decir, comparando los vectores de aptitud de los cromosomas, teniendo en cuenta las prioridades de los objetivos. La diferencia, como ya se ha comentado, está en que en el presente método las prioridades no son fijas, e incluso distintos cromosomas de la misma población tendrán prioridades diferentes.

Sean $\boldsymbol{u} = g(\boldsymbol{x_u})$ y $\boldsymbol{v} = g(\boldsymbol{x_v})$ los vectores de aptitud normalizados, obtenidos por la función de evaluación para dos cromosomas diferentes, $\boldsymbol{x_u}$ y $\boldsymbol{x_v}$, y sean $\boldsymbol{\gamma}(\boldsymbol{u})$ y $\boldsymbol{\gamma}(\boldsymbol{v})$ las permutaciones respectivas de sus elementos en las que éstos han sido reordenados de forma decreciente, quedando como primer elemento en cada cromosoma el correspondiente al objetivo que más diste de cumplirse. $\boldsymbol{\gamma}_i(\boldsymbol{u})$ y $\boldsymbol{\gamma}_i(\boldsymbol{v})$ denotarán sus elementos i-ésimos respectivos.

Considérense además los conjuntos disjuntos C y D, cuyos elementos serán índices de los elementos de las permutaciones $\boldsymbol{\gamma}(\boldsymbol{u})$ y $\boldsymbol{\gamma}(\boldsymbol{v})$, generados de la siguiente forma:

$$\forall i \in \{1, \cdots, k\} \begin{cases} i \in C \Leftrightarrow \boldsymbol{\gamma}_i(\boldsymbol{u}) < \boldsymbol{\gamma}_i(\boldsymbol{v}) \\ i \in D \Leftrightarrow \boldsymbol{\gamma}_i(\boldsymbol{u}) > \boldsymbol{\gamma}_i(\boldsymbol{v}) \end{cases}$$

Definición 2.3 (preferibilidad) *El vector $\boldsymbol{u}$ es preferible al vector $\boldsymbol{v}$ ($\boldsymbol{u} \succ \boldsymbol{v}$), si y sólo si $C \neq \phi$ y además $\nexists\, j \in D \mid j < \text{mín}\,(C)$. Análogamente, $\boldsymbol{v}$ es preferible a $\boldsymbol{u}$ ($\boldsymbol{v} \succ \boldsymbol{u}$), si y sólo si $D \neq \phi$ y además $\nexists\, r \in C \mid r < \text{mín}\,(D)$.*

Definición 2.4 (equivalencia) *El vector $\boldsymbol{u}$ es equivalente al vector $\boldsymbol{v}$ ($\boldsymbol{u} \equiv \boldsymbol{v}$), si y sólo si $C = D = \phi$, es decir, ninguno de los dos vectores es preferible al otro.*

Los teoremas que siguen, completan lo necesario para realizar la ordenación completa de una población de cromosomas, en función de sus vectores de aptitud.

Teorema 2.1 *Si el vector $\boldsymbol{u}$ no es preferible al vector $\boldsymbol{v}$, y ambos vectores no son equivalentes, entonces el vector $\boldsymbol{v}$ es preferible al vector $\boldsymbol{u}$.*

Demostración: Si $\boldsymbol{u}$ no es preferible a $\boldsymbol{v}$, ha de cumplirse que

$$C = \phi \qquad \text{ó} \qquad \exists\, j \in D \mid j < \text{mín}\,(C)$$

- Si $C = \phi$ entonces $D \neq \phi$, pues de lo contrario sería $\boldsymbol{u} \equiv \boldsymbol{v}$. Además $C = \phi \quad \Rightarrow \quad \nexists\, r \in C \mid r < \text{mín}\,(D)$. Luego $\boldsymbol{v} \succ \boldsymbol{u}$.
- Si $\exists\, j \in D \mid j < \text{mín}\,(C)$ entonces $D \neq \phi$ y además $\nexists\, r \in C \mid r < \text{mín}\,(D)$, concluyéndose que $\boldsymbol{v} \succ \boldsymbol{u}$.

Teorema 2.2 *$\boldsymbol{u}$ es preferible a $\boldsymbol{v}$, si y sólo si $\exists i \in \{1, \cdots, k\} \mid \boldsymbol{\gamma}_i(\boldsymbol{u}) < \boldsymbol{\gamma}_i(\boldsymbol{v})$ y además $\boldsymbol{\gamma}_j(\boldsymbol{u}) = \boldsymbol{\gamma}_j(\boldsymbol{v}) \quad \forall j \in \{1, \cdots, k\}$ tal que $j < i$.*

Demostración: $C \neq \phi \quad \Rightarrow \quad \exists i \in \{1, \cdots, k\} \mid \boldsymbol{\gamma}_i(\boldsymbol{u}) < \boldsymbol{\gamma}_i(\boldsymbol{v})$ y además, suponiendo que $i = \text{mín}\,(C)$, lo que no hace perder generalidad, $\nexists\, j \in D \mid j < \text{mín}\,(C) \quad \Rightarrow \quad \boldsymbol{\gamma}_j(\boldsymbol{u}) = \boldsymbol{\gamma}_j(\boldsymbol{v}) \quad \forall j \in \{1, \cdots, k\}$ tal que $j < i$.

En el otro sentido, $\exists i \in \{1, \cdots, k\} \mid \boldsymbol{\gamma}_i(\boldsymbol{u}) < \boldsymbol{\gamma}_i(\boldsymbol{v}) \;\Rightarrow\; C \neq \phi$, y además $\boldsymbol{\gamma}_j(\boldsymbol{u}) = \boldsymbol{\gamma}_j(\boldsymbol{v}) \quad \forall j \in \{1, \cdots, k\}$ tal que $j < i \;\Rightarrow\; \nexists\, j \in D \mid j < \text{mín}\,(C)$

Teorema 2.3 (propiedad transitiva) *Si $\boldsymbol{u} \succ \boldsymbol{v}$ y $\boldsymbol{v} \succ \boldsymbol{w}$, entonces $\boldsymbol{u} \succ \boldsymbol{w}$.*

Demostración: A partir del teorema 2.2 se obtiene:

$\boldsymbol{v} \succ \boldsymbol{w} \;\Rightarrow\; \exists i \in \{1, \cdots, k\} \mid \boldsymbol{\gamma}_i(\boldsymbol{v}) < \boldsymbol{\gamma}_i(\boldsymbol{w})$ y además $\boldsymbol{\gamma}_j(\boldsymbol{v}) = \boldsymbol{\gamma}_j(\boldsymbol{w})$
$\forall j \in \{1, \cdots, k\}$ tal que $j < i$.
$\boldsymbol{u} \succ \boldsymbol{v} \;\Rightarrow\; \exists r \in \{1, \cdots, k\} \mid \boldsymbol{\gamma}_r(\boldsymbol{u}) < \boldsymbol{\gamma}_r(\boldsymbol{v})$ y además $\boldsymbol{\gamma}_s(\boldsymbol{u}) = \boldsymbol{\gamma}_s(\boldsymbol{v})$
$\forall s \in \{1, \cdots, k\}$ tal que $s < r$.

Hay dos posibilidades:

- $r < i \;\Rightarrow\; \boldsymbol{\gamma}_r(\boldsymbol{u}) < \boldsymbol{\gamma}_r(\boldsymbol{w})$ y $\boldsymbol{\gamma}_m(\boldsymbol{u}) = \boldsymbol{\gamma}_m(\boldsymbol{w})$
  $\forall m \in \{1, \cdots, k\}$ tal que $m < r$.
- $r \geq i \;\Rightarrow\; \boldsymbol{\gamma}_i(\boldsymbol{u}) < \boldsymbol{\gamma}_i(\boldsymbol{w})$ y $\boldsymbol{\gamma}_m(\boldsymbol{u}) = \boldsymbol{\gamma}_m(\boldsymbol{w})$
  $\forall m \in \{1, \cdots, k\}$ tal que $m < i$.

Y ambos casos implican, por el teorema 2.2, que $\boldsymbol{u} \succ \boldsymbol{w}$.

2.6. Conclusiones.

Muchos problemas en ciencias e ingeniería requieren la optimización simultánea de múltiples objetivos. Sin embargo, al intentar realizar dicha optimización aparecen conflictos, debido a que la mejora en alguno de los objetivos trae consigo un empeoramiento de algún otro. Se deberá, por tanto, llegar a una situación de compromiso en la que todos los objetivos cumplan unos mínimos razonables desde el punto de vista del diseñador.

Los algoritmos evolutivos, por su parte, se han mostrado como un método adecuado para resolver problemas de optimización multiobjetivo, gracias a su capacidad para trabajar de forma simultánea con múltiples posibles soluciones al problema.

En este capítulo se han presentado diversos métodos, empleados en la resolución de problemas de optimización multiobjetivo mediante algoritmos evolutivos. Dichos métodos han sido agrupados en:

- Métodos basados en el concepto de dominancia de Pareto.
- Métodos basados en la combinación de objetivos.
- Otros métodos no basados en la dominancia de Pareto.

Finalmente se ha propuesto un nuevo método que será utilizado ampliamente en los capítulos que se verán a continuación. El método permite la ordenación completa de los cromosomas de la población en función de sus vectores de aptitud, y trabajar con un conjunto de objetivos que tengan la misma prioridad.

Capítulo 3

Del problema de control a la construcción del algoritmo

3.1. Introducción.

En el presente capítulo se estudiará cómo construir un algoritmo evolutivo para resolver un problema de control, a partir del modelo de la planta y de las especificaciones de diseño que indican el comportamiento esperado tras incluir en el sistema el controlador obtenido.

Durante el desarrollo de esta tesis se trabajó de forma extensiva en dos problemas:

- El primero de ellos, conocido como RCAM (iniciales de su denominación en inglés: *Research Civil Aircraft Model*), corresponde a un avión y fue preparado por GARTEUR Action Group 08 (*Group for Aeronautical Research and Technology in EURope*) [55]. Además del modelo del avión, también se proporciona un entorno de trabajo con el que poder evaluar y comparar de una forma homogénea los controladores obtenidos por distintos métodos y distintos investigadores. Los controladores son evaluados durante una maniobra automática de aproximación y aterrizaje. El objetivo es comparar la teoría de control robusto moderna con los métodos de diseño clásicos, en su aplicación a problemas reales, y aumentar la confianza de la industria aeronáutica europea en las técnicas de control robusto.

- El segundo modelo utilizado corresponde a un barco y se generó dentro del proyecto CRIBAV (*Control Robusto e Inteligente de Buques de Alta Velocidad*). El

objetivo en este caso es reducir las aceleraciones a que está sometido el barco debido al oleaje, haciendo decrecer de esta forma el índice de mareo entre los pasajeros. El problema fue tratado también por Aranda *et al* [9], De la Cruz *et al* [38], y Díaz *et al* [40].

En los siguientes apartados se expondrán con más detalle ambos modelos.

Una vez estudiado el problema propuesto, se diseña un algoritmo que se encargue de la obtención del controlador. Habrá que identificar cuales son las variables de interés para el algoritmo, cuál es la mejor forma de codificar estos valores, de qué forma se tienen en cuenta las especificaciones de diseño, y algún detalle más que se verá en los apartados siguientes. El planteamiento será general, dejando los detalles específicos para los capítulos siguientes, en los que el algoritmo se aplica a casos concretos.

3.2. Definición del problema de control de vuelo de un avión comercial.

Desde hace tiempo, los investigadores de universidades e institutos de investigación han venido desarrollando nuevos métodos matemáticos avanzados para el diseño y análisis de controladores. Estos métodos presentan un gran potencial para la mejora de los procesos de desarrollo de leyes de control de vuelo. Con objeto de investigar los beneficios y desventajas que estos nuevos métodos podían presentar, en la pasada década GARTEUR (*Group for Aeronautical Research and Technology in EURope*) propuso un reto, invitando a investigadores de distintos países a aplicar métodos de diseño de control robusto a un modelo de avión, el RCAM (*Research Civil Aircraft Model*), que está basado en un avión de transporte civil provisto de dos motores turbofán y configurado para el aterrizaje.

El RCAM es un modelo matemático no-lineal de un avión con 6 grados de libertad. Incluye modelos de la aerodinámica, los motores, la atmósfera y la gravedad. Además son tenidas en cuenta las características de los actuadores y sensores, junto con modelos para el viento y las turbulencias atmosféricas. En la figura 3.1 se muestra un esquema del modelo global, en el que se pueden ver los bloques correspondientes a la distintas partes modeladas.

A partir del modelo no-lineal se obtendrán modelos linealizados para distintas condiciones de funcionamiento, siendo estos últimos con los que se trabajará en la presente

tesis para la obtención de los controladores. En [55] se puede encontrar una exposición más detallada del modelo no-lineal.

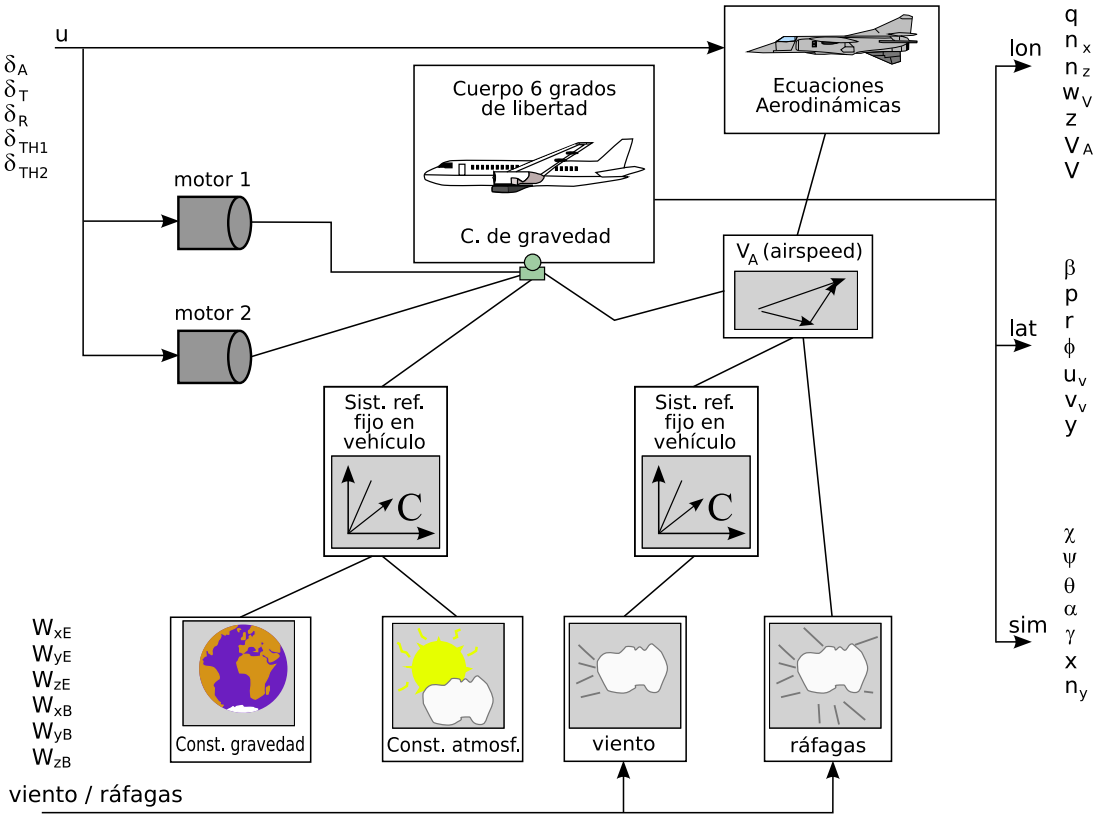


Figura 3.1: Esquema del modelo RCAM.

La figura 3.2 muestra un diagrama de bloques del modelo.

En la documentación del modelo, se incluye el conjunto de especificaciones de diseño. Además *GARTEUR* proporcionó el *software* necesario para evaluar los distintos controladores obtenidos. El objeto de proporcionar este *software*, es el de posibilitar que los controladores obtenidos por distintos métodos y diferentes investigadores, pudieran ser evaluados de una forma homogénea y los resultados obtenidos por unos y otros fueran comparables entre sí.

A continuación se detallan las distintas variables del modelo, especificadas en las figuras 3.1 y 3.2.

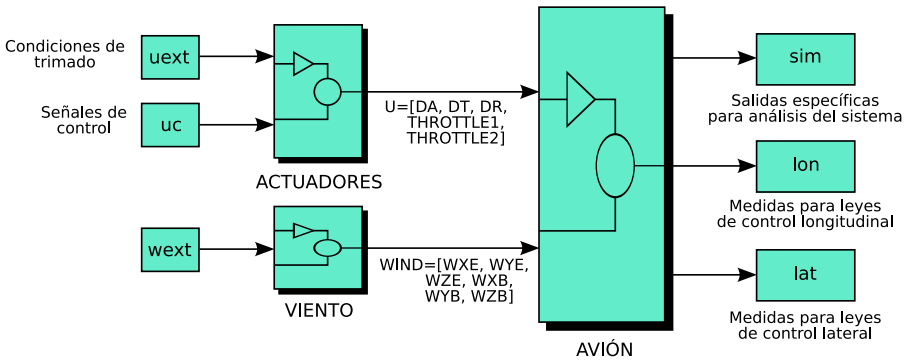


Figura 3.2: Diagrama de bloques del modelo RCAM.

3.2.1. Variables del modelo.

En las tablas que siguen a continuación, se recogen las variables utilizadas por el modelo. La tabla 3.1 muestra las entradas al modelo, expresadas en el SI de unidades.

Símbolo	Alfanumérico			Descripción	Unidades
Entradas de control					
δ_A	DA	u(1)	=	deflexión alerón	rad
δ_T	DT	u(2)	=	deflexión timón de profundidad	rad
δ_R	DR	u(3)	=	deflexión timón de dirección	rad
δ_{TH1}	THROTTLE1	u(4)	=	posición palanca motor 1	rad
δ_{TH2}	THROTTLE2	u(5)	=	posición palanca motor 2	rad
Entradas de viento					
W_{XE}	WXE	u(6)	=	velocidad viento eje x de F_E	m/s
W_{YE}	WYE	u(7)	=	velocidad viento eje y de F_E	m/s
W_{ZE}	WZE	u(8)	=	velocidad viento eje z de F_E	m/s
W_{XB}	WXB	u(9)	=	velocidad viento eje x de F_B	m/s
W_{YB}	WYB	u(10)	=	velocidad viento eje y de F_B	m/s
W_{ZB}	WZB	u(11)	=	velocidad viento eje z de F_B	m/s

Tabla 3.1: Definiciones de las entradas al modelo.

En esta tabla, F_E representa el sistema de referencia fijo en la Tierra y definido de la siguiente forma:

> El origen O_E está localizado en el comienzo del eje longitudinal de la pista. x_E será positivo hacia al norte, el cual será a su vez el rumbo de aterrizaje de la pista. z_E será positivo hacia abajo, e y_E será positivo hacia el este.

F_B por su parte representa el sistema de referencia fijo en el cuerpo, que está definido de la siguiente forma:

> El origen O_B está localizado en el centro de gravedad del vehículo. x_B será positivo hacia delante, z_B será positivo hacia abajo, e y_B será positivo hacia la derecha.

Las tres entradas de viento en el sistema F_E, u(6)-u(8), están pensadas para utilizarse con componentes del viento de velocidad constante, mientras que las entradas del viento en el sistema F_B lo están para utilizarse con ráfagas.

Los estados utilizados internamente por el *software* están expresados en el SI de unidades y se recogen en la tabla 3.2.

Símbolo	Alfanumérico			Descripción	Unidades
p	P	x(1)	=	velocidad de alabeo (*roll*) (en F_B)	rad/s
q	Q	x(2)	=	velocidad de cabeceo (*pitch*) (en F_B)	rad/s
r	R	x(3)	=	velocidad de guiñada (*yaw*) (en F_B)	rad/s
ϕ	PHI	x(4)	=	ángulo de alabeo (*roll*)	rad
θ	THETA	x(5)	=	ángulo de cabeceo (*pitch*)	rad
ψ	PSI	x(6)	=	ángulo de guiñada (*yaw*)	rad
u_B	UB	x(7)	=	componente x velocidad inercial en F_B	m/s
v_B	VB	x(8)	=	componente y velocidad inercial en F_B	m/s
w_B	WB	x(9)	=	componente z velocidad inercial en F_B	m/s
x	X	x(10)	=	posición x del avión en F_E	m
y	Y	x(11)	=	posición y del avión en F_E	m
z	Z	x(12)	=	posición z del avión en F_E	m

Tabla 3.2: Definiciones de los estados del modelo.

Las salidas del modelo están dadas en el SI de unidades y se recogen en la tabla 3.3. En esta tabla, F_V denota el sistema de referencia vertical fijo en el vehículo, que se define de la siguiente forma:

> El sistema de referencia vertical fijo en el vehículo es paralelo al sistema de referencia fijo en la Tierra definido anteriormente como F_E, pero moviéndose con el vehículo. Su origen O_V se localiza en el centro de gravedad del vehículo. x_V es positivo apuntando hacia el norte, y_V es positivo hacia el este, y z_V es positivo hacia abajo.

Sólo las salidas del modelo etiquetadas como 'medidas' se podrán considerar dispo-

nibles como entradas al controlador que se esté diseñando. Las salidas de "simulación" sólo están disponibles a efectos de evaluación y no deberán utilizarse por el controlador.

Símbolo	Alfanumérico			Descripción	Unidades
Medidas					
q	Q	y(1)	=	velocidad de cabeceo (*pitch*) (en F_B) = x(2)	rad/s
n_x	NX	y(2)	=	Factor de carga horizontal (en F_B) = $\frac{F_x}{mg}$	-
n_z	NZ	y(3)	=	Factor de carga vertical (en F_B) = $\frac{F_z}{mg}$	-
w_V	WV	y(4)	=	componente z de velocidad inercial (en F_V)	m/s
z	Z	y(5)	=	posición z del avión en F_E = x(12)	m
V_A	VA	y(6)	=	velocidad respecto al aire (*airspeed*)	m/s
V	V	y(7)	=	velocidad inercial total	m/s
β	BETA	y(8)	=	ángulo de deslizamiento (*sideslip*)	rad
p	P	y(9)	=	velocidad de alabeo (*roll*) (en F_B) = x(1)	rad/s
r	R	y(10)	=	velocidad de guiñada (*yaw*) (en F_B) = x(3)	rad/s
ϕ	PHI	y(11)	=	ángulo de alabeo (*roll*) = x(4)	rad
u_V	UV	y(12)	=	componente x de velocidad inercial (en F_V)	m/s
v_V	VV	y(13)	=	componente y de velocidad inercial (en F_V)	m/s
y	Y	y(14)	=	posición y del avión en F_E = x(11)	m
χ	CHI	y(15)	=	ángulo de trayectoria inercial	rad
Simulación					
ψ	PSI	y(16)	=	ángulo de rumbo (*heading*) = x(6)	rad
θ	THETA	y(17)	=	ángulo de cabeceo (*pitch*) = x(5)	rad
α	ALPHA	y(18)	=	ángulo de ataque	rad
γ	GAMMA	y(19)	=	ángulo de senda de vuelo inercial	rad
x	X	y(20)	=	posición x del avión en F_E = x(10)	m
n_y	NY	y(21)	=	Factor de carga lateral (en F_B) = $\frac{F_y}{mg}$	-

Tabla 3.3: Definiciones de las salidas del modelo.

3.2.2. Especificaciones de diseño.

El objetivo perseguido es la obtención de una ley de control capaz de realizar de forma automática una maniobra de aproximación bajo diversas condiciones externas y que sea robusta a los cambios de parámetros. Además, el guiado del avión no debe degradarse ante un fallo de motor. La trayectoria que deberá seguir el avión con el controlador obtenido se muestra en la figura 3.3.

Para obtener los objetivos mencionados se establecieron una serie de criterios de diseño, distribuidos en cinco clases:

- Criterios de comportamiento.

- Criterios de robustez.
- Criterios de calidad de viaje.
- Criterios de seguridad.
- Criterios de actividad de control.

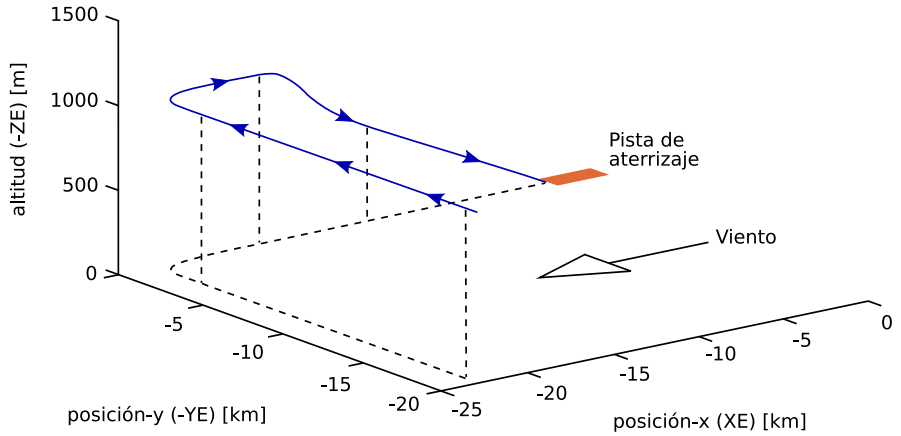


Figura 3.3: Maniobra de aproximación a realizar por el avión.

3.2.2.1. Criterios de comportamiento.

El comportamiento se especificará en términos de la respuesta del sistema frente a la introducción de señales de referencia normalizadas. Las respuestas ante señales de *tipo escalón* se definirán en términos de tiempo de subida, t_r, tiempo de asentamiento, t_s, y máximo sobreimpulso, M_p. El tiempo de subida se considerará aquí como el tiempo que tarda la salida del sistema, $y(t)$ en pasar del 10 % al 90 % del valor del escalón, es decir: $t_r = t(y_{90\,\%}) - t(y_{10\,\%})$. El tiempo de asentamiento se considerará como el tiempo que tarda $y(t)$ en mantenerse dentro del rango del 99 % de su valor final. Por último, el máximo sobreimpulso se considerará como el máximo pico relativo de $y(t)$, es decir: $M_p = \frac{(y_{pico}) - y(\infty)}{y(\infty)} \times 100\,\%$.

Los criterios que se considerarán son los siguientes:

Desviación lateral. La desviación lateral, $e_{yb}(t)$, definida como la diferencia entre la posición lateral requerida y la posición real, debe reducirse al 10 % en 30 s. El máximo sobreimpulso ante señales de tipo escalón será pequeña, ($M_p < 5\,\%$),

para altitudes superiores a 1000 ft. A menores altitudes, M_p podrá incrementarse hasta el 30 % con objeto de acortar el tiempo de respuesta. No deberá haber error en el estacionario debido al viento lateral constante.

Respuesta de altitud. El sistema debe ser capaz de seguir los mandatos de altitud, h_c, con un tiempo de subida t_r <12 s y un tiempo de asentamiento t_s <45 s. Ante señales de tipo escalón, M_p ha de ser menor del 5 % por encima de 1000 ft, pudiendo incrementarse hasta el 30 % para altitudes menores.

Respuesta de ángulo de rumbo. El ángulo de rumbo (*heading angle*) requerido, ψ_c, deberá ser seguido por el ángulo de rumbo real, ψ, con un tiempo de subida t_r <10 s y un tiempo de asentamiento t_s <30 s. Ante señales de tipo escalón, M_p deberá ser menor del 5 % por encima de 1000 ft, pudiendo incrementarse hasta el 30 % para altitudes menores.

Respuesta de ángulo de senda de vuelo. El ángulo de senda de vuelo (*flight path angle*) solicitado, γ_c, deberá ser seguido por el ángulo de senda de vuelo real, γ, con un tiempo de subida t_r <5 s y un tiempo de asentamiento t_s <20 s. Ante señales de tipo escalón, M_p deberá ser menor del 5 % por encima de 1000 ft, pudiendo incrementarse hasta el 30 % para altitudes menores.

Respuesta de ángulo de alabeo. En caso de fallo de motor, el ángulo de alabeo (*roll angle*), ϕ, no debería exceder 10 grados. Su desviación máxima en estado estacionario no deberá exceder 5 grados. Durante un fallo de motor, el ángulo de deslizamiento (*sideslip angle*), β, debería minimizarse; el ángulo de alabeo en estado estacionario necesario para conseguir esto debería reducirse a cero con un sobreimpulso menor del 50 %, cuando el motor que hubiera fallado fuera rearrancado. Bajo condiciones de turbulencia moderada, ϕ debería mantenerse menor de 5 grados.

Respuesta de velocidad respecto al aire. La velocidad respecto al aire, V_A, deberá seguir las señales de velocidad requerida, V_{Ac}, con un tiempo de subida t_r <12 s y un tiempo de asentamiento t_s <45 s. Ante señales de tipo escalón, M_p deberá ser menor del 5 % por encima de 1000 ft, pudiendo incrementarse hasta el 30 % para altitudes menores. No deberá haber errores en el estacionario debido a perturbaciones de viento constante.

Régimen de variación de rumbo. En el caso de fallo de motor, el máximo de régimen de variación de rumbo, $\dot{\psi}$, deberá ser menor de 3 $grados/seg$.

Acoplamiento entre la velocidad respecto al aire V_A y la altitud h. Para una señal de escalón en la altitud requerida, h_c, de 30 m, el valor de pico del transitorio del error absoluto entre V_A y V_{Ac} deberá ser menor de 0.5 m/s. De igual modo, para un escalón de 13 m/s en V_{Ac}, el valor de pico del transitorio del error absoluto entre h y h_c deberá ser menor de 10 m.

3.2.2.2. Criterios de robustez.

Variación del centro de gravedad. El sistema deberá mantenerse estable y con un comportamiento suficiente, frente a variaciones horizontales y verticales del centro de gravedad.

Variaciones de masa. Deberá mantenerse también la estabilidad y un comportamiento suficiente, ante variaciones de la masa del avión entre 100000 y 150000 kg.

Variaciones de velocidad. La estabilidad y el comportamiento deberán mantenerse también para variaciones de velocidad entre $1.23V_S$ y 90 m/s, siendo V_S la velocidad de entrada en pérdida (que es función del peso, de la altura de vuelo y de la configuración del avión).

3.2.2.3. Criterios de calidad de viaje.

Los criterios de calidad de viaje deberán asegurar el confort de los pasajeros y la tripulación.

Aceleración vertical máxima. Las aceleraciones verticales se deberán minimizar. Durante el vuelo rectilíneo deberán ser menores de ± 0.05 G's, y durante los giros de 30 grados menores de ± 0.2 G's.

Aceleración lateral máxima. Las aceleraciones laterales también deberán ser minimizadas. Durante el vuelo rectilíneo deberán ser menores de ± 0.02 G's, y durante los giros de 30 grados menores de ± 0.04 G's.

Amortiguamiento. Excepto en los casos en que se establezca otra cosa, no deberá haber sobreimpulso en la respuesta a ninguna señal de tipo escalón por encima de 1000 ft. Por debajo de esta altitud, el sobreimpulso podrá incrementarse hasta el 30 % con objeto de reducir los tiempos de respuesta en situaciones más críticas.

3.2.2.4. Criterios de seguridad.

Velocidad respecto al aire. La velocidad respecto al aire siempre será mayor que $1.05V_S$, donde V_S es la velocidad de pérdida, es decir, la velocidad por debajo de la cual el avión es incapaz de mantener la altura. Como ya se ha comentado, esta velocidad dependerá del peso, de la altura de vuelo y de la configuración del avión.

Ángulo de ataque. El ángulo de ataque se deberá mantener por debajo de 18 grados. Un valor de 12 grados se considerará aceptable.

Ángulo de alabeo. El máximo ángulo de alabeo estará limitado a 30 grados.

Ángulo de deslizamiento. En todo momento, el ángulo de deslizamiento, β, deberá minimizarse.

3.2.2.5. Criterios de actividad de control.

Esfuerzo de los actuadores. En condiciones de turbulencias moderadas, las velocidades medias de actuación de los alerones, timón de dirección y timón de profundidad, deberán ser menores del 33 % de sus velocidades máximas.

Esfuerzo de los motores. En condiciones de turbulencias moderadas, las velocidades medias de variación de las palancas de los motores, deberán ser menores del 15 % de la velocidad máxima.

3.2.3. Saturaciones y límites en los controles.

A la hora de diseñar el controlador, se deberán tener en cuenta también los valores extremos que pueden tomar las señales de control, así como sus variaciones. Estos valores son los siguientes:

- Saturaciones para los ángulos de las palancas de los motores:

$$0.5\frac{\pi}{180}rad \leq \delta_{TH} \leq 10\frac{\pi}{180}rad$$

- Velocidades angulares límites en el movimiento de las palancas de los motores:

$$-16\frac{\pi}{180}rad/s \leq \dot{\delta}_{TH} \leq 16\frac{\pi}{180}rad/s$$

- Saturaciones para el ángulo de deflexión de los alerones:

$$-25\frac{\pi}{180}rad \leq \delta_A \leq 25\frac{\pi}{180}rad$$

- Velocidades angulares límites en el movimiento de los alerones:

$$-25\frac{\pi}{180}rad/s \leq \dot{\delta}_A \leq 25\frac{\pi}{180}rad/s$$

- Saturaciones para el ángulo de deflexión del timón de profundidad:

$$-25\frac{\pi}{180}rad \leq \delta_T \leq 10\frac{\pi}{180}rad$$

- Velocidades angulares límites en el movimiento del timón de profundidad:

$$-15\frac{\pi}{180}rad/s \leq \dot{\delta}_T \leq 15\frac{\pi}{180}rad/s$$

- Saturaciones para el ángulo de deflexión del timón de dirección:

$$-30\frac{\pi}{180}rad \leq \delta_R \leq 30\frac{\pi}{180}rad$$

- Velocidades angulares límites en el movimiento del timón de dirección:

$$-25\frac{\pi}{180}rad/s \leq \dot{\delta}_R \leq 25\frac{\pi}{180}rad/s$$

3.3. Definición del problema de control robusto de un buque de alta velocidad.

El proyecto CRIBAV (*Control Robusto e Inteligente de Buques de Alta Velocidad*) tenía como objetivo investigar las posibilidades y ventajas del uso de las técnicas de control robusto y de control inteligente en el diseño de sistemas de control para buques de alta velocidad.

En una fase previa al diseño de controladores, dentro del proyecto CRIBAV se llevó a cabo un proceso de identificación ([8], [7] y [4]), por el que se obtuvieron modelos lineales de la dinámica vertical del buque de alta velocidad TF-120. El punto de partida fueron los datos que se obtuvieron en unos ensayos realizados con oleaje regular e irregular aplicado a una maqueta a escala sin actuadores del TF-120, en el Canal de Experiencias Hidrodinámicas del Pardo (CEHIPAR).

El barco dispone de dos actuadores para controlar los movimientos de '*pitch*' y '*heave*'. Uno de ellos es un alerón situado en la popa del barco (FLAP) y el otro lo constituyen unas aletas situadas en la proa (T-FOIL). Para que las aletas de proa sean útiles, es necesario que se encuentren sumergidas y a ser posible a la suficiente profundidad como para que no haya cavitación. Esto lo consigue el alerón de popa, que hace bajar la proa mediante la elevación de la popa. La figura 3.4 muestra un detalle ampliado de las superficies de control del barco.

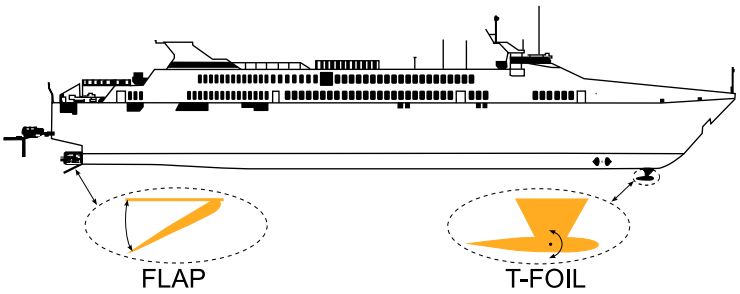


Figura 3.4: Detalle con las superficies de control ampliadas.

Durante el diseño del controlador llevado a cabo en la presente tesis, se utilizó el modelo global del proceso compuesto por el modelo de la dinámica vertical del buque TF-120 y por el modelo de los actuadores (el mismo que fue empleado por Aranda *et al* en [9]). El modelo tiene dos variables de entrada manipulables (referencia del Flap y referencia del T-foil), una variable de perturbación (altura de las olas) y dos variables de salida controladas (*pitch* y *heave*). La figura 3.5 muestra un diagrama de bloques del modelo del proceso.

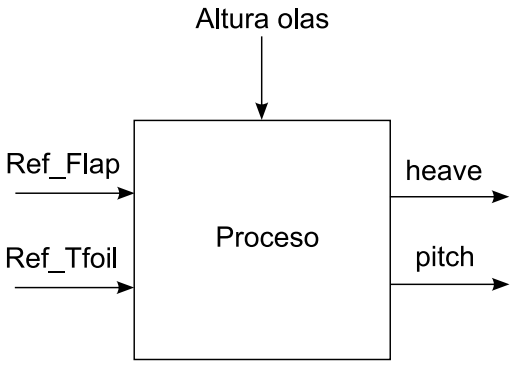


Figura 3.5: Diagrama de bloques del modelo.

El modelo lineal detallado se muestra en la figura 3.6, donde puede verse la relación

entre los distintos pares entrada-salida. También puede verse la nomenclatura empleada en las funciones de transferencia.

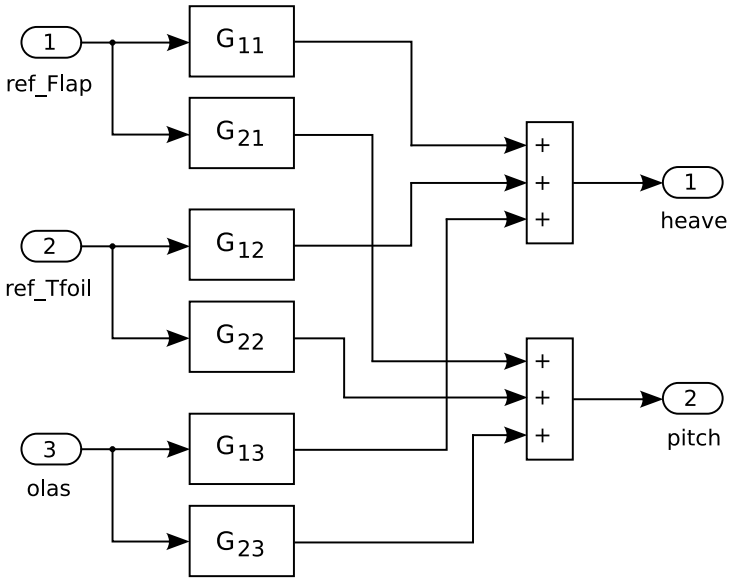


Figura 3.6: Diagrama detallado del modelo lineal.

La relación entre las variables de entrada-salida del proceso serán, por tanto:

$$heave(s) = G_{11}(s) \cdot ref\_Flap(s) + G_{12}(s) \cdot ref\_Tfoil(s) + G_{13}(s) \cdot olas(s) \quad (3.1)$$

$$pitch(s) = G_{21}(s) \cdot ref\_Flap(s) + G_{22}(s) \cdot ref\_Tfoil(s) + G_{23}(s) \cdot olas(s) \quad (3.2)$$

Las expresiones correspondientes a las funciones de transferencia $G_{11}(s)$, $G_{21}(s)$, $G_{12}(s)$, $G_{22}(s)$, $G_{13}(s)$ y $G_{23}(s)$, se encuentran en el apéndice B para distintos puntos de linealización (velocidades del barco de 20, 30 y 40 nudos).

3.3.1. Especificaciones de diseño.

El objetivo principal en este caso es reducir el índice de mareo, para lo cual habrá que disminuir las aceleraciones verticales del buque. Este objetivo deberá conseguirse manteniendo las señales de control dentro de unos límites conocidos.

3.3.1.1. MSI.

El índice de mareo, MSI (Motion Sickness Incidence), o porcentaje de personas que vomitan dentro de las dos horas, viene dado por la expresión [79]:

$$MSI = 100 \cdot \left[0.5 \pm \operatorname{erf} \left(\frac{\pm \log_{10} \left(\left| \ddot{s}_3 \right| / g \right) \mp \mu_{MSI}}{0.4} \right) \right] \tag{3.3}$$

donde $|\ddot{s}_3|$ es la aceleración vertical en el punto elegido promediada sobre la mitad de un ciclo, y donde

$$\mu_{MSI} = -0.819 + 2.32 \left(\log_{10} \omega_e \right)^2 \tag{3.4}$$

siendo ω_e la frecuencia de encuentro.

La función erf se define como

$$\operatorname{erf}(z) = \frac{1}{\sqrt{2\pi}} \int_0^x \exp(-\frac{z^2}{2}) dz \tag{3.5}$$

Se comprueba en las expresiones anteriores que el índice de mareo es función de la aceleración vertical y de la frecuencia de encuentro. En la representación gráfica de la figura 3.7 se observa que para todas las aceleraciones se da un máximo en las proximidades de la frecuencia de encuentro de 1 rad/s.

3.3.1.2. Aceleración vertical.

Al ser la frecuencia de encuentro una característica del oleaje, los esfuerzos se dirigirán a reducir la aceleración vertical debida a los movimientos de *heave* y *pitch*. Se considerará la aceleración vertical medida a 40 metros del centro de gravedad del barco. Para un instante t_i, dicha aceleración vendrá dada por la expresión:

$$acv40(t_i) = a_{VH}(t_i) + a_{VP}(t_i) = \frac{d^2 heave(t_i)}{dt^2} - 40 \cdot \frac{\pi}{180} \cdot \frac{d^2 pitch(t_i)}{dt^2} \tag{3.6}$$

También será de interés durante el diseño del controlador, la aceleración media durante N instantes de tiempo, $\overline{acv40}$, que viene dada por la expresión:

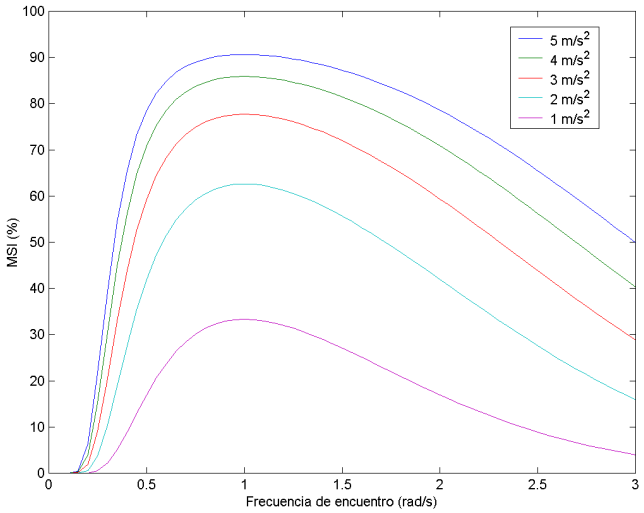


Figura 3.7: Índice de mareo (MSI) para diferentes aceleraciones y un rango de frecuencias de encuentro.

$$J = \overline{acv40} = \frac{1}{N}\sum_{i=1}^{N} |acv40\,(t_i)| \tag{3.7}$$

3.3.1.3. Saturaciones de las señales de entrada a los actuadores.

Las superficies de control tienen unas limitaciones en forma de ángulos máximos y mínimos que pueden adoptar, por tanto, las señales de entrada a los actuadores presentan saturaciones cuyos valores se indican a continuación:

- Valor mínimo de Flap: 0º.
- Valor máximo de Flap: 15º.
- Valor mínimo de T-Foil: -15º.
- Valor máximo de T-Foil: 15º.

Con el fin de evitar excursiones de la señal de control hacia la zona negativa de los valores del ángulo de flap, se establecerá un valor de trimado o posición inicial del flap en el punto medio de su rango de valores, es decir, el flap presentará una deflexión inicial de 7.5º.

3.4. Diseño del algoritmo

Una vez visto el planteamiento de los problemas, se verá a continuación la metodología empleada para su resolución. Se verán en este apartado aquellos aspectos que tengan cierta generalidad, dejando las consideraciones particulares de cada diseño para ser vistas más adelante, en sus capítulos correspondientes.

A la hora de diseñar un controlador, básicamente se deberán llevar a cabo dos pasos:

1. Elegir la estructura del controlador. Para ello, una opción será acudir a la bibliografía y emplear estructuras ya utilizadas con éxito en problemas similares. Otra opción será seguir algún procedimiento de toma de decisiones con el que se realice la selección de la estructura. En la presente tesis se hará uso de los dos procedimientos, como se verá más adelante.

2. Determinar los valores de los parámetros para la estructura seleccionada en el paso anterior. Una vez que se decida qué tipo de controlador se va a emplear, el trabajo se reducirá a elegir los valores de los coeficientes de una o varias funciones de transferencia, o los valores de los elementos de unas matrices si se trabaja en el espacio de estados.

El método empleado, está basado en el uso de los algoritmos evolutivos. Dichos algoritmos, como ya se ha visto en el capítulo 1, tienen una importante componente aleatoria, por lo que en primer lugar se prestará atención a la forma en que son generados los números aleatorios requeridos por el algoritmo. Durante la fase de diseño se comprobó que, en un principio, no se obtenían los resultados deseados. La situación cambió sin embargo al variar la forma en que se obtenían dichos números, pasando de utilizar una densidad de probabilidad uniforme a usar una función de densidad más elaborada.

3.4.1. Función de densidad de probabilidad.

Cuando se utilizan los algoritmos evolutivos para solucionar algún problema de optimización de parámetros, como puede ser la determinación de los coeficientes de la función de transferencia de un controlador, o los elementos de sus matrices en el espacio de estados, la elección de la función de densidad de probabilidad utilizada por el algoritmo puede ser un tema importante. Durante el desarrollo de esta tesis, se sustituyó el uso de la función de densidad de probabilidad uniforme por una función

de densidad de probabilidad particular. No se ha encontrado ninguna referencia en la bibliografía en la que se documente el uso de una función similar.

3.4.1.1. Densidad de probabilidad uniforme

La función de densidad uniforme es ampliamente utilizada en la generación de números aleatorios y se puede decir que es la función de densidad por defecto en gran cantidad de programas matemáticos. Por ejemplo en Matlab es la que utiliza su función rand(). Su representación gráfica, para un caso particular, se muestra en la figura 3.8.

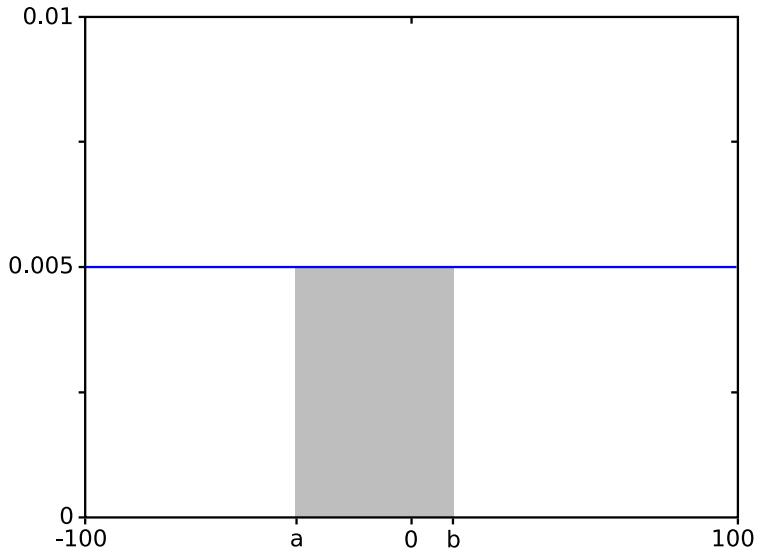


Figura 3.8: Función de densidad de probabilidad uniforme.

Supóngase que se están obteniendo números reales aleatorios con distribución uniforme, comprendidos entre -100 y 100. La probabilidad que tendría un número de encontrarse en el intervalo $[a, b]$, vendría dada por el área sombreada de la figura 3.8, es decir:

$$p[a, b] = \frac{b - a}{200}$$

3.4.1.2. Utilización en problemas de control

A la hora de determinar los coeficientes de una función de transferencia, o los elementos de una matriz en el espacio de estados por medio de un algoritmo evolutivo,

dichos parámetros podrán ser de distintos órdenes de magnitud. Como ejemplo, supóngase la siguiente función de transferencia de un determinado controlador, que satisface las especificaciones de diseño:

$$\frac{s + 11.2}{s + 0.12}$$

Se observa que el término independiente del numerador tiene orden de magnitud 1, mientras que el del denominador tiene orden de magnitud -1.

Sea Γ el conjunto formado por todos los controladores de la forma:

$$\frac{s + a}{s + b}$$

donde a es un número de orden de magnitud 1, y b un número de orden de magnitud -1. Se establecerá la hipótesis de que todos los controladores con un comportamiento aceptable, para una determinada planta, pertenecen a Γ.

Si la hipótesis anterior es cierta, resulta que la función de densidad de probabilidad uniforme que suele utilizarse por defecto como generador de números aleatorios, no es la más adecuada, ya que para distintos órdenes de magnitud la obtención de números aleatorios no es equiprobable.

Suponiendo que se establezca el intervalo $[-100, 100]$ como espacio de búsqueda para los parámetros a y b, la probabilidad de que un número obtenido aleatoriamente (con una distribución de probabilidad uniforme) tenga orden de magnitud 1 será:

$$p(ord1) = \frac{2 \cdot (100 - 10)}{200} = 0.9$$

Mientras que la probabilidad de que tenga orden de magnitud -1 será:

$$p(ord2) = \frac{2 \cdot (1 - 0.1)}{200} = 0.09$$

Si se utiliza la función de densidad uniforme, el algoritmo evolutivo perderá mucho tiempo buscando en un subespacio en el que no se encuentra la solución al problema.

Se propone por tanto el uso de una nueva función de densidad de probabilidad, en la que todos los órdenes de magnitud a considerar tengan la misma probabilidad.

3.4.1.3. Función de densidad de probabilidad variable

En la función de densidad de probabilidad que se propone, al disminuir en una unidad el orden de magnitud, se multiplicará por 10 el valor de la función. De esta forma, todos los órdenes de magnitud tendrán la misma probabilidad a la hora de obtenerse un número de forma aleatoria. La función se representa en la figura 3.9, con escala logarítmica para ambos ejes, con objeto de obtener una presentación más conveniente.

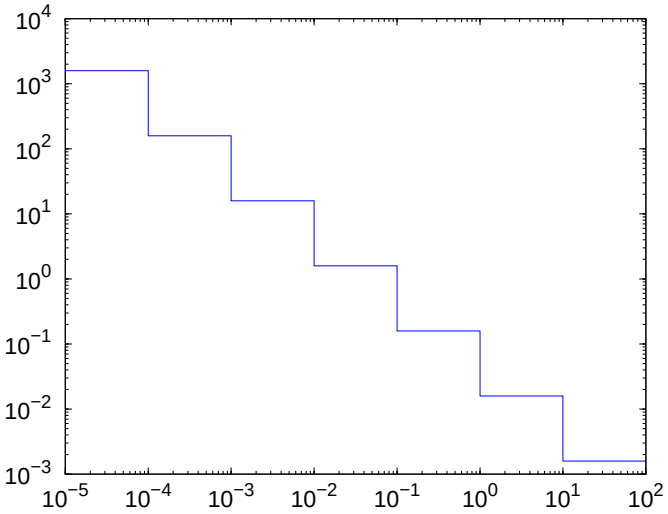


Figura 3.9: Función de densidad de probabilidad variable (escala logarítmica).

Esta función de densidad de probabilidad fue utilizada en los trabajos [8], [7], [103], obteniéndose buenos resultados en todos ellos.

En el apéndice C se recogen los resultados de un estudio comparativo sobre el uso de la función de densidad de probabilidad variable definida en este apartado, frente a la utilización de la función de densidad de probabilidad uniforme, en la obtención de los números aleatorios requeridos por un algoritmo evolutivo.

3.4.2. Codificación del problema: los cromosomas.

Los cromosomas representan soluciones al problema, codificadas de una forma adecuada para el algoritmo. En el capítulo 1 se vio que había dos formas principales de codificación: expresar las soluciones en forma de cadenas de bits, o utilizar números reales para los distintos elementos que compondrían una solución.

En la presente tesis se trabajó con poblaciones de 100 cromosomas, por ser éste un número muy utilizado en la bibliografía, según lo visto en la sección 1.3.8. Los resultados obtenidos con este valor fueron satisfactorios, por lo que no hubo nada que indujera a cambiarlo.

Respecto al tipo de codificación de los cromosomas, se emplearon los dos tipos. Cuando se diseñó un algoritmo que se encargara de seleccionar una estructura de control, se utilizó la codificación binaria, pues con un 1 se representaba que el controlador contenía uno de una serie de subbloques básicos considerados y un 0 representaba que no lo contenía. Esto se verá con más detalle en el capítulo 6. En el resto de los casos se empleó una codificación en forma de vectores de números reales, por ser ésta una codificación más cercana al problema a resolver. También se verá con más detalle cada codificación en sus capítulos correspondientes.

3.4.3. La función de evaluación o función objetivo.

En los algoritmos evolutivos, como imitación de los procesos de evolución natural, los cromosomas deberán competir entre sí, siendo el fin último la supervivencia del más apto. Se necesitará por tanto una función, que incluida en el algoritmo, permita determinar la aptitud de los distintos cromosomas.

En los problemas de control, los cromosomas, como se ha visto en el apartado 3.4.2 serán representaciones codificadas de los controladores. Dichos controladores, una vez incluidos en el sistema, darán lugar a un determinado comportamiento del sistema global. Evidentemente, distintos controladores darán lugar a comportamientos diferentes, que serán traducidos en las aptitudes requeridas por el algoritmo.

Al tratar con dos problemas diferentes (modelo RCAM y modelo CRIBAV vistos en los apartados 3.2 y 3.3), se deberán construir dos funciones de evaluación. Dichas funciones de evaluación, una vez diseñadas, podrán aplicarse a cualquier algoritmo o estrategia de control que se desee probar. Como ejemplo, en las distintas formas de abordar el problema del modelo RCAM que se verán en los capítulos que siguen, se utilizó la misma función de evaluación, con la única salvedad de los modelos empleados en la simulación, que evidentemente dependerán de la estructura de control empleada.

Para cada cromosoma, la función de evaluación generará un vector de aptitud, correspondiendo cada elemento de dicho vector a uno de los objetivos considerados. Para cada elemento del vector de aptitud, se tendrá un mejor comportamiento cuanto mayor sea su valor, considerándose cumplido el objetivo cuando se alcance el valor de

cero.

3.4.3.1. Función de evaluación para el problema de control de vuelo de un avión comercial.

Para la obtención del controlador del modelo RCAM, se separaron los modos longitudinal y lateral, por estar éstos poco interconectados.

El diseño de la función de evaluación, se realizó a partir de las especificaciones de diseño vistas anteriormente (apartado 3.2.2). A partir de dichas especificaciones se seleccionó una serie de criterios suficientemente amplia.

Para el modo longitudinal, los criterios que se utilizaron fueron los siguientes:

1. Valor máximo de las partes reales de los autovalores del sistema menor que -0.15. Si este criterio no se cumple, no se continuará con los siguientes.

Frente a una entrada de tipo escalón de $4.2\ m/s$ *en* w_V*:*

2. El máximo sobreimpulso de w_V deberá ser menor que $0.21\ m/s$.
3. El tiempo de subida de w_V deberá ser menor que $5\ s$.
4. El error medio de w_V al final de la simulación deberá ser menor que $0,042\ m/s$ ($1\,\%$ del valor del escalón).
5. El pico máximo en la salida V_A debido a la entrada escalón en w_V deberá ser menor que $0,5\ m/s$.
6. El error medio de V_A al final de la simulación, debido a la entrada escalón en w_V, deberá ser menor que $0,01\ m/s$.
7. El valor de δ_T deberá estar comprendido entre $-0.3215\ rad$ y $0.2894\ rad$.
8. El valor de δ_{TH} deberá estar comprendido entre $-0.08483\ rad$ y $0.09844\ rad$.
9. La variación máxima de δ_T deberá ser, en valor absoluto, menor que $0.2618\ rad/s$.
10. La variación máxima de δ_{TH} deberá ser, en valor absoluto, menor que $0.2793\ rad/s$.

Frente a una entrada de tipo escalón de $13\ m/s$ *en* V_A*:*

11. El máximo sobreimpulso de V_A deberá ser menor que $0.65\ m/s$.

12. El tiempo de subida de V_A deberá ser menor que 12 s.

13. El error medio de V_A al final de la simulación, deberá ser menor que 0.13 m/s (1 % del valor del escalón).

14. El pico máximo en la salida w_V, debido a la entrada escalón en V_A, deberá ser menor que 0.7 m/s.

15. El error medio de w_V al final de la simulación, debido a la entrada escalón en V_A, deberá ser menor que 0.01 m/s.

16. El valor de δ_T deberá estar comprendido entre -0.3215 rad y 0.2894 rad.

17. El valor de δ_{TH} deberá estar comprendido entre -0.08483 rad y 0.09844 rad.

18. La variación máxima de δ_T deberá ser, en valor absoluto, menor que 0.2618 rad/s.

19. La variación máxima de δ_{TH} deberá ser, en valor absoluto, menor que 0.2793 rad/s.

Robustez:

20. El margen de ganancia a la entrada deberá ser mayor que 10 dB.

21. El margen de fase a la entrada deberá ser mayor que 50^o.

22. El margen de ganancia a la salida deberá ser mayor que 10 dB.

23. El margen de fase a la salida deberá ser mayor que 50^o.

Para el modo lateral, a su vez, los criterios fueron los siguientes:

1. Valor máximo de las partes reales de los autovalores del sistema menor que -0.15. Si este criterio no se cumple, no se continuará con los siguientes.

Frente a una entrada de tipo escalón de 0.3491 rad *en* χ*:*

2. El máximo sobreimpulso de χ deberá ser menor que 0.01745 rad.

3. El tiempo de subida de χ deberá ser menor que 10 s.

4. El error medio de χ al final de la simulación deberá ser menor que 0.003491 rad.

5. El pico máximo en la salida β, debido a la entrada escalón en χ, deberá ser menor que 0.0052 rad.

6. El error medio de β al final de la simulación, debido a la entrada escalón en χ deberá ser menor que 0.0001 rad.
7. El valor de δ_A deberá estar comprendido entre -0.4363 rad y 0.4363 rad.
8. El valor de δ_R deberá estar comprendido entre -0.5236 rad y 0.5236 rad.
9. La variación máxima de δ_A deberá ser, en valor absoluto, menor que 0.4363 rad/seg.
10. La variación máxima de δ_R deberá ser, en valor absoluto, menor que 0.4363 rad/seg.

Robustez:

11. El margen de ganancia a la entrada deberá ser mayor que 10 dB.
12. El margen de fase a la entrada deberá ser mayor que 50^{o}.
13. El margen de ganancia a la salida deberá ser mayor que 10 dB.
14. El margen de fase a la salida deberá ser mayor que 50^{o}.

Al no ser posible obtener de forma analítica los valores correspondientes a muchos de los criterios, se realizará una simulación numérica para estimar posteriormente los distintos valores a partir de los datos obtenidos. En primer lugar hay que decodificar el cromosoma y convertirlo en el controlador correspondiente, luego se construye el sistema global formado por el modelo del avión y el del controlador, y por último se lleva a cabo la simulación para el sistema resultante.

En lo que sigue se detalla la forma en que se obtienen los distintos elementos del vector de aptitud. Se supondrá que se dispone de un vector y con los valores que toma la variable de salida que se esté considerando, durante la duración del experimento. Excepto en el problema que se tratará en el capítulo 6, se utilizó Matlab como lenguaje de programación, por lo que los procedimientos serán mostrados en dicho lenguaje.

Valor máximo de las partes reales de los autovalores. En un principio se pensó en considerar únicamente la estabilidad del sistema resultante tras incluir el controlador que se esté evaluando, pero durante el diseño del algoritmo se observó que se producían soluciones con buena aptitud para los distintos objetivos pero que sufrían grandes oscilaciones no deseadas durante los primeros instantes de la simulación. Se comprobó que estas soluciones, si bien eran estables, tenían polos con su parte real muy próxima al origen de coordenadas. Debido a esto, en lugar de comprobar la estabilidad, se fue algo más restrictivo, considerando como buenas soluciones aquellas cuyo valor

máximo de las partes reales de los polos del sistema, fuera menor que -0.15. Si no se cumplía el criterio comentado, el vector de aptitud tendría como primer elemento el máximo de los valores reales de los polos, cambiado de signo, y al resto de los elementos se les asignará un valor de $-Inf$. Por el contrario, si se cumple el criterio, se asignará un valor de 1 al primer elemento del vector de aptitud, y se pasará a evaluar el resto de criterios. El código correspondiente se encuentra en el apartado E.1 del apéndice E, donde A representa la matriz de la ecuación de estados del sistema global.

Máximo sobreimpulso. Se calcula el máximo sobreimpulso producido durante el experimento (y_o en la figura 3.10), para un salto de tipo escalón en la señal de referencia.

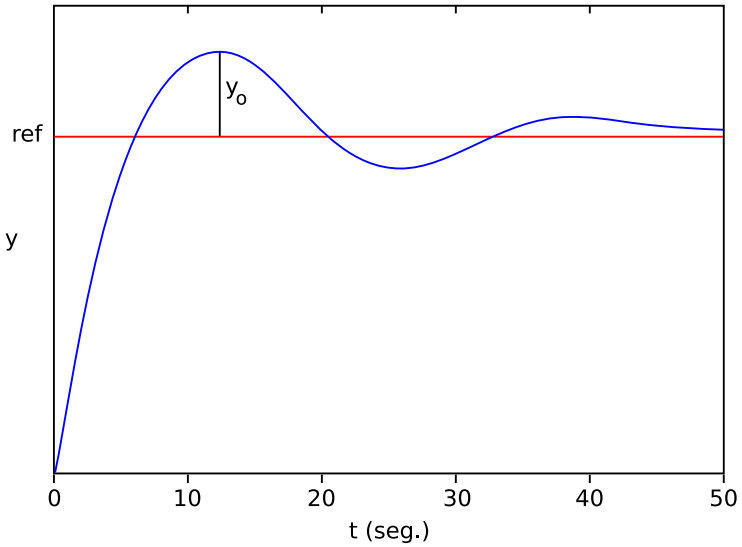


Figura 3.10: Sobreimpulso ante un escalón en la señal de referencia.

La función de Matlab empleada tiene por nombre `overshoot` y se encuentra dentro del apartado E.1 del apéndice E. Sus argumentos son `salida`, que es un vector con los datos obtenidos en el experimento y `estacionario`, que es el valor del escalón o el valor deseado en el estacionario.

Tiempo de subida. Con objeto de simplificar el algoritmo, se escogió como tiempo de subida el tiempo que tarda la salida en alcanzar el 80 % de su valor en el estacionario. En señales con un mal comportamiento, es posible que no se llegue a alcanzar dicho valor del 80 % durante el experimento; en esos casos se realizará una extrapolación por medio de la función `polyfit` de Matlab. El resultado obtenido mediante la extrapolación no es muy fiable para las señales con mal comportamiento, pues no se tiene la certeza

que la señal vaya a evolucionar en la forma supuesta, pero será suficiente a efectos de obtener una medida que pueda ser empleada en la función de evaluación. Para las señales que alcancen durante el experimento el 80 % de su valor en el estacionario, el tiempo de subida se calcula de forma precisa. La figura 3.11 ilustra el tiempo de subida considerado.

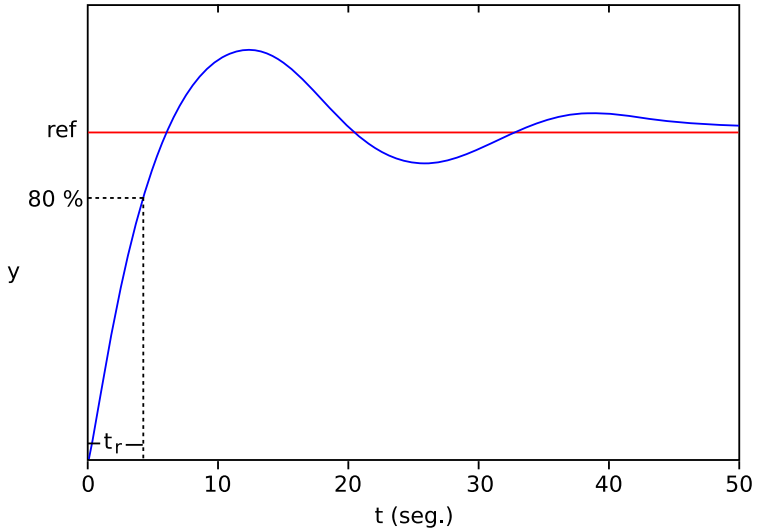


Figura 3.11: Tiempo de subida ante un escalón en la señal de referencia.

La función de Matlab utilizada tiene por nombre `trise` y se encuentra también en el apartado E.1 del apéndice E. Sus argumentos son en este caso: `salida` y `estacionario`, idénticos a los de la función `overshoot`, y `periodo`, que es el tiempo que transcurre entre dos muestras del experimento.

Error medio al final de la simulación. El error medio calculado será el del último 20 % de los puntos disponibles tras la simulación. La zona sombreada de la figura 3.12 representa la región temporal para la que se calcula el error medio.

El código de Matlab corresponde en este caso a la función de nombre `errormed`, presente también en el apartado E.1 del apéndice E. Sus argumentos son: `salida` y `estacionario`, que ya han sido comentados para las funciones anteriores.

Pico máximo de las salidas. Se deberá reducir el acoplamiento entre V_A y w_V, y también entre β y χ, de forma que un salto de tipo escalón en una de las señales implique una variación pequeña en la otra. La función que se encargará de calcular el pico máximo de una salida tiene por nombre `peakmax`, encontrándose también en el

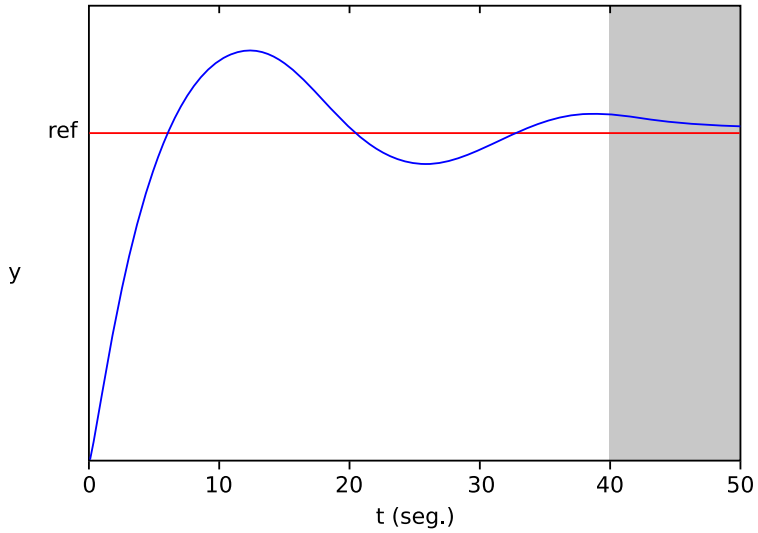


Figura 3.12: Zona en que se calcula el error medio.

apartado E.1 del apéndice E.

Valores de las señales de control comprendidos entre dos extremos. Las señales de control no deberán llegar a la saturación, por tanto, se comprobará que durante el experimento dichas señales permanecen entre los límites requeridos. En el apartado E.1 del apéndice E se muestra el código utilizado correspondiente a las saturaciones de las señales δ_T y δ_{TH} para una señal escalón en W_v, en el modo longitudinal. Para un escalón en V_A, y para el modo lateral se utilizará un código similar.

Variación máxima en las señales de control. La velocidad con la que varían las señales de control, también deberá mantenerse dentro de sus límites. En el apartado E.1 del apéndice E se recoge el código utilizado correspondiente a las variaciones de las señales δ_T y δ_{TH} para una señal escalón en W_v, del modo longitudinal. El código utilizado para una señal escalón en V_A, y para el modo lateral será similar. La variable Ts que aparece en el código es el periodo de muestreo de las señales.

Robustez. Con objeto de dotar a la función de evaluación de una medida de robustez, se incorporará un procedimiento que se encargue de calcular el margen de ganancia y el margen de fase.

Sea el sistema de control mostrado en la figura 3.13, donde $G(s)$ y $H(s)$ son las matrices de funciones de transferencia de la planta y el controlador respectivamente, u_o

y u_i son vectores de entradas al sistema, y_o e y_i son los vectores de salida de la planta y el controlador respectivamente; y e_o, e_i son vectores de señales de error.

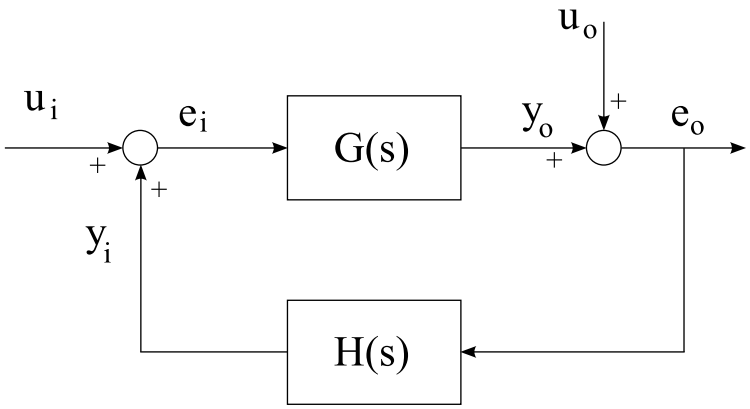


Figura 3.13: Sistema de control.

Se define la función de sensibilidad S, como la matriz de transferencia de la entrada u a la salida e (u_i, e_i para el nodo de entrada, y u_o, e_o para el nodo de salida). Por su parte, la función de sensibilidad complementaria T se define como la matriz de transferencia de la entrada u a la señal de retorno del lazo, y, correspondiente. Por último, la función de sensibilidad compensada se define como la matriz de transferencia de u a $e + y$.

De la figura 3.13, se deduce inmediatamente que

$$e = (I - L)^{-1}u \tag{3.8}$$

siendo $L = H \cdot G$ a la entrada de los actuadores y $L = G \cdot H$ a la salida (se hace notar que la matriz L correspondiente a la entrada será en general distinta a la matriz L correspondiente a la salida, al no cumplir el producto de matrices la propiedad conmutativa). Por tanto de la ecuación 3.8 se obtiene que $S = (I - L)^{-1}$.

Por otro lado, teniendo en cuenta también la ecuación 3.8, se obtiene

$$y = Le \Rightarrow y = L(I - L)^{-1}u \tag{3.9}$$

siendo por tanto $T = L(I - L)$.

Finalmente, sumando las ecuaciones 3.8 y 3.9, se obtiene

$$e + y = (I - L)^{-1}u + L(I - L)^{-1}u = (I + L)(I - L)^{-1}u \tag{3.10}$$

siendo por tanto $S + T = (I + L)(I - L)^{-1}$.

Los valores singulares de las funciones de sensibilidad S, T y $S+T$, pueden utilizarse para medir los márgenes de estabilidad de un sistema de control realimentado: el valor del pico del valor singular máximo ($\overline{\sigma}$) de S, T o $S + T$, respecto a la frecuencia, da una medida de robustez [17], [31] y [73].

A continuación se indica la forma de obtener los márgenes de ganancia MG y fase MF utilizando S, T y $S + T$.

Utilizando la función S:

$$\begin{aligned} K_m &= 1/\overline{\sigma}(S) \\ MG &= [1/(1 + K_m), 1/(1 - K_m)] \qquad (3.11) \\ MF &= \pm 2\, arcsen(K_m/2) \qquad (3.12) \end{aligned}$$

donde $K_m \leq 1$. La interpretación de (3.11) es la siguiente: las ganancias de los lazos pueden ser perturbadas simultáneamente por ganancias Δ_i que cumplan $1/(1+K_m) < \Delta_i < 1/(1 - K_m)$ sin desestabilizar el sistema de lazo cerrado. Similarmente para (3.12), los lazos de realimentación pueden ser perturbados simultáneamente por fases ϕ_i que cumplan $|\phi_i| < 2\, arcsen(K_m/2)$ sin desestabilizar el sistema de lazo cerrado. Los mejores márgenes de ganancia y fase se obtienen cuando $\overline{\sigma}(S) = 1$, siendo en este caso el margen de ganancia $[-6\, dB, +\infty\, dB]$ y el margen de fase $\pm 60^{\text{o}}$.

Utilizando la función T:

Cuando se utiliza la función T, las expresiones a emplear son las siguientes:

$$\begin{aligned} K_m &= 1/\overline{\sigma}(T) \\ MG &= [1 - K_m, 1 + K_m] \qquad (3.13) \\ MF &= \pm 2\, arcsen(K_m/2) \qquad (3.14) \end{aligned}$$

donde $K_m \leq 1$. La interpretación es similar al caso de la función de sensibilidad S. Los

mejores márgenes de ganancia y fase tienen lugar cuando $\overline{\sigma}(T) = 1$ y son en este caso $[-\infty\, dB, 6\, dB]$ y $\pm 60^{o}$ respectivamente.

Utilizando la función S+T:

Por último, cuando se utilice la función de sensibilidad compensada se emplearán las siguientes expresiones:

$$\begin{aligned} K_m &= 1/\overline{\sigma}(S+T) \\ MG &= [(1-K_m)/(1+K_m), (1+K_m)/(1-K_m)] \qquad (3.15) \\ MF &= \pm 2\, arctan(K_m) \qquad (3.16) \end{aligned}$$

donde $K_m \leq 1$. En este caso, los mejores valores posibles para el margen de ganancia y el margen de fase son $[-\infty\, dB, \infty\, dB]$ y $\pm 90^{o}$ respectivamente, y tienen lugar cuando $\overline{\sigma}(S+T) = 1$.

En la función de evaluación empleada, se utilizaron los márgenes de ganancia y márgenes de fase calculados a partir de la función $S+T$.

3.4.3.2. Función de evaluación para el problema de control robusto de un buque de alta velocidad.

Para el modelo CRIBAV, la función de evaluación es más sencilla que la del modelo RCAM detallada en los apartados anteriores. Éste fue el motivo por el que se seleccionó este problema para intentar la selección y sintonía de una estructura de control. Al ser un problema con unos requerimientos de cálculo muy elevados, no se consideró la robustez con objeto de simplificar la función de evaluación. Los elementos considerados en la función de evaluación fueron los siguientes:

1. El sistema deberá ser estable. Si este criterio no se cumple, no se continuará con los siguientes.
2. El valor de la posición del *Flap* deberá estar comprendido entre 0^{o} y 15^{o}.
3. El valor de la posición del *T-Foil* deberá estar comprendido entre -15^{o} y 15^{o}.
4. Si se cumplen todos los criterios anteriores, el objetivo será minimizar las aceleraciones verticales del buque, dadas por la expresión 3.7.

La forma de proceder para la obtención de los distintos valores es la misma que la vista para el caso del problema del modelo RCAM, por lo que no se considera necesaria ninguna explicación adicional.

3.4.3.3. Normalización y construcción del vector de aptitud.

Los distintos objetivos incluidos en los vectores de aptitud se normalizarán de tal modo que el objetivo será satisfecho cuando tome el valor cero. Valores negativos indicarán que el objetivo no se ha cumplido aún y estará más lejos de cumplirse cuanto menor sea su valor (mayor en valor absoluto). Valores positivos indicarán que los resultados han superado los objetivos esperados.

En lo que sigue se detalla la forma en que se normalizan los distintos objetivos del problema del modelo RCAM. Para el caso del problema del modelo CRIBAV se procederá de forma análoga. Los parámetros a normalizar se pueden incluir en alguno de los siguientes grupos:

1. **Objetivos cuyos posibles valores se encuentren en el intervalo $[0, \infty)$, siendo 0 el mejor valor posible.** En este grupo se encuentran la mayor parte de los objetivos considerados. Como ejemplo se considerará el caso del máximo sobreimpulso de w_V frente a una entrada de tipo escalón de $4.2m/s$ en la referencia de w_V, que como se vio en el apartado 3.4.3.1 deberá ser menor que $0.21m/s$. Sea O el valor del sobreimpulso de w_V, el valor normalizado vendrá dado por la expresión:

$$O_N = 1 - \frac{O}{0.21} \tag{3.17}$$

2. **Objetivos que pueden tomar valores positivos y negativos, siendo 0 el mejor valor posible.** En este grupo estarían las señales de control δ_T, δ_{TH}, δ_A y δ_R. En este caso se procederá de igual forma que en el grupo anterior, pero utilizando expresiones diferentes para el caso de los valores positivos y para el de los negativos. Para estos últimos, se trabajará con sus valores absolutos. Como ejemplo se considerará el caso del valor de δ_T, que deberá estar comprendido entre $-0.3215rad$ y $0.2894rad$. Sea T el valor que toma δ_T. Si $T \geq 0$ se normalizará mediante la expresión:

$$T_N = 1 - \frac{T}{0.2894} \tag{3.18}$$

Sin embargo, si $T < 0$ se utilizará la expresión:

$$T_N = 1 - \frac{|T|}{0.3215} \tag{3.19}$$

3. **Objetivos cuyos posibles valores se encuentren en el intervalo $[0, \infty)$, siendo ∞ el mejor valor posible.** En este grupo se encontraría únicamente el margen de ganancia. Para hacer que siga siendo cero el mejor valor, se trabajará con los valores inversos. Sea G el valor que toma el margen de ganancia, que deberá ser mayor que $10dB$. Su valor normalizado vendrá dado por la expresión:

$$G_N = 1 - \frac{10}{G} \tag{3.20}$$

4. **Objetivos cuyos posibles valores se encuentren en el intervalo $[0, 90^{\mathbf{o}})$, siendo $90^{\mathbf{o}}$ el mejor valor posible.** En este grupo se encontraría únicamente el margen de fase. Para convertirlo en el intervalo $[0, \infty)$ con cero como mejor valor posible, se ideó trabajar con la *cotg* de los valores, obteniéndose buenos resultados. Sea F el valor que toma el margen de fase, que deberá ser mayor que 50^{o}. Su valor normalizado vendrá dado por la expresión:

$$F_N = 1 - \frac{cotg(F)}{cotg(50^{\text{o}})} \tag{3.21}$$

Los objetivos considerados tienen distintas prioridades, pues en primer lugar se considerará la mayor de las partes reales de los autovalores, y si este objetivo no se cumple, no se continuará evaluando el resto. Por ello, la función de evaluación construirá el vector de aptitud en la siguiente forma:

1. Se crea un vector de aptitud con tantos elementos como número de objetivos se estén considerando, y se les da a todos el valor de $-\infty$. Para n elementos se tendría:

$$F_{fit} = \{\overbrace{-\infty, \ldots, -\infty}^{\text{n elementos}}\}$$

2. Se asigna un valor al primer elemento del vector de aptitud (el correspondiente al máximo de las partes reales de los autovalores) de la siguiente forma:

   - Si el valor máximo de las partes reales de los autovalores es mayor que el valor requerido, se establecerá como primer elemento la distancia entre ambos valores con signo negativo.

- Por el contrario, si el objetivo se cumple, se asignará al primer elemento del vector de aptitud el valor 1.

3. Se procede con el resto de valores del vector de aptitud de la siguiente forma:
   - Si no se cumplió el primer objetivo, se dejará el resto de elementos del vector de aptitud con el valor $-\infty$.
   - Sin embargo, si se cumplió el primer objetivo, se evaluarán todos los demás y se asignará a cada elemento del vector de aptitud su valor correspondiente normalizado.

Con objeto de clarificar un poco lo anterior, se verá el ejemplo 3.1, en el que se ilustran diferentes casos posibles. Se considerarán vectores de aptitud con 10 elementos, si bien éste no será el número de elementos que realmente se emplee en los problemas que serán tratados en los capítulos siguientes. Al elegir un número de elementos menor al real, se consigue una mayor claridad, sin perjudicar ningún otro aspecto.

Ejemplo 3.1 *Vectores de aptitud.*

- *Vector de aptitud correspondiente a un caso en el que no se cumple el primer objetivo (el relativo al máximo de los valores reales del los autovalores).*

$$\{-3.25,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty\}$$

- *Vector de aptitud correspondiente a un caso en el que únicamente se cumple el primer objetivo.*

$$\{1,\ -0.12,\ -0.32,\ -0.41,\ -0.28,\ -1.7,\ -0.43,\ -0.42,\ -0.65,\ -0.12\}$$

- *Vector de aptitud correspondiente a un caso en el que se cumplen los objetivos 1, 2, 5 y 8, y no se cumple el resto.*

$$\{1,\ 0.15,\ -0.62,\ -0.21,\ 0.07,\ -1.5,\ -1.04,\ 0.23,\ -0.5,\ -0.02\}$$

- *Vector de aptitud correspondiente a un caso en el que se cumplen todos los objetivos.*

$$\{1,\ 0.15,\ 0.12,\ 0.1,\ 0.01,\ 0.23,\ 0.12,\ 0.23,\ 0.05,\ 0.21\}$$

3.4.4. Ordenando la población.

Tras la evaluación de todos los cromosomas de la población, hay que proceder a su ordenación, con objeto de poder aplicar las técnicas de selección de padres que se verán en el siguiente apartado.

El problema que se está tratando es un problema multiobjetivo, en el que se pretenden optimizar una serie de parámetros que, como ya se ha visto, fueron incluidos en la función de evaluación.

En el capítulo 2, se trató el tema de la optimización multiobjetivo y se vieron distintas estrategias que pueden ser empleadas a la hora de comparar dos posibles soluciones al problema. En el apartado 2.5, se vio el *método de prioridades variables para optimización multiobjetivo evolutiva*, que como ya se mencionó es el empleado en esta tesis.

A continuación se explicará el modo en que se aplica el *método de las prioridades variables* para llevar a cabo la ordenación de la población. El punto de partida serán los vectores de aptitud de los cromosomas que forman la población, debidamente normalizados.

Para realizar la ordenación, se deberá disponer de un método que permita comparar dos cromosomas y decidir cual de ellos es mejor. A partir de aquí se podrían seguir básicamente dos estrategias de ordenación: generar una nueva población de cromosomas y luego ordenarla, o ir insertando los nuevos cromosomas que se vayan produciendo en la posición que les corresponde atendiendo a su aptitud. Se escogió la segunda opción por ser más eficaz computacionalmente hablando.

Se describe ahora la forma de aplicar el método de las prioridades variables en la comparación de las aptitudes de dos cromosomas:

1. Se reordenan los elementos de los dos vectores de aptitud en orden creciente.
2. Se comienzan a comparar los elementos de ambos vectores de aptitud:
   - Si el primer elemento del primer vector es mayor que el primer elemento del segundo vector, el primer cromosoma será el más apto.
   - Si el primer elemento del segundo vector es mayor que el primer elemento del primer vector, el segundo cromosoma será el más apto.
   - Si el primer elemento del primer vector es igual al primer elemento del segundo vector, habrá que acudir al segundo elemento de ambos vectores. Se

compararán dichos elementos siguiendo los mismos criterios que se siguieron para los primeros. Si volviera a producirse la igualdad en los segundos elementos se continuará con los siguientes hasta deshacer la igualdad. En el caso improbable de que no se deshaga la igualdad, el primer cromosoma será considerado más apto.

El ejemplo 3.2 muestra la comparación de vectores de aptitud, correspondientes a diferentes casos.

Ejemplo 3.2 *Comparación de vectores de aptitud.*

- Comparación de dos vectores de aptitud en los que no se cumple el primer objetivo:

$$f_1 = \{-2.25,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty\}$$
$$f_2 = \{-1.5,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty\}$$

  En este caso no será necesaria la reordenación y simplemente se deberán comparar el primer elemento de cada vector, siendo el cromosoma más apto, el correspondiente a f_2.

- Comparación de dos vectores de aptitud en los que en el primero se cumple el primer objetivo y en el segundo no:

$$f_1 = \{1,\ 0.15,\ -0.62,\ -0.21,\ 0.07,\ -1.5,\ -1.04,\ 0.23,\ -0.5,\ -0.02\}$$
$$f_2 = \{-2.3,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty\}$$

  Tras la reordenación se obtiene:

$$f_1^* = \{-1.5,\ -1.04,\ -0.62,\ -0.5,\ -0.21,\ -0.02,\ 0.07,\ 0.15,\ 0.23,\ 1\}$$
$$f_2^* = \{-\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -\infty,\ -2.3\}$$

  Al comparar el primer elemento de cada vector, se obtiene que el cromosoma más apto es el correspondiente a f_1.

- Comparación de dos vectores de aptitud en los que se cumple el primer objetivo:

$$f_1 = \{1,\ -0.11,\ -0.2,\ 0.31,\ 0.18,\ -1.05,\ -2.04,\ 0.37,\ -0.05,\ 0.04\}$$
$$f_2 = \{1,\ -0.15,\ -0.72,\ -2.04,\ 0.01,\ -0.45,\ 0.11,\ 0.12,\ -0.05,\ 0.32\}$$

Tras la reordenación se obtiene:

$$f_1^* = \{-2.04,\ -1.05,\ -0.2,\ -0.11,\ -0.05,\ 0.04,\ 0.18,\ 0.31,\ 0.37,\ 1\}$$
$$f_2^* = \{-2.04,\ -0.72,\ -0.45,\ -0.15,\ -0.05,\ 0.01,\ 0.11,\ 0.12,\ 0.32,\ 1\}$$

En este caso, al ser idénticos los primeros elementos de ambos vectores tras la reordenación, habrá que acudir al segundo elemento, obteniéndose que el cromosoma más apto es el correspondiente a f_2.

3.4.5. La selección de padres.

En el apartado 1.3.4 se estudió la selección de padres y las técnicas que pueden utilizarse para llevarla a cabo. Al ser vectores las aptitudes con las que se van a tratar, el *método de la ruleta* presenta dificultades de aplicación, pues como se vio en el apartado 1.3.4.1 la probabilidad de selección de un cromosoma era directamente proporcional a su índice de aptitud, debiendo ser éste, por tanto, escalar. Una opción sería asignar un valor a la probabilidad de selección en función de la posición que ocupe el cromosoma en la población, aunque en este caso surgiría la dificultad de en qué forma se distribuía la probabilidad entre los distintos cromosomas.

Ante los inconvenientes que presenta el método de la ruleta, se optó por la utilización del *método del torneo o concurso*, visto en el apartado 1.3.4.2. Este método presenta entre sus ventajas, la posibilidad de trabajar de forma natural con vectores de aptitud, sin necesidad de ninguna modificación.

Además el método de torneo es muy sencillo de aplicar y con muy pocos requerimientos computacionales, lo que es de agradecer cuando se emplea en algoritmos complejos, en los que el tiempo es un factor importante.

El único parámetro del método que requerirá ser ajustado es el del número de concursantes que participan en cada torneo. El valor que se eligió fue 2, por ser el valor más habitual; con este valor se obtuvieron buenos resultados.

3.4.6. Generación de la descendencia.

Tras la selección de los cromosomas padres, el siguiente paso será la obtención de los cromosomas que compondrán la nueva generación. Para ello se utilizarán los operadores habituales.

Como ya se comentó en el apartado 3.4.2, en los capítulos siguientes se trabajará con dos tipos de cromosomas: cromosomas con codificación binaria, y cromosomas con codificación en forma de números reales. Dependiendo del tipo de codificación, se podrán usar unos operadores u otros, según se vio en el apartado 1.3.5.

Con independencia del tipo de codificación empleada y de los operadores utilizados, la forma en que se procedió consiste básicamente en lo siguiente:

1. Se asigna una probabilidad a cada uno de los operadores que se quiera utilizar.
2. Se toman dos padres de los seleccionados durante el proceso de selección de padres.
3. Se determina de forma aleatoria el operador a emplear.
4. Se aplica el operador elegido a los dos cromosomas escogidos en el punto 2, y se generan dos descendientes.
5. Si quedan aún padres disponibles, se vuelve al punto 2; si no, se completa la población con inmigrantes.

3.4.7. Condiciones de finalización.

Como se vio en el apartado 1.3.7, el algoritmo necesita conocer de alguna forma cuándo debe finalizar. También se vieron distintas posibilidades existentes.

En los trabajos que se verán en los capítulos siguientes se utilizó el método interactivo siempre que fue posible. Esto ocurrió en todos los casos excepto en aquel en el que el algoritmo se encargaba de sintonizar una estructura de control, determinada por otro algoritmo en un lazo externo.

Con el método interactivo lo que se hace es encargar al algoritmo un número determinado de generaciones, estudiando luego los resultados obtenidos. Si se considera que existen aún posibilidades de mejora, se vuelve a encargar al algoritmo un nuevo número de generaciones. El procedimiento se repite las veces que se consideran oportunas.

En el caso de la excepción mencionada, no era posible aplicar el método interactivo en el lazo interno, puesto que éste debía ejecutarse en múltiples ocasiones durante el funcionamiento del lazo externo y no existía la posibilidad de interactuar con él. Esto se verá mejor en el capítulo correspondiente, cuando se estudie este algoritmo.

3.4.8. Algoritmo utilizado.

A partir de lo visto en los últimos apartados, se construirá el algoritmo que será empleado en los capítulos siguientes. En la figura 3.14 se muestra su diagrama de flujo.

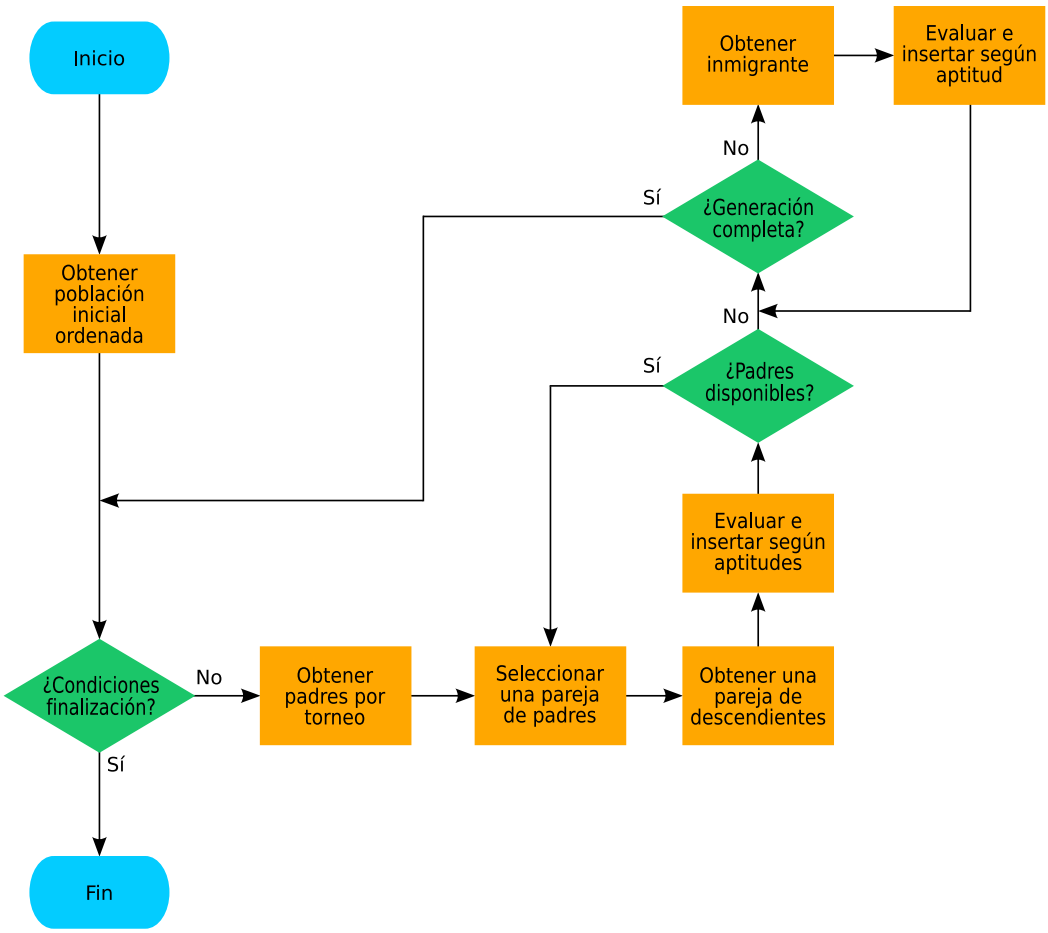


Figura 3.14: Algoritmo utilizado.

Los pasos son los siguientes:

1. Se crea la generación inicial de forma aleatoria. A medida que se van creando los cromosomas, se van evaluando e insertando en la población, en la posición que les corresponde según su aptitud.

2. Si se dan las condiciones de finalización, el algoritmo concluye, si no, se continúa con el paso siguiente. En este caso, si se está operando en modo interactivo, las condiciones de finalización serán haber obtenido el número de generaciones solicitadas.

3. Mediante la técnica de torneo o concurso, se obtienen los que serán los padres de los cromosomas de la siguiente generación.
4. Se selecciona una pareja de padres, de los obtenidos en el punto anterior.
5. Mediante la aplicación de un operador genético obtenido de forma aleatoria, se obtienen dos descendientes.
6. Se evalúan los dos descendientes y se incluyen en la nueva generación, ocupando el puesto que les corresponde en función de sus aptitudes.
7. Si quedan padres de los obtenidos en el punto 3, sin haber sido utilizados, se vuelve al punto 4. Si no, se continúa con el punto siguiente.
8. Si la nueva generación está completa, se vuelve al punto 2, si no, se continúa con el punto siguiente.
9. Se genera un nuevo cromosoma de forma aleatoria (inmigrante).
10. Se evalúa y se incluye en la nueva generación, ocupando el lugar que le corresponde por su aptitud.
11. Se vuelve al punto 8.

3.4.9. Conclusiones.

En el presente capítulo se han visto los modelos con los que va a trabajarse en los capítulos posteriores y se ha diseñado el algoritmo evolutivo que se utilizará.

Los problemas que se tratarán, son típicos problemas de optimización multiobjetivo, para los cuales se estudiaron diversas técnicas en el capítulo 2, habiéndose escogido de todas ellas el *método de prioridades variables para optimización multiobjetivo evolutiva* visto en el apartado 2.5. También hubo que tomar diversas decisiones relativas a los parámetros del algoritmo: tamaño de las poblaciones empleadas, tipos de operadores a utilizar, método empleado para la selección de padres, etc. Las opciones elegidas estaban en general basadas en lo que se puede encontrar con frecuencia en la bibliografía. En algún caso en el que había una razón especial se detalló convenientemente, como en el caso de la elección del método de torneo en la selección de padres.

Los algoritmos evolutivos tienen una importante componente aleatoria, por lo que habrá que decidir la forma en la que se generarán los números aleatorios que precisa el algoritmo. Una opción sería utilizar una densidad de probabilidad uniforme, disponible

en prácticamente todos los entornos de programación. Sin embargo, se ha demostrado que esta opción no es necesariamente la mejor, habiéndose diseñado una función de densidad que ofrece mejores resultados.

Un punto importante, y quizá en el que se ha hecho más hincapié, es el diseño de la función de evaluación que determinará las aptitudes de los distintos cromosomas de la población. La construcción de dicha función es muy flexible, pues permite incluir en ella cualquier parámetro del que se pueda obtener una medida. Se ha visto que han podido incluirse variables típicas como tiempo de subida, máximo sobreimpulso, magnitud de los errores, saturaciones de las señales de control, y también medidas de robustez.

Se han visto algunas ventajas que presenta el tipo de algoritmos empleado. Entre ellas destaca la de permitir trabajar desde una perspectiva más próxima al usuario, empleándose directamente las especificaciones de diseño.

Falta ahora llevar el método a la práctica y ver el comportamiento del algoritmo en la resolución de problemas concretos. Esto es lo que se verá en los capítulos que vienen a continuación.

Capítulo 4

El uso de los algoritmos evolutivos en el proceso de sintonía de un controlador clásico: diseño de un controlador LQ

4.1. Introducción.

El control óptimo cuadrático lineal es una técnica muy conocida y utilizada, desde que sus bases fueran establecidas en los años 50 del pasado siglo. Desde entonces se han escrito gran cantidad de libros sobre el tema [2], [11], [76], [89] y [93].

Esta técnica de control permite tener en cuenta tanto los requerimientos de amplitud de la señal de control, como el tiempo de asentamiento de las variables de estado. Sin embargo, no se trata directamente con las especificaciones de diseño. Además, el método se basa en la minimización de una función de coste en la que se incluyen dos matrices de peso, $\boldsymbol{Q}$ y $\boldsymbol{R}$, que determinan la importancia relativa entre la diferencia de los elementos del vector de estados respecto a cero y el gasto de energía en las señales de control. Mediante la variación de estas matrices de peso se puede sintonizar el controlador, hasta conseguir que el sistema se comporte de la forma deseada.

No existe, sin embargo, un procedimiento general que indique los pasos a seguir a

la hora de seleccionar dichas matrices de peso. Conviene destacar que cada problema particular tiene sus propias especificaciones de diseño y que la elección de las matrices de peso ha de hacerse teniendo en cuenta dichas especificaciones. Se han realizado estudios buscando reglas que ayuden en la elección de $\boldsymbol{Q}$ y $\boldsymbol{R}$ [2], pero se asume que finalmente habrá que recurrir a métodos iterativos para la obtención de dichos valores.

Resulta pues que el control óptimo LQ, desarrollado de forma elegante desde el punto de vista matemático y con solución analítica, requiere finalmente apoyarse en métodos iterativos a la hora de ser llevado a la práctica.

Viendo las dificultades que plantea la selección de las matrices $\boldsymbol{Q}$ y $\boldsymbol{R}$, se advierte la posibilidad de realizar dicha selección por otros métodos. Los algoritmos evolutivos serán la opción elegida. Mediante ellos, se determinarán las matrices de peso, de forma que se cumplan las especificaciones de diseño, teniendo en cuenta la robustez del sistema.

4.2. Control LQ.

El problema del control óptimo LQ ha sido abordado por distintos autores, empleando distintas herramientas, como el cálculo de variaciones [11] y [76], o la programación dinámica desarrollada por P.E. Bellman a finales de los años 50 del pasado siglo, como una alternativa a la aproximación variacional para el control óptimo [76] y [89].

En lo que sigue, se verá la solución al problema mediante el segundo método de Liapunov. También se estudiará una forma de convertir un problema de control en otro de regulación y poder aplicarle el método visto.

4.2.1. Control óptimo con función de coste cuadrática.

Sea el sistema invariante en el tiempo, representado por la ecuación:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.1}$$

donde

$\boldsymbol{x}$ = vector de estados n-dimensional.
$\boldsymbol{u}$ = vector de control r-dimensional.
$\boldsymbol{A}$ = matriz constante de n×n.

$\boldsymbol{B}$ = matriz constante de n×r.

Se ha de determinar la matriz $\boldsymbol{K}$ del vector de control óptimo

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t) \tag{4.2}$$

que minimice la función de coste

$$J = \int_0^\infty \left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\right) dt \tag{4.3}$$

donde $\boldsymbol{Q}$ es una matriz simétrica real definida positiva (o semidefinida positiva) y $\boldsymbol{R}$ es una matriz simétrica real definida positiva. Las matrices $\boldsymbol{Q}$ y $\boldsymbol{R}$ determinan la importancia relativa del error y el gasto de energía de control.

Se puede demostrar que la ley de control lineal dada por la ecuación 4.2 es la ley de control óptimo. Por tanto, si se determinan los elementos desconocidos de la matriz $\boldsymbol{K}$ de manera que se haga mínimo el valor de la función de coste, entonces $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t)$ es óptima para cualquier estado inicial $\boldsymbol{x}(0)$. En la figura 4.1 se muestra el diagrama de bloques del sistema.

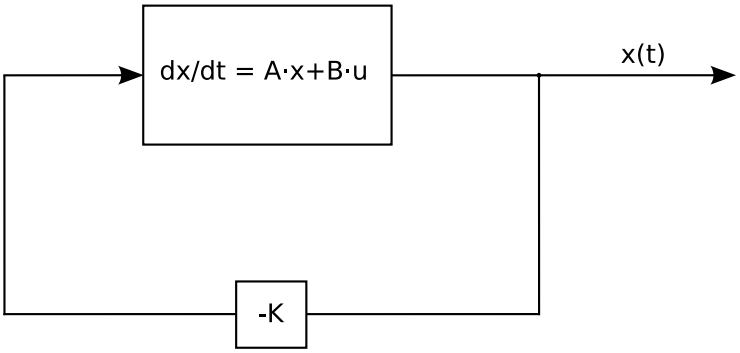


Figura 4.1: Sistema de control óptimo.

Reemplazando la ecuación 4.2 en la ecuación 4.1 se obtiene

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{x} \tag{4.4}$$

En lo que sigue se supone que la matriz $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ es estable, o que todos sus autovalores tienen partes reales negativas. Reemplazando la ecuación 4.2 en 4.3 se

obtiene

$$J = \int_0^\infty \left(\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} \boldsymbol{x} \right) dt = \int_0^\infty \boldsymbol{x}^T \left(\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} \right) \boldsymbol{x} dt \tag{4.5}$$

Suponiendo que

$$\boldsymbol{x}^T \left(\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} \right) \boldsymbol{x} = -\frac{d}{dt} \left(\boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} \right) \tag{4.6}$$

siendo $\boldsymbol{P}$ una matriz simétrica real definida positiva, se obtiene

$$\begin{aligned} \boldsymbol{x}^T \left(\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} \right) \boldsymbol{x} = -\dot{\boldsymbol{x}}^T \boldsymbol{P} \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{P} \dot{\boldsymbol{x}} = \\ -\boldsymbol{x}^T \left[\left(\boldsymbol{A} - \boldsymbol{B} \boldsymbol{K} \right)^T \boldsymbol{P} + \boldsymbol{P} \left(\boldsymbol{A} - \boldsymbol{B} \boldsymbol{K} \right) \right] \boldsymbol{x} \end{aligned} \tag{4.7}$$

Comparando términos en 4.7, y teniendo en cuenta que la ecuación debe ser válida para cualquier valor de $\boldsymbol{x}$, tiene que cumplirse

$$\left(\boldsymbol{A} - \boldsymbol{B} \boldsymbol{K} \right)^T \boldsymbol{P} + \boldsymbol{P} \left(\boldsymbol{A} - \boldsymbol{B} \boldsymbol{K} \right) = - \left(\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} \right) \tag{4.8}$$

Por el segundo método de Liapunov, si la matriz $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ es estable, existe una matriz $\boldsymbol{P}$ definida positiva que satisface la ecuación 4.8.

Se puede evaluar la función de coste a partir de las ecuaciones 4.5 y 4.6 como

$$\begin{aligned} J = \int_0^\infty \boldsymbol{x}^T \left(\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} \right) \boldsymbol{x} dt = -\boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} \Big|_0^\infty = \\ -\boldsymbol{x}^T (\infty) \boldsymbol{P} \boldsymbol{x} (\infty) + \boldsymbol{x}^T (0) \boldsymbol{P} \boldsymbol{x} (0) \end{aligned} \tag{4.9}$$

Entonces, para $\boldsymbol{x}^T(\infty) = 0$, se puede escribir la función de coste como

$$J = \boldsymbol{x}^T (0) \boldsymbol{P} \boldsymbol{x} (0) \tag{4.10}$$

Como se ha supuesto que $\boldsymbol{R}$ es una matriz simétrica real definida positiva, se puede hacer

$$\boldsymbol{R} = \boldsymbol{T}^T \boldsymbol{T} \tag{4.11}$$

donde $\boldsymbol{T}$ es una matriz no singular. Entonces se puede escribir la ecuación 4.8 como

$$\left(\boldsymbol{A}^T - \boldsymbol{K}^T \boldsymbol{B}^T\right) \boldsymbol{P} + \boldsymbol{P} \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\right) + \boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{T}^T \boldsymbol{T} \boldsymbol{K} = 0 \tag{4.12}$$

que puede ponerse en la forma

$$\boldsymbol{A}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \left[\boldsymbol{T}\boldsymbol{K} - \left(\boldsymbol{T}^T\right)^{-1} \boldsymbol{B}^T \boldsymbol{P}\right]^T \left[\boldsymbol{T}\boldsymbol{K} - \left(\boldsymbol{T}^T\right)^{-1} \boldsymbol{B}^T \boldsymbol{P}\right] - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = 0 \tag{4.13}$$

Se puede demostrar que la minimización de J respecto a $\boldsymbol{K}$ requiere la minimización de

$$\boldsymbol{x}^T \left[\boldsymbol{T}\boldsymbol{K} - \left(\boldsymbol{T}^T\right)^{-1} \boldsymbol{B}^T \boldsymbol{P}\right]^T \left[\boldsymbol{T}\boldsymbol{K} - \left(\boldsymbol{T}^T\right)^{-1} \boldsymbol{B}^T \boldsymbol{P}\right] \boldsymbol{x} \tag{4.14}$$

con respecto a $\boldsymbol{K}$. Como este valor no es negativo, el mínimo tiene lugar cuando es cero, o cuando

$$\boldsymbol{T}\boldsymbol{K} = (\boldsymbol{T}^T)^{-1} \boldsymbol{B}^T \boldsymbol{P} \tag{4.15}$$

Por tanto,

$$\boldsymbol{K} = \boldsymbol{T}^{-1} (\boldsymbol{T}^T)^{-1} \boldsymbol{B}^T \boldsymbol{P} = \boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{P} \tag{4.16}$$

La ecuación 4.16 calcula la matriz óptima $\boldsymbol{K}$. La matriz $\boldsymbol{P}$ de dicha ecuación, debe satisfacer la ecuación 4.8, o la ecuación reducida siguiente (obtenida a partir de las ecuaciones 4.13 y 4.15):

$$\boldsymbol{A}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = 0 \tag{4.17}$$

La ecuación 4.17 recibe el nombre de ecuación de la matriz reducida de Riccati. Los pasos de diseño son los siguientes:

1. Se obtiene la matriz $\boldsymbol{P}$ a partir de la ecuación 4.17.

2. Se sustituye en la ecuación 4.16 la matriz $\boldsymbol{P}$ obtenida en el paso anterior. La matriz $\boldsymbol{K}$ resultante es la óptima.

En el presente trabajo, se utilizará la función de Matlab

```
lqr(A, B, Q, R)
```

que resuelve el problema del regulador cuadrático lineal en tiempo continuo, y su ecuación de Riccati asociada. Esta función calcula la matriz de ganancia $\boldsymbol{K}$ tal que la ley de control

$$\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$$

minimiza la función de coste

$$J = \int_0^\infty \left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\right) dt$$

sujeta a la restricción

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

4.2.2. Conversión de un problema de control en otro de regulación equivalente.

En el apartado 4.2.1 se ha visto como resolver un problema de regulación. Ahora se verá como convertir un problema de control, en el que la planta debe seguir una señal de referencia constante, en otro de regulación equivalente, con objeto de poder aplicar lo anterior. El esquema del sistema se muestra en la figura 4.2, donde la matriz $\boldsymbol{K}_r$ se encargará de seleccionar las salidas que deben ser comparadas con la señal de referencia y generar la señal de error que será la entrada del compensador.

La planta estará representada por las siguientes ecuaciones en el espacio de estados

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_p & = & \boldsymbol{A}_p \cdot \boldsymbol{x}_p + \boldsymbol{B}_p \cdot \boldsymbol{u}_p \\ \boldsymbol{y}_p & = & \boldsymbol{C}_p \cdot \boldsymbol{x}_p + \boldsymbol{D}_p \cdot \boldsymbol{u}_p \end{array}\right\} \tag{4.18}$$

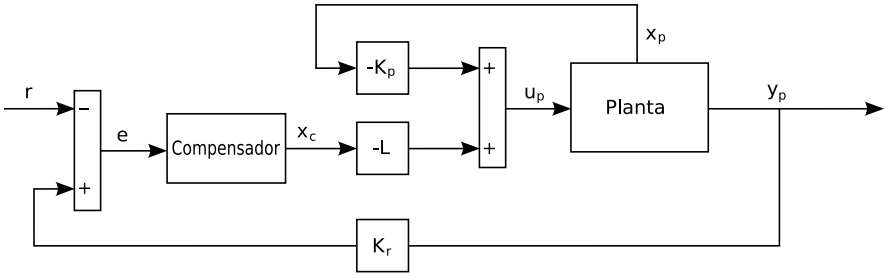


Figura 4.2: Sistema de control.

El compensador por su parte responderá a ecuaciones de la forma

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_c & = & \boldsymbol{A}_c \cdot \boldsymbol{x}_c + \boldsymbol{B}_c \cdot \boldsymbol{e} \\ \boldsymbol{y}_c & = & \boldsymbol{x}_c \end{array}\right\} \tag{4.19}$$

siendo

$$\boldsymbol{e} = \boldsymbol{K}_r \boldsymbol{y}_p - \boldsymbol{r} \tag{4.20}$$

Los integradores, por ejemplo, se ajustan a las ecuaciones de estado del compensador. La señal de control de la planta, según puede verse en la figura 4.2, es de la forma

$$\boldsymbol{u}_p = -\boldsymbol{K}_p \boldsymbol{x}_p - \boldsymbol{L}\boldsymbol{x}_c \tag{4.21}$$

Y teniendo en cuenta que

$$\dot{\boldsymbol{x}}_c = \boldsymbol{A}_c \boldsymbol{x}_c + \boldsymbol{B}_c \left(\boldsymbol{K}_r \boldsymbol{y}_p - \boldsymbol{r}\right) = \boldsymbol{A}_c \boldsymbol{x}_c + \boldsymbol{B}_c \boldsymbol{K}_r \boldsymbol{C}_p \boldsymbol{x}_p + \boldsymbol{B}_c \boldsymbol{K}_r \boldsymbol{D}_p \boldsymbol{u}_p - \boldsymbol{B}_c \boldsymbol{r} \tag{4.22}$$

se pueden agrupar las dinámicas de la planta y el compensador en la ecuación

$$\begin{bmatrix} \dot{\boldsymbol{x}}_p \\ \dot{\boldsymbol{x}}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_p & \boldsymbol{0} \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{C}_p & \boldsymbol{A}_c \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{D}_p \end{bmatrix} \cdot \boldsymbol{u}_p + \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{B}_c \end{bmatrix} \cdot \boldsymbol{r} \quad (4.23)$$

En lo que sigue se supondrá una señal de referencia, $\boldsymbol{r}$, constante. Indicando con una barra sobre las variables sus valores en el estacionario, y con una tilde sus desviaciones respecto a este valor, se puede escribir (suponiendo n estados):

$$\boldsymbol{0}_{n\times 1} = \begin{bmatrix} \boldsymbol{A}_p & \boldsymbol{0} \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{C}_p & \boldsymbol{A}_c \end{bmatrix} \cdot \begin{bmatrix} \overline{\boldsymbol{x}}_p \\ \overline{\boldsymbol{x}}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{D}_p \end{bmatrix} \cdot \overline{\boldsymbol{u}}_p + \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{B}_c \end{bmatrix} \cdot \boldsymbol{r} \quad (4.24)$$

Restando las ecuaciones 4.23 y 4.24 se obtiene

$$\begin{bmatrix} \dot{\tilde{\boldsymbol{x}}}_p \\ \dot{\tilde{\boldsymbol{x}}}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_p & \boldsymbol{0} \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{C}_p & \boldsymbol{A}_c \end{bmatrix} \cdot \begin{bmatrix} \tilde{\boldsymbol{x}}_p \\ \tilde{\boldsymbol{x}}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{D}_p \end{bmatrix} \cdot \tilde{\boldsymbol{u}}_p \quad (4.25)$$

siendo

$$\tilde{\boldsymbol{u}}_p = -\begin{bmatrix} \boldsymbol{K}_p & \boldsymbol{L} \end{bmatrix} \cdot \begin{bmatrix} \tilde{\boldsymbol{x}}_p \\ \tilde{\boldsymbol{x}}_c \end{bmatrix} \quad (4.26)$$

Estableciendo las siguientes equivalencias:

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} \tilde{\boldsymbol{x}}_p \\ \tilde{\boldsymbol{x}}_c \end{bmatrix}; \quad \boldsymbol{u} = \tilde{\boldsymbol{u}}_p\,; \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_p & \boldsymbol{0} \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{C}_p & \boldsymbol{A}_c \end{bmatrix}; \\ \boldsymbol{B} &= \begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{D}_p \end{bmatrix}; \quad \boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_p & \boldsymbol{L} \end{bmatrix} \end{aligned} \quad (4.27)$$

se obtiene el problema del regulador cuadrático lineal visto en el apartado 4.2.1. El estado que se intentará mantener próximo a cero en este caso es el calculado como la

diferencia entre los estados instantáneos y su valores en el estacionario.

A partir del sistema de la ecuación 4.25, se obtendrán las matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ del sistema de control representado en la figura 4.2 .

4.3. Diseño de un controlador LQ para la estabilización y autopiloto de un avión mediante algoritmos evolutivos.

En este apartado se va a diseñar un controlador LQ para el modelo de avión RCAM visto en el apartado 3.2. El sistema de control se corresponde con el mostrado en la figura 4.2, donde el compensador consistirá simplemente en un integrador para cada señal de error. Dicho sistema será aplicable tanto al modo longitudinal como al lateral, con los que se trabajará de forma separada.

Por medio del algoritmo presentado en el capítulo 3 se determinarán las matrices de peso $\boldsymbol{Q}$ y $\boldsymbol{R}$, que deberán codificarse convenientemente en forma de cromosomas: $\boldsymbol{Q}$ y $\boldsymbol{R}$ serán matrices diagonales y los elementos de sus diagonales serán los genes de los cromosomas. A partir de dichas matrices de peso y de las matrices $\boldsymbol{A}$ y $\boldsymbol{B}$ de la ecuación 4.27, se obtendrá la matriz $\boldsymbol{K}$, por medio de la función `lqr` de Matlab. Una vez obtenida $\mathbf{K}$ se descompondrá finalmente en las matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$, según se indica también en la ecuación 4.27.

Tras la obtención de $\boldsymbol{K}_p$ y $\boldsymbol{L}$ se construirá el modelo global de la figura 4.2, con el fin de llevar a cabo la simulación y poder determinar la aptitud del controlador. Se verá dicho modelo en el apartado siguiente.

En lo que sigue se tratarán las particularidades del algoritmo que se va a emplear, aplicado a este caso concreto. Habrá que asignar valores a los distintos parámetros del algoritmo evolutivo. Se estudiarán además las particularidades de la función de evaluación, que es donde más diferencias pueden encontrase a la hora de aplicar el algoritmo a un problema u otro.

4.3.1. Modelo global.

Como se vio en el capítulo 3, la función de evaluación precisará realizar la simulación del sistema global de la figura 4.2, cuya entrada es la señal de referencia y su salida

coincide con la salida de la planta. Para ello son necesarias las ecuaciones de dicho sistema global, que se obtendrán sustituyendo 4.21 en 4.18 y 4.22, resultando

$$\begin{aligned} \begin{bmatrix} \dot{\boldsymbol{x}}_p \\ \dot{\boldsymbol{x}}_c \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A}_p - \boldsymbol{B}_p\boldsymbol{K}_p & -\boldsymbol{B}_p\boldsymbol{L} \\ \boldsymbol{B}_c\boldsymbol{K}_r\left(\boldsymbol{C}_p - \boldsymbol{D}_p\boldsymbol{K}_p\right) & \boldsymbol{A}_c - \boldsymbol{B}_c\boldsymbol{K}_r\boldsymbol{D}_p\boldsymbol{L} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{B}_c \end{bmatrix} \cdot \boldsymbol{r} \\ \boldsymbol{y}_p &= \begin{bmatrix} \boldsymbol{C}_p - \boldsymbol{D}_p\boldsymbol{K}_p & -\boldsymbol{D}_p\boldsymbol{L} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \end{bmatrix} \end{aligned} \tag{4.28}$$

El modelo de la ecuación 4.28 es válido tanto para el modo longitudinal como para el lateral. Únicamente habrá que sustituir las matrices $\boldsymbol{A}_p$, $\boldsymbol{B}_p$, $\boldsymbol{C}_p$ y $\boldsymbol{D}_p$, por las del modelo del modo que se esté tratando. Dichos modelos pueden verse en el apéndice A.

4.3.2. Parámetros del algoritmo.

El algoritmo evolutivo tiene una serie de parámetros a los que hay que asignarles un valor, como por ejemplo las probabilidades de utilización de los distintos operadores o el tamaño de la población. Los valores empleados se recogen en la tabla 4.1.

Parámetro	**Valor**
Tamaño de la población	100
Número de genes por cromosoma (modo longitudinal)	10
Número de genes por cromosoma (modo lateral)	11
Probabilidad de cruce en un punto	0.2
Probabilidad de cruce uniforme	0.2
Probabilidad de cruce aritmético	0.2
Probabilidad de cruce heurístico	0.2
Probabilidad de mutación aleatoria	0.1
Probabilidad de mutación no uniforme	0.1
Número de competidores en concurso de selección	2
Porcentaje de descendientes generados a partir de padres	85 %
Máxima parte real permitida para los autovalores	-0.35
Exponente para mutación no uniforme	3
Valor mínimo para los genes	1
Valor máximo para los genes	10^{10}
Número de generaciones estimadas para la convergencia	1200

Tabla 4.1: Valores de los parámetros del algoritmo evolutivo.

Se observa que en el modo longitudinal, los cromosomas tendrán 1 gen más que en

el modo lateral; esto es debido a que sus ecuaciones en el espacio de estados tienen 1 estado más, por lo que las dimensiones de la matriz cuadrada $\boldsymbol{Q}$ también deberán incrementarse en una unidad para que sean congruentes con el producto $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$.

Siguiendo la norma general, se asignó una probabilidad significativamente mayor a los operadores de cruce, en sus distintas variantes, que a los operadores de mutación: la probabilidad de que tuviera lugar un cruce era de 0.8, frente al 0.2 de la probabilidad de que tuviera lugar una mutación.

Como tamaño de la población y número de competidores en los concursos de selección de padres, se tomaron los valores bastante habituales de 100 y 2, respectivamente, obteniéndose con ellos buenos resultados.

De los cromosomas de las nuevas generaciones, únicamente el 85 % se obtiene mediante la aplicación de operadores genéticos sobre los cromosomas de la generación anterior. La población se completará con un 15 % de inmigrantes con objeto de incrementar la diversidad y evitar la convergencia prematura.

Como se comentó en el apartado 3.4.3.1, se decidió restringir la parte real de los autovalores a valores menores que -0.35, con objeto de evitar las oscilaciones no deseadas que se obtenían cuando dichos autovalores se encontraban más próximos al origen de coordenadas.

Se decidió en una primera aproximación, tomar los valores de los genes del intervalo $[1, 10^{10}]$. Los buenos resultados obtenidos con estos valores, llevaron a no considerar otros.

Por último, el valor del exponente para mutación no uniforme, y el número de generaciones estimadas para la convergencia, se corresponden con los parámetros b y T de la ecuación 1.1. El valor de 3 para b es un valor típico y el valor de 1200 para T se determinó tras unas cuantas de ejecuciones de prueba del algoritmo, empleando el método interactivo para las condiciones de finalización.

4.3.3. La función de evaluación.

Como ya se ha comentado, la función de evaluación será la parte del algoritmo donde tengan lugar los cambios más significativos a la hora de aplicarlo a un problema u otro. En este apartado se tratarán los aspectos particulares de la función de evaluación correspondientes al problema del diseño de un controlador LQ para el modelo RCAM.

Con objeto de simplificar la exposición, se particularizará al caso del modo longitudinal, si bien todo lo que se comente será aplicable, con modificaciones mínimas, al modo lateral.

La función de evaluación recibirá como argumento un cromosoma que estará formado por los elementos de las diagonales de las matrices de peso $\boldsymbol{Q}$ y $\boldsymbol{R}$. También dispondrá de las matrices $\boldsymbol{A}$ y $\boldsymbol{B}$ de la ecuación 4.27 que se denominarán aquí $\boldsymbol{A_{lq}}$ y $\boldsymbol{B_{lq}}$ con objeto de diferenciarlas de otras posibles matrices $\boldsymbol{A}$ y $\boldsymbol{B}$.

Sea el cromosoma:

$$C = \{c_1,\ c_2,\ c_3,\ c_4,\ c_5,\ c_6,\ c_7,\ c_8,\ c_9,\ c_{10}\}$$

Las matrices $\boldsymbol{Q}$ y $\boldsymbol{R}$ se construirán de la siguiente forma

$$\boldsymbol{Q} = \begin{bmatrix} c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_8 \end{bmatrix} \quad \text{y} \quad \boldsymbol{R} = \begin{bmatrix} c_9 & 0 \\ 0 & c_{10} \end{bmatrix}$$

Una vez que se dispone de $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{A}_{lq}$ y $\boldsymbol{B}_{lq}$, se obtendrá la matriz $\boldsymbol{K}$ mediante la siguiente llamada a la función `lqr` de Matlab

```
K = lqr(Alq, Blq, Q, R);
```

La matriz $\boldsymbol{K}$ obtenida será de la forma

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} & k_{18} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} & k_{28} \end{bmatrix}$$

De donde se extraerán finalmente las matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ de la siguiente forma

$$\boldsymbol{K}_p = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} \end{bmatrix} \quad \text{y} \quad \boldsymbol{L} = \begin{bmatrix} k_{17} & k_{18} \\ k_{27} & k_{28} \end{bmatrix}$$

Obtenidas las matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$, se podrá construir el modelo global de las ecuaciones 4.28 y a partir de él, llevar a cabo la simulación y determinación de los índices de aptitud correspondientes a los distintos objetivos. Quedará por último el tratamiento de la robustez, que por requerir algo más de detenimiento se verá en el apartado siguiente.

4.3.3.1. Robustez.

La función de evaluación del algoritmo evolutivo calculará los márgenes de ganancia y márgenes de fase para la función de sensibilidad compensada $S + T$ a la entrada y a la salida. El cálculo se llevará a cabo según lo explicado en el apartado 3.4.3.1, para lo que previamente habrá que obtener la función de sensibilidad $S + T$, tanto a la entrada como a la salida.

A partir de las figuras 3.13 y 4.2 se obtiene el esquema de la figura 4.3. Las ecuaciones de los bloques G y H se deducen a continuación.

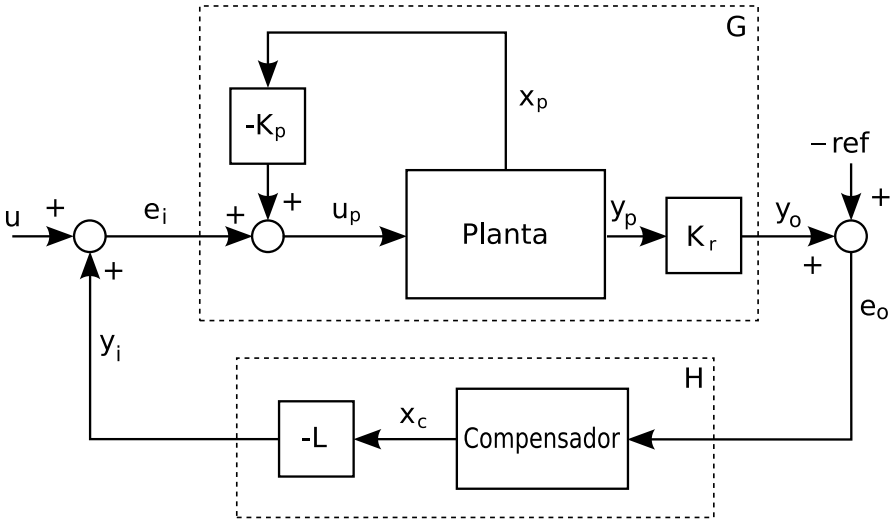


Figura 4.3: Esquema para la determinación de las funciones de sensibilidad.

Según la figura 4.3 el vector de entradas a la planta será

$$\boldsymbol{u}_p = -\boldsymbol{K}_p\boldsymbol{x}_p + \boldsymbol{e}_i \tag{4.29}$$

Sustituyendo 4.29 en las ecuaciones 4.18 de la planta, se obtiene

$$\begin{aligned}\dot{\boldsymbol{x}}_p &= \boldsymbol{A}_p\boldsymbol{x}_p + \boldsymbol{B}_p\left(-\boldsymbol{K}_p\boldsymbol{x}_p + \boldsymbol{e}_i\right) = \left(\boldsymbol{A}_p - \boldsymbol{B}_p\boldsymbol{K}_p\right)\boldsymbol{x}_p + \boldsymbol{B}_p\boldsymbol{e}_i \\ \boldsymbol{y}_p &= \boldsymbol{C}_p\boldsymbol{x}_p + \boldsymbol{D}_p\left(-\boldsymbol{K}_p\boldsymbol{x}_p + \boldsymbol{e}_i\right) = \left(\boldsymbol{C}_p - \boldsymbol{D}_p\boldsymbol{K}_p\right)\boldsymbol{x}_p + \boldsymbol{D}_p\boldsymbol{e}_i\end{aligned}$$

Y como $\boldsymbol{y}_o = \boldsymbol{K}_r\boldsymbol{y}_p$, las ecuaciones del bloque G de la figura 4.3 serán finalmente

$$\left.\begin{aligned}\dot{\boldsymbol{x}}_p &= \left(\boldsymbol{A}_p - \boldsymbol{B}_p\boldsymbol{K}_p\right)\boldsymbol{x}_p + \boldsymbol{B}_p\boldsymbol{e}_i \\ \boldsymbol{y}_o &= \boldsymbol{K}_r\left(\boldsymbol{C}_p - \boldsymbol{D}_p\boldsymbol{K}_p\right)\boldsymbol{x}_p + \boldsymbol{K}_r\boldsymbol{D}_p\boldsymbol{e}_i\end{aligned}\right\} \tag{4.30}$$

A su vez, las ecuaciones del bloque H son

$$\left.\begin{aligned}\dot{\boldsymbol{x}}_c &= \boldsymbol{A}_c\boldsymbol{x}_c + \boldsymbol{B}_c\boldsymbol{e}_o \\ \boldsymbol{y}_i &= -\boldsymbol{L}\boldsymbol{x}_c\end{aligned}\right\} \tag{4.31}$$

Para el cálculo de la función de sensibilidad $S+T$ se utilizará el *producto en estrella de Redheffer* (ver apéndice D).

La función de sensibilidad compensada $S+T$ a la entrada, será la función de transferencia de u a $e+y$, y puede obtenerse a partir del esquema de la figura 4.4.

El código de Matlab empleado se muestra a continuación. Las Matrices `Along`, `Blong`, `Clong` y `Dlong` son las matrices del modelo en el espacio de estados, del modo longitudinal del avión. Por su parte, las matrices `Ai` y `Bi` corresponden al modelo de los integradores que, como ya se ha comentado, componen el compensador:

```
% Se obtienen los bloques G y H:
G = pck(Along-Blong*Kp, Blong, Kr*(Clong-Dlong*Kp), Kr*Dlong);
H = pck(Ai, Bi, -L, zeros(size(L, 1), size(Bi, 2)));
% Número de entradas a los integradores del compensador:
ni = size(Kr, 1);
```

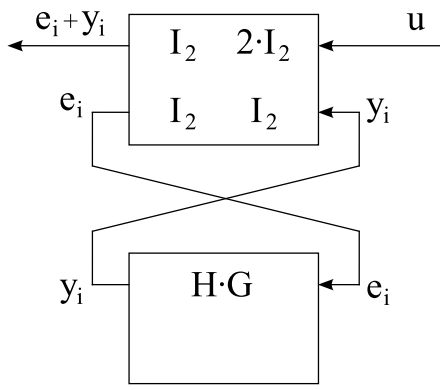


Figura 4.4: Esquema para el cálculo de la función de sensibilidad compensada S+T a la entrada.

```
% Se obtienen las matrices de los bloques de la figura 4.4:
TOP = [eye(ni), 2*eye(ni); eye(ni), eye(ni)];
BOT = mmult(H, G);
% Se obtiene el sistema correspondiente a S+T:
sgsti = starp(TOP, BOT, ni, ni);
% Se obtienen las matrices del sistema:
[As, Bs, Cs, Ds] = unpck(sgsti);
% Se obtienen los valores singulares:
singvali = sigma(As, Bs, Cs, Ds);
% Valores singulares máximos:
maxsingvali = max(singvali');
% Se calculan márgenes de ganancia y fase:
km = 1/max(maxsingvali);
MGi(1) = 20*log10((1-km)/(1+km));
MGi(2) = 20*log10((1+km)/(1-km));
MFi(1) = -2*atan(km)*180/pi;
MFi(2) = 2*atan(km)*180/pi;
```

De igual forma se procederá con la función de sensibilidad compensada $S + T$ a la salida. En este caso, el esquema correspondiente se muestra en la figura 4.5.

El código de Matlab para el cálculo del margen de ganancia y el margen de fase a partir de la función de sensibilidad $S + T$, se muestra a continuación:

```
% Se obtienen las matrices de los bloques de la figura 4.5:
TOP = [eye(ni), 2*eye(ni); eye(ni), eye(ni)];
```

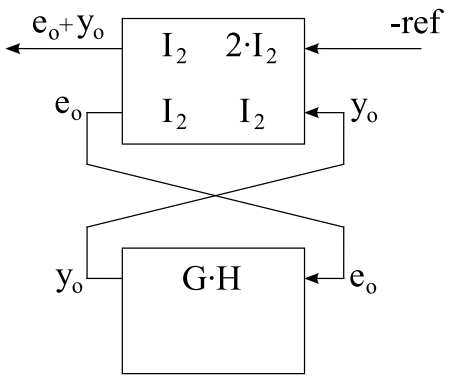


Figura 4.5: Esquema para el cálculo de la función de sensibilidad compensada S+T a la salida.

```
BOT = mmult(G, H);
% Se obtiene el sistema correspondiente a S+T:
sgsto = starp(TOP, BOT, ni, ni);
% Se obtienen las matrices del sistema:
[As, Bs, Cs, Ds] = unpck(sgsto);
% Se obtienen los valores singulares:
singvalo = sigma(As, Bs, Cs, Ds);
% Valores singulares máximos:
maxsingvalo = max(singvalo');
% Se calculan márgenes de ganancia y fase:
km = 1/max(maxsingvalo);
MGo(1) = 20*log10((1-km)/(1+km));
MGo(2) = 20*log10((1+km)/(1-km));
MFo(1) = -2*atan(km)*180/pi;
MFo(2) = 2*atan(km)*180/pi;
```

Finalmente, la función de evaluación devolverá vectores de aptitud de 23 elementos para el caso del modo longitudinal y de 14 elementos para el modo lateral. Dichos valores son los que se especificaron en el apartado 3.4.3.1.

4.3.4. Resultados.

El algoritmo fue ejecutado en 3 ocasiones para el modo longitudinal y otras 3 para el modo lateral. Los resultados se muestran a continuación ordenados de mejor a peor aptitud.

4.3.4.1. Modo Longitudinal.

Experimento 1.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.0182577
- Elementos de la diagonal principal de $\boldsymbol{Q}$:

  $(4.77311 \cdot 10^7,\ 10^{10},\ 3.14782 \cdot 10^7,\ 2.50429 \cdot 10^7,\ 8.49307 \cdot 10^7,\ 10^{10},\ 4.88292 \cdot 10^6,\ 907615)$
- Elementos de la diagonal principal de $\boldsymbol{R}$: $(2.45321 \cdot 10^9,\ 4.37834 \cdot 10^9)$
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -2.98368 & -11.0841 & 0.00517751 & 0.0376092 & 0.772437 & -0.365414 \\ 0.119603 & 1.49324 & 0.106106 & -0.012954 & -0.0204744 & 2.11039 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} 0.0420835 & 0.00638598 \\ -0.0110874 & 0.0135811 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.145602	OK
Máximo tiempo de subida de Wv (<5 seg):	4.8	OK
Error medio de Wv al final (<0.042 m/s):	$1.66821 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.159674	OK
Error medio de VA al final (<0.01 m/s):	0.000920426	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00782056, 0.0221898]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	[-0.0270154, 0]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.0120251	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.172185	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0	OK
Máximo tiempo de subida de VA (<12 seg):	11.7	OK
Error medio de VA al final (<0.13 m/s):	0.0260077	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.250176	OK
Error medio de Wv al final (<0.01 m/s):	0.000489967	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[0, 0.0289091]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	[0, 0.0966095]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.00195153	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.0811129	OK

Medidas de Robustez:		
Margen de ganancia entrada (-10dB >MG >10dB):	[-10.186, 10.186]	OK
Margen de fase entrada (-50º >MF >50º):	[-55.6024, 55.6024]	OK
Margen de ganancia salida (-10dB >MG >10dB):	[-10.1929, 10.1929]	OK
Margen de fase salida (-50º >MF >50º):	[-55.6282, 55.6282]	OK

Tabla 4.2: Modo longitudinal. Resultados del experimento 1.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

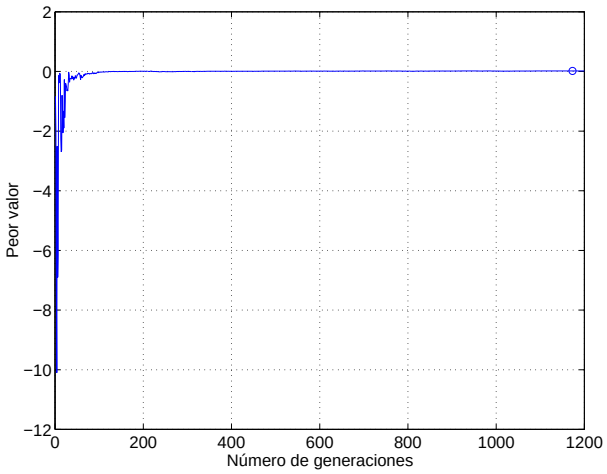
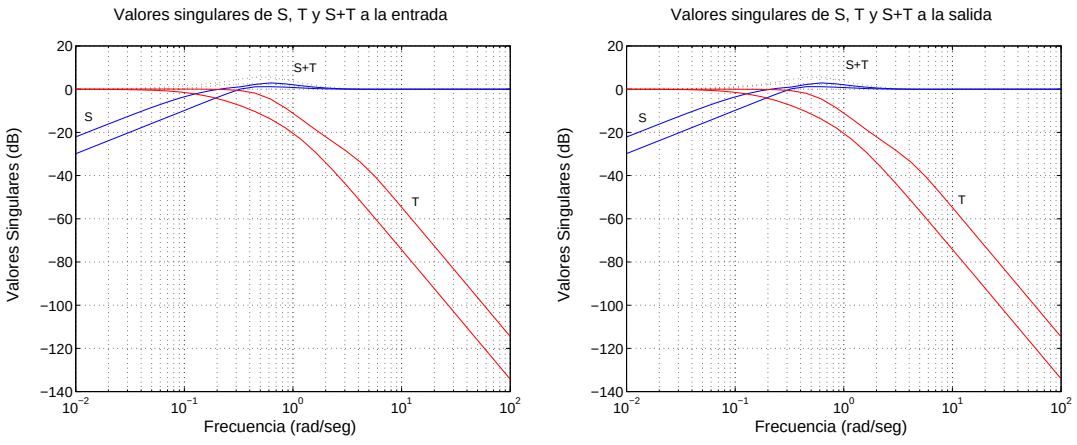


Figura 4.6: Evolución del peor de los índices de aptitud.

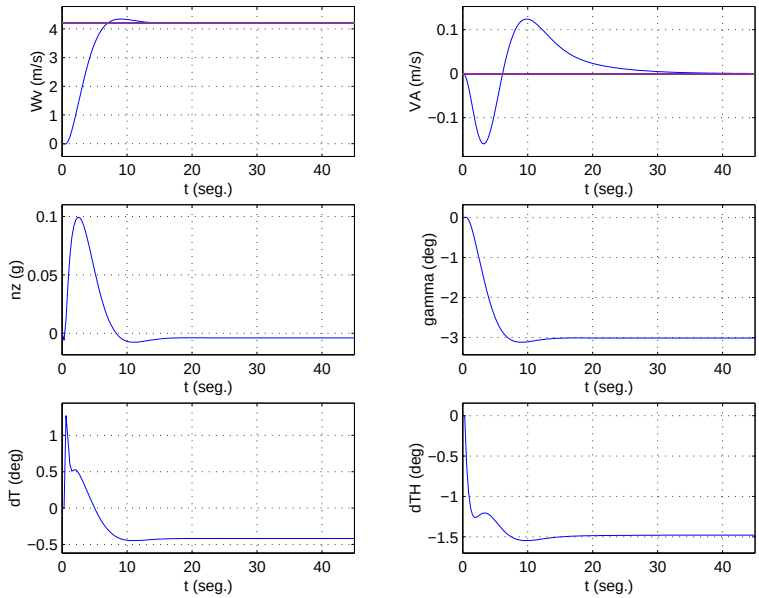
- Gráficas de los valores singulares de las funciones de sensibilidad:



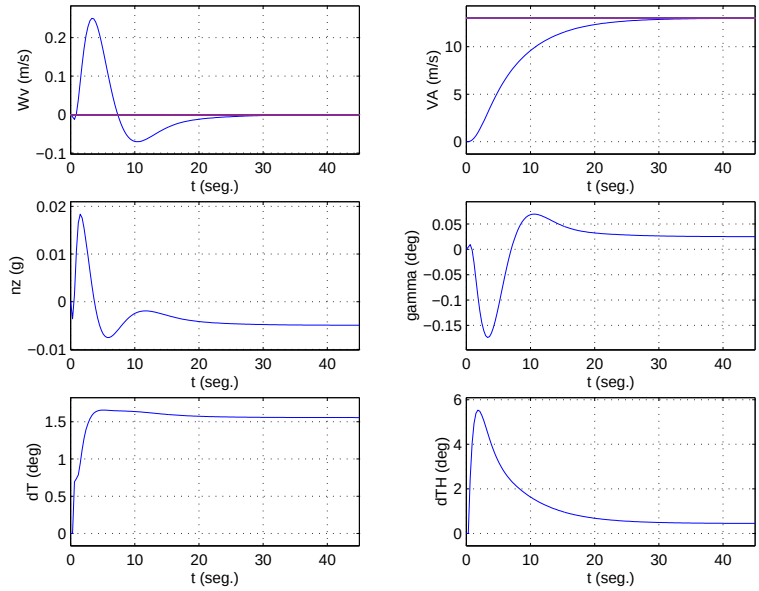
(a) (b)

Figura 4.7: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en w_V y V_A:



(a)



(b)

Figura 4.8: (a) Respuesta ante una señal de tipo escalón en w_V, (b) Respuesta ante una señal de tipo escalón en V_A.

Experimento 2.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.0111124
- Elementos de la diagonal principal de $\boldsymbol{Q}$:

  $(1,\ 10^{10},\ 4.5825 \cdot 10^{7},\ 3.78974 \cdot 10^{7},\ 1.262 \cdot 10^{7},\ 10^{10},\ 7.22417 \cdot 10^{6},\ 1.31362 \cdot 10^{6})$
- Elementos de la diagonal principal de $\boldsymbol{R}$: $(4.13905 \cdot 10^{9},\ 7.1605 \cdot 10^{9})$
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -2.83417 & -10.339 & 0.00441431 & 0.0350532 & 0.733664 & -0.367443 \\ 0.119695 & 1.45397 & 0.100025 & -0.0126134 & -0.0212397 & 1.90278 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} 0.0393995 & 0.00592473 \\ -0.0105635 & 0.0127735 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.153909	OK
Máximo tiempo de subida de Wv (<5 seg):	4.8	OK
Error medio de Wv al final (<0.042 m/s):	$1.53335 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.161251	OK
Error medio de VA al final (<0.01 m/s):	0.000961768	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00785647, 0.021616]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	[-0.0271056, 0]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.0120425	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.161411	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0	OK
Máximo tiempo de subida de VA (<12 seg):	11.7	OK
Error medio de VA al final (<0.13 m/s):	0.0268105	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.247666	OK
Error medio de Wv al final (<0.01 m/s):	0.000447211	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[0, 0.0290448]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	[0, 0.0972734]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.00194365	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.0753552	OK

Medidas de Robustez:		
Margen de ganancia entrada (-10dB MG >10dB):	[-10.1127, 10.1127]	OK
Margen de fase entrada (-50º >MF >50º):	[-55.3282, 55.3282]	OK
Margen de ganancia salida (-10dB >MG >10dB):	[-10.1124, 10.1124]	OK
Margen de fase salida (-50º >MF >50º):	[-55.3271, 55.3271]	OK

Tabla 4.3: Modo longitudinal. Resultados del experimento 2.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

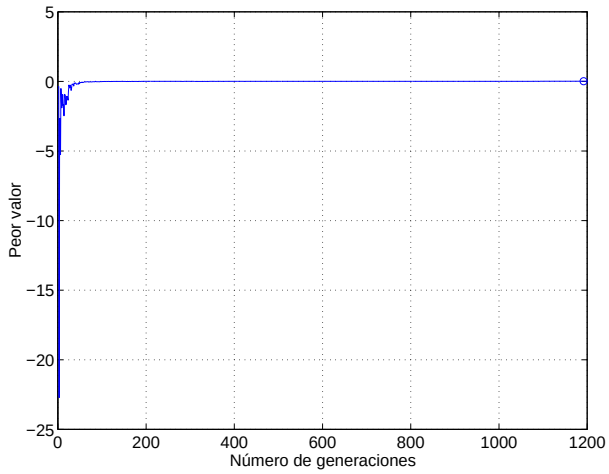
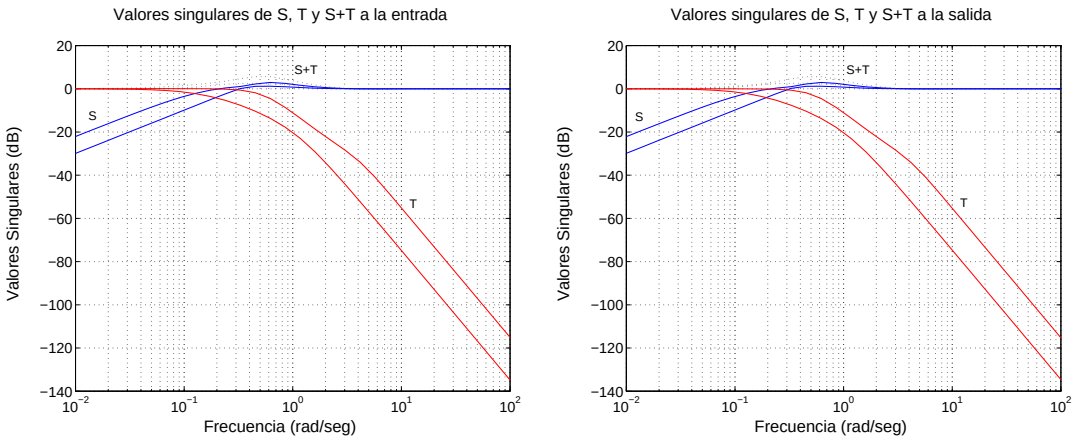


Figura 4.9: Evolución del peor de los índices de aptitud.

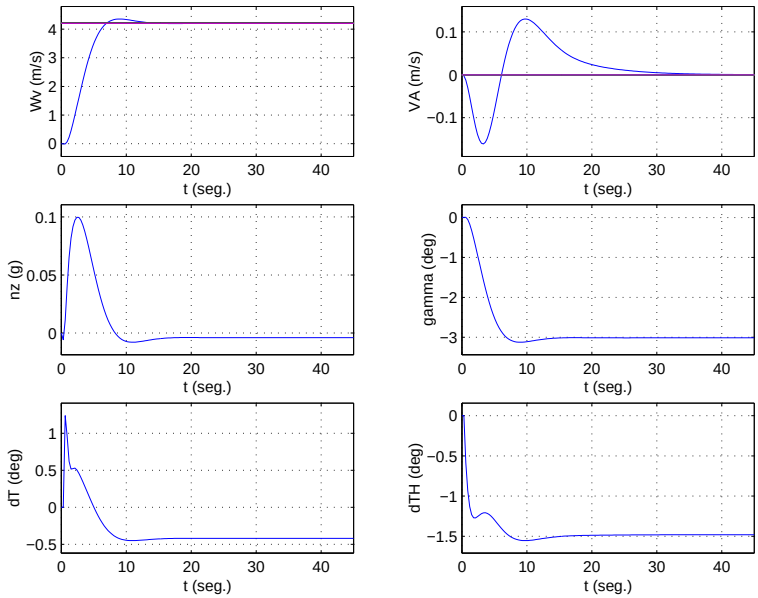
- Gráficas de los valores singulares de las funciones de sensibilidad:



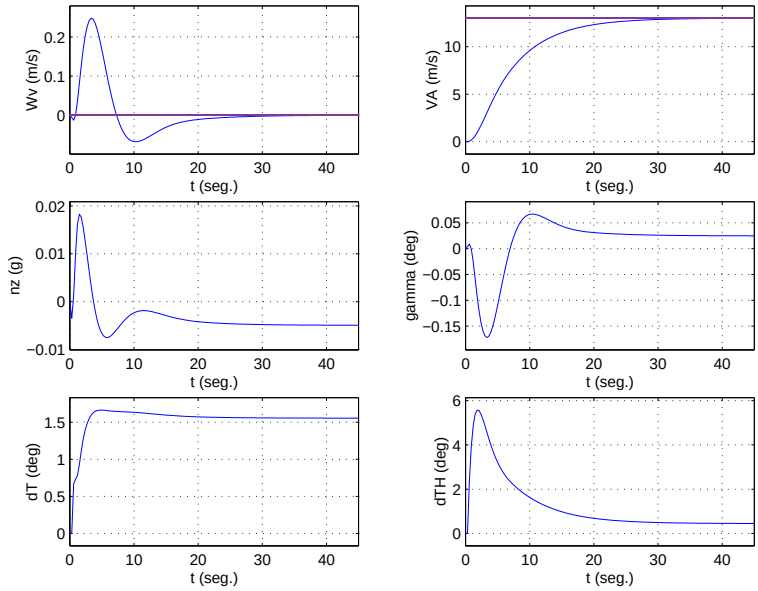
(a) (b)

Figura 4.10: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en w_V y V_A:



(a)



(b)

Figura 4.11: (a) Respuesta ante una señal de tipo escalón en w_V, (b) Respuesta ante una señal de tipo escalón en V_A.

Experimento 3.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.0110259
- Elementos de la diagonal principal de $\boldsymbol{Q}$:

  $(2.80297 \cdot 10^7,\ 9.92418 \cdot 10^9,\ 4.15209 \cdot 10^7,\ 3.9403 \cdot 10^7,\ 2.06589 \cdot 10^7,$
  $9.98661 \cdot 10^9,\ 7.27782 \cdot 10^6,\ 1.20955 \cdot 10^6)$
- Elementos de la diagonal principal de $\boldsymbol{R}$: $(3.77192 \cdot 10^9,\ 6.40195 \cdot 10^9)$
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -2.96073 & -10.8511 & 0.00297537 & 0.0352176 & 0.760675 & -0.393232 \\ 0.134287 & 1.65442 & 0.101469 & -0.0145366 & -0.0231686 & 1.94759 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} 0.0413403 & 0.00605291 \\ -0.0113967 & 0.0129363 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.155019	OK
Máximo tiempo de subida de Wv (<5 seg):	4.8	OK
Error medio de Wv al final (<0.042 m/s):	$1.66167 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.182019	OK
Error medio de VA al final (<0.01 m/s):	0.00103655	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00784925, 0.0219097]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	[-0.0271604, 0]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.011963	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.169202	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0	OK
Máximo tiempo de subida de VA (<12 seg):	11.7	OK
Error medio de VA al final (<0.13 m/s):	0.0265886	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.238196	OK
Error medio de Wv al final (<0.01 m/s):	0.000445789	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[0, 0.0290895]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	[0, 0.0973397]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.00185724	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.0769283	OK

Medidas de Robustez:		
Margen de ganancia entrada (-10dB >MG >10dB):	[-10.1196, 10.1196]	OK
Margen de fase entrada (-50º >MF >50º):	[-55.3543, 55.3543]	OK
Margen de ganancia salida (-10dB >MG >10dB):	[-10.1115, 10.1115]	OK
Margen de fase salida (-50º >MF >50º):	[-55.3238, 55.3238]	OK

Tabla 4.4: Modo longitudinal. Resultados del experimento 3.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

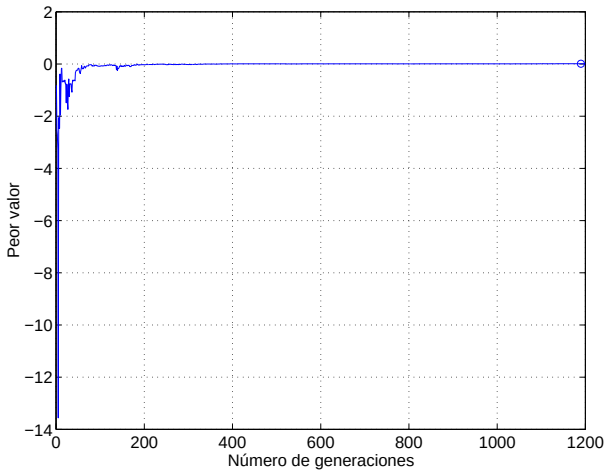
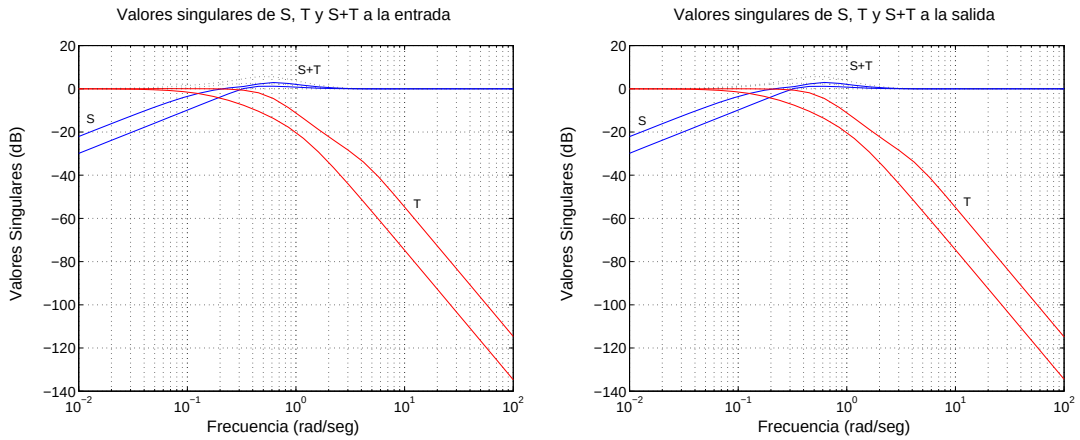


Figura 4.12: Evolución del peor de los índices de aptitud.

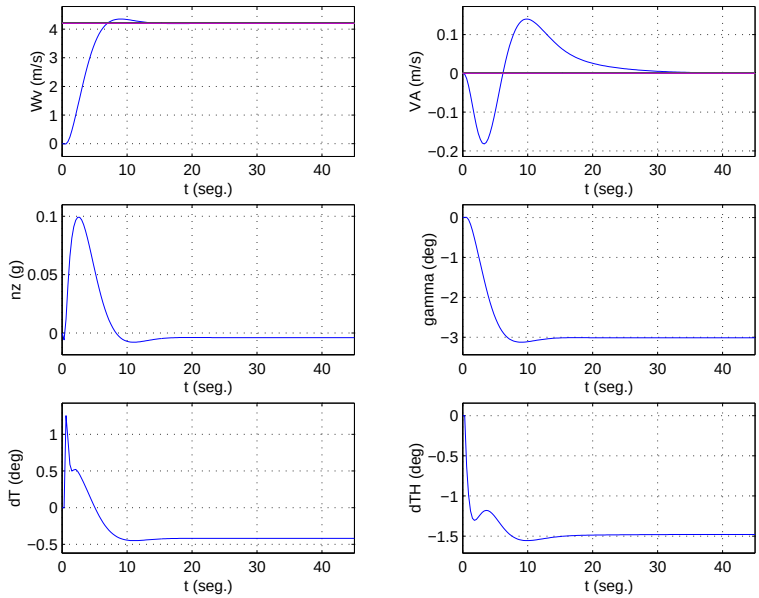
- Gráficas de los valores singulares de las funciones de sensibilidad:



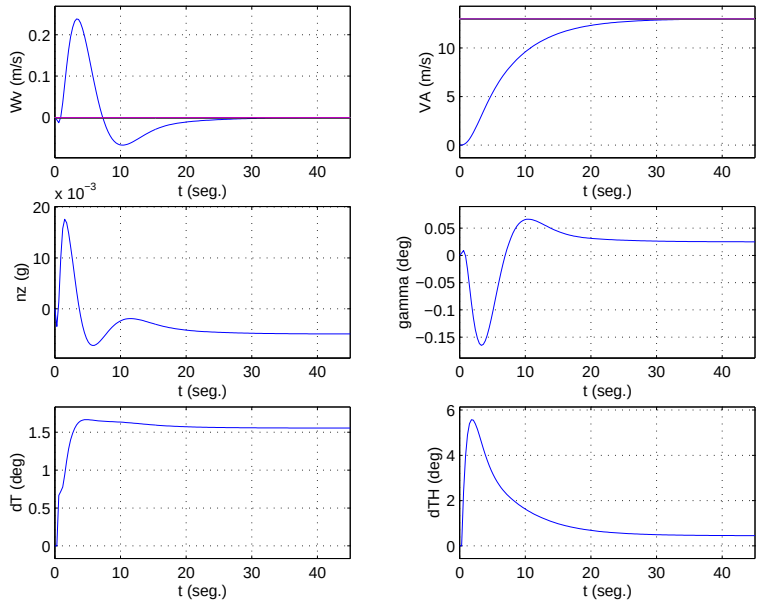
(a) (b)

Figura 4.13: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en w_V y V_A:



(a)



(b)

Figura 4.14: (a) Respuesta ante una señal de tipo escalón en w_V, (b) Respuesta ante una señal de tipo escalón en V_A.

4.3.4.2. Modo Lateral.

Experimento 1.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.04
- Elementos de la diagonal principal de $\boldsymbol{Q}$: (176125, $3.86243 \cdot 10^8$, 475018, 10^{10}, $3.52478 \cdot 10^7$, 14764.6, 1, $5.46433 \cdot 10^9$, $5.75523 \cdot 10^8$)
- Elementos de la diagonal principal de $\boldsymbol{R}$: ($4.00049 \cdot 10^8$, $1.2663 \cdot 10^8$)
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -1.321 & -1.6441 & -2.0341 & -8.4562 & -0.08902 & 0.1583 & -0.003535 \\ 0.5525 & -19.0198 & 2.1986 & -3.356 & 0.4708 & -0.005584 & 1.3374 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} -0.632105 & -1.18176 \\ 6.47222 & -0.36462 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Chi:		
Máximo sobreimpulso en chi (<0.0174533 rad):	0	OK
Máximo tiempo de subida de chi (<10 seg):	9.6	OK
Error medio de chi al final (<0.00349066 rad):	$8.40678 \cdot 10^{-5}$	OK
Pico máximo en beta debido a chi (<0.0052 rad):	0.00344411	OK
Error medio de beta al final (<0.0001 rad):	$6.90281 \cdot 10^{-5}$	OK
Saturación dA (-0.4363 rad <dA <0.4363 rad):	[-0.244384, 0.113059]	OK
Saturación dR (-0.5236 rad <dR <0.5236 rad):	[-0.0756334, 0.000126287]	OK
Máxima variación de dA (|d(dA)| <0.4363 rad/seg):	0.410345	OK
Máxima variación de dR (|d(dR)| <0.4363 rad/seg):	0.12454	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 dB >MG >10 dB):	[-10.6319, 10.6319]	OK
Margen de fase entrada (-50º >MF >50º):	[-57.2293, 57.2293]	OK
Margen de ganancia salida (-10 dB >MG >10 dB):	[-10.631, 10.631]	OK
Margen de fase salida (-50º >MF >50º):	[-57.226, 57.226]	OK

Tabla 4.5: Modo lateral. Resultados del experimento 1.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

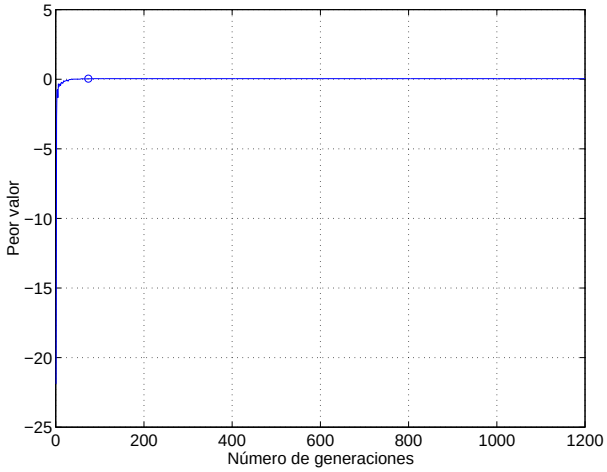
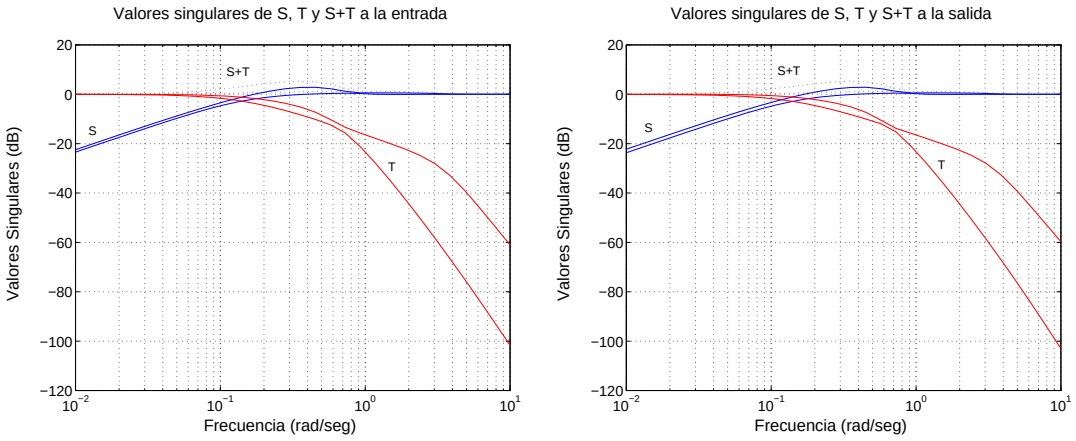


Figura 4.15: Evolución del peor de los índices de aptitud.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 4.16: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en χ:

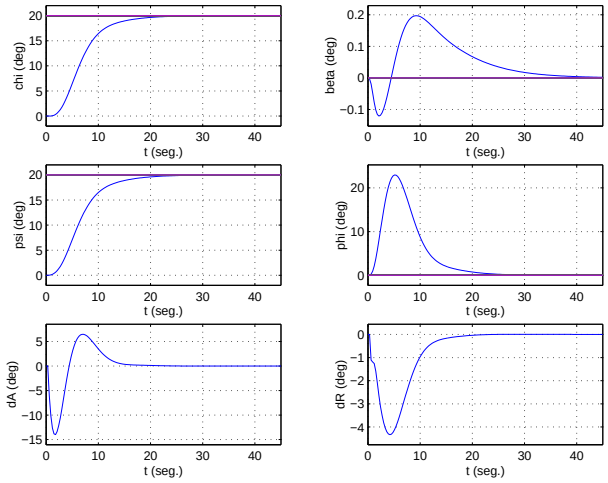


Figura 4.17: Respuesta ante una señal de tipo escalón en χ.

Experimento 2.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.04
- Elementos de la diagonal principal de $\boldsymbol{Q}$:

  $(1.78854 \cdot 10^6,\ 5.38413 \cdot 10^7,\ 4.62677 \cdot 10^6,\ 10^{10},\ 5.49153 \cdot 10^7,\ 171918,\ 3.24503 \cdot 10^7,\ 8.35895 \cdot 10^9,\ 5.84957 \cdot 10^8)$
- Elementos de la diagonal principal de $\boldsymbol{R}$: $(4.09039 \cdot 10^8,\ 1.32008 \cdot 10^8)$
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -1.3286 & -1.5774 & -2.0507 & -8.485 & -0.0909 & 0.1591 & -0.007920 \\ 0.6871 & -21.6186 & 2.6242 & -2.7447 & 0.5898 & -0.01227 & 1.5289 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} -0.649269 & -1.18346 \\ 7.87497 & -0.302338 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Chi:		
Máximo sobreimpulso en chi (<0.0174533 rad):	0	OK
Máximo tiempo de subida de chi (<10 seg):	9.6	OK
Error medio de chi al final (<0.00349066 rad):	$6.90919 \cdot 10^{-5}$	OK
Pico máximo en beta debido a chi (<0.0052 rad):	0.0022309	OK
Error medio de beta al final (<0.0001 rad):	$4.47421 \cdot 10^{-5}$	OK
Saturación dA (-0.4363 rad <dA <0.4363 rad):	[-0.24481, 0.114497]	OK
Saturación dR (-0.5236 rad <dR <0.5236 rad):	[-0.0763943, $4.19404 \cdot 10^{-5}$]	OK
Máxima variación de dA (|d(dA)| <0.4363 rad/seg):	0.410932	OK
Máxima variación de dR (|d(dR)| <0.4363 rad/seg):	0.103043	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 dB >MG >10 dB):	[-10.6329, 10.6329]	OK
Margen de fase entrada (-50º >MF >50º):	[-57.2326, 57.2326]	OK
Margen de ganancia salida (-10 dB >MG >10 dB):	[-10.617, 10.617]	OK
Margen de fase salida (-50º >MF >50º):	[-57.176, 57.176]	OK

Tabla 4.6: Modo lateral. Resultados del experimento 2.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

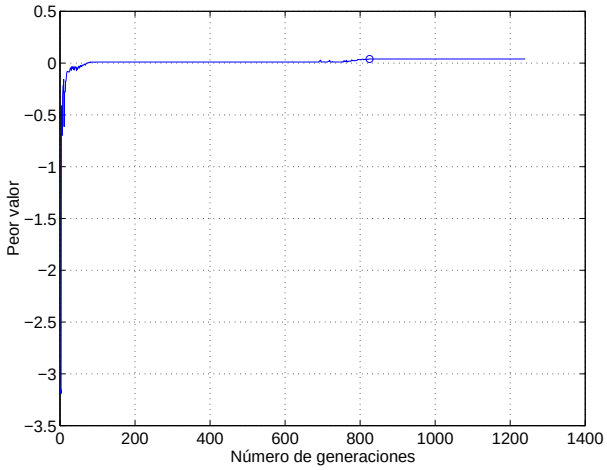
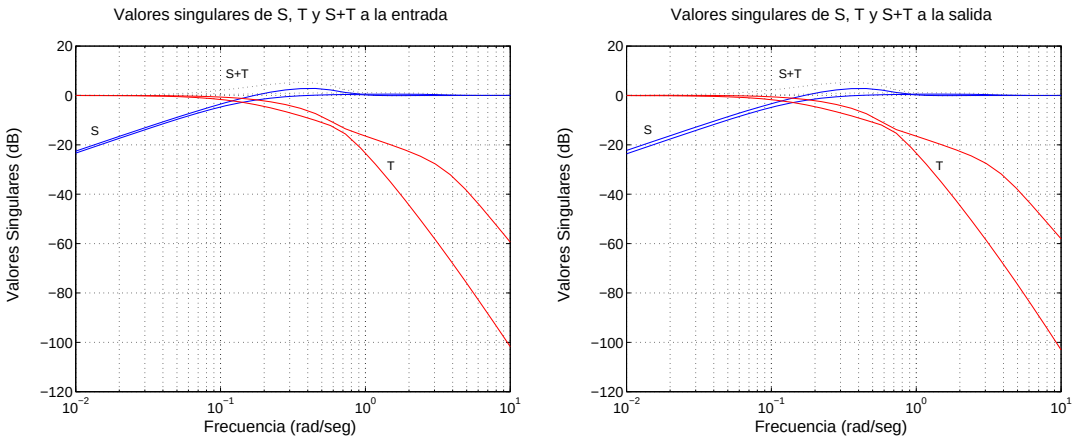


Figura 4.18: Evolución del peor de los índices de aptitud.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 4.19: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en χ:

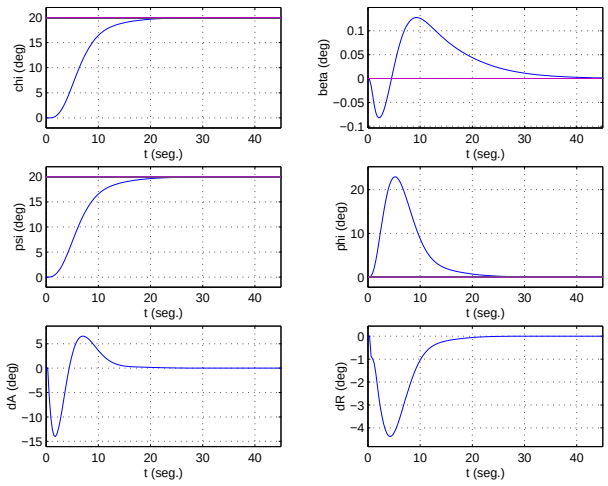


Figura 4.20: Respuesta ante una señal de tipo escalón en χ.

Experimento 3.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.0333558
- Elementos de la diagonal principal de $\boldsymbol{Q}$:

  $(7.15861 \cdot 10^6,\ 2.84257 \cdot 10^8,\ 3.54716 \cdot 10^8,\ 9.94269 \cdot 10^9,\ 4.78576 \cdot 10^7,\ 1,\ 1.63876 \cdot 10^8,\ 9.9476 \cdot 10^9,\ 7.58978 \cdot 10^8)$
- Elementos de la diagonal principal de $\boldsymbol{R}$: $(5.03346 \cdot 10^8,\ 2.58961 \cdot 10^8)$
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{L}$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -1.4164 & -1.5875 & -2.2299 & -8.4941 & -0.08968 & 0.1685 & -0.01204 \\ 0.5539 & -16.9978 & 2.0714 & -2.3047 & 0.3859 & -0.0117 & 1.364 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} -0.713793 & -1.21202 \\ 6.11745 & -0.274881 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Chi:		
Máximo sobreimpulso en chi (<0.0174533 rad):	0	OK
Máximo tiempo de subida de chi (<10 seg):	9.6	OK
Error medio de chi al final (<0.00349066 rad):	$1.77408 \cdot 10^{-5}$	OK
Pico máximo en beta debido a chi (<0.0052 rad):	0.00221967	OK
Error medio de beta al final (<0.0001 rad):	$2.19588 \cdot 10^{-5}$	OK
Saturación dA (-0.4363 rad <dA <0.4363 rad):	[-0.237974, 0.103094]	OK
Saturación dR (-0.5236 rad <dR <0.5236 rad):	[-0.0749055, $8.0372 \cdot 10^{-5}$]	OK
Máxima variación de dA (|d(dA)| <0.4363 rad/seg):	0.420722	OK
Máxima variación de dR (|d(dR)| <0.4363 rad/seg):	0.0939583	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 dB >MG >10 dB):	[-10.3543, 10.3543]	OK
Margen de fase entrada (-50º >MF >50º):	[-56.2247, 56.2247]	OK
Margen de ganancia salida (-10 dB >MG >10 dB):	[-10.3451, 10.3451]	OK
Margen de fase salida (-50º >MF >50º):	[-56.1908, 56.1908]	OK

Tabla 4.7: Modo lateral. Resultados del experimento 3.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

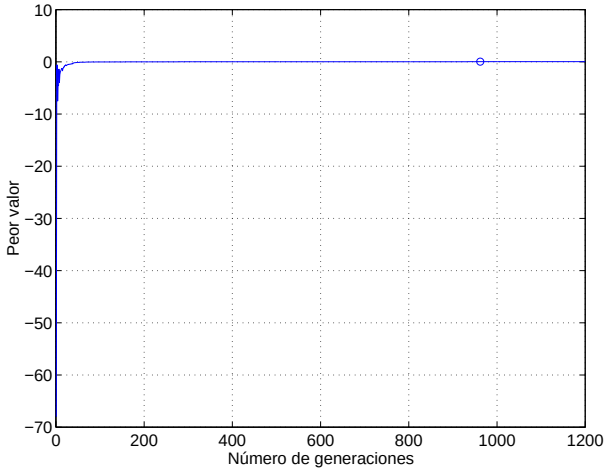
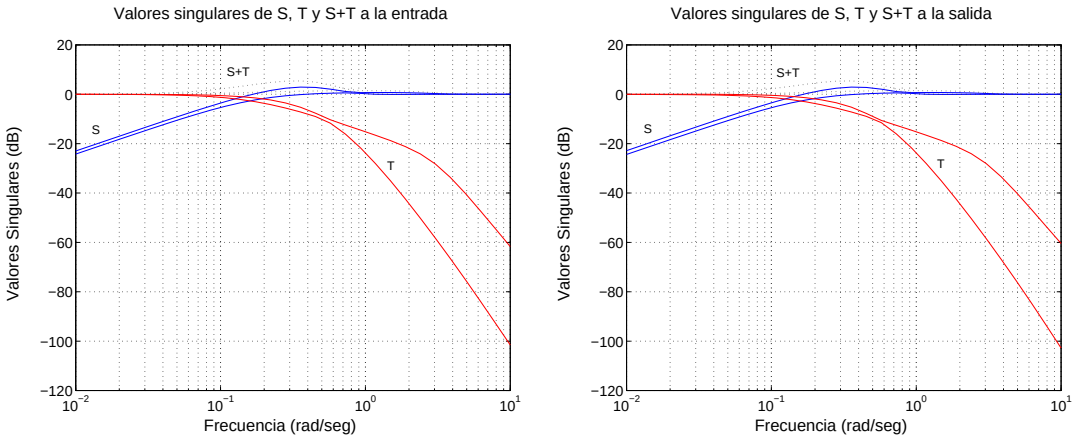


Figura 4.21: Evolución del peor de los índices de aptitud.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 4.22: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en χ:

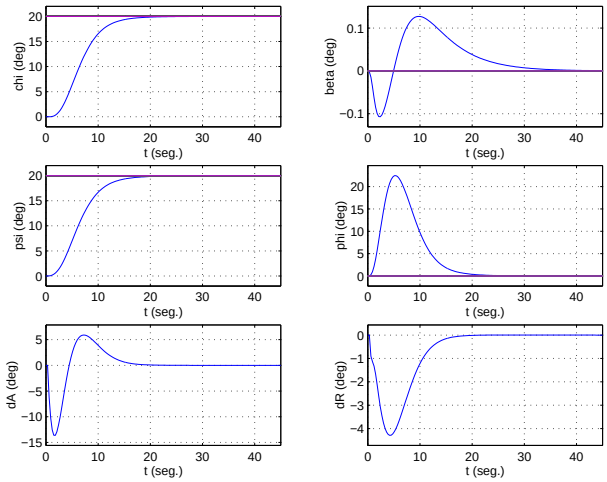


Figura 4.23: Respuesta ante una señal de tipo escalón en χ.

4.3.5. Comentario final.

Se ha comprobado como el algoritmo ha sido capaz de sintonizar múltiples veces una estructura de control preestablecida, apoyándose en el método LQ. Sin embargo a la hora de incluir el controlador diseñado en el sistema propuesto por *GARTEUR* surge una dificultad, por no estar disponibles los estados y estar el controlador basado en la realimentación de dichos estados. Para salvar esta dificultad habría que realizar un controlador más complejo, añadiéndole un estimador de estados. Sin embargo, se optó por otro camino, diseñando un controlador con realimentación de las salidas y obteniendo las ganancias del controlador directamente, sin la ayuda de ningún método de sintonía adicional. Esto es lo que se verá en el capítulo siguiente.

4.4. Conclusiones.

Mediante el algoritmo evolutivo diseñado en el capítulo 3, se ha realizado con éxito la sintonía de un controlador, basándose en el método LQ. El procedimiento empleado aporta las siguientes ventajas:

- Se trabaja desde un punto de vista más próximo al usuario, tratando directamente con las especificaciones de diseño, y siendo posible hacerlo en el dominio temporal, utilizando variables como *máximo sobreimpulso*, *tiempo de subida*, *error en el estacionario*, etc.
- Se resuelve el problema existente sobre la elección de las matrices de peso $\boldsymbol{Q}$ y $\boldsymbol{R}$. No serán necesarios los procedimientos iterativos de prueba y error por parte del usuario, pues será el algoritmo el que se encargue de la determinación de dichas matrices.
- Se obtiene un controlador robusto, a partir de las medidas de robustez que se incluyen en la función de evaluación.

Los objetivos considerados fueron satisfechos en su totalidad, tanto en el modo longitudinal como en el lateral.

Capítulo 5

Sintonía directa de un controlador con una estructura predeterminada, para la estabilización y el autopiloto de un avión, mediante algoritmos evolutivos

5.1. Introducción.

En el capítulo 4 se realizó la sintonía de dos estructuras de control, correspondientes a los modos longitudinal y lateral del problema de control de vuelo de un avión comercial. Sin embargo, para llevar a cabo la implementación del controlador completo eran necesarios estados que no estaban disponibles, requiriéndose su obtención mediante un estimador. En el presente capítulo se diseñará un controlador completo basado en la realimentación de las salidas, siendo evaluado posteriormente con las herramientas proporcionadas con el problema.

El punto de partida será una estructura de control predeterminada, en concreto se utilizará la misma que se empleó en [36] y en [81]. El objetivo será comprobar que el algoritmo diseñado en el capítulo 3 es capaz de realizar la sintonía de dicha estructura,

empleando los mismos objetivos que se usaron en el capítulo 4, tanto para el modo longitudinal como para el lateral. Para realizar la sintonía, el algoritmo evolutivo deberá ser capaz de obtener directamente los parámetros del controlador.

La función de evaluación será modificada en lo relativo a los modelos globales del sistema para la simulación. También requerirán modificación los modelos que serán empleados para el cálculo de los márgenes de ganancia y fase. El resto de la función de evaluación será idéntica a la empleada en el capítulo 4.

Como se verá en los siguientes apartados, el controlador estará compuesto por un lazo interno y otro externo, aplicándose el algoritmo únicamente al cálculo de las ganancias del lazo interno. El lazo externo será un sistema SISO, por lo que su ganancia se obtendrá directamente por el método del lugar de las raíces.

Una vez obtenido el controlador completo, se aplicará al modelo de avión y se evaluará su comportamiento durante una maniobra de aproximación, mediante las herramientas proporcionadas en *RCAM* para llevar a cabo dicha evaluación.

5.2. La estructura del controlador.

En el capítulo 3 se describió el modelo *RCAM*, recogiéndose en la figura 3.2 el diagrama del sistema. En lo que sigue, se describirá la estructura del bloque controlador, nombrado en la figura 3.2 como *ACTUADORES*.

La figura 5.1 representa la estructura del controlador global. Se observa que el controlador se ha dividido en dos bloques desacoplados: un controlador que se encargará del modo longitudinal, y otro que lo hará del modo lateral.

Tanto el controlador longitudinal como el lateral están compuestos por un lazo interno y un lazo externo. Los lazos internos se encargarán de hacer que el avión tenga un vuelo fácil y agradable, mientras que los lazos externos se encargarán de sustituir al piloto en ciertas maniobras de vuelo.

5.2.1. El controlador longitudinal.

La estructura del controlador longitudinal se muestra en la figura 5.2. El valor K del bloque *LonKout* será la ganancia de lazo externo, que como ya se ha comentado se obtendrá por el método del lugar de las raíces.

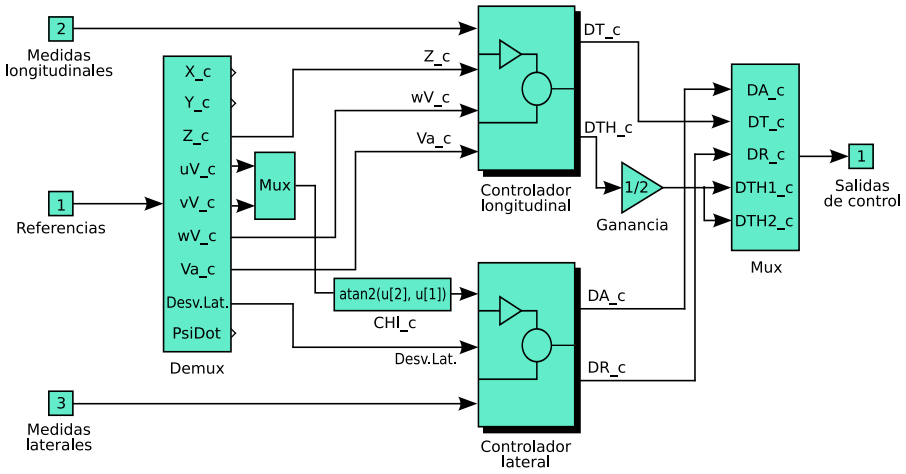


Figura 5.1: Estructura del controlador global.

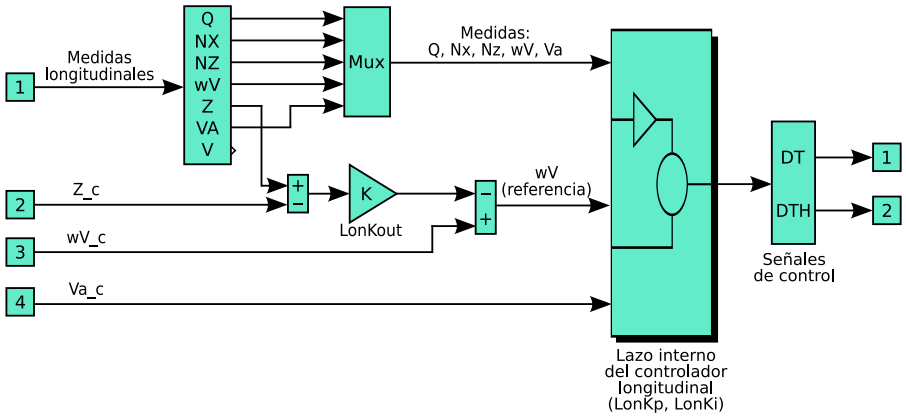


Figura 5.2: Controlador longitudinal.

El modelo linealizado del avión proporciona 7 medidas para el modo longitudinal, de las cuales la velocidad inercial total, V, no se tuvo en cuenta para el diseño del controlador. Las medidas empleadas se muestran en la tabla 5.1.

El controlador longitudinal generará dos señales de control: el ángulo de deflexión del timón de profundidad, d_T y el ángulo de las palancas de los motores, d_{TH}.

Las señales de referencia se eligieron teniendo en cuenta las especificaciones de diseño que se vieron en el capítulo 3. Las seleccionadas fueron la señal de referencia de la componente z de la velocidad inercial, w_{Vc}, y la referencia de la velocidad respecto al aire, V_{Ac}, para el lazo interno, y la referencia de la posición Z_c para el lazo externo.

Controlador de lazo interno	
q	Velocidad de cabeceo
n_x	Factor de carga horizontal
n_x	Factor de carga vertical
w_V	Componente z de la velocidad inercial
V_A	Velocidad respecto al aire
Controlador de lazo externo	
z	Posición z del centro de gravedad del avión

Tabla 5.1: Medidas utilizadas en el modo longitudinal.

El controlador de lazo interno se representa en la figura 5.3. En dicha figura puede verse que el controlador tiene dos partes: una matriz de ganancias constantes actuando sobre las cinco señales de medida seleccionadas, *LonKp* y otra matriz de ganancias constantes actuando sobre la integral de los errores en las variables w_V y V_A para eliminar los errores en el estacionario, *LonKi*.

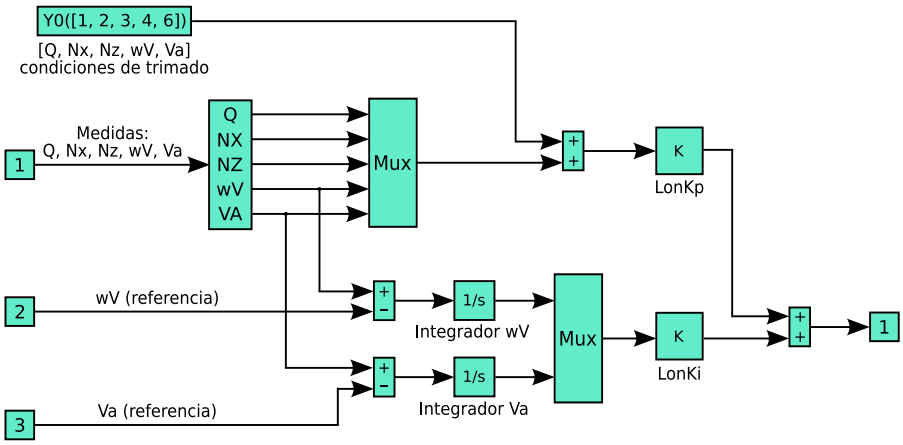


Figura 5.3: Lazo interno del controlador longitudinal.

5.2.2. El controlador lateral.

El controlador lateral tendrá una estructura similar a la que se acaba de ver para el modo longitudinal, como puede comprobarse en la figura 5.4.

El valor K del bloque *LatKout* será en este caso la ganancia de lazo externo que, de igual forma que en el modo longitudinal, se obtendrá por el método del lugar de las

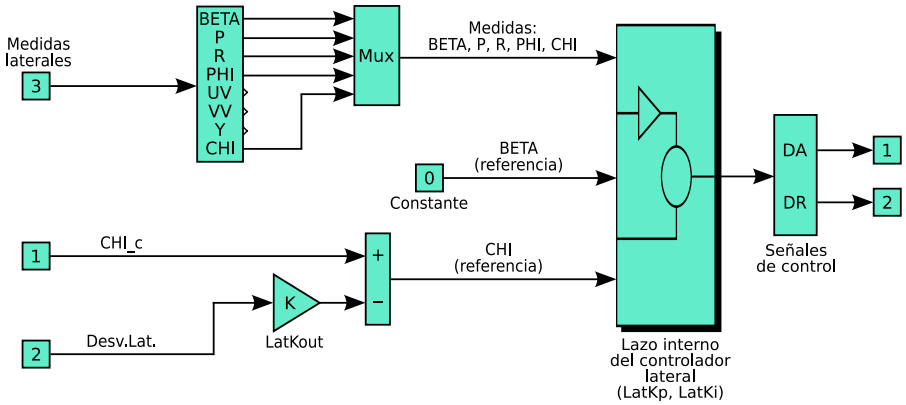


Figura 5.4: Controlador lateral.

raíces.

De las 8 medidas proporcionadas por el modelo linealizado, sólo 5 se tuvieron en cuenta para el diseño del controlador. Las medidas empleadas se muestran en la tabla 5.2.

Controlador de lazo interno	
β	Ángulo de deslizamiento
p	Velocidad de alabeo
r	Velocidad de guiñada
ϕ	Ángulo de alabeo
χ	Ángulo de trayectoria inercial
Controlador de lazo externo	
Δy	Desviación lateral

Tabla 5.2: Medidas utilizadas en el modo lateral.

Las señales de control generadas por el controlador lateral serán: el ángulo de deflexión de los alerones, δ_A y el ángulo de deflexión del timón de dirección, δ_R.

Las señales de referencia seleccionadas para el modo lateral fueron el ángulo de trayectoria inercial, χ_C y el ángulo de deslizamiento , β_C, para el lazo interno, y la posición lateral, y_C, para el lazo externo. Se escogió un valor nulo para β_C, con objeto de mantener siempre el valor de β próximo a cero.

La estructura del controlador de lazo interno se muestra en la figura 5.5. En dicha figura puede verse que el controlador tiene dos partes: una matriz de ganancias cons-

tantes actuando sobre las cinco señales de medida seleccionadas, *LatKp* y otra matriz de ganancias constantes actuando sobre la integral de los errores en las variables β y χ para eliminar los errores en el estacionario, *LonKi*.

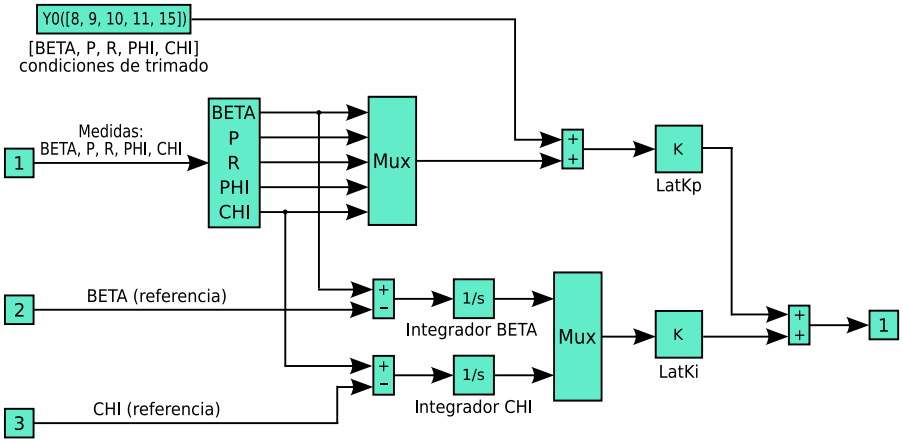


Figura 5.5: Lazo interno del controlador lateral.

5.3. Particularidades del algoritmo.

Como ya se ha comentado en la introducción, para la determinación de las ganancias de lazo interno, se empleará el algoritmo que se diseñó en el capítulo 3 y se aplicó en el capítulo 4. Las diferencias del presente caso respecto al anterior, se encuentran en los modelos empleados en la simulación y en los que se utilizarán para obtener las medidas de robustez. Dichos modelos, descritos a continuación, serán utilizados por la función de evaluación para determinar los distintos elementos del vector de aptitud.

Los parámetros del algoritmo evolutivo para el presente caso, se recogen en la tabla 5.3.

5.3.1. Modelos empleados en la simulación.

Los modelos que ahora se describen, serán utilizados por la función de evaluación para llevar a cabo una simulación, a partir de la cual determinará los distintos elementos del vector de aptitud, a excepción de aquellos relativos a la robustez.

Se trabajará con dos modelos, uno para el modo longitudinal y otro para el lateral. La estructura será idéntica en ambos casos y se ajusta a la mostrada en la figura 5.6.

Parámetro	**Valor**
Tamaño de la población	100
Número de genes por cromosoma	14
Probabilidad de cruce en un punto	0.2
Probabilidad de cruce uniforme	0.2
Probabilidad de cruce aritmético	0.2
Probabilidad de cruce heurístico	0.2
Probabilidad de mutación aleatoria	0.1
Probabilidad de mutación no uniforme	0.1
Número de competidores en concurso de selección	2
Porcentaje de descendientes generados a partir de padres	85 %
Máxima parte real permitida para los autovalores	-0.15
Exponente para mutación no uniforme	3
Valor mínimo para los genes	-10
Valor máximo para los genes	10
Número de generaciones estimadas para la convergencia (modo longitudinal)	5000
Número de generaciones estimadas para la convergencia (modo lateral)	3000

Tabla 5.3: Valores de los parámetros del algoritmo evolutivo.

La diferencia estará en las matrices de estados que se emplean para el avión en cada caso. Dichas matrices, tanto para el modo longitudinal como para el lateral, son las que se recogen en el apéndice A. Además las salidas realimentadas hacia K_p y hacia la generación de la señal de error, son las que se muestran en las figuras 5.3 y 5.5.

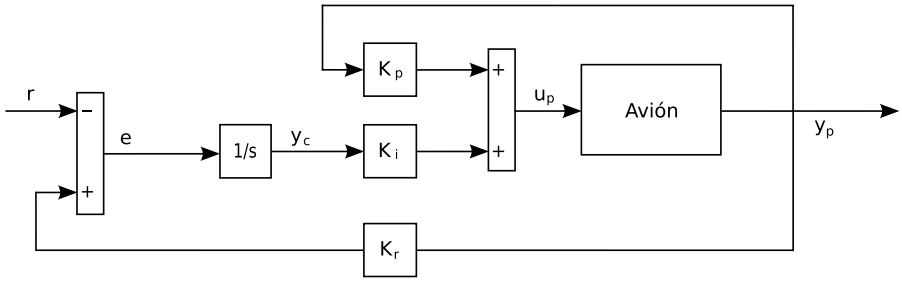


Figura 5.6: Estructura del modelo empleado en la simulación.

El modelo global que se va a obtener, será aplicable en ambos casos. Sea

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_p & = & \boldsymbol{A}_p \cdot \boldsymbol{x}_p + \boldsymbol{B}_p \cdot \boldsymbol{u}_p \\ \boldsymbol{y}_p & = & \boldsymbol{C}_p \cdot \boldsymbol{x}_p \end{array}\right\} \tag{5.1}$$

el sistema en el espacio de estados correspondiente al avión, y

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_c & = & \boldsymbol{A}_c \cdot \boldsymbol{x}_c + \boldsymbol{B}_c \cdot \boldsymbol{e} \\ \boldsymbol{y}_c & = & \boldsymbol{C}_c \cdot \boldsymbol{x}_c \end{array}\right\} \tag{5.2}$$

el sistema en el espacio de estados correspondiente a los integradores, siendo

$$\boldsymbol{e} = \boldsymbol{K}_r \boldsymbol{y}_p - \boldsymbol{r} \tag{5.3}$$

donde K_r es una matriz que seleccionará las salidas a integrar. La señal de control de la planta será

$$\boldsymbol{u}_p = \boldsymbol{K}_p \boldsymbol{y}_p + \boldsymbol{K}_i \boldsymbol{y}_c \tag{5.4}$$

Sustituyendo 5.3 en 5.2, se obtiene

$$\dot{\boldsymbol{x}}_c = \boldsymbol{A}_c \boldsymbol{x}_c + \boldsymbol{B}_c \left(\boldsymbol{K}_r \boldsymbol{y}_p - \boldsymbol{r}\right) = \boldsymbol{A}_c \boldsymbol{x}_c + \boldsymbol{B}_c \boldsymbol{K}_r \boldsymbol{C}_p \boldsymbol{x}_p - \boldsymbol{B}_c \boldsymbol{r} \tag{5.5}$$

Y sustituyendo a su vez 5.4 en 5.1

$$\dot{\boldsymbol{x}}_p = \boldsymbol{A}_p \boldsymbol{x}_p + \boldsymbol{B}_p \left(\boldsymbol{K}_p \boldsymbol{y}_p + \boldsymbol{K}_i \boldsymbol{y}_c\right) = \boldsymbol{A}_p \boldsymbol{x}_p + \boldsymbol{B}_p \boldsymbol{K}_p \boldsymbol{C}_p \boldsymbol{x}_p + \boldsymbol{B}_p \boldsymbol{K}_i \boldsymbol{C}_c \boldsymbol{x}_c \tag{5.6}$$

Y agrupando las dinámicas de la planta y los integradores en una única ecuación, e incluyendo las señales de control como salidas, por ser necesarias para la función de evaluación, se obtiene el modelo global

$$\begin{array}{rcl} \begin{bmatrix} \dot{\boldsymbol{x}}_p \\ \dot{\boldsymbol{x}}_c \end{bmatrix} & = & \begin{bmatrix} \boldsymbol{A}_p + \boldsymbol{B}_p \boldsymbol{K}_p \boldsymbol{C}_p & \boldsymbol{B}_p \boldsymbol{K}_i \boldsymbol{C}_c \\ \boldsymbol{B}_c \boldsymbol{K}_r \boldsymbol{C}_p & \boldsymbol{A}_c \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{B}_c \end{bmatrix} \cdot \boldsymbol{r} \\ \begin{bmatrix} \boldsymbol{y}_p \\ \boldsymbol{u}_p \end{bmatrix} & = & \begin{bmatrix} \boldsymbol{C}_p & \boldsymbol{0} \\ \boldsymbol{K}_p \boldsymbol{C}_p & \boldsymbol{K}_i \boldsymbol{C}_c \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \end{bmatrix} \end{array} \tag{5.7}$$

5.3.2. Modelos para el cálculo de la robustez.

El procedimiento que se utilizará para calcular los márgenes de ganancia y fase de cada controlador, será análogo al que se empleó en el apartado 4.3.3.1. En este caso, el esquema que se utilizará para el cálculo de las funciones de sensibilidad es el que se muestra en la figura 5.7.

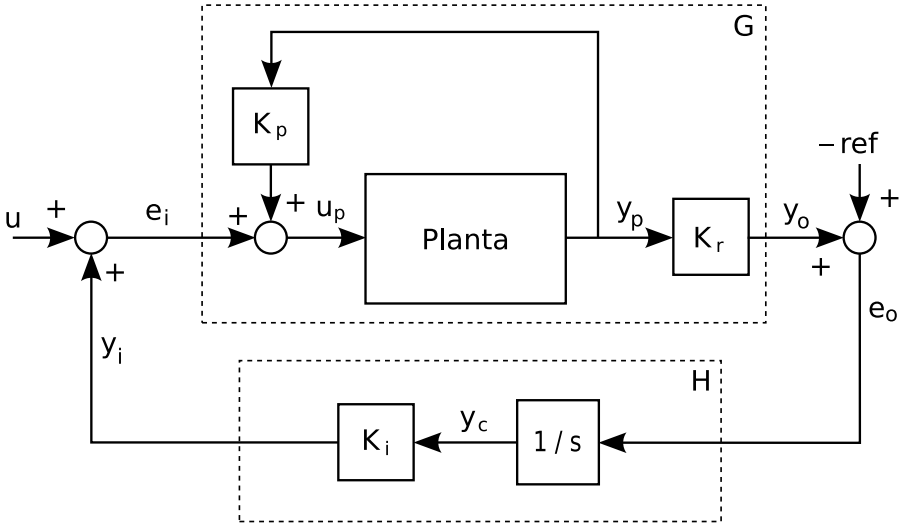


Figura 5.7: Esquema para la determinación de las funciones de sensibilidad.

De la figura 5.7 se deduce que las ecuaciones correspondientes a los bloques G y H serán, en este caso,

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_p &=& (\boldsymbol{A}_p + \boldsymbol{B}_p\boldsymbol{K}_p\boldsymbol{C}_p)\,\boldsymbol{x}_p + \boldsymbol{B}_p\boldsymbol{e}_i \\ \boldsymbol{y}_o &=& \boldsymbol{K}_r\boldsymbol{C}_p\boldsymbol{x}_p \end{array}\right\} \tag{5.8}$$

y

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_c &=& \boldsymbol{A}_c\boldsymbol{x}_c + \boldsymbol{B}_c\boldsymbol{e}_o \\ \boldsymbol{y}_i &=& \boldsymbol{K}_i\boldsymbol{C}_c\boldsymbol{x}_c \end{array}\right\} \tag{5.9}$$

Una vez obtenidas las ecuaciones correspondientes a los bloques G y H, el resto del procedimiento será idéntico al utilizado en el apartado 4.3.3.1. De nuevo serán de

aplicación los esquemas de las figuras 4.4 y 4.5, y se volverá a utilizar el código de Matlab empleado en aquella ocasión, a partir de la obtención de los bloques mencionados.

5.4. Resultados.

El algoritmo fue ejecutado en 3 ocasiones para el modo longitudinal y otras 3 para el modo lateral, al igual que en el caso del controlador LQ. Los resultados se muestran a continuación, ordenados de mejor a peor aptitud. El tiempo medio empleado en la sintonía para el modo longitudinal fue de 3h 15', y para el modo lateral de 1h 45', utilizando un procesador AMD XP 2300.

5.4.1. Modo longitudinal.

Experimento 1.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.16
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 3.6023 & -1.4253 & -0.5771 & -0.1562 & -0.0553 \\ 0.9736 & -2.3184 & 0.6120 & 0.0764 & -0.1378 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0463 & -0.0090 \\ 0.0358 & -0.0184 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.000878209	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$4.35741 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.302042	OK
Error medio de VA al final (<0.01 m/s):	0.000397498	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00745357, 0.0492449]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	[-0.0423243, 0.0107021]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.217933	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.157997	OK

Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.157542	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.00156993	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.255911	OK
Error medio de Wv al final (<0.01 m/s):	0.000861861	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[0, 0.0343377]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	[0, 0.0820886]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.116049	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.234169	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	[-11.9872, 11.9872]	OK
Margen de fase entrada (-50º >MF >50º):	[-61.7594, 61.7594]	OK
Margen de ganancia salida (-10 >MG >10):	[-12.0413, 12.0413]	OK
Margen de fase salida (-50º >MF >50º):	[-61.9277, 61.9277]	OK

Tabla 5.4: Modo longitudinal. Resultados del experimento 1.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

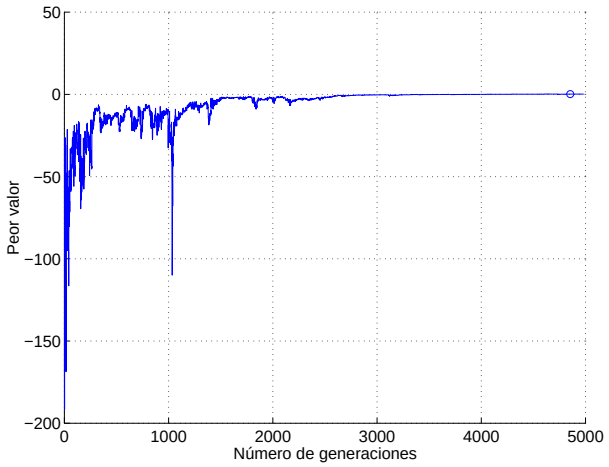
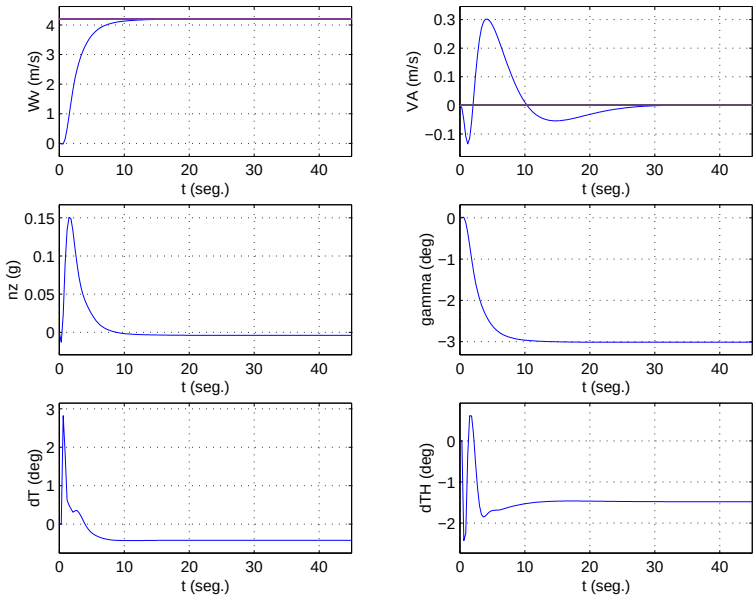
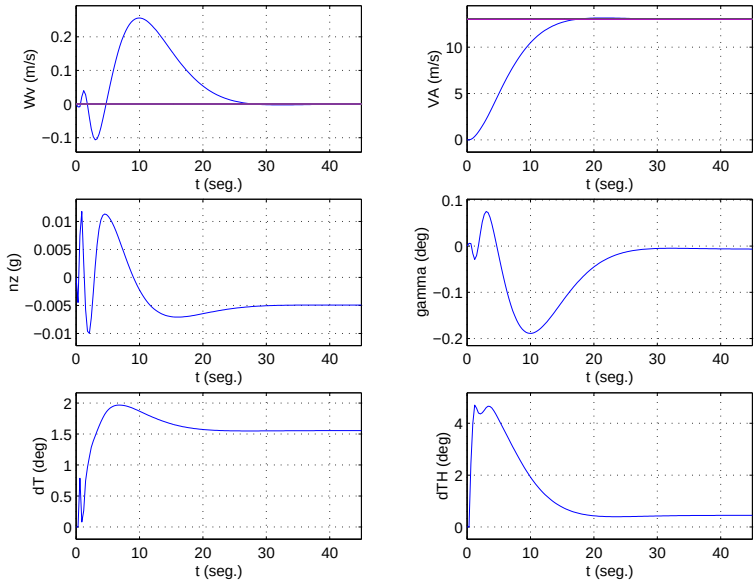


Figura 5.8: Evolución del peor de los índices de aptitud.

- Respuestas de las distintas variables ante señales de tipo escalón en w_V y V_A:



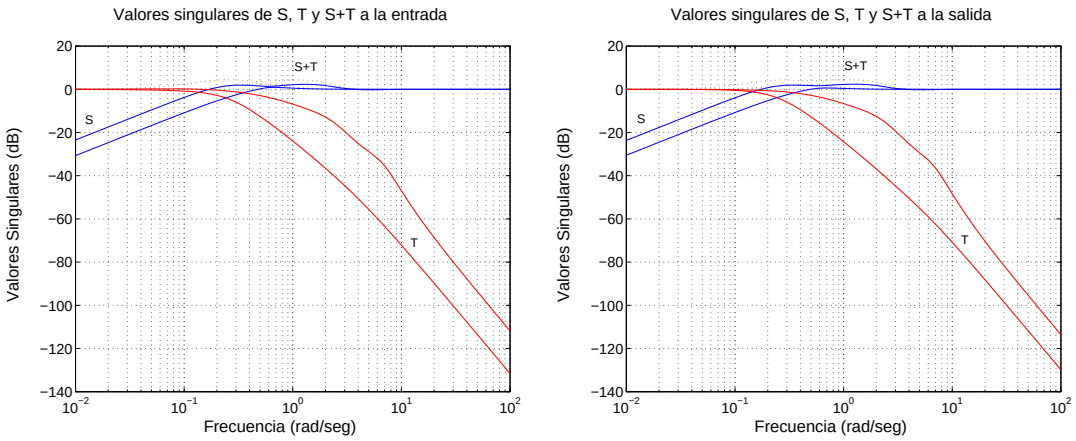
(a)



(b)

Figura 5.9: (a) Respuesta ante una señal de tipo escalón en w_V, (b) Respuesta ante una señal de tipo escalón en V_A.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 5.10: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

Experimento 2.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.159126
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 5.4949 & -2.6082 & -0.4698 & -0.1750 & -0.1099 \\ 0.0274 & -1.9366 & 0.8384 & 0.1097 & -0.1109 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0436 & -0.0160 \\ 0.0491 & -0.0152 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00234222	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$5.03364 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.257533	OK
Error medio de VA al final (<0.01 m/s):	0.00157807	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00771901, 0.0455914]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	[-0.0518527, 0.00832677]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.217806	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.212584	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.278069	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.00404751	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.139788	OK
Error medio de Wv al final (<0.01 m/s):	0.000963353	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00343694, 0.0347812]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	[0, 0.0827043]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.205926	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.191268	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	[-12.0467, 12.0467]	OK
Margen de fase entrada (-50º >MF >50º):	[-61.9447, 61.9447]	OK
Margen de ganancia salida (-10 >MG >10):	[-11.8924, 11.8924]	OK
Margen de fase salida (-50º >MF >50º):	[-61.4621, 61.4621]	OK

Tabla 5.5: Modo longitudinal. Resultados del experimento 2.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

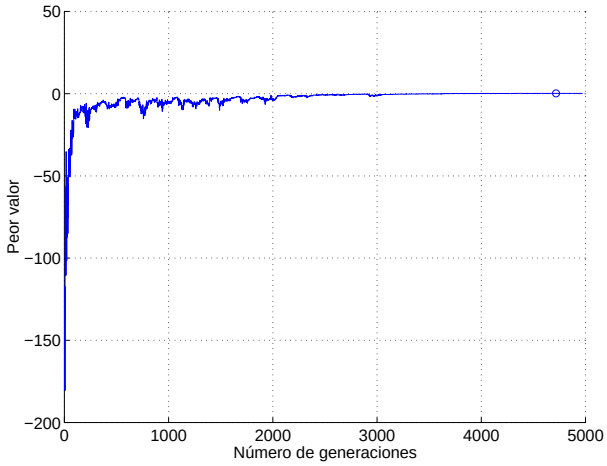
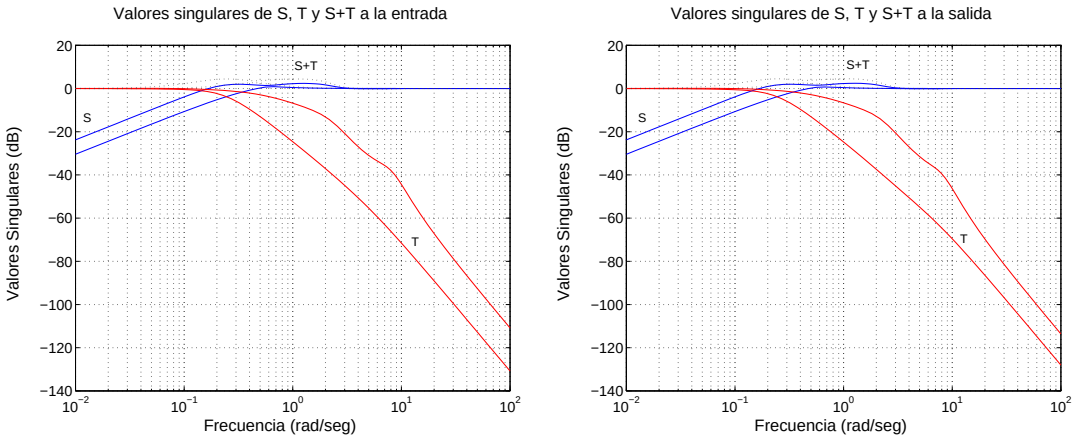


Figura 5.11: Evolución del peor de los índices de aptitud.

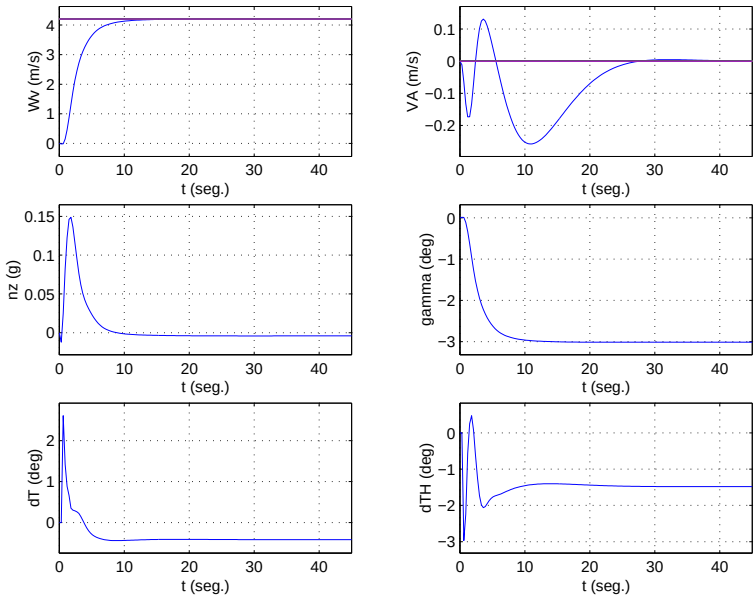
- Gráficas de los valores singulares de las funciones de sensibilidad:



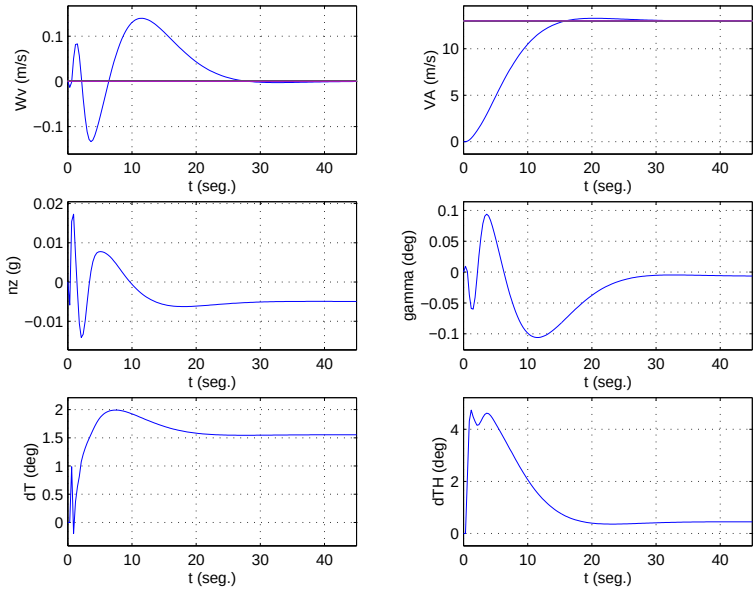
(a) (b)

Figura 5.12: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en w_V y V_A:



(a)



(b)

Figura 5.13: (a) Respuesta ante una señal de tipo escalón en w_V, (b) Respuesta ante una señal de tipo escalón en V_A.

Experimento 3.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.152044
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 5.4151 & -2.4709 & -0.3746 & -0.1776 & -0.1030 \\ 2.0093 & -2.8396 & 0.9802 & 0.0715 & -0.1453 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0473 & -0.0168 \\ 0.0407 & -0.0187 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00549834	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$9.67167 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.265444	OK
Error medio de VA al final (<0.01 m/s):	0.0019705	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[-0.00796367, 0.0460317]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	[-0.0516819, 0.0216384]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.220615	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.195381	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.320835	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.0057247	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.337064	OK
Error medio de Wv al final (<0.01 m/s):	0.00103332	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	[0, 0.0337099]	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	[0, 0.0834728]	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.215222	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.234126	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	[-11.8655, 11.8655]	OK
Margen de fase entrada (-50º >MF >50º):	[-61.3774, 61.3774]	OK
Margen de ganancia salida (-10 >MG >10):	[-11.8093, 11.8093]	OK
Margen de fase salida (-50º >MF >50º):	[-61.199, 61.199]	OK

Tabla 5.6: Modo longitudinal. Resultados del experimento 3.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

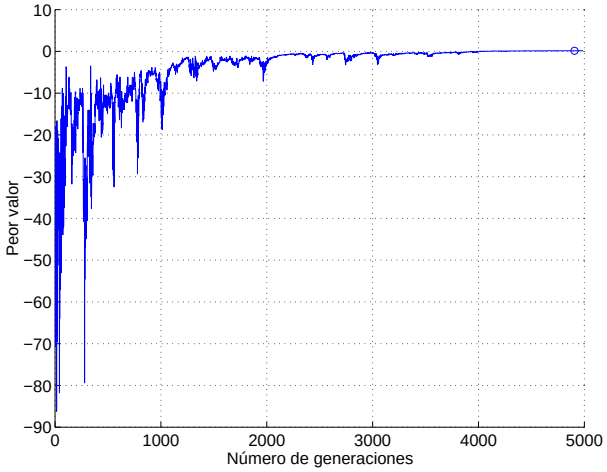
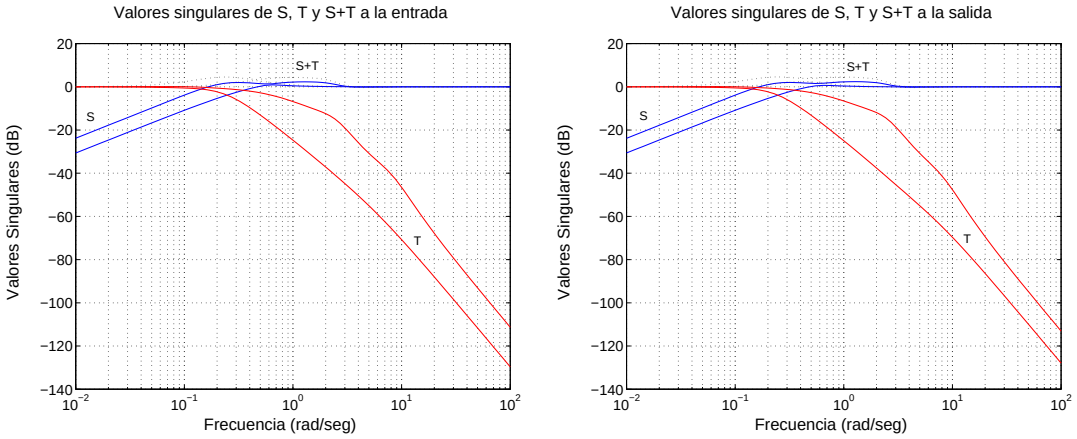


Figura 5.14: Evolución del peor de los índices de aptitud.

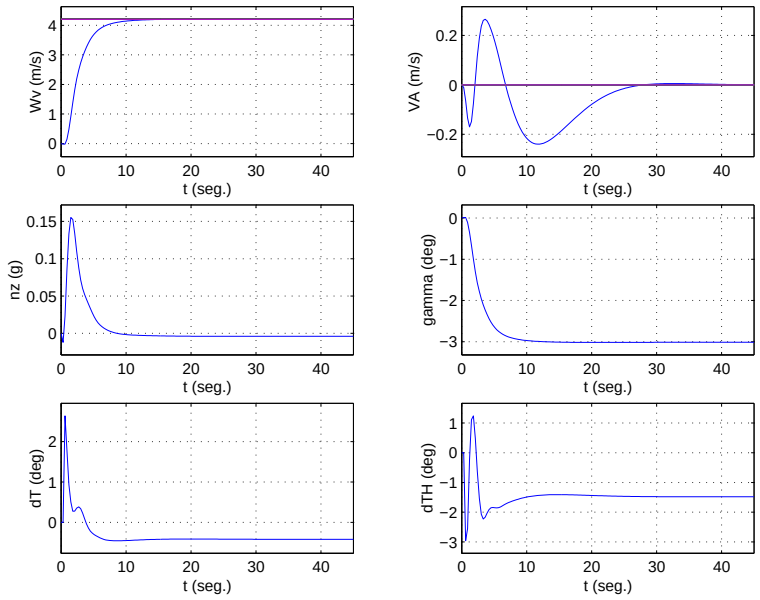
- Gráficas de los valores singulares de las funciones de sensibilidad:



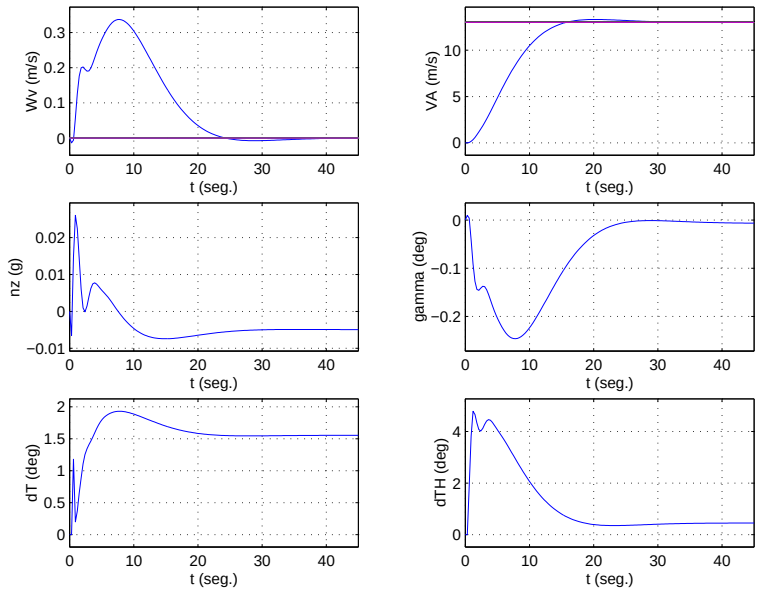
(a) (b)

Figura 5.15: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en w_V y V_A:



(a)



(b)

Figura 5.16: (a) Respuesta ante una señal de tipo escalón en w_V, (b) Respuesta ante una señal de tipo escalón en V_A.

5.4.1.1. Obtención de la ganancia de lazo externo.

Con el mejor controlador obtenido para el modo longitudinal y el mejor del modo lateral, se construirá el controlador completo, con el fin de poder evaluarlo con la herramienta *evaluate*, como ya se ha comentado.

Para implementar dicho controlador se deberán calcular previamente las ganancias de lazo externo $LonKout$ y $LatKout$ de las figuras 5.2 y 5.4 respectivamente.

Para el modo longitudinal, se obtendrá la función de transferencia de lazo abierto del sistema representado en la figura 5.17, y a partir de ésta, mediante el método del lugar de las raíces se obtendrá la ganancia de lazo externo $LonKout$.

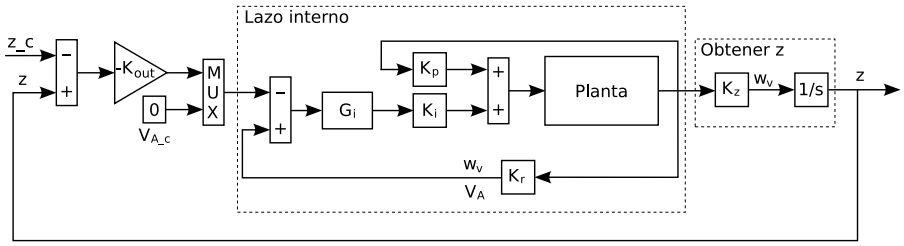


Figura 5.17: Sistema para la determinación de la ganancia del lazo externo (modo longitudinal).

Para la determinación de $LonKout$ se utilizarán las funciones `rlocus` y `rlocfind` de Matlab. El código empleado se corresponde con la función `extrlooplon.m`, recogida en el apartado E.2 del apéndice E. Utilizando dicha función, se escogió

$$LonKout = 0.18362$$

Para el sistema resultante, la función `damp` de Matlab, devuelve los siguientes valores:

```
      Eigenvalue            Damping     Freq. (rad/s)

-2.46e+000 + 6.33e+000i     3.62e-001      6.79e+000
-2.46e+000 - 6.33e+000i     3.62e-001      6.79e+000
-1.26e+000 + 2.00e+000i     5.33e-001      2.36e+000
-1.26e+000 - 2.00e+000i     5.33e-001      2.36e+000
-1.48e+000                  1.00e+000      1.48e+000
-2.26e-001 + 2.02e-001i     7.46e-001      3.03e-001
```

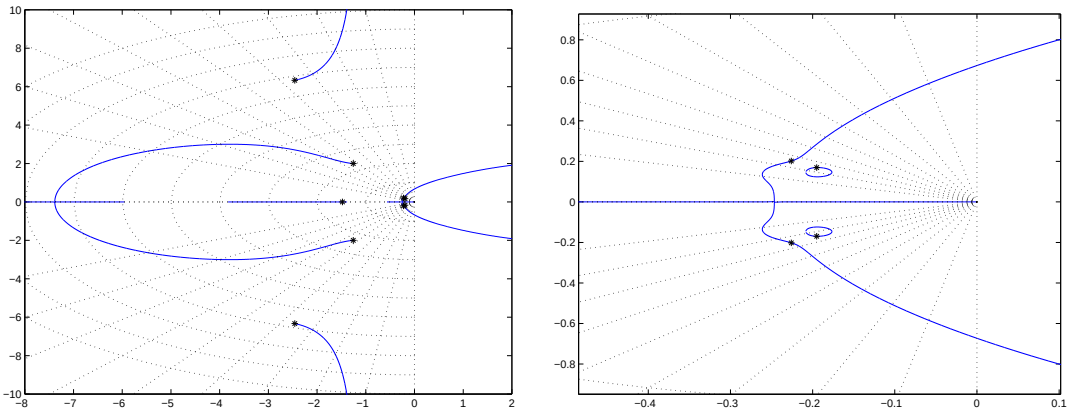
```
-2.26e-001 - 2.02e-001i     7.46e-001      3.03e-001
-1.95e-001 + 1.70e-001i     7.55e-001      2.59e-001
-1.95e-001 - 1.70e-001i     7.55e-001      2.59e-001
```

Por su parte, la función `trts.m` de [36] y [37], recogida también en la sección E.2 del apéndice E, devuelve los valores:

```
          Root              Rise time        Settling time

  -2.4572 +     6.3312i       0.18128             1.9027
  -2.4572 -     6.3312i       0.18128             1.9027
  -1.2564 +     1.9962i       0.68076              3.798
  -1.2564 -     1.9962i       0.68076              3.798
  -1.4785                      1.4862              3.1149
 -0.22596 +    0.20179i       7.6317              22.179
 -0.22596 -    0.20179i       7.6317              22.179
 -0.19507 +    0.16964i       9.0766              25.767
 -0.19507 -    0.16964i       9.0766              25.767
```

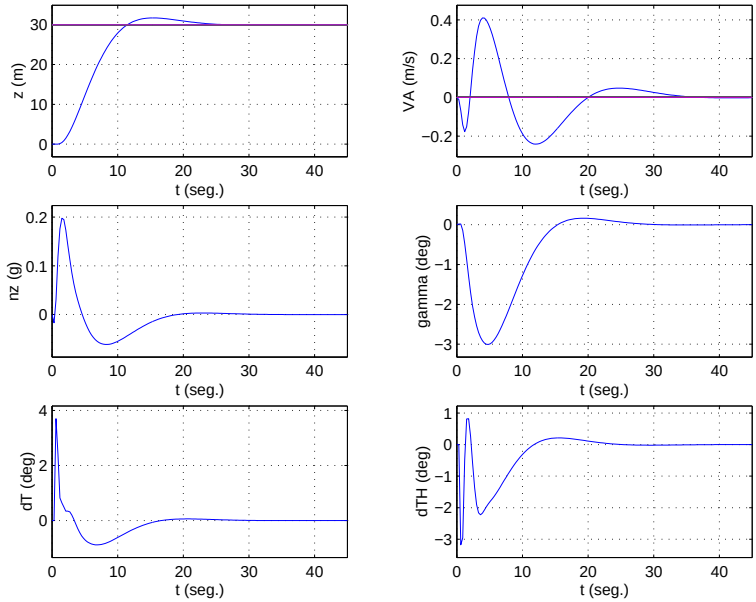
La figura 5.18 muestra el lugar de las raíces para dos escalas diferentes.



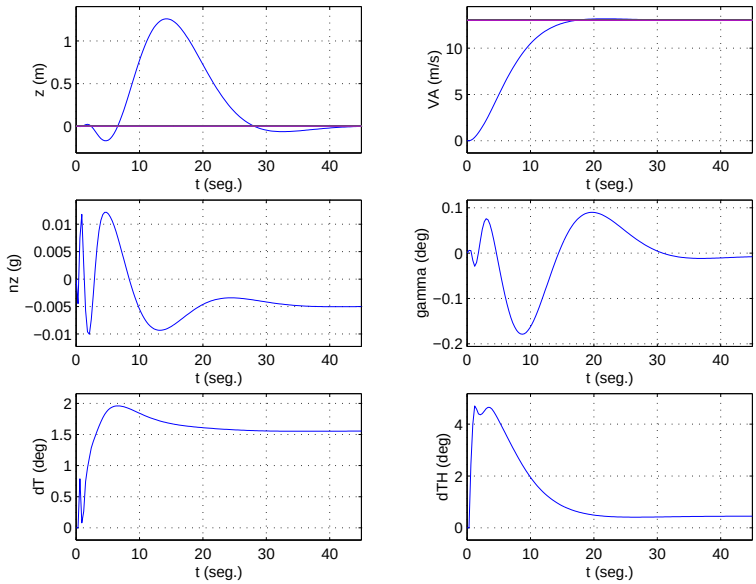
(a) (b)

Figura 5.18: Lugar de las raíces con dos escalas diferentes (modo longitudinal).

Finalmente, se comprobó el comportamiento del controlador longitudinal completo resultante, mediante la aplicación de sendas señales de tipo escalón en z y en V_A. Los resultados se muestran en la figura 5.19.



(a)



(b)

Figura 5.19: (a) Respuesta ante una señal de tipo escalón en z, (b) Respuesta ante una señal de tipo escalón en V_A.

Para el escalón en z, se observa un sobreimpulso menor que el 5 % y un acoplamiento con V_A menor de 0.5 m/s. El tiempo de subida es menor de 12 s y el tiempo de asentamiento es menor de 45 s. La aceleración vertical es algo mayor que la especificada de 0.05 g.

Para el escalón en V_A, la aceleración vertical es menor que para el caso del escalón en z. El resto de objetivos es satisfecho.

5.4.2. Modo lateral.

Experimento 1.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.113851
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -2.9363 & 0.0448 & 1.5478 & 1.5823 & 6.8657 \\ -9.1056 & -0.3034 & 4.3653 & -0.5592 & 0.5051 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} 0.1254 & 1.0988 \\ -2.0561 & 0.0990 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Chi:		
Máximo sobreimpulso en chi (<0.0174533 rad):	$6.59622 \cdot 10^{-5}$	OK
Máximo tiempo de subida de chi (<10 seg):	8.7	OK
Error medio de chi al final (<0.00349066 rad):	$2.65092 \cdot 10^{-6}$	OK
Pico máximo en beta debido a chi (<0.0052 rad):	0.00296938	OK
Error medio de beta al final (<0.0001 rad):	$6.17927 \cdot 10^{-6}$	OK
Saturación dA (-0.4363 rad <dA <0.4363 rad):	[-0.386151, 0.283549]	OK
Saturación dR (-0.5236 rad <dR <0.5236 rad):	[-0.10399, $3.37271 \cdot 10^{-5}$]	OK
Máxima variación de dA (|d(dA)| <0.4363 rad/seg):	0.383565	OK
Máxima variación de dR (|d(dR)| <0.4363 rad/seg):	0.0414864	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 dB >MG >10 dB):	[-11.2848, 11.2848]	OK
Margen de fase entrada (-50º >MF >50º):	[-59.4876, 59.4876]	OK
Margen de ganancia salida (-10 dB >MG >10 dB):	[-11.286, 11.286]	OK
Margen de fase salida (-50º >MF >50º):	[-59.4918, 59.4918]	OK

Tabla 5.7: Modo lateral. Resultados del experimento 1.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

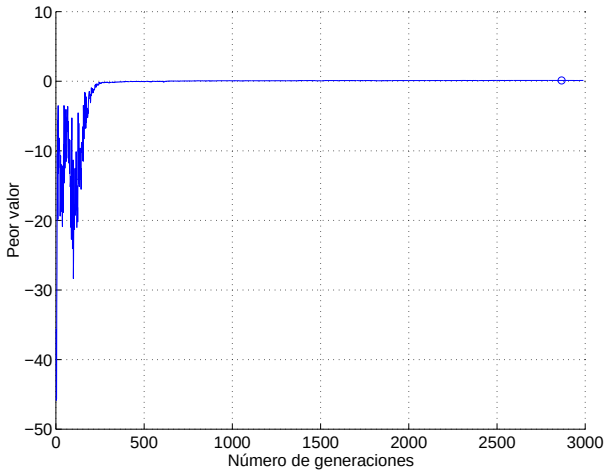
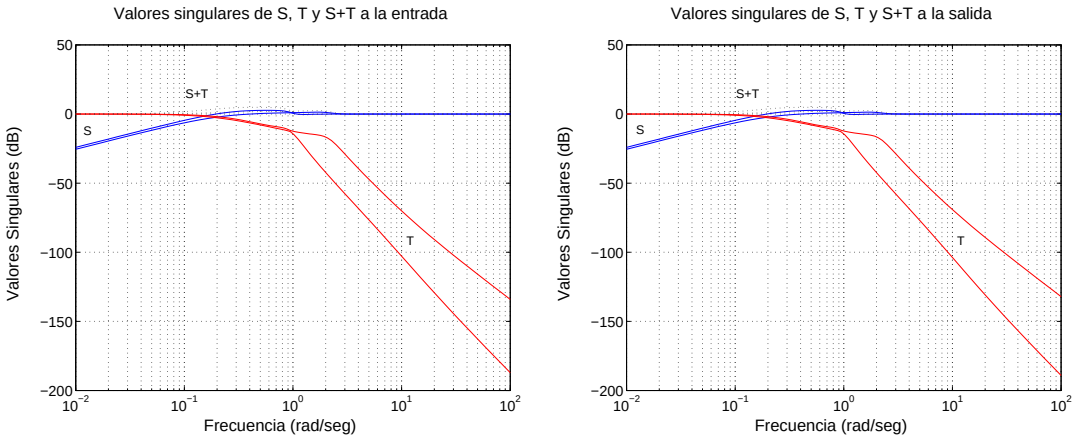


Figura 5.20: Evolución del peor de los índices de aptitud.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 5.21: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en χ:

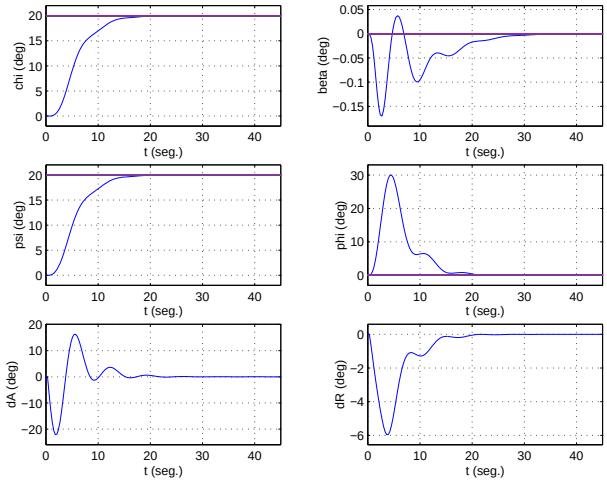


Figura 5.22: Respuesta ante una señal de tipo escalón en χ.

Experimento 2.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.111829
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} -0.2954 & 0.1732 & -1.8456 & 2.0070 & 6.9432 \\ -9.7913 & 0.1983 & 7.4759 & -0.6051 & 2.2677 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} 0.9413 & 1.0968 \\ -2.6575 & 0.3544 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Chi:		
Máximo sobreimpulso en chi (<0.0174533 rad):	0	OK
Máximo tiempo de subida de chi (<10 seg):	8.7	OK
Error medio de chi al final (<0.00349066 rad):	$7.18675 \cdot 10^{-6}$	OK
Pico máximo en beta debido a chi (<0.0052 rad):	0.00406351	OK
Error medio de beta al final (<0.0001 rad):	$3.65478 \cdot 10^{-6}$	OK
Saturación dA (-0.4363 rad <dA <0.4363 rad):	[-0.387509, 0.246084]	OK
Saturación dR (-0.5236 rad <dR <0.5236 rad):	[-0.111269, $2.22082 \cdot 10^{-5}$]	OK
Máxima variación de dA (|d(dA)| <0.4363 rad/seg):	0.382858	OK
Máxima variación de dR (|d(dR)| <0.4363 rad/seg):	0.123665	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 dB >MG >10 dB):	[-11.2596, 11.2596]	OK
Margen de fase entrada (-50º >MF >50º):	[-59.4032, 59.4032]	OK
Margen de ganancia salida (-10 dB >MG >10 dB):	[-11.2626, 11.2626]	OK
Margen de fase salida (-50º >MF >50º):	[-59.4131, 59.4131]	OK

Tabla 5.8: Modo lateral. Resultados del experimento 2.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

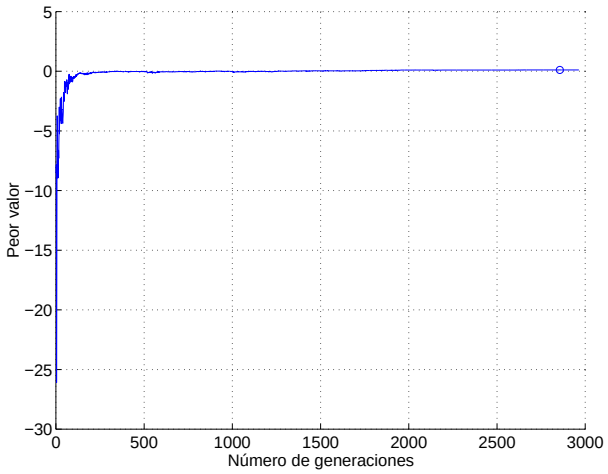
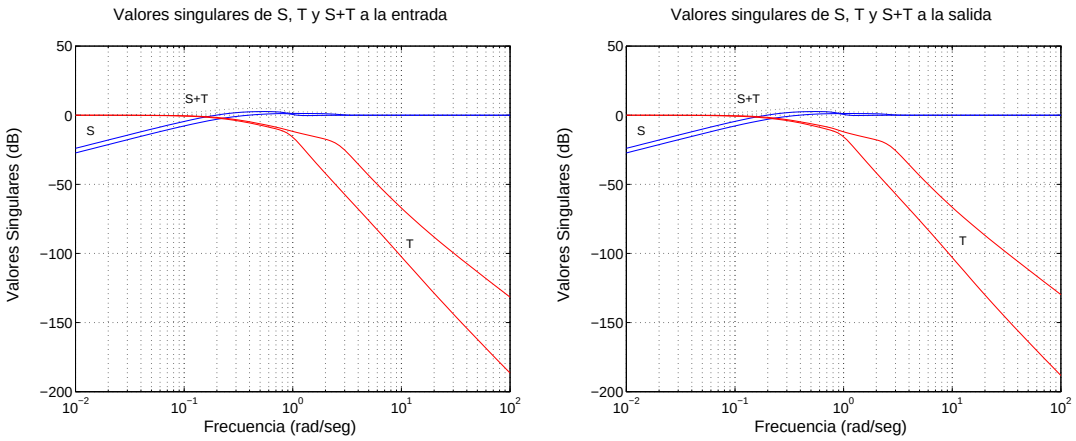


Figura 5.23: Evolución del peor de los índices de aptitud.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 5.24: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables frente a señales de tipo escalón en χ:

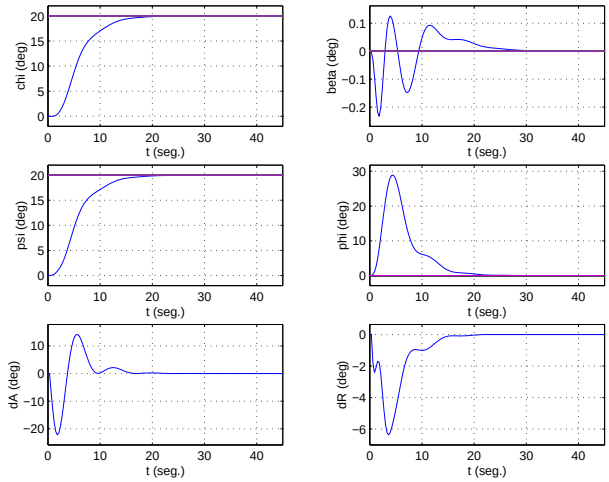


Figura 5.25: Respuesta ante una señal de tipo escalón en χ.

Experimento 3.

- Peor índice del vector de aptitud del mejor cromosoma obtenido: 0.1
- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 1.1886 & 0.3550 & -4.0413 & 2.3193 & 7.1324 \\ -9.9722 & 0.4960 & 9.7104 & -0.6274 & 3.5377 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} 1.9061 & 1.0918 \\ -3.6561 & 0.5087 \end{bmatrix}$$

- Objetivos de la función de evaluación:

Escalón en Chi:		
Máximo sobreimpulso en chi (<0.0174533 rad):	0	OK
Máximo tiempo de subida de chi (<10 seg):	9	OK
Error medio de chi al final (<0.00349066 rad):	$3.41151 \cdot 10^{-5}$	OK
Pico máximo en beta debido a chi (<0.0052 rad):	0.00452948	OK
Error medio de beta al final (<0.0001 rad):	$3.14578 \cdot 10^{-6}$	OK
Saturación dA (-0.4363 rad <dA <0.4363 rad):	[-0.371392, 0.195118]	OK
Saturación dR (-0.5236 rad <dR <0.5236 rad):	[-0.102773, 0]	OK
Máxima variación de dA (|d(dA)| <0.4363 rad/seg):	0.385059	OK
Máxima variación de dR (|d(dR)| <0.4363 rad/seg):	0.177512	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 dB >MG >10 dB):	[-11.2958, 11.2958]	OK
Margen de fase entrada (-50 >MF >50):	[-59.5244, 59.5244]	OK
Margen de ganancia salida (-10 dB >MG >10 dB):	[-11.3225, 11.3225]	OK
Margen de fase salida (-50 >MF >50):	[-59.6138, 59.6138]	OK

Tabla 5.9: Modo lateral. Resultados del experimento 3.

- Evolución del peor índice de aptitud del mejor cromosoma, a través de las sucesivas generaciones:

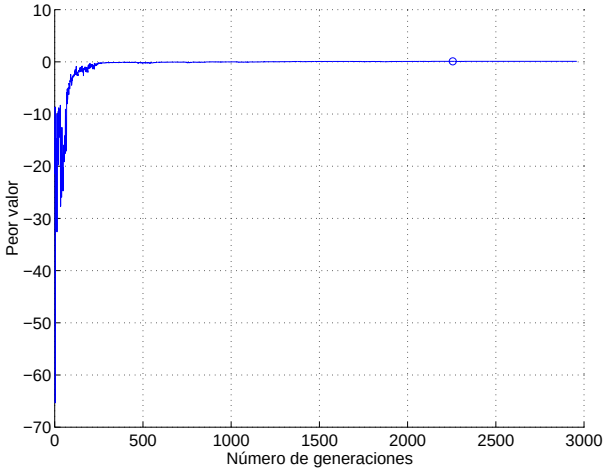
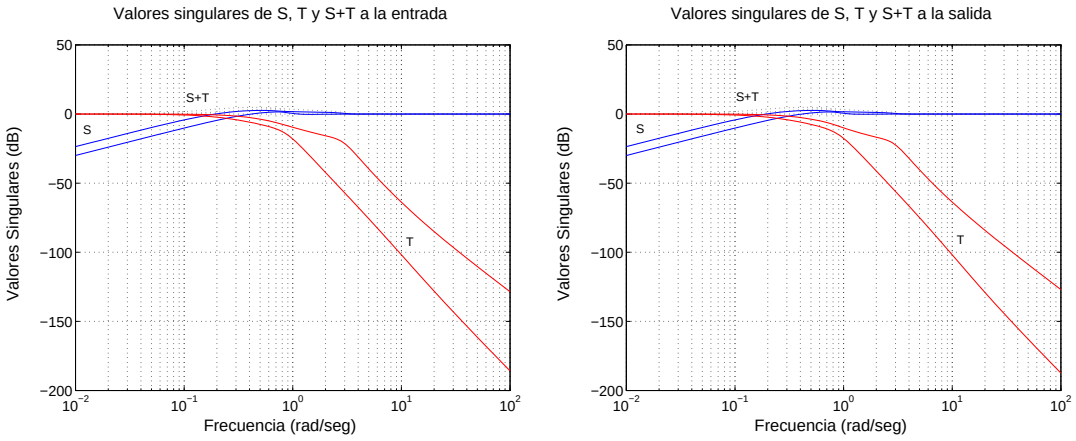


Figura 5.26: Evolución del peor de los índices de aptitud.

- Gráficas de los valores singulares de las funciones de sensibilidad:



(a) (b)

Figura 5.27: (a) Valores singulares de las funciones de sensibilidad a la entrada, (b) Valores singulares de las funciones de sensibilidad a la salida.

- Respuestas de las distintas variables ante señales de tipo escalón en χ:

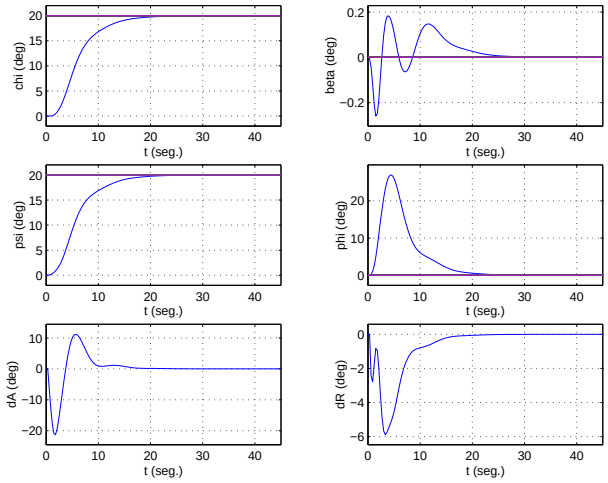


Figura 5.28: Respuesta ante una señal de tipo escalón en χ.

5.4.2.1. Obtención de la ganancia de lazo externo.

Procediendo de igual forma a como se hizo para el modo longitudinal, se obtendrá la ganancia de lazo externo $LatKout$. En el caso del modo lateral, se obtendrá la función de transferencia de lazo abierto del sistema representado en la figura 5.29, para obtenerse posteriormente la ganancia por el método del lugar de las raíces, como se hizo para el modo longitudinal.

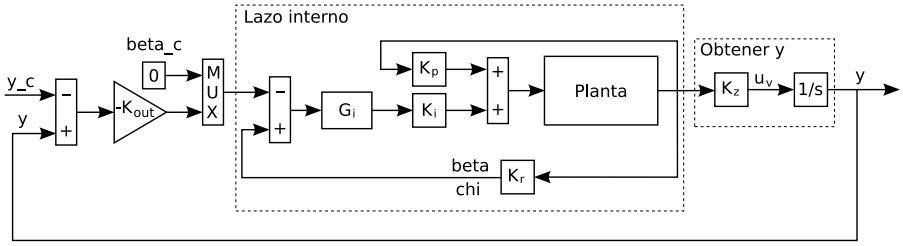


Figura 5.29: Sistema para la determinación de la ganancia del lazo externo (modo lateral).

Para el modo lateral, la ganancia escogida fue

$$LatKout = 0.0010885$$

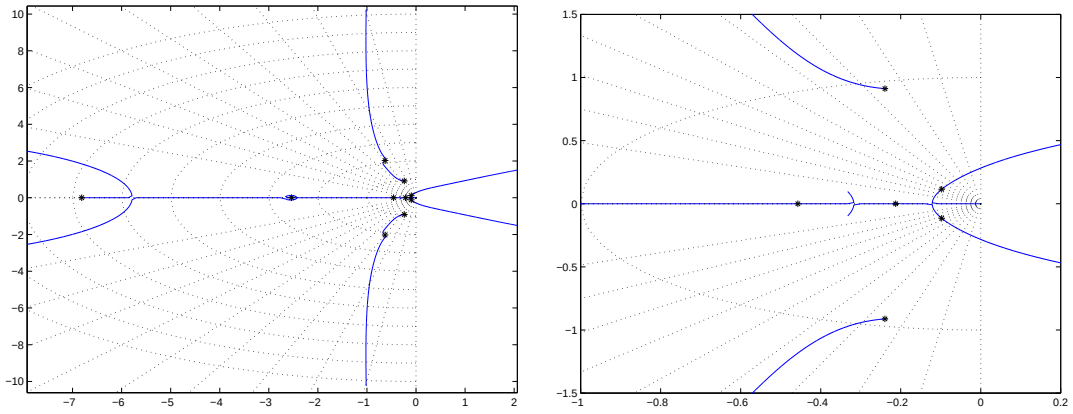
Para el sistema resultante, la función `damp` de Matlab, devuelve los siguientes valores:

Eigenvalue	Damping	Freq. (rad/s)
-6.81e+000	1.00e+000	6.81e+000
-2.54e+000	1.00e+000	2.54e+000
-6.31e-001 + 2.03e+000i	2.97e-001	2.12e+000
-6.31e-001 - 2.03e+000i	2.97e-001	2.12e+000
-2.39e-001 + 9.12e-001i	2.54e-001	9.43e-001
-2.39e-001 - 9.12e-001i	2.54e-001	9.43e-001
-4.57e-001	1.00e+000	4.57e-001
-2.13e-001	1.00e+000	2.13e-001
-9.76e-002 + 1.15e-001i	6.48e-001	1.51e-001
-9.76e-002 - 1.15e-001i	6.48e-001	1.51e-001

Por su parte, la función `trts`, ya empleada en el modo longitudinal, devuelve los valores:

Root	Rise time	Settling time
-6.8142	0.32245	0.67582
-2.5373	0.86595	1.815
-0.631 + 2.0271i	0.53406	7.3714
-0.631 - 2.0271i	0.53406	7.3714
-0.23944 + 0.91219i	1.1475	19.372
-0.23944 - 0.91219i	1.1475	19.372
-0.4571	4.8069	10.075
-0.21258	10.336	21.663
-0.097587 + 0.11477i	12.971	49.978
-0.097587 - 0.11477i	12.971	49.978

La figura 5.30 muestra el lugar de las raíces para dos escalas diferentes.



(a) (b)

Figura 5.30: Lugar de las raíces con dos escalas diferentes (modo lateral).

Finalmente, se aplicaron señales de tipo escalón en y para comprobar el comportamiento del controlador lateral resultante. Los resultados se muestran en la figura 5.31.

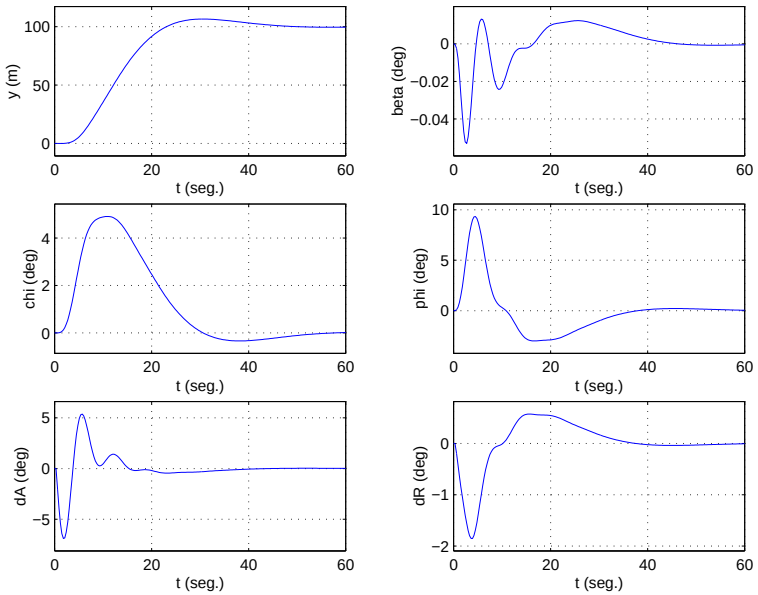


Figura 5.31: Respuesta ante una señal de tipo escalón en y.

El sobreimpulso es menor que el 5 % especificado. La desviación lateral se reduce por debajo del 10 % dentro de los 30 s especificados. El acoplamiento con β es muy pequeño.

5.4.3. Evaluación del controlador completo.

Esta sección está basada en el procedimiento de evaluación definido en [55]. El controlador será evaluado durante una maniobra de aproximación. La trayectoria se ha descompuesto en cuatro segmentos, que serán estudiados de forma independiente. Con objeto de estudiar la robustez, se llevarán a cabo cuatro simulaciones en diferentes condiciones. Las figuras que se muestran a continuación contienen superpuestas las gráficas de las cuatro simulaciones.

La figura 5.32 muestra la trayectoria seguida por el modelo, junto con la trayectoria de referencia, aunque son prácticamente indistinguibles. También se muestran los cuatro segmentos mencionados.

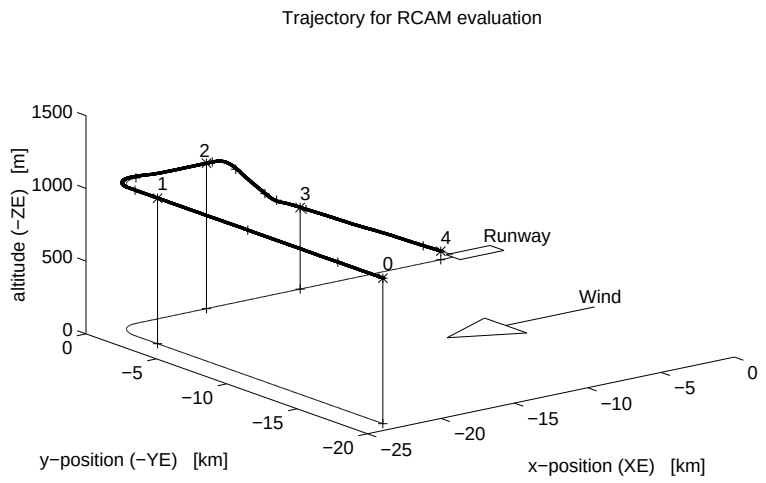


Figura 5.32: Trayectoria seguida por el modelo durante la evaluación del controlador.

La figura 5.33 corresponde al segmento I (de 0 a 1 en la figura 5.32). Durante este segmento se estudian algunas características laterales del controlador, mediante la simulación de un fallo en el motor izquierdo: el fallo ocurre en el punto a, tras lo cual el motor es rearrancado en el punto b. Las lineas discontinuas muestran los límites de las especificaciones, comprobándose que en ningún caso son sobrepasados. Además, las representaciones de las cuatro simulaciones son casi indistinguibles, lo que indica una buena robustez del diseño.

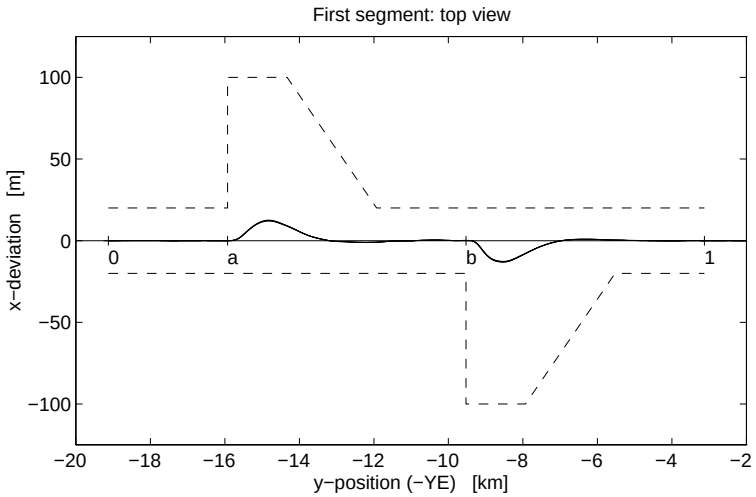


Figura 5.33: Segmento I: Efectos laterales de un fallo en el motor izquierdo.

Las figuras 5.34 y 5.35 corresponden al segmento II (de 1 a 2 en la figura 5.32). Durante este segmento se estudia el comportamiento del controlador en un giro de 90 grados. Los límites de la desviación lateral son sobrepasados, debido a la situación de compromiso existente con las aceleraciones laterales. Un mejor comportamiento durante el giro, supondría una disminución en el confort.

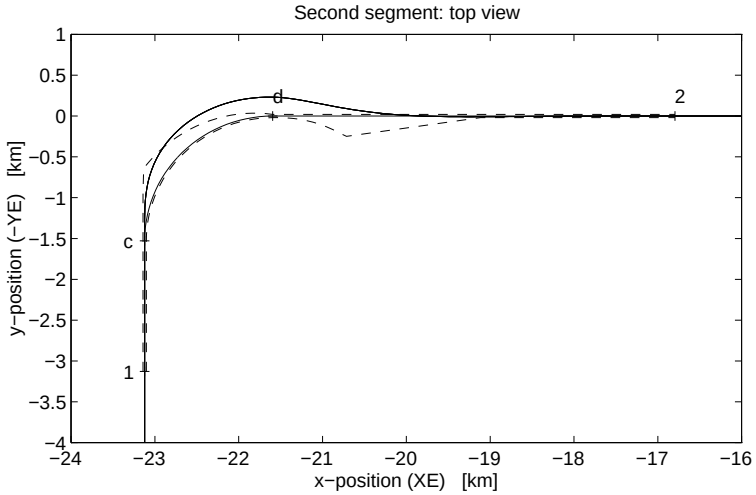


Figura 5.34: Segmento II: Efectos laterales durante un giro de 90º (vista superior).

En la figura 5.35 se ha enderezado la curva con objeto de incrementar el nivel de detalle. De nuevo se observa que las gráficas correspondientes a las cuatro simulaciones están prácticamente superpuestas.

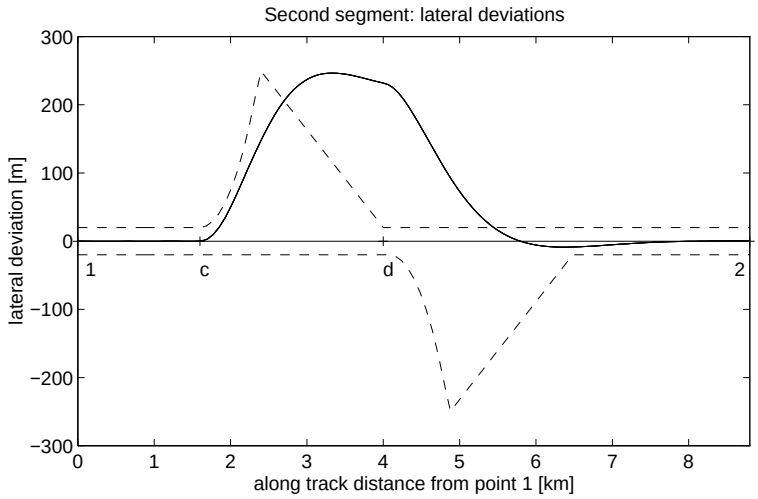


Figura 5.35: Segmento II: Efectos laterales durante un giro de 90º (vista detallada).

Las figuras 5.36 y 5.37 corresponden al segmento III (de 2 a 3 en la figura 5.32) y muestran el comportamiento del controlador durante la fase de descenso.

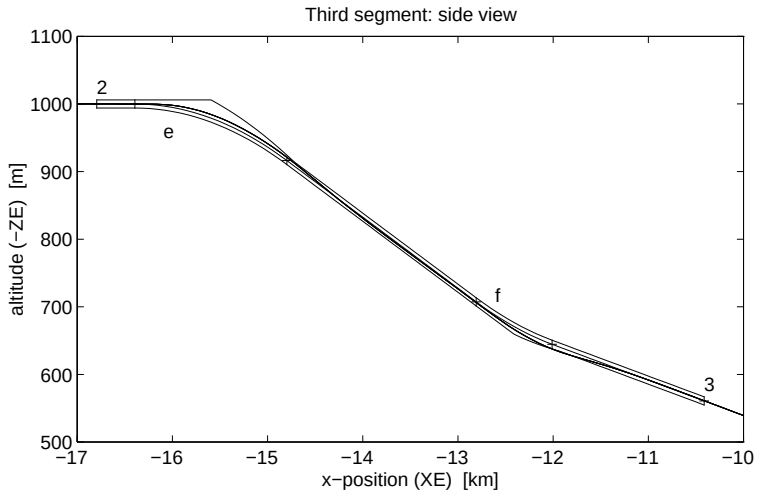


Figura 5.36: Segmento III: Fase de descenso (vista lateral).

En la figura 5.37, que presenta un mayor nivel de detalle, se puede ver que los límites son sobrepasados ligeramente, si bien en ningún caso la desviación vertical excede el valor de 20 m y al final del segmento la desviación es igual a cero. Al igual que ocurría en el segmento II, intentar mantener la desviación dentro de los límites traería consigo una disminución del confort. De nuevo se observa una buena robustez.

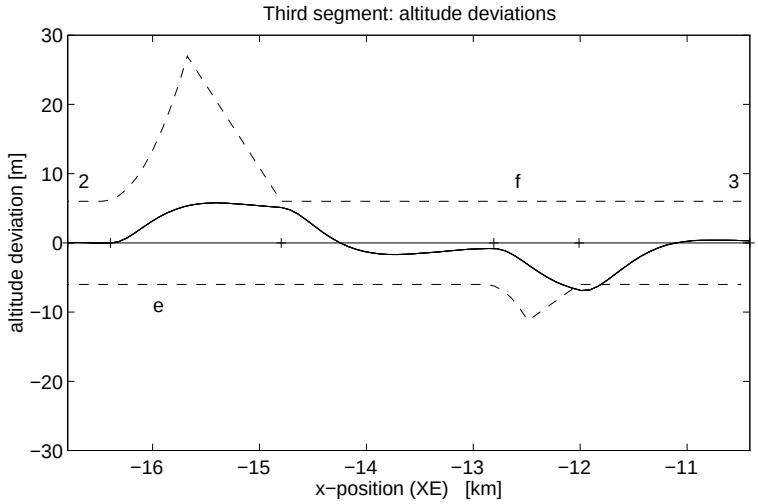


Figura 5.37: Segmento III: Fase de descenso (vista detallada).

Las figuras 5.38 y 5.39 corresponden al segmento IV (de 3 a 4 en la figura 5.32) y muestran el comportamiento del controlador durante la aproximación final.

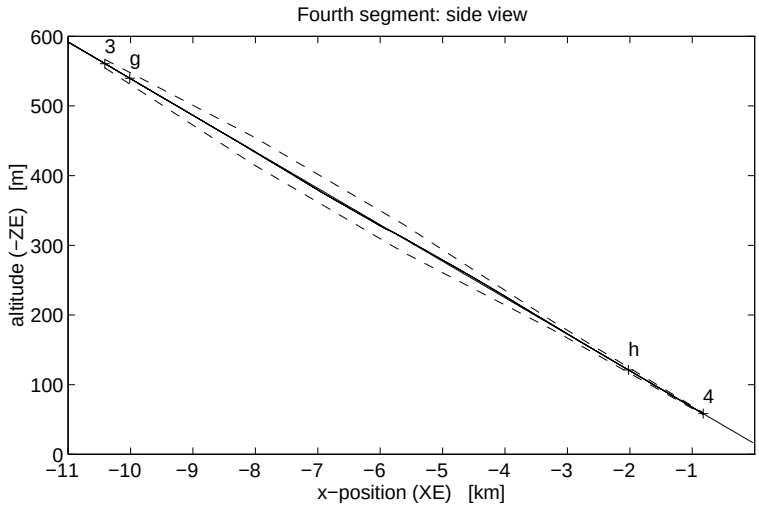


Figura 5.38: Segmento IV: Fase de aproximación final (vista lateral).

La figura 5.39 presenta un mayor nivel de detalle, y puede comprobarse que en todo momento se mantiene la trayectoria dentro de los límites establecidos.

La tabla 5.10 resume los resultados obtenidos por el controlador durante la maniobra de aproximación. Todos los valores numéricos, excepto los correspondientes a la potencia, están normalizados sobre los límites permitidos, de forma que valores menores

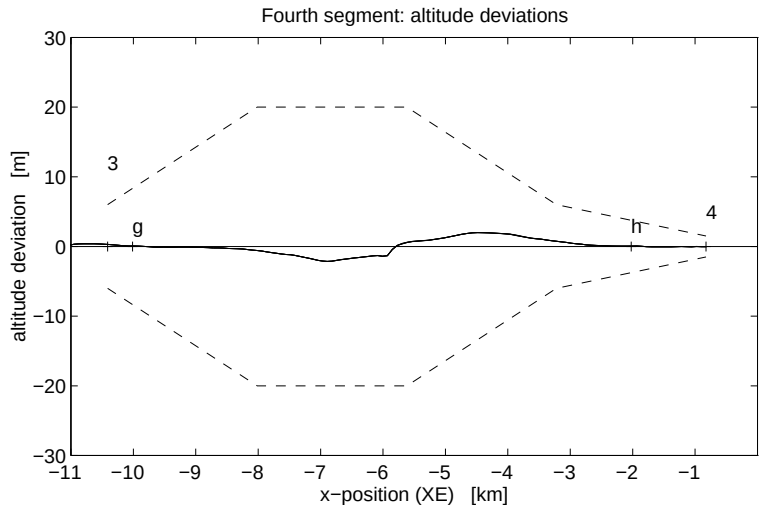


Figura 5.39: Segmento IV: Fase de aproximación final (vista detallada).

que la unidad son los permitidos.

En general los resultados son buenos, excepto para el criterio de confort en el segmento III, ya que los valores son menores que 1. Como ya se ha comentado, el problema del confort es debido a aceleraciones verticales ligeramente altas. Se pueden destacar los buenos resultados obtenidos para los criterios de robustez y seguridad.

	Segment I	Segment II	Segment III	Segment IV	Total
Performance	0.0647	0.6256	0.1530	0.0637	0.2267
Robustness	0.0140	0.0055	0.0107	0.0191	0.0123
Comfort	0.3738	0.7944	1.0926	0.4923	0.6883
Safety	0.0038	0.0150	0.0073	0.0122	0.0096
Power	0.0027	0.0025	0.0150	0.0308	0.0127

Tabla 5.10: Resultados numéricos del controlador completo.

5.5. Conclusiones.

El algoritmo evolutivo empleado, ha sido capaz de resolver completamente un problema de control multivariable. Ha tratado con múltiples objetivos, algunos de ellos contrapuestos, y ha sintonizado una estructura de controlador predefinida.

El problema ha podido afrontarse desde el punto de vista del usuario, tratándose directamente con las especificaciones de diseño: se ha trabajado con variables como el

máximo sobreimpulso, el tiempo de subida, el tiempo de asentamiento, etc.

Además, la función de evaluación del algoritmo ha demostrado su flexibilidad, permitiendo las incorporación de cualquier parámetro que pueda resultar de utilidad. En este caso se incluyeron medidas de robustez, dando lugar a un controlador robusto.

Finalmente el controlador fue evaluado con las herramientas proporcionadas en [55], obteniéndose unos muy buenos resultados. Cabe destacar los excelentes resultados obtenidos para los criterios de robustez y seguridad.

El algoritmo empleado es el mismo que el que se utilizó también con éxito en el capítulo 4, para la sintonía del controlador LQ, por lo que podría ser aplicable a la sintonía de otro tipo de controladores.

Frente a otros métodos utilizados en el diseño de controladores, presenta también como ventaja la facilidad de su aprendizaje, no requiriéndose demasiado tiempo para ser capaz de aplicarlo al diseño.

Capítulo 6

Selección de la estructura del controlador y su sintonía mediante algoritmos genéticos y evolutivos.

6.1. Introducción.

En el capítulo 4, se empleó un algoritmo evolutivo para facilitar la sintonía de una estructura de control fija mediante el método LQ, de forma que el controlador obtenido cumpliera las especificaciones de diseño. Luego, en el capítulo 5, se avanzó un paso más, siendo el algoritmo evolutivo el que realizaba directamente la sintonía de una estructura de control fija y elegida a priori, sin el apoyo de ningún otro método. En el presente trabajo, se pretende ir más lejos y dejar que sea un algoritmo genético el que se encargue de elegir un tipo de controlador y sintonizarlo.

Por tanto, se propone un nuevo método que realizará la tarea apuntada, es decir, seleccionará la estructura y orden del controlador, y realizará la sintonía de los parámetros del mismo. El algoritmo constará de dos lazos: un lazo externo se encargará de seleccionar la estructura del controlador, y otro interno realizará la sintonía de cada una de las estructuras obtenidas por el lazo externo. La estructura, no debe entenderse en este caso como la relación que hay entre las distintas entradas y salidas del controlador, ni la forma en que estarán interconectados los distintos bloques que formen el controla-

dor, pues esto será fijo; se refiere a la estructura interna de los bloques que compondrán el controlador, concretamente a los polos y ceros de sus funciones de transferencia.

El algoritmo, tratará de obtener un controlador que reduzca el índice de mareo en un buque de alta velocidad (el problema se expuso en 3.3). El controlador actuará sobre dos superficies de control (Flap y T-foil en figura 3.4) para compensar las aceleraciones verticales del barco debidas a los movimientos de *pitch* y *heave* provocados por el oleaje.

La selección de la estructura del controlador se hará mediante un proceso de *loop shaping*. En este proceso, se trabajará con unos bloques básicos de control, como las ganancias, los polos y ceros simples, los retardos y los adelantos, los ceros y polos de segundo orden y los bloques *Notch*. Este proceso se utiliza al diseñar controladores QFT [126], utilizando en este caso diagramas de Nichols para hacer el *loop shaping*. En Santander y Aranda [110] se utilizó QFT para obtener un controlador para el problema de *RCAM*, y en Aranda *et al* [9] y Díaz *et al* [40] para el problema de *CRIBAV*.

Se pretende sustituir el conocimiento heurístico del diseñador —necesario para interpretar los diagramas de Nichols— por un algoritmo que dé una medida de lo buena que es una estructura de control, tras encontrar su "mejor sintonía" posible.

Debido a los altos requerimientos de cálculo, el algoritmo fue implementado en C y paralelizado con la ayuda de la librería de funciones MPI. Una vez diseñado, se ejecutó en un superordenador con múltiples procesadores del Centro de Supercomputación Complutense (CSC) de la Universidad Complutense de Madrid (UCM).

6.2. La estructura del controlador.

En el apartado 3.3 se describió con detalle el problema de control del modelo *CRIBAV*. Como se expuso en dicho apartado, el problema consiste en reducir las aceleraciones verticales de un buque de alta velocidad mediante la actuación sobre dos superficies de control: una de ellas situada en la popa (Flap) y la otra en la proa (T-foil). En la figura 3.4 se mostraba un detalle de dichas superficies.

El modelo linealizado del barco, según se mostró en los esquemas de las figuras 3.5 y 3.6, tiene 3 entradas (las señales de entrada al Flap y T-foil, y las perturbaciones debidas al oleaje) y 2 salidas (movimientos de *heave* y *pitch*) que se intentarán reducir mediante el algoritmo de control.

La figura 6.1 representa la estructura del controlador que será utilizado. Se observa

que el sistema resultante tendrá 3 entradas (las mencionadas en el párrafo anterior) y 4 salidas (las 2 mencionadas en el párrafo anterior más Ref_Flap y Ref_Tfoil, que han sido añadidas para poder penalizar aquellos controladores que den lugar a saturaciones en los actuadores). También se puede observar que se ha separado la dinámica de olas.

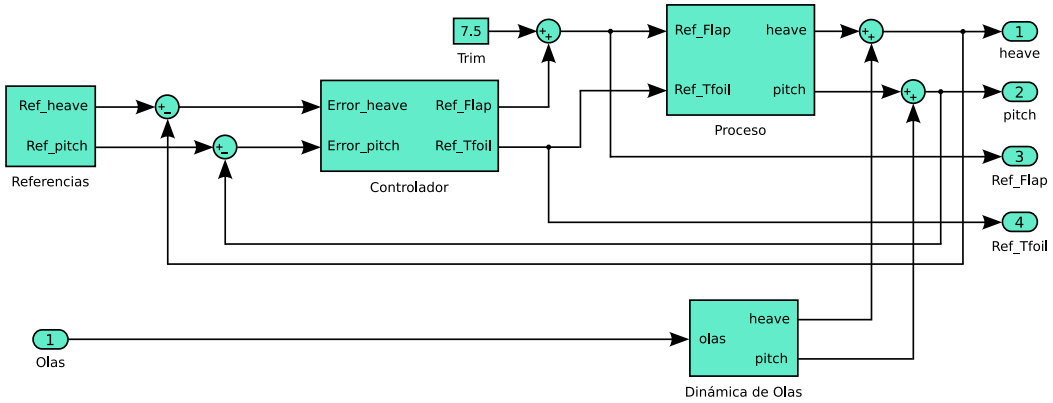


Figura 6.1: Estructura del controlador.

Las señales de referencia (Ref_heave y Ref_pitch) serán nulas, ya que lo ideal sería que no se produjeran movimientos de *heave* y *pitch*. Se trata por tanto de un problema de regulación, y las señales de entrada al controlador serán los valores de *heave* y *pitch* cambiados de signo.

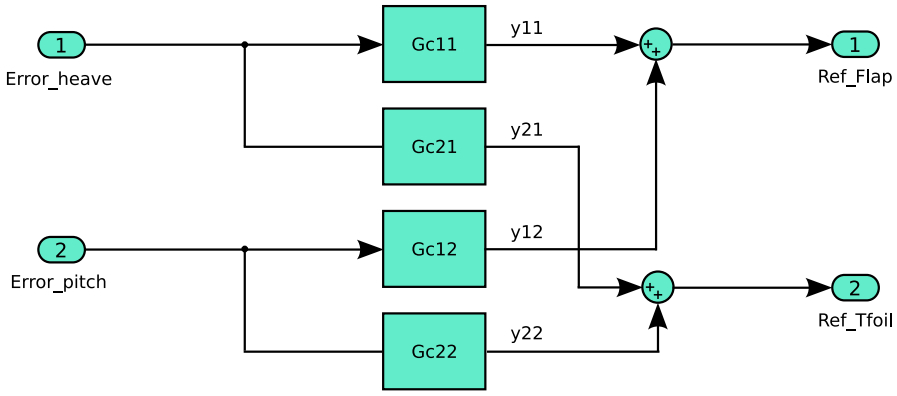


Figura 6.2: Diagrama de bloques del controlador.

La posición del T-foil deberá estar comprendida en el intervalo $[-15^o, 15^o]$, mientras que la posición del Flap deberá encontrarse en el intervalo $[0^o, 15^o]$. Para evitar excursiones hacia valores negativos de la posición del Flap, se establece un valor de 7.5^o como punto de *trimado*. Esta es la función del bloque denominado como *Trim* en la

figura 6.1.

La figura 6.2 muestra el diagrama de bloques del controlador. En él se detallan los 4 subbloques que lo componen: uno para cada par entrada-salida.

El algoritmo, que será explicado más adelante, obtendrá los distintos bloques del controlador en forma de funciones de transferencia, los convertirá al espacio de estados y los agrupará en un único sistema para realizar la simulación del experimento según se describe a continuación.

Sean las ecuaciones de los bloques G_{c11} y G_{c21} de la figura 6.2

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_{i1} & = & \boldsymbol{A}_{ci1} \cdot \boldsymbol{x}_{i1} + \boldsymbol{B}_{ci1} \cdot \text{Error\_heave} \\ \boldsymbol{y}_{i1} & = & \boldsymbol{C}_{ci1} \cdot \boldsymbol{x}_{i1} + D_{ci1} \cdot \text{Error\_heave} \end{array}\right\} \quad \forall i \in \{1,2\} \tag{6.1}$$

Y las de los bloques G_{c12} y G_{c22}

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}}_{i2} & = & \boldsymbol{A}_{ci2} \cdot \boldsymbol{x}_{i2} + \boldsymbol{B}_{ci2} \cdot \text{Error\_pitch} \\ \boldsymbol{y}_{i2} & = & \boldsymbol{C}_{ci2} \cdot \boldsymbol{x}_{i2} + D_{ci2} \cdot \text{Error\_pitch} \end{array}\right\} \quad \forall i \in \{1,2\} \tag{6.2}$$

A partir de la figura 6.2, las salidas Ref_Flap y Ref_Tfoil se pueden escribir en la forma

$$\begin{array}{rcl} \text{Ref\_Flap} & = & \boldsymbol{y}_{11} + \boldsymbol{y}_{12} = \boldsymbol{C}_{c11} \cdot \boldsymbol{x}_{11} + D_{c11} \cdot \text{Error\_heave} + \\ & & \boldsymbol{C}_{c12} \cdot \boldsymbol{x}_{12} + D_{c12} \cdot \text{Error\_pitch} \\ \text{Ref\_Tfoil} & = & \boldsymbol{y}_{21} + \boldsymbol{y}_{22} = \boldsymbol{C}_{c21} \cdot \boldsymbol{x}_{21} + D_{c21} \cdot \text{Error\_heave} + \\ & & \boldsymbol{C}_{c22} \cdot \boldsymbol{x}_{22} + D_{c22} \cdot \text{Error\_pitch} \end{array} \tag{6.3}$$

Y de las ecuaciones 6.1, 6.2 y 6.3 se obtienen las del controlador completo

$$
\begin{pmatrix} \dot{\boldsymbol{x}}_{11} \\ \dot{\boldsymbol{x}}_{21} \\ \dot{\boldsymbol{x}}_{12} \\ \dot{\boldsymbol{x}}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{c11} & 0 & 0 & 0 \\ 0 & \boldsymbol{A}_{c21} & 0 & 0 \\ 0 & 0 & \boldsymbol{A}_{c12} & 0 \\ 0 & 0 & 0 & \boldsymbol{A}_{c22} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_{11} \\ \boldsymbol{x}_{21} \\ \boldsymbol{x}_{12} \\ \boldsymbol{x}_{22} \end{pmatrix} + \begin{pmatrix} \boldsymbol{B}_{c11} & 0 \\ \boldsymbol{B}_{c21} & 0 \\ 0 & \boldsymbol{B}_{c12} \\ 0 & \boldsymbol{B}_{c22} \end{pmatrix} \cdot \begin{pmatrix} \text{Error\_heave} \\ \text{Error\_pitch} \end{pmatrix}
$$

$$
\begin{pmatrix} \text{Ref\_Flap} \\ \text{Ref\_Tfoil} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}_{c11} & 0 & \boldsymbol{C}_{c12} & 0 \\ 0 & \boldsymbol{C}_{c21} & 0 & \boldsymbol{C}_{c22} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_{11} \\ \boldsymbol{x}_{21} \\ \boldsymbol{x}_{12} \\ \boldsymbol{x}_{22} \end{pmatrix} + \begin{pmatrix} D_{c11} & D_{c12} \\ D_{c21} & D_{c22} \end{pmatrix} \cdot \begin{pmatrix} \text{Error\_heave} \\ \text{Error\_pitch} \end{pmatrix} \tag{6.4}
$$

Una vez obtenidas las ecuaciones del bloque controlador, se obtendrán las del sistema global. Dichas ecuaciones serán las utilizadas por la función de evaluación para obtener las aptitudes de los distintos controladores. Se procederá en la forma que se expone a continuación.

Sean las ecuaciones siguientes, las correspondientes a los bloques de la figura 6.1 denominados *Proceso*, *Controlador* y *Dinámica de Olas* respectivamente

$$
\left.\begin{aligned} \dot{\boldsymbol{x}}_p &= \boldsymbol{A}_p \cdot \boldsymbol{x}_p + \boldsymbol{B}_p \cdot \boldsymbol{u}_p \\ \boldsymbol{y}_p &= \boldsymbol{C}_p \cdot \boldsymbol{x}_p \end{aligned}\right\} \tag{6.5}
$$

$$
\left.\begin{aligned} \dot{\boldsymbol{x}}_c &= \boldsymbol{A}_c \cdot \boldsymbol{x}_c + \boldsymbol{B}_c \cdot \boldsymbol{u}_c \\ \boldsymbol{y}_c &= \boldsymbol{C}_c \cdot \boldsymbol{x}_c + \boldsymbol{D}_c \cdot \boldsymbol{u}_c \end{aligned}\right\} \tag{6.6}
$$

$$
\left.\begin{aligned} \dot{\boldsymbol{x}}_o &= \boldsymbol{A}_o \cdot \boldsymbol{x}_o + \boldsymbol{B}_o \cdot \text{olas} \\ \boldsymbol{y}_o &= \boldsymbol{C}_o \cdot \boldsymbol{x}_o + \boldsymbol{D}_o \cdot \text{olas} \end{aligned}\right\} \tag{6.7}
$$

De la figura 6.1 se obtiene

$$\boldsymbol{u}_c \;=\; -\boldsymbol{y}_p - \boldsymbol{y}_o = -\boldsymbol{C}_p \cdot \boldsymbol{x}_p - \boldsymbol{C}_o \cdot \boldsymbol{x}_o - \boldsymbol{D}_o \cdot \text{olas} \tag{6.8}$$

y

$$\begin{aligned} \boldsymbol{u}_p &= \text{trim} + \boldsymbol{y}_c = \text{trim} + \boldsymbol{C}_c \cdot \boldsymbol{x}_c + \boldsymbol{D}_c \cdot \boldsymbol{u}_c \\ &= \text{trim} + \boldsymbol{C}_c \cdot \boldsymbol{x}_c - \boldsymbol{D}_c \cdot \boldsymbol{C}_p \cdot \boldsymbol{x}_p - \boldsymbol{D}_c \cdot \boldsymbol{C}_o \cdot \boldsymbol{x}_o - \boldsymbol{D}_c \cdot \boldsymbol{D}_o \cdot \text{olas} \end{aligned} \tag{6.9}$$

donde trim $= \left(\begin{smallmatrix} 7.5 \\ 0 \end{smallmatrix}\right)$

Y a partir de las ecuaciones 6.5, 6.6, 6.7, 6.8 y 6.9, se obtiene

$$\begin{aligned} \begin{pmatrix} \dot{\boldsymbol{x}}_p \\ \dot{\boldsymbol{x}}_c \\ \dot{\boldsymbol{x}}_o \end{pmatrix} &= \begin{pmatrix} \boldsymbol{A}_p - \boldsymbol{B}_p\boldsymbol{D}_c\boldsymbol{C}_p & \boldsymbol{B}_p\boldsymbol{C}_c & -\boldsymbol{B}_p\boldsymbol{D}_c\boldsymbol{C}_o \\ -\boldsymbol{B}_c\boldsymbol{C}_p & \boldsymbol{A}_c & -\boldsymbol{B}_c\boldsymbol{C}_o \\ 0 & 0 & \boldsymbol{A}_o \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \\ \boldsymbol{x}_o \end{pmatrix} + \\ &\quad \begin{pmatrix} \boldsymbol{B}_p & -\boldsymbol{B}_p\boldsymbol{D}_c\boldsymbol{D}_o \\ 0 & -\boldsymbol{B}_c\boldsymbol{D}_o \\ 0 & \boldsymbol{B}_o \end{pmatrix} \cdot \begin{pmatrix} \text{trim} \\ \text{olas} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} \text{heave} \\ \text{pitch} \end{pmatrix} \\ \begin{pmatrix} \text{Ref\_Flap} \\ \text{Ref\_Tfoil} \end{pmatrix} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{C}_p & 0 & \boldsymbol{C}_o \\ -\boldsymbol{D}_c\boldsymbol{C}_p & \boldsymbol{C}_c & -\boldsymbol{D}_c\boldsymbol{C}_o \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_p \\ \boldsymbol{x}_c \\ \boldsymbol{x}_o \end{pmatrix} + \\ &\quad \begin{pmatrix} 0 & \boldsymbol{D}_o \\ \boldsymbol{I} & -\boldsymbol{D}_c\boldsymbol{D}_o \end{pmatrix} \cdot \begin{pmatrix} \text{trim} \\ \text{olas} \end{pmatrix} \end{aligned} \tag{6.10}$$

El algoritmo calculará las matrices del modelo de la ecuación 6.10 para cada controlador obtenido. Posteriormente, los utilizará en la simulación para determinar la aptitud de cada uno.

6.3. Diseño del algoritmo.

El algoritmo que se utilizará dispondrá de dos lazos: uno externo para obtener las estructuras del controlador y otro interno para sintonizarlo. El lazo interno será similar al que se diseñó en el capítulo 3, representado por el diagrama de flujo de la figura 3.14. Para el lazo externo se utilizará un algoritmo inspirado en la técnica conocida como *loop shaping*, que se describe a continuación.

6.3.1. Loop shaping.

La técnica de *loop shaping* consiste en generar un controlador, $G(s)$, que vaya modificando la forma de la función de transferencia de lazo abierto $L(s) = G(s) \cdot P(s)$, de manera que se satisfagan ciertas especificaciones de diseño. Las especificaciones pueden ser de muchos tipos, siendo la más importante que $L(s)$ satisfaga el criterio de estabilidad de Nyquist. Para construir el controlador, se trabaja con una serie de bloques básicos de control. Estos bloques son los siguientes:

1. Ganancia: k
2. Polo simple: $\frac{p}{s+p}$
3. Cero simple: $\frac{s+p}{p}$
4. Retardo o adelanto: $\frac{s+a}{s+b}$
5. Polo de 2º orden: $\frac{\omega^2}{s^2+2\xi\omega s+\omega^2}$
6. Cero de 2º orden: $\frac{s^2+2\xi\omega s+\omega^2}{\omega^2}$
7. Notch: $\frac{s^2+2\xi_1\omega s+\omega^2}{s^2+2\xi_2\omega s+\omega^2}$

El procedimiento consiste en ir añadiendo estos bloques a la función de transferencia del controlador, y comprobar su comportamiento mediante alguna técnica como puede ser el estudio sobre el diagrama de Nichols, tal como se hace en QFT.

El algoritmo que se va a diseñar, sustituirá los conocimientos que necesitaría el diseñador para interpretar los diagramas de Nichols o utilizar cualquier otra técnica, por la evaluación de los controladores mediante simulaciones sucesivas.

6.3.2. Algoritmo para la síntesis del controlador.

El lazo externo, como ya se ha comentado, se encargará de obtener el tipo de función de transferencia del controlador. Consistirá en un algoritmo genético, cuyos cromosomas estarán formados por unos y ceros. Un 1 en una posición concreta del cromosoma, indicará la presencia de un bloque básico de control determinado, mientras que un 0 indicará su ausencia.

> **Ejemplo 6.1** Supóngase que se van a utilizar los 7 bloques de control ya comentados, sin posibilidad de que se repita ninguno. Se necesitaría entonces un cromosoma con 7 genes: uno para cada bloque básico de control. El cromosoma siguiente:
>
> 1 1 0 0 1 0 0
>
> indicaría la presencia en la función de transferencia de una ganancia, un polo simple y un polo de 2º orden.

Una vez que el lazo externo obtiene el tipo de función de transferencia, el lazo interno se encargará de su sintonía. En el ejemplo 6.1 habría que determinar el valor k de la ganancia, el valor p del polo simple y los valores ξ y ω del polo de segundo orden. En este caso, el número de genes de los cromosomas del lazo interno (formados por números reales) será 4; este número variará en función del tipo de función de transferencia.

Para cada controlador, se construirá el modelo en el espacio de estados correspondiente al sistema representado en la figura 6.1, y se realizará la simulación con las muestras de olas disponibles, obteniéndose la aptitud del controlador.

La figura 6.3 muestra el diagrama de flujo simplificado del algoritmo completo. Se observa que el lazo interno se encargará de evaluar las estructuras de controlador que vaya obteniendo el lazo externo. Una vez evaluadas, sus aptitudes serán devueltas al lazo externo, y éste ordenará la población en función de dichas aptitudes. Si se dan las condiciones de finalización, se concluye la ejecución y la estructura con mejor aptitud

de las obtenidas hasta ese momento, será escogida como solución al problema; si no, se obtendrá una nueva generación de estructuras de controlador.

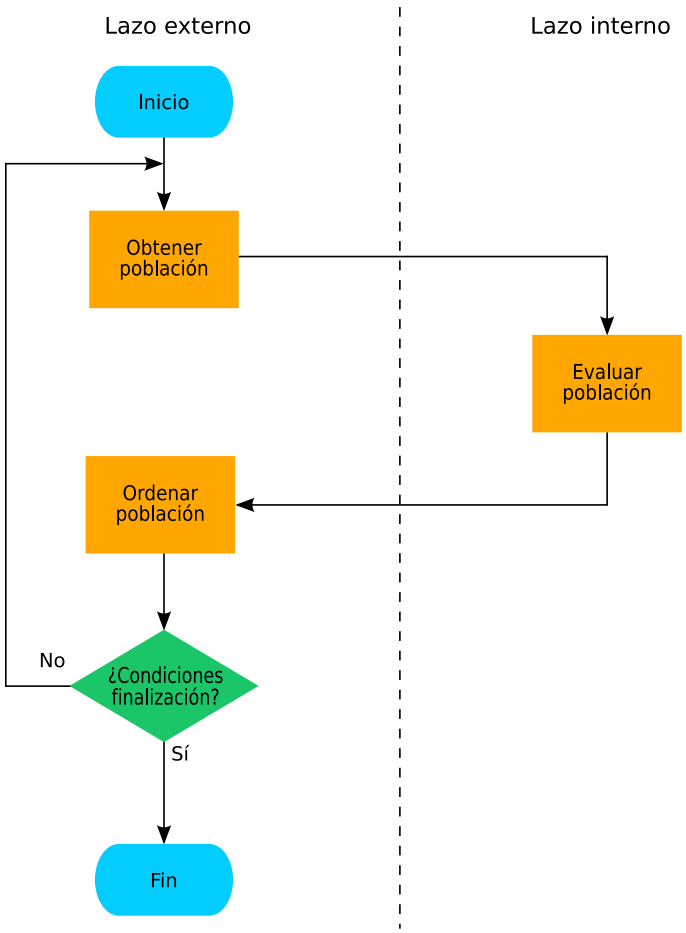


Figura 6.3: Diagrama de flujo del algoritmo completo.

La figura 6.4 muestra el diagrama de flujo correspondiente a la evaluación de un cromosoma obtenido por el lazo externo, por parte del lazo interno. Dicho diagrama está simplificado, pues el algoritmo empleado para el lazo interno es idéntico al que se diseñó en el capítulo 3 (representado en la figura 3.14). El procedimiento deberá repetirse para todos los cromosomas que componen la población.

Se observa en la figura 6.4 que el lazo interno irá obteniendo nuevas generaciones de cromosomas hasta que se cumplan las condiciones de finalización. Los cromosomas, en este caso, deberán codificar los distintos parámetros a sintonizar de la estructura de controlador que se esté evaluando. Su longitud, por tanto, variará dependiendo del número de parámetros de la estructura. Las condiciones de finalización del lazo interno

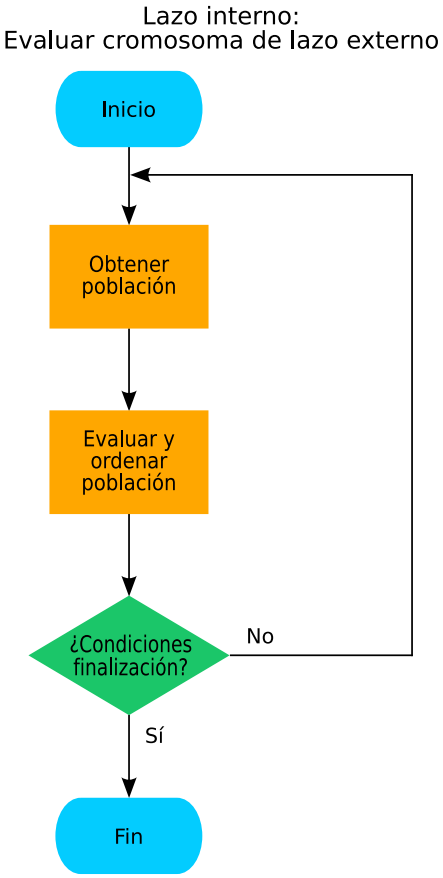


Figura 6.4: Diagrama de flujo correspondiente a la evaluación de un cromosoma de lazo externo.

se reducen a un número fijo de generaciones a obtener, pues no es posible tener la interactividad que se tendrá con el lazo externo.

Los objetivos del problema, ya fueron expuestos en el apartado 3.4.3.2, y la forma de proceder con los vectores de aptitud para llevar a cabo la ordenación de las poblaciones, será idéntica a la que se expuso en los capítulos anteriores, por lo que no se volverá a explicar aquí.

6.3.2.1. Paralelización del algoritmo.

Debido al alto coste computacional requerido por el algoritmo, éste fue implementado en C y paralelizado con la ayuda de la librería de funciones especializadas MPI [100]. Tras su diseño, fue ejecutado en un ordenador *Silicon Graphics Origin 2000* del

CSC de la Universidad Complutense de Madrid.

Descripción del hardware y su modo de uso. Se utilizó el ordenador del CSC denominado *seymour.csc.ucm.es*, que era un *Silicon Graphics Origin 2000* con 32 procesadores MIPS R10000 a 250 MHz y sistema operativo Irix.

En todo momento se trabajó en modo remoto, mediante *telnet* y *ftp*, utilizando el sistema de colas disponible: el trabajo a ejecutar se dejaba en una cola, y el sistema operativo gestionaba dicha cola y lo hacía pasar a ejecución cuando le llegara el turno. Una vez concluida la ejecución, el sistema operativo mandaba un mensaje de correo, avisando de este hecho, pudiéndose recoger los resultados por *ftp*.

El código fuente debía colocarse en *seymour*, y compilarse y enlazarse para convertirlo en código objeto compatible con la máquina. Al ejecutarlo había que indicar cuántos procesadores se deseaban utilizar. En realidad, lo único que se podía imponer era el número de procesos distintos en los que debía correr el programa. Éstos sólo corresponderían a un número equivalente de procesadores cuando el control de carga ejercido por *seymour* lo permitiera.

Para mandar un trabajo a la cola, se utilizó el comando `qsub`, con un *script* adjunto en el que figuraban las órdenes de trabajo que se deseaban realizar. Si, por ejemplo, el *script* tuviera por nombre *miscript*, el comando para enviar el trabajo a la cola sería:

```
qsub miscript
```

En la figura 6.5 se muestra el código de un *script* de ejemplo.

```
# QSUB -q batch15d30pe
# QSUB -mb
# QSUB -me
# QSUB -mu mparrilla@dia.uned.es

setenv MP_SET_NUMTHREADS 30
cd /user/users/parrilla
./ejemplo.exe
```

Figura 6.5: Ejemplo de *script* adjunto a qsub.

La primera línea del *script* indica la cola a la que se enviará el trabajo. Existían distintas colas disponibles, en función del tiempo de ejecución y el número de proce-

sadores. La cola elegida en el presente ejemplo tiene un tiempo máximo de ejecución por proceso de 15 días y el número de procesadores es 30. La segunda y tercera líneas requieren que se mande un mensaje de correo cuando comience y termine la ejecución respectivamente, y la cuarta indica la dirección de correo a la que deberán enviarse dichos mensajes. La siguiente línea establece el valor de la variable `MP_SET_NUMTHREADS`, que define el número de procesadores solicitado; es muy importante, pues en caso de no incluirla el programa se ejecutaría en un único procesador. Por último se establece como actual el directorio en el que se encuentra el archivo del programa y se ejecuta.

Descripción del proceso de paralelización. Como ya se ha comentado, el algoritmo fue implementado en C, utilizando MPI, que es una librería de funciones especializadas en la paralelización de programas, mediante el paso de mensajes entre los distintos procesos.

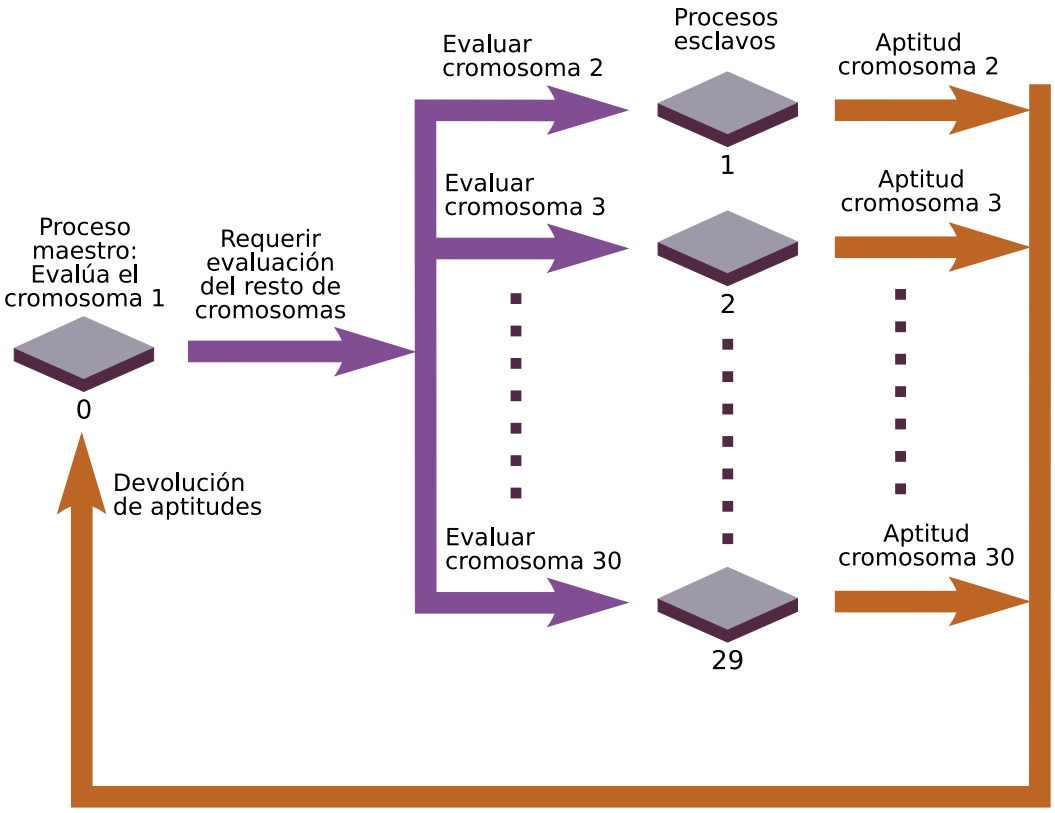


Figura 6.6: Distribución de tareas Maestro-Esclavos.

Para el lazo externo, se utilizaron poblaciones de 30 cromosomas. No se buscaba un algoritmo optimizado en cuanto al consumo de recursos por lo que se llevó a cabo

la paralelización por la técnica conocida como *fuerza bruta*. Siguiendo dicha técnica, se trabajó con un proceso maestro y 29 procesos esclavos. El proceso maestro llevaba el control de la ejecución del programa y se encargaba de la evaluación del primer cromosoma. Además, encargaba la evaluación de cada uno de los cromosomas restantes de la población a un proceso esclavo. Los esclavos, una vez evaluado el cromosoma que les correspondía, devolvían los resultados al maestro para que pudiera gestionarlos convenientemente. El procedimiento seguido se encuentra resumido en la figura 6.6.

Estructura del programa con MPI. Para poder hacer uso de las funciones de la librería MPI, el programa se debe ajustar a la plantilla siguiente:

```
#include <mpi.h>
/* Se añadirá también el resto de archivos de cabecera */

main(int argc, char **argv)
{
   /* Se inicializa MPI: */
   MPI_Init(&argc, &argv);

   /* Aquí irá el resto del programa. */

   /* Se concluye MPI: */
   MPI_Finalize();

   /* Se abandona el programa: */
   exit (0);
}
```

En la primera línea se incluye el archivo de cabecera de la librería MPI. Luego, todas las llamadas a las funciones de la librería deberán ir entre las instrucciones `MPI_Init(&argc, &argv)` y `MPI_Finalize()`, que inicializan y concluyen MPI.

El programa se ejecutará con el comando `mpirun`, que es el lanzador primario de aplicaciones en la implementación de MPI para Silicon Graphics. A dicho comando habrá que pasarle como argumentos el nombre del archivo del programa y el número de procesos en el que deberá ejecutarse. Este comando deberá incluirse en el *script* adjunto al comando `qsub`, que como ya se comentó sirve para enviar el trabajo a la cola de trabajos. En el *script* de ejemplo de la figura 6.5 se sustituiría la última línea por la siguiente:

```
mpirun -np 30 nombre_del_programa
```

El mismo programa será ejecutado por todos los procesos, por lo que en el código se deberán incluir una serie de condicionales para controlar qué procesos realizan las distintas tareas. Cada proceso lleva asociado un número de identificación único, comprendido entre cero y el número de tareas menos uno, que se utilizará en una sentencia `if` para determinar las tareas que deberá realizar. El número de identificación de cada tarea se obtiene mediante la sentencia

```
MPI_Comm_rank(MPI_COMM_WORLD, &mi_id);
```

quedando almacenado en la variable `mi_id`. Se escogerá como proceso maestro a aquel cuyo número de identificación sea cero.

Las tareas a realizar por el proceso maestro estarán incluidas en un bloque como el siguiente:

```
if (mi_id == 0)
{
   /* Aquí irá el código a ejecutar por el proceso maestro */
}
```

Por su parte, las tareas a realizar por los procesos esclavos estarán incluidas en un bloque como el que se muestra a continuación:

```
if (mi_id != 0)
{
   /* Aquí irá el código a ejecutar por los procesos esclavos */
}
```

La comunicación entre el proceso maestro y los procesos esclavos se lleva a cabo mediante las funciones: `MPI_Bcast`, `MPI_Send` y `MPI_Recv`. El proceso maestro enviará a los procesos esclavos todos los datos necesarios para que lleven a cabo la evaluación del cromosoma que les corresponda; dicho envío se realizará mediante la función `MPI_Bcast`, que permite el envío simultáneo a todos los procesos esclavos con una única instrucción. Estos datos serán recibidos por los esclavos mediante la función `MPI_Recv`. Los esclavos, una vez hayan evaluado su cromosoma, devolverán al maestro los datos resultantes de la evaluación mediante la función `MPI_Send`, y el maestro por su parte los recibirá haciendo uso de la función `MPI_Recv`.

Tras la recepción de las aptitudes, el maestro se encargará de realizar todas las tareas necesarias para la obtención de una nueva generación de cromosomas. Todo este procedimiento se repetirá hasta que se den las condiciones de finalización.

6.4. Resultados.

El programa, una vez diseñado, fue ejecutado para tres condiciones de velocidad y estado de la mar diferentes: velocidad 20 nudos y estado de la mar 4, velocidad 40 nudos y estado de la mar 4, y velocidad 40 nudos y estado de la mar 5. Dependiendo de la velocidad, se tomó para cada caso el modelo linealizado correspondiente, de los que se encuentran recogidos en el apéndice B, además se utilizó un histórico de olas para cada estado de la mar, proporcionado dentro del proyecto *CRIBAV*. Los controladores obtenidos se evaluaron también con *mibav*, que es un entorno de simulación en *Simulink*, desarrollado por Esteban en su tesis doctoral [47], con el fin de evaluar de una forma homogénea distintos tipos de controladores.

La figura 6.7 representa la estructura interna del controlador tal como fue introducida en el modelo de *mibav*. El algoritmo se encargará de obtener las funciones de transferencia de los distintos bloques que la componen. Dicha estructura interna será la misma para todas las condiciones de velocidad y estado de la mar.

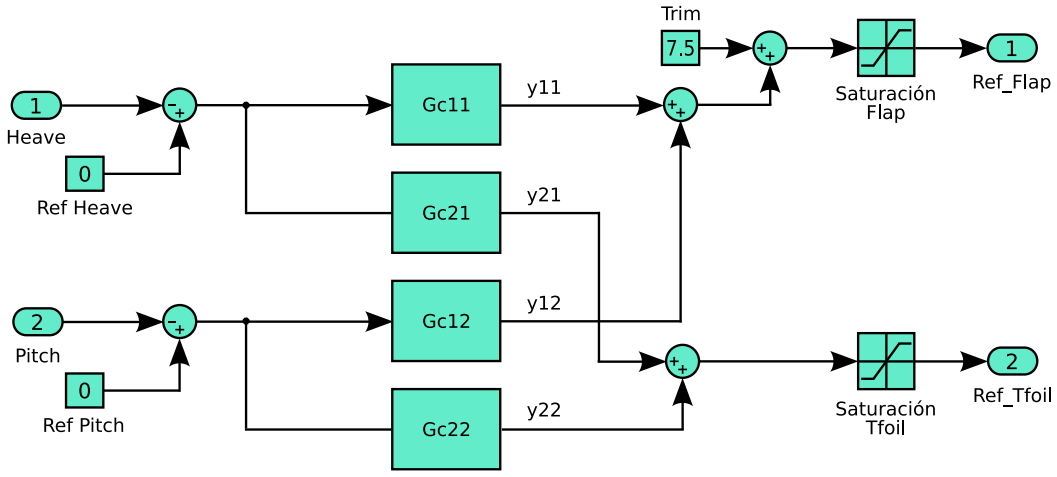


Figura 6.7: Estructura interna del controlador.

6.4.1. Resultados para velocidad de 20 nudos y estado de la mar 4.

La figura 6.8(a) muestra las funciones de transferencia del controlador obtenido para las condiciones de velocidad 20 nudos y estado de la mar 4. Durante la ejecución del algoritmo se obtuvieron 200 generaciones de lazo externo, dándose el mejor controlador para la generación 176, para el cual la aceleración media fue de 0.50404 m/s^2. La figura 6.8(b) muestra la evolución de la mejor aceleración media durante las 200 generaciones.

$$\frac{1.7398 \cdot 10^6 s^3 + 9.888 \cdot 10^8 s^2 + 3.6478 \cdot 10^{10} s + 7.0738 \cdot 10^{12}}{s^5 + 6.9235 \cdot 10^3 s^4 + 9.239 \cdot 10^6 s^3 + 1.3666 \cdot 10^{10} s^2 + 1.5025 \cdot 10^{12} s + 9.465 \cdot 10^{13}}$$

Gc_{11}

$$\frac{4.7567 \cdot 10^5 s^2 + 5.1398 \cdot 10^9 s + 8.8756 \cdot 10^{12}}{s^3 + 5.7616 \cdot 10^4 s^2 + 1.0695 \cdot 10^9 s + 5.6359 \cdot 10^{12}}$$

Gc_{21}

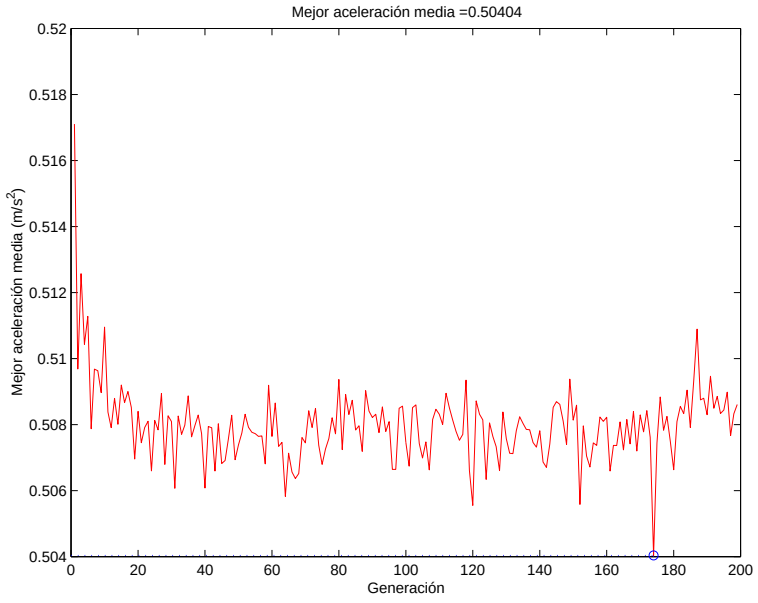
$$\frac{1.5051 \cdot 10^7 s^2 + 1.132 \cdot 10^{12} s + 4.6381 \cdot 10^{16}}{3.0815 \cdot 10^9 s^2 + 2.3619 \cdot 10^{13} s + 4.6381 \cdot 10^{16}}$$

Gc_{12}

$$\frac{1.9715 \cdot 10^{10} s^5 + 7.7159 \cdot 10^{13} s^4 + 2.6794 \cdot 10^{14} s^3 + 3.1702 \cdot 10^{14} s^2 + 3.3126 \cdot 10^{14} s + 2.6956 \cdot 10^{14}}{s^6 + 1.994 \cdot 10^4 s^5 + 1.0259 \cdot 10^8 s^4 + 3.3694 \cdot 10^{10} s^3 + 2.1266 \cdot 10^{13} s^2 + 2.9066 \cdot 10^{12} s + 2.6175 \cdot 10^{13}}$$

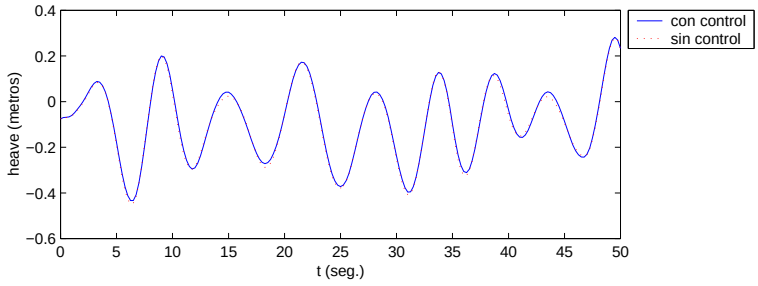
Gc_{22}

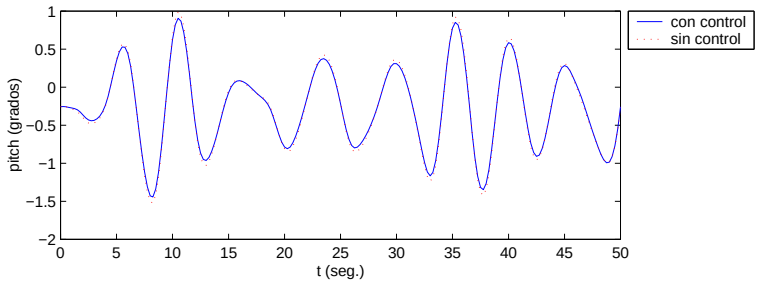
(a)



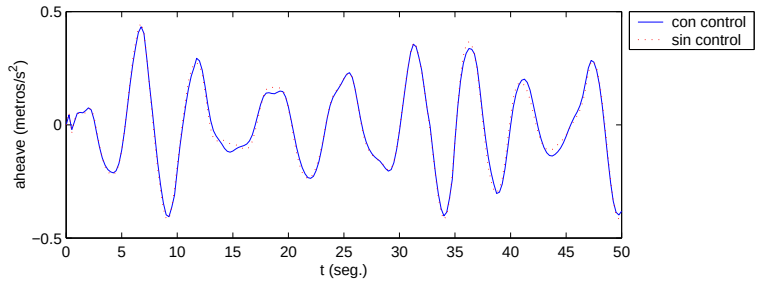
(b)

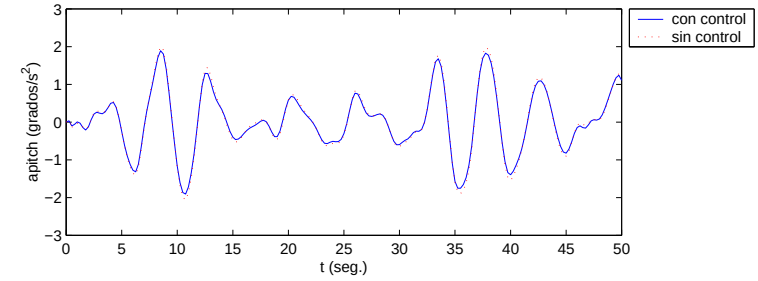
Figura 6.8: (a) Funciones de transferencia del controlador obtenido, (b) Evolución de la mejor aceleración media.





(a)

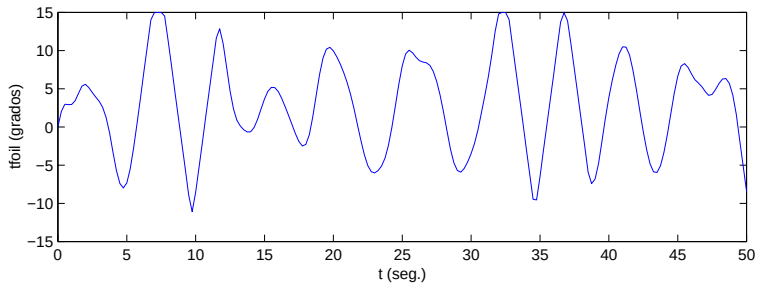


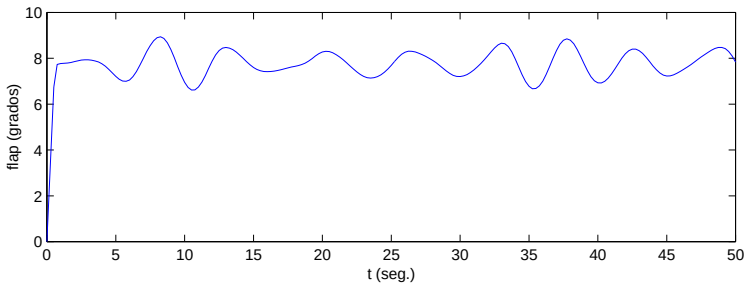


(b)

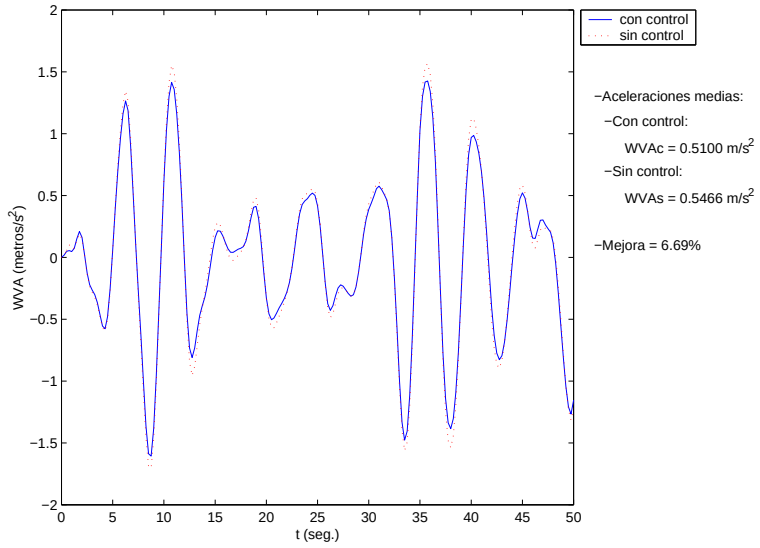
Figura 6.9: (a) Salidas del sistema, (b) Aceleraciones en *heave* y *pitch*.

La figura 6.9(a) muestra la evolución de las salidas durante el experimento, mientras que la figura 6.9(b) muestra la evolución de las aceleraciones en *heave* y *pitch*. En ambos casos se muestran los resultados con y sin acción de control.





(a)



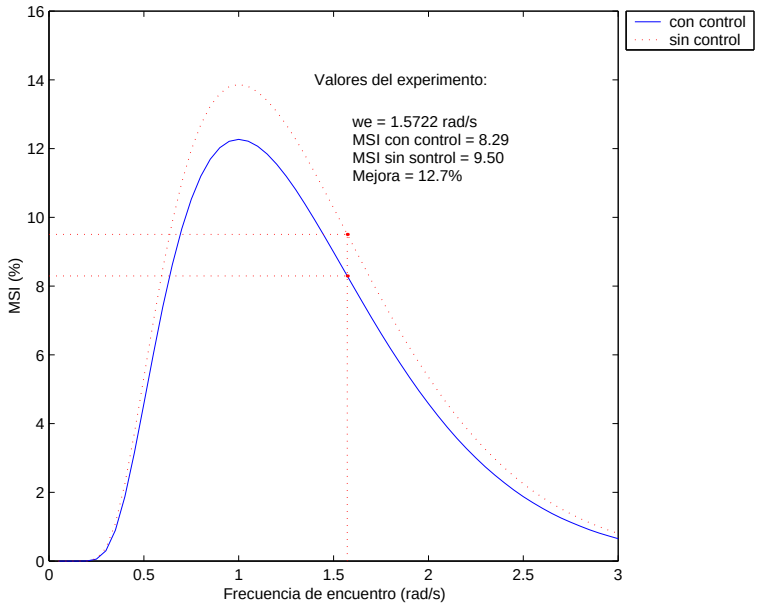
(b)

Figura 6.10: (a) Posiciones de las superficies de control, (b) Evolución de la aceleración total y cálculo de aceleraciones medias.

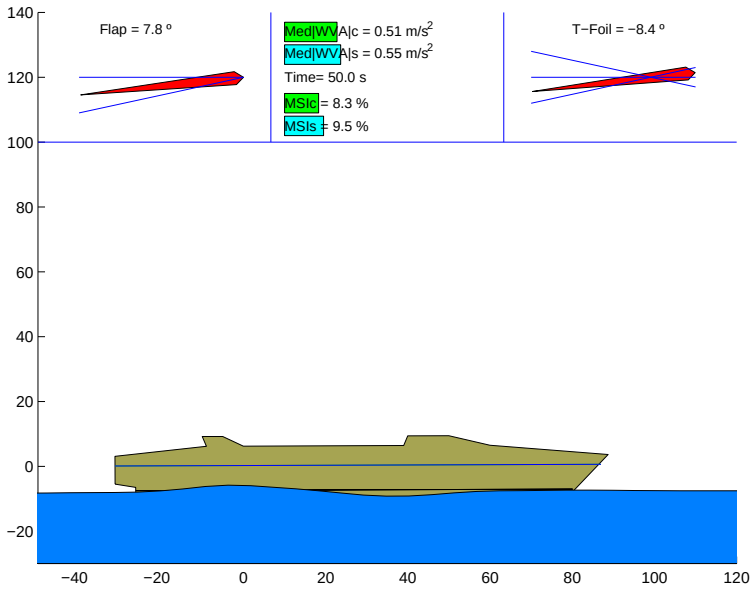
La figura 6.10(a) muestra la evolución de las posiciones de las superficies de control durante el experimento. La figura 6.10(b) muestra la evolución de la aceleración total y el cálculo de las aceleraciones medias, con y sin acción de control. Se observa una mejora del 6.69 %.

La figura 6.11(a) muestra el índice de mareo (MSI), con controlador y sin acción de control, para un rango de frecuencias de encuentro, detallándose además para la frecuencia de encuentro del experimento (1.5722 rad/s). Se observa que la acción de

control produce una mejora del 12.7 %. La figura 6.11(b) muestra el estado final de la ventana de *mibav* tras 50 segundos de simulación. La aceleración media pasa de 0.55 m/s^2 a 0.51 m/s^2 y el índice de mareo pasa de 9.5 % a 8.3 %.



(a)



(b)

Figura 6.11: (a) MSI para un rango de frecuencias de encuentro y cálculos para la frecuencia de encuentro del experimento (1.5722 rad/s), (b) Resultado de evaluación en mibav.

6.4.2. Resultados para velocidad de 40 nudos y estado de la mar 4.

$$\frac{82.118s^5 + 4.8096 \cdot 10^6 s^4 + 4.5367 \cdot 10^{11} s^3 + 9.9049 \cdot 10^{15} s^2 + 1.106 \cdot 10^{20} s + 1.0109 \cdot 10^{23}}{4.3786 \cdot 10^9 s^6 + 1.7906 \cdot 10^{14} s^5 + 2.6196 \cdot 10^{18} s^4 + 1.4536 \cdot 10^{22} s^3 + 2.6808 \cdot 10^{25} s^2 + 7.6257 \cdot 10^{23} s + 1.1827 \cdot 10^{23}}$$

Gc_{11}

$$\frac{6.6628 \cdot 10^{-4} s^3 + 31.551 s^2 + 1.008 \cdot 10^6 s + 6.5725 \cdot 10^8}{s^6 + 7.7857 \cdot 10^4 s^5 + 2.4421 \cdot 10^9 s^4 + 2.4438 \cdot 10^{13} s^3 + 7.5332 \cdot 10^{16} s^2 + 5.6647 \cdot 10^{13} s + 5.0192 \cdot 10^{13}}$$

Gc_{21}

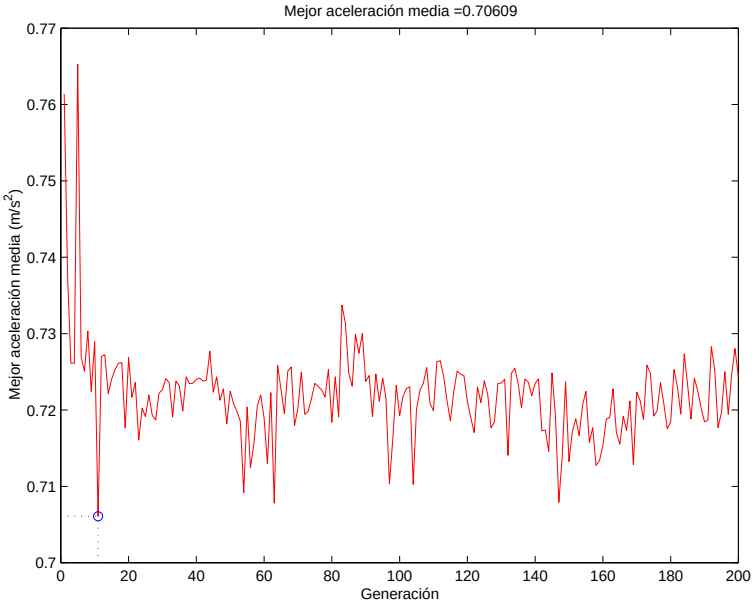
$$\frac{s^5 + 1.2395 \cdot 10^4 s^4 + 3.6353 \cdot 10^7 s^3 + 4.4525 \cdot 10^9 s^2 + 2.552 \cdot 10^{12} s + 2.157 \cdot 10^{14}}{s^5 + 1.3098 \cdot 10^4 s^4 + 3.3756 \cdot 10^7 s^3 + 1.6526 \cdot 10^{10} s^2 + 2.6398 \cdot 10^{12} s + 1.003 \cdot 10^{14}}$$

Gc_{12}

$$\frac{7.3937 \cdot 10^4 s^3 + 7.0585 \cdot 10^8 s^2 + 9.347 \cdot 10^8 s + 1.8513 \cdot 10^9}{2.6232 s^3 + 2.0071 \cdot 10^3 s^2 + 5.4481 \cdot 10^5 s + 9.5974 \cdot 10^7}$$

Gc_{22}

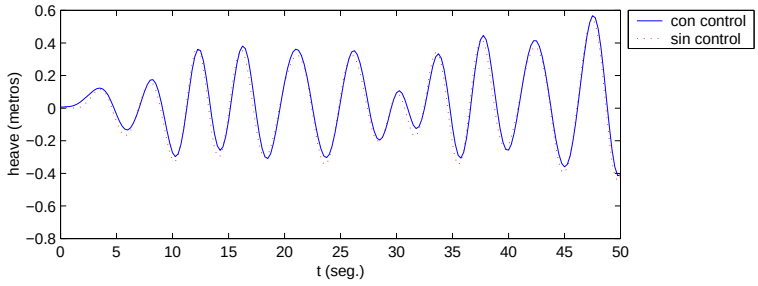
(a)

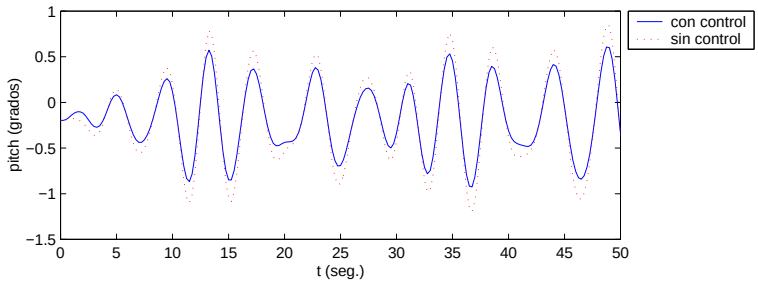


(b)

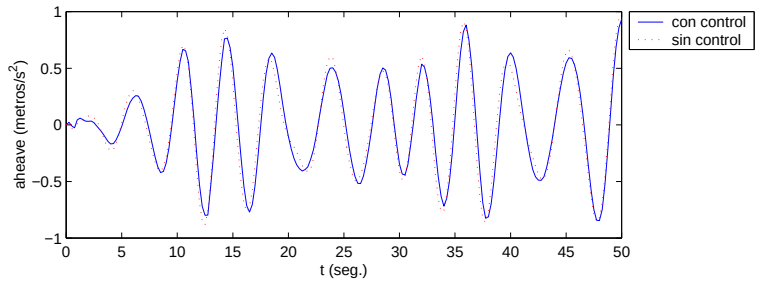
Figura 6.12: (a) Funciones de transferencia del controlador obtenido, (b) Evolución de la mejor aceleración media.

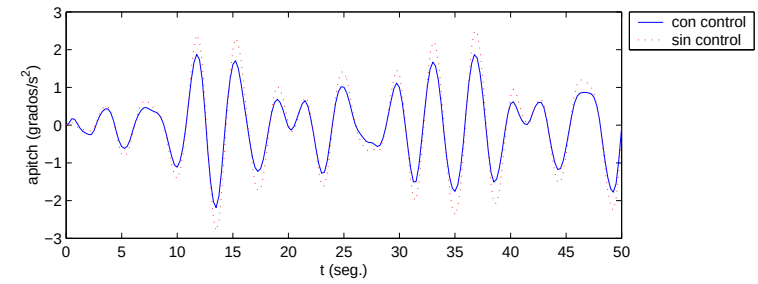
La figura 6.12(a) muestra las funciones de transferencia del controlador obtenido para las condiciones de velocidad 40 nudos y estado de la mar 4. Durante la ejecución del algoritmo se obtuvieron 200 generaciones de lazo externo, obteniéndose una mejor aceleración media de 0.70609 m/s^2. La figura 6.12(b) muestra la evolución de la mejor aceleración media durante las 200 generaciones.





(a)

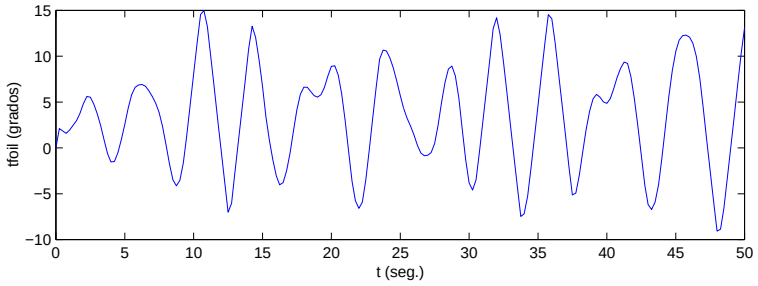


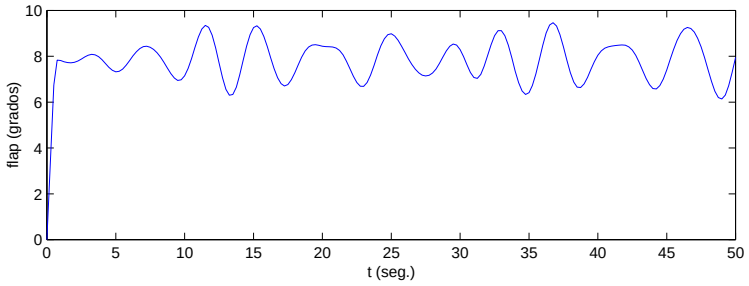


(b)

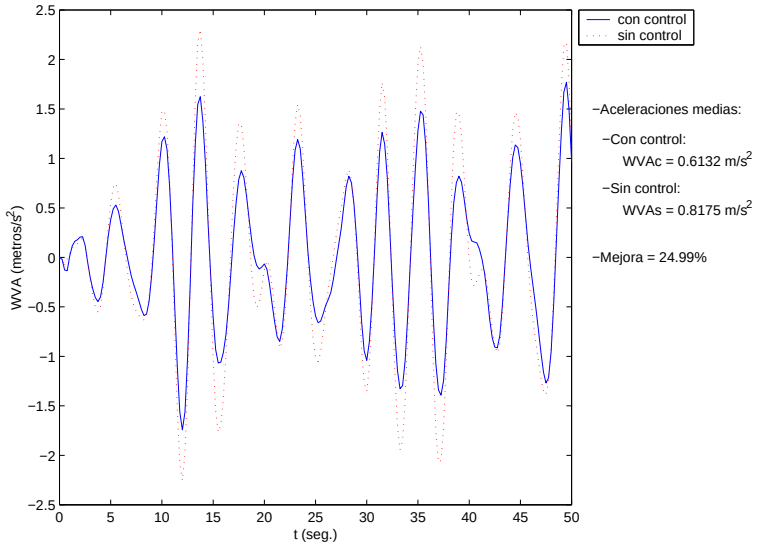
Figura 6.13: (a) Salidas del sistema, (b) Aceleraciones en *heave* y *pitch*.

La figura 6.13(a) muestra la evolución de las salidas durante el experimento, mientras que la figura 6.13(b) muestra la evolución de las aceleraciones en *heave* y *pitch*. En ambos casos se muestran los resultados con y sin acción de control.





(a)



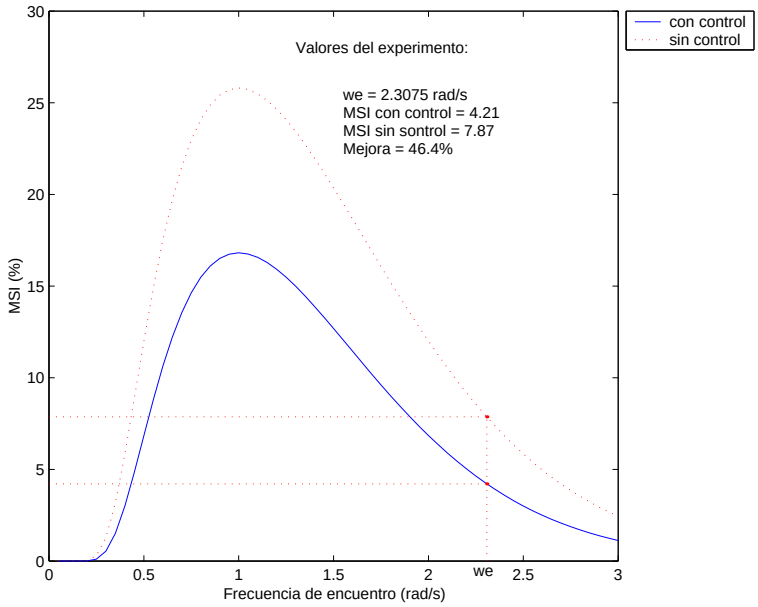
(b)

Figura 6.14: (a) Posiciones de las superficies de control, (b) Evolución de la aceleración total y cálculo de aceleraciones medias.

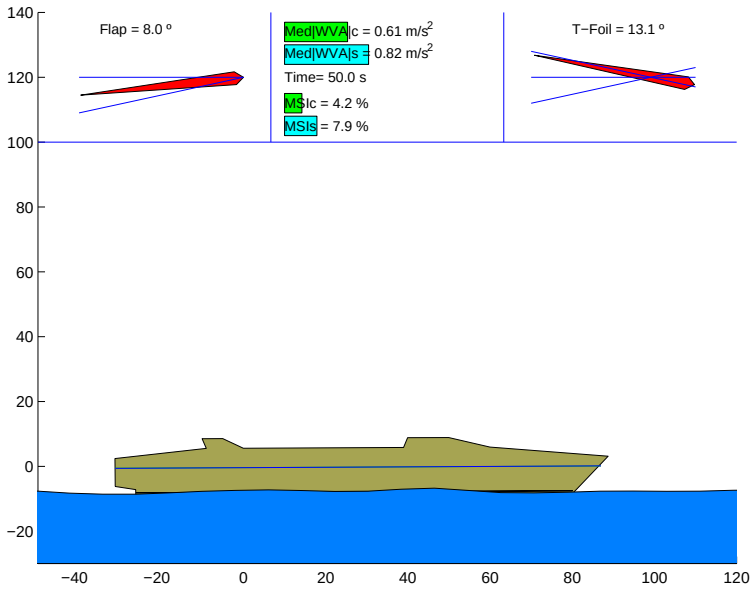
La figura 6.14(a) muestra la evolución de las posiciones de las superficies de control durante el experimento. La figura 6.14(b) muestra la evolución de la aceleración total y el cálculo de las aceleraciones medias, con y sin acción de control. Se observa una mejora del 24.99 %.

La figura 6.15(a) muestra el índice de mareo (MSI), con controlador y sin acción de control, para un rango de frecuencias de encuentro, detallándose además para la frecuencia de encuentro del experimento (2.3075 rad/s). Se observa que la acción de

control produce una mejora del 46.4 %. La figura 6.15(b) muestra el estado final de la ventana de *mibav* tras 50 segundos de simulación. La aceleración media pasa de 0.82 m/s^2 a 0.61 m/s^2 y el índice de mareo pasa de 7.9 % a 4.2 %.



(a)



(b)

Figura 6.15: (a) MSI para un rango de frecuencias de encuentro y cálculos para la frecuencia de encuentro del experimento (2.3075 rad/s), (b) Resultado de evaluación en mibav.

6.4.3. Resultados para velocidad de 40 nudos y estado de la mar 5.

$$\frac{35.008 \cdot s^5 + 2.7719 \cdot 10^5 s^4 + 1.1574 \cdot 10^{11} s^3 + 5.9726 \cdot 10^{14} s^2 + 1.0779 \cdot 10^{19} s + 4.8659 \cdot 10^{22}}{9.3747 \cdot 10^7 s^6 + 7.0514 \cdot 10^{12} s^5 + 3.802 \cdot 10^{17} s^4 + 4.1932 \cdot 10^{21} s^3 + 1.2962 \cdot 10^{25} s^2 + 2.4261 \cdot 10^{23} s + 8.7203 \cdot 10^{21}}$$

Gc_{11}

$$\frac{23.091 \cdot s^2 + 8.1307 \cdot 10^4 s + 3.4818 \cdot 10^7}{s^5 + 1.7428 \cdot 10^4 s^4 + 9.5213 \cdot 10^7 s^3 + 1.627 \cdot 10^{11} s^2 + 2.3569 \cdot 10^{10} s + 8.9988 \cdot 10^9}$$

Gc_{21}

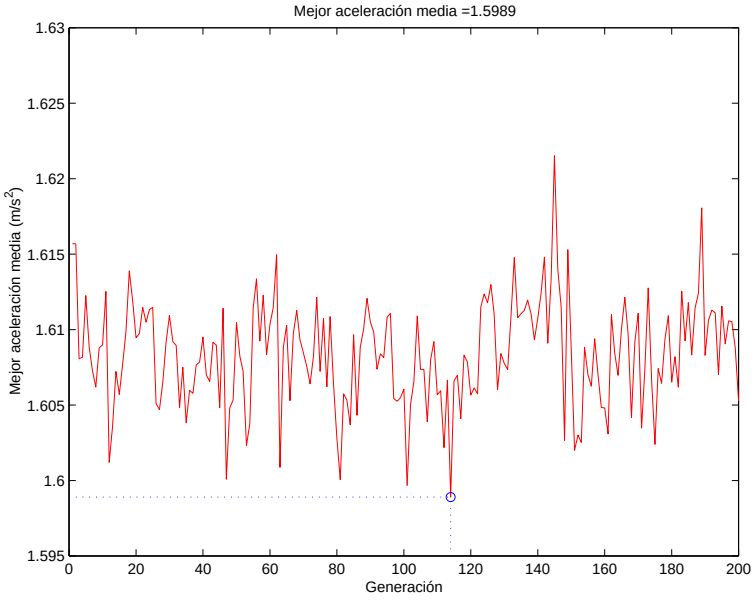
$$\frac{4.0879 \cdot 10^3 s^4 + 3.5481 \cdot 10^7 s^3 + 2.8094 \cdot 10^8 s^2 + 7.0362 \cdot 10^{10} s + 2.2024 \cdot 10^{10}}{s^4 + 9.7205 \cdot 10^3 s^3 + 1.7562 \cdot 10^7 s^2 + 1.0598 \cdot 10^9 s + 3.3638 \cdot 10^{10}}$$

Gc_{12}

$$\frac{4.694 \cdot 10^{11} s^2 + 2.0681 \cdot 10^{12} s + 1.9155 \cdot 10^{12}}{s^3 + 1.3363 \cdot 10^4 s^2 + 1.2192 \cdot 10^8 s + 3.9542 \cdot 10^{11}}$$

Gc_{22}

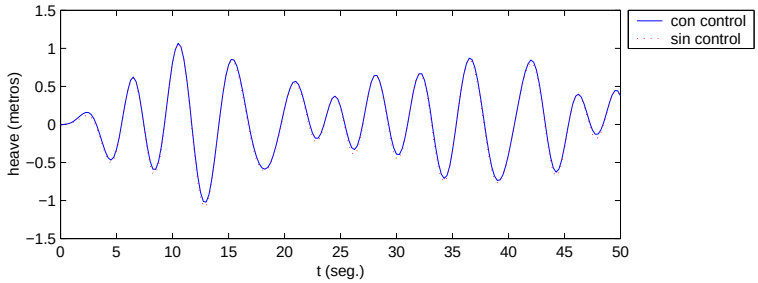
(a)

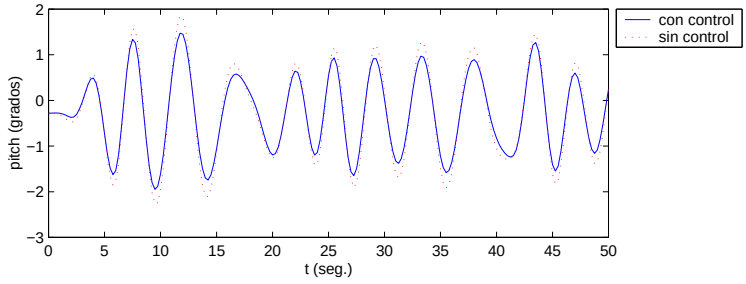


(b)

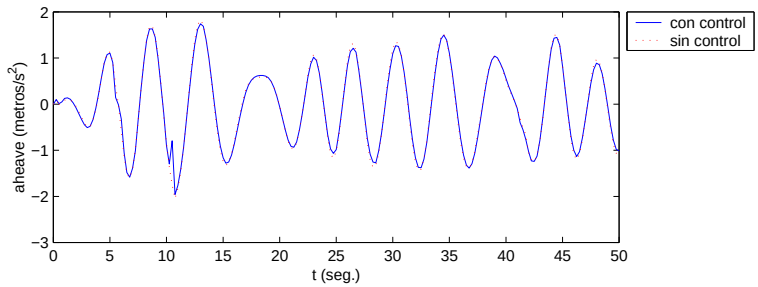
Figura 6.16: (a) Funciones de transferencia del controlador obtenido, (b) Evolución de la mejor aceleración media.

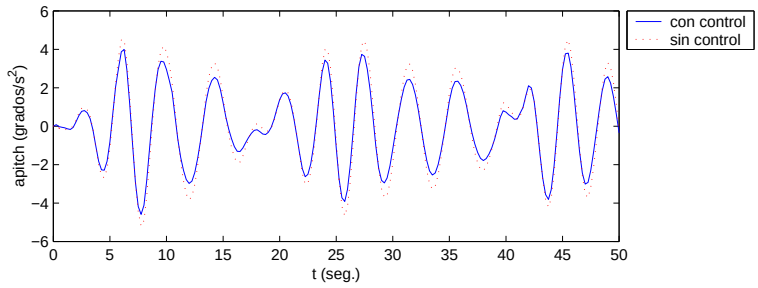
La figura 6.16(a) muestra las funciones de transferencia del controlador obtenido para las condiciones de velocidad 40 nudos y estado de la mar 5. Durante la ejecución del algoritmo se obtuvieron 200 generaciones de lazo externo, obteniéndose una mejor aceleración media de 1.5989 m/s^2. La figura 6.16(b) muestra la evolución de la mejor aceleración media durante las 200 generaciones.





(a)

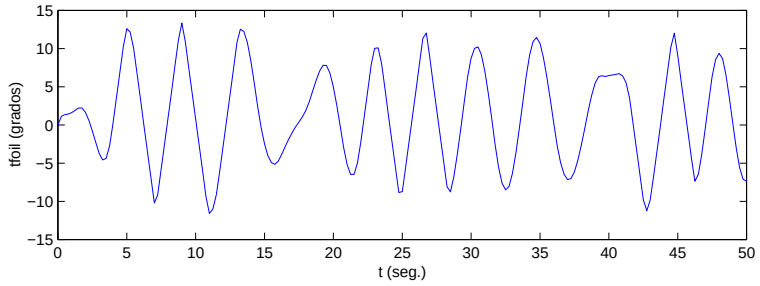


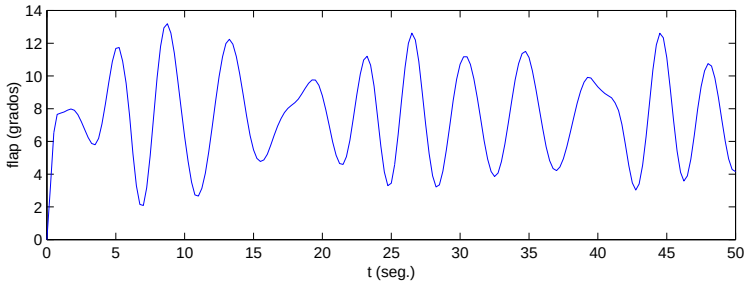


(b)

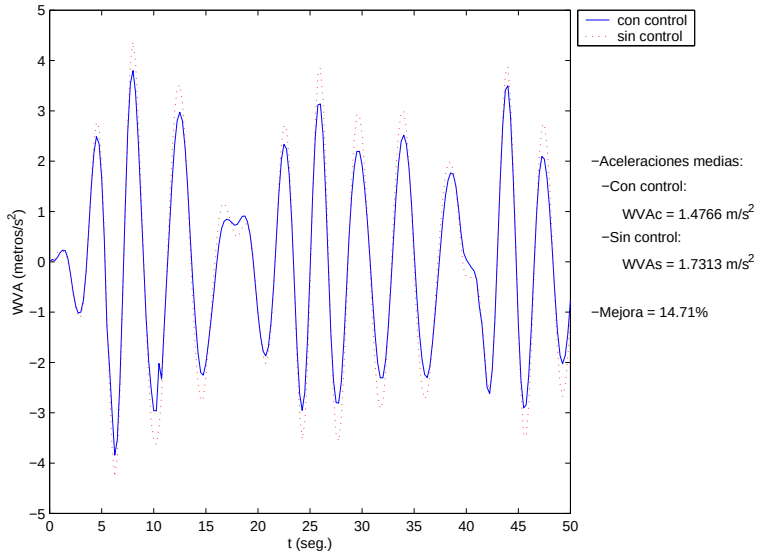
Figura 6.17: (a) Salidas del sistema, (b) Aceleraciones en *heave* y *pitch*.

La figura 6.17(a) muestra la evolución de las salidas durante el experimento, mientras que la figura 6.17(b) muestra la evolución de las aceleraciones en *heave* y *pitch*. En ambos casos se muestran los resultados con y sin acción de control.





(a)



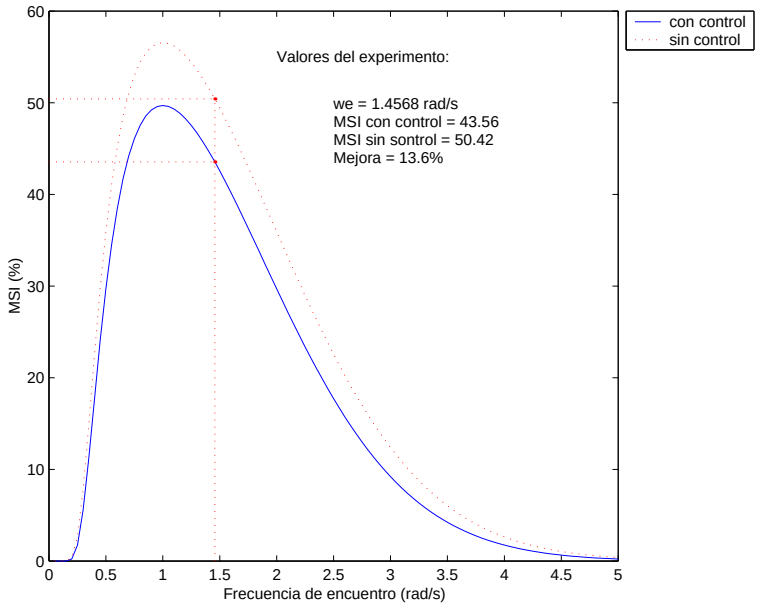
(b)

Figura 6.18: (a) Posiciones de las superficies de control, (b) Evolución de la aceleración total y cálculo de aceleraciones medias.

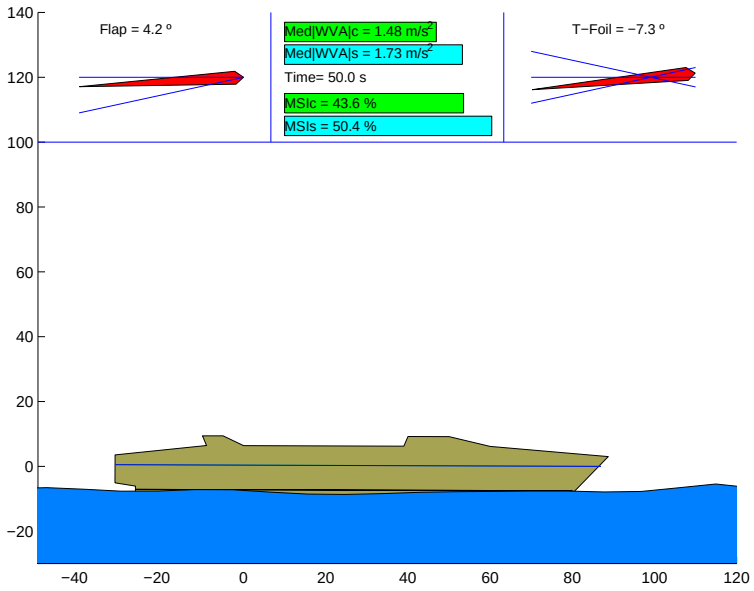
La figura 6.18(a) muestra la evolución de las posiciones de las superficies de control durante el experimento. La figura 6.18(b) muestra la evolución de la aceleración total y el cálculo de las aceleraciones medias, con y sin acción de control. Se observa una mejora del 14.71 %.

La figura 6.19(a) muestra el índice de mareo (MSI), con controlador y sin acción de control, para un rango de frecuencias de encuentro, detallándose además para la frecuencia de encuentro del experimento (1.4568 rad/s). Se observa que la acción de

control produce una mejora del 13.6 %. La figura 6.19(b) muestra el estado final de la ventana de *mibav* tras 50 segundos de simulación. La aceleración media pasa de 1.73 m/s^2 a 1.48 m/s^2 y el índice de mareo pasa de 50.4 % a 43.6 %.



(a)



(b)

Figura 6.19: (a) MSI para un rango de frecuencias de encuentro y cálculos para la frecuencia de encuentro del experimento (1.4568 rad/s), (b) Resultado de evaluación en mibav.

6.5. Conclusiones.

En el presente capítulo se ha vuelto a mostrar la utilidad de los algoritmos genéticos y evolutivos en el diseño de controladores. En este caso se han utilizado para reducir las aceleraciones verticales y por consiguiente el índice de mareo en un barco.

El método que se ha propuesto y desarrollado en el presente capítulo, ha sido capaz de obtener las estructuras de los bloques del controlador para distintas condiciones de velocidad y estado de la mar, y sintonizarlas. Una vez implementado el algoritmo, su aplicación a los diferentes modelos correspondientes a distintas condiciones de linealización es trivial.

El trabajo expuesto en el presente capítulo, en el momento de su realización, se desarrolló en el límite de lo posible con la tecnología disponible. Por ello se desestimaron algunos objetivos que hubieran hecho dispararse el coste computacional.

El ordenador utilizado, el más potente de los accesibles en el momento en que se realizaron los experimentos, era del año 1999, por lo que sus prestaciones han sido ampliamente superadas con la aparición en el mercado de nuevas máquinas más potentes. Es por esto que en el año 2005 el Centro de Supercomputación Complutense (CSC) de la Universidad Complutense de Madrid (UCM) ha sustituido la máquina *seymour* por un servidor Altix 3700 Bx2 con 64 procesadores Itanium II 1,6Ghz 6MB L3 (con un sistema cache denominado SSI) y 128 GB de memoria. La aparición de máquinas como la anteriormente descrita (y mucho más con las que irán apareciendo en el futuro), hace posible la ejecución de algoritmos más ambiciosos. Se podría abordar el problema del presente capítulo trabajando con poblaciones de lazo externo más grandes y aumentar el número de generaciones a obtener, hasta que se produjera una clara convergencia. También se podría incluir como objetivo alguna medida sobre la robustez del controlador, como se hizo en los problemas tratados en los capítulos 4 y 5. Todos estos aspectos que se comentan, no fueron contemplados en el presente experimento con el fin de ahorrar coste computacional.

La aparición de ordenadores más potentes, también hará posible la aplicación del presente algoritmo a la solución de problemas de control más complejos, en los que la evaluación del cumplimento de sus especificaciones de diseño implica un gran consumo de recursos computacionales, como ocurre por ejemplo, con el del controlador del avión RCAM. Téngase en cuenta que en los resultados que se dieron en el capítulo 5, el tiempo medio empleado en la sintonía de una estructura de controlador fue de 3h 15' para el modo longitudinal, y de 1h 45' para el modo lateral, utilizando un procesador AMD

XP 2300. Con estos tiempos, no parece viable la evaluación de múltiples generaciones de estructuras.

Capítulo 7

Arquitecturas para la paralelización y ejecución de algoritmos genéticos y evolutivos.

7.1. Introducción.

En el capítulo 6, se resolvió un problema de control complejo mediante la paralelización de un algoritmo genético. El programa, una vez implementado, se ejecutó en un superordenador del Centro de Supercomputación Complutense (CSC) de la Universidad Complutense de Madrid (UCM).

En aquella ocasión, no se hizo hincapié en la optimización del algoritmo paralelizado, utilizándose la técnica conocida como *fuerza bruta*. En el presente capítulo, se va a realizar un estudio de optimización para la paralelización del lazo interno del algoritmo que se empleó en el capítulo 6, pero aplicado ahora al problema del controlador del avión. Al paralelizar el lazo interno se conseguirá un mejor aprovechamiento de los procesadores, al disminuirse los tiempos muertos en los que algunos de ellos debían esperar hasta que los otros concluyeran su trabajo.

Antes de hacer esto, se repasará la situación actual de los supercomputadores, y se presentarán también los *clusters* de PC's como una alternativa a dichos superordenadores. Un *clúster* es una agrupación de múltiples ordenadores interconectados por red, para aprovechar la potencia de cálculo de todos sus procesadores. Existen desde *clusters* de superordenadores, hasta *clusters* de PC's.

Tras repasar las distintas arquitecturas, se comenzará el estudio de optimización ya comentado, utilizando para ello un clúster de PC's del Departamento de Informática y Automática de la E.T.S. de Ingeniería Informática de la UNED. El objetivo perseguido en este caso no será obtener unos buenos resultados para la sintonía del controlador (los resultados serán similares a los que se obtuvieron en el capítulo 5), sino maximizar la aceleración conseguida con la utilización de múltiples PC's.

7.2. Superordenadores y clusters de PC's.

7.2.1. Superordenadores.

Los superordenadores son aquellos ordenadores que presentan, en el momento de su fabricación, unas capacidades computacionales muy por encima a las comúnmente disponibles. Dada su potencia, son empleados en la solución de problemas muy complejos que requieren enormes cantidades de cálculo. Entre sus aplicaciones se pueden mencionar: predicciones meteorológicas, seguimiento de huracanes, dinámica de fluidos, simulaciones de explosiones nucleares, ingeniería genética, estudio de modelos económicos a escala nacional y mundial, y un largo etcétera, pues tienen aplicación en cualquier campo de la ciencia y la tecnología. También tienen utilidad, como se ha demostrado en el capítulo 6, en la resolución de problemas de control complejos mediante el uso de algoritmos evolutivos.

La potencia de los superordenadores se mide por el número de operaciones en coma flotante por segundo, FLOPS (en inglés, **FL**oating-point **O**perations **P**er **S**econd). Habitualmente se utilizan sus múltiplos: MEGA FLOPS (MFlops, 10^6 Flops), GIGA FLOPS (GFlops, 10^9 Flops), TERA FLOPS (TFlops, 10^{12} Flops) y PETA FLOPS (PFlops, 10^{15} Flops).

Tradicionalmente, los superordenadores han obtenido sus mayores velocidades respecto a los ordenadores convencionales mediante la utilización de diseños innovadores. Suelen estar especializados en un tipo determinado de computación, normalmente cálculo numérico, comportándose pobremente en tareas más generales. La memoria del sistema se diseña cuidadosamente, para asegurar que los procesadores disponen en todo momento del conjunto de datos e instrucciones necesarios.

7.2.1.1. Tipos de arquitectura.

La arquitectura del hardware determina en gran medida las posibilidades del sistema. A continuación se describen las cuatro clases principales de arquitecturas de ordenador según la clasificación de Flynn [51], que pese a su antigüedad sigue siendo un referente válido. La clasificación está basada en la forma en la que se manejan los datos y las instrucciones.

- Máquinas **SISD** (Single Instruction/Single Data stream): Consiste en un único procesador recibiendo un flujo de instrucciones individuales que operan sobre un único flujo de datos. En cada paso, la unidad de control emite una instrucción que opera sobre un dato obtenido de la unidad de memoria (véase la figura 7.1). Los algoritmos que se ejecutan en máquinas SISD se llaman secuenciales (o en serie), ya que no contienen ninguna paralelización.

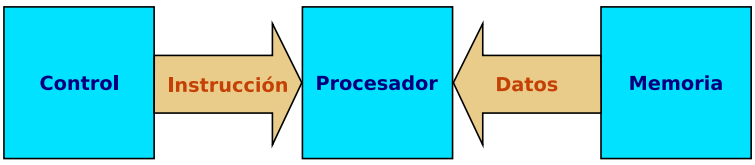


Figura 7.1: SISD.

  Actualmente, muchos grandes servidores cuentan con más de una CPU, aunque la ejecución de los distintos flujos de instrucciones no están relacionados. Por ello, estos sistemas se podrían considerar como un grupo de máquinas SISD actuando sobre conjuntos de datos diferentes. Como ejemplos de máquinas SISD se pueden mencionar la mayoría de las estaciones de trabajo de DEC, Hewlett-Packard, IBM y SGI. Se han incluido en la relación con el único propósito de ser completos.

- Máquinas **SIMD** (Single Instruction/Multiple Data stream): Consiste en N procesadores bajo un control común, que les proporciona la misma instrucción a todos ellos. Hay además N flujos de datos, uno por cada procesador. Cada ciclo de reloj, todos los procesadores ejecutan una misma instrucción sobre un elemento diferente de datos (véase la figura 7.2).

  Numerosos problemas pueden resolverse por algoritmos paralelos sobre ordenadores SIMD y los algoritmos son relativamente fáciles de diseñar, analizar e implementar. Sin embargo, sólo podrán abordarse con ordenadores SIMD aquellos problemas que puedan subdividirse en un conjunto de subproblemas idénticos y

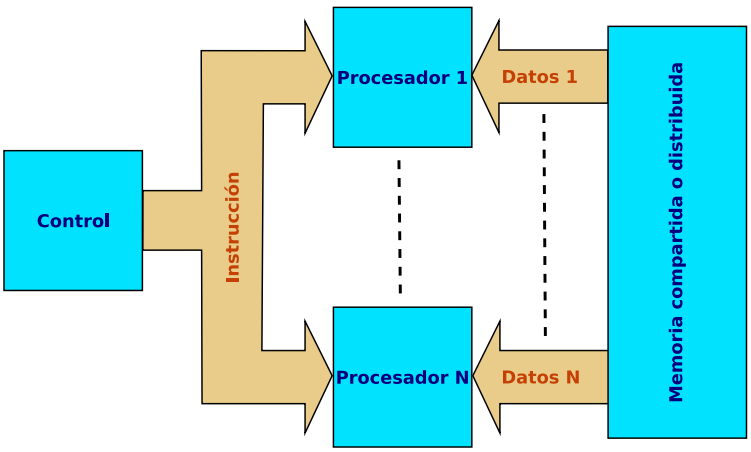


Figura 7.2: SIMD.

resolverse por un mismo conjunto de instrucciones ejecutándose simultáneamente. Hay muchos problemas que no se ajustan a este patrón: esos problemas se subdividirán típicamente en subproblemas no necesariamente idénticos, y se resolverán utilizando ordenadores MIMD.

Estos sistemas tienen normalmente un número grande de unidades de proceso, entre 1024 y 16384, ejecutando la misma instrucción sobre datos diferentes. Por tanto, una única instrucción manipula muchos datos en paralelo.

Una subclase de sistemas SIMD son los procesadores vectoriales. Los procesadores vectoriales actúan sobre arrays de datos similares en lugar de sobre elementos individuales de datos, utilizando CPUs con una estructura especial. Cuando los datos son manipulados por estas unidades vectoriales, los resultados son entregados con una tasa de uno, dos y – en casos especiales – tres por ciclo de reloj. Por tanto, los procesadores vectoriales tienen un funcionamiento casi en paralelo, cuando funcionan en modo vectorial. En este caso son varias veces más rápidos que cuando funcionan en el modo escalar convencional. A efectos prácticos los procesadores vectoriales se consideran, por tanto, máquinas SIMD.

- Máquinas **MISD** (Multiple Instructions/Single Data stream): Consisten en N procesadores, cada uno de ellos con su propia unidad de control, compartiendo una unidad de memoria común. Para cada ciclo de reloj, un dato recibido de la memoria se procesa de forma simultánea en todos los procesadores: cada procesador ejecuta la instrucción recibida desde su unidad de control (véase la figura 7.3). La paralelización se consigue al hacer a la vez cosas diferentes sobre los mismos datos. El tipo de computación que puede llevarse a cabo de forma eficiente en

máquinas MISD es bastante especializado: p. ej. problemas de clasificación.

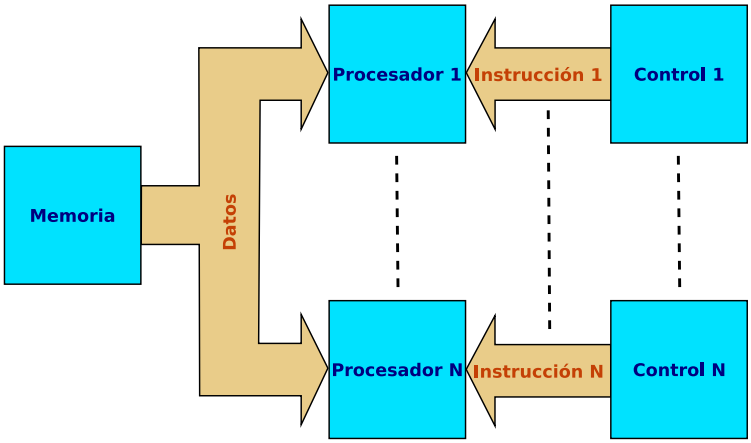


Figura 7.3: MISD.

- Máquinas **MIMD** (Multiple Instructions/Multiple Data stream): Esta clase de máquinas es la más general y la más poderosa de todas las contenidas en la clasificación de Flynn. En este caso hay N procesadores, N flujos de instrucciones y N flujos de datos. Cada procesador tiene su propia unidad de control que le suministra su flujo de instrucciones, y puede acceder a sus propios datos. Por tanto, los distintos procesadores pueden ejecutar diferentes programas sobre datos diferentes para resolver los distintos subproblemas que componen un problema común (véase la figura 7.4). Esto significa que los procesadores operan habitualmente de forma asíncrona. En su contra tienen el hecho de que los algoritmos asíncronos son más complicados de diseñar, analizar e implementar.

  La diferencia con las máquinas SISD con múltiples procesadores mencionadas anteriormente, está en el hecho de que las instrucciones y los datos están relacionados porque representan partes diferentes de la misma tarea a ejecutar. Por ello, los sistemas MIMD pueden ejecutar múltiples subtareas en paralelo, para acortar el tiempo de ejecución de la tarea principal. Hay una gran variedad de sistemas MIMD.

A continuación se hará otra importante distinción entre clases de sistemas:

- **Sistemas de memoria compartida**: Los sistemas de memoria compartida tienen múltiples CPUs compartiendo el mismo espacio de direcciones de memoria. Esto significa que el conocimiento acerca de dónde están almacenados los datos,

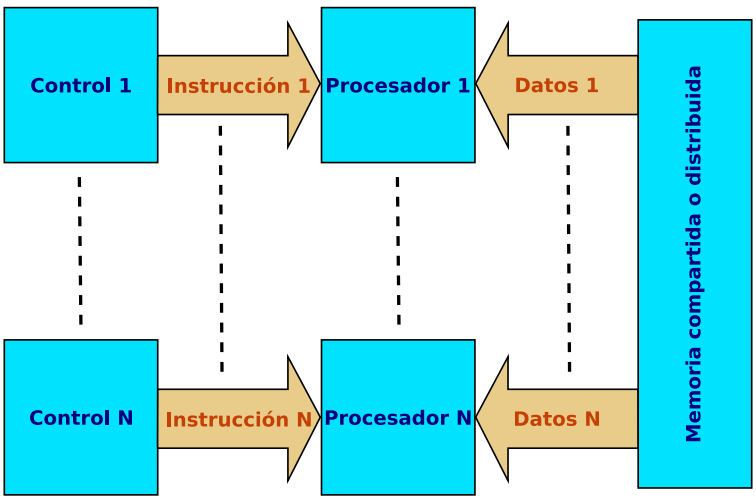


Figura 7.4: MIMD.

no concierne al usuario, ya que hay un único espacio de memoria al que acceden todos los procesadores (véase la figura 7.5). Los sistemas de memoria compartida pueden ser SIMD o MIMD, utilizándose en cada caso las abreviaturas SM-SIMD y SM-MIMD para las dos subclases mencionadas. Este tipo de arquitectura presenta una *latencia de red*[1] baja y un *ancho de banda*[2] alto.

- **Sistemas de memoria distribuida**: En este caso cada CPU lleva su propia memoria asociada. Las CPUs están conectadas por red y pueden intercambiar datos entre sus respectivas memorias cuando lo precisen(véase la figura 7.6). A diferencia de lo que ocurría en las máquinas de memoria compartida, el usuario tiene que conocer la localización de los datos en las memorias locales y moverlos o distribuirlos de manera explícita cuando sea necesario, utilizándose para ello el paradigma de programación de paso de mensajes. También los sistemas de memoria distribuida pueden ser SIMD o MIMD. Se usarán las abreviaturas DM-SIMD y DM-MIMD para las dos subclases que se acaban de ver.

7.2.1.2. Procesadores.

En los últimos años se ha producido un cambio drástico en lo relativo a los procesadores. Mientras que en el período de 1980–1990, los supercomputadores estaban equipados con procesadores propietarios y en particular con procesadores vectoriales,

[1] latencia de red: tiempo que se tarda en enviar un mensaje a través de ésta
[2] ancho de banda: número de bits que se pueden enviar por unidad de tiempo

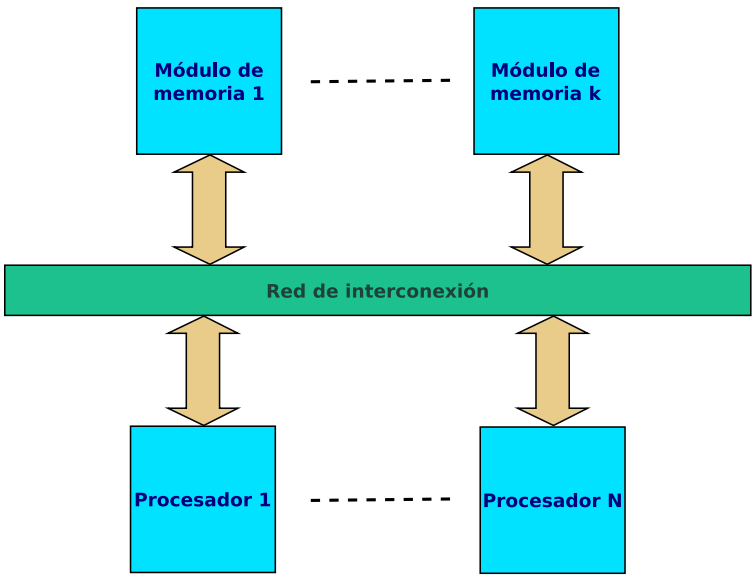


Figura 7.5: Sistema de memoria compartida.

hoy día éstos han sido sustituidos por procesadores comunes con tecnología RISC y procesadores Intel x86 compatibles. De hecho sólo dos empresas siguen produciendo sistemas vectoriales, mientras que el resto de sistemas ofertados están basados en CPUs RISC y x86 compatibles.

Pero además, el papel de los procesadores RISC también se ha reducido drásticamente en el último año. Aunque todavía se producen los procesadores HP/Compaq EV7 y HP PA-RISC, los sistemas que contienen el primero de ellos únicamente se mantendrán en el mercado hasta 2006, debido a la decisión de HP de reemplazar estos procesadores por otros de la línea Itanium. Los procesadores PA-RISC 8800 también desaparecerán en breve, si bien los PA-RISC 8900 continuarán de momento en producción.

Los procesadores RISC tienen generalmente una frecuencia de reloj más baja que la de los procesadores Intel Pentium 3/4 o los AMD, sin embargo, disponen de algunas características que les hacen encabezar el *ranking* de velocidad. En primer lugar, todos los procesadores RISC son capaces de realizar dos o más operaciones en coma flotante de 64 bits por cada ciclo de reloj. En segundo lugar, todos ellos ejecutan las instrucciones *out-of-order*[3], lo que mejora el número de instrucciones procesadas por ciclo (aunque

[3]En la ejecución de instrucciones *in-order*, se recibe una instrucción, se espera hasta recibir sus datos correspondientes y se ejecuta la instrucción sobre dichos datos. En la ejecución de instrucciones *out-of-order*, las instrucciones recibidas se almacenan en un búfer y se ejecutan según van recibiéndose los datos, pudiendo el orden de ejecución de instrucciones ser diferente al de recepción: se ejecutarán primero aquellas instrucciones que reciban antes los datos sobre los que deben actuar.

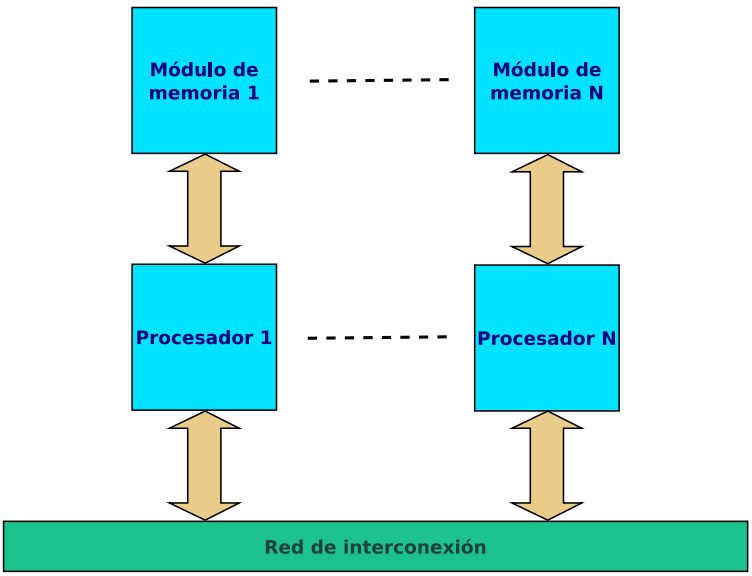


Figura 7.6: Sistema de memoria distribuida.

los nuevos procesadores de AMD tienen también 2 caminos para las instrucciones en coma flotante y ejecución de instrucciones *out-of-order*, están algo limitados por su adhesión al conjunto de instrucciones de Intel x86). Además, el ancho de banda en su comunicación con la memoria, solía ser mayor que en los procesadores compatibles con Intel.

Éste no va a ser ya el caso para procesador AMD Opteron, debido a su controlador de memoria incorporado y su bus HyperTransport que mejora el ancho de banda en las operaciones de E/S. Existe también la versión de doble núcleo, en la que se conectan dos CPUs en un único procesador, mejorándose la eficiencia y el rendimiento global del sistema.

El Power5 de IBM, es un procesador que incorpora dos CPUs RISC de 64 bits, cada uno con múltiples unidades de ejecución y su propia caché L1 de instrucciones y de datos. Además, incluye la caché L2 compartida entre ambas CPUs, y el directorio y la controladora de la caché L3. La propia caché L3 es demasiado grande para formar parte del chip por lo que se encuentra fuera de él. Además, el chip alberga también las controladoras de memoria.

El procesador Itanium 2 de Intel es el segundo de la familia de procesadores de 64 bits basado en la arquitectura Itanium. El procesador se ha diseñado para soportar grandes sistemas con miles de procesadores. Todos los procesadores Itanium 2 com-

partían una misma jerarquía de memoria caché. Tenían una caché L1 de 16 KB para instrucciones y otra de 16 KB para datos. La caché L2 estaba unificada (la misma para datos e instrucciones) y tenía un tamaño de 256 KB. La caché L3 también estaba unificada y variaba en tamaño, según el modelo, desde los 1,5 MB hasta los 9 MB. Recientemente ha aparecido la cuarta generación de procesadores Itanium 2, cuyo nombre en clave es *Montecito*. Consiste en un procesador de doble núcleo con la jerarquía de memoria mejorada y multihilo (*multi-threading*). *Montecito* lleva una jerarquía de memoria por cada núcleo y además los 256 KB de cache L2 de sus predecesores se han dedicado a datos, incorporando 1 MB más para instrucciones. Además la caché L3 pasa a ser de 12 MB, con lo que en total se pasa a tener casi 27 MB de caché en el procesador.

Ya se ha comentado la decisión de HP de incluir procesadores de la línea Itanium en sus ordenadores. Lo mismo ha ocurrido con Silicon Graphics, que ha decidido basar sus nuevos sistemas en procesadores Itanium, en lugar de los procesadores MIPS que había estado utilizando; así, mientras que los ordenadores de la serie Origin3000 usaban el procesador MIPS R16000, los más recientes superordenadores de la gama Altix, incorporan ya procesadores Itanium. Por todo lo comentado, los procesadores Itanium cobran bastante peso dentro de la supercomputación actual.

Los procesadores UltraSPARC IV+ constituyen la quinta generación de la familia UltraSPARC (Ultra Scalable Processor ARChitecture). Son procesadores RISC de 64 bits y doble núcleo de Sun Microsystems, que constituyen una mejora respecto a los UltraSPARC IV. Poseen 96 KB de caché L1 repartida en 64 KB para datos y 32 KB para instrucciones, 2 MB de caché L2, y 32 MB de caché L3 externa. Se trata de procesadores multihilo con 2 hilos de ejecución por procesador. Los procesadores UltraSPARC son utilizados en todos los productos SUN.

Recientemente han aparecido en el mercado los procesadores UltraSPARC T1, situándose en la cabeza de la gama. Los UltraSPARC T1 vienen con hasta 8 núcleos y 4 hilos de proceso por núcleo, con lo que se consiguen hasta 32 hilos de proceso en un único chip.

7.2.1.3. Redes de interconexión.

La velocidad de las redes de conexión entre procesadores junto con la velocidad de dichos procesadores, son dos factores decisivos para obtener buenos resultados tanto en sistemas paralelos integrados como en *clusters*. En los primeros días de los *clusters*,

la comunicación entre los procesadores estaba limitada, debido a la alta latencia y la falta de ancho de banda de la red utilizada (principalmente Ethernet). La situación ha cambiado mucho con la aparición de mejores redes de interconexión, que han sido utilizadas incluso en sistemas paralelos integrados.

Una complicación de las redes rápidas ofrecidas para *clusters* es la conexión con los nodos. Donde en el caso de las máquinas paralelas integradas, el acceso a los nodos está adaptado y puede hacerse de tal modo que el ancho de banda de la red se ajuste al ancho de banda interno en un nodo, en los *clusters* hay que utilizar el bus PCI que viene con los nodos basados en PC (los *clusters* de PCs se verán en un apartado posterior). Los anchos de banda disponibles se encontraban en el rango de 110–480 MB/s, pero desde 1999 está disponible el bus PCI-X que presentaba inicialmente un ancho de banda de 1GB/s y en la versión 2.0 anchos de banda de 2 y 4 MB/s. Este bus es el normalmente utilizado en nodos de PCs que van a formar parte de un *clúster*.

Una red ampliamente utilizada es Gigabit Ethernet, que con un ancho de banda teórico máximo de 125 MB/s y latencias del orden de 100 μs es utilizada en aplicaciones en las que la latencia no sea muy importante. Su bajo coste, en comparación con otras alternativas, han hecho de Gigabit Ethernet una opción interesante. La aparición de 10-Gigabit Ethernet, ha incrementado su atractivo al multiplicarse por 10 el ancho de banda, y reducirse la latencia a valores inferiores a 10 μs.

Infiniband se ha convertido rápidamente en un medio ampliamente aceptado para redes de conexión entre nodos. Se utiliza para conectar varios componentes que forman parte de un sistema. Puede utilizarse como red de conexión entre procesadores, uniendo subsistemas de entrada/salida, o entre *switches* multiprotocolo, como switches Gbit Ethernet, etc. Infiniband no depende de una tecnología particular, por lo que es una buena base sobre la que implementar una librería de comunicación (como MPI). Infiniband define una velocidad de conexión de 2.5 Gb/s (312.5 MB/s), y también velocidades de 4x y 12x de 1.25 GB/s y 3.75 GB/s, con latencias inferiores a 7 μs para pequeños mensajes.

Myrinet es un sistema de red de altas prestaciones diseñado por Myricom. Es el líder del mercado en redes de conexión rápidas, siendo empleado en gran cantidad de *clusters*. Está disponible en forma de dos productos: Myrinet-2000 y Myri-10G, ambos con la misma arquitectura de red y los mismos protocolos. Myrinet-2000 proporciona 2 Gb/s en ambas direcciones con latencias del orden de 3 μs, Myri-10G por su parte proporciona 10 Gb/s en ambas direcciones con latencias de 2 μs.

QsNet, de la empresa Quadrics, es otro sistema de conexión de alta velocidad. Su

última generación es QsNet^{II}, que es una evolución de la arquitectura QsNet y está en el mercado desde finales de 2003. Proporciona un ancho de banda de 900 MB/s con latencias de 3 μs.

SCI (Scalable Coherent Interface), por su parte, es el más antiguo de los sistemas de conexión vistos en el presente apartado. Proporciona un ancho de banda de 320 MB/s con latencias inferiores a 2 μs.

7.2.1.4. La lista TOP500.

Desde 1993 se puede consultar una lista con los 500 ordenadores más potentes del momento. Esta lista, llamada TOP500, se actualiza dos veces al año y está disponible en http://www.top500.org/. En el momento de escribir este capítulo, la última versión disponible era de junio de 2006.

Además de la lista, en el sitio *web* mencionado puede encontrarse información complementaria sobre la situación actual y la evolución del dinámico mundo de la supercomputación.

Los 10 más potentes. En la tabla 7.1, se muestran los 10 más potentes del mundo según la clasificación Top500 de junio de 2006. El campo R_{max} indica el rendimiento máximo obtenido, y R_{peak} el pico teórico, ambos medidos en GIGA FLOPS.

Algunos aspectos interesantes de la lista de los 10 más potentes son:

- El número 1 de la lista continúa siendo el sistema IBM BlueGene/L de DOE, instalado en el *DOE's Lawrence Livermore National Laboratory* (LLNL), que ya se encontraba en esta posición en las tres últimas listas TOP500. Este sistema ha conseguido un récord de comportamiento de 280.6 TFlops, siendo todavía el único que ha superado la barrera de los 100 TFlops.

- Tres de los 10 sistemas más potentes en noviembre de 2005, se han visto desplazados por otros nuevos sistemas instalados. El sistema más potente en Europa ocupa la posición número 5 y está ubicado en el *Commissariat a l'Energie Atomique (CEA)* en Francia.

- El sistema MareNostrum, instalado en el Centro de Supercomputación de Barcelona, que ocupaba la octava posición en la lista de Noviembre de 2005, ha dejado

Pos.	**Sitio**	**Ordenador**	**Proces.**	**Año**	R_{max}	R_{peak}
1	DOE/NNSA/LLNL USA	BlueGene/L - eServer Blue Gene Solution IBM	131072	2005	280600	367000
2	IBM Thomas J. Watson Research Center, USA	BGW - eServer Blue Gene Solution IBM	40960	2005	91290	114688
3	DOE/NNSA/LLNL USA	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM	12208	2006	75760	92781
4	NASA/Ames Research Center/NAS USA	Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband SGI	10160	2004	51870	60960
5	Commissariat a l'Energie Atomique (CEA) France	Tera-10 - NovaScale 5160, Itanium2 1.6 GHz, Quadrics Bull SA	8704	2006	42900	55705.6
6	Sandia National Laboratories USA	Thunderbird - PowerEdge 1850, 3.6 GHz, Infiniband Dell	9024	2006	38270	64972.8
7	GSIC Center, Tokyo Institute of Technology, Japan	TSUBAME Grid Cluster, Sun Fire X64 Cluster, Opteron 2.4/2.6 GHz, Infiniband NEC/Sun	10368	2006	38180	49868.8
8	Forschungszentrum Juelich (FZJ) Germany	JUBL - eServer Blue Gene Solution IBM	16384	2006	37330	45875
9	Sandia National Laboratories USA	Red Storm Cray XT3, 2.0 GHz Cray Inc.	10880	2005	36190	43520
10	The Earth Simulator Center, Japan	Earth-Simulator NEC	5120	2002	35860	40960

Tabla 7.1: Los 10 ordenadores más potentes del mundo en junio de 2006 (Fuente: http://www.top500.org/).

de estar entre los 10 primeros, pasando a ocupar la posición 11 dentro de la lista TOP500 de junio de 2006.

- El sistema *Earth Simulator*, que ocupó la primera posición durante 5 listas consecutivas hasta ser reemplazado por el sistema IBM BlueGene/L en noviembre de 2004, todavía se mantiene en la décima posición.
- La marca requerida para encontrarse entre los 10 primeros supera los 35 TFlops.

Aspectos destacados de la última publicación de la lista TOP500. De la lista TOP500 de junio de 2006, se puede destacar lo siguiente:

- IBM se mantiene como la empresa dominante en supercomputación, con casi la mitad de los equipos de la lista (48,6 %); además 4 de los 10 primeros son de esta marca. Hewlett-Packard (HP) se mantiene en segunda posición con el 30.8 % del total. Ningún otro fabricante llega siquiera al 5 %.

- Los procesadores más utilizados son los de la marca Intel, que están presentes en 301 de los 500 sistemas. Los procesadores Opteron de AMD han incrementado su presencia, siendo utilizados ahora por 81 sistemas, frente a los 25 en que eran utilizados hace solamente un año.

- Estados Unidos es claramente el líder en consumo de supercomputadores, estando allí instalados 298 sistemas de los 500 de la lista. Europa cae hasta los 83 sistemas, desde los 100 que tenía hace seis meses, mientras que Asia sube desde los 66 que tenía hace seis meses hasta los 93 actuales.

- Todos los sistemas presentes en la lista superan los 2.03 TFlops, frente a los 1.17 TFlops de hace un año.

- La potencia total acumulada ha crecido hasta los 2.79 PFlops, desde los 2.30 PFlops de hace seis meses y los 1.69 PFlops de hace un año.

- 365 sistemas están etiquetados como *clusters*, siendo ésta la arquitectura más común de la lista. De ellos, 255 utilizan Gigabit Ethernet como tecnología de interconexión del sistema, frente a los 87 que utilizan Myrinet de Myricom.

7.2.1.5. Otras consideraciones.

Los superordenadores tienen en su contra, principalmente, el precio, llegando en el caso de los más potentes a centenares de millones de euros. Pero sin necesidad de llegar a esos extremos, los superordenadores "más modestos" tampoco están al alcance de cualquiera, y sólo unos pocos organismos públicos y algunas grandes empresas, pueden permitírselos. No es únicamente el ordenador lo que hay que pagar, pues también se requiere una ubicación debidamente acondicionada y climatizada, con sistema contra-incendios, y suelen venir acompañados por un costoso contrato de mantenimiento.

Los investigadores ajenos al organismo propietario que precisen la utilización de un superordenador, habitualmente deberán pagar por ello, pero aparte del coste por la utilización, hay además una serie de inconvenientes que también se deben tener en cuenta:

- Necesidad de compartir la máquina con otros usuarios, lo que puede limitar el número de procesadores que se podrán utilizar, o el tiempo máximo de duración de un determinado experimento.

- Necesidad de adaptación al software instalado en la máquina, no siendo posible la realización de ningún tipo de tareas de administración, como podría ser la instalación de algún programa de interés.

- Necesidad de trabajar de forma remota: aunque en ocasiones pueda ser una ventaja poder enviar los trabajos para su ejecución y recoger los resultados de forma remota, en otras ocasiones, sobre todo cuando haya que realizar tareas de depuración de código, resulta más beneficioso trabajar en modo local.

Algunos de los puntos enumerados podrían ser negociados, pero en todo caso, existe una clara dependencia del organismo propietario.

7.2.2. Clusters de PC's.

Ante los inconvenientes mencionados en el apartado anterior, en la década de los 90 se contempló la posibilidad de conectar en red varios PC's, para poder utilizar de forma conjunta la potencia de cálculo de sus procesadores. En 1994 Thomas Sterling y Donald Becker, trabajando para la NASA, construyeron un ordenador paralelo formado por 16 procesadores 486 DX4 conectados en red y con sistema operativo Linux [14]. La idea era mostrar que se podían utilizar ordenadores de propósito general para tratar problemas computacionales complejos. Sterling y Becker le dieron a su máquina el nombre de *Beowulf*, en honor de un guerrero escandinavo del siglo VI cuyas aventuras se relatan en el primer poema épico conocido en lengua inglesa, y desde entonces el nombre se ha venido utilizando para describir a los *clusters* de ordenadores construidos a partir de componentes asequibles.

Los *clusters* de PC's permiten a instituciones educativas y científicas el acceso a la *computación de alto rendimiento* (HPC, *High Performance Computing*), sin la necesidad de adquirir equipos costosos que lo convertirían en algo prohibitivo.

Las principales características de estos sistemas son las siguientes:

- Presentan una muy ventajosa relación rendimiento/precio con respecto a los superordenadores.

- Los nodos del sistema son ordenadores de propósito general, que pueden incorporar uno o varios procesadores.
- Constituyen un sistema distribuido, con una red (o varias) dedicada a soportar la comunicación de paquetes entre los distintos nodos. En principio se utilizaron redes Ethernet, aunque actualmente se utilizan otras más avanzadas como Myrinet, Gigabit Ethernet o Infiniband.
- Presentan una mayor flexibilidad de configuración, permitiendo modificar de forma sencilla la estructura del sistema.
- Reemplazar un elemento defectuoso del sistema es mucho más barato que la cuota del contrato de mantenimiento que habitualmente viene con un supercomputador.
- Permiten una rápida actualización, en respuesta a la aparición de nuevo *hardware* más avanzado.
- Utilizan Linux, principalmente, como sistema operativo y algún entorno de programación paralelo como pueden ser PVM (*Parallel Virtual Machine*) o MPI (*Message Passing Interface*).

Entre los inconvenientes se pueden mencionar:

- Los *clusters* se comportan pobremente en aplicaciones que requieren una gran cantidad de memoria compartida.
- Presentan pobres latencias y anchos de banda en las interconexiones.
- Imposibilidad de ejecutar algunos tipos de aplicaciones en los que los factores comentados en los dos puntos anteriores sea importante.

En un estudio sobre el comportamiento de distintos tipos de ordenadores, Farley [49] concluye que los *clusters* de PC's exhiben comportamientos comparables a los de los superordenadores utilizados en dicho estudio, pero con una relación rendimiento/precio muy superior.

A continuación se describe el *clúster* de PCs diseñado por Dormido en el Departamento de Informática y Automática de la Universidad Nacional de Educación a Distancia (UNED), durante el desarrollo de su tesis doctoral [41], y al que se le dio el nombre de *Smaug*. El *clúster* fue utilizado posteriormente por Dormido en [42], y fue el que se empleó en el estudio de paralelización que se expone más adelante en este mismo capítulo.

7.2.2.1. El *clúster Smaug.*

El *clúster* está compuesto por 16 ordenadores interconectados mediante un *switch Fast Ethernet*. Cada uno de los ordenadores contiene un procesador AMD K7 500 MHz, 512 KB de caché y 384 MB de memoria RAM. Se utilizaron también 5 controladores KVM (*Keyboard-Video-Mouse*), configurados en cascada, para poder acceder a cualquiera de los ordenadores utilizando únicamente un teclado, un ratón y un monitor. En la figura 7.7 puede verse el aspecto de Smaug, por las partes delantera y trasera, mientras que la figura 7.8 muestra los controladores KVMs utilizados.

(a) (b)

Figura 7.7: (a) Vista frontal de Smaug (b) Vista trasera de Smaug.

De los 16 ordenadores, uno de ellos actúa como servidor (el que aparece en la esquina inferior derecha de la figura 7.7a). Este ordenador se diferencia del resto en que dispone de un disco duro de 17 GB y dos tarjetas de red, mientras que los otros tienen un disco duro de 6 GB y una única tarjeta de red.

Las placas base son Gigabyte GA-5AX con bus PCI de 64 bits a 66 MHz. El subsistema de memoria se conecta mediante un bus de 64 bits a 100 MHz. Las tarjetas de red son 3Com 3c905B-TX-MN.

Cada uno de los 15 clientes del *clúster* dispone de una conexión dedicada a un puerto del *switch* 3Com SuperStack II 3300 de 24 puertos, como se muestra en la figura 7.9. El *switch* permite establecer transmisiones simultáneas entre las máquinas conectadas a él

Figura 7.8: Controladores KVMs.

siempre y cuando no coincida el receptor. Otro puerto del *switch* se dedica a conectar el servidor. Éste dispone de una segunda tarjeta de red, 3Com 3c905C-TX-M, que se diferencia de la primera en que dispone de activación remota (RWU, *Remote Wake Up*), lo que permite encender el equipo desde un terminal remoto. De las dos tarjetas 3Com de que dispone el servidor, una se dedica a la mencionada conexión con el *switch* y el resto de ordenadores, mientras que la otra permite (además de la activación remota ya comentada) el acceso remoto al *clúster* a través de Internet. Únicamente el servidor es *visible* desde el exterior.

Figura 7.9: *Switch* utilizado en Smaug.

El sistema operativo es Linux (Red Hat 6.1) y también tiene instalado el sistema de paso de mensajes PVM (Parallel Virtual Machine) para la programación en paralelo.

Dispone del sistema de archivos NFS (*Network File System*) y el servicio de información de redes NIS (*Network Information Service*). NFS permite compartir directorios entre las máquinas del *clúster*. En Smaug, en concreto, se comparte el directorio /home de la máquina que actúa como maestro. NIS permite distribuir la información que tiene que ser conocida por todas las máquinas de la red. Dentro de esta información está la de usuarios (/etc/passwd) y grupos (/etc/group).

7.3. Paralelización de un algoritmo evolutivo aplicado al control, en un *clúster* de PCs.

Como ya se comentó en la introducción, se va a realizar un estudio de optimización para la paralelización del lazo interno del algoritmo que se empleó en el capítulo 6, aplicado al problema del controlador del avión.

Durante un estudio previo, se determinarán cuáles son las partes del algoritmo que consumen más recursos computacionales, por ser éstas las más interesantes a la hora de paralelizar y en las que habrá que centrar los esfuerzos.

Una característica importante a tener en cuenta en este tipo de algoritmos es que cada generación depende de la anterior, lo que impone alguna limitación a la hora de llevar a cabo la paralelización. Además, la evaluación de los distintos cromosomas requerirá, en general, tiempos diferentes, siendo éstos mucho menores para los cromosomas que dan lugar a sistemas inestables, por no requerir la simulación numérica del experimento; esto último hará difícil la distribución equilibrada de la carga de trabajo entre los distintos procesadores.

Se empleará el paradigma de programación Maestro/Esclavo con paso de mensajes: el maestro será responsable de dividir el problema en tareas más pequeñas, distribuir dichas tareas entre los esclavos, recoger la información producida por los esclavos tras la realización de su trabajo y gestionarla para llegar, tras múltiples iteraciones de los pasos anteriores, a la solución del problema. Para ello se utilizará la *toolbox* PVMTB (*Parallel Virtual Machine ToolBox*) desarrollada en Matlab por Baldomero [12], basada en la librería estándar PVM. PVMTB permite aprovechar la potencia de las *toolboxes* de Matlab especializadas en control, en el diseño de programas en paralelo.

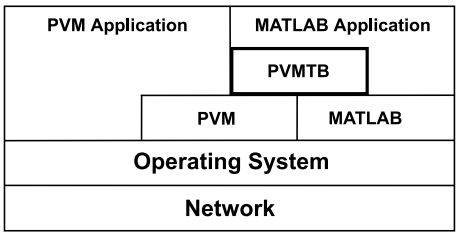


Figura 7.10: Diagrama de alto nivel de PVMTB.

La figura 7.10 muestra un diagrama global de alto nivel de PVMTB. La *toolbox* hace llamadas a PVM y al API de Matlab para habilitar los mensajes entre procesos

de Matlab.

7.3.1. El algoritmo secuencial.

El algoritmo evolutivo utilizado, se muestra en la figura 7.11. Se genera, de forma aleatoria, una población inicial de cromosomas cuyos miembros son posibles soluciones al problema, debidamente codificadas. Los cromosomas son evaluados y ordenados de acuerdo con sus aptitudes. El cromosoma con la mejor aptitud es elegido como solución al problema. Después se entra en un bucle en el que se van obteniendo nuevas generaciones de cromosomas a partir de la anterior, mediante la aplicación de los operadores evolutivos sobre los cromosomas seleccionados en un paso anterior. Además, en cada generación se añadirá un pequeño porcentaje de inmigrantes –cromosomas obtenidos de forma aleatoria– con objeto de aumentar la diversidad de la población y evitar la convergencia prematura hacia subóptimos locales. La nueva generación es de nuevo evaluada y ordenada de acuerdo con las aptitudes. Si el mejor cromosoma de la población actual es más adecuado que el previamente elegido como solución al problema, ocupará su lugar. Finalmente, si se dan las condiciones de finalización el programa terminará, si no comenzará una nueva iteración.

```
k ← 0
obtener población inicial P_k(x) aleatoriamente
for each x_i in P_k(x)
   decodificar x_i
   evaluar x_i ← obtener aptitud
endfor
ordenar P_k(x) de acuerdo con las aptitudes
solución ← mejor x_i en P_k(x)
while no condiciones de finalización
    k ← k + 1
   seleccionar padres por técnica de torneo
   obtener nueva población P_{k+1}(x)
   for each x_i in P_k(x)
      decodificar x_i
      evaluar x_i ← obtener aptitud
   endfor
   ordenar P_k(x) de acuerdo con las aptitudes
   if mejor x_i en P_k(x) es mejor que solución
      solución ← mejor x_i en P_k(x)
   endif
endwhile
```

Figura 7.11: Algoritmo evolutivo secuencial.

7.3.2. Estudio previo de tiempos.

Antes de comenzar a paralelizar el algoritmo secuencial, se realizó un estudio del consumo de tiempo que tenía lugar en cada parte del programa, así como el tiempo empleado en el paso de mensajes. El experimento se llevó a cabo en 10 ocasiones, recogiéndose en la tabla 7.2 los valores medios.

Concepto	t (seg.)
Obtener la población inicial	1.4764
Obtener nueva población a partir de la anterior	0.29011
Ordenar la población inicial	0.007018
Evaluar población inicial (sin paralelizar)	0.081052
Evaluar población final (sin paralelizar)	8.2353
Enviar y recibir 1 cromosoma (14 genes)	0.001299
Enviar y recibir 10 cromosomas	0.001625
Enviar y recibir 20 cromosomas	0.001925
Enviar y recibir 30 cromosomas	0.002192
Enviar y recibir 40 cromosomas	0.002931
Enviar y recibir 50 cromosomas	0.003355
Enviar y recibir 60 cromosomas	0.003509
Enviar y recibir 70 cromosomas	0.003761
Enviar y recibir 80 cromosomas	0.004556
Enviar y recibir 90 cromosomas	0.004906
Enviar y recibir 100 cromosomas	0.005141
Enviar y recibir 1 vector de aptitud (23 elementos)	0.001274
Enviar y recibir 10 vectores de aptitud	0.00175
Enviar y recibir 20 vectores de aptitud	0.002109
Enviar y recibir 30 vectores de aptitud	0.003048
Enviar y recibir 40 vectores de aptitud	0.003494
Enviar y recibir 50 vectores de aptitud	0.004301
Enviar y recibir 60 vectores de aptitud	0.00488
Enviar y recibir 70 vectores de aptitud	0.005305
Enviar y recibir 80 vectores de aptitud	0.006334
Enviar y recibir 90 vectores de aptitud	0.006769
Enviar y recibir 100 vectores de aptitud	0.007551

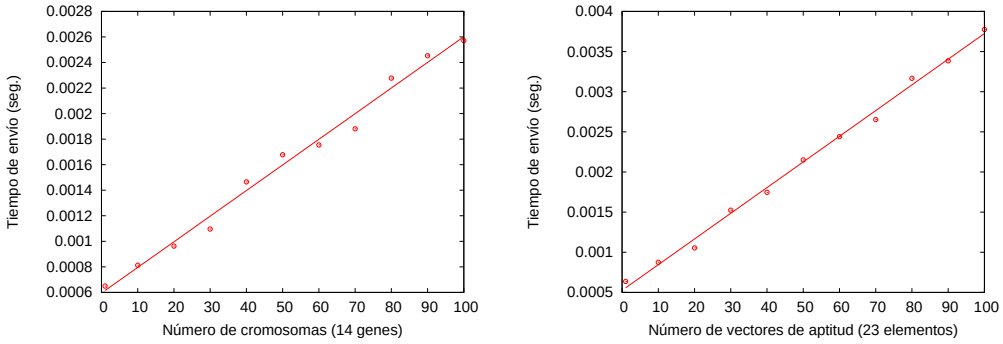
Tabla 7.2: Estudio temporal.

Suponiendo que el tiempo de envío de un mensaje desde el proceso maestro hasta un proceso esclavo es el mismo que en sentido inverso (desde el esclavo al maestro), se pueden obtener los tiempos de envío dividiendo por dos los valores de la tabla 7.2. Los tiempos de envío así obtenidos se recogen en la tabla 7.3.

Nº de cromosomas	t (seg.)	Nº de vectores de aptitud	t (seg.)
1	0.0006495	1	0.0006370
10	0.0008125	10	0.0008750
20	0.0009625	20	0.0010545
30	0.0010960	30	0.0015240
40	0.0014655	40	0.0017470
50	0.0016775	50	0.0021505
60	0.0017545	60	0.0024400
70	0.0018805	70	0.0026525
80	0.0022780	80	0.0031670
90	0.0024530	90	0.0033845
100	0.0025705	100	0.0037755

Tabla 7.3: Tiempos de envío de cromosomas y vectores de aptitud.

Si se representan gráficamente los tiempos de envío de los cromosomas y de los vectores de aptitud, se observa que en ambos casos se ajustan bastante bien a una recta (figura 7.12), por lo que se pueden obtener sendas expresiones que permitan calcular el tiempo de envío en función del número de cromosomas y del número de vectores de aptitud a enviar. Dichas expresiones corresponden a las de las rectas de regresión calculadas por el método de mínimos cuadrados.



(a) Tiempo vs. Num. cromosomas (b) Tiempo vs. Num. vectores aptitud

Figura 7.12: Tiempo de envío de mensajes en función del número de cromosomas y el número de vectores de aptitud.

Las ecuaciones de las rectas de regresión son:

$$t = 0.00002 \cdot NCrom + 0.000595 \tag{7.1}$$

$$t = 0.000032 \cdot NFit + 0.000526 \tag{7.2}$$

donde $NCrom$ y $NFit$ son respectivamente el número de cromosomas y el de vectores de aptitud enviados. Haciendo dichos valores iguales a cero se obtienen latencias de $595\,\mu s$ y $526\,\mu s$ respectivamente.

En la tabla 7.2 se observa que lo que más tiempo consume es la evaluación de las poblaciones que se irán generando durante la ejecución del algoritmo. Hay una gran diferencia entre el tiempo empleado en evaluar la población inicial y el tiempo que se tarda en evaluar la población final. Esto se debe a que al principio, los cromosomas que se han obtenido de forma aleatoria dan lugar, en su mayoría, a sistemas inestables que son rápidamente evaluados, no perdiéndose tiempo en evaluar el resto de objetivos si no se cumple el considerado como básico: la estabilidad. A medida que la ejecución del algoritmo progresa, la población en general se va volviendo más apta, requiriéndose un mayor tiempo para su evaluación. Pero aunque en general los cromosomas se vuelvan más aptos con el tiempo, en todas las generaciones puede haber cromosomas que den lugar a sistemas inestables (recuérdese que hay cromosomas generados aleatoriamente); esto dificultará el reparto equilibrado de la carga de trabajo entre los distintos procesadores.

El tiempo en obtener la población inicial también toma un valor relativo elevado, pero como sólo deberá obtenerse una vez, el tiempo ($\approx$ 1.5 seg.) no será significativo en comparación con el tiempo total de ejecución del algoritmo.

Sin embargo, el tiempo en obtener el resto de generaciones, si bien es menor que el correspondiente a la generación inicial, es relativamente elevado, por lo que deberá ser tenido en cuenta a la hora de llevar a cabo la paralelización.

Pueden parecer excesivamente elevados los tiempos empleados en obtener los cromosomas de las nuevas generaciones, al ser simplemente el resultado de aplicar los operadores evolutivos sobre la población anterior. La explicación a este hecho está en que la función encargada de generar los números aleatorios es una función especializada que requiere un tiempo mayor que el que hubiera sido necesario si se hubiera empleado una densidad de probabilidad uniforme. Dicha función especializada se ajusta a la función de densidad de probabilidad variable descrita en el apartado 3.4.1.3.

El tiempo empleado en el envío de cromosomas y vectores de aptitud es muy inferior al tiempo medio de evaluación de un cromosoma, por lo que se podrá hacer un reparto fino del trabajo con objeto de equilibrar la carga.

A partir de los resultados del presente estudio, se diseñaron los algoritmos de paralelización que serán descritos en los apartados siguientes.

7.3.3. Paralelización del algoritmo secuencial. Resultados obtenidos.

Como ya se ha comentado en el apartado anterior, las partes del algoritmo secuencial que más tiempo consumen son la evaluación de los cromosomas y la obtención de nuevas generaciones, siendo por tanto las que serán tenidas más en cuenta al diseñar el algoritmo paralelo.

Se utilizaron 4 estrategias de paralelización, con algunas diferencias entre ellas, que dieron lugar a distintos resultados. Estas estrategias, junto con los resultados obtenidos serán expuestos a continuación.

Debido a problemas de sobrecalentamiento que surgieron en dos de las máquinas, únicamente pudieron utilizarse 14 de las 16 que componían el *clúster*.

7.3.3.1. Estrategia 1.

En la primera estrategia, cada proceso recibe de una vez todos los cromosomas de la población que le corresponden para evaluar. El proceso maestro también participa en la evaluación de cromosomas. El diagrama de flujo correspondiente se muestra en la figura 7.13.

En dicha figura se observa que el proceso maestro se encargará de iniciar los procesos esclavos en el comienzo de la ejecución del programa. Luego generará la población inicial y enviará a cada esclavo el grupo de cromosomas que deba evaluar, evaluando él a su vez los que le correspondan. Este proceso de generación de la población y reparto de los cromosomas a evaluar, será repetido por el maestro en todas las generaciones hasta el final del programa.

Una vez que el maestro ha evaluado sus cromosomas, pasará a la escucha para recibir los resultados de la evaluación realizada por los esclavos, y cuando disponga de todos los datos ordenará la nueva población en función de su aptitudes. Comprobará después si el mejor cromosoma de la nueva generación es más apto que el mejor de los obtenidos hasta el momento, en cuyo caso lo sustituirá, y finalmente comprobará si se dan las condiciones de finalización o deberá iniciarse el proceso de obtención de una nueva generación.

Los esclavos por su parte, estarán a la escucha para recibir los cromosomas que les encomiende el maestro para su evaluación. Cuando reciban estos cromosomas los evaluarán y los devolverán al maestro junto con sus vectores de aptitud, volviendo a

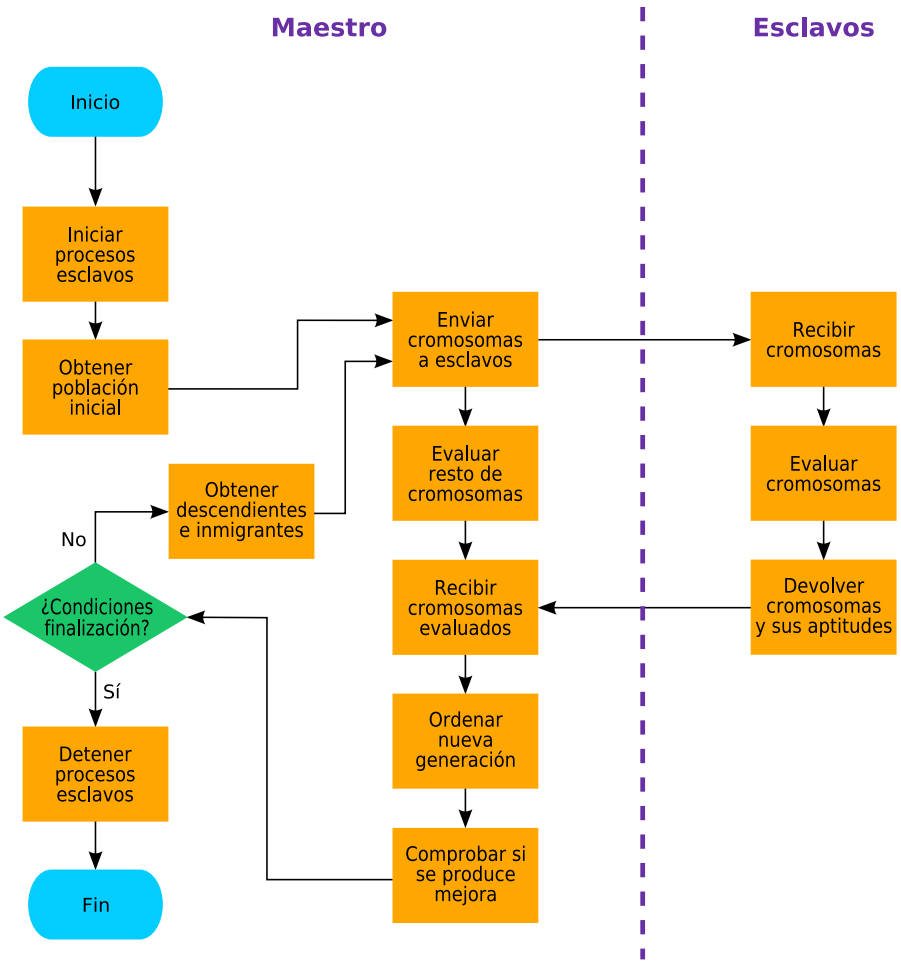
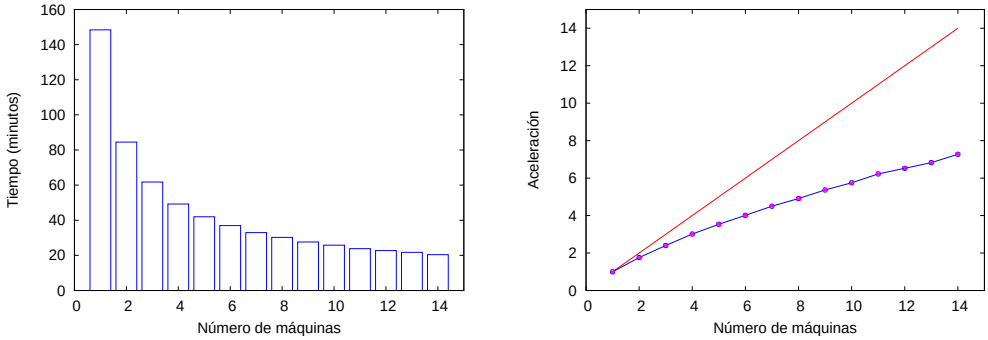


Figura 7.13: Diagrama de flujo correspondiente a la estrategia 1.

pasar a la escucha.

Con esta estrategia, no se realiza una distribución homogénea de la carga de trabajo, por lo que habrá esclavos que tengan que permanecer a la escucha demasiado tiempo (mientras el resto de esclavos terminan sus tareas).

El programa correspondiente a esta estrategia se ejecutó en 4 ocasiones, recogiéndose los resultados completos en la tablas F.1 y F.2 del apéndice F. En la figura 7.14 se muestran gráficamente los resultados medios obtenidos.



(a) Tiempo vs. Num. máquinas (b) Aceleración vs. Num. máquinas

Figura 7.14: Tiempo empleado por el algoritmo y aceleración en función del número de máquinas (para estrategia 1).

La tabla 7.4, por su parte, recoge los valores numéricos.

Nº proces.	tiempo	acel.	desv.
1	2:28:24	1	0
2	1:24:30	1.75614	0.0160058
3	1:01:46	2.40232	0.00497672
4	0:49:15	3.01306	0.0144666
5	0:41:59	3.53418	0.0187174
6	0:36:58	4.01414	0.0203438
7	0:32:59	4.49996	0.0247556
8	0:30:13	4.90989	0.00594494
9	0:27:38	5.36988	0.0315031
10	0:25:48	5.75143	0.0125055
11	0:23:51	6.22299	0.0298395
12	0:22:45	6.52514	0.0639573
13	0:21:44	6.82734	0.0241109
14	0:20:25	7.26966	0.0232065

Tabla 7.4: Tiempos medios y aceleraciones medias con desviación estándar vs. nº de procesadores (estrategia 1).

Se observa una aceleración media de 7.26966 para el caso de 14 máquinas, pasando el tiempo empleado para obtener la solución de 2 h 28 min 24 seg, a 20 min 25 seg.

7.3.3.2. Estrategia 2.

La segunda estrategia es prácticamente igual a la primera, estando la diferencia en que el maestro ahora no evaluará cromosomas, y siendo idéntico en el resto. El diagrama

de flujo correspondiente a este caso es el que se muestra en la figura 7.15.

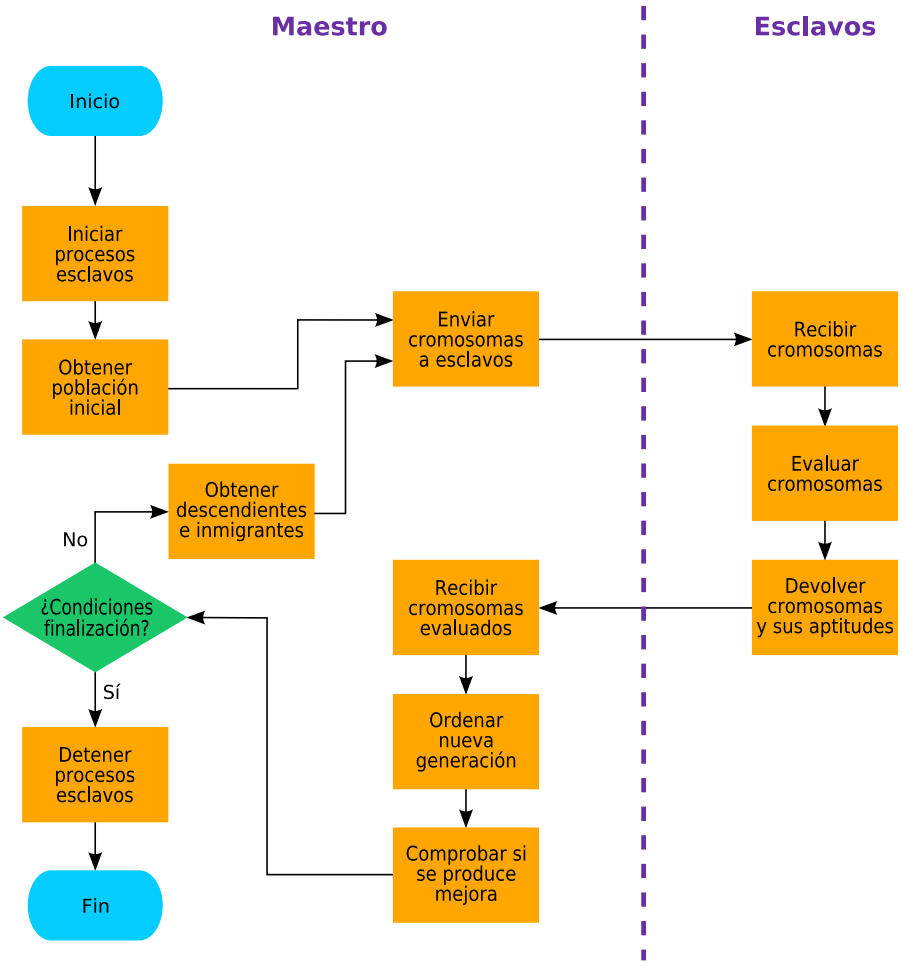
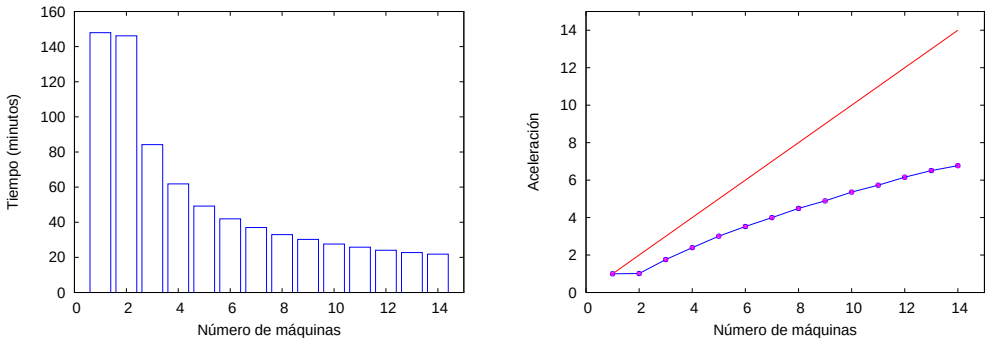


Figura 7.15: Diagrama de flujo correspondiente a la estrategia 2.

De igual forma que ocurría con la estrategia 1, el programa se ejecutó en 4 ocasiones, recogiéndose los resultados completos en la tablas F.3 y F.4 del apéndice F. En la figura 7.16 se muestran gráficamente los resultados medios obtenidos, y en la tabla 7.5, se recogen los valores numéricos.

Se comprueba en este caso, que las aceleraciones obtenidas son inferiores a las que se obtuvieron con la estrategia 1, pasando, para el caso de 14 máquinas, de 7.26966 a 6.7724, lo que significa una disminución del 6.84 % respecto a la primera estrategia. Esta circunstancia indica que en la estrategia 2 el proceso maestro pasará un tiempo inactivo en cada generación, a la espera de recibir los resultados de los procesos esclavos.

Analizando los datos de la tabla 7.2, si se divide $8.2353seg$ (el tiempo empleado en



(a) Tiempo vs. Num. máquinas (b) Aceleración vs. Num. máquinas

Figura 7.16: Tiempo empleado por el algoritmo y aceleración en función del número de máquinas (para estrategia 2).

evaluar una población final) entre 13 (el número de procesos esclavos), se obtiene que el tiempo medio empleado por cada esclavo es $0.63348seg$. Como el tiempo empleado por el proceso maestro en realizar sus tareas está en torno a los $0.3seg$ (aproximadamente la suma del tiempo empleado en ordenar una población con el necesario para obtener una población a partir de la anterior, ver tabla 7.2), se comprueba que en las últimas generaciones el proceso maestro pasará mucho tiempo inactivo.

Sin embargo, durante las primeras generaciones, el tiempo de evaluación de los cromosomas estará próximo al correspondiente a la generación inicial. Si se divide el tiempo empleado en evaluar una población inicial ($0.081052seg$), entre el número de máquinas esclavas (13), se obtiene que el tiempo medio empleado por cada esclavo es $0.0062348seg$, que es muy inferior al tiempo aproximado de $0.3seg$ empleado por el maestro en realizar sus tareas. En este caso, por tanto, serán los esclavos los que pasarán mucho tiempo inactivos.

Durante la ejecución del algoritmo, se producirá una evolución desde una situación inicial en la que los esclavos pasen mucho tiempo inactivos hacia otra en la que será el maestro el que permanezca gran parte del tiempo a la espera. La comparativa entre los resultados obtenidos con la estrategia 1 y la estrategia 2 permite concluir que será la segunda situación la que predomine durante el tiempo; esto hace que sea interesante que el maestro también se encargue de la evaluación de cromosomas.

Como ya se ha comentado anteriormente, la diferencia entre los tiempos de evaluación de distintos cromosomas está, principalmente, en que algunos cromosomas dan lugar a sistemas inestables que son rápidamente evaluados por el algoritmo. A medida que se incrementa el número de generaciones producidas, se reducirá el número de

Nº proces.	tiempo	acel.	desv.
1	2:27:56	1	0
2	2:26:09	1.01226	0.00527483
3	1:24:11	1.75746	0.0103322
4	1:01:50	2.3928	0.0145569
5	0:49:14	3.00506	0.0192328
6	0:41:58	3.52582	0.0172895
7	0:37:01	3.99611	0.0253283
8	0:32:57	4.49056	0.0233474
9	0:30:15	4.89165	0.0301996
10	0:27:36	5.35925	0.0319129
11	0:25:49	5.72905	0.0481884
12	0:24:03	6.15351	0.0892574
13	0:22:44	6.50879	0.0407916
14	0:21:51	6.7724	0.0835687

Tabla 7.5: Tiempos medios y aceleraciones medias con desviación estándar vs. nº de procesadores (estrategia 2).

cromosomas que dan lugar a sistemas inestables, con lo que los tiempos de evaluación serán mayores y más homogéneos.

7.3.3.3. Estrategia 3.

En la estrategia 3 se incrementan las diferencias con respecto a las dos estrategias anteriores. En la tabla 7.2 se observa que el tiempo empleado en obtener una nueva generación a partir de la anterior (0.29011 seg), es del orden de magnitud del empleado por los esclavos en el proceso de evaluación en las generaciones finales (0.63348 seg) y muy superior al tiempo empleado en las generaciones iniciales, siendo por tanto muy conveniente su paralelización. El diagrama de flujo correspondiente a esta estrategia es el que se muestra en la figura 7.17.

En esta estrategia, el proceso maestro obtendrá únicamente los índices de los padres, y los pasará a los procesos esclavos para que sean éstos los que se encarguen de la obtención de los cromosomas que compondrán la nueva generación y de su posterior evaluación.

La descarga de trabajo que le supone al maestro el hecho de no tener que obtener los cromosomas que componen la población, hace que se incremente su tiempo de inactividad respecto a los de las estrategias 1 y 2. Por tanto, con mayor motivo, se hará

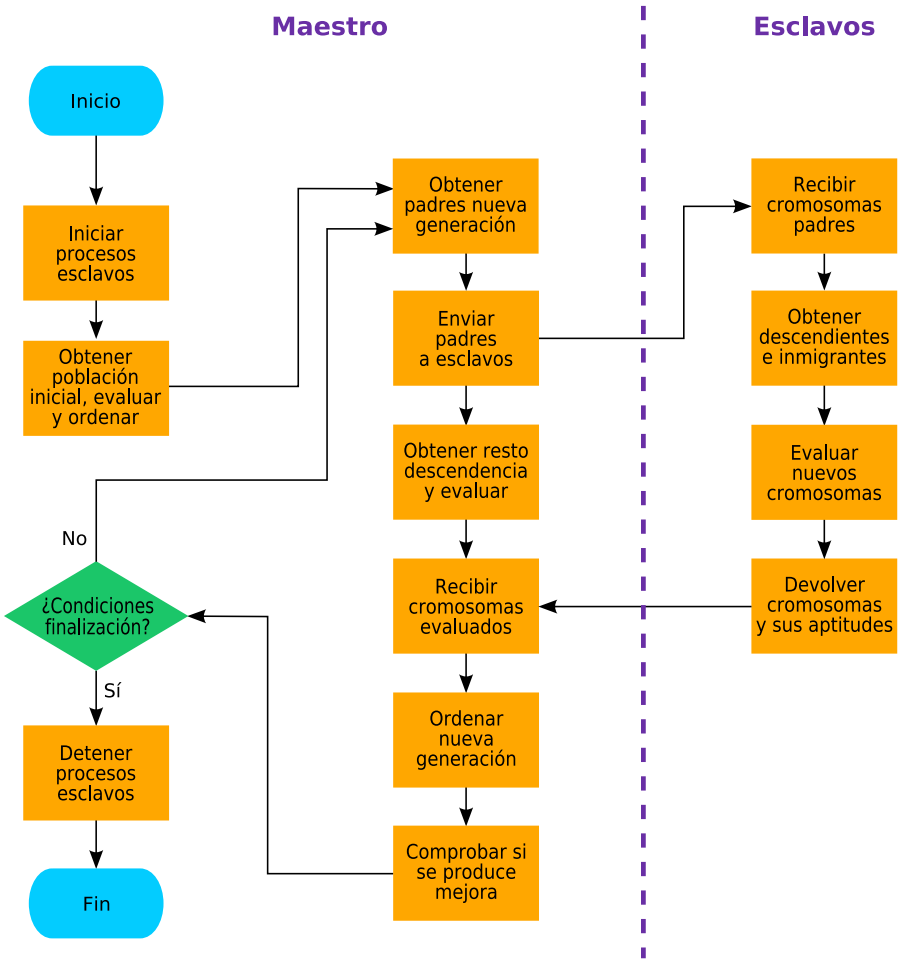


Figura 7.17: Diagrama de flujo correspondiente a la estrategia 3.

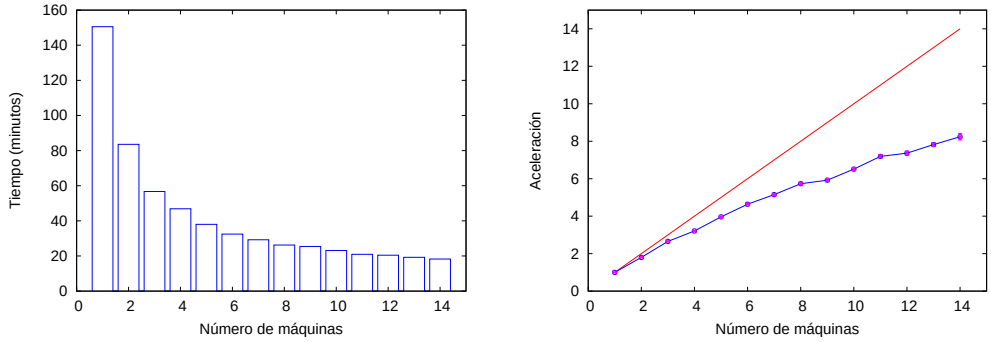
que el maestro se encargue también de evaluar una parte de los cromosomas.

En este caso, los resultados correspondientes a las 4 ejecuciones del programa se recogen en las tablas F.5 y F.6 del apéndice F. En la figura 7.18 se muestran gráficamente los resultados medios obtenidos, y en la tabla 7.6, se encuentran recogidos los valores numéricos.

Comparando los resultados con los que se obtuvieron para la estrategia 1, se observa una mejora significativa en las aceleraciones obtenidas, pasando, para el caso de 14 máquinas, de 7.26966 a 8.23941, lo que significa una mejora del 13.34 % respecto a la primera estrategia.

Los resultados confirman lo que se dedujo del estudio temporal, aliviándose con la

paralelización de la obtención de los cromosomas, el cuello de botella que se formaba.



(a) Tiempo vs. Num. máquinas (b) Aceleración vs. Num. máquinas

Figura 7.18: Tiempo empleado por el algoritmo y aceleración en función del número de máquinas (para estrategia 3).

Nº proces.	tiempo	acel.	desv.
1	2:30:32	1	0
2	1:23:34	1.80134	0.00823289
3	0:56:43	2.65403	0.0209437
4	0:46:52	3.21164	0.0193903
5	0:38:01	3.96004	0.0339853
6	0:32:27	4.6382	0.031533
7	0:29:15	5.14782	0.0399825
8	0:26:15	5.73404	0.0423396
9	0:25:25	5.92358	0.0600083
10	0:23:07	6.51371	0.0645128
11	0:20:56	7.19125	0.067745
12	0:20:26	7.36684	0.110707
13	0:19:15	7.81935	0.0848827
14	0:18:17	8.23941	0.171783

Tabla 7.6: Tiempos medios y aceleraciones medias con desviación estándar vs. nº de procesadores (estrategia 3).

7.3.3.4. Estrategia 4.

La estrategia 4 es la que más diferencias presenta con el resto. Al ser los tiempos de envío y recepción de mensajes (cromosomas y vectores de aptitud) muy inferiores a los tiempos de evaluación, se procurará ahora realizar una distribución más equilibrada de la carga de trabajo. El diagrama de flujo correspondiente a esta estrategia se muestra en la figura 7.19.

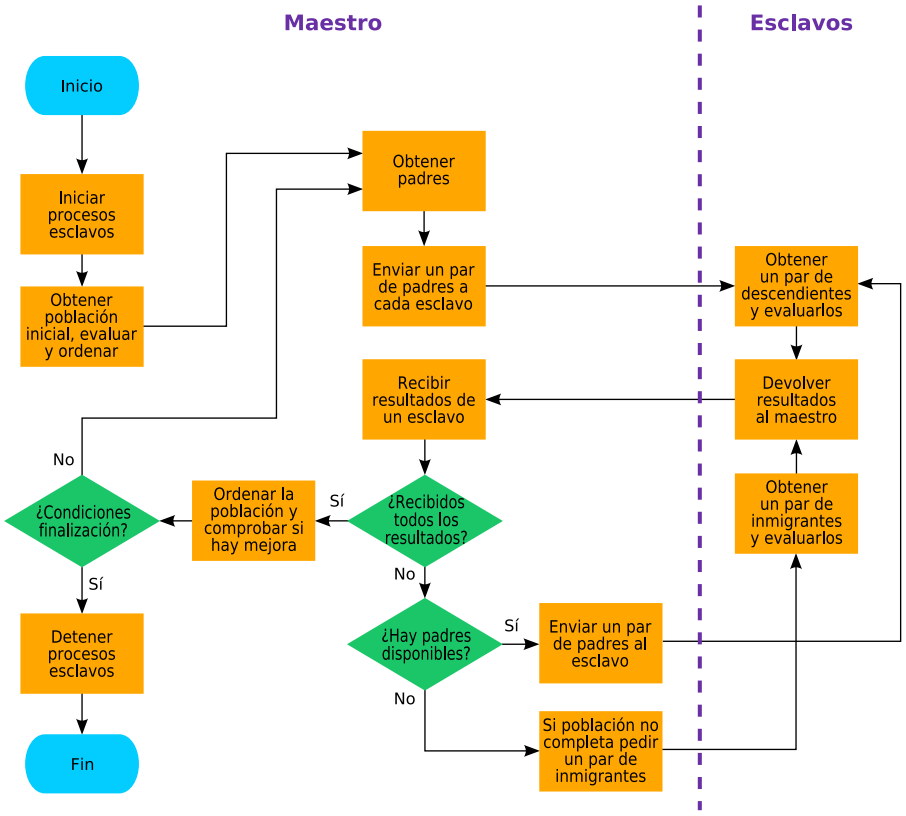


Figura 7.19: Diagrama de flujo correspondiente a la estrategia 4.

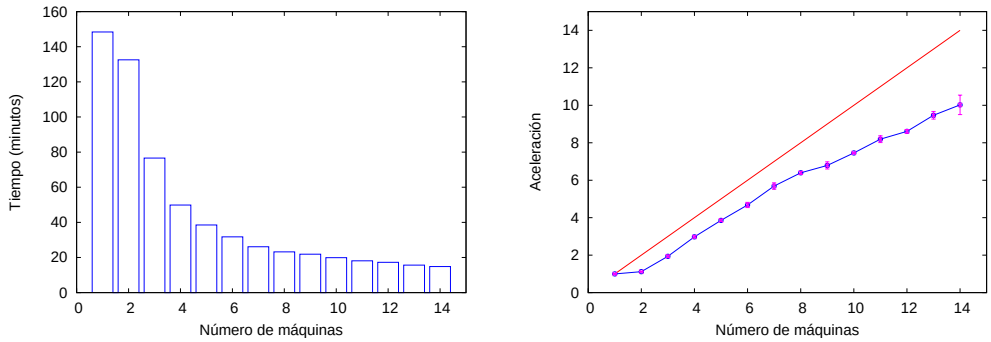
En el diagrama de flujo se observa que el proceso maestro se encargará de iniciar los procesos esclavos en el comienzo de la ejecución del programa. Luego generará la población inicial, la evaluará y la ordenará. Tras tener la población ordenada, obtendrá los índices de los cromosomas que serán los padres de la siguiente generación y se los irá pasando a los procesos esclavos, pero haciéndolo ahora por parejas (que es la cantidad mínima de cromosomas necesaria para llevar a cabo la reproducción). Al comienzo de cada generación pasará una pareja de cromosomas a cada proceso esclavo, pasando luego a la escucha.

El proceso maestro en este caso no evaluará cromosomas, pues se encargará de que los esclavos estén trabajando en todo momento. De esta forma se consigue que los esclavos sólo tengan tiempos de espera tras evaluar los últimos cromosomas que les correspondan en cada generación. Estos tiempos de espera, por otra parte, se han minimizado, ya que como mucho habrán de esperar el tiempo que necesite otro esclavo para crear y evaluar dos cromosomas, más el tiempo que tarde el maestro en iniciar el proceso de la siguiente generación. En las estrategias anteriores, los procesos esclavos

debían evaluar en cada paso un número mayor de cromosomas, por lo que podía haber también tiempos de espera mayores.

Cuando el maestro recibe los resultados de un esclavo, inmediatamente le envía otra pareja de padres para que siga trabajando. Así continuará hasta que haya enviado todos los padres, en cuyo instante pasará a solicitar parejas de inmigrantes a los esclavos que vayan devolviendo resultados, hasta completarse la población.

Los resultados de las 4 ejecuciones del programa para esta estrategia, se encuentran recogidos en las tablas F.7 y F.8 del apéndice F. En la figura 7.20 se muestran gráficamente los resultados medios obtenidos, mientras que la tabla 7.7, recoge los valores numéricos.



(a) Tiempo vs. Num. máquinas (b) Aceleración vs. Num. máquinas

Figura 7.20: Tiempo empleado por el algoritmo y aceleración en función del número de máquinas (para estrategia 4).

Comparando los resultados obtenidos ahora con los que se obtuvieron para la estrategia 1, se observa una importante mejora en las aceleraciones, pasando, para el caso de 14 máquinas, de 7.26966 a 10.023, lo que significa una mejora del 37.87 % respecto a la primera estrategia.

Al principio de esta sección se mencionó que el hecho de que los tiempos empleados en el paso de mensajes fueran muy pequeños en comparación con los tiempos de evaluación, invitaba a intentar un reparto más equilibrado de la carga de trabajo mediante el paso de más mensajes pero más pequeños. De nuevo se confirma con los resultados experimentales lo que se intuía a partir del estudio temporal, y se consiguen con esta última estrategia los mejores resultados.

Nº proces.	tiempo	acel.	desv.
1	2:28:26	1	0
2	2:12:33	1.11976	0.00795886
3	1:16:38	1.93886	0.0611588
4	0:49:52	2.97769	0.0541746
5	0:38:30	3.85578	0.0610961
6	0:31:45	4.68043	0.13802
7	0:26:07	5.6894	0.170272
8	0:23:13	6.39501	0.0752763
9	0:21:52	6.79106	0.19198
10	0:19:55	7.4551	0.0350726
11	0:18:07	8.19288	0.182387
12	0:17:15	8.60917	0.0984071
13	0:15:41	9.4677	0.202807
14	0:14:51	10.023	0.518451

Tabla 7.7: Tiempos medios y aceleraciones medias con desviación estándar vs. nº de procesadores (estrategia 4).

7.4. Conclusiones.

A la hora de resolver problemas de control complejos por medio de algoritmos evolutivos, se necesitará en muchas ocasiones una gran potencia de cálculo. Los ordenadores personales actuales se quedan cortos para abordar esta tarea, por lo que habrá que recurrir a sistemas más potentes y utilizar programación paralela.

Los superordenadores podrían ser la herramienta adecuada, pero su alto coste los convierten en algo prohibitivo en muchos casos. La aparición de *clusters* de ordenadores y en concreto de *clusters* de PCs, ha resultado ser una alternativa muy interesante para muchos organismos sin capacidad para adquirir un superordenador.

En el presente capítulo, se ha comprobado la utilidad de un *clúster* de PCs para reducir el tiempo de ejecución de un algoritmo evolutivo, encargado de realizar la sintonía de un controlador para un avión. Por medio de 14 PCs conectados en red se han conseguido aceleraciones importantes, situadas en torno a 10.

Con anterioridad al diseño del algoritmo se realizó un estudio temporal para determinar las partes del programa secuencial que consumían más tiempo, y por consiguiente aquellas hacia las que habría que dirigir la paralelización. Se utilizaron diferentes estrategias, con diferentes resultados, confirmándose la importancia del estudio temporal previo.

El algoritmo diseñado puede incluirse en otro más general como el utilizado en el capítulo 6, pero aplicado ahora al problema del avión *RCAM*, encargándose del lazo interno descrito en dicho capítulo.

Capítulo 8

Conclusiones

Durante los trabajos que han conducido a la presente tesis, se ha desarrollado una metodología, basada en la utilización de los algoritmos evolutivos, para el diseño de sistemas de control. Dicha metodología ha sido aplicada a distintos problemas, demostrando su eficacia en todos ellos.

Se escogieron los algoritmos evolutivos por sus características intrínsecas, que los hacen un método adecuado para el tratamiento de problemas de optimización multiobjetivo. El hecho de trabajar de forma simultánea con un conjunto de posibles soluciones al problema, permite explorar a la vez distintas regiones del espacio de búsqueda, lo que es interesante cuando se está buscando una solución que debe satisfacer múltiples objetivos, muchos de ellos contrapuestos.

Otras características importantes de los algoritmos evolutivos son su flexibilidad y adaptabilidad al tipo de problema a resolver. La metodología desarrollada permite trabajar directamente con las especificaciones de diseño, e incluir en la función de evaluación todos los parámetros que se consideren oportunos. En los problemas tratados se utilizaron de forma simultánea parámetros propios del dominio temporal y de la frecuencia, sin que el método impusiera ninguna restricción en este sentido. Para obtener un controlador robusto, únicamente hubo que incluir los parámetros necesarios para medir la robustez.

Para llevar a cabo la ordenación de las distintas poblaciones, se utilizó también un método propio: *el método de las prioridades variables*. En problemas de optimización multiobjetivo, las aptitudes de los cromosomas son función de los distintos objetivos considerados; se tienen entonces dos posibilidades: agrupar las medidas sobre los gra-

dos de consecución de cada objetivo en un único valor escalar (mediante una suma ponderada), o trabajar directamente con aptitudes vectoriales. En el último caso surge el problema de cómo tener en cuenta los distintos elementos de los vectores a la hora de comparar aptitudes. El método propuesto fue utilizado en todos los algoritmos diseñados, demostrando un buen comportamiento en este escenario.

Por su parte, la función de densidad de probabilidad variable, también propuesta, supone una mejora notable respecto al uso de una función de densidad de probabilidad uniforme para la determinación de los valores aleatorios propios del método, al menos para el tipo de problemas tratados en esta tesis. En las pruebas que se realizaron se comprobó que utilizando la función de densidad de probabilidad propuesta se lograban cumplir todos los objetivos, mientras que si se utilizaba la función de densidad uniforme, siempre quedaban objetivos sin cumplirse.

La metodología desarrollada fue aplicada a problemas de control concretos, adaptándose el algoritmo en cada caso al problema en cuestión. En primer lugar, se utilizó en conjunción con un método tradicional como el LQ, comprobándose su correcto funcionamiento. Puede por tanto emplearse como complemento a otros métodos que, en alguna fase de su aplicación, requieran la determinación heurística de algún parámetro. Esta vez se utilizó para la determinación de las matrices de peso propias del método LQ, obtenidas normalmente por procedimientos de prueba y error.

Posteriormente se particularizó para un problema en el que el algoritmo debía ser capaz, por sí solo, de realizar la sintonía completa del controlador, sin apoyarse en ningún otro método. El algoritmo diseñado realizó la sintonía completa de un controlador para el autopiloto de un avión comercial. Se trabajó con una estructura de controlador predeterminada y el algoritmo se encargó de la obtención de los parámetros de dicha estructura.

Finalmente, se propuso una metodología para resolver un problema de control más complejo: se debía determinar la estructura (orden) del controlador y realizar su sintonía. El método demostró también su validez, permitiendo la obtención de controladores para distintas condiciones de velocidad y estado de la mar, para la reducción del índice de mareo de un buque de alta velocidad. La complejidad del problema requería la realización de gran cantidad de cálculos, por lo que se tuvo que paralelizar el algoritmo y ejecutarse en una máquina con múltiples procesadores.

La paralelización de algoritmos evolutivos y su ejecución en sistemas con múltiples procesadores, permite abordar problemas más complejos. El alto coste económico que suponen los supercomputadores, ha hecho que los *clúster* de PCs se conviertan en una

buena alternativa para disponer de gran potencia de cálculo a un precio asequible. En el capítulo 7 se ha demostrado que la metodología desarrollada en esta tesis es fácilmente paralelizable, y que con la utilización de un *clúster* de PCs se consiguen reducciones importantes del tiempo de ejecución.

La presente tesis deja abiertas algunas líneas de investigación, enfocadas fundamentalmente a la aplicación de la metodología a problemas más complejos. Lo más inmediato sería adaptar el algoritmo utilizado en la selección y sintonía del controlador del buque de alta velocidad, al problema del avión *RCAM*. Con el *hardware* del que se disponía en su momento, no era viable afrontar un problema con una función de evaluación tan compleja como la del problema del avión; éste fue el motivo por el que se optó por otro problema en el que la función de evaluación fuera menos exigente computacionalmente hablando. Sin embargo, con el *hardware* actual y con el que irá apareciendo próximamente, sí se podrá abordar el problema.

Otra línea de investigación sería la utilización de otros métodos de paralelización, en particular el modelo de islas. Estos métodos –que podrían aplicarse al problema del avión comentado en el párrafo anterior– son intrínsecamente paralelos y no consisten en la paralelización de una algoritmo en serie, como ocurría en el de tipo maestro-esclavo utilizado en la presente tesis. Puede encontrarse información sobre los distintos tipos de algoritmos genéticos paralelos en el estudio de Cantú-Paz [18] o en la revisión de Nowostawski y Poli [92]. Gustafson y Burke, por su parte, exponen en [61] un novedoso modelo de isla para algoritmos evolutivos.

Capítulo 9

Publicaciones

- En el capítulo 4, se realizó la sintonía de un controlador LQ por medio de un algoritmo evolutivo, dando lugar a la publicación:
  - Aranda, J.; De la Cruz, J.M.; Parrilla, M., and Ruipérez, P.: *Design of a linear quadratic optimal control for aircraft flight control by genetic algorithm*, Controlo'2000: 4th Portuguese Conference on Automatic Control. Guimarães-Portugal. October, 2000.
- En el capítulo 5 se realizó la sintonía directa de un controlador con una estructura predeterminada, para la estabilización y el autopiloto de un avión mediante algoritmos evolutivos, dando lugar a la publicación:
  - Aranda, J.; De la Cruz, J.M.; Parrilla, M., and Ruipérez, P.: *Evolutionary algorithm for the design of a multivariable control for an aircraft flight control*, AIAA Guidance Navigation and Control Conference and Exhibit, paper A00-37196. Denver, Colorado, USA, August 2000.
- En el capítulo 6 se realizó la selección de la estructura de un controlador y su sintonía, para la reducción del índice de mareo en un buque de alta velocidad mediante algoritmos genéticos y evolutivos, dando lugar a las publicaciones:
  - Parrilla Sánchez, M., and Aranda Almansa, J.: *A Real Application Example of a Control Structure Selection by Means of a Multiobjective Genetic Algorithm*, 7th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2003 Proceedings, Part II. Lecture Notes in Computer Science, Springer, pp369-376, Maó, Menorca, Spain, June 2003.
  - Parrilla Sánchez, M.; Aranda Almansa, J., and Díaz Martínez, J. M.: *Selection and Tuning of Controller by evolutionary algorithms: Applications*

to fast ferries control, CAMS 2004 IFAC Control Applications in Marine Systems, pp351-356, Ancona, July 2004.

- En el capítulo 7 se realizó un estudio sobre la paralelización en un *clúster* de PCs del algoritmo utilizado en el capítulo 6, dando lugar a la publicación:

  - Parrilla, M.; Aranda, J., and Dormido-Canto, S.: *Parallel Evolutionary Computation: Application of an EA to Controller Design*, Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach: First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005 Proceedings, Part II. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp153-162, Las Palmas, Canary Islands, Spain, June 2005.

Bibliografía

[1] Ahmad, M.; Zhang, L., and Readle, J. C.: *On-line genetic algorithm tuning of a PI controller for a heating system.* GALESIA '97 - Genetic Algorithms in Engineering Systems: Innovations and Applications, pp510-515, 1997.

[2] Anderson, Brian D. O., and Moore, John B.: *Optimal Control. Linear Quadratic Methods.* Prentice-Hall International, 1989.

[3] Andrés-Toro B.; Girón-Sierra J.M.; Fernández-Blanco P.; López-Orozco J.A., and Besada-Portas E.: *Multiobjective optimization and multivariable control of the beer fermentation process with the use of evolutionary algorithms.* Journal of Zhejiang University SCIENCE, pp378-389, 2004.

[4] Aranda, J.; De la Cruz, J. M.; Díaz, J. M., and Ruipérez P.: *Interval Modelling of High Speed Craft for Robust Control.* 2nd International Congress on Maritime Technological Innovations and Research, ISBN:84-7786-670-8, pp195-207, 2000.

[5] Aranda, J.; De la Cruz, J. M.; Parrilla, M., and Ruipérez P.: *Evolutionary algorithms for the design of a multivariable control for an aircraft flight control.* Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit. Denver, August 2000.

[6] Aranda, J.; De la Cruz, J. M.; Parrilla, M., and Ruipérez P.: *Design of a linear quadratic optimal control for aircraft flight control by genetic algorithms.* Proceedings of Controllo'2000: 4th Portuguese Conference on Automatic Control. 2000.

[7] Aranda, J.; De la Cruz, J. M.; Díaz, J. M.; De Andrés, B.; Ruipérez P., and Girón J.M: *Modelling of a high speed craft by a non-linear least squares method with constraints.* Manoeuvring and Control of Marine Craft 2000 Proceedings of the 5th IFAC Conference. Edited by M. Blanke, M.M.A. Pourzajani, Z.Z. Vukic, Pergamon Press, ISBN:0-08-043659-5, 2001.

[8] Aranda, J.; De la Cruz, J. M.; Díaz, J. M.: *Identification of Multivariable Models of Fast Ferries.* European Journal of Control, Vol. 10, n. 2, pp187-198, May 2004.

[9] Aranda, J.; De la Cruz, J. M.; Díaz: *Design of a multivariable robust controller to decrease the motion sickness incidence in fast ferries.* Control Engineering Practice, vol 13/8, pp985-999, 2005.

[10] Aranda, J.; Muñoz, R.; Dormido Canto, S.; Díaz, J. M., and Dormido Bencomo, S.: *An analysis of models identification methods for high speed crafts.* Journal of Maritime Research, Vol.II, No. 1, pp51-67, 2005.

[11] Athans, M., and Falb, P.: *Optimal Control.* New York: McGraw-Hill, 1966.

[12] Baldomero, J. F.: *PVMTB: Parallel Virtual Machine ToolBox.* II Congreso de Usuarios Matlab'99, Dpto. de Informática y Automática, UNED, pp523-532, Madrid, 1999.

[13] Barrido, S. C., and Dadios, E. P.: *Online robot tracking using genetic algorithms.* Proceedings of the IEEE International Symposium on Intelligent Control, pp479-484, 2002.

[14] Becker, D. J.; Sterling, T.; Savarese, D.; Dorband, J. E.; Ranawake, U. A., and Packer, C. V.: *Beowulf: A Parallel Workstation for Scientific Computation.* Proceedings of the 1995 International Conference on Parallel Processing, 1995.

[15] Ben-Tal, A.: *Characterization of Pareto and lexicographic optimal solutions.* In Fandel and Gal, pp1-11, 1980.

[16] Billings, S. A., and Mao, K. Z.: *Structure detection for nonlinear rational models using genetic algorithms.* International Journal of Systems Science, Vol. 29, No. 3, pp223-231, 1998.

[17] Blight, J. D.; Dailey, R. L., and Gangsaas, D.: *Practical control law design for aircraft using multivariable techniques.* Int. J. Contr., vol. 59, pp. 93-137, 1994.

[18] Cantú-Paz, E.: *A Survey of Parallel Genetic Algorithms.* Calculateurs Paralleles, Reseaux et Systems Repartis, Paris: Hermes, 10(2): 141-171, 1998.

[19] Chang, C. S, and Sim, S. S.: *Optimising train movements through coast control using genetic algorithms.* IEEE Proceedings - Electric power applications, Vol. 144, No. 1, January 1997.

[20] Charnes, A., and Cooper, W. W.: *Management Models and Industrial Applications of Linear Programming.* volume 1, John Wiley, New York, 1961.

[21] Chen, B. S., and Cheng, Y. M.: *A Structure-Specified H-Infinity Optimal Control Design for Practical Applications: A Genetic Approach.* IEEE Transactions on Control Systems Technology, Vol. 6, pp707-718, November 1998.

[22] Choi, S. H.; Lee, C. O., and Cho, H. S.: *Friction compensation control of an electropneumatic servovalve by using an evolutionary algorithm.* Proceedings of the Institution of Mechanical Engineers, Vol. 214, Part I, pp173-184, 2000.

[23] Chow, K. M., and Rad, A. B.: *On-line fuzzy identification using genetic algorithms.* Fuzzy Sets and Systems, Vol. 132, Issue 2, pp147-171, December 2002.

[24] Coello Coello, Carlos A.: *A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques.* Knowledge and Information Systems. An International Journal, 1(3): 269-308, August 1999.

[25] Coello Coello, Carlos A.: *Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art.* Computer Methods in Applied Mechanics and Engineering, 191, 1245-1287, 2002.

[26] Coello Coello, C. A.; Van Veldhuizen, D. A, and Lamont, G. B.: *Evolutionary algorithms for solving multi-objective problems.* New York: Kluwer Academic, 2002.

[27] Coello Coello, C. A.; Hernández Aguirre, Arturo, and Zitzler, Eckart (Eds.): *Evolutionary Multi-Criterion Optimization.* Proceedings of the Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Series: Lecture Notes in Computer Science, Vol. 3410. Springer. 2005.

[28] Cupertino, F.; Mininno, E.; Naso, D.; Turchiano, B., and Salvatore, L.: *On-line genetic optimization of unstructured controllers for electric drives.* Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE 2002, Vol. 1, pp347-352, 2002.

[29] Cupertino, F.; Giordano, V.; Naso, D.; Turchiano, B., and Salvatore, L.: *On-line genetic design of fuzzy controllers for DC drives with variable load.* Electronics Letters, Vol. 39, Issue 5, pp479-480, March 2003.

[30] Cupertino, F.; Mininno, E.; Naso, D.; Turchiano, B., and Salvatore, L.: *On-line genetic design of anti-windup unstructured controllers for electric drives with variable load.* IEEE Transactions on Evolutionary Computation, Vol. 8, Issue 4, pp347-364, August 2004.

[31] Dailey, R. L.: *Lecture Notes for the Workshop on H_{inf} and μ Methods for Robust Control.* IEEE Conference on Decision and Control, Brighton, U.K., 1991.

[32] Davis, Lawrence: *Adapting operator probabilities in genetic algorithms.* In J. David Schaffer (ed.), Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, Calif.: Morgan Kaufmann Publishers, 1989.

[33] Davis, L., (Editor): *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[34] Deb, K.: *Multi-objective optimization using evolutionary algorithms.* Chichester, New York: John Wiley & Sons, 2001.

[35] De Jong, Kenneth A.: *An Analysis of the Behavior of a class of Genetic Adaptive Systems.* University of Michigan, Ph. D. Dissertation, 1975.

[36] De la Cruz, J.M.; Ruipérez, P., and Aranda, J.: *RCAM Design Challenge Presentation Document: an Eigenstructure Assignment Approach.* GARTEUR/TP-088-22, April, 1997.

[37] De la Cruz, J.M.; Ruipérez, P., and Aranda, J.: *An Eigenstructure Assignment Approach (2).* Lecture Notes in Control and Information Sciences no. 224, pp238-257, 1997.

[38] De la Cruz, J.M.; Aranda, J.; Girón Sierra, J. M.; Velasco, F.; Esteban S.; Díaz, J. M.; de Andrés Toro, B.: *Improving the Comfort of a Fast Ferry, Smoothing a ship's vertical motion with the control of flaps and T-foil.* IEEE Control System Magazine, pp47-60, April 2004.

[39] Dessalegne, T., and Nicklow, J. W.: *Evolutionary Algorithms for Optimal Control of the Illinois Waterway.* World Water and Environmental Resources Congress 2003. Paul Bizier, Paul DeBarry-Editors, Philadelphia, Pennsylvania, USA, June 2003.

[40] Díaz, J. M.; Dormido, S., and Aranda, J.: *Interactive computer-aided control design using Quantitative Feedback Theory: The problem of vertical movement stabilization on a high-speed ferry.* International Journal of Control, Vol 78, issue 11, pp793-805, 2005.

[41] Dormido Canto, S.: *Programación Dinámica Paralela: Aplicación a Problemas de Control*. Tesis Doctoral, Dept. Informática y Automática, UNED, Diciembre 2001.

[42] Dormido Canto, S.; De Madrid, A. P., and Dormido Bencomo, S.: *Parallel Dynamic Programming on Clusters of Workstations*. IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 9, pp785-798, September 2005.

[43] Echtle, K., and Eusgeld, I.: *A Genetic Algorithm for Fault-Tolerant System Design*. Lecture Notes in Computer Science, ISSU 2847, pp197-213, Springer-Verlag, 2003.

[44] Eiben, A. E., and Smith, J. E.: *Introduction to Evolutionary Computing*. Natural Computing Series, Springer, November 2003.

[45] Erickson, M.; Mayer, A., and Horn, J.: *Multi-objective optimal design of groundwater remediation systems: application of the niched Pareto genetic algorithm (NPGA)*. Advances in Water Resources, Vol. 25, No. 1, pp51-65, January 2002.

[46] Esteban, S.; De la Cruz, J. M.; Girón-Sierra, J.M.; De Andrés, B.; Díaz, J.M., and Aranda, J.: *Fast Ferry vertical accelerations reductions with active flaps and T-foil*. Manoeuvring and Control of Marine Craft 2000 Proceedings of the 5th IFAC Conference. Edited by M. Blanke, M.M.A. Pourzajani, Z.Z. Vukic. Pergamon Press. ISBN:0-08-043659-5. 2001.

[47] Esteban, S.: *Modelado y control del movimiento longitudinal de un ferry de alta velocidad*. Tesis Doctoral, Universidad Complutense de Madrid, 2002.

[48] Fadali, M. S.; Zhang, Y., and Louis S. J.: *Robust Stability Analysis of Discrete-Time Systems Using Genetic Algorithms*. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 29, No. 5, pp503-508, September 1999.

[49] Farley, D. L.: *Performance Comparison of Mainframe, Workstations, Clusters, and Desktop Computers*. NASA/TM-2005-213505, Langley Research Center, Hampton, Virginia, January 2005.

[50] Fleming, P. J., and Purshouse, R. C.: *Evolutionary algorithms in control systems engineering: a survey*. Control Engineering Practice 10 (2002) 1223-1241. Published by Elsevier Science Ltd. 2002.

[51] Flynn, M. J.: *Some computer organizations and their effectiveness*. IEEE Transactions on Computing, Vol. C-21, pp948-960, 1972.

[52] Fogel, G. B.; Weekes, D. G.; Sampath, R., and Ecker, D. J.: *Parameter optimization of an evolutionary algorithm for RNA structure discovery.* Congress on Evolutionary Computation, 2004. CEC2004, Vol. 1, pp607- 613, June 2004.

[53] Fonseca, C. M.; Mendes, E. M.; Fleming, P. J., and Billings, S. A.: *Non-linear model term selection with genetic algorithms.* IEE/IEEE Workshop on Natural Algorithms in Signal Processing (Essex, UK), Vol. 2, pp27/1-27/8, 1993.

[54] Fonseca, Carlos M., and Fleming, Peter J.: *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-Part I: A Unified Formulation and Part II: Application Example.* IEEE Transactions on Systems. Man and Cybernetics. Part A: Systems and Humans. Vol. 28. No. 1, pp26-37 and pp38-47, January 1998.

[55] GARTEUR (Group for Aeronautical Research and Technology in EURope): *Robust Flight Control Design Challenge Problem Formulation and Manual: the Research Civil Aircraft Model (RCAM).* February 1997.

[56] Ge, P.; Wang, N., and Lu, S.: *An Evolutionary Modeling Approach for Automotive Bumper System Design and Analysis.* Journal of Computing and Information Science in Engineering, Volume 2, Issue 3, pp141-149, September 2002.

[57] Goldberg, David E.: *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[58] Goldberg, D. E., Sastry, K.: *A Practical Schema Theorem for Genetic Algorithms Design and Tuning.* Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), 2001.

[59] Grefenstette, John J.: *Optimization of control parameters for genetic algorithms.* IEEE Transactions on Systems, Man and Cybernetics, SMC-16(1), 1986.

[60] Grefenstette, J.J.: *Genetic algorithms for changing environments, in Parallel Problem Solving from Nature 2.* R. Manner and B. Manderick, Eds. Amsterdam, The Netherlands: North-Holland, pp137-144, 1992.

[61] Gustafson, S., and Burke E. K.: *A Niche for Parallel Island Models: Outliers and Local Search.* Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05) pp612-619, 2005.

[62] Haijun, Y.; Minqiang, L., and Jisong, K.: *Exact schema theorem based on the space of schema.* IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, pp349-354, 2003.

[63] Holland, John H. *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.

[64] Horn, J., and Nafpliotis, N.: *Multiobjective Optimization using the Niched Pareto Genetic Algorithm.* Technical Report IlliGAL Report 93005. University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1995.

[65] Ichikawa, Y., and Sawa, T.: *Neural Network Application for Direct Feedback Controllers.* IEEE Transactions on Neural Networks, Vol. 3, No. 2, pp224-231, March 1992.

[66] Iglesias Otero, M. T.: *Algoritmos genéticos generalizados: Variaciones sobre un tema.* Universidade da Coruña. Servicio de Publicacións. Setembro de 1998.

[67] Ijiri, Y.: *Management Goals and Accounting for Control.* North-Holland. Amsterdam, 1965.

[68] Janikow, C., and Michalewicz, Z.: *Specialized Genetic Algorithms for Numerical Optimization Problems.* Proceedings of the International Conference on Tools for AI, pp798-804, 1990.

[69] Jonsson, R., and Malec, J.: *Towards computing the parameters of the Simple Genetic Algorithm.* Proceedings of the 2001 Congress on Evolutionary Computation CEC2001, 2001.

[70] Juang, J.: *Application of Genetic Algorithm and Recurrent Network to Nonlinear System Identification.* IEEE Conference on Control Applications, Istambul, Turkey, June 2003.

[71] Kristinsson, K., and Dumont, G. A.: *System Identification and Control Using Genetic Algorithms.* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 5, pp1033-1046, September/October 1992.

[72] Kumon, T.; Suzuki, T.; Iwasaki, M.; Matsuzaki, M.; Matsui, N., and Okuma, S.: *System identification using a genetic algorithm and its application to internal adaptive model control.* Electrical Engineering in Japan, Volume 142, Issue 4, pp45-55, Wiley Periodicals, Inc., A Wiley Company, 2003.

[73] Lehtomaki, N. A.; Sandell, N. R., and Athans, M.: *Robustness Results in Linear-Quadratic Gaussian Based Multivariable Control Designs.* IEEE Trans. Automatic Control, vol. AC-26, n 1, pp75-93, 1981.

[74] Lennon, W. K., and Passino, K. M.: *Intelligent Control for Brake Systems.* IEEE Transactions on Control Systems Technology, Vol. 7, No. 2, pp188-202, March 1999.

[75] Leontaris, I. J., and Billings, S. A.: *Input Output Parametric Models for Non-Linear Systems, Part 1: Deterministic Non-Linear Systems* and *Part 2: Stochastic Non-Linear Systems.* International Journal of Control, Vol. 41, No. 2, pp303-328 and pp329-344, 1985.

[76] Lewis, Frank L.: *Optimal Control.* Wiley-Interscience Publication, 1986.

[77] Lin, H., and Wang, Y.: *A Novel Multiobjective Evolutionary Algorithm Based on Min-Max Strategy.* Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Computer Science, Volume 2690, pp361-368, 2003.

[78] Linkens, D. A., and Nyongesa, H. O.: *Genetic algorithms for fuzzy control - Part 1: Offline system development and application and Part 2: Online systems development and application.* IEE Proceedings - Control Theory and Applications, Vol. 142, No. 3, pp161-176 and pp177-185, May 1995.

[79] Lloyd, A. R. J. M.: *Seakeaping: ship behaviour in rough water.* Ellis Horwood Limited, 1989.

[80] Luh, G. C., and Wu, C. Y.: *Non-linear system identification using genetic algorithms.* Proceedings of the Institution of Mechanical Engineers, Vol. 213, Part I, pp105-118, 1999.

[81] Magni, J. F.; Bennani, S., and Terlouw, J. (editors): *Robust Flight Control: A Design Challenge.* Lecture Notes in Control and Information Sciences, 224. Springer-Verlag, 1997.

[82] Marcu, T.: *A multiobjective evolutionary approach to pattern recognition for robust diagnosis of process faults.* SAFEPROCESS '97: Fault Detection, Supervision and Safety for Technical Processes, Vols. 1-3, pp1183-1188, 1997.

[83] Marrison, C. I., and Stengel, R. F.: *Robust Control System Design Using Random Search and Genetic Algorithms.* IEEE Transactions on Automatic Control, Vol. 42, No. 6, pp835-839, June 1997.

[84] Matsuura, K.; Shiba, H.; Hirotsune, M., and Nunokawa, Y.: *Optimal control of sensory evaluation of the sake mashing process.* Journal of Process Control, Vol. 6, No. 5, pp323-326, 1996.

[85] Michalewicz, Z.; Vignaux, G. A.; Hobbs, M.: *A Non-Standard Genetic Algorithm for the Nonlinear Transportation Problem.* ORSA Journal on Computing, Vol. 3, No. 4, pp307-316, 1991.

[86] Michalewicz, Z.; Janikov, C., and Krawczyk, J.: *A Modified Genetic Algorithm for Optimal Control Problems.* Computers & Mathematics with Applications, Vol. 23, No. 12, pp83-94, 1992.

[87] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1996.

[88] Michalewicz, Z.: *A survey of constraint handling techniques in evolutionary computation methods.* In Proceedings of the 4th Annual Conference on Evolutionary Programming. Cambridge, MA, pp135-155, 1995.

[89] Mosca, E.: *Optimal, Predictive, and Adaptive Control.* Prentice Hall, 1995.

[90] Motoki, T.: *Calculating the expected loss of diversity of selection schemes.* Evolutionary Computation, pp397-422, 2002.

[91] Naeem W.; Sutton R.; Chudley J.; Dalgleish F. R., and Tetlow S.: *An online genetic algorithm based model predictive control autopilot design with experimental verification.* International Journal of Control, Taylor & Francis publishers, Vol. 78, number 14, pp1076-1090, September 2005.

[92] Nowostawski, M., and Poli, R.: *Parallel Genetic Algorithm Taxonomy.* Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES'99), pp88-92, 1999.

[93] Ogata, K.: *Ingeniería de Control Moderna.* Prentice-Hall, 4ª edición, 2003.

[94] Oliveira, P.; Sequeira, J., and Sentieiro, J.: *Selection of Controller Parameters using Genetic Algorithms.* Engineering Systems with Intelligence. Concepts, Tools, and Applications, Kluwer Academic Publishers, Dordrecht, Netherlands, pp431-438, 1991.

[95] Onnen, C.; Babuska, R.; Kaymak, U.; Sousa, J. M.; Verbruggen, H. B., and Isermann, R.: *Genetic Algorithms for optimization in predictive control.* Control Engineering Practice, Vol. 5, Iss. 10, pp1363-1372, 1997.

[96] Osyczka, A.: *An approach to muticriterion optimization problems for engineering design.* Computer Methods in Applied Mechanics and Engineering, 15:309-333, 1978.

[97] Osyczka, A.: *An approach to muticriterion optimization for structural design.* In Proceedings of International Symposium on Optimal Structural Design. University of Arizona, 1981.

[98] Osyczka, A.: *Muticriterion Optimization in Engineering with FORTRAN programs.* Ellis Horwood Limited, 1984.

[99] Ouyang, P. R.; Zhang, W. J., and Gupta M. M.: *A Robust PD-Type Evolutionary Learning Control for Nonlinear Time-varying Systems.* Proceedings of the 2004 IEEE International Symposium on Intelligent Control. Taipei, Taiwan, September 2004.

[100] Pacheco, P.S.: *Parallel programming with MPI.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1997.

[101] Painton, L., and Campbell, J.: *Genetic Algorithms in Optimization of System Reliability.* IEEE Transactions on Reliability, Vol. 44, No. 2, pp172-178, June 1995.

[102] Pan, F.; Han, R., and Feng, D.: *An Identification Method of Time-Varying Delay Based on Genetic Algorithm.* Proceedings of the Second International Conference on Machine Learning and Cybernetics, Wan, November 2003.

[103] Pareto, Vilfredo: *Cours D'Economie Politique.* Volume I and II. F. Rouge, Lausanne, 1896.

[104] Patton, R. J.; Chen, J., and Liu, G. P.: *Robust fault detection of dynamical systems via genetic algorithms.* Proceedings of the Institution of Mechanical Engineers, Vol. 211, Part I, pp357-364, 1997.

[105] Pongcharoen, P.; Hicks, C.; Braiden, P. M., and Stewardson, D. J.: *Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products.* International Journal of Production Economics 78, pp311-322, Elsevier, 2002.

[106] Rao, S.: *Game theory approach for multiobjective structural optimization.* Computers and Structures, 25(1):119-127, 1986.

[107] Ritzel, B. J.; Wayland, J., and Ranjithan, S.: *Using genetic algorithms to solve a multiple objective groundwater pollution containment problem.* Water Resources Research, 30(5):1589-1603, May 1994.

[108] Rocha, M.; Neves, J.; Rocha, I., and Ferreira, E. C.: *Evolutionary Algorithms for Optimal Control in Fed-Batch Fermentation Processes.* Applications of Evolutionary Computing: EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoISAP, EvoMUSART, and EvoSTOC, Coimbra, Portugal. Proceedings. Publisher: Springer-Verlag GmbH. April 2004.

[109] Rosemberg, R. S.: *Simulation of genetic populations with biochemical properties.* PhD thesis, University of Michigan, Ann Harbor, Michigan, 1967.

[110] Santander, A., and Aranda, J.: *QFT for the Design of an Aircraft Flight Control.* IFAC World Congress. 2005.

[111] Schaffer, J. D.: *Multiple objective optimization with vector evaluated genetic algorithms.* In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp93-100. Lawrence Erlbaum, 1985.

[112] Schaffer, J. David; Caruana, Richard A.; Eshelman, Larry J., and Das Rajarshi: *A study of control parameters affecting online performance of genetic algorithms for function optimization.* In J. David Schaffer (ed.), Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, Calif.: Morgan Kaufmann Publishers, 1989.

[113] Schubert, W. M., and Stengel, R. F.: *Parallel Synthesis of Robust Control Systems.* IEEE Transactions on control Systems Technology, Vol. 6, pp701-706, November 1998.

[114] Schwefel, H.-P.: *Evolution Strategies: A Family of Non-Linear Optimization Techniques Based on Imitating Some Principles of Organic Evolution.* Annals of Operations Research, Vol.1, pp165-167, 1984.

[115] Srinivas, N., and Deb, K.: *Multiobjective optimization using nondominated sorting in genetic algorithms.* Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India. 1993.

[116] Stirrup, R., and Chipperfield, A. J.: *Hybrid Control and Evolutionary Decision Support within a Sustainable Environment.* Proceedings of the IEEE International Symposium on Intelligent Control. Vancouver, Canada, October 2002.

[117] Teng T. K.; Shieh J. S., and Chen C. S.: *Genetic algorithms applied in online autotuning PID parameters of a liquid-level control system.* Transactions of the

Institute of Measurement and Control, Volume 25, Number 5, pp. 433-450(18), December 2003.

[118] Tseng, C. H., and Lu, T. W.: *Minimax multiobjective optimization in structural design.* International Journal for Numerical Methods in Engineering, 30:1213-1228, 1990.

[119] Tzes, A.; Peng, P. Y., and Guthy, J.: *Genetic-Based Fuzzy Clustering for DC-Motor Friction Identification and Compensation.* IEEE Transactions on Control Systems Technology, Vol. 6, No. 4, pp462-472, July 1998.

[120] Varsek, A.; Urbancic, T., and Fillipic, B.: *Genetic Algorithms in Controller Design and Tuning.* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 5, pp1330-1339, September/October 1993.

[121] Vieira, D. A. G.; Adriano, R. L. S.; Vasconcelos, J. A., and Krähenbühl, L.: *Treating Constraints as Objectives in Multiobjective Optimization Problems Using Niched Pareto Genetic Algorithm.* IEEE Transactions on Magnetics, Vol. 40, No. 2, March 2004.

[122] Vlachos, C.; Williams, D., and Gomm, J. B.: *Genetic approach to decentralised PI controller tuning for multivariable processes.* IEEE Proceedings - Control Theory and Applications, Vol. 146, No. 1, pp58-64, January 1999.

[123] Wang, P., and Kwok, D. P.: *Autotuning of Classical PID Controllers Using an Advanced Genetic Algorithm.* International Conference on Industrial Electronics, Control, Instrumentation and Automation (IECON 92), Vol. 3, pp1224-1229, 1992.

[124] Watanabe, K., and Hashem, M. M. A.: *Evolutionary Computations.* Studies in Fuzziness and Soft Computing, April 2004.

[125] Wieczorek, W., and Czech, Z. J.: *Selection Schemes in Evolutionary Algorithms.* Proceedings of the IIS'2002, Symposium on Intelligent Information Systems, pp185-194, 2002.

[126] Yaniv, O.: *Quantitative Feedback Design of Linear and Nonlinear Control Systems.* Kluwer Academic Publishers, 1999.

[127] Zadeh, L. A.: *Optimality and Non-Scalar-Valued Performance Criteria.* IEEE Transactions on Automatic Control, AC-8, pp59-60, 1963.

[128] Zhang, T.-z.; Teng, C.-x., and Han, Z.-g.: *Application of genetic algorithms to system reliability optimization.* Control and Decision, Vol. 17; part 3, pp378-380, 2002.

[129] Zitzler, E.; Laumanns, M., and Bleuler, S.: *A Tutorial on Evolutionary Multiobjective Optimization.* Workshop on Multiple Objective Metaheuristics (MOMH 2002), Springer-Verlag, Berlin, Germany, 2004.

Apéndice A

Ecuaciones en el espacio de estados del modelo RCAM linealizado

A partir del modelo proporcionado se obtuvo un modelo lineal para las siguientes condiciones de funcionamiento: velocidad respecto al aire $V_A = 80m/s$, altitud $h = 1000m$, masa $m = 120000kg$, centro de gravedad $cgx = 0.23$, $cgz = 0.1$ y retardo en el transporte $\delta = 0$. Una vez obtenido el modelo lineal, se separaron las dinámicas longitudinal y lateral para trabajar con ellas de forma independiente, y se añadieron las dinámicas de los actuadores. Los modelos correspondientes se muestran a continuación.

A.1. Modelo para la dinámica longitudinal.

Para la dinámica longitudinal del avión, se obtuvieron las ecuaciones A.1 y A.2 que corresponden al vector de estados y al vector de salidas respectivamente.

$$\begin{pmatrix} \dot{q} \\ \dot{\theta} \\ \dot{u}_B \\ \dot{w}_B \\ \dot{X}_T \\ \dot{\chi}_{TH} \end{pmatrix} = \begin{pmatrix} -0.981706 & 0 & -0.000728686 & -0.0152927 & -2.43599 & 0.613125 \\ 1.0 & 0 & 0 & 0 & 0 & 0 \\ -2.23433 & -9.77542 & -0.0325234 & 0.0743948 & 0.18712 & 19.62 \\ 77.3559 & -0.772662 & -0.226086 & -0.668474 & -6.47836 & 0 \\ 0 & 0 & 0 & 0 & -6.66667 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.666667 \end{pmatrix} \begin{pmatrix} q \\ \theta \\ u_B \\ w_B \\ X_T \\ \chi_{TH} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 6.66667 & 0 \\ 0 & 0.666667 \end{pmatrix} \begin{pmatrix} d_T \\ d_{TH} \end{pmatrix} \tag{A.1}$$

$$\begin{pmatrix} q \\ n_x \\ n_z \\ w_V \\ V_A \end{pmatrix} = \begin{pmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0.00768699 & 0 & -0.00331533 & 0.00758356 & 0.0190744 & 2.0 \\ -0.266135 & 0 & -0.0230465 & -0.0681421 & -0.660384 & 0 \\ 0 & -79.8667 & -0.0288718 & 0.999583 & 0 & 0 \\ 0 & 0 & 0.999584 & 0.0294977 & 0 & 0 \end{pmatrix} \begin{pmatrix} q \\ \theta \\ u_B \\ w_B \\ X_T \\ \chi_{TH} \end{pmatrix} \tag{A.2}$$

A.2. Modelo para la dinámica lateral.

Por su parte, las ecuaciones A.3 y A.4 corresponden al vector de estados y el vector de salidas para la dinámica lateral.

$$\begin{pmatrix} \dot{p} \\ \dot{r} \\ \dot{\phi} \\ \dot{\psi} \\ \dot{v}_B \\ \dot{X}_A \\ \dot{X}_R \end{pmatrix} = \begin{pmatrix} -1.26673 & 0.549836 & 0 & 0 & -0.0214443 & -0.840226 & 0.256775 \\ 0.0521529 & -0.520707 & 0 & 0 & 0.00455557 & -0.0175929 & -0.333225 \\ 1.0 & 0.0288838 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.00042 & 0 & 0 & 0 & 0 & 0 \\ 2.30974 & -79.9666 & 9.78958 & 0 & -0.169867 & 0 & 2.0384 \\ 0 & 0 & 0 & 0 & 0 & -6.66667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3.33333 \end{pmatrix} \begin{pmatrix} p \\ r \\ \phi \\ \psi \\ v_B \\ X_A \\ X_R \end{pmatrix}$$

$$+ \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 6.66667 & 0 \\ 0 & 3.33333 \end{pmatrix} \begin{pmatrix} d_A \\ d_R \end{pmatrix} \tag{A.3}$$

$$\begin{pmatrix} \beta \\ p \\ r \\ \phi \\ \chi \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.0125 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0288237 & 1.0 & 0.0125 & 0 & 0 \\ 0 & 0 & -2.3059 & 79.8667 & 1.0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p \\ r \\ \phi \\ \psi \\ v_B \\ X_A \\ X_R \end{pmatrix} \tag{A.4}$$

Apéndice B

Funciones de transferencia del modelo CRIBAV linealizado

Las funciones de transferencia que se muestran a continuación corresponden al modelo de la figura 3.6.

B.1. Funciones de transferencia del modelo linealizado para una velocidad de 20 nudos.

Las funciones de transferencia de los distintos bloques del modelo serán:

$$G_{ij} = \frac{N20_{ij}}{D20_{ij}} \qquad \forall (i,j) \ \in \ \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3)\}$$

siendo los numeradores y denominadores correspondientes a estas funciones de transferencia los siguientes:

$$\begin{aligned}
N20_{11} &= 0.1689s^6 + 1.603s^5 + 1.188s^4 + 12.28s^3 + 17.13s^2 + 31.05s + 13.07 \\
D20_{11} &= s^8 + 118.8s^7 + 1428s^6 + 5133s^5 + 1.284 \cdot 10^4 s^4 + 2.236 \cdot 10^4 s^3 + 2.542 \cdot 10^4 s^2 + 2.071 \cdot 10^4 s + 6130 \\
\\
N20_{12} &= -0.1621s^6 - 1.584s^5 - 0.7848s^4 - 12.64s^3 - 14.43s^2 - 30.1s - 12.55 \\
D20_{12} &= s^8 + 118.8s^7 + 1428s^6 + 5133s^5 + 1.284 \cdot 10^4 s^4 + 2.236 \cdot 10^4 s^3 + 2.542 \cdot 10^4 s^2 + 2.071 \cdot 10^4 s + 6130 \\
\\
N20_{13} &= 3.956s^{13} + 48.43s^{12} + 124.4s^{11} + 715.1s^{10} + 1606s^9 + 4761s^8 + 8806s^7 + 1.531 \cdot 10^4 s^6 + \\
&\quad + 1.981 \cdot 10^4 s^5 + 1.935 \cdot 10^4 s^4 + 1.343 \cdot 10^4 s^3 + 4911s^2 + 9.578s - 309.3 \\
D20_{13} &= s^{15} + 120.4s^{14} + 1626s^{13} + 7983s^{12} + 2.784 \cdot 10^4 s^{11} + 7.032 \cdot 10^4 s^{10} + 1.396 \cdot 10^5 s^9 + 2.197 \cdot 10^5 s^8 + \\
&\quad + 2.772 \cdot 10^5 s^7 + 2.802 \cdot 10^5 s^6 + 2.228 \cdot 10^5 s^5 + 1.365 \cdot 10^5 s^4 + 6.109 \cdot 10^4 s^3 + 1.811 \cdot 10^4 s^2 + \\
&\quad + 3246s + 309.3
\end{aligned}$$

$$N20_{21} = -0.003887s^6 - 0.4549s^5 - 1.273s^4 + 19.7s^3 + 9.454s^2 + 70.19s + 1.115$$
$$D20_{21} = s^8 + 118.8s^7 + 1428s^6 + 5133s^5 + 1.284 \cdot 10^4 s^4 + 2.236 \cdot 10^4 s^3 + 2.542 \cdot 10^4 s^2 + 2.071 \cdot 10^4 s + 6130$$

$$N20_{22} = -0.005237s^6 - 0.5485s^5 - 2.202s^4 + 27.66s^3 + 10.12s^2 + 94.54s + 1.502$$
$$D20_{22} = s^8 + 118.8s^7 + 1428s^6 + 5133s^5 + 1.284 \cdot 10^4 s^4 + 2.236 \cdot 10^4 s^3 + 2.542 \cdot 10^4 s^2 + 2.071 \cdot 10^4 s + 6130$$

$$N20_{23} = 0.3225s^{13} + 34.76s^{12} + 126.4s^{11} - 1512s^{10} + 135.3s^9 - 1.338 \cdot 10^4 s^8 - 7485s^7 - 3.921 \cdot 10^4 s^6 - \\ - 2.427 \cdot 10^4 s^5 - 3.304 \cdot 10^4 s^4 - 9368s^3 - 2048s^2 - 30.36s$$
$$D20_{23} = s^{15} + 120.4s^{14} + 1626s^{13} + 7983s^{12} + 2.784 \cdot 10^4 s^{11} + 7.032 \cdot 10^4 s^{10} + 1.396 \cdot 10^5 s^9 + 2.197 \cdot 10^5 s^8 + \\ + 2.772 \cdot 10^5 s^7 + 2.802 \cdot 10^5 s^6 + 2.228 \cdot 10^5 s^5 + 1.365 \cdot 10^5 s^4 + 6.109 \cdot 10^4 s^3 + 1.811 \cdot 10^4 s^2 + \\ + 3246s + 309.3$$

B.2. Funciones de transferencia del modelo linealizado para una velocidad de 30 nudos.

Para una velocidad de 30 nudos, las funciones de transferencia serán análogas a las vistas para la velocidad de 20 nudos, siendo ahora sus numeradores y denominadores los siguientes:

$$N30_{11} = 0.1562s^6 + 1.504s^5 + 1.507s^4 + 15.19s^3 + 17.48s^2 + 36.97s + 14.51$$
$$D30_{11} = s^8 + 116.6s^7 + 1394s^6 + 4711s^5 + 1.173 \cdot 10^4 s^4 + 2.022 \cdot 10^4 s^3 + 2.267 \cdot 10^4 s^2 + 1.984 \cdot 10^4 s + 5717$$

$$N30_{12} = -0.15s^6 - 1.476s^5 - 1.176s^4 - 15.22s^3 - 14.84s^2 - 35.91s - 13.94$$
$$D30_{12} = s^8 + 116.6s^7 + 1394s^6 + 4711s^5 + 1.173 \cdot 10^4 s^4 + 2.022 \cdot 10^4 s^3 + \\ + 2.267 \cdot 10^4 s^2 + 1.984 \cdot 10^4 s + 5717$$

$$N30_{13} = 2.837s^{13} + 36.95s^{12} + 139.7s^{11} + 821.9s^{10} + 2242s^9 + 7267s^8 + 1.545 \cdot 10^4 s^7 + 2.978 \cdot 10^4 s^6 + \\ + 4.257 \cdot 10^4 s^5 + 4.566 \cdot 10^4 s^4 + 3.477 \cdot 10^4 s^3 + 1.336 \cdot 10^4 s^2 + 17.65s - 799.8$$
$$D30_{13} = s^{15} + 118.8s^{14} + 1660s^{13} + 8557s^{12} + 3.171 \cdot 10^4 s^{11} + 8.523 \cdot 10^4 s^{10} + 1.816 \cdot 10^5 s^9 + 3.086 \cdot 10^5 s^8 + \\ + 4.238 \cdot 10^5 s^7 + 4.691 \cdot 10^5 s^6 + 4.101 \cdot 10^5 s^5 + 2.77 \cdot 10^5 s^4 + 1.362 \cdot 10^5 s^3 + 4.397 \cdot 10^4 s^2 + 8460s + \\ + 799.8$$

$$N30_{21} = -0.004473s^6 - 0.4983s^5 - 1.148s^4 + 27.37s^3 + 12.85s^2 + 85.19s + 2.191$$
$$D30_{21} = s^8 + 116.6s^7 + 1394s^6 + 4711s^5 + 1.173 \cdot 10^4 s^4 + 2.022 \cdot 10^4 s^3 + 2.267 \cdot 10^4 s^2 + 1.984 \cdot 10^4 s + 5717$$

$$N30_{22} = -0.006028s^6 - 0.6257s^5 - 1.918s^4 + 37.7s^3 + 14.79s^2 + 114.7s + 2.952$$
$$D30_{22} = s^8 + 116.6s^7 + 1394s^6 + 4711s^5 + 1.173 \cdot 10^4 s^4 + 2.022 \cdot 10^4 s^3 + 2.267 \cdot 10^4 s^2 + 1.984 \cdot 10^4 s + 5717$$

$$
\begin{aligned}
N30_{23} &= 0.2587s^{18} + 40.29s^{17} + 1712s^{16} + 3.282\cdot 10^4 s^{15} + 3.027\cdot 10^5 s^{14} + 8.473\cdot 10^5 s^{13} - 7.742\cdot 10^6 s^{12} - \\
&\quad - 7.933\cdot 10^7 s^{11} - 3.329\cdot 10^8 s^{10} - 1.088\cdot 10^9 s^9 - 3.199\cdot 10^9 s^8 - 5.394\cdot 10^9 s^7 - 9.52\cdot 10^9 s^6 - \\
&\quad - 8.621\cdot 10^9 s^5 - 7.758\cdot 10^9 s^4 - 2.578\cdot 10^9 s^3 - 5.01\cdot 10^8 s^2 - 1.135\cdot 10^7 s \\
D30_{23} &= s^{20} + 168.8s^{19} + 8600s^{18} + 2.203\cdot 10^5 s^{17} + 3.357\cdot 10^6 s^{16} + 3.286\cdot 10^7 s^{15} + 2.166\cdot 10^8 s^{14} + \\
&\quad + 1.006\cdot 10^9 s^{13} + 3.491\cdot 10^9 s^{12} + 9.579\cdot 10^9 s^{11} + 2.114\cdot 10^{10} s^{10} + 3.832\cdot 10^{10} s^9 + \\
&\quad + 5.717\cdot 10^{10} s^8 + 7.022\cdot 10^{10} s^7 + 7.032\cdot 10^{10} s^6 + 5.626\cdot 10^{10} s^5 + 3.495\cdot 10^{10} s^4 + 1.59\cdot 10^{10} s^3 + \\
&\quad + 4.828\cdot 10^9 s^2 + 8.859\cdot 10^8 s + 7.998\cdot 10^7
\end{aligned}
$$

B.3. Funciones de transferencia del modelo linealizado para una velocidad de 40 nudos.

Para la velocidad de 40 nudos, los numeradores y denominadores de las distintas funciones de transferencia serán:

$$
\begin{aligned}
N40_{11} &= 0.1694s^6 + 1.542s^5 + 1.206s^4 + 21.44s^3 + 24.19s^2 + 58.57s + 25.83 \\
D40_{11} &= s^8 + 115.9s^7 + 1384s^6 + 4662s^5 + 1.208\cdot 10^4 s^4 + 2.147\cdot 10^4 s^3 + 2.5\cdot 10^4 s^2 + 2.326\cdot 10^4 s + 7324
\end{aligned}
$$

$$
\begin{aligned}
N40_{12} &= -0.1626s^6 - 1.511s^5 - 0.8486s^4 - 21.48s^3 - 20.22s^2 - 57.07s - 24.83 \\
D40_{12} &= s^8 + 115.9s^7 + 1384s^6 + 4662s^5 + 1.208\cdot 10^4 s^4 + 2.147\cdot 10^4 s^3 + 2.5\cdot 10^4 s^2 + 2.326\cdot 10^4 s + 7324
\end{aligned}
$$

$$
\begin{aligned}
N40_{13} &= 1.78s^{13} + 24.87s^{12} + 125.5s^{11} + 790s^{10} + 2531s^9 + 9595s^8 + 2.334\cdot 10^4 s^7 + 5.072\cdot 10^4 s^6 + 7.92\cdot 10^4 s^5 + \\
&\quad + 9.255\cdot 10^4 s^4 + 7.569\cdot 10^4 s^3 + 3.045\cdot 10^4 s^2 - 57.23s - 2080 \\
D40_{13} &= s^{15} + 118.5s^{14} + 1690s^{13} + 9142s^{12} + 3.6\cdot 10^4 s^{11} + 1.028\cdot 10^5 s^{10} + 2.343\cdot 10^5 s^9 + 4.262\cdot 10^5 s^8 + \\
&\quad + 6.29\cdot 10^5 s^7 + 7.502\cdot 10^5 s^6 + 7.087\cdot 10^5 s^5 + 5.187\cdot 10^5 s^4 + 2.762\cdot 10^5 s^3 + 9.678\cdot 10^4 s^2 + \\
&\quad + 2.027\cdot 10^4 s + 2080
\end{aligned}
$$

$$
\begin{aligned}
N40_{21} &= -0.00525s^6 - 0.5713s^5 - 0.6887s^4 + 38.38s^3 + 21.69s^2 + 126.7s + 5.261 \\
D40_{21} &= s^8 + 115.9s^7 + 1384s^6 + 4662s^5 + 1.208\cdot 10^4 s^4 + 2.147\cdot 10^4 s^3 + 2.5\cdot 10^4 s^2 + 2.326\cdot 10^4 s + 7324
\end{aligned}
$$

$$
\begin{aligned}
N40_{22} &= -0.007074s^6 - 0.7262s^5 - 1.353s^4 + 52.85s^3 + 25.35s^2 + 170.6s + 7.089 \\
D40_{22} &= s^8 + 115.9s^7 + 1384s^6 + 4662s^5 + 1.208\cdot 10^4 s^4 + 2.147\cdot 10^4 s^3 + 2.5\cdot 10^4 s^2 + 2.326\cdot 10^4 s + 7324
\end{aligned}
$$

$$
\begin{aligned}
N40_{23} &= 0.1605s^{18} + 24.87s^{17} + 1049s^{16} + 1.982\cdot 10^4 s^{15} + 1.749\cdot 10^5 s^{14} + 3.161\cdot 10^5 s^{13} - 7.388\cdot 10^6 s^{12} - \\
&\quad - 7.396\cdot 10^7 s^{11} - 3.806\cdot 10^8 s^{10} - 1.508\cdot 10^9 s^9 - 4.567\cdot 10^9 s^8 - 8.843\cdot 10^9 s^7 - 1.537\cdot 10^{10} s^6 - \\
&\quad - 1.556\cdot 10^{10} s^5 - 1.401\cdot 10^{10} s^4 - 5.119\cdot 10^9 s^3 - 1.086\cdot 10^9 s^2 - 3.742\cdot 10^7 s \\
D40_{23} &= s^{20} + 168.5s^{19} + 8615s^{18} + 2.221\cdot 10^5 s^{17} + 3.418\cdot 10^6 s^{16} + 3.397\cdot 10^7 s^{15} + 2.291\cdot 10^8 s^{14} + \\
&\quad + 1.101\cdot 10^9 s^{13} + 3.999\cdot 10^9 s^{12} + 1.154\cdot 10^{10} s^{11} + 2.693\cdot 10^{10} s^{10} + 5.182\cdot 10^{10} s^9 + 8.231\cdot 10^{10} s^8 + \\
&\quad + 1.08\cdot 10^{11} s^7 + 1.159\cdot 10^{11} s^6 + 9.967\cdot 10^{10} s^5 + 6.666\cdot 10^{10} s^4 + 3.266\cdot 10^{10} s^3 + 1.071\cdot 10^{10} s^2 + \\
&\quad + 2.131\cdot 10^9 s + 2.08\cdot 10^8
\end{aligned}
$$

Apéndice C

Estudio comparativo de funciones de densidad de probabilidad

Para realizar el estudio comparativo, se trabajó con el modelo de avión propuesto por *GARTEUR*, y explicado en el capítulo 3. El algoritmo empleado es el que se describe con detalle en el capítulo 5, realizándose únicamente, en este estudio comparativo, la sintonía del modo longitudinal del controlador.

El algoritmo se ejecutó repetidamente en dos tandas de 5 experimentos. En la primera tanda se empleó una función de densidad de probabilidad uniforme para la generación de los números aleatorios, mientras que en la segunda se utilizó la función de densidad de probabilidad variable definida en el apartado 3.4.1.3 del capítulo 3.

C.1. Resultados.

A continuación se muestran los resultados obtenidos en las dos tandas de experimentos.

C.1.1. Resultados obtenidos con la función de densidad uniforme.

Experimento 1.

- Vector de aptitud del mejor cromosoma:

  $\{\ 1,\ 0.999889,\ -0.11,\ 0.99939,\ 0.0472532,\ 0.959698,\ 0.867928,\ 0.599766,\ -0.225452,$

−0.276709, 0.991527, −0.225, 0.966495, 0.383008, 0.969687, 0.902071, 0.439309, 0.205184, −0.155653, 0.0592035, 0.232553, 0.117896, 0.302492 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 9.53442 & -2.92911 & -0.911225 & -0.275739 & -0.147022 \\ -10 & -8.89547 & 1.34797 & 0.27756 & -0.335918 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0631151 & -0.0162544 \\ 0.0879273 & -0.0268528 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	$2.34012 \cdot 10^{-5}$	OK
Máximo tiempo de subida de Wv (<5 seg):	5.55	
Error medio de Wv al final (<0.042 m/s):	$2.56099 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.476373	OK
Error medio de VA al final (<0.01 m/s):	0.000403024	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.010874, 0.0382216]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0339518, 0.014503]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.320823	
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.356585	
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.0055072	OK
Máximo tiempo de subida de VA (<12 seg):	14.7	
Error medio de VA al final (<0.13 m/s):	0.00435562	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.431894	OK
Error medio de Wv al final (<0.01 m/s):	0.000303129	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.000120416, 0.0283406]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0551945]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.208083	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.322774	
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-10.6293, 10.6293]$	OK
Margen de fase entrada (-50º >MF >50º):	$[-57.2199, 57.2199]$	OK
Margen de ganancia salida (-10 >MG >10):	$[-11.3365, 11.3365]$	OK
Margen de fase salida (-50º >MF >50º):	$[-59.6605, 59.6605]$	OK

Experimento 2.

- Vector de aptitud del mejor cromosoma:

{ 1, 0.964964, **−0.2**, 0.992713, 0.174167, 0.517411, 0.85225, 0.318773, −0.168293,

$-0.161804,\ 0.976785,\ -0.125,\ 0.916082,\ 0.000296112,\ 0.843583,\ 0.902041,\ 0.334659,\ 0.261882,\ -0.133266,\ -0.156848,\ -0.0205075,\ -0.0935188,\ 0.0529651\ \}$

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 6.71332 & -1.05057 & -0.048893 & -0.325986 & -0.0969124 \\ 4.3266 & -7.52099 & 1.69107 & 0.183888 & -0.310061 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0719165 & -0.0150555 \\ 0.0781009 & -0.0262264 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00735758	OK
Máximo tiempo de subida de Wv (<5 seg):	6	
Error medio de Wv al final (<0.042 m/s):	0.000306073	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.412916	OK
Error medio de VA al final (<0.01 m/s):	0.00482589	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00800291, 0.0427587]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0577885, 0.0211587]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.305859	
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.324492	
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.0150895	OK
Máximo tiempo de subida de VA (<12 seg):	13.5	
Error medio de VA al final (<0.13 m/s):	0.0109094	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.699793	OK
Error medio de Wv al final (<0.01 m/s):	0.00156417	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[0, 0.0283492]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0654962]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.193239	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.316521	
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-8.64418, 8.64418]$	
Margen de fase entrada (-50º >MF >50º):	$[-49.4263, 49.4263]$	
Margen de ganancia salida (-10 >MG >10):	$[-9.14479, 9.14479]$	
Margen de fase salida (-50º >MF >50º):	$[-51.5274, 51.5274]$	OK

Experimento 3.

- Vector de aptitud del mejor cromosoma:

$\{\ 1,\ 1,\ -0.32,\ 0.986543,\ 0.653717,\ 0.593177,\ 0.877946,\ 0.566297,\ -0.309701,\ \mathbf{-0.323433},\ 1,\ -0.125,\ 0.846552,\ -0.158251,\ 0.683875,\ 0.892721,\ 0.222903,$

$-0.0293721,\ -0.30867,\ -0.230772,\ -0.105589,\ -0.124563,\ 0.0168788\ \}$

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 8.60996 & 2.06837 & 0.494343 & -0.358976 & 0.176887 \\ 8.7056 & -7.76381 & -2.89472 & -0.483786 & -0.338338 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.081306 & 0.0213691 \\ -0.0880517 & -0.0307684 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0	OK
Máximo tiempo de subida de Wv (<5 seg):	6.6	
Error medio de Wv al final (<0.042 m/s):	0.000565202	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.173142	OK
Error medio de VA al final (<0.01 m/s):	0.00406823	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00733123, 0.0353225]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.036791, 0.00442436]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.34288	
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.369635	
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0	OK
Máximo tiempo de subida de VA (<12 seg):	13.5	
Error medio de VA al final (<0.13 m/s):	0.0199483	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.810776	
Error medio de Wv al final (<0.01 m/s):	0.00316125	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00872588, 0.0310465]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0764975]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.26949	
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.365512	
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-8.12498, 8.12498]$	
Margen de fase entrada (-50º >MF >50º):	$[-47.148, 47.148]$	
Margen de ganancia salida (-10 >MG >10):	$[-8.89235, 8.89235]$	
Margen de fase salida (-50º >MF >50º):	$[-50.4795, 50.4795]$	OK

Experimento 4.

- Vector de aptitud del mejor cromosoma:

  $\{\ 1,\ 1,\ -0.32,\ 0.962921,\ -0.22742,\ 0.920629,\ 0.868252,\ 0.667794,\ -0.309098,$ **-0.327792**$,\ 0.993923,\ -0.1125,\ 0.974359,\ 0.0845745,\ 0.214382,\ 0.897764,\ 0.334869,$ $0.455603,\ -0.135801,\ -0.252802,\ -0.130809,\ -0.0225828,\ 0.135947\ \}$

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 9.66387 & 0.692407 & -0.523324 & -0.339284 & 0.130768 \\ 9.18226 & -8.88547 & -1.0122 & -0.450021 & -0.245318 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0786211 & 0.0109786 \\ -0.0924771 & -0.0260297 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0	OK
Máximo tiempo de subida de Wv (<5 seg):	6.6	
Error medio de Wv al final (<0.042 m/s):	0.00155731	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.61371	
Error medio de VA al final (<0.01 m/s):	0.000793705	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00731839, 0.0381279]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0281811, 0.0206789]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.342722	
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.370852	
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.00394993	OK
Máximo tiempo de subida de VA (<12 seg):	13.35	
Error medio de VA al final (<0.13 m/s):	0.00333334	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.640798	OK
Error medio de Wv al final (<0.01 m/s):	0.00785618	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.0128689, 0.029587]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0654755]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.142523	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.317229	
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-7.98211, 7.98211]$	
Margen de fase entrada (-50º >MF >50º):	$[-46.5031, 46.5031]$	
Margen de ganancia salida (-10 >MG >10):	$[-9.77916, 9.77916]$	
Margen de fase salida (-50º >MF >50º):	$[-54.0569, 54.0569]$	OK

Experimento 5.

- Vector de aptitud del mejor cromosoma:

  { 1, 0.997808, −0.2, 0.990364, 0.378414, 0.706261, 0.882623, 0.616085, **−0.220652**, −0.19812, 0.995555, −0.2125, 0.955766, 0.0523545, −0.0604322, 0.900737, 0.425304, 0.486651, −0.131833, −0.216483, −0.0891981, 0.0361728, 0.205247 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 9.47736 & 0.756201 & -0.621621 & -0.304068 & -0.0936219 \\ -8.74928 & -9.96221 & -2.69134 & 0.231573 & -0.287848 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0754121 & -0.00945798 \\ 0.0892346 & -0.0254387 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.000460329	OK
Máximo tiempo de subida de Wv (<5 seg):	6	
Error medio de Wv al final (<0.042 m/s):	0.000404699	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.310793	OK
Error medio de VA al final (<0.01 m/s):	0.00293739	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.0073232, 0.0339688]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0325675, 0]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.319567	
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.334635	
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.00288939	OK
Máximo tiempo de subida de VA (<12 seg):	14.55	
Error medio de VA al final (<0.13 m/s):	0.00575037	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.663352	OK
Error medio de Wv al final (<0.01 m/s):	0.0106043	
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[0, 0.0287266]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.056573]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.134395	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.316121	
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-8.22042, 8.22042]$	
Margen de fase entrada (-50º >MF >50º):	$[-47.5744, 47.5744]$	
Margen de ganancia salida (-10 >MG >10):	$[-10.3753, 10.3753]$	OK
Margen de fase salida (-50º >MF >50º):	$[-56.3016, 56.3016]$	OK

C.1.2. Resultados obtenidos con la función de densidad variable.

Experimento 1.

- Vector de aptitud del mejor cromosoma:

{ 1, 0.985282, 0.16, 0.998006, 0.548599, 0.823212, 0.849827, 0.318479, 0.156343, 0.155407, 0.508969, 0.175, 0.958966, 0.699355, 0.848816, 0.877995, **0.150762**, 0.150944, 0.202399, 0.158856, 0.351585, 0.152835, 0.344354 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 6.63105 & -2.97884 & -0.327679 & -0.182878 & -0.117705 \\ 0.57876 & -2.20712 & 1.11084 & 0.120511 & -0.129342 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0450419 & -0.0173813 \\ 0.0524452 & -0.017828 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00309073	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$8.37408 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.2257	OK
Error medio de VA al final (<0.01 m/s):	0.00176788	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00786915, 0.0434602]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0578134, 0.0172832]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.220869	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.235895	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.31917	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.0053344	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.210452	OK
Error medio de Wv al final (<0.01 m/s):	0.00151184	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00345761, 0.0353083]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.083599]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.222283	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.22277	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-11.8886, 11.8886]$	OK
Margen de fase entrada (-50º >MF >50º):	$[-61.4501, 61.4501]$	OK
Margen de ganancia salida (-10 >MG >10):	$[-11.8041, 11.8041]$	OK
Margen de fase salida (-50º >MF >50º):	$[-61.1825, 61.1825]$	OK

Experimento 2.

- Vector de aptitud del mejor cromosoma:

{ 1, 0.995818, **0.16**, 0.998963, 0.395917, 0.96025, 0.829838, 0.501069, 0.167561, 0.434311, 0.757628, 0.175, 0.987924, 0.634413, 0.913814, 0.881349, 0.166105, 0.556727, 0.161585, 0.165776, 0.359899, 0.169522, 0.364403 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 3.60226 & -1.42535 & -0.577105 & -0.156228 & -0.0552617 \\ 0.973621 & -2.31841 & 0.611959 & 0.0764131 & -0.137814 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.046259 & -0.00900339 \\ 0.0357519 & -0.0184263 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.000878209	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$4.35741 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.302042	OK
Error medio de VA al final (<0.01 m/s):	0.000397498	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00745357, 0.0492449]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0423243, 0.0107021]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.217933	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.157997	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.157542	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.00156993	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.255911	OK
Error medio de Wv al final (<0.01 m/s):	0.000861861	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[0, 0.0343377]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0820886]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.116049	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.234169	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-11.9872, 11.9872]$	OK
Margen de fase entrada (-50º >MF >50º):	$[-61.7594, 61.7594]$	OK
Margen de ganancia salida (-10 >MG >10):	$[-12.0413, 12.0413]$	OK
Margen de fase salida (-50º >MF >50º):	$[-61.9277, 61.9277]$	OK

Experimento 3.

- Vector de aptitud del mejor cromosoma:

{ 1, 0.991932, **0.16**, 0.998499, 0.406942, 0.830147, 0.840243, 0.358214, 0.163636, 0.177609, 0.579262, 0.175, 0.970179, 0.636645, 0.914867, 0.87855, 0.170172, 0.214293, 0.210737, 0.172272, 0.36771, 0.163383, 0.357024 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 5.58934 & -2.70356 & -0.472657 & -0.176163 & -0.114836 \\ -0.00352998 & -2.20283 & 0.937159 & 0.119411 & -0.1225 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0424105 & -0.0160183 \\ 0.053948 & -0.0175505 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00169421	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$6.30418 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.296529	OK
Error medio de VA al final (<0.01 m/s):	0.00169853	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00776059, 0.0462338]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0544427, 0.00558074]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.21896	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.229694	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.27348	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.00387668	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.254348	OK
Error medio de Wv al final (<0.01 m/s):	0.000851331	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.0055273, 0.0351477]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0816883]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.205698	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.220441	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-12.0813, 12.0813]$	OK
Margen de fase entrada (-50º >MF >50º):	$[-62.0516, 62.0516]$	OK
Margen de ganancia salida (-10 >MG >10):	$[-11.9529, 11.9529]$	OK
Margen de fase salida (-50º >MF >50º):	$[-61.6522, 61.6522]$	OK

Experimento 4.

- Vector de aptitud del mejor cromosoma:

{ 1, 0.973817, 0.16, 0.997697, 0.469112, 0.80295, 0.840941, 0.39076, 0.157314, 0.300463, 0.506408, 0.175, 0.955964, 0.51848, 0.896668, 0.883518, **0.152044**, 0.177913, 0.161741, 0.157224, 0.349624, 0.153208, 0.344802 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 5.41512 & -2.47093 & -0.374564 & -0.177619 & -0.103005 \\ 2.00928 & -2.83956 & 0.98018 & 0.0714733 & -0.145306 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0473063 & -0.016769 \\ 0.0406635 & -0.0187425 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00549834	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$9.67167 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.265444	OK
Error medio de VA al final (<0.01 m/s):	0.0019705	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00796367, 0.0460317]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0516819, 0.0216384]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.220615	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.195381	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.320835	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.0057247	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.337064	OK
Error medio de Wv al final (<0.01 m/s):	0.00103332	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[0, 0.0337099]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0834728]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.215222	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.234126	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-11.8655, 11.8655]$	OK
Margen de fase entrada (-50º >MF >50º):	$[-61.3774, 61.3774]$	OK
Margen de ganancia salida (-10 >MG >10):	$[-11.8093, 11.8093]$	OK
Margen de fase salida (-50º >MF >50º):	$[-61.199, 61.199]$	OK

Experimento 5.

- Vector de aptitud del mejor cromosoma:

{ 1, 0.988847, 0.16, 0.998802, 0.484934, 0.842193, 0.842462, 0.388746, 0.168044, 0.238869, 0.572201, 0.175, 0.968865, 0.800303, 0.903665, 0.879816, 0.159851, 0.213421, 0.315187, 0.169899, 0.364856, **0.159126**, 0.351909 }

- Matrices $\boldsymbol{K}_p$ y $\boldsymbol{K}_i$ obtenidas:

$$\boldsymbol{K}_p = \begin{bmatrix} 5.49489 & -2.60824 & -0.469816 & -0.175012 & -0.109918 \\ 0.0273788 & -1.93664 & 0.838388 & 0.109687 & -0.110907 \end{bmatrix}$$

$$\boldsymbol{K}_i = \begin{bmatrix} -0.0436125 & -0.0159852 \\ 0.0491122 & -0.0152105 \end{bmatrix}$$

- Objetivos de la función de evaluación (del 2 al 23):

Escalón en Wv:		
Máximo sobreimpulso en Wv (<0.21 m/s):	0.00234222	OK
Máximo tiempo de subida de Wv (<5 seg):	4.2	OK
Error medio de Wv al final (<0.042 m/s):	$5.03364 \cdot 10^{-5}$	OK
Pico máximo en VA debido a Wv (<0.5 m/s):	0.257533	OK
Error medio de VA al final (<0.01 m/s):	0.00157807	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00771901, 0.0455914]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844rad):	$[-0.0518527, 0.00832677]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.217806	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.212584	OK
Escalón en VA:		
Máximo sobreimpulso en VA (<0.65 m/s):	0.278069	OK
Máximo tiempo de subida de VA (<12 seg):	9.9	OK
Error medio de VA al final (<0.13 m/s):	0.00404751	OK
Pico máximo en Wv debido a VA (<0.7 m/s):	0.139788	OK
Error medio de Wv al final (<0.01 m/s):	0.000963353	OK
Saturación dT (-0.3215 rad <dT <0.2894 rad):	$[-0.00343694, 0.0347812]$	OK
Saturación dTH (-0.08483 rad <dTH <0.09844 rad):	$[0, 0.0827043]$	OK
Máxima variación de dT (|d(dT)| <0.2618 rad/seg):	0.205926	OK
Máxima variación de dTH (|d(dTH)| <0.2793 rad/seg):	0.191268	OK
Medidas de Robustez:		
Margen de ganancia entrada (-10 >MG >10):	$[-12.0467, 12.0467]$	OK
Margen de fase entrada (-50º >MF >50º):	$[-61.9447, 61.9447]$	OK
Margen de ganancia salida (-10 >MG >10):	$[-11.8924, 11.8924]$	OK
Margen de fase salida (-50º >MF >50º):	$[-61.4621, 61.4621]$	OK

C.1.3. Análisis de los resultados.

Analizando los resultados obtenidos en las dos tandas de experimentos, se observa que son sensiblemente mejores para el caso de la función de densidad de probabilidad variable. Para el caso de la función de densidad uniforme, en ninguno de los 5 experimentos se cumplieron todos los objetivos, pues en sus vectores de aptitud se observa la existencia de valores negativos (recuérdese que el objetivo se alcanza cuando el índice de aptitud toma el valor cero, y se mejora cuando toma valores positivos). Sin embargo, para el caso de la función de densidad de probabilidad variable se cumplieron todos los objetivos en todos los experimentos (todos los elementos del vector de aptitud son positivos). En cada vector de aptitud se encuentra resaltado en negrita el peor de los valores.

En las tablas que acompañan a los resultados de los distintos experimentos, se pueden consultar los límites y los valores alcanzados para cada objetivo.

Queda por tanto justificado el uso de la función de densidad de probabilidad variable.

Apéndice D

El producto en estrella de Redheffer

Sean dos matrices particionadas en la forma siguiente:

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \quad , \quad \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

tales que la matriz producto $T_{11} \cdot B_{11}$ esté bien definida, y sea de hecho, cuadrada.

Si $I - T_{22} \cdot B_{11}$ es invertible, se define el **producto en estrella de** T **y** B como

$$S(T,B) := \begin{bmatrix} F_l(T, B_{11}) & T_{12}(I - B_{11}T_{22})^{-1}B_{12} \\ B_{21}(I - T_{22}B_{11})^{-1}T_{21} & F_u(B, T_{22}) \end{bmatrix} \tag{D.1}$$

donde $F_l(T, B_{11})$ está definido como

$$F_l(T, B_{11}) := T_{11} + T_{12}B_{11}(I - T_{22}B_{11})^{-1}T_{21} \tag{D.2}$$

y $F_u(B, T_{22})$ está definido como

$$F_u(B, T_{22}) := B_{22} + B_{21}T_{22}(I - B_{11}T_{22})^{-1}B_{12} \tag{D.3}$$

La operación se representa en el diagrama de bloques de la figura D.1.

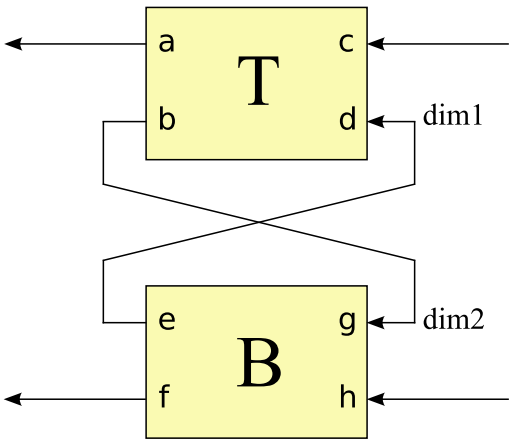


Figura D.1: Producto en estrella de Redheffer.

Cuya ecuación matricial es:

$$\begin{bmatrix} a \\ f \end{bmatrix} = \begin{bmatrix} S(T,B) \end{bmatrix} \cdot \begin{bmatrix} c \\ h \end{bmatrix} \tag{D.4}$$

En lo que sigue se deducirá dicha ecuación. Según la figura D.1 se tiene:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \cdot \begin{bmatrix} c \\ d \end{bmatrix} \Rightarrow \begin{array}{l} a = T_{11} \cdot c + T_{12} \cdot d \\ b = T_{21} \cdot c + T_{22} \cdot d \end{array} \Bigg\} \tag{D.5}$$

$$\begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \cdot \begin{bmatrix} g \\ h \end{bmatrix} \Rightarrow \begin{array}{l} e = B_{11} \cdot g + B_{12} \cdot h \\ f = B_{21} \cdot g + B_{22} \cdot h \end{array} \Bigg\} \tag{D.6}$$

Según se observa en la Figura D.1, $d = e$ y $b = g$, por lo que realizando estos cambios en la segunda ecuación de (D.5) y en la primera de (D.6) y operando luego en esta última se obtiene

$$\begin{aligned} & d = B_{11}(T_{21}c + T_{22}d) + B_{12}h = B_{11}T_{21}c + B_{11}T_{22}d + B_{12}h \\ & \Rightarrow (I - B_{11}T_{22})d = B_{11}T_{21}c + B_{12}h \\ & \Rightarrow d = (I - B_{11}T_{22})^{-1}B_{11}T_{21}c + (I - B_{11}T_{22})^{-1}B_{12}h \end{aligned} \tag{D.7}$$

Operando de igual forma con la segunda ecuación de (D.5) se obtiene a su vez

$$\begin{aligned} g &= T_{21}c + T_{22}(B_{11}g + B_{12}h) = T_{21}c + T_{22}B_{11}g + T_{22}B_{12}h \\ &\Rightarrow (I - T_{22}B_{11})g = T_{21}c + T_{22}B_{12}h \\ &\Rightarrow g = (I - T_{22}B_{11})^{-1}T_{21}c + (I - T_{22}B_{11})^{-1}T_{22}B_{12}h \end{aligned} \tag{D.8}$$

Y sustituyendo finalmente las expresiones anteriores en la primera ecuación de (D.5) y la segunda de (D.6) se obtiene

$$a = T_{11}c + T_{12}\left[(I - B_{11}T_{22})^{-1}B_{11}T_{21}c + (I - B_{11}T_{22})^{-1}B_{12}h\right] \tag{D.9}$$

$$f = B_{21}\left[(I - T_{22}B_{11})^{-1}T_{21}c + (I - T_{22}B_{11})^{-1}T_{22}B_{12}h\right] + B_{22}h \tag{D.10}$$

que puede escribirse en forma matricial como

$$\begin{bmatrix} a \\ f \end{bmatrix} = \begin{bmatrix} T_{11} + T_{12}(I - B_{11}T_{22})^{-1}B_{11}T_{21} & T_{12}(I - B_{11}T_{22})^{-1}B_{12} \\ B_{21}(I - T_{22}B_{11})^{-1}T_{21} & B_{22} + B_{21}(I - T_{22}B_{11})^{-1}T_{22}B_{12} \end{bmatrix} \cdot \begin{bmatrix} c \\ h \end{bmatrix}$$

siendo por tanto

$$S(T,B) = \begin{bmatrix} T_{11} + T_{12}(I - B_{11}T_{22})^{-1}B_{11}T_{21} & T_{12}(I - B_{11}T_{22})^{-1}B_{12} \\ B_{21}(I - T_{22}B_{11})^{-1}T_{21} & B_{22} + B_{21}(I - T_{22}B_{11})^{-1}T_{22}B_{12} \end{bmatrix} \tag{D.11}$$

Para que (D.1) y (D.11) sean iguales, han de cumplirse

$$B_{11}(I - T_{22}B_{11})^{-1} = (I - B_{11}T_{22})^{-1}B_{11} \tag{D.12}$$

y

$$T_{22}(I - B_{11}T_{22})^{-1} = (I - T_{22}B_{11})^{-1}T_{22} \tag{D.13}$$

Se demostrará únicamente (D.12) al ser idéntica la demostración para (D.13). Post-

multiplicando ambos miembros de (D.12) por $(I - T_{22}B_{11})$ se obtiene

$$B_{11} = (I - B_{11}T_{22})^{-1}B_{11}(I - T_{22}B_{11})$$

y premultiplicando ahora por $(I - B_{11}T_{22})$

$$(I - B_{11}T_{22})B_{11} = B_{11}(I - T_{22}B_{11})$$

realizando finalmente las multiplicaciones se obtiene

$$B_{11} - B_{11}T_{22}B_{11} = B_{11} - B_{11}T_{22}B_{11}$$

con lo que queda probada la equivalencia.

Apéndice E

Código utilizado.

E.1. Código utilizado en la evaluación de cromosomas en el problema de control de vuelo de un avión comercial.

E.1.1. Determinación del valor máximo de las partes reales de los autovalores.

```
maxautovalor=max(real(eig(A)));
fitness=ones(1, NumObjetivos);
if maxautovalor>=-0.15
   fitness=-Inf*fitness;
   % El primer valor de valores se utiliza para la estabilidad:
   fitness(1)=-maxautovalor;
else
   fitness(1)=Inf;
   % A continuación se establecen el resto de valores de fitness.
   ..........................
end,
```

E.1.2. overshoot.m

```
function valor=overshoot(salida, estacionario)
% Se obtiene el mayor de los valores de salida:
```

```
maximo = max(salida);
% Si no hay términos mayores que el valor estacionario:
if (maximo <= estacionario)
   valor = 0;
% Si los hay:
else
   valor = maximo - estacionario;
end;
```

E.1.3. trise.m

```
function value=trise(salida, estacionario, periodo)
% Términos que sobrepasan el 80%:
n = find(salida > 0.8*estacionario);
if ~isempty(n)
   value = (min(n) - 1)*periodo;
% Si no hay ninguno se realiza una estimación:
else
   long = length(salida);    % Número de puntos de la salida.
   decimo = round(long/10);  % Décima parte num. puntos de la salida.
   % Se han de tener al menos 3 puntos:
   if decimo < 3
      decimo = 3;
   end,
   % Índice del primer valor de la décima parte:
   indmin = long - decimo + 1;
   indmin1 = indmin -1;
   % Instantes de tiempo correspondientes a los valores:
   tfit = indmin1*periodo:periodo:(indmin1+decimo-1)*periodo;
   % Valores correspondientes de la salida:
   yfit = salida(indmin:end);
   % Se realiza el ajuste a una función de segundo grado:
   coef = polyfit(tfit, yfit, 2);
   % La estimación se hará con la recta tangente al último
   % punto. Se calcula por tanto la pendiente de dicha recta:
   m = 2 * coef(1) * tfit(end) + coef(2);
   % Si la recta tiene una pendiente positiva, se realiza la
```

```
   % extrapolación:
   if m > 0
      value = (0.8 * estacionario - yfit(end) + m * tfit(end)) / m;
   % Si no el tiempo de subida será infinito:
   else
      value = inf;
   end,
end,
```

E.1.4. errormed.m

```
function value=errormed(salida, estacionario)
% Se calcula la longitud de la salida:
long=length(salida);
% Se calculan sus últimos valores (20 %):
puntos=fix(0.2*long);
% Último 20 % de la salida:
ypercent=salida((long-puntos):long);
% Error medio de los últimos valores:
value=mean(abs(ypercent-estacionario));
```

E.1.5. peakmax.m

```
function value=peakmax(salida)
value=max(abs(salida));
```

E.1.6. Saturación de δ_T y δ_{TH}.

```
% Los valores de la salida 6 corresponden a dT:
dT = y(:, 6);
% Se obtienen sus valores máximo y mínimo:
MaxdT = max(dT);
MindT = min(dT);
% Se obtienen valores normalizados:
if MaxdT > 0
   MaxdT = 1 - MaxdT / 0.2894;
else
```

```
   MaxdT = 1;
end,
if MindT < 0
   MindT = 1 + MindT / 0.3215;
else
   MindT = 1;
end,
% Se añade al vector de aptitudes el peor de los 2 valores:
Fit(7) = min(MaxdT, MindT);


% Los valores de la salida 7 corresponden a dTH:
dTH = y(:, 7);
% Se obtienen sus valores máximo y mínimo:
MaxdTH = max(dTH);
MindTH = min(dTH);
% Se obtienen valores normalizados:
if MaxdTH > 0
   MaxdTH = 1 - MaxdTH / 0.09844;
else
   MaxdTH = 1;
end,
if MindTH < 0
   MindTH = 1 + MindTH / 0.08483;
else
   MindTH = 1;
end,
% Se añade al vector de aptitudes el peor de los 2 valores:
Fit(8) = min(MaxdTH, MindTH);
```

E.1.7. Variación máxima de δ_T y δ_{TH}.

```
% Máxima variación de dT  < 0.2618:
vardT = abs(diff(dT)) ./ Ts;
% Se normaliza y añade al vector de aptitudes:
Fit(9) = 1 - max(vardT) / 0.2618;
% Máxima variación de dTH < 0.2793:
```

```
vardTH = abs(diff(dTH)) ./ Ts;
% Se normaliza y añade al vector de aptitudes:
Fit(10) = 1 - max(vardTH) / 0.2793;
```

E.2. Funciones para la obtención de la ganancia del lazo externo en la sintonía directa de un controlador con una estructura predeterminada.

E.2.1. extrlooplon.m

```
function extrlooplon(archivo)
% Función para los cálculos de la ganancia del lazo externo, a partir de
% un archivo histórico conteniendo un controlador.

% Matrices de las ecuaciones en el espacio de estados, correspondientes
% al modelo linealizado:
sysdata;   % Archivo que contiene las matrices del modelo.
A = Along; B = Blong; C = Clong;
D=zeros(size(C,1), size(B, 2));

% Matriz que permite la realimentación de Wv y VA:
Kr=[0 0 0 1 0;     % Wv
    0 0 0 0 1]; % VA

% Permitirá obtener Wv para el cálculo de z:
Kz=Kr(1,:);

% Matriz que establece la referencia de VA=0:
Kk=[1; 0];

% Matrices correspondientes a los dos integradores de los errores de
% Wv y VA:
Ag=zeros(2);
Bg=eye(2);
Cg=eye(2);
```

```
Dg=zeros(2);

% Integrador para cálculo de z:
AI=0;
BI=1;
CI=1;
DI=0;

% Ganancias del controlador:
% Se carga el archivo pasado como argumento:
eval(['load ' archivo]);

% Se obtienen las ganancias del controlador:
chrom = elmejor;
Kp=[chrom(1:5); chrom(6:10)];
Ki=[chrom(11:12); chrom(13:14)];

% Bloque correspondiente a los integradores para Wv y VA:
Gi=pck(Ag, Bg, Cg, Dg);

% Construcción del sistema correspondiente al lazo interno (innerloop):
Planta=pck(A, B, C, D);
systemnames = ' Planta Kp Ki Kr Gi ';
inputvar = '[ ref{2} ]';
outputvar = '[ Planta ]';
input_to_Planta = '[ Kp + Ki ]';
input_to_Kp = '[ Planta ]';
input_to_Ki = '[ Gi ]';
input_to_Kr = '[ Planta ]';
input_to_Gi = '[ -ref + Kr ]';
sysoutname = 'innerloop';
cleanupsysic = 'yes';
sysic;

% Bloque que obtendrá la se˜al z a partir de las salidas de la planta:
I=pck(AI, BI, CI, DI);
systemnames = ' I Kz ';
```

```
inputvar = '[ uI{5} ]';
outputvar = '[ I ]';
input_to_I = '[ Kz ]';
input_to_Kz = '[ uI ]';
sysoutname = 'getz';
cleanupsysic = 'yes';
sysic;

% Construcción del sistema global en lazo abierto:
systemnames = ' innerloop getz Kk ';
inputvar = '[ Z_c ]';
outputvar = '[ getz ]';
input_to_getz = '[ innerloop ]';
input_to_innerloop = '[ Kk ]';
input_to_Kk = '[ Z_c ]';
sysoutname = 'openloop';
cleanupsysic = 'yes';
sysic;

[Ahla,Bhla,Chla,Dhla]=unpck(openloop);
[num,den]=ss2tf(Ahla,Bhla,Chla,Dhla);
% Se eliminan los ceros a la izquierda:
lonnum = length(num);
prueba = 0;
for k=1:lonnum
   if abs(num(k)) > 1e-12
      primernocero = k;
      prueba = 1;
      break;
   end,
end,
if prueba == 1
   num = num(primernocero:end);
end,
h = figure,
rlocus(num, den); sgrid
axis([-1 1 -1 1])
```

```
for i=1:5
   [Kout, poles]=rlocfind(num, den);
   Kout,
   damp(poles),
   trts(poles);
end
```

E.2.2. trts.m

Función extraída de [36]:

```
function trs=trts(raices)

% TRTS Rise Time Settling Time
%
%trs=trts(raices)
%
% Computes the rise time and the settling time
% of the roots in vector raices
%

% 18/4/96

n=size(raices, 1);
for i=1:n,
   if imag(raices(i,1))==0,
      tau=-1/raices(i,1);
      t10=tau*(-log(0.9));
      t90=tau*(-log(0.1));
      t9010=t90-t10;
      t99=tau*(-log(0.01));
   else
      [wn, psi]=damp(raices(i,1));
      t9010=(1-0.4167*psi+2.917*psi^2)/wn;
      t99=-log(0.01*sqrt(1-psi^2))/(psi*wn);
   end
```

```
   trs(i, 1:3)=[raices(i,1), t9010, t99];
end
disp('     Raiz           Rise time         Settling time');
trs,
```

Apéndice F

Resultados de los experimentos de paralelización en el *clúster* de PCs.

F.1. Estrategia 1.

Nº proces.	t1	t2	t3	t4	tmedio
1	2:28:08	2:29:16	2:28:19	2:27:52	2:28:24
2	1:24:09	1:24:00	1:24:31	1:25:22	1:24:30
3	1:01:49	1:01:58	1:01:39	1:01:38	1:01:46
4	0:49:12	0:49:10	0:49:19	0:49:21	0:49:15
5	0:41:49	0:41:56	0:42:17	0:41:55	0:41:59
6	0:36:58	0:36:53	0:37:09	0:36:53	0:36:58
7	0:33:01	0:32:55	0:32:54	0:33:05	0:32:59
8	0:30:13	0:30:21	0:30:14	0:30:07	0:30:13
9	0:27:49	0:27:43	0:27:25	0:27:36	0:27:38
10	0:25:50	0:25:57	0:25:42	0:25:43	0:25:48
11	0:23:46	0:23:50	0:23:52	0:23:55	0:23:51
12	0:22:33	0:22:40	0:23:06	0:22:39	0:22:45
13	0:21:37	0:21:59	0:21:43	0:21:37	0:21:44
14	0:20:18	0:20:29	0:20:28	0:20:24	0:20:25
Total:	10:27:12	10:29:07	10:28:38	10:28:08	10:28:16

Tabla F.1: Tiempos (hh:mm:ss) vs. nº de procesadores, y tiempo medio.

Nº proces.	a1	a2	a3	a4	amedia	desv.
1	1	1	1	1	1	0
2	1.76054	1.77695	1.75484	1.73225	1.75614	0.0160058
3	2.39626	2.4085	2.40573	2.39878	2.40232	0.00497672
4	3.01138	3.03625	3.00799	2.99661	3.01306	0.0144666
5	3.54227	3.55903	3.50833	3.5271	3.53418	0.0187174
6	4.007	4.04781	3.99316	4.00859	4.01414	0.0203438
7	4.48754	4.53571	4.50761	4.46897	4.49996	0.0247556
8	4.90333	4.91914	4.90639	4.9107	4.90989	0.00594494
9	5.3266	5.38484	5.41117	5.35691	5.36988	0.0315031
10	5.73477	5.75124	5.77001	5.74971	5.75143	0.0125055
11	6.23273	6.26325	6.21524	6.18073	6.22299	0.0298395
12	6.56775	6.58471	6.4203	6.52781	6.52514	0.0639573
13	6.85308	6.78807	6.83033	6.83788	6.82734	0.0241109
14	7.29736	7.28785	7.24806	7.24538	7.26966	0.0232065

Tabla F.2: Aceleraciones vs. nº de procesadores, aceleración media y desviación estándar

F.2. Estrategia 2.

Nº proces.	t1	t2	t3	t4	tmedio
1	2:28:59	2:26:47	2:28:06	2:27:55	2:27:56
2	2:26:02	2:25:56	2:26:41	2:25:58	2:26:09
3	1:24:11	1:24:14	1:23:57	1:24:20	1:24:11
4	1:01:53	1:01:58	1:01:47	1:01:41	1:01:50
5	0:49:11	0:49:19	0:49:19	0:49:06	0:49:14
6	0:41:59	0:41:56	0:41:59	0:41:56	0:41:58
7	0:37:02	0:37:04	0:36:52	0:37:08	0:37:01
8	0:32:59	0:32:57	0:32:58	0:32:53	0:32:57
9	0:30:22	0:30:02	0:30:03	0:30:31	0:30:15
10	0:27:60	0:27:24	0:27:23	0:27:40	0:27:36
11	0:25:59	0:25:57	0:25:35	0:25:46	0:25:49
12	0:23:54	0:24:21	0:24:14	0:23:42	0:24:03
13	0:22:42	0:22:33	0:22:43	0:22:57	0:22:44
14	0:21:42	0:22:05	0:21:41	0:21:55	0:21:51
Total:	12:34:55	12:32:33	12:33:17	12:33:27	12:33:33

Tabla F.3: Tiempos (hh:mm:ss) vs. nº de procesadores, y tiempo medio.

Nº proces.	a1	a2	a3	a4	amedia	desv.
1	1	1	1	1	1	0
2	1.02016	1.00585	1.00969	1.01334	1.01226	0.00527483
3	1.76957	1.74248	1.76399	1.7538	1.75746	0.0103322
4	2.40735	2.36855	2.39709	2.39821	2.3928	0.0145569
5	3.02897	2.976	3.00261	3.01265	3.00506	0.0192328
6	3.54868	3.49996	3.52766	3.52696	3.52582	0.0172895
7	4.0234	3.96063	4.01647	3.98392	3.99611	0.0253283
8	4.51794	4.45348	4.49337	4.49746	4.49056	0.0233474
9	4.90477	4.88778	4.92836	4.84568	4.89165	0.0301996
10	5.32222	5.35806	5.40975	5.34696	5.35925	0.0319129
11	5.73204	5.65444	5.78882	5.74091	5.72905	0.0481884
12	6.23382	6.0267	6.11251	6.24101	6.15351	0.0892574
13	6.5609	6.50757	6.51971	6.44698	6.50879	0.0407916
14	6.86583	6.64822	6.82847	6.74709	6.7724	0.0835687

Tabla F.4: Aceleraciones vs. nº de procesadores, aceleración media y desviación estándar

F.3. Estrategia 3.

Nº proces.	t1	t2	t3	t4	tmedio
1	2:30:46	2:31:18	2:31:46	2:28:18	2:30:32
2	1:23:33	1:23:41	1:24:04	1:22:59	1:23:34
3	0:56:43	0:56:40	0:56:51	0:56:38	0:56:43
4	0:46:57	0:46:54	0:46:60	0:46:38	0:46:52
5	0:37:58	0:37:58	0:38:06	0:38:01	0:38:01
6	0:32:26	0:32:31	0:32:32	0:32:21	0:32:27
7	0:29:23	0:29:15	0:29:13	0:29:08	0:29:15
8	0:26:18	0:26:15	0:26:17	0:26:11	0:26:15
9	0:25:22	0:25:23	0:25:25	0:25:29	0:25:25
10	0:23:06	0:23:06	0:23:06	0:23:09	0:23:07
11	0:20:53	0:20:58	0:20:56	0:20:57	0:20:56
12	0:20:27	0:20:20	0:20:19	0:20:39	0:20:26
13	0:19:14	0:19:14	0:19:13	0:19:19	0:19:15
14	0:17:57	0:18:19	0:18:11	0:18:39	0:18:17
Total:	9:51:03	9:51:53	9:52:59	9:48:25	9:51:05

Tabla F.5: Tiempos (hh:mm:ss) vs. nº de procesadores, y tiempo medio.

Nº proces.	a1	a2	a3	a4	amedia	desv.
1	1	1	1	1	1	0
2	1.80452	1.80809	1.80548	1.78726	1.80134	0.00823289
3	2.65819	2.66967	2.66959	2.61867	2.65403	0.0209437
4	3.21132	3.22592	3.22917	3.18014	3.21164	0.0193903
5	3.9708	3.98478	3.98266	3.9019	3.96004	0.0339853
6	4.64932	4.65295	4.66589	4.58463	4.6382	0.031533
7	5.13034	5.17307	5.19599	5.09187	5.14782	0.0399825
8	5.73431	5.76253	5.77419	5.66515	5.73404	0.0423396
9	5.94393	5.96	5.96953	5.82086	5.92358	0.0600083
10	6.52904	6.54992	6.57093	6.40496	6.51371	0.0645128
11	7.21917	7.21762	7.25188	7.07633	7.19125	0.067745
12	7.37455	7.43958	7.4688	7.18442	7.36684	0.110707
13	7.83977	7.86483	7.89632	7.67648	7.81935	0.0848827
14	8.39754	8.25877	8.34675	7.95457	8.23941	0.171783

Tabla F.6: Aceleraciones vs. nº de procesadores, aceleración media y desviación estándar

F.4. Estrategia 4.

Nº proces.	t1	t2	t3	t4	tmedio
1	2:27:16	2:28:05	2:28:20	2:30:02	2:28:26
2	2:12:28	2:12:34	2:12:46	2:12:26	2:12:33
3	1:17:24	1:19:03	1:12:52	1:17:12	1:16:38
4	0:49:04	0:51:19	0:49:03	0:50:01	0:49:52
5	0:38:30	0:37:23	0:38:56	0:39:13	0:38:30
6	0:31:34	0:32:21	0:30:13	0:32:51	0:31:45
7	0:26:25	0:25:51	0:24:56	0:27:16	0:26:07
8	0:23:16	0:22:43	0:23:12	0:23:40	0:23:13
9	0:22:07	0:22:29	0:21:43	0:21:10	0:21:52
10	0:19:45	0:19:46	0:20:03	0:20:04	0:19:55
11	0:18:20	0:18:12	0:18:18	0:17:39	0:18:07
12	0:17:25	0:17:12	0:17:09	0:17:12	0:17:15
13	0:15:55	0:15:10	0:15:56	0:15:43	0:15:41
14	0:15:14	0:13:36	0:15:29	0:15:04	0:14:51
Total:	10:34:43	10:35:45	10:28:56	10:39:32	10:34:44

Tabla F.7: Tiempos (hh:mm:ss) vs. nº de procesadores, y tiempo medio.

Nº proces.	a1	a2	a3	a4	amedia	desv.
1	1	1	1	1	1	0
2	1.11175	1.117	1.11727	1.133	1.11976	0.00795886
3	1.90286	1.87343	2.03562	1.94352	1.93886	0.0611588
4	3.00108	2.88544	3.0246	2.99963	2.97769	0.0541746
5	3.82544	3.96101	3.8102	3.82646	3.85578	0.0610961
6	4.66596	4.57711	4.91017	4.56848	4.68043	0.13802
7	5.57563	5.7303	5.94798	5.50369	5.6894	0.170272
8	6.329	6.51796	6.39455	6.33852	6.39501	0.0752763
9	6.66124	6.58765	6.82791	7.08744	6.79106	0.19198
10	7.45548	7.49185	7.39859	7.47448	7.4551	0.0350726
11	8.03135	8.13406	8.10404	8.50209	8.19288	0.182387
12	8.4533	8.61021	8.65154	8.72162	8.60917	0.0984071
13	9.25005	9.75952	9.3097	9.55152	9.4677	0.202807
14	9.66597	10.8875	9.58013	9.95851	10.023	0.518451

Tabla F.8: Aceleraciones vs. nº de procesadores, aceleración media y desviación estándar