Máster en Ingeniería de Sistemas y Control

Proyecto Fin de Máster

Control de rumbo de un vehículo marino autónomo mediante técnicas de Inteligencia Artificial

Autor: Joseba Menoyo Larrazabal

Directora: Matilde Santos Peñas

Curso 2012 – 2013

Convocatoria Septiembre

Máster en Ingeniería de Sistemas y Control

Proyecto Fin de Máster

Control de rumbo de un vehículo marino autónomo mediante técnicas de Inteligencia Artificial

Autor: Joseba Menoyo Larrazabal

Directora: Matilde Santos Peñas

Resumen

Los distintos tipos de vehículos autónomos tienen cada día mayor importancia en nuestra sociedad. En el caso concreto de barcos autónomos, este tipo de sistemas pueden ser utilizados en una amplia gama de aplicaciones: tareas de rescate, transporte, delimitación de manchas en el mar, búsqueda de náufragos, a modo de barcos nodriza para transportar otro tipo de vehículos autónomos, etc.

Este Trabajo Fin de Máster presenta la aplicación de varias metodologías basadas en propuestas computacionales inteligentes de la rama conexionista (redes neuronales, lógica difusa y algoritmos genéticos), para abordar la problemática del control del rumbo en vehículos marinos autónomos.

Se diseñará un controlador difuso, un controlador optimizado mediante AG con una estrategia de control de ganancia programada, y se comparará su funcionamiento con un controlador PID clásico.

El funcionamiento y validación de los controladores se simulará utilizando un modelo del barco ya existente, que ha sido generado a partir de los datos obtenidos en pruebas realizadas con un prototipo a escala.

Finalmente, se desarrollará una red neuronal artificial para aplicar una estrategia de control adaptativa que trate de contrarrestar los posibles efectos negativos que provocan en el desplazamiento del barco las perturbaciones a las que se debería hacer frente en un entorno real, como serían el viento, oleaje, mareas, etc.

Autorización de difusión

El/la abajo firmante, matriculado/a en el Máster de Ingeniería de Sistemas y de Control, autoriza a la de la Universidad Nacional de Educación a Distancia (UNED) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: “Control de rumbo de un vehículo marino autónomo mediante técnicas de Inteligencia Artificial”, realizado durante el curso académico 2012-2013 bajo la dirección de Dª. Matilde Santos Peñas.

Joseba Menoyo Larrazabal
Septiembre de 2013

Índice

Índice de figuras

Índice de tablas

1. Introducción

Este primer capítulo sirve para introducir el contenido del presente Trabajo Fin de Máster. Se realizará un breve resumen de la literatura existente en aplicaciones de control en navegación de vehículos marinos, y se indicarán cuales son los objetivos que se pretenden alcanzar con el desarrollo del trabajo.

Por último, se dará una visión general de los distintos capítulos y secciones que conforman el presente documento.

1.1. Motivación

La aplicación de técnicas de control de vehículos marinos comenzó a principios del siglo XIX con un sistema automático de seguimiento del rumbo o autopiloto, basado en un diseño con una brújula giroscópica. Fue Elmer Sperry quien en 1911 realizó el primer sistema de conducción automático de un barco. En 1922 Nicholas Minorsky introdujo el controlador PID en su trabajo "Estabilidad direccional de cuerpos dirigidos automáticamente", siendo desde entonces uno de los sistemas de control más utilizados en las aplicaciones de barcos autopilotados (así como en muchos otros tipos de aplicaciones de control).

En la actualidad, los sistemas de control automático son un aspecto de gran interés dentro de las aplicaciones marítimas, existiendo numerosos escenarios donde se pueden emplear aplicaciones con barcos autónomos, como por ejemplo: las tareas de reconocimiento, rescate, protección, salvamento o búsqueda de náufragos, recogida de vertidos, delimitación de manchas en el mar, despliegue de redes de pesca, creación de mapas batimétricos, transporte, etc.

El interés por tratar de reducir la intervención humana, así como por obtener un control más eficiente (estabilización de alabeo, amortiguación del movimiento, eficiencia en el gasto de combustible, etc.), ha llevado a evolucionar desde el control PID clásico a un control más robusto y adaptativo, donde pueden encajar a la perfección las técnicas de IA.

Además, dependiendo de las condiciones, los barcos pueden ser sistemas poco estables y difíciles de controlar. Esta complejidad se debe, en primer lugar, a que son sistemas con una dinámica no lineal, multivariables y con acoplos entre sus variables de estado. Este hecho dificulta la obtención de un modelo matemático que represente con precisión su dinámica, y hay que recurrir a aproximaciones que crean incertidumbre respecto a su comportamiento [1].

Por otro lado, este tipo de aplicaciones exigen un control en tiempo real, lo que implica que la computación de los algoritmos de control debe tratar de ser lo más rápida posible.

Por estas razones, la utilización de técnicas de control inteligente en aplicaciones de guiado y direccionamiento del rumbo de barcos ha aumentado notablemente en los últimos años, ya que gracias a su robustez, pueden hacer frente de manera más efectiva a las no linealidades e incertidumbres que se dan en la conducción de barcos de forma autónoma.

En el guiado automático de barcos autónomos, los sistemas de control avanzado suelen ser en muchas ocasiones el resultado de la combinación de diferentes técnicas de control, que tratan de actuar y tomar decisiones como lo haría un experimentado timonel.

De este modo, en este trabajo se pretende aplicar diferentes tipos de controladores inteligentes, beneficiándose de las características para ofrecer un mejor funcionamiento en el autoguiado de barcos, convirtiéndolos en una alternativa de gran interés.

1.2. Objetivos

El objetivo de este Proyecto Fin de Máster es desarrollar varios controladores basados en técnicas de control inteligente para el guiado automático del rumbo en un barco.

De este modo, con este trabajo se pretende investigar las posibilidades y ventajas del uso de las técnicas no lineales y las que provienen de la IA para abordar el tema del control de un vehículo marino autónomo en su sentido más amplio: diseño, simulación, y validación de los controladores inteligentes.

Al igual que ocurre con otro tipo de vehículos móviles autónomos, se puede considerar que un sistema de control del movimiento de un barco contiene tres sistemas básicos [1]:

- Sistema de guiado. Es el encargado de proporcionar de forma continuada las indicaciones necesarias para que el barco siga una ruta precisa.
- Sistema de navegación. Sirve para determinar aspectos relevantes a la navegación como la posición y rumbo del barco, haciendo uso de dispositivos como DGPS, giroscopios, etc.
- Sistema de control. Su función es actuar sobre los diferentes elementos encargados de controlar el movimiento del barco, como pueden ser: hélices y/o motores para producir el empuje de avance del vehículo, control del timón para generar movimientos de giro acimutal, aletas estabilizadoras, impulsores laterales, etc.

En la figura 1.1 se muestra un esquema básico de estos sistemas, así como de la interactuación entre ellos:

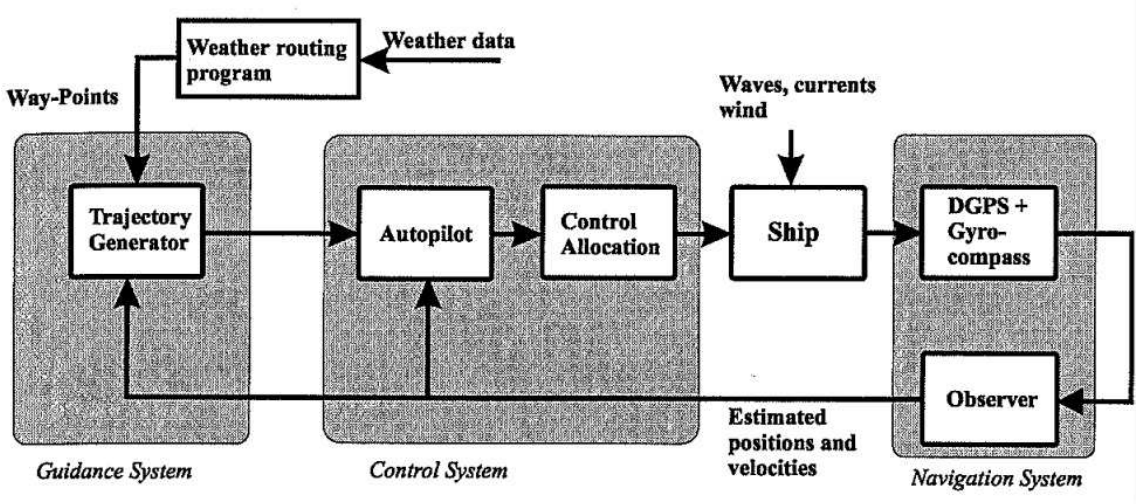

Figura 1.1. Esquema de control

Este trabajo principalmente se centra en el sistema de control de un barco. Los controladores pueden tener distintos tipos de funcionamiento: control de velocidad, estabilización de alabeo y cabeceo, control de posición, autopiloto o controlador de rumbo, etc. (los cuales, dependiendo de la aplicación pueden llegar a utilizarse de forma combinada).

En el caso concreto que se va a tratar, se diseñarán diversos controladores de autopiloto encargados de que el barco siga una ruta determinada a una cierta velocidad constante, a partir del rumbo definido.

Como se analizará más adelante, al quedar la ruta definida en el plano horizontal, solo son de interés los movimientos longitudinal, transversal y de rumbo. Además, considerando la velocidad de avance constante (controlada de forma independiente mediante los elementos de empuje del barco), el único actuador a controlar será el timón. Éste se utilizará para controlar el rumbo, mientras que de manera indirecta se tratará de controlar el desplazamiento lateral.

En general, dentro de la IA el trabajo se centra en técnicas de “soft computing” que abarcan una gama de metodologías basadas en propuestas computacionales inteligentes de la rama conexionista: redes neuronales artificiales (RNA), lógica difusa y algoritmos genéticos (AG).

Concretamente, se analizarán los resultados obtenidos con un controlador difuso y un controlador optimizado mediante un algoritmo genético. También se aplicará a modo comparativo un control PID clásico, y se analizarán en cada caso las desviaciones existentes respecto al rumbo que se deseaba seguir.

Por otro lado, se aplicará una RNA a modo de control adaptativo para contrarrestar el posible efecto de las diferentes perturbaciones que podrían afectar a la embarcación (mareas, oleaje o viento).

El objetivo principal es tratar de conseguir que los controladores inteligentes, valiéndose de sus características beneficiosas, permitan mejorar los resultados obtenidos con un control clásico. Principalmente se pretende minimizar los efectos de la sobrelongación, o sobrepico, que la respuesta del sistema pueda generar en el barco (estos efectos se analizar en los futuros apartados). Además, con el desarrollo de este trabajo se pretende estudiar cómo pueden afectar en el sistema algunas variaciones del sistema borroso, así como algunos de los aspectos configurables del algoritmo genético.

1.3. Estado del arte

En la actualidad los autopilotos controlan tanto el mantenimiento de un rumbo concreto como el cambio del mismo. Dentro de este tipo de sistemas existen diferentes modos de funcionamiento [2]:

- Modo manual.
- Modo dirección o “heading”: Se especifica el ángulo respecto a un sistema geográfico que debe mantener el eje longitudinal del barco.
- Modo punto de paso: El controlador debe conducir al barco a un punto determinado.
- Modo ruta: El barco debe seguir un camino determinado.
- Modo rumbo o “course”: Se especifica el ángulo que el vector de velocidad del barco debe mantener.

Dependiendo del modo de autopiloto a utilizar, la información a manejar por el controlador será distinta (será necesario disponer de: señal de interfaz de usuario para mando manual, información del rumbo del barco mediante giroscopio, posicionamiento mediante DGPS, etc.).

Este trabajo se centra en la utilización del Modo Rumbo para la simulación del control del barco. Al considerarlo como un sistema sub-actuado, los objetivos del controlador a aplicar en este proyecto se pueden lograr controlando en cada instante el rumbo del barco (sin necesidad de tratar otro tipo de señales que den información sobre la posición real del barco, etc.).

Existen infinidad de tipos de controladores, así como las distintas estrategias de control que pueden aplicarse para su desarrollo. Una de las más ampliamente utilizadas es el control PID [1]. Actualmente, un control PID puede utilizar, además de los sensores para la lectura del rumbo y su velocidad, acciones “feedforward” para realizar un seguimiento preciso de los cambios de rumbo y/o compensación de la acción de perturbaciones como el viento.

Se han propuesto muchos tipos de técnicas para el diseño de controladores de autopilotos de barcos. La literatura existente es muy amplia, y abarca desde el primer autopiloto adaptativo presentado en 1975 [3], hasta la aplicación de técnicas de “backsteeping” para la obtención de controladores que garantizan la convergencia de la orientación y posición respecto a un trayectoria de referencia ([4] y [5]). En [6] y [7] se diseña un algoritmo de guiado basado en la proyección de la LOS (Line-of-Sight) para moverse mediante puntos de paso unidos por segmentos rectos.

Actualmente los controladores que utilizan técnicas de IA están adquiriendo cada vez más relevancia, pudiendo encontrar diversos trabajos que aplican este tipo de técnicas al control de barcos. Se han propuesto autopilotos basados en técnicas de control inteligente, tales como lógica difusa ([8] y [9]) y redes neuronales [10].

1.4. Estructura del Trabajo Fin de Máster

La estructura del Trabajo se ha organizado en distintos capítulos, que pasan a describirse brevemente a continuación:

- En primer lugar se realiza una breve descripción del sistema. Se detallan algunas de las características del barco sobre el que se realizarán las simulaciones, el esquema del controlador, etc.
- En el siguiente capítulo se describe el modelo del barco que se utilizará, y se realizará un análisis tanto del movimiento, como de las diferentes trayectorias que se aplicarán sobre el barco.

- Los tipos de controladores que se analizarán se detallan en el capítulo 4. Además de describir las principales características de las técnicas utilizadas, se determinarán sus peculiaridades concretas para el presente caso.
- El siguiente capítulo mostrará los resultados de las diferentes simulaciones realizadas aplicando los controladores desarrollados bajo diversas trayectorias.
- En el capítulo 6 se analizan las RNAs, y se describe el sistema desarrollado para minimizar los posibles efectos de las perturbaciones en el funcionamiento del control del rumbo del barco. Se realizarán varias simulaciones para comprobar las ventajas que aporta la red neuronal diseñada.
- Finalmente se detallan brevemente las conclusiones obtenidas con el presente proyecto, y se ofrece una idea general de posibles trabajos a futuro que se podrían contemplar para completar y/o mejorar los resultados obtenidos.

Después de los apartados existentes para la bibliografía, y la definición de acrónimos utilizados, se incluyen a modo de anexo los diferentes resultados obtenidos a la hora de optimizar los controladores desarrollados. El controlador optimizado mediante AG dispone de una importante cantidad de elementos configurables, y la determinación de éstos parámetros se analiza en dicho Anexo.

2. Descripción del Sistema

En el presente capítulo se describe el vehículo autónomo marino sobre el que se va a trabajar. El objetivo es dar una visión general del conjunto de elementos que componen el sistema, sin entrar en detalles de implementación. En los posteriores capítulos se analizará en profundidad cada uno de los distintos bloques que forman el sistema completo de control.

En primer lugar se describen las características del barco que se ha modelado para la realización de las simulaciones del control. Posteriormente se analiza el sistema de control, describiendo los diferentes elementos que lo componen, y los tipos de trayectorias que se van a aplicar en las simulaciones.

2.1. Prototipo del barco

Este trabajo se centra en el control del rumbo de un vehículo marino autónomo, actuando sobre el timón del mismo:

Figura 2.1. Hélice y timón del barco

Algunos de los elementos de interés que dispone el barco son los siguientes:

- El barco dispone de un motor Beckhoff para el control del timón.
- Además, para el control de la propulsión el barco también dispone de un motor ABB. Sin embargo, se estimará que la velocidad del barco se mantiene constante durante todo el proceso de control del rumbo, quedando fuera del alcance de este trabajo el control de este motor.
- Otro elemento de interés del barco es su unidad de medición inercial o IMU. Concretamente se trata del modelo F-185 de Octopus, el cual dispone de un giroscopio de sistema microelectromecánico (MEMS). Este elemento permite medir en todo momento la orientación y rumbo que sigue el barco, y se utilizará para la realimentación del sistema de control del rumbo.

En general, la dinámica de un barco es bastante compleja. Se trata de un sistema no lineal de ecuaciones con coeficientes no constantes y fuertes acoplamientos. Sus movimientos son de tipo oscilatorio y presentan grandes inercias. Con mares de popa el movimiento vertical se puede considerar bastante estable, aunque con mares de proa, dependerá del estado de la mar (principalmente de la altura de las olas y de la frecuencia de encuentro [11]).

2.2. Esquema del sistema de control

Como se podrá ver con mayor detalle a continuación, el sistema se basará en controlar el rumbo del barco actuando únicamente sobre el timón del mismo. La siguiente imagen muestra

de forma esquemática una representación del plano XY horizontal del barco, donde ψ_d será el ángulo que se pretende obtener (rumbo deseado) y δ representará la posición del timón:

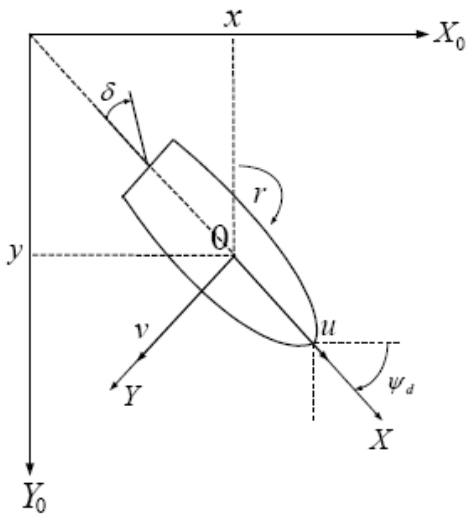

Figura 2.2. Plano XY del barco

De este modo, el diseño del controlador tendrá como objetivo actuar sobre el timón para tratar de mantener en el barco el rumbo deseado. Por consiguiente, el esquema básico para la simulación del funcionamiento del controlador de rumbo será el que se muestra en la siguiente figura 2.3:

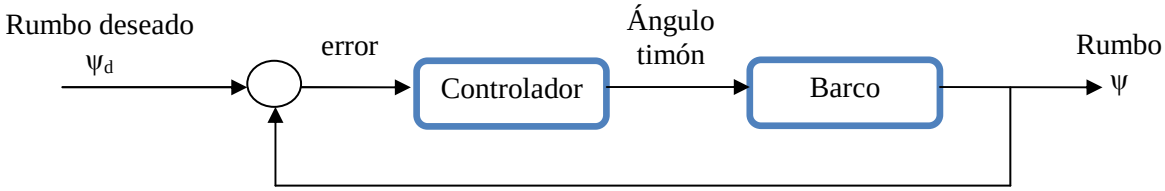

Figura 2.3. Esquema básico del sistema de control del rumbo

El sistema de control recibirá como orden un rumbo que se desea seguir, y mediante la realimentación de la lectura del mismo (obtenida con un giroscopio o similar, que proporcione información del rumbo real que sigue el barco), se generará una señal de control para que el controlador ajuste la posición del timón.

Dependiendo de la trayectoria que se desee obtener con el barco, se generará un $\psi_d(t)$ determinado. El rumbo $\psi(t)$ que se obtenga con la simulación servirá para calcular la trayectoria real seguida por el barco, comparándola con la que se deseaba obtener, analizando de este modo el error (tanto en posición como en rumbo) que se generará para cada uno de los tipos de controladores diseñados.

A la hora de generar el sistema definitivo, existirán algunos aspectos a matizar respecto al esquema anterior. Por ejemplo, tal y como se verá en el siguiente capítulo, el modelo del barco que se utilizará tiene como entrada la posición del timón, y su salida es la velocidad angular de giro del barco. De este modo, se aplicará un proceso integrador a la salida del modelo para obtener el rumbo del barco a partir de la velocidad angular. También se dispondrá de otro modelo, que a partir de la misma entrada entrega como salida la velocidad de deslizamiento del barco. Estos datos permitirán calcular las posiciones geográficas X e Y en todo momento, pudiendo representar gráficamente el recorrido real seguido por el vehículo marino.

En esta aplicación únicamente se pretende controlar el rumbo del barco mediante un lazo cerrado. No se contempla ningún tipo de monitorización de la posición real del barco, para en caso de existir una desviación respecto a una posición deseada, aplicar la correspondiente

corrección de rumbo en el sistema (por ejemplo, haciendo uso de señales DGPS que permitan conocer la ubicación exacta).

2.3. Tipos de trayectoria

Para analizar y comparar el funcionamiento de los diferentes controladores que se van a desarrollar en el presente trabajo, se utilizarán una serie de trayectorias específicas de diferente tipo que permitan estudiar la respuesta de cada controlador ante los cambios de rumbo.

Para eso se ha determinado emplear las siguientes trayectorias básicas, que por sus características supondrán diferentes tipos de cambios de rumbo sobre el barco:

- Trayectoria recta y aplicación de cambio de rumbo específico: Este tipo de trayectoria se basa en mantener el rumbo del barco fijo, para en un momento determinado aplicar un cambio de rumbo que podrá ser más o menos suave (en función de la cantidad de grados por unidad de tiempo que se desee modificar el rumbo).
- Trayectoria circular: Servirá para analizar la respuesta ante un cambio de rumbo constante para que el barco realice una trayectoria circular. El diámetro del círculo vendrá determinado por la velocidad angular a la que se le ordene virar al barco.
- Trayectoria en espiral: Es similar a la anterior, con la diferencia que la velocidad angular aumenta al realizar una vuelta (de este modo, cada vez se van realizando círculos de un menor diámetro). Este tipo de trayectorias pueden ser apropiadas en ciertos tipos de aplicaciones con barcos, como podrían ser la búsqueda de náufragos en una superficie concreta, etc.
- Trayectoria en zig-zag y trayectoria sinusoidal: Este tipo de trayectorias servirán para analizar el cambio de rumbo periódico, tanto en un sentido como otro. Se tendrá la libertad de fijar tanto la frecuencia, como la amplitud de este tipo de recorridos.

3. Modelado y análisis del movimiento

El presente capítulo contiene las bases matemáticas, así como los diferentes modelos que se utilizarán para el desarrollo de este trabajo. Los contenidos de este capítulo se han obtenido principalmente de la literatura relacionada con el modelado, guiado y control de vehículos en general, y de barcos en particular.

3.1. Dinámica del barco

La dinámica de un barco se puede obtener aplicando las leyes de Newton. El barco dispone de 6 grados de libertad o GDL (uno de traslación y otro de rotación en cada uno de los 3 ejes cartesianos x, y, z), ya que las 6 coordenadas son necesarias para determinar la orientación y posición espacial de este sólido rígido.

Estas 6 componentes se denominan:

- Avance
- Desplazamiento lateral
- Arfada
- Balanceo
- Cabeceo
- Guiñada

La Figura 3.1 representa los sistemas de coordenadas y la definición de los movimientos de traslación y rotación del barco.

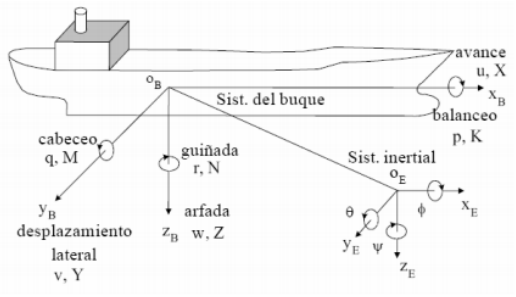

Figura 3.1. Sistema de coordenadas del barco

Del mismo modo, la tabla 3.1 muestra la nomenclatura que describe el movimiento de un buque, así como las fuerzas y momentos.

Traslación	Fuerza	Velocidad lineal	Posición
Avance	X	U	x
Desp. lateral	Y	V	y
Arfada	Z	W	z

Rotación	Momento	Velocidad angular	Posición
Balanceo	K	P	φ
Cabeceo	M	Q	θ
Guiñada	N	R	ψ

Tabla 3.1. Nomenclatura del movimiento del barco

Las ecuaciones de movimiento de un barco se pueden expresar de forma compacta del siguiente modo [1]:

$$M\dot{v} + C(v)v = \tau \quad (1)$$

Siendo M la matriz de inercias, C la matriz de fuerzas centrífugas y de Coriolis, *v* el vector de velocidades lineales y angulares del sólido rígido ($v=[u,v,w,p,q,r]^T$), y τ el vector de las fuerzas y momentos ($\tau=[X,Y,Z,K,M,N]^T$).

En el presente trabajo únicamente se quiere analizar el control del rumbo en el desplazamiento del barco al seguir diversas trayectorias, por lo que sólo se va a tener en cuenta el plano horizontal. De este modo, el modelo de 6 grados de libertad se puede simplificar a uno de solamente 3 GDL.

Para representar el movimiento en el plano horizontal se podría utilizar un modelo en el que únicamente se consideran los movimientos de avance, guiñada y desplazamiento lateral (los movimientos de balanceo, cabeceo y arfada se considerarán despreciables).

Si se sitúa el sistema de coordenadas OB del barco de la figura 3.1 coincidiendo con los ejes principales de inercia, la ecuación (1) se simplifica de la siguiente manera:

$$M_{RB(3x3)}\dot{v} + C_{RB(3x3)}(v)v = \tau_{RB(3x1)} \tag{2}$$

$$M_{RB(3x3)} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_G \\ 0 & mx_G & I_z \end{bmatrix} \tag{3}$$

$$C_{RB(3x3)}(v) = \begin{bmatrix} 0 & 0 & -m(x_G r + v) \\ 0 & 0 & mu \\ m(x_G r + v) & -mu & 0 \end{bmatrix} \tag{4}$$

$$\tau_{RB(3x3)} = [X \quad Y \quad N]^T \tag{5}$$

A partir de esas expresiones, se pueden obtener las siguientes ecuaciones:

Avance:	$m(\dot{u} - vr - x_G r^2) = X$
Desp. lateral:	$m(\dot{v} - ur + x_G \dot{r}) = Y$
Guiñada:	$I_z \dot{r} + mx_G(ur + \dot{v}) = N$

A continuación, partiendo de este desarrollo, se realizarán una serie de consideraciones aplicables al caso concreto que se va a tratar en este trabajo.

Se considerará que la velocidad transversal, la velocidad de guiñada y el ángulo del timón serán pequeños. Además, se supondrá que la velocidad de avance del barco no varía, manteniéndose constante el empuje en todo momento. Esto implicará que la primera ecuación (movimiento longitudinal, X) se puede desacoplar de las otras dos (movimiento transversal Y y guiñada N).

También se considerará que las velocidades transversal y de guiñada son nulas (v_0=0 y r_0=0). De este modo, para pequeñas perturbaciones alrededor de los valores nominales u_0, v_0 y r_0 el anterior sistema de ecuaciones se puede simplificar de la siguiente forma:

$$\begin{gathered} m\dot{u} = X \\ m(\dot{v} - u_0 r + x_G \dot{r}) = Y \\ I_z \dot{r} + mx_G(u_0 r + \dot{v}) = N \end{gathered} \tag{6}$$

La primera ecuación relaciona la velocidad del barco con el empuje de sus propulsores. Aunque esta relación sea no lineal, en el presente trabajo se considerará que la velocidad del barco se mantiene constante, y por lo tanto sólo se consideran las otras dos ecuaciones.

3.2. Modelo del barco

Para el desarrollo del presente trabajo se ha utilizado un modelo del barco calculado a partir de los datos obtenidos en las pruebas realizadas con una réplica del mismo a escala.

Se ha aprovechado el modelo generado por J.M. de la Cruz, David Moreno, Eva Besada, J.A. López, y J. Aranda. Este modelo ha sido obtenido experimentalmente a partir de las pruebas realizadas en el pantano de Valmayor, utilizando máquinas de soporte vectorial (SVM).

Concretamente, en el desarrollo del trabajo se van a utilizar dos modelos: uno de ellos servirá para obtener la velocidad angular del barco, y el otro para la velocidad de deslizamiento.

3.2.1. Modelo velocidad angular

El modelo para la velocidad angular tiene la siguiente función de transferencia:

$$G_w = \frac{KT_3s + K}{T_1T_2s^2 + (T_1 + T_2)s + 1}$$

Siendo:

$K=0.3573$

$T_1=0.8138$

$T_2=1.1500$

$T_3=0.9752$

La entrada de este modelo es el ángulo del timón, entregando como salida la velocidad angular de giro del barco.

3.2.2. Modelo de velocidad de deslizamiento

El modelo para la velocidad de deslizamiento tiene la siguiente función de transferencia:

$$G_v = \frac{K_vT_vs + K_v}{T_1T_2s^2 + (T_1 + T_2)s + 1}$$

Siendo:

$Kv=0.3759$

$Tv=0.0277$

En este caso, la salida será la velocidad de deslizamiento (velocidad perpendicular a la de avance).

3.3. Modelo de Trayectorias

Para la definición de los diferentes tipos de trayectorias a aplicar en el barco, se determinará una relación entre el rumbo deseado $\psi_d(t)$ y la velocidad angular necesaria a aplicar para conseguirlo. La función encargada de generar los rumbos a seguir, en primer lugar calculará la velocidad angular necesaria en cada momento $\omega_d(t)$, y utilizando una función integral generará $\psi_d(t)$.

Esto se podía haber realizado calculando directamente $\psi_d(t)$ a partir de la trayectoria que se desea seguir. Sin embargo, puede ser interesante disponer de la velocidad angular deseada. Por ello se ha optado por seguir este procedimiento para definir el rumbo deseado:

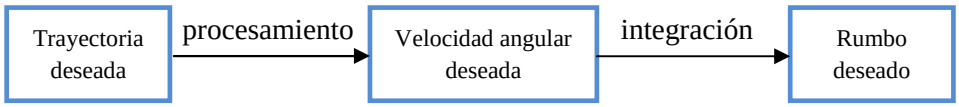

Figura 3.2. Esquema de obtención del vector de rumbo deseado

En función del tipo de trayectoria que se quiera realizar, la velocidad angular deseada, $\omega_d(t)$ se generará de las siguientes formas:

- Mantener rumbo: $\omega_d(t)=0$.
  - Si $\omega_d=0$, no hay cambio de rumbo y por lo tanto en barco sigue en la misma dirección. Su integral será un valor constante, lo que implica mantener el rumbo en el tiempo.

- Trayectoria circular: $\omega_d(t)$=constante.
    - Al ser ω_d un valor constante (X grados/segundo), el barco ira virando la misma cantidad de grados por unidad de tiempo, trazando así un circulo. En función del valor que se le de a la constante X, se podrá determinar el tamaño del diámetro del círculo.
- Trayectoria en zig-zag: $\omega_d(t)$= tren de pulsos.
    - Durante el tiempo que transcurra el impulso, el barco virará un número de terminado de grados. Hasta el siguiente impulso ω_d será cero, y por lo tanto el barco mantendrá el rumbo constante. El siguiente impulso tendrá el signo opuesto, y por lo tanto el barco virará en dirección contraria hasta fijar la siguiente dirección a seguir.

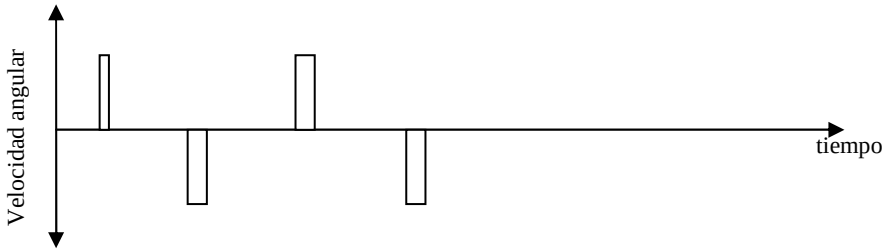

Figura 3.3. Velocidad angular para una trayectoria en zig-zag

    - El ancho (tiempo) y la altura (grados/segundo) de cada pulso, así como la distancia entre pulsos, determinarán la forma de la trayectoria en zig-zag
    - También se definirá un tipo de trayectoria similar a la de zig-zag, tomando como $\omega_d(t)$ una señal sinusoidal. En este caso serán la amplitud y frecuencia de la señal las que marcarán la forma de la trayectoria.
- Trayectoria en espiral: $\omega_d(t)$= función escalera.
    - Otro tipo de trayectoria interesante sería una especie de recorrido en forma de espiral. En principio este tipo de trayectoria podría ser aplicable a barcos orientados a tareas de rescate, búsqueda de náufragos, etc. Partiendo de una trayectoria circular amplia, se irán realizando círculos cada vez más pequeños y así cubrir la mayor parte de la superficie interior de la circunferencia de mayor diámetro.
    - Para esto se utilizará una función de tipo escalera. Calculando la forma de cada escalón (tanto amplitud como tiempo de duración), se definirán los diferentes recorridos circulares, y por lo tanto la forma de la trayectoria en “espiral”.

De este modo, se podrá analizar cualquier tipo de trayectoria deseada, y utilizando este procedimiento se calculará la velocidad angular deseada $\omega_d(t)$ para aplicar dicho recorrido en el sistema de control.

Una vez que se han definido los diferentes tipos de trayectorias, uno de los aspectos que mayor efecto tendrá en los resultados del sistema será el determinar el tiempo requerido para obtener un rumbo deseado.

La obtención de un rumbo determinado dependerá de la velocidad angular requerida y el tiempo durante la que se aplica. De este modo, se pueden realizar cambios de rumbo suaves (variaciones pequeñas de la velocidad angular aplicadas durante mayor tiempo) o bruscos (variaciones mayores de la velocidad angular aplicadas durante menor tiempo). Lógicamente, los cambios de rumbo suaves requerirán mayor tiempo para alcanzar el rumbo deseado. Sin embargo, aumentar la velocidad de giro para tratar de obtener el rumbo deseado en el menor tiempo posible puede provocar la aparición de respuestas con mayor sobrelongación, oscilaciones, etc., empeorando la respuesta general del sistema.

Por lo tanto, será un aspecto importante el conseguir optimizar la relación entre el tiempo necesario para alcanzar el rumbo deseado, tratando de realizar el giro de la forma más suave posible (minimizando los efectos derivados de giros bruscos, como pueden ser los derivados de la sobrelongación, oscilaciones hasta obtener el régimen estacionario, etc.).

3.4. Modelo del sistema completo

A partir de los aspectos definidos, se puede describir el sistema a partir del siguiente diagrama de bloques:

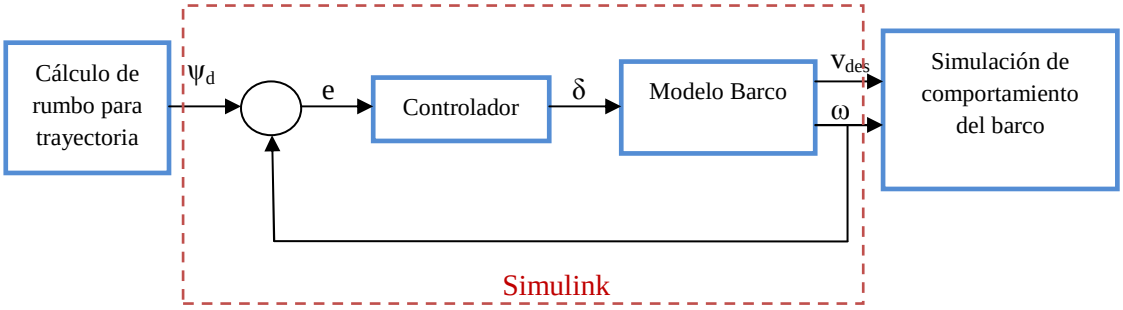

Figura 3.4. Diagrama de bloques del sistema de control

En primer lugar, se dispone del bloque encargado de calcular el vector de rumbo deseado. En función del tipo de trayectoria que se desee realizar con el barco, esta parte del sistema se encargará de calcular la velocidad angular a aplicar en cada momento, y mediante una función integradora se generará el rumbo deseado $\psi_d(t)$. Esta funcionalidad se calculará mediante código desarrollado en Matlab, y se pasará a Simulink para su procesamiento.

La diferencia entre esa señal de consigna y la realimentación del rumbo real que se calculará a partir del modelo del barco será la señal de entrada al controlador correspondiente. Tal y como se ha comentado, se desarrollarán tres tipos de controladores distintos para analizar su funcionamiento en diferentes escenarios.

En cada caso, la salida del controlador simulará la orden de control al timón, y está será la entrada para las funciones de modelado del barco. Las dos salidas que se generan aquí (velocidad angular y velocidad de deslizamiento), se pasarán a Matlab para su procesamiento y cálculo de las condiciones del barco a lo largo de su recorrido (rumbo, posiciones XY, etc.).

Se debe tener en cuenta la necesidad de aplicar una función integradora a la salida de la velocidad angular del modelo antes realizar la realimentación del sistema, para que de este modo se calcule correctamente la diferencia de error en el rumbo.

Además de estos bloques, a la hora de definir los diferentes controladores se añadirán otras funcionalidades para aportar una mayor robustez al sistema. Por ejemplo, para los casos en los que el controlador tienda a realizar acciones demasiado agresivas, pudiendo llegar a sobrepasar los límites físicos del propio actuador, es interesante limitar la actuación del controlador, saturando la salida del mismo. De ese modo, se van a establecer como ángulos máximos de giro del timón +40º y -40º, por lo que a la salida del controlador se incluirá un elemento de saturación.

4. Tipos de controladores aplicados

En el presente capítulo se describirán los tres tipos de controladores utilizados para simular el control del rumbo en el barco.

En primer lugar se analizará el controlador PID clásico de lazo cerrado. A continuación se presenta una breve descripción de los algoritmos genéticos, para así describir el desarrollo del controlador optimizado mediante AG. Finalmente, se describirá el controlador difuso realizado.

4.1. Controlador PID clásico

El primero de los tres tipos de control del rumbo del barco que se analizarán en este trabajo es un regulador PID clásico. Es un control en lazo cerrado ya que se utilizan sensores para realimentar el controlador con el estado del sistema.

Los controladores PID se basan en la combinación de tres actuaciones: Proporcional (relativa al error actual), Integral (relativa a los errores acumulados o pasados), y Derivativa (la cual proporciona una predicción lineal del error futuro) [12].

La figura 4.1 muestra el esquema tradicional de este tipo de controlador:

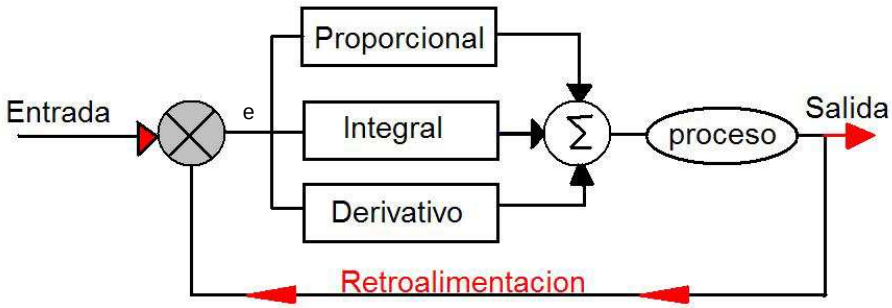

Figura 4.1. Esquema del controlador PID

En este esquema la entrada al controlador la denominaremos e(t), error del sistema. Como se puede observar se trata de la diferencia entre el valor de consigna y el estado del sistema. La salida del controlador será la señal de control o actuación sobre la planta (en este caso, la señal de control del timón).

La función del controlador PID es la siguiente:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \qquad (7)$$

En esta representación se pueden ver los tres tipos de actuaciones de forma diferenciada. La elección de esos tres valores o ganancias (K_p, K_i y K_d) determinará el comportamiento del controlador.

- Ganancia proporcional (K_p): El aumento de este valor hace que el controlador sea más "agresivo", haciendo que pueda alcanzar el valor de referencia en menos tiempo. Sin embargo, un valor demasiado alto puede provocar que el sistema sea inestable y/o presente grandes oscilaciones.
- Ganancia integral (K_i): El aumento de este valor permite eliminar antes los errores en el estado estacionario. Sin embargo, un valor muy alto puede generar una gran sobreoscilación inicial (también conocida como sobrelongación), y para algunos sistemas, esta oscilación es inaceptable.
- Ganancia derivativa (K_d): El aumento de este valor permite disminuir la sobrelongación, a costa de hacer al controlador menos agresivo, y por lo tanto más lento. La ganancia derivativa es muy sensible a ruidos en la señal, los cuales pueden llevar al sistema a la inestabilidad.

Determinar unos valores incorrectos puede generar una considerable inestabilidad en el sistema, por lo que con este tipo de controladores es importante ajustar los tres parámetros a sus valores óptimos.

Los efectos de cada uno de los controladores K_p, K_d, y K_i en un sistema de lazo cerrado están resumidos en la tabla 4.1 que se presenta a continuación. Se debe tener en cuenta que estas correlaciones podría no ser exactamente precisas, ya que el efecto de cada parámetro será dependiente de los otros. Es por ello que la tabla simplemente se muestra a modo representativo:

Parámetro	Tiempo de subida	Sobrelongación	Tiempo de establecimiento	Error estacionario
K_p	Disminuye	Aumenta	Cambia poco	Disminuye
K_i	Disminuye	Aumenta	Aumenta	Eliminado
K_d	Cambia poco	Disminuye	Disminuye	Cambia poco

Tabla 4.1. Tabla resumen de efecto de los parámetros

Existen diferentes formas de realizar el ajuste de un controlador PID (como son el ajuste empírico, el método Ziegler-Nichols, métodos de lazo abierto y lazo cerrado, etc.). De todos modos, es importante conocer el funcionamiento de este tipo de controladores, ya que variaciones concretas en las ganancias pueden generar resultados más o menos predecibles en relación al comportamiento del controlador.

En el caso concreto que nos ocupa, se utilizarán unos parámetros optimizados que han sido definidos junto con el modelado del barco. De este modo, los valores del controlador PID han sido fijados a unos que ofrecen un resultado con poco sobrepico:

- K_p=3.2
- K_i=0.75
- K_d=4.0

4.2. Controlador optimizado con AG

El siguiente tipo de control que se va a analizar es un controlador en el que se calcularán los valores de los parámetros K_p, K_i y K_d mediante un algoritmo genético.

Tal y como se verá en el siguiente capítulo, cuando se analicen los resultados obtenidos mediante la simulación del funcionamiento de los diferentes controladores, el controlador PID clásico presentará unos resultados en función de las características del cambio de rumbo aplicado. Aprovechando las características de optimización que ofrecen los AG, se pretende conseguir un controlador más flexible, que pueda ajustar su salida en función del tipo de cambio de rumbo, tratando de reducir aún más la sobrelongación, independientemente del tipo de trayectoria a realizar por el barco.

4.2.1. Algoritmos genéticos

Los Algoritmos Genéticos reciben ese nombre por inspirarse en la base genética de la evolución biológica. Se basan en hacer evolucionar una población de individuos mediante acciones aleatorias como las que ocurren en la evolución biológica (mutaciones, recombinaciones, etc.), y haciendo que sobrevivan los que son considerados individuos más "aptos" [13].

Los AG son métodos adaptativos, que en general se pueden utilizar en todos los problemas que requieran ciertos procesos de optimización. Aunque estos algoritmos no garantizan encontrar la solución óptima de un problema, son adecuados para encontrar una posible solución en un tiempo "aceptablemente rápido" (siempre que se realice un diseño correcto, ya que esto es un aspecto imprescindible para conseguir que la población converja a una solución óptima).

De este modo, la idea fundamental de estos algoritmos consiste en encontrar una solución aceptable a un problema por medio del mejoramiento de un conjunto de individuos, cuya función de evaluación corresponde a una solución del problema.

Los principales aspectos a tener en cuenta a la hora de definir el algoritmo genético serán:

- Genes: Representan el conjunto de parámetros del problema.
- Cromosomas: Unión de los diversos genes de un individuo.
- Genotipo: Definición utilizada para el paquete genético total.
- Fenotipo: Interacción del genotipo con su entorno.
- Población: Conjunto de individuos existentes en una generación.
- Función objetivo: Función que proporciona una medida de desempeño del sistema asociado a cada individuo. Al ser una función completamente dependiente del problema, puede determinar qué tan buenos o malos serán los resultados y la convergencia o no del método.
- Operador de selección: Proceso que busca escoger ciertos individuos de la población, que serán los encargados de dar origen a la siguiente generación (normalmente la selección dependerá del valor de la función objetivo, para tratar de elegir los individuos más "aptos").
- Operador de cruce: Acción que consiste en mezclar la información genética de dos individuos para generar nuevos individuos.
- Operador de mutación: Acción que consiste en modificar las características genéticas de un individuo, con el objeto de aumentar la probabilidad de exploración del espacio de búsqueda y disminuir el riesgo de estancamiento del algoritmo en óptimos locales.
- Sustitución generacional: Una vez se dispone del nuevo conjunto de descendientes creados mediante la selección-cruce-mutación, se evalúan y se decide que individuos pasarán a formar la siguiente generación.

4.2.2. Características del controlador AG para el controlador del rumbo

El objetivo del ajuste de los tres parámetros que definen el controlador PID es lograr que el bucle de control corrija eficazmente y en el mínimo tiempo los efectos de las perturbaciones. Del mismo modo, se pretende reducir el efecto de la sobrelongación lo máximo posible. Si los parámetros del controlador se escogen de forma incorrecta, el proceso a controlar podría ser inestable. Así, se pretende diseñar un algoritmo genético que calcule los parámetros K_p, K_i, K_d más adecuados posibles para aplicarlos al controlador PID

La siguiente imagen muestra del diagrama de bloques del sistema que se pretende desarrollar:

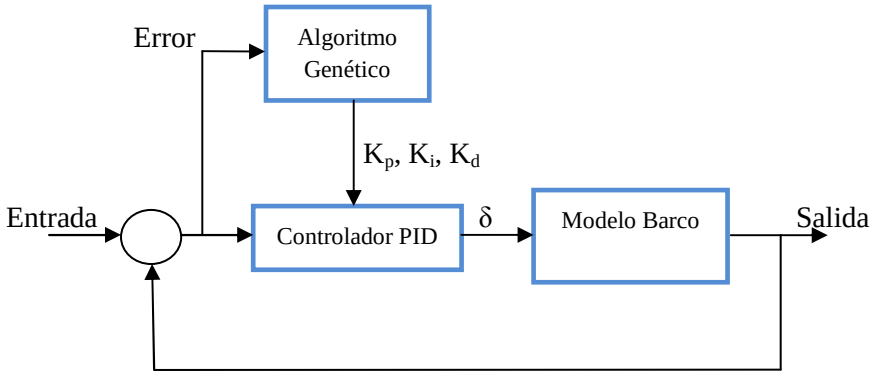

Figura 4.2. Diagrama de bloques del controlador sintonizado mediante AG

Tal y como se ha descrito anteriormente, el utilizar un algoritmo genético para determinar el valor de los parámetros PID no garantiza encontrar la solución óptima del problema, pero servirá para conseguir una solución aceptable.

Debido a sus características, existen infinidad de opciones a la hora de definir el algoritmo que afectarán de diferente forma a la solución obtenida (como por ejemplo: definir los cromosomas, tipos de funciones objetivo, como realizar la selección, el cruce o la mutación, cantidad de individuos y generaciones a definir, probabilidades de cruce/mutación, etc.). El efecto de cambios en cualquiera de estas opciones es en ocasiones difícil de predecir.

Por ello se ha tomado la decisión de desarrollar el programa completo para simular el algoritmo genético (sin utilizar ningún tipo de toolbox o herramienta ya existente), y de este modo conseguir una libertad total para determinar que características concretas se desean aplicar al algoritmo, y probar así el efecto que pueden tener en el resultado final.

A continuación se definen algunos de los aspectos que el programa que se ha realizado permite cambiar, permitiendo así el tratar de ver el posible efecto que pueden causar en el algoritmo. Esto ayudará a escoger las características particulares que tendrá el algoritmo a la hora de realizar el cálculo de los parámetros definitivos que serán aplicados al controlador en el momento de la simulación.

Algunos de estos aspectos variables son habituales de todos los algoritmos genéticos (nº de individuos, generaciones, etc.), y otros más específicos de este programa en concreto (tipos de reinserción a realizar, posibilidad de función objetivo variable, etc.):

- Forma del cromosoma: Se ha dejado como opción la posibilidad de determinar la cantidad de genes que forman el cromosoma (considerando cada gen como uno de los parámetros del controlador). Lo habitual será definir esta variable con el valor 3 (K_p, K_i y K_d), sin embargo, el dejar esta variable como configurable permitiría hacer pruebas para controladores del tipo PI, PD, etc. También se ha dejado como variable la cantidad de bits que dispondrá cada cromosoma, de este modo, a mayor nº de bits se conseguirá una mayor precisión de los valores (pero a costa de una mayor carga computacional).
- Probabilidades: El valor de las probabilidades de cruce y mutación se guardan en unas variables, permitiendo ajustarlas para realizar las diferentes simulaciones.
- Tipo de función objetivo: El código que se encarga de obtener la función objetivo (que determina como de buena es la solución que proporciona cada individuo) permite la posibilidad de aplicar diferentes tipos de funciones, dependiendo del tipo de cambio de rumbo que se desee aplicar al barco. De este modo, se conseguirá un controlador PID de ganancias programadas, permitiendo obtener soluciones más orientadas a optimizar los valores en caso de cambios de rumbo suaves en trayectorias circulares, cambios de rumbo rápidos etc. Por otro lado, el código también permite una opción para determinar si la función objetivo se calculará a partir del Error Cuadrático Medio (MSE) o de la Integral del Tiempo por el Error Absoluto (ITAE) [14] y [15].
- Tipo de re-inserción de individuos: El programa realizado permite seleccionar la forma en que se realiza la selección de los individuos que formarán la siguiente generación. De este modo, se puede optar por una opción que crea la siguiente generación únicamente a partir de los nuevos individuos generados de las operaciones de cruce entre padres, o una opción diferente que crea la siguiente generación a partir de los individuos más “aptos” del conjunto de individuos de la generación actual (padres) y los nuevos creados de las operaciones de cruce-mutación (hijos).
- Elección de la solución final: El programa irá almacenando el mejor individuo de cada generación en un vector. De este modo, una vez finalice el algoritmo, se podrá elegir como solución final, bien el mejor individuo de la última generación, o bien el mejor individuo de dicho vector.

4.2.3. Esquema de funcionamiento del AG

Básicamente, el diagrama de flujo del programa desarrollado para el cálculo de los parámetros PID mediante la aplicación de un algoritmo genético se ha definido como el mostrado en la figura 4.3.

Siguiendo este diagrama, el esquema de funcionamiento del programa realizado se puede resumir de la siguiente forma:

- El algoritmo comienza definiendo una población inicial de forma aleatoria, en la que cada individuo representa una solución concreta de los parámetros del controlador.
- A continuación se inicia un bucle para que comience la evolución de los individuos a lo largo de nuevas generaciones.
- En cada iteración, los individuos se evaluarán, se aplicarán las operaciones de cruce y mutación, y se seleccionarán los cromosomas que conformarán la siguiente generación.

- Una vez finalizado el bucle, se seleccionará el individuo que representa la mejor solución al sistema.

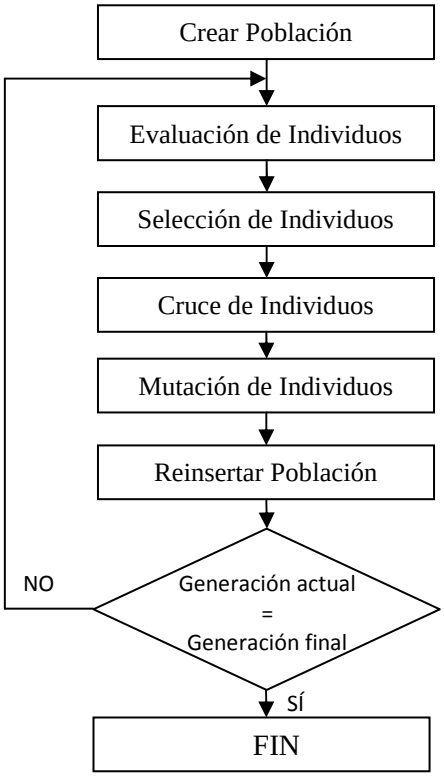

Figura 4.3. Diagrama de flujo del algoritmo genético

Partiendo de este esquema básico, a continuación se analizará con mayor profundidad cómo se realizan las diferentes operaciones, entrando en detalle en la funcionalidad de cada una de las partes que conforman el programa completo.

El programa principal se denomina "*pid_ga.m*". En primer lugar se definen los valores de las diferentes variables que toman parte en el algoritmo (nº de individuos de la población, máximo nº de generaciones, tipo de reinserción, tipo de función de objetivo a aplicar, probabilidades de mutación, etc.).

A continuación se realiza una llamada a la función "*crear_poblacion.m*". Ese código es el encargado de crear de forma aleatoria todos los individuos de la población inicial. Cada individuo se almacena de forma binaria en una fila de la matriz que representa al cromosoma.

La función "*convertir_crom.m*" es la encargada de realizar la conversión de cada individuo, transformado la cadena de 1s y 0s en los valores reales correspondientes a los parámetros del controlador. Estos valores son utilizados por la función "*Func_Obj.m*", que tiene como cometido calcular el error de la respuesta del sistema formado por el modelo del barco y el controlador PID correspondiente a cada individuo.

Los valores devueltos por esta función sirven para estimar cuales son los individuos que mejor resultado dan al problema.

A continuación se inicia un bucle con tantas iteraciones como número de generaciones se han definido:

- En primer lugar se ejecuta la función "*fitness.m*", encargada de calcular el desempeño de cada individuo, generando un vector de valores que servirá para aplicar el Método de la Ruleta en la selección de individuos.
- Se realiza la llamada a la función "*selección.m*", que seleccionará los individuos que se utilizarán para las operaciones de cruce y mutación por medio del Método de la Ruleta.
- Se ejecuta la función "*cruce.m*", la cual recombina los individuos de dos en dos cruzando sus cromosomas a partir de una posición (o bit) aleatoria.

- A continuación, se realiza la función de mutación ("*mutacion.m*") sobre los individuos que han sido cruzados.
- Sobre estos nuevos individuos creados a partir de las funciones de selección-cruce-mutación, se realiza una evaluación para determinar su aptitud respecto al problema definido (mediante las funciones anteriormente descritas "*convertir_crom.m*" y "*Func_Obj.m*"). Se realiza una llamada a la función "*reinsertar.m*", en la que se pasan como parámetros de entrada los cromosomas de la población actual y los nuevos que han sido generados. Dependiendo del tipo de reinserción que se haya definido mediante la variable correspondiente está función creará la nueva generación cogiendo únicamente los nuevos individuos creados (hijos), o por el contrario, seleccionando los mejores individuos existentes en el conjunto de la población actual y los nuevos individuos creados a partir de ellos (mejores individuos del conjunto de padres e hijos).

Una vez realizadas todas las iteraciones del bucle, se dispondrá de una población final. Se seleccionará el individuo que proporcione el menor error en el sistema, determinando sus valores como los parámetros definitivos del controlador PID.

4.2.4. Optimización de AG

Tal y como se ha indicado, el algoritmo desarrollado dispone de una serie de elementos configurables para tratar de optimizar la solución de la mejor forma posible.

En este apartado se describirá como han quedado finalmente definidos estos aspectos a la hora de aplicarlos en el controlador PID optimizado por GA que se utilizará en la simulación del control del rumbo.

A continuación, únicamente se va a dar una breve descripción de cómo ha quedado configurado el algoritmo. Para analizar con más detalle el razonamiento para tomar estas decisiones, en el Anexo I se incluyen los resultados de las diferentes pruebas a las que se hace referencia.

Algunas de las variables configurables del algoritmo tienen un efecto directo en el tiempo de ejecución del programa. Uno de los objetivos del algoritmo es tratar de reducir al máximo el tiempo de cálculo, para obtener así la solución lo más rápido posible. Esto puede ser un aspecto de gran importancia, si por ejemplo, se pretende que la aplicación funcione de manera que el controlador pueda cambiar online los parámetros optimizados por el algoritmo durante el trayecto en función de solicitudes de cambio de rumbo, etc.

Los aspectos que tienen una mayor relevancia en el tiempo de ejecución del algoritmo son el nº de individuos a definir en la población, así como la cantidad de generaciones a simular. Existen otros aspectos, como por ejemplo, el tiempo de simulación definido para el cálculo de la función objetivo de cada cromosoma, etc., pero se ha verificado en las pruebas que este aspecto es despreciable frente a los otros dos.

Se han realizado diversas pruebas para tratar de buscar un balance apropiado para determinar el valor óptimo de estas variables, ya que una reducción excesiva de nº de individuos y/o generaciones puede suponer que el algoritmo no proporcione una solución suficientemente buena. Tras analizar los resultados expuestos en el Anexo I, se ha optado por considerar que la utilización de una población de 40 individuos, procesada durante 40 generaciones ofrece unos resultados aceptables, en un tiempo relativamente bueno, y con una diferencia despreciable entre sucesivas repeticiones del algoritmo bajo las mismas condiciones.

Por otro lado, se ha optado por fijar las variables de probabilidad de cruce y probabilidad de mutación en 1 y 0.01 respectivamente.

Tal y como se ha comentado anteriormente, se han definido dos tipos de reinserciones aplicables al algoritmo. La primera crea la siguiente generación a partir de la selección entre los mejores padres e hijos. En la segunda las siguientes generaciones se forman únicamente con los nuevos individuos creados, o dicho de otra forma, la generación i+1 estará formada únicamente por los hijos generados a partir de los individuos de la generación i.

Es factible pensar que la primera opción reduce el abanico de nuevos individuos creados, ya que puede haber padres que pasen de generación en generación, reduciendo las posibilidades de incluir nuevos individuos. De todos modos, en las pruebas se ha podido ver claramente que la

reinserción a partir de una selección de mejores individuos (padres/hijos) mejora los resultados obtenidos.

Finalmente, un aspecto crítico en la definición del algoritmo es la utilización de la función objetivo variable. Al ser este un punto relevante en la obtención de los resultados, en el siguiente capítulo se mostrará un análisis donde se muestran los beneficios de aplicar una función objetivo adaptable en lugar de una fija.

4.3. Controlador Difuso

El tercer tipo de control del sistema se ha realizado mediante un controlador difuso, desarrollado utilizando la toolbox *fuzzy* de Matlab [16].

4.3.1. Lógica difusa

Los sistemas de control difuso o borroso utilizan expresiones difusas para formular las reglas que controlan el sistema. En este tipo de sistemas debe tenerse en cuenta el conocimiento de expertos para la realización de las reglas o base de conocimientos, ya que la toma de decisiones se fundamentará en lo que se defina en ellos [17].

El control difuso se aplica a innumerables tipos de aplicaciones, principalmente aquellas en los que los modelos matemáticos son muy complejos. De este modo, a partir de técnicas de razonamiento aproximado es factible realizar un control adecuado incluso cuando el entorno no se conoce de forma precisa.

La siguiente figura muestra los diferentes bloques que formarían un sistema de control difuso:

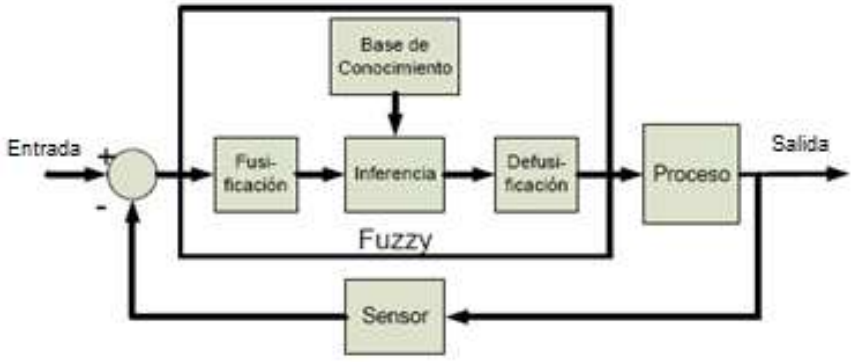

Figura 4.4. Diagrama de bloques de un controlador difuso

La fuzzificación es el proceso mediante el cual se transforma un valor numérico concreto de una variable de entrada en un conjunto borroso que representa dicho valor de forma imprecisa.

La base de reglas es la parte de la arquitectura del controlador donde se almacena el conocimiento en forma de reglas. El formato más habitual para representar las reglas es del tipo IF-THEN, pudiendo aplicar conectivos lógicos (and, or, etc.) y/o modificadores para modelar la base de reglas. Al diseñar este bloque se deberán considerar numerosos aspectos además de la propia definición de las reglas, como son el número de entradas y salidas, número de conjuntos y sus funciones, el solapamiento, etc.

El mecanismo de inferencia es el encargado de activar las reglas (que pueden ser una, varias, o ninguna) de la base de reglas.

Una vez procesada la entrada por el mecanismo de inferencias, la defuzzificación se encarga de obtener un valor de salida preciso (representativo del conjunto borroso), que se mandará al proceso a modo de señal de control.

Las características de la lógica difusa hacen que este tipo de control sea muy apropiado para aplicaciones en las que nos encontraremos con situaciones de incertidumbre, convirtiéndose en una alternativa muy interesante para este tipo de controladores.

4.3.2. Diseño de controlador difuso del barco

Utilizando la toolbox *fuzzy* de Matlab se ha definido un controlador PD que se encargará de actuar sobre el timón del barco para tratar de obtener el rumbo deseado.

La siguiente figura muestra el esquema básico del controlador difuso que se ha desarrollado para esta aplicación:

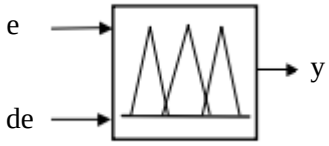

Figura 4.5. Esquema del controlador

Como se puede observar, para el controlador difuso se han definido dos entradas y una salida. La entrada "e" es el error en el rumbo del barco (diferencia entre el rumbo deseado ψ_d y el real obtenido por medio de la realimentación del sistema). La entrada "de" es la derivada del error. La salida "y" del controlador representa la señal de actuación sobre el ángulo del timón del barco.

Tal y como muestra la siguiente imagen, a la hora de definir el controlador se ha optado por un modelo tipo Mamdani. En la imagen también se pueden observar los tipos de operadores, métodos de defuzzificación, etc. por lo que se ha optado a la hora de determinar las características del controlador:

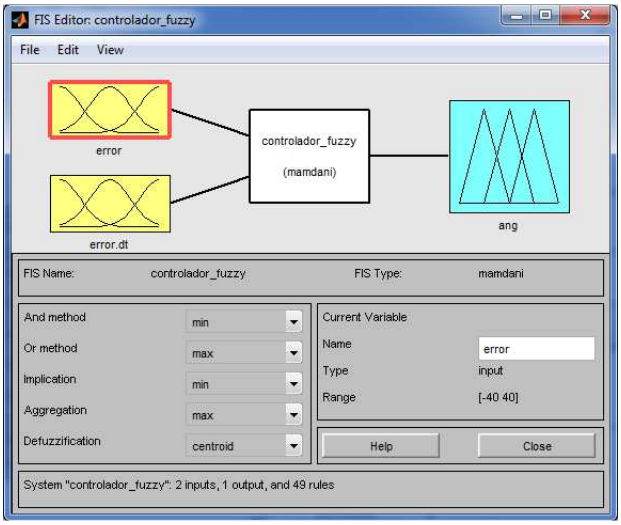

Figura 4.6. Definición del controlador mediante la toolbox

4.3.2.1. Entradas

Para la señal de entrada correspondiente al error en el rumbo, se ha considerado el dominio de números reales [-40, 40], mientras que para el caso de la señal de entrada correspondiente al cambio de error en el tiempo (de/dt) se tomarán los valores en el rango de [-15, 15].

Sobre cada uno de estos dominios se definen 7 conjuntos borrosos no simétricos. Estos conjuntos han quedado definidos de esta forma tras la realización de varias pruebas y simulaciones. En función de su valor, desde el mínimo valor negativo al máximo positivo, estos conjuntos se han denominado de la siguiente forma: NB (NegativeBig), NM (NegativeMedium), NS (NegativeSmall), ZE (ZEro), PS (PositiveSmall), PM (PositiveMedium) y PB (PositiveBig).

La siguiente figura muestra la definición de las dos señales de entrada, donde se puede observar que las funciones de pertenencia tienen forma triangular:

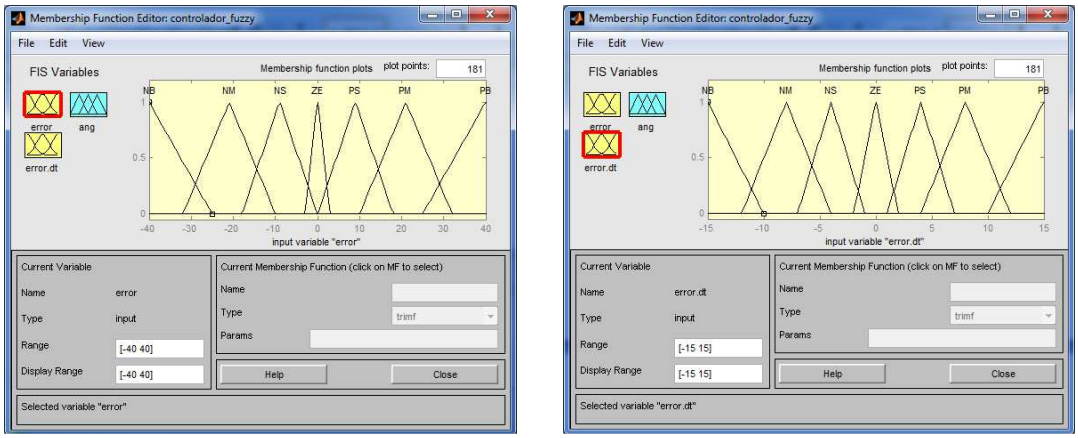

Figura 4.7. Conjuntos borrosos de las entradas

4.3.2.2. Salidas

Para la señal de salida se han definido 7 conjuntos borrosos en función del ángulo que se tiene que aplicar al timón. Como se puede verificar en la figura 4.8, la denominación de estos conjuntos es la misma que para las señales de entrada: NB, NM, NS, ZE, PS, PM y PB. También se puede observar que las funciones de pertenencia tienen forma triangular, salvo las de ambos extremos, correspondientes a una actuación sobre el timón para giros que impliquen generar velocidades angulares altas en el barco (ángulos máximos del timón, tanto en un sentido como otro). Para estos casos, tras realizar diferentes ensayos se ha optado por unas utilizar funciones del tipo zmf y smf. El rango es el del rumbo, es decir, de -40º a 40º.

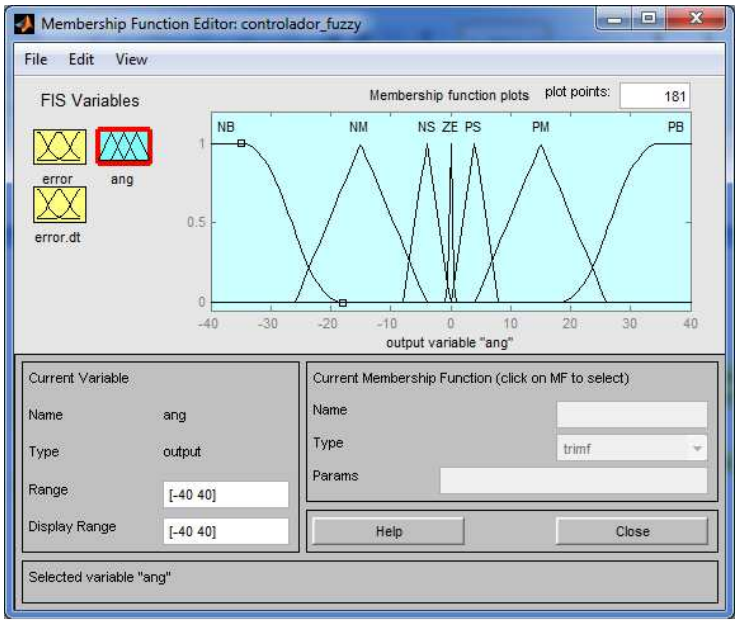

Figura 4.8. Conjuntos borrosos para la salida

4.3.2.3. Reglas

Para el controlador se han definido 49 reglas borrosas, considerando todas las posibles combinaciones de las entradas. La siguiente figura muestra el detalle de las reglas borrosas que se han definido mediante el editor de reglas:

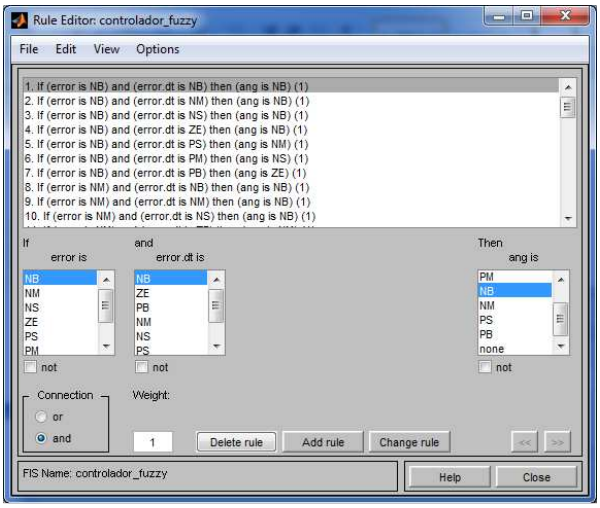

Figura 4.9. Editor de reglas

A partir de estas definiciones, Matlab permite obtener la superficie de control que determina el comportamiento del controlador difuso:

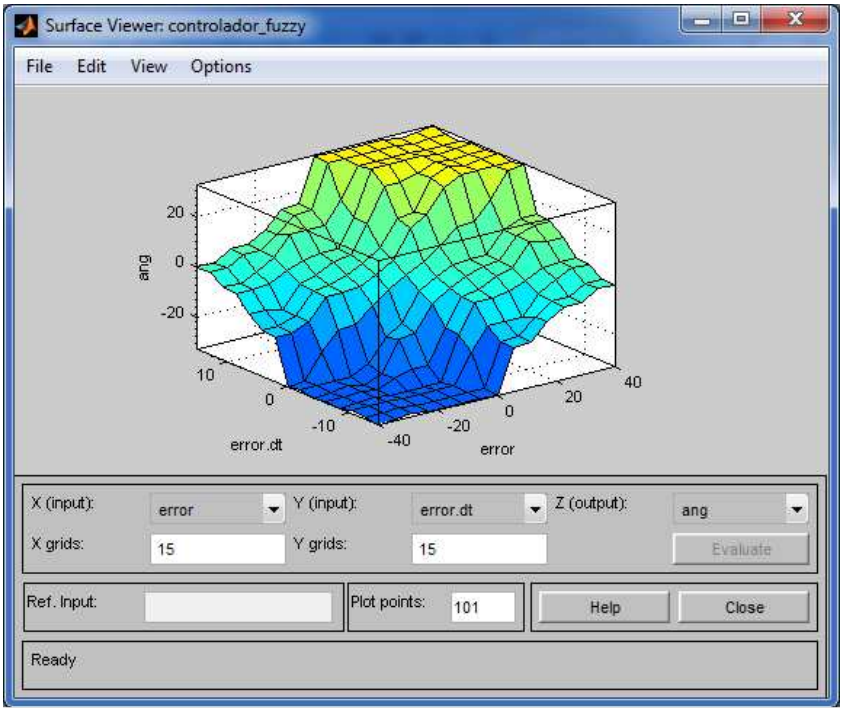

Figura 4.10. Superficie de reglas del controlador difuso

4.3.2.4. Ajuste del controlador

Teniendo en cuenta las posibles combinaciones de entradas y sus correspondientes salidas, el controlador difuso puede convertirse en un elemento muy laborioso de sintonizar. Tras realizar varios ensayos, finalmente se ha optado por dejar fijas las reglas y los conjuntos borrosos, y aplicar unas ganancias a la entrada de las variables de error y de derivada del error, así como a la salida de control.

Aplicar cambios en esas tres ganancias es equivalente a modificar las funciones de pertenencia del controlador difuso, por lo que se ha utilizado este procedimiento para ajustar el controlador realizando diversos ensayos de prueba. Para esta parte ha sido de ayuda utilizar el bloque de Simulink “Fuzzy Logic Controller with Ruleviewer”, ya que permite verificar el

tiempo real el estado del controlador (el valor del error, su derivada, la señal de salida, los conjuntos borrosos activados, etc.)

El siguiente diagrama de bloques tomado de Simulink muestra como ha quedado finalmente definido el controlador completo, y los valores finales de las ganancias que se han fijado para las tres señales:

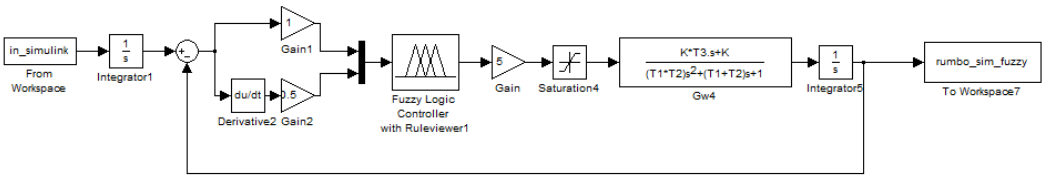

Figura 4.11. Sistema con controlador difuso

5. Simulación para diferentes trayectorias

El presente capítulo muestra los resultados obtenidos mediante la simulación del control del barco al aplicar los diferentes tipos de controladores desarrollados.

En primer lugar se realiza una descripción del sistema completa, analizando el funcionamiento del programa generado en Matlab, los diagramas de bloques de Simulink, etc.

Finalmente se muestran los resultados de las simulaciones con los diferentes tipos de controladores y para diversos tipos de trayectorias y cambios de rumbo.

5.1. Esquema del programa

La siguiente figura muestra de forma resumida el esquema del código desarrollado para la simulación del sistema. El programa calculará de forma paralela los resultados al aplicar los diferentes tipos de controladores diseñados, permitiendo comparar el funcionamiento de cada uno de ellos:

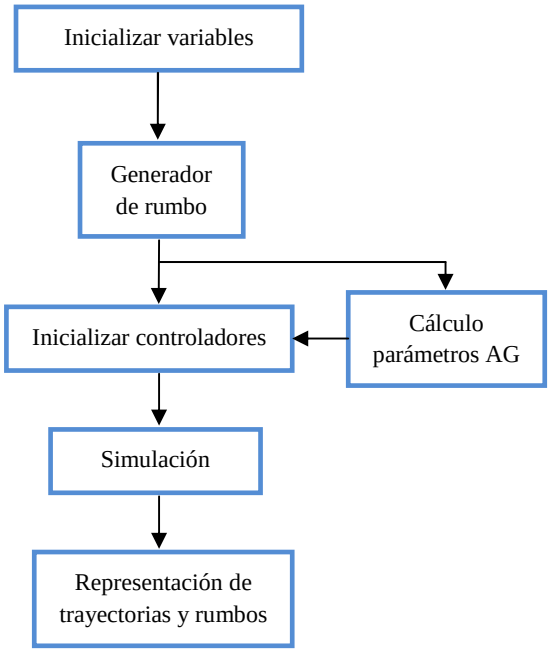

Figura 5.1. Esquema del programa

El desarrollo se ha realizado íntegramente en Matlab, haciendo uso de Simulink para la simulación del funcionamiento de los diferentes tipos de controladores.

Básicamente la programación realizada se puede separar en tres bloques principales que pasan a describirse brevemente a continuación:

- Inicialización de variables y controladores
- Simulación
- Procesamiento y cálculo de resultados

5.1.1. Inicialización de variables y controladores

La función de este bloque es definir todas las variables necesarias para el correcto funcionamiento del sistema. De forma esquemática, el funcionamiento de esta parte del código se puede resumir de la siguiente forma:

- En primer lugar se definen algunas variables del sistema que se mantendrán fijas a lo largo de la ejecución del programa, como por ejemplo: las constantes del modelo del

barco, los parámetros K_p, K_i y K_d del controlador PID clásico, la velocidad del barco, su posición y rumbo inicial, etc.

- A continuación se realiza una llamada a la función generadora del rumbo deseado. Dependiendo del tipo de trayectoria que se desea realizar, esta función calculará el vector de la velocidad angular deseada, que será considerado como la entrada del sistema a simular en Simulink. Aquí se podrá escoger entre distintos tipos de trayectorias (circulares, en zig-zag, con un cambio de rumbo determinado en un instante dado, etc.). Esa función también permite definir el tipo y duración de los cambios de rumbo, determinando así si los giros serán suaves, abruptos, etc.).

  Nota: Dependiendo del tipo de trayectoria que se determine en esta parte, la función devolverá un valor que servirá para definir el tipo de función objetivo a aplicar en el algoritmo genético. De este modo, se tratará de optimizar los valores del controlador calculados mediante AG para un determinado cambio de rumbo.
- Se realiza una llamada a la función del algoritmo genético, para que éste calcule los parámetros K_p, K_i y K_d del controlador PID optimizado con AG. La función del algoritmo genético ya ha quedado descrita en el capítulo anterior, así como las variables configurables que dispone. Estas variables deberán ser definidas antes de lanzar el programa principal, y sus valores óptimos se analizan en el Anexo I.
- A continuación, el programa principal carga el controlador difuso, inicializa los tres tipos de controladores y realiza una llamada al diagrama de bloques de Simulink.

5.1.2. Simulación

La simulación del funcionamiento del sistema se realiza en Simulink. El paso de variables tanto de Matlab a Simulink, como de Simulink a Matlab, se realiza a través del Workspace de Matlab.

La entrada del sistema es el vector de la velocidad angular calculado en la función generadora de rumbo. Por lo tanto, en primer lugar se aplica una integración para obtener el rumbo deseado $\psi_d(t)$. La siguiente figura muestra el diagrama de bloques realizado en Simulink, y en él se puede observar como también se aplica una integración a la salida del modelo de barco, obteniendo así el rumbo real del mismo:

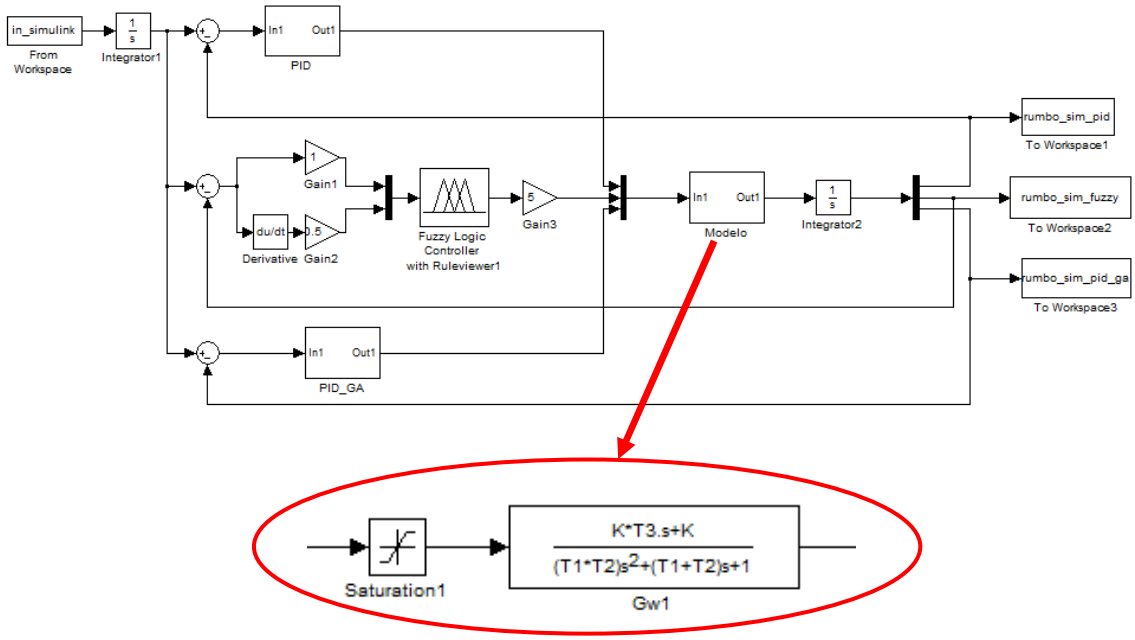

Figura 5.2. Diagrama Simulink

En la figura 5.2 se muestra que el bloque “modelo” del diagrama incluye la propia función de transferencia del modelo del barco, así como un elemento de saturación. La saturación se aplica

a la salida de los controladores para evitar acciones demasiado agresivas sobre el timón del barco, que pudieran sobrepasar los límites físicos del propio actuador.

Cada controlador actúa sobre el modelo del barco G_w y sus salidas son pasadas de nuevo a Matlab mediante el Workspace para su posterior procesamiento.

5.1.3. Procesamiento y cálculo de resultados

Una vez finalizada la simulación, se procesan los datos generados para calcular y representar tanto las diferencias entre los rumbos deseados y reales, como las trayectorias realizadas en el plano XY.

El programa permite generar gráficas en las que se pueden mostrar conjuntamente los resultados de los distintos controladores, para comparar así el funcionamiento de cada uno de ellos. Del mismo modo, también se generan representaciones del cuerpo sólido del barco para analizar de forma más visual el efecto de los virajes, la aparición de sobrepicos en la respuesta, etc.

5.2. Análisis de resultados de simulación

A continuación se van a analizar los resultados obtenidos al aplicar diferentes tipos de trayectorias sobre los tres controladores desarrollados.

5.2.1. Análisis de respuestas

Como ya se ha comentado, en función del tipo de controlador que se seleccione, la respuesta ante distintos tipos de virajes (más o menos abruptos, con mayor o menor prolongación en el tiempo, etc.) no será la misma. A continuación se van a mostrar los resultados obtenidos al aplicar distintos tipos de simulaciones, y se podrá analizar el efecto de los controladores sobre los aspectos de interés que afectan al comportamiento del barco.

Los principales puntos a tener en cuenta serán los siguientes:

- Error en el rumbo real respecto al deseado. Es el aspecto principal a analizar en estas pruebas, ya que el objetivo del controlador es tratar de mantener el rumbo deseado.
- Error/diferencia de la trayectoria real realizada respecto a la deseada. En relación al punto anterior, si no se consigue mantener el rumbo deseado en todo momento, es de esperar que esas variaciones provoquen que la trayectoria del barco no sea la esperada a lo largo de su recorrido.
- Sobrepico de la respuesta y el tiempo que necesita para alcanzar un valor estable. Este aspecto es crítico a la hora de determinar la forma en la que el barco realizará el viraje.
- Tiempo requerido para cubrir el mismo recorrido. Este aspecto está muy relacionado con el punto anterior, ya que un exceso de oscilación y/o sobrepico, provocarán que el barco recorra una mayor distancia hasta conseguir el rumbo estable, y por lo tanto necesite de más tiempo para alcanzar el mismo punto de destino.

Antes de comenzar el análisis de los tres tipos de controladores conjuntamente, se va a estudiar el efecto de la función objetivo del algoritmo genético a la hora de calcular sus parámetros optimizados. Esta parte se ha sacado del anexo correspondiente a la optimización del AG, ya que se considera que muestra claramente las diferencias en la respuesta del sistema al aplicar distintos tipos de cambios de rumbo, así como la influencia del tipo de controlador aplicado.

5.2.2. Respuesta en controlador AG en función del tipo de viraje

El controlador del barco debe estar preparado para hacer frente a diferentes tipos de cambios de rumbo. Generalmente, el barco realizará cambios de rumbos suaves que tratarán de optimizar de la mejor manera el recorrido a realizar (evitando el malestar que podrían causar en los tripulantes virajes demasiado bruscos, tratando de optimizar los recorridos, etc.).

Tal y como se podrá comprobar en el siguiente apartado al analizar los resultados de los diferentes controladores, tanto el controlador PID clásico como el controlador difuso darán una respuesta fija en función del cambio de rumbo que se exija. De este modo, si uno de estos controladores ofrece peores prestaciones ante virajes bruscos, y mejores ante cambios de rumbo suaves, su respuesta estará totalmente influenciada por la característica de la trayectoria que se vaya a realizar,

Sin embargo, tal y como se ha descrito anteriormente, el controlador AG se ha diseñado de manera que sea capaz de amoldar su resultado en función del tipo de cambio de rumbo que se desee aplicar al barco. Esto se consigue modificando la función objetivo que determina cómo de bueno es un individuo concreto (o dicho de otra forma, el error que generaría la respuesta del sistema en unas condiciones dadas). De este modo, el algoritmo permite cambiar el tipo de función objetivo, adaptándose al tipo de trayectoria definida para cada simulación. Si la trayectoria requiere cambios del rumbo puntuales o continuos, suaves o bruscos, etc., la función objetivo a aplicar será diferente, y por lo tanto, también lo serán los parámetros PID que entregue el algoritmo como solución final.

Para estas pruebas se ha optado por comparar el resultado de la simulación para tres tipos de controladores que se obtendrían al aplicar distintos tipos de funciones objetivo en el algoritmo genético: una respuesta optimizada para cambios de rumbo rápidos, otra para cambios de rumbo progresivos, y otra para cambios de rumbo suaves.

Se ha ejecutado el algoritmo con estas condiciones, y los parámetros obtenidos han sido los siguientes:

- Controlador 1: K_p=9.9988, K_i=7.1733, K_d=9.9828
- Controlador 2: K_p=9.9896, K_i=4.5362, K_d=3.7546
- Controlador 3: K_p=9.9012, K_i=0.0223, K_d=9.5564

Para la obtención de estos parámetros se ha ejecutado el algoritmo utilizando las condiciones que se consideran óptimas en cuanto a la relación entre resultados y tiempo de cómputo (las definidas en el apartado 4.2). Por esto, en caso de repetir la ejecución del AG bajo esas mismas condiciones, se puede considerar que los valores obtenidos, aunque no iguales, serán parecidos (y por consiguiente, también lo serán los resultados de las simulaciones).

Realizando la simulación del control del barco con estos tres tipos de controladores, en primer lugar se va a analizar la diferencia en la respuesta a la hora de aplicar un cambio de rumbo suave. Para esto se aplicará una trayectoria recta, que sufre un cambio de rumbo suave de +10º a los 20 segundos de recorrido.

Los resultados obtenidos bajo estas condiciones de simulación son los que se muestran a continuación (cada figura muestra dos gráficas, donde la de la derecha representa el rumbo deseado y el obtenido al aplicar el controlador, y la de la izquierda muestra el recorrido deseado y el que realmente realiza el barco en la simulación una vez se aplica el controlador):

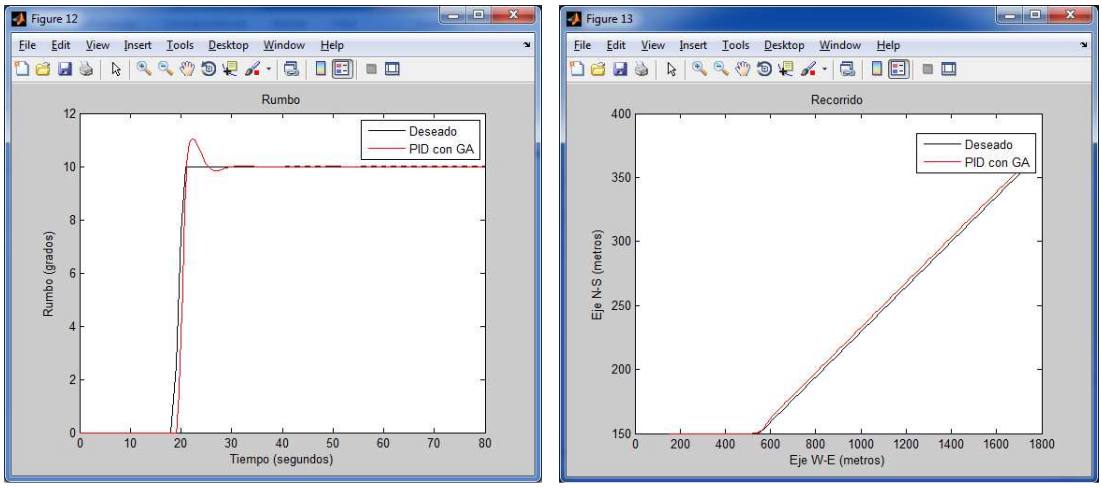

Figura 5.3. Controlador 1 para cambio de rumbo suave

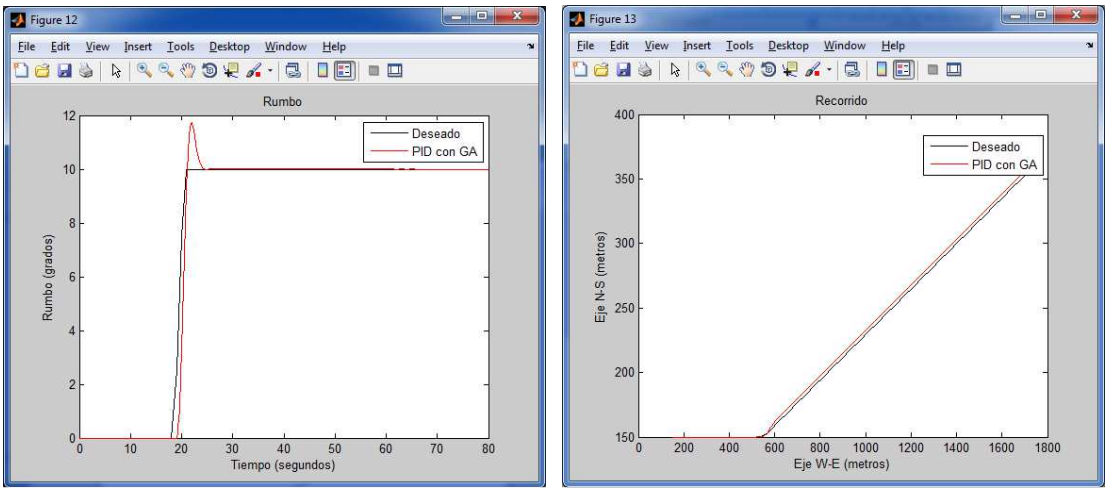

Figura 5.4. Controlador 2 para cambio de rumbo suave

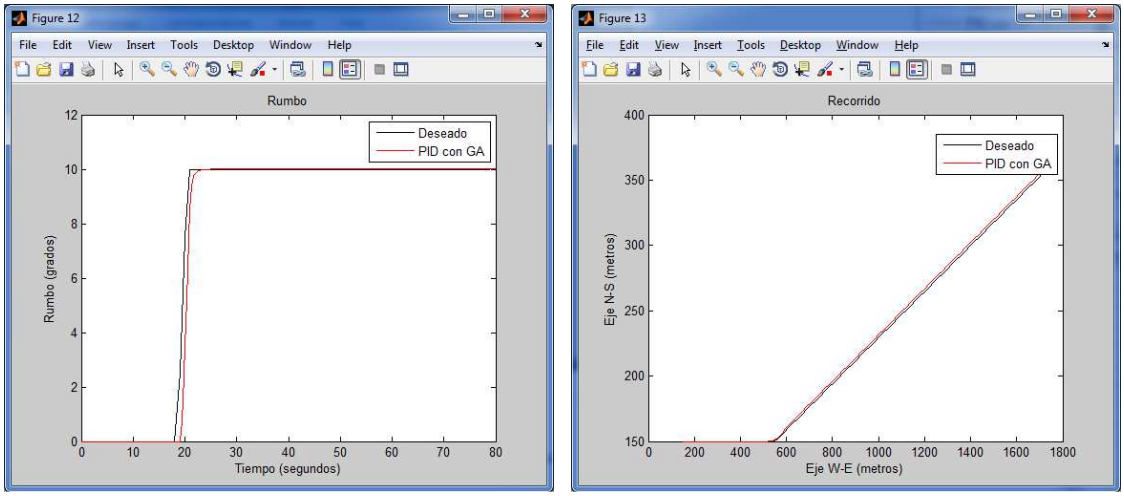

Figura 5.5. Controlador 3 para cambio de rumbo suave

Se ha de tener en cuenta que los gráficos de recorrido no están con sus ejes X e Y a la misma escala. Por ello en las gráficas de la derecha el cambio de rumbo parece mayor de lo que realmente es (+10º). Esto está así para poder apreciar mejor el efecto en el recorrido debido a los cambios de rumbo.

En las anteriores gráficas se puede ver la diferencia en la respuesta del controlador dependiendo de la función objetivo que haya utilizado el algoritmo genético. Cada tipo de controlador proporcionará diversas ventajas y desventajas, por lo que en función de lo que sea más beneficioso para aplicación real, será recomendable optar por un tipo de optimización u otro. Esto se realizará a la hora de ejecutar el algoritmo genético, permitiendo que se determinen los valores con los que configurar el controlador (en este caso la función objetivo) dependiendo de las características de la trayectoria seleccionada.

En el caso anterior, las diferencias de un controlador a otro no son excesivamente visibles. Al tratarse de un cambio de rumbo relativamente suave, los tres controladores se adaptan rápido y prácticamente no se aprecia ninguna diferencia en el recorrido que se obtiene al realizar las simulaciones. Es cierto que los controladores 1 y 2 generan un sobrepico y muestran un periodo de oscilación hasta alcanzar un valor estable; sin embargo, como se puede comprobar, el sobrepico apenas alcanza los 2º, y su efecto se podría considerar prácticamente despreciable a la hora de aplicar la simulación al cálculo del recorrido realizado.

El programa realizado genera unas gráficas que muestran el estado (orientación, posición) a lo largo de su trayectoria del solido rígido que representa al barco. Esta representación servirá

de ayuda para ver de forma visual el efecto de esos sobrepicos y oscilaciones en forma de cambios de rumbo sobre el timón:

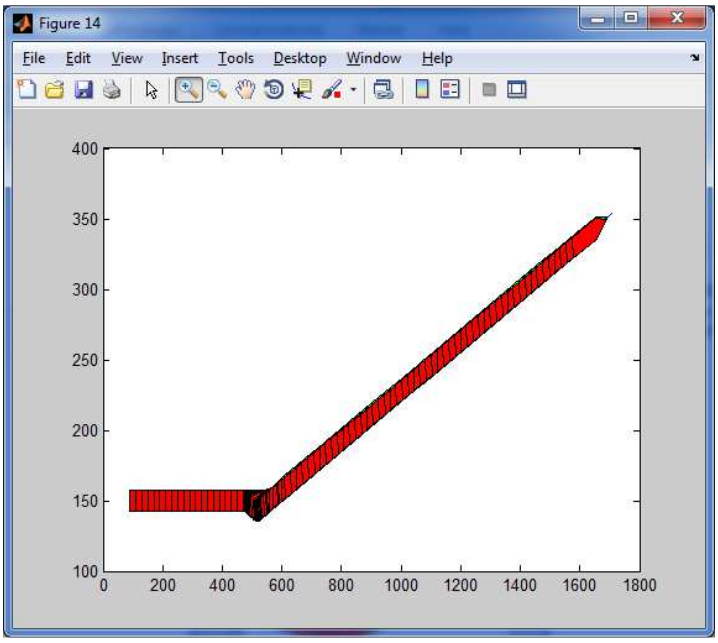

Figura 5.6. Representación del barco con Controlador 1 (verde) y Controlador 3 (rojo)

En la imagen anterior apenas se aprecia la diferencia entre el resultado de ambos controladores, ya que las dos representaciones quedan prácticamente superpuestas. Al tratarse de un cambio de rumbo muy suave, el efecto del sobrepico y las oscilaciones apenas se perciben en la trayectoria final que sigue el barco.

Aplicando un zoom a la zona donde se realiza el viraje, se puede observar la pequeña diferencia existente entre un controlador y el otro. Por lo tanto, se puede concluir que apenas se perciben las diferencias mostradas en las gráficas de la respuesta del sistema:

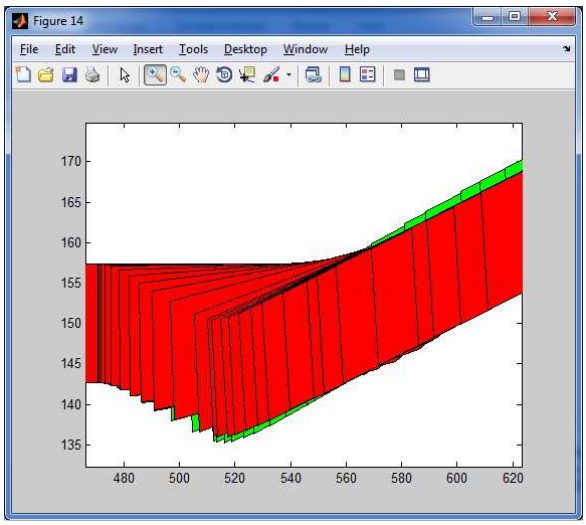

Figura 5.7. Zoom de la zona de cambio de rumbo

Es por ello que, para analizar estos aspectos de una forma más clara, a continuación se repetirá la misma prueba aplicando un cambio de rumbo más brusco. En este caso, se seguirá una trayectoria similar, pero en lugar de aplicar un cambio de rumbo de +10º, al controlador le llegará la orden de virar +105º (en el mismo periodo de tiempo).

Los resultados que se obtienen con estas nuevas condiciones son los que se muestran a continuación:

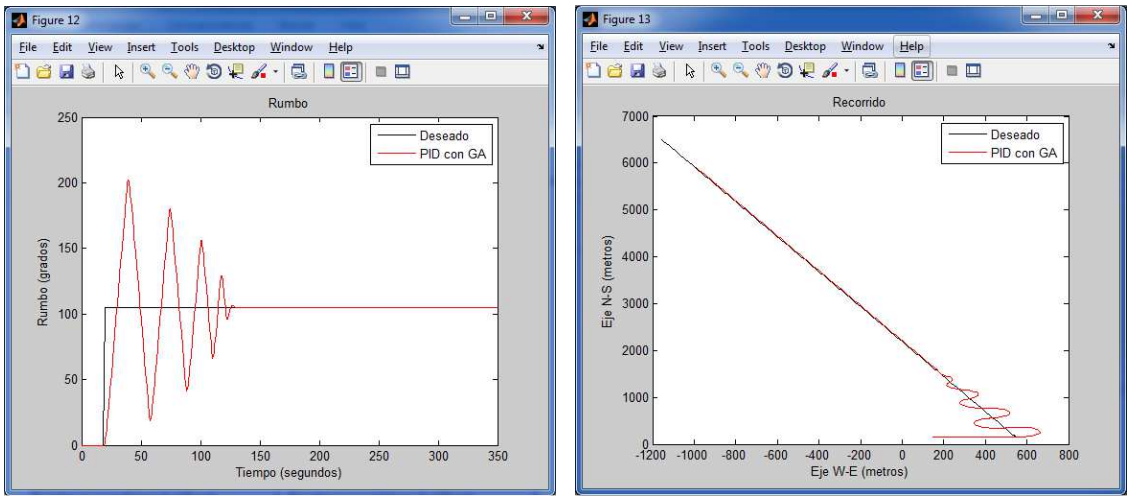

Figura 5.8. Controlador 1 para cambio de rumbo brusco

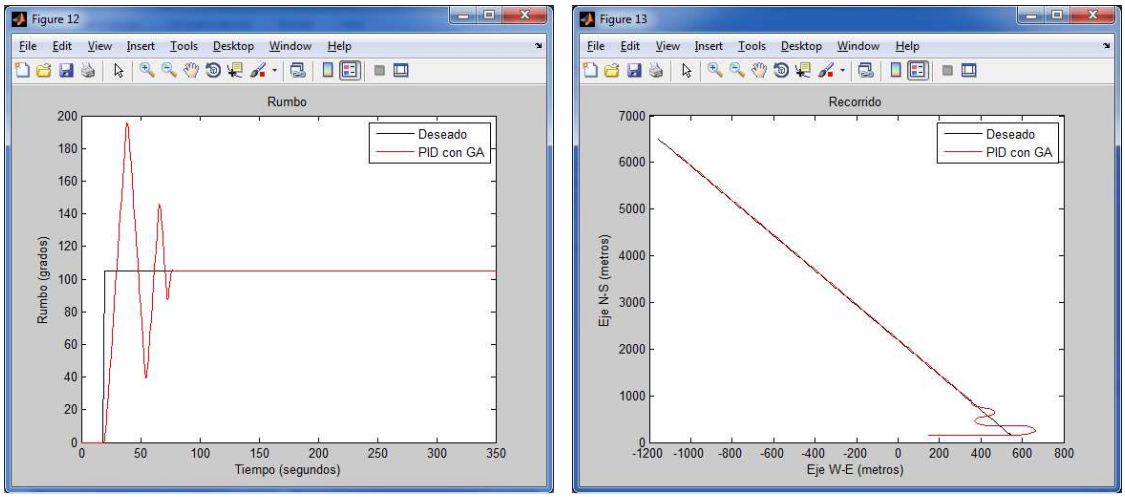

Figura 5.9. Controlador 2 para cambio de rumbo brusco

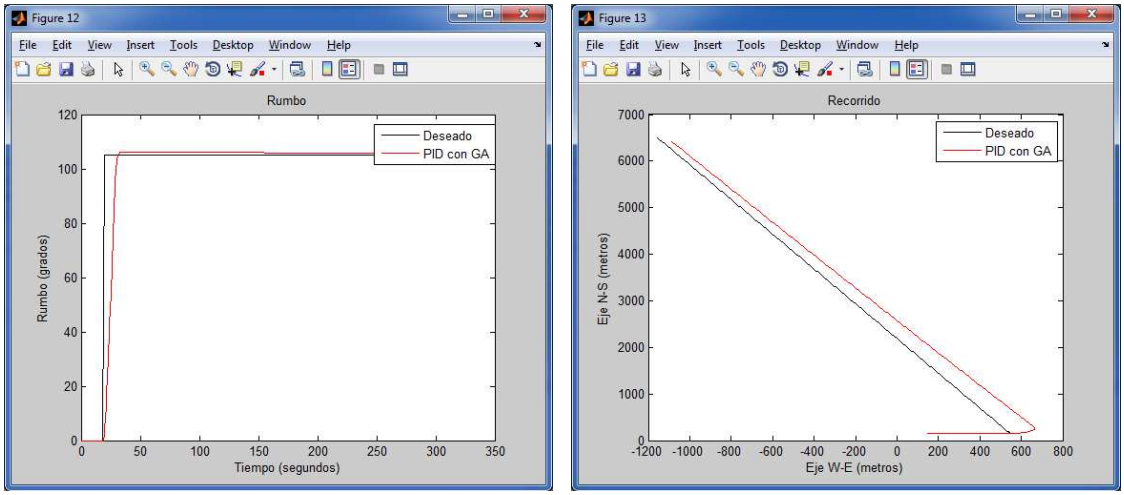

Figura 5.10. Controlador 3 para cambio de rumbo brusco

En este caso se aprecian claramente las diferencias en el resultado al aplicar los diferentes tipos de controladores:

- El sistema con el Controlador 3 ofrece una respuesta mucho más suave. No aparecen oscilaciones y alcanza un rumbo estable más rápido que en los otros dos casos. Sin embargo, en la respuesta del rumbo de la figura 5.10 existe una pequeña desviación respecto al rumbo deseado ya que le cuesta llegar al régimen estacionario. Esto provoca que al comparar la diferencia entre el recorrido deseado y el realmente realizado, se observa una desviación considerable que tiende a reducirse de forma muy lenta. De hecho, como se muestra en la siguiente imagen, el Controlador 3 no alcanza el rumbo deseado en régimen estacionario hasta pasados los 730 segundos:

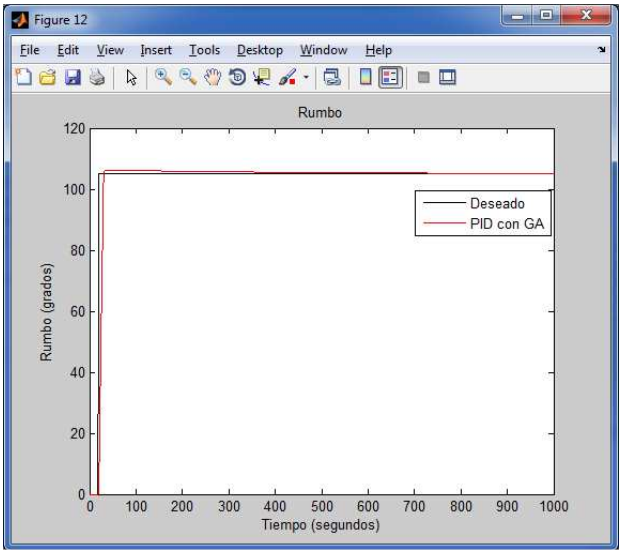

Figura 5.11. Régimen estacionario en respuesta del Controlador 3

- Al analizar los resultados obtenidos con los controladores 1 y 2, se observa que aparece una importante componente oscilatoria en la respuesta, así como un sobrepico muy elevado. En ambos casos, una vez que se alcanza la estabilidad el rumbo se mantiene constante en el valor deseado (con una pequeña desviación respecto a la ruta deseada), sin embargo, el recorrido realizado durante la fase oscilatoria provoca que el barco recorra una mayor distancia, y por lo tanto necesite de un mayor tiempo para alcanzar el punto de destino.

En función de los resultados, se puede considerar que para cambios de rumbo suaves los distintos tipos de controladores analizados ofrecen una solución aceptable y similar en todos los casos. Es por ello que en estos casos no tiene una influencia excesiva el hecho de fijar la función objetivo del AG que mejor resultado pueda dar.

Sin embargo, tal y como se ha podido ver en las últimas representaciones, el problema surge cuando aparecen cambios de rumbo bruscos. En el caso de cambios de rumbo bruscos, se considera que la mejor opción es aplicar controladores similares al tipo 3. La respuesta del sistema será mucho más suave y desaparecen los problemas derivados del sobrepico y las oscilaciones, que provocan que el barco vaya girando en un sentido y otro hasta alcanzar la estabilidad. Es cierto que la desventaja que aparece en forma de desviación del recorrido real respecto a la trayectoria deseada puede ser mayor, sin embargo, el objetivo principal es tratar de mantener el rumbo correcto (cosa que se consigue), y se consideraría factible el aplicar cualquier tipo de corrección sobre el vector del rumbo deseado (por ejemplo, mediante indicaciones provenientes del sistema de navegación al detectar una desviación en las coordenadas de posición deseadas) que permitiese corregir dichas desviaciones.

A continuación se mostrarán los diferentes resultados obtenidos al aplicar diversos tipos de trayectorias sobre los controladores desarrollados:

5.2.3. Resultados

A continuación se van a analizar los resultados obtenidos con los tres tipos de controladores (PID clásico, PID optimizado con AG y controlador difuso).

Como se ha visto en el apartado anterior, para el controlador AG el algoritmo permite aplicar diferentes tipos de optimizaciones en función del tipo de resultado que se pretende obtener. Para las posteriores simulaciones se va a tratar de aplicar la opción de controlador que minimice los sobrepicos y los periodos largos de oscilación que conllevan a necesitar de un mayor tiempo para alcanzar un rumbo estable.

5.2.3.1. Cambios de rumbo sobre trayectorias rectas

A continuación se van a analizar los resultados obtenidos al aplicar sobre una trayectoria recta un cambio de rumbo puntual de +50º. Se trata de un término medio entre las dos simulaciones probadas en el anterior apartado (para un cambio de rumbo suave de +10º y uno brusco de +105º).

Este cambio de rumbo se ha definido de forma que se le aplica un cambio a la velocidad angular deseada de 10 grados/segundo durante un periodo de 5 segundos de duración. De este modo, la representación gráfica del vector de la velocidad angular deseada sería el que se muestra a continuación:

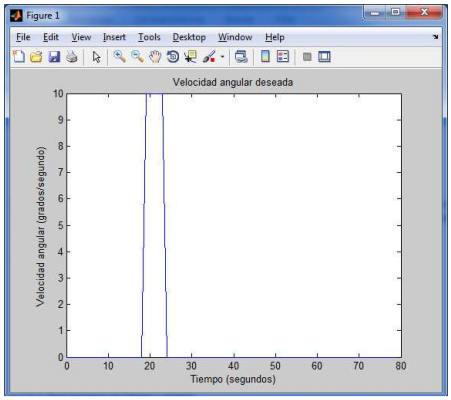

Figura 5.12. Velocidad angular deseada

Aplicando esta trayectoria sobre los 3 tipos de controladores, se obtienen las siguientes respuestas al sistema:

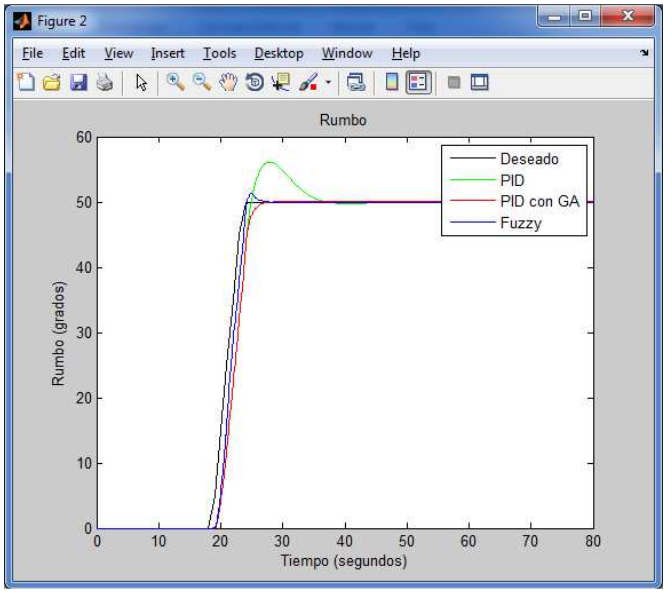

Figura 5.13. Respuesta del sistema

Analizando este resultado, se contempla que el controlador PID optimizado mediante AG y el controlador difuso ofrecen una respuesta más adecuada que el PID clásico (con menor sobrepico y oscilaciones). El controlador difuso presenta un pequeño sobrepico, pero como contrapartida, alcanza el valor del rumbo deseado (+50º) más rápido que cualquiera de los otros controladores.

Para analizar el efecto en el recorrido real del barco, y compararlo con el deseado, a continuación se muestra una representación de los recorridos que realizaría el barco al aplicarle los tres tipos de controladores. La imagen muestra el zoom tanto de la zona de viraje, como de la zona final del recorrido, para analizar más claramente las diferencias respecto a la trayectoria deseada:

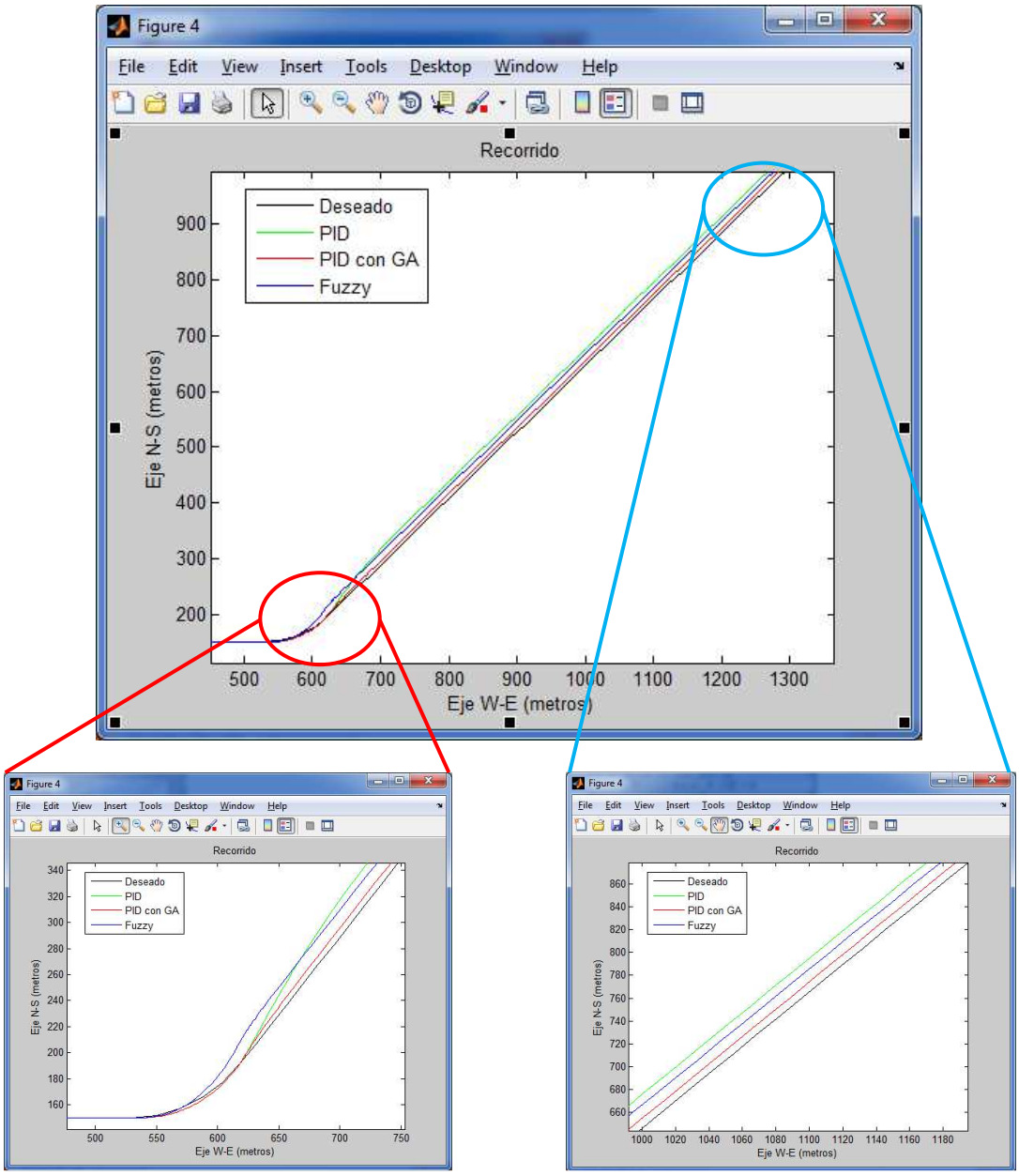

Figura 5.14. Recorrido realizado por el barco y zooms de las zonas de viraje y final

De esta última representación se desprende que los tres consiguen finalmente un rumbo constante igual al deseado. Sin embargo, es el controlador optimizado mediante AG el que menos se desvía de la trayectoria deseada (seguido por el controlador difuso). Del mismo modo, el efecto del sobrepico, aunque pequeño, prácticamente solo aparece en el controlador PID clásico.

5.2.3.2. Trayectorias circulares

A continuación se van a analizar las respuestas de los controladores al aplicarlos a trayectorias del tipo circular.

En primer lugar se dispone de una trayectoria circular que dibuja una circunferencia grande, aplicando una velocidad angular deseada de 1 grado/segundo. La siguiente imagen muestra que al tratarse de un cambio de rumbo muy suave respecto al recorrido total, la diferencia entre la trayectoria obtenida para los diferentes controladores apenas es apreciable:

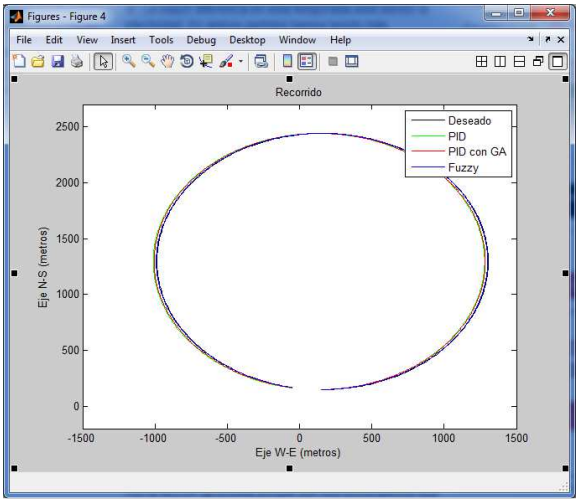

Figura 5.15. Trayectoria circular (ω_d=1 grados/seg)

Aplicando un zoom, se puede observar que el controlador difuso es que consigue una mayor similitud con la trayectoria deseada:

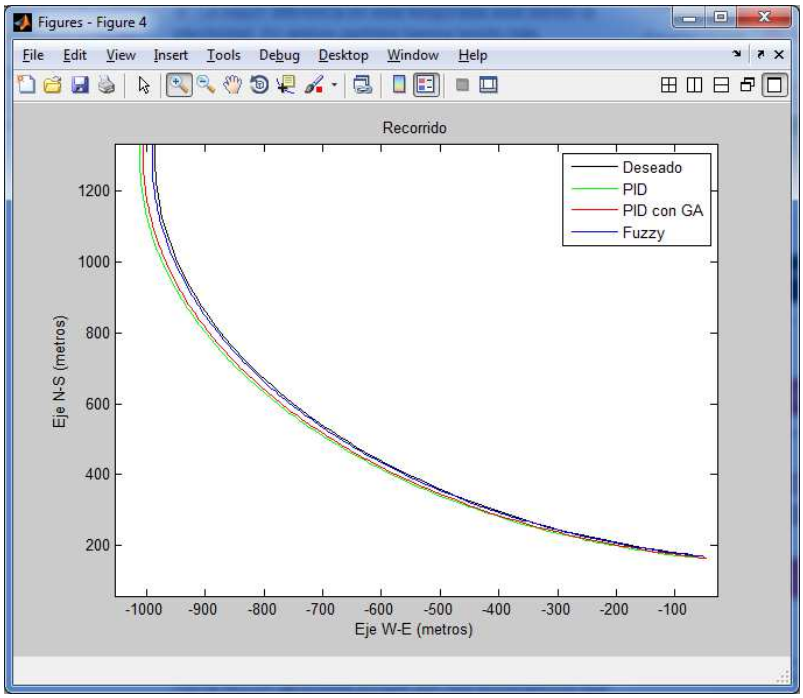

Figura 5.16. Zoom de la trayectoria circular (ω_d=1 grados/seg)

Si se aumenta el valor de la velocidad angular a 4 grados/segundo, se consigue una circunferencia de menor diámetro, y se puede apreciar más claramente la diferencia de una

trayectoria a otra. Esto se debe a que a los controladores les cuesta más mantener el rumbo deseado ante virajes con mayor velocidad angular.

En este caso, con los dos controladores que utilizan técnicas de control inteligente se obtiene un resultado similar (algo mejor que el obtenido mediante el controlador PID clásico):

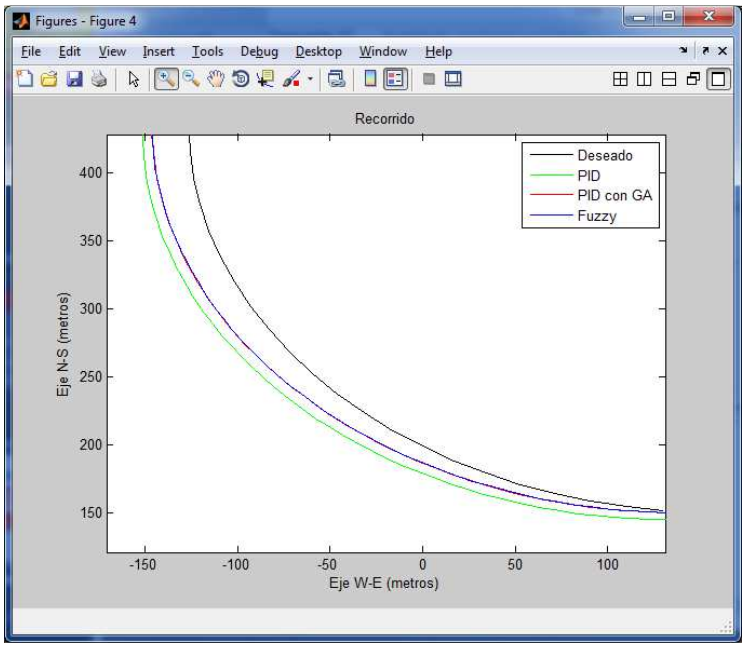

Figura 5.17. Trayectoria circular (ω_d=4 grados/seg)

5.2.3.3. Trayectorias en zig-zag

Para establecer la trayectoria en zig-zag que se va a analizar en este apartado, se ha generado el siguiente vector de velocidad angular deseada:

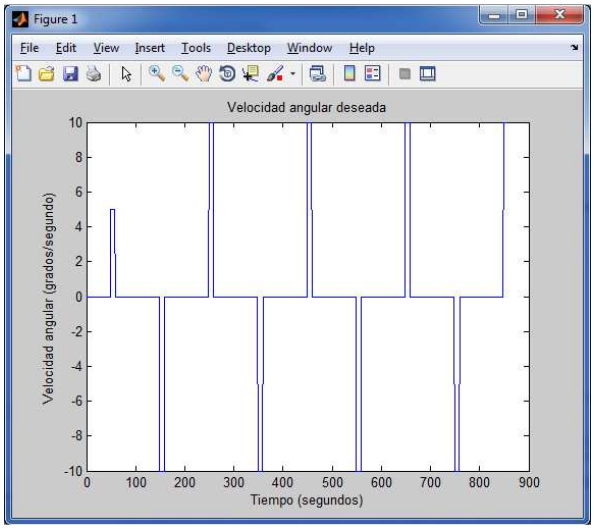

Figura 5.18. Velocidad angular deseada para trayectoria en zig-zag

Esto permite realizar una trayectoria en zig-zag, que irá cambiando el rumbo de +50º a -50º. Para esto se deben aplicar cambios de rumbo de 100º, quedando definido de la siguiente forma:

- Sobre una trayectoria recta (ω_d=0 grados/seg), a los 50 segundos se aplica una velocidad angular de 5 grados/segundo durante un periodo de 10 segundos. Esto hace que el barco comience a virar hasta alcanzar el rumbo de +50º.

- Se mantiene el rumbo constante, y a los 100 segundos se aplica otro cambio de rumbo en sentido contrario. Este nuevo cambio de rumbo también tendrá una duración de 10 segundos, pero aplicando el doble de velocidad angular (ω_d=10 grados/seg). De este modo, se consigue que el barco cambie su rumbo en 100º, pasando del +50º al -50º.
- Se sigue repitiendo el proceso de forma periódica, modificando el sentido de giro en positivo a negativo.

La siguiente imagen muestra el rumbo obtenido al aplicar esta trayectoria a los diferentes controladores. Se observa que en el primer cambio de rumbo (en t=50 segundos) las oscilaciones de la respuesta son mucho más suaves. Esto se debe a que la velocidad angular aplicada en ese primer viraje es inferior (5 grados/seg respecto a los 10grados/seg que se aplican en los siguientes cambios de rumbo):

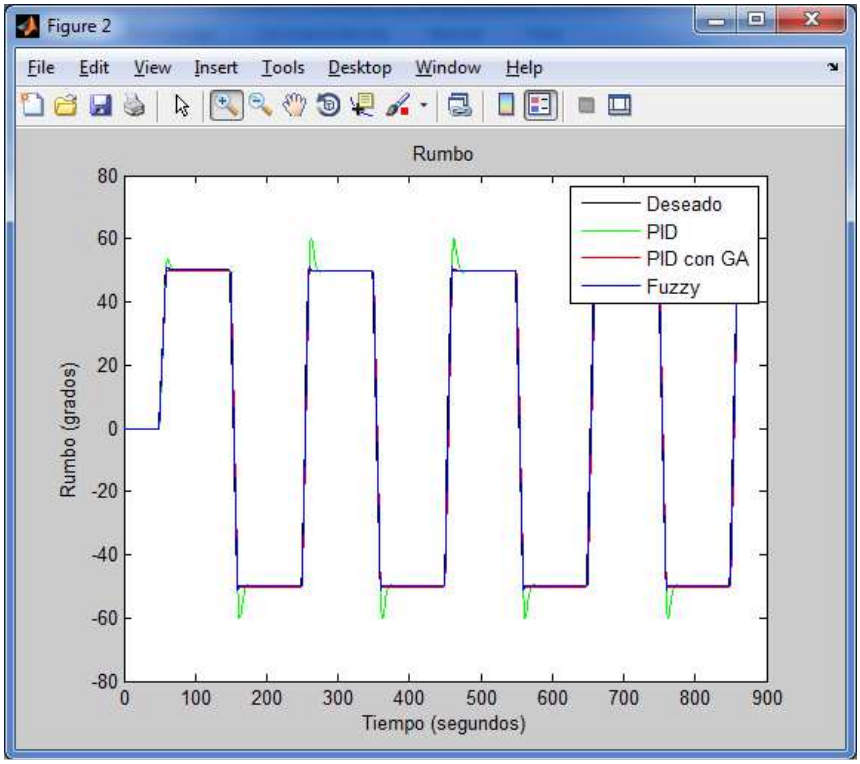

Figura 5.19. Rumbo para trayectoria en zig-zag

Aplicando un zoom a cualquiera de las zonas de viraje, se pueden ver con más claridad las diferencias de la respuesta del sistema en función del controlador utilizado:

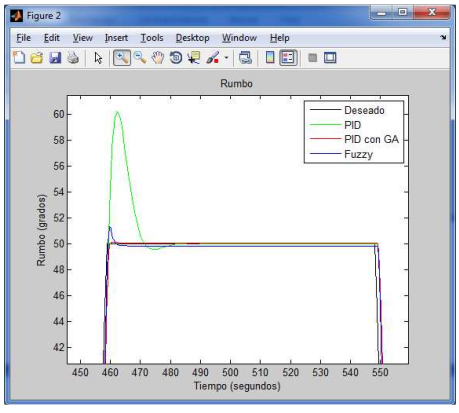

Figura 5.20. Zoom para rumbo con trayectoria en zig-zag

Obteniendo las gráficas del recorrido realizado, se observa que a medida que aumenta el tiempo, las diferencias respecto al recorrido deseado se hacen más notables. Esto principalmente ocurre con el controlador PID clásico, para el cual, al final del recorrido prácticamente se ve una desviación de 200 metros respecto al recorrido deseado:

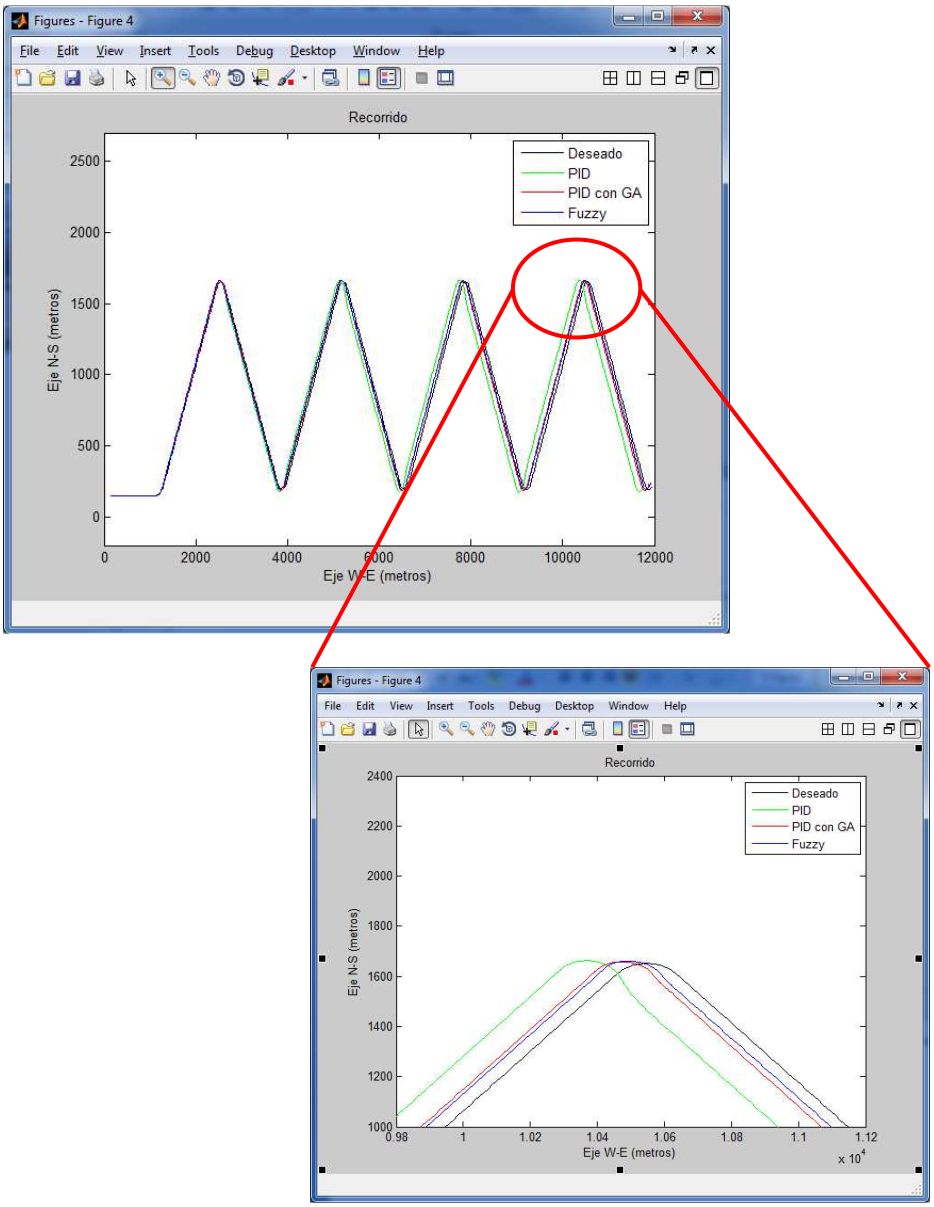

Figura 5.21. Recorrido para trayectoria en zig-zag

6. Control adaptativo con Redes Neuronales

El presente capítulo muestra el desarrollo realizado de una funcionalidad del sistema para tratar de contrarrestar posibles efectos negativos procedentes de las perturbaciones que pueden aparecer en un entorno real: oleaje, viento, etc.

Para esto se ha realizado el diseño de una red neuronal haciendo uso de la toolbox de Matlab, que tendrá como objetivo calcular un vector de corrección aplicable al rumbo deseado, de manera que se minimicen de la mejor forma posible los efectos de ese tipo de perturbaciones sobre el rumbo del barco.

6.1. Efecto de las perturbaciones

Todos los resultados que se han analizado hasta el momento han sido fruto de simulaciones realizadas en un entorno ideal. Esto no es aplicable a una situación real, donde existen perturbaciones externas que afectan al barco a lo largo de su recorrido, como pueden ser el viento, las mareas, etc.

Para analizar cómo responde el sistema y los distintos controladores ante este tipo de perturbaciones, se ha incluido un bloque diseñado para simular el efecto que tendrían este tipo de agentes externos sobre el barco.

Al no disponer de ningún modelo aplicable, se ha generado una función a la que se le determina el ángulo de ataque respecto al barco de una perturbación ideal (ya sea viento, oleaje, marea, etc.), así como su intensidad. A partir de la relación entre el ángulo de ataque de la perturbación y el rumbo que sigue el barco, la función generará cierto efecto negativo proporcional a la intensidad de la perturbación, tanto en la velocidad angular del barco como en su velocidad de desplazamiento.

El siguiente diagrama de bloques muestra el funcionamiento de este sistema:

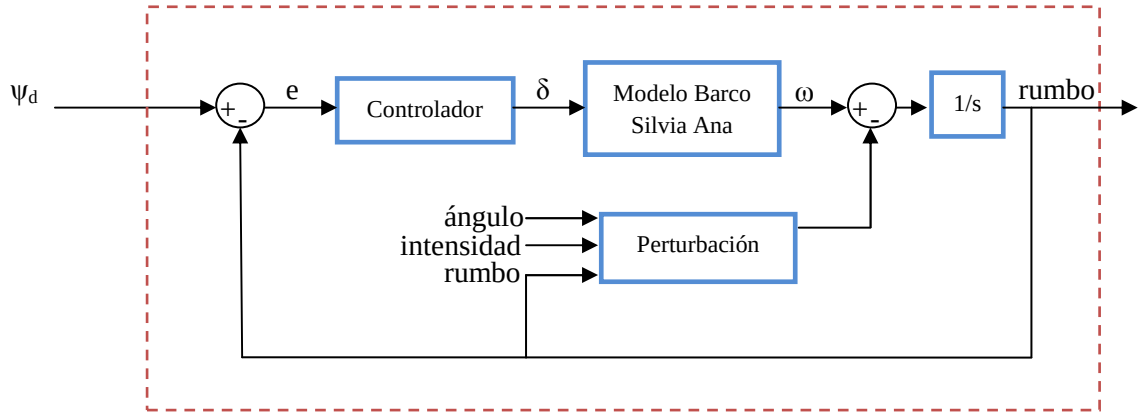

Figura 6.1. Diagrama de bloques incluyendo agentes externos

En el esquema se observa que el bloque encargado de calcular el efecto de la perturbación tomará como variables externas el ángulo e intensidad de la misma, las cuales serán definidas al inicio de cada simulación. Además, también tendrá como entrada el rumbo del barco, para calcular así la relación entre el ángulo de la dirección de la perturbación y del barco.

Tal y como se ha comentado, al no disponer de un modelo concreto de los distintos tipos de perturbaciones posibles que pueden afectar al barco, se ha determinado desarrollar una función genérica, que únicamente dependerá de la intensidad que se le defina a la perturbación, así como de su ángulo de ataque respecto al rumbo que lleva el barco. De este modo, la salida del bloque generador de la perturbación afectará a la velocidad angular del barco mediante una relación del tipo sinusoidal entre el ángulo de ataque de la perturbación y el rumbo del mismo. Por otro lado, el efecto en la velocidad de desplazamiento tendrá una relación del tipo cosenoidal.

Para aclarar este punto, a continuación se muestran unos ejemplos simplificados del efecto que producirían las perturbaciones en el cálculo del rumbo y el recorrido del barco, con el fin de aclarar la relación recientemente expuesta:

6.1.1. Perturbación perpendicular

La siguiente imagen muestra un ejemplo en el que una perturbación (ya sea debida a una marea, viento, u oleaje) ataca al barco con un ángulo de 90º respecto al rumbo que sigue el mismo. En este caso concreto, el barco se desplaza con rumbo ESTE, mientras que la perturbación ataca desde el SUR, dirección NORTE.

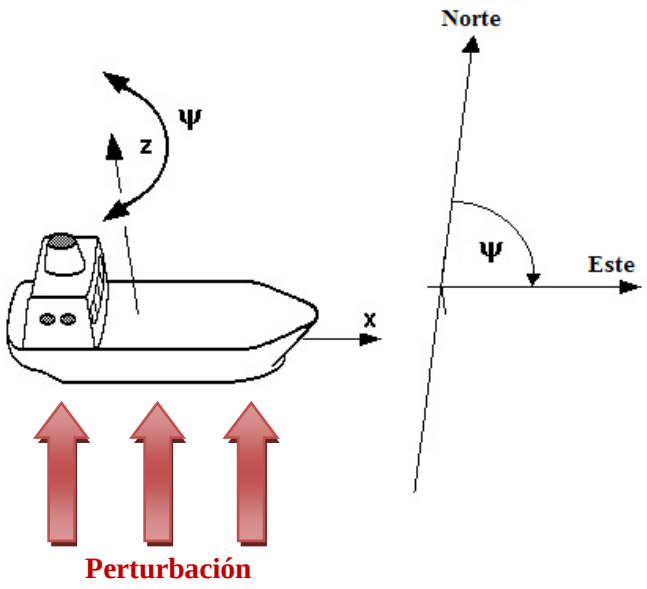

Figura 6.2. Efecto de perturbación perpendicular

En este caso, si el barco comenzase a girar dirección NORTE, el efecto de la perturbación provocaría que el barco virase con una mayor velocidad angular, haciendo que el rumbo cambiase más de lo esperado. El efecto contrario ocurriría si el barco intentase virar dirección SUR, ya que tendría que hacer frente al efecto de la perturbación (siendo la velocidad angular menor de la esperada).

Así mismo, si el barco tratase de mantener el rumbo constante hacia el ESTE, el efecto de la perturbación tenderá a hacer girar el barco hacia el NORTE en lugar de mantenerse fijo (aunque como se podrá observar, esta tendencia se llega a minimizar debido al efecto de la realimentación del controlador).

El efecto sobre la velocidad de desplazamiento que provoca en el barco una perturbación que ataca de forma perpendicular respecto a su rumbo es mínimo, por lo que se puede considerar que la velocidad se mantiene prácticamente constante.

6.1.2. Perturbación paralela

No ocurre así en el caso de una perturbación que ataca de forma paralela respecto al rumbo de desplazamiento del barco.

En el caso que se muestra en la figura 6.3, la perturbación no tiene ningún efecto sobre el cambio de rumbo del barco (no afecta a la velocidad angular), sin embargo, se intuye fácilmente que la velocidad de desplazamiento del barco aumentará debido al empuje que pueda ejercer la perturbación.

Si por el contrario, la perturbación fuese por ejemplo viento de levante (viento procedente del ESTE), el barco tendería a desplazarse más despacio debido a que debe contrarrestar el efecto de la perturbación (siempre considerando que los actuadores encargados de controlar la velocidad: hélices, turbinas, etc., se mantienen constantes).

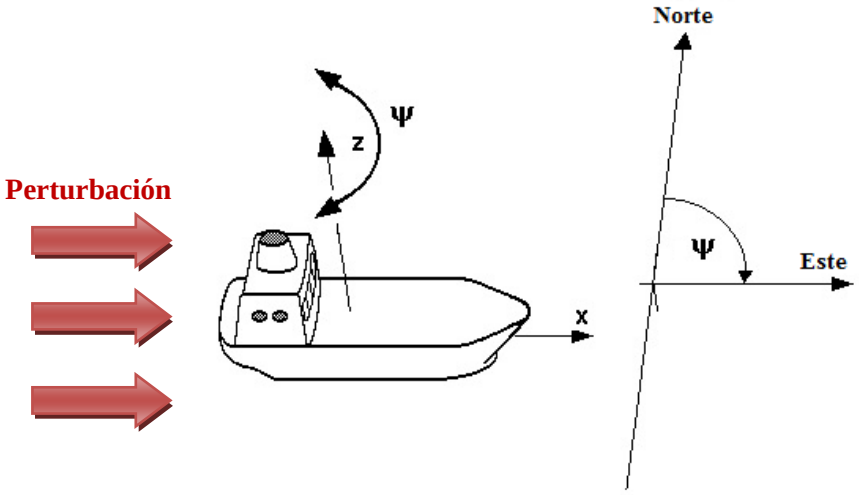

Figura 6.3. Efecto de perturbación paralela

6.1.3. Perturbación a +45º

El resto de posibles casos serían una combinación de lo descrito hasta el momento. Por ejemplo, la siguiente imagen muestra un caso en el que el barco se desplaza con rumbo ESTE, mientras sufre una perturbación procedente del NORDESTE.

En estas condiciones, la velocidad de desplazamiento se reducirá debido al efecto contrario de la perturbación (aunque su efecto será menor que si atacase directamente desde el ESTE, como se ha visto en el ejemplo anterior). Del mismo modo, el rumbo del barco también se verá afectado por la perturbación, al aplicarse una fuerza que tiende a hacer virar el barco en dirección SUR (provocando que se reduzca la velocidad angular de giro cuando se desea virar hacia el NORTE, y haciéndola aumentar cuando se desee virar hacia el SUR):

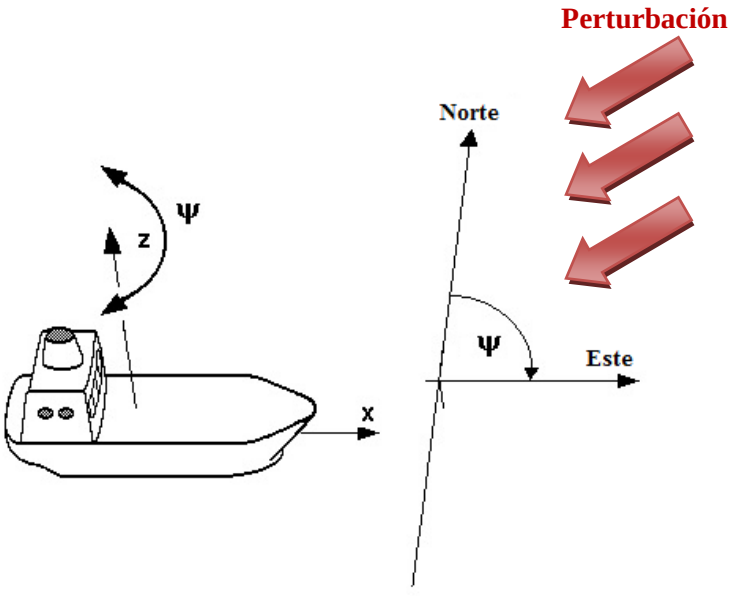

Figura 6.4. Efecto de perturbación con ángulo de ataque de 45º

6.2. Diseño de la Red Neuronal Artificial (RNA)

Para tratar de minimizar los efectos debidos a estas perturbaciones, se pretende aplicar un control adaptativo basado en una Red Neuronal Artificial.

El objetivo es diseñar una red neuronal que estime la corrección que se debería aplicar al rumbo deseado para tratar de minimizar el posible efecto negativo de las perturbaciones que afecten al barco. De este modo, se creará una red neuronal que en función de la dirección e intensidad de la perturbación, así como del rumbo que vaya a llevar el barco, sea capaz de generar una salida que se aplique a modo de corrección al vector del rumbo deseado de forma que se contrarreste el efecto de la perturbación a lo largo de toda la trayectoria a realizar.

6.2.1. RNA

Una red neuronal artificial es un modelo heurístico que simula la estructura de un sistema neuronal biológico con el fin de alcanzar una funcionalidad similar [18]. El procesador elemental, o neurona artificial, es un dispositivo simple de cálculo (unidad de procesamiento de la información), que ante un vector de entradas proporciona una única salida:

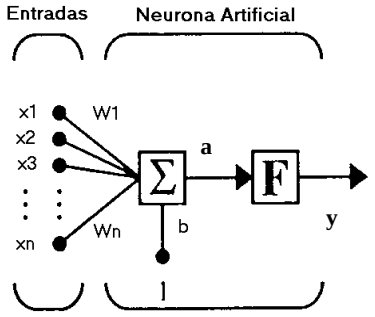

Figura 6.5. Esquema de una unidad neuronal de la RNA

La imagen anterior muestra el esquema básico de una neurona artificial, siendo:

- Conjunto de entradas: x_j
- Pesos sinápticos: w_i
- Función de activación: $w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n = a$
- Función de transferencia: $y = F(w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n)$
- Bias o polarización (b): entrada constate de magnitud 1, y peso b que se introduce en el sumador.

Una neurona aislada dispone de poca potencia de cálculo. Es por ello que, en general, una red neuronal consiste en el establecimiento de un conjunto de nodos o neuronas, agrupadas en varias capas e interconectadas entre sí.

Tal y como se muestra en la siguiente imagen, existen tres tipos de capas: capa de entrada, capas ocultas o intermedias y capa de salida:

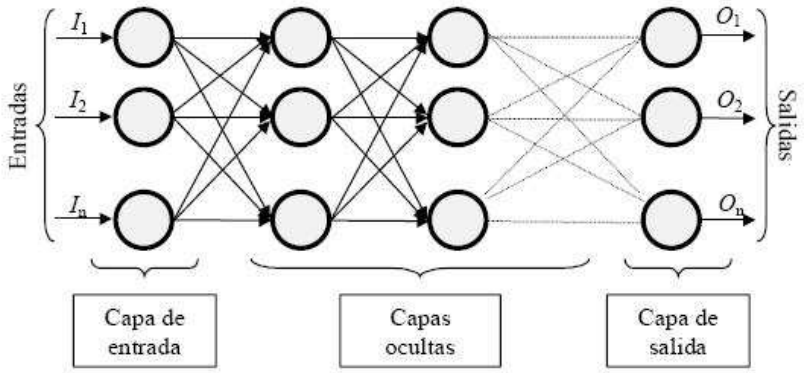

Figura 6.6. Relación de capas de una RNA

Las conexiones que unen a las neuronas tienen asociado un peso numérico, y es mediante la actualización de estos valores como se realiza el proceso de aprendizaje de la red.

Existen diferentes tipos de algoritmos de aprendizaje (supervisado, no supervisado, modelos híbridos, etc.). En este trabajo se va a utilizar, uno de los algoritmos supervisados más empleados en el entrenamiento de RNAs: el aprendizaje de propagación hacia atrás de errores o backpropagation [19].

Su funcionamiento se basa en modificar los valores de los pesos de forma proporcional al gradiente de la función de error con el objeto de alcanzar mínimos locales. El algoritmo emplea un ciclo propagación–adaptación de dos fases. En primer lugar se aplica a modo de estimulo un patrón a la entrada de la red, que será propagado desde la primera capa a través del resto de capas intermedias hasta generar la salida correspondiente. Esta salida se comparará con la salida deseada, calculando una señal de error para cada una de las salidas.

Las salidas de error se propagarán hacia atrás, recorriendo el camino inverso a través de todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento.

6.2.2. Esquema control adaptativo RNA

Como ya se ha comentado anteriormente, el efecto de la perturbación sobre el movimiento del barco dependerá del rumbo que sigue el barco, el ángulo de ataque de la perturbación respecto a dicho rumbo, y la intensidad de la propia perturbación.

La solución que se ha tomado es desarrollar una red neuronal que tenga como objetivo aplicar una corrección al rumbo deseado, ψ_d (t), que trate de minimizar el efecto negativo de las perturbaciones. De este modo, la RNA deberá generar como salida un vector de igual longitud al del rumbo deseado, cuyos valores serán la diferencia a aplicar en cada momento sobre ψ_d para contrarrestar el efecto que vaya a producir la perturbación concreta que se de en cada caso.

De forma esquemática, la red quedaría definida de la siguiente forma:

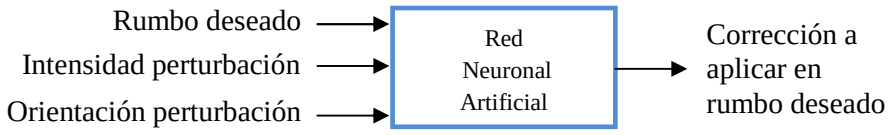

Figura 6.7. Esquema de la RNA

El esquema que se va a seguir para lograr esto es el siguiente:

- En primer lugar se creará la red neuronal (sus principales características se definirán a continuación).
- Esa RNA se entrenará aplicando diferentes vectores de rumbo, así como intensidades y ángulos de ataque de la perturbación. Como salida objetivo se tomará la diferencia entre el rumbo deseado y el rumbo real derivado del modelado del barco tras aplicar el efecto de la perturbación (la diferencia de estos dos valores se calcula en Simulink y se pasa a Matlab para que se aplique en el entrenamiento de la red).
- Una vez que se disponga de la red correctamente entrenada, cuando se determine un tipo de trayectoria a realizar por el barco, la red neuronal calculará, en función de las

características de la perturbación existentes en esos momentos, la corrección necesaria para reducir los efectos de dicha perturbación en esa trayectoria concreta. Esta corrección se aplicarla al vector ψ_d (t), generando un nuevo vector de rumbo deseado que será el que se pase al sistema de control.

Tal y como se describirá a continuación, la RNA se define íntegramente en Matlab (así como su entrenamiento, simulación, etc.).

El funcionamiento de este sistema queda dividido en dos partes principales:

- En la primera, se define en Matlab la red neuronal, así como unos vectores genéricos de rumbo deseado que servirán para entrenar la RNA (asociados a unas condiciones concretas de perturbaciones). Estos vectores son pasados a Simulink para proceder a una simulación que permita calcular las diferencias angulares provocadas por las perturbaciones. Los valores obtenidos en la simulación se pasarán de vuelta a Matlab por medio del Workspace para ser utilizados en el proceso de entrenamiento la red neuronal.
- Una vez que la red ya está correctamente entrenada, se procede al funcionamiento normal del sistema. La única variación respecto al sistema descrito en el capítulo 5 será el hecho de realizar un procesamiento previo del vector de rumbo deseado en la red neuronal. Esto se realizará antes de lanzar la simulación de Simulink. De este modo, la red generará como salida el vector de corrección del rumbo, almacenándolo en la variable correspondiente para su posterior aplicación al rumbo deseado en Simulink.

Aspecto importante para el correcto funcionamiento del sistema es verificar el correcto entrenamiento de la red. Para considerar que el proceso de entrenamiento ha sido correcto, se ha estimado que el error de la red debe ser menor al 0.001 (valor "Performance" en la figura 6.8).

En los resultados que se van a mostrar en los siguientes apartados únicamente se aplicará en el sistema el controlador PID de ganancias programadas optimizado mediante AG. Se ha tomado la decisión de estudiar únicamente el caso de este controlador por haber sido el que mejores resultados ha dado en las simulaciones realizadas en trayectorias normales. De todos modos, la RNA se podría aplicar sin ningún problema en un sistema con cualquiera de los otros controladores desarrollados.

6.2.3. Características de la RNA

Para crear la RNA se ha utilizado el comando *newff* de Matlab, que sirve para crear una red neuronal feedforward con backpropagation. El formato de ejecución de esta función es el siguiente:

net = newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)

siendo sus principales parámetros:

- P: Matriz con los patrones de entrada.
- T: Matriz con las salidas deseadas.
- S_i: Tamaño de la capa i (para las capas ocultas).
- TF_i: Función de transferencia de la capa i (por defecto "*tansig*" para capas ocultas y "*purelin*" para la capa de salida).
- BTF: Función de entrenamiento (por defecto "*trainlm*").
- BLF: Función de aprendizaje de los pesos/bias (por defecto "*learngdm*").
- PF: Función de evaluación (por defecto "*mse*").
- net: La función devuelve una red feedforward con N capas.

Una vez creada la red, el siguiente paso es realizar el entrenamiento con los patrones de entrada y salida deseados. Para esto existen dos tipos de entrenamiento:

- Estático: En cada ciclo de entrenamiento se recalculan los pesos de la red tras presentar todos los patrones de entrenamiento (función *train*).
- Adaptativo: En cada ciclo de entrenamiento se recalculan los pesos tras presentar cada uno de los patrones de entrenamiento (función *adapt*).

En este caso concreto se ha optado por el entrenamiento estático:

[net,TR,Y,E] = train(NET,P,T)

Los parámetros de entrada y salida serán:

- NET: Red inicializada.
- P: Patrones de entrada.
- T: Salidas deseadas.
- net: Red entrenada.
- TR: Error en función de la iteración
- Y: Salida de la red.
- E: Errores de la red.

Existen diversos métodos para realizar el entrenamiento estático de una RNA. Estos métodos se basan en algoritmos que intentan minimizar el error en base a diversas técnicas. Cada uno de estos métodos presenta sus ventajas e inconvenientes en cuanto a convergencia y coste computacional. Algunas de las funciones de entrenamiento que se pueden utilizar son trainlm, traingd, traingdx, trainbr, etc.

Al igual que las funciones de transferencia, la función de entrenamiento se establece al crear la red o bien alterando el valor del parámetro *NET.transferFcn*.

Tras la fase de entrenamiento, la red estará lista para su aplicación en el sistema. Esto implica que la RNA será capaz de producir una salida adecuada a un conjunto de patrones de entrada. La función *sim* de Matlab es la encargada de pasar un conjunto de patrones de entrada a la red y para obtener su salida, la forma de llamar a la función es la siguiente:

Y = sim(NET,P)

siendo:

- NET: Red entrenada.
- P: Conjunto de patrones de entrada.
- Y: Salida de la red.

Para esta aplicación, la red neuronal ha sido definida de la siguiente manera:

- Se definen 3 entradas para la RNA: El vector del rumbo deseado, el ángulo de ataque de la perturbación y la intensidad de la perturbación. Los valores relacionados con la perturbación (ángulo e intensidad) no se almacenan como una variable constante, sino que se consideran un vector de iguales dimensiones a ψ_d (t). De este modo, la aplicación permite que las características de la perturbación puedan sufrir variaciones a lo largo del recorrido del barco.
- La salida generada por la RNA es la corrección a aplicar al vector ψ_d (t) para minimizar el efecto producido por las perturbaciones.
- En primer lugar se crea la matriz R de 3x2. La primera columna muestra el valor mínimo de las 3 entradas, mientras que la segunda columna hace referencia al valor máximo (rumbo deseado, ángulo de perturbación, intensidad de perturbación):

  R=[-360 360; 0 360 ; 0 20];
- Se define el vector "S" que indica el tamaño de las capas de la red. Tras realizar diversas pruebas con varias configuraciones de capas, finalmente se ha definido una red con una única capa intermedia de 5 neuronas. La representación de esta configuración de red se muestra la figura 6.9:

  S=[5 1];
- De este modo, se crea la red con las funciones de transferencia por defecto, tansig y purelin:

  net=newff(R,S,{'tansig','purelin'});
- Se entrena la red con los datos generados en una primera simulación (almacenados en la variable "*O*") a partir de los vectores genéricos definidos para la fase de entrenamiento (almacenados en la variable "*I*"). La figura 6.8 muestra el GUI de entrenamiento de la red neuronal que proporciona la toolbox:

  I=[input'; ángulo; fuerza];
  O=output;
  net=train(net,I,O);

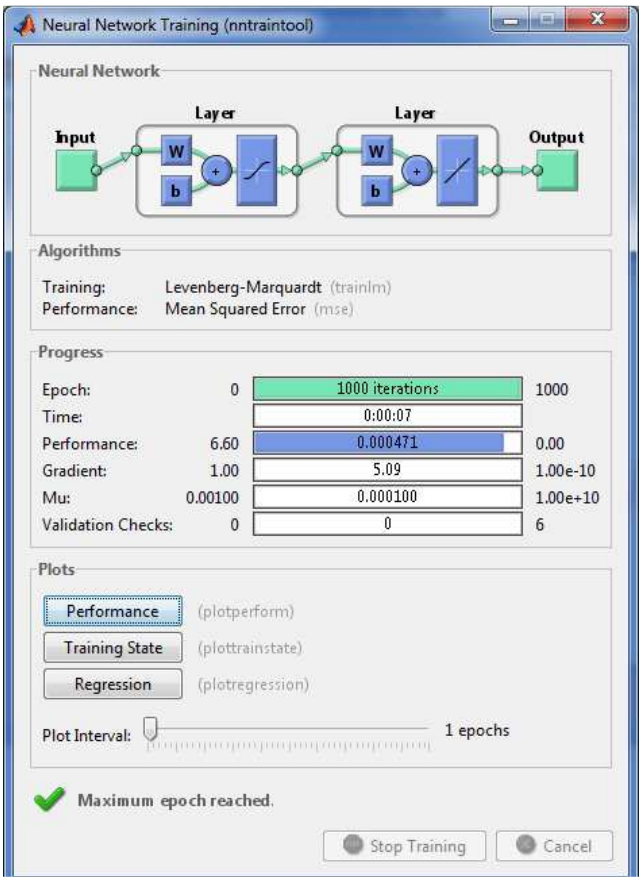

Figura 6.8. Interface de Matlab para el entrenamiento de la red, "nntraintool"

- Una vez entrenada la red, se pueden realizar simulaciones con el rumbo deseado para cualquier tipo de trayectoria, y obtener así la salida de la RNA y aplicarla a modo de corrección en el rumbo deseado:

  O1=sim(net,I);

  - En este caso "*I*" sería una matriz que contendría el rumbo deseado y el estado de las perturbaciones a aplicar, mientras que "*O1*" sería la salida calculada por la RNA (equivalente a la corrección a aplicar posteriormente sobre el rumbo deseado para minimizar el efecto de las perturbaciones).

La variable *IW* de la red *net* permitirá comprobar el valor con el que han quedado los pesos entre las neuronas de entrada y las de la capa intermedia, mientras que la variable *LW* permitirá ver los pesos aplicados de las neuronas de capas intermedias hasta la neurona de salida. En este caso, al haber definido una red con una capa intermedia de 5 neuronas, la variable *IW* almacenará los valores de los pesos en una matriz de 5x3, mientras que *LW* lo hará en una de 1x5.

A modo de ejemplo, a continuación se muestra los pesos de las conexiones entre las neuronas de entrada y de la capa intermedia obtenidos tras uno de los procesos de entrenamiento realizados:

```
>> net.IW{1}
ans =
  -0.0072   -0.0106    0.0618
  -0.0037   -0.0645   -0.0802
  -0.0479   -0.0052   -0.1014
   0.0037   -0.0133   -0.0358
   0.0109   -0.0083    0.1589
```

Los cinco valores de la primera columna representan los pesos entre la primera neurona de entrada y las 5 neuronas de la capa intermedia, y así sucesivamente. A modo de ejemplo, a continuación se muestra la representación de la red neuronal definida, así como los valores de los pesos obtenidos para la tercera neurona de entrada (en este caso se trata de la entrada correspondiente a la intensidad de la perturbación):

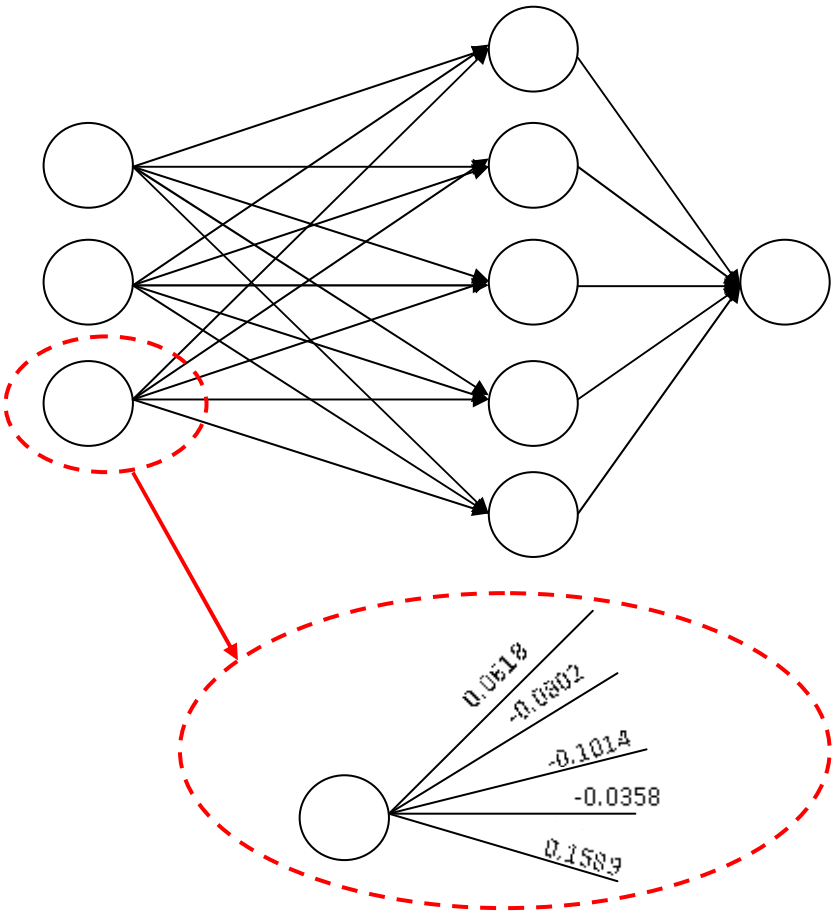

Figura 6.9. Esquema de las capas neuronales de red

Como ya se ha indicado anteriormente, un aspecto importante a la hora de generar la red es crear una serie de vectores de rumbo, asociados a intensidades y ángulos de ataque de las perturbaciones, que sirvan para entrenar a la red neuronal de la mejor forma posible, haciéndola capaz de calcular a la hora de aplicar un rumbo deseado cualquiera (bajo el efecto de cualquier tipo de perturbación) una salida adecuada.

Teniendo en cuenta que la dirección de ataque de las perturbaciones puede ser cualquiera, un tipo de trayectoria interesante a la hora de entrenar la red son los recorridos circulares, ya que se trata de la forma más rápida de ver la relación del efecto para cualquier dirección del rumbo.

A continuación se muestran una serie de resultados obtenidos con diferentes vectores de entrenamiento que se han definido, donde se aprecia la diferencia entre la salida de la RNA y el vector real que determina la diferencia entre el rumbo deseado y el obtenido como consecuencia de las perturbaciones:

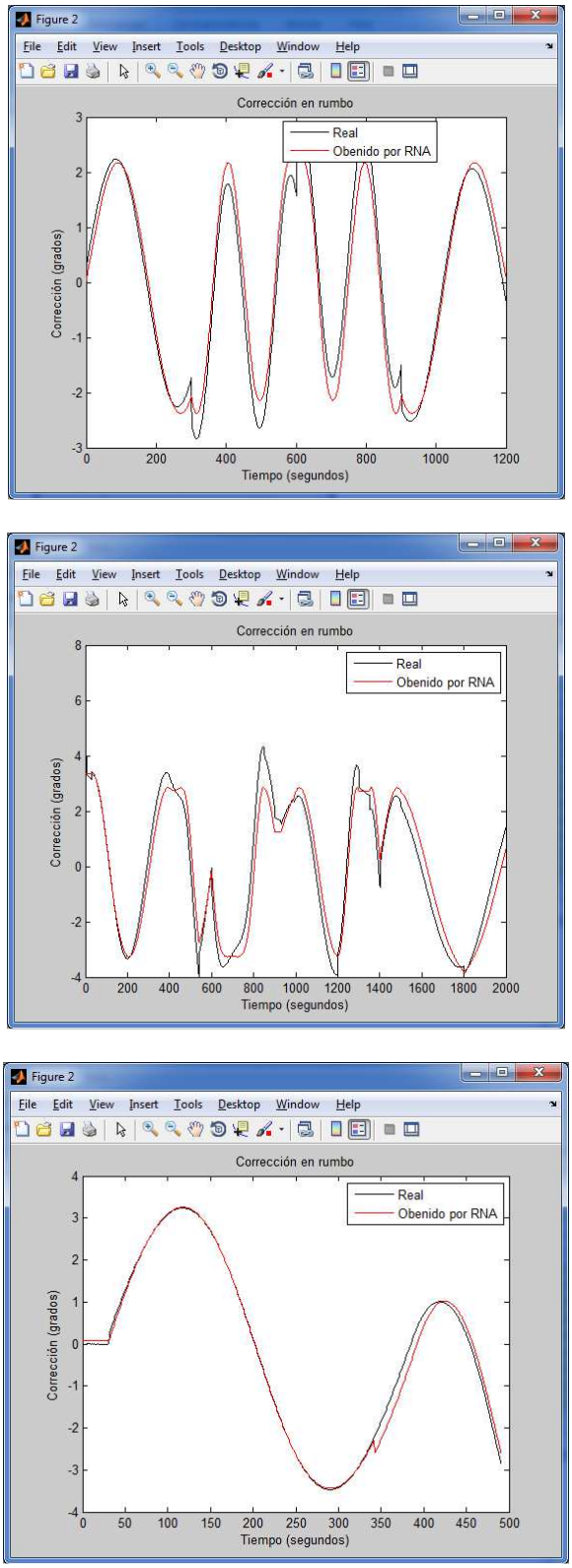

Figura 6.10. Ejemplo de resultado con varios patrones de entrenamiento

Aplicando la red entrenada a un caso de simulación real de rumbo deseado, la siguiente figura muestra el resultado para una perturbación de intensidad media-alta, con dirección OESTE a ESTE, sobre un barco que realiza una trayectoria circular. En negro se representa la corrección real que se debería aplicar para suplir el efecto de la perturbación y en rojo la salida de la RNA (y por consiguiente, lo que realmente se va a aplicar como corrección al vector de rumbo deseado):

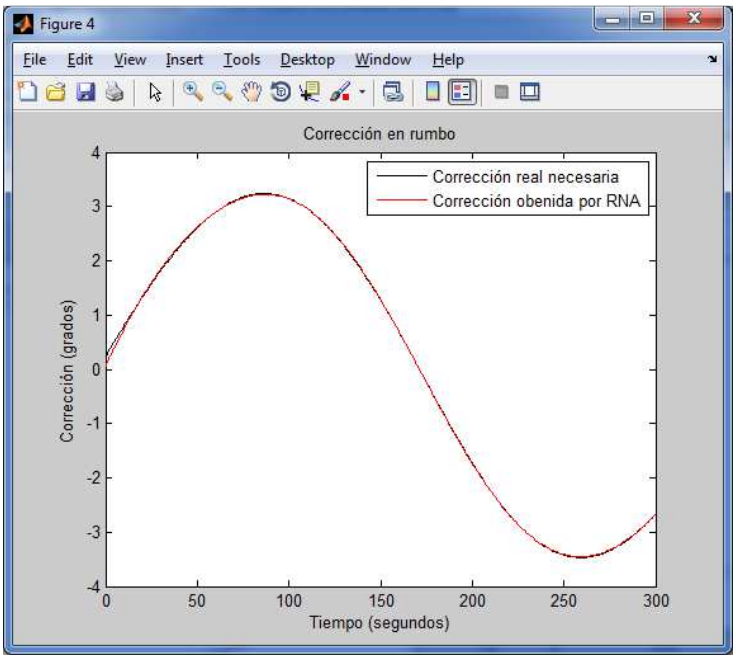

Figura 6.11. Relación de vector de corrección en trayectoria circular

A pesar de que en los siguientes apartados se van a analizar los resultados obtenidos en las diversas simulaciones, la siguiente figura muestra a modo de ejemplo el resultado que se obtiene en la trayectoria del barco al aplicar la corrección de la figura anterior:

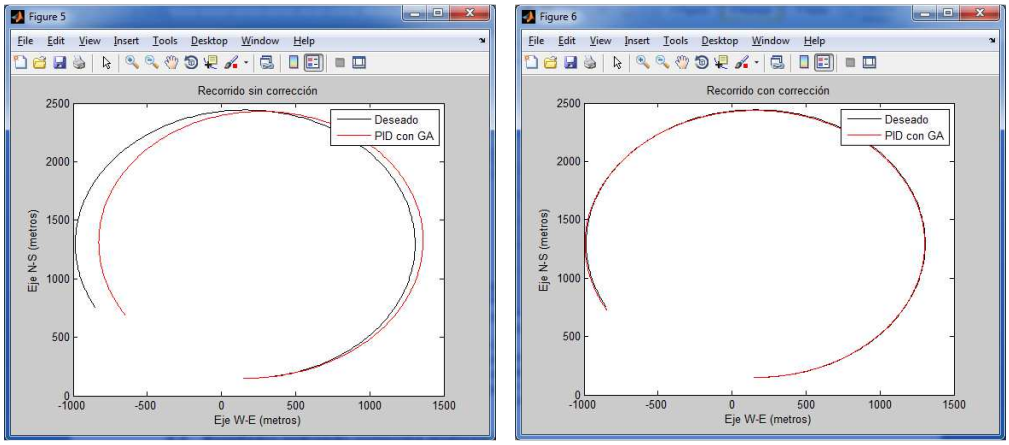

Figura 6.12. Recorrido sin corrección (izquierda) y con corrección (derecha)

En la figura de la izquierda se observa claramente el efecto de la perturbación que ataca desde el OESTE, haciendo que el barco se desvíe (aspecto que sobre todo se aprecia en los tramos de la trayectoria en que sufre el ataque de la perturbación de forma perpendicular). En la figura de la derecha se puede ver el efecto de la corrección aplicada sobre el vector de rumbo deseado, ya que se consigue que la trayectoria real del barco se mantenga lo más parecido posible a la deseada.

6.3. Resultados aplicando corrección mediante RNA

En este apartado se van a presentar una serie de simulaciones donde se puede verificar el efecto de las perturbaciones sobre el barco al realizar diferentes tipos de recorridos, así como el efecto de la corrección de rumbo calculada mediante la RNA diseñada. Para esto se han elegido tres tipos de trayectorias representativas:

- Trayectoria recta con cambio de rumbo puntual. El análisis de este tipo de trayectorias permitirá comprobar el efecto de las perturbaciones al aplicarse de forma constante sobre el barco que trata de mantener un rumbo fijo.
- Trayectorias circulares y sinusoidales. Con este tipo de trayectorias se podrá comprobar el efecto cambiante de las perturbaciones a lo largo del recorrido.

A cada una de ellas se le aplicarán distintos tipos de perturbaciones (modificando tanto la intensidad, como el ángulo de ataque de las mismas), para analizar en su totalidad el efecto que provocan.

Se podrán ver conjuntamente dos representaciones: a la izquierda la simulación sin aplicar la corrección de rumbo, y a la derecha la simulación aplicando la salida de la RNA a modo de corrección del rumbo. Se representará con una flecha roja la dirección de ataque de la perturbación, así como su intensidad.

En la siguientes figuras (6.13 y 6.14) se muestra cómo se consigue minimizar el error en una trayectoria recta con un cambio de rumbo puntual. En la imagen de la izquierda de la figura 6.13 se observa que a medida que avanza el tiempo, el barco se aleja más de su trayectoria deseada debido al empuje de la perturbación en dirección ESTE. La principal diferencia con la figura 6.14 se puede ver en la zona de cambio de rumbo. En el segundo caso el barco realiza un giro mucho más abierto (lo que sitúa al barco más alejado de su trayectoria deseada), debido a que la perturbación le ataca de forma perpendicular, y por lo tanto su efecto es mucho más evidente a la hora de realizar el viraje.

Una vez cambiado el rumbo, el segundo caso sufre con menos impacto el efecto de la perturbación debido a su ángulo de ataque relativo.

Al aplicar en el caso mostrado en la figura 6.13 la corrección calculada mediante RNA, se observa la trayectoria realizada sigue estando desviada en dirección al efecto perturbador, sin embargo, se observa que por lo menos se consigue mantener el rumbo constante y más parecido al deseado.

Para el caso de una perturbación que ataca de forma perpendicular al barco, se puede observar como al inicio de su trayectoria el barco no consigue mantenerse en línea recta (ver zoom de la parte izquierda de la figura 6.14). De este modo, si en lugar de un cambio de rumbo, el barco tuviese como objetivo mantener un recorrido en línea recta (con rumbo fijo), en el caso de no aplicar la corrección obtenida por la RNA la trayectoria quedaría totalmente desviada debido al efecto de la perturbación.

En el caso en que se aplica la corrección de rumbo (ver zoom de la parte derecha de la figura 6.14), se observa que el barco consigue mantenerse en línea recta, y por lo tanto mantener su rumbo constante.

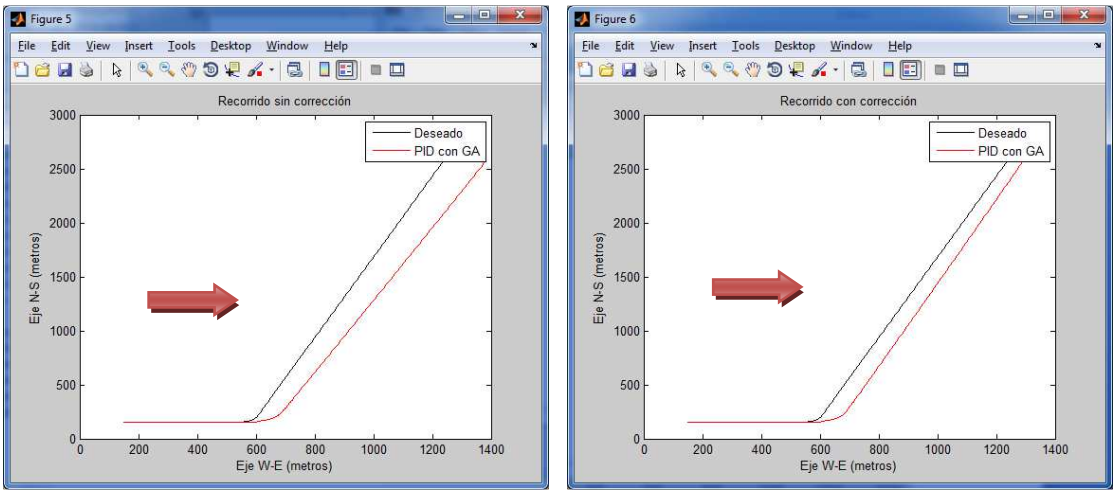

Figura 6.13. Corrección en una trayectoria recta para una perturbación paralela y de intensidad media

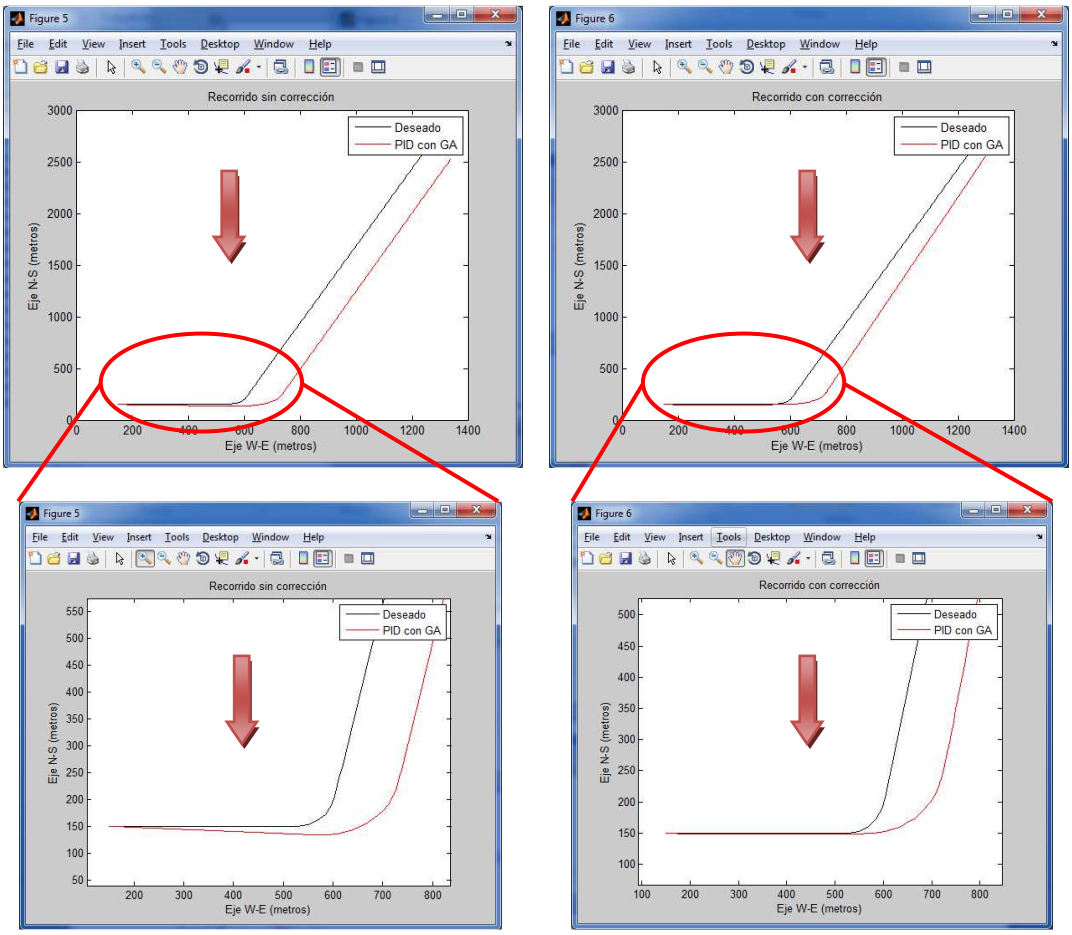

Figura 6.14. Corrección en una trayectoria recta para una perturbación perpendicular y de intensidad media

A continuación se va a observar como la aplicación de la corrección de rumbo en las trayectorias circulares y sinusoidales corrige prácticamente en su totalidad los posibles desfases y diferencias respecto al recorrido deseado.

En las trayectorias circulares es interesante analizar el efecto de una perturbación, ya que si ésta se mantiene constante en todo su recorrido, el efecto sobre el barco cubrirá todos los ángulos de ataque posibles respecto al rumbo (de 0 a 360º al dar una vuelta completa).

Las figuras 6.15 y 6.16 muestran el efecto sobre una trayectoria circular, para una perturbación de intensidad baja y con dirección ESTE a OESTE (figura 6.15), y para otra con intensidad alta procedente del NORDESTE (figura 6.16). La figura 6.17 muestra el efecto sobre una trayectoria sinusoidal con una perturbación de intensidad alta.

El efecto de la perturbación es claro en las tres figuras, ya que la trayectoria realmente realizada por el barco tiende a desplazarse según lo esperado. Del mismo modo, la intensidad de la propia perturbación hace que cuanto mayor sea ésta, la desviación respecto a la trayectoria deseada sea más evidente.

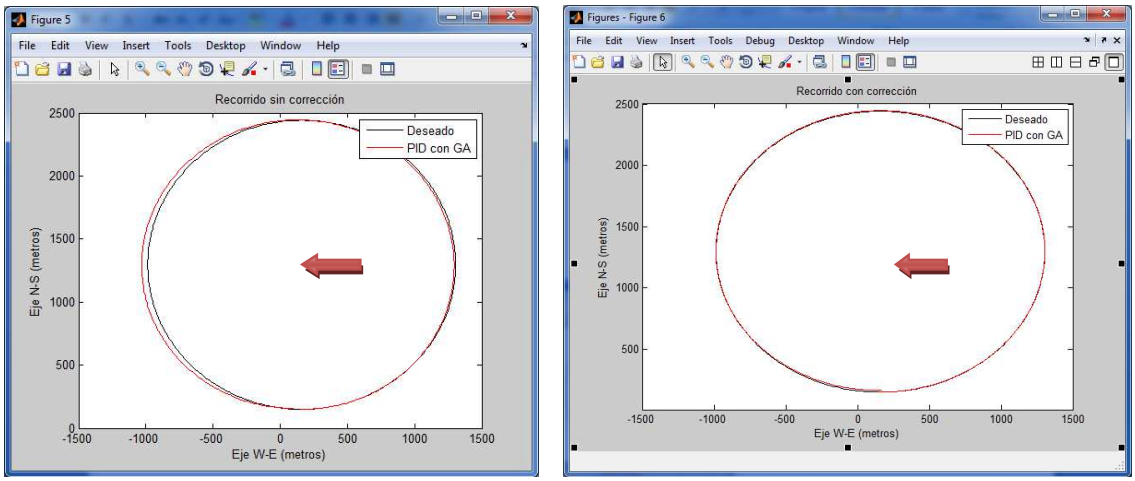

Figura 6.15. Corrección en trayectoria circular para una perturbación de intensidad baja

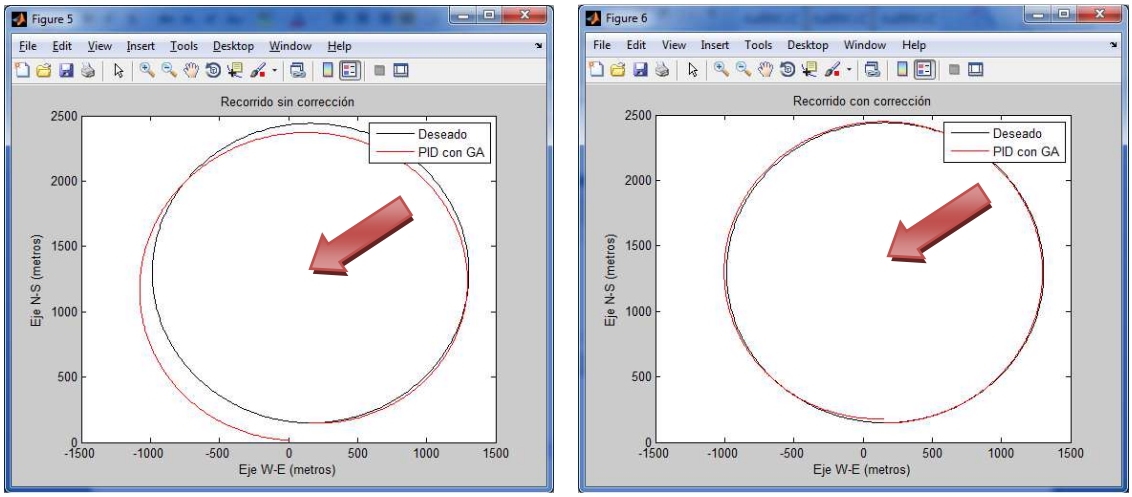

Figura 6.16. Corrección en trayectoria circular para una perturbación de intensidad alta

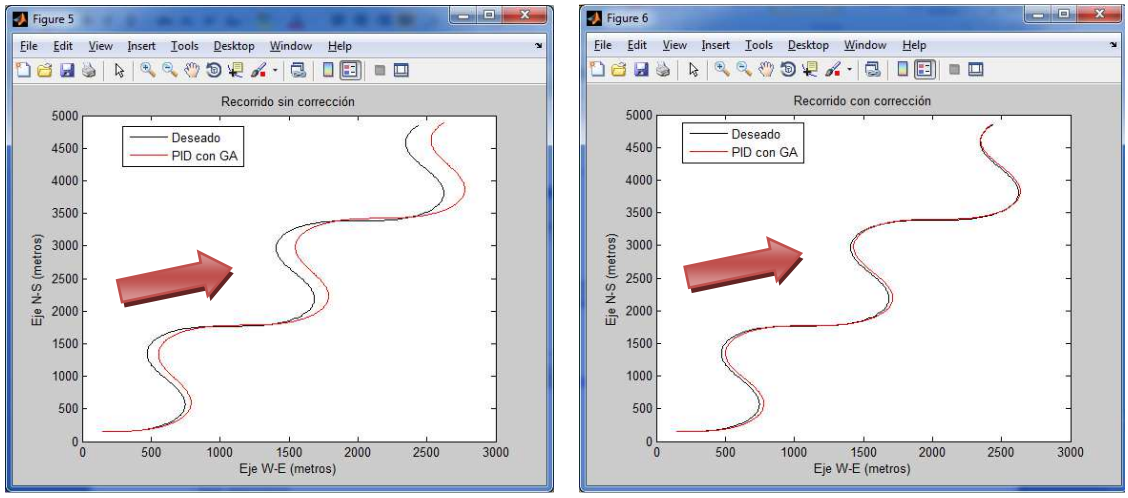

Figura 6.17. Corrección en trayectoria sinusoidal para una perturbación de intensidad alta

En las trayectorias circulares y sinusoidales los cambios de rumbo son más progresivos y por lo tanto se pueden considerar relativamente más suaves. Esto hace que los resultados al aplicar la corrección mediante RNA sean más parecidos a las trayectorias deseadas.

7. Conclusiones

En este capítulo se enlazan los objetivos del trabajo con los resultados obtenidos, recogiendo las conclusiones generadas a partir del desarrollo del sistema final. Adicionalmente, se hacen una serie de comentarios en relación a los trabajos y líneas futuras que podrían seguirse para refinar, mejorar y extender el sistema.

7.1. Conclusiones

El objetivo principal de este trabajo ha sido desarrollar varios tipos de controladores inteligentes para el control del rumbo de un barco, tratando de mejorar los resultados obtenidos con un control clásico.

Tal y como se ha podido observar en los resultados del capítulo 5, los controladores inteligentes desarrollados (tanto el controlador difuso, como el optimizado mediante AG) ofrecen una mejor respuesta en comparación al controlador PID clásico analizado.

Además, al haber implementado un algoritmo genético que se adapta modificando la función objetivo en función del tipo de cambio de rumbo, se consigue optimizar la respuesta del sistema para cualquier tipo de trayectoria. También se ha conseguido desarrollar un sistema capaz de contrarrestar los efectos de las perturbaciones en el barco mediante una red neuronal. Las mejoras obtenidas con esta RNA se han podido comprobar en el capítulo anterior. De este modo, se ha demostrado que la aplicación de técnicas de soft-computing al control del rumbo de un barco proporcionan buenos resultados y los hace verdaderamente interesantes para su desarrollo.

Además, con el desarrollo de este trabajo se ha podido analizar cómo pueden afectar en el sistema algunas variaciones del sistema borroso, así como algunos de los aspectos configurables del algoritmo genético.

7.2. Trabajos a futuro

A modo de trabajos a futuro para tratar de mejor u optimizar los controladores de rumbo desarrollados mediante técnicas de control inteligente, a continuación se numeran algunas que podrían ser de interés:

- Optimización o mejora del algoritmo genético:
    - Terminación del algoritmo: El código actual finaliza el algoritmo cuando se realizan todas las iteraciones del bucle definidas por la variable que determina el máximo número de generaciones. Se podría aplicar una modificación en el programa para que el algoritmo pudiera finalizar antes de tiempo, como por ejemplo: en caso de alcanzar una solución satisfactoria (cuando un individuo generase un error menor a un valor determinado que se considere adecuado), o cuando se diesen sucesivas iteraciones sin ninguna mejoría en los resultados.

    - Operación de cruce: La operación de cruce entre dos individuos se realiza determinando de forma aleatoria la posición, o el bit, a partir del cual se realiza el intercambio de información entre los dos cromosomas. En vista de la aplicación, podría ser interesante definir un nuevo método de cruce en el que se determinasen 3 puntos de crossover (uno para cada parámetro del controlador).

- Optimización o mejora del controlador difuso:
    - Hacerlo adaptable en función del tipo de cambio de rumbo como se ha hecho con el algoritmo genético. Aprovechando las 3 ganancias aplicadas a las entradas y salida del controlador, sería interesante analizar el efecto de su

modificación dependiendo del tipo de trayectoria, y optimizar el programa haciendo que el controlador difuso también se adapte en función de los cambios de rumbo que se vayan a aplicar.

- La toolbox *fuzzy* de Matlab permite una infinidad de posibilidades de configuración del controlador difuso, como por ejemplo: tipos de curva de las señales de entrada/salida (gaussiana, sigmoidal…), método de defuzificación, etc. Sería interesante aplicar un mayor abanico de configuraciones y analizar los resultados para determinar que opciones ofrecen un mejor comportamiento del sistema.

8. Bibliografía

1. Fossen, T. I. (2002). *Marine control systems: Guidance, navigation and control of ships, rigs and underwater vehicles* (pp. 82-92356). Trondheim: Marine Cybernetics.
2. Golding, B. K. (2004). *Industrial systems for guidance and control of marine surface vessels*. Technical Report Project assignment, Department of Engineering Cybernetics, Norwegian University of Science and Technology NTNU, Trondheim, Norway.
3. Van Amerongen, J., & Udink Ten Cate, A. J. (1975). Model reference adaptive autopilots for ships. *Automatica*, *11*(5), 441-449.
4. Do, K. D. (2010). Practical control of underactuated ships. *Ocean Engineering*,*37*(13), 1111-1119.
5. Do, K. D., Jiang, Z. P., & Pan, J. (2003). Robust global stabilization of underactuated ships on a linear course: state and output feedback.*International Journal of Control*, *76*(1), 1-17.
6. Fossen, T. I., Breivik, M., & Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. *Proceedings of the 6th IFAC MCMC, Girona, Spain*, 244-249.
7. Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.
8. Vukić, Z., & Velagić, J. (1999, August). Comparative analysis of Mamdani and Sugeno type fuzzy autopilots for ships. In *5th European Control Conference (ECC'99)* (Vol. 31, p. 157).
9. Velagic, J., Vukic, Z., & Omerdic, E. (2003). Adaptive fuzzy ship autopilot for track-keeping. *Control engineering practice*, *11*(4), 433-443.
10. Zhang, Y., Hearn, G. E., & Sen, P. (1996). A neural network approach to ship track-keeping control. *Oceanic Engineering, IEEE Journal of*, *21*(4), 513-527.
11. Santos, M. (2011). Un enfoque aplicado del control inteligente. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, *8*(4), 283-296.
12. Astrom, K. J., & Hagglund, T. (2006). *Advanced PID control*. Isa.
13. Mitchell, M. (1998). An introduction to genetic algorithms (complex adaptive systems).
14. Mirzal, A., Yoshii, S., & Furukawa, M. (2012). PID Parameters Optimization by Using Genetic Algorithm. *arXiv preprint arXiv:1204.0885*.
15. Maiti, D., Acharya, A., Chakraborty, M., Konar, A., & Janarthanan, R. (2008, December). Tuning PID and PI/λ D δ Controllers using the Integral Time Absolute Error Criterion. In *Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference on* (pp. 457-462). IEEE.
16. MathWorks, Inc, & Wang, W. C. (1998). *Fuzzy Logic Toolbox: for Use with MATLAB: User's Guide*. MathWorks, Incorporated.
17. Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, *8*(3), 338-353.
18. Freeman, J. A., & Skapura, D. M. (1992). Neural Networks: Algorithms, Applications and Programming Techniques. *Reading, PA: Addison-Wesley Publishing Company*, 401.
19. Rummelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(9), 533-535.
20. Perez, T., & Blanke, M. (2002). *Mathematical Ship Modelling for Control Applications*. Technical Report. Technical University of Denmark.
21. Antsaklis, P. (1994). Defining intelligent control. *Report of the Task Force on Intelligent Control, IEEE Control Systems*, 0272-1708.
22. García, J. M., Almansa, J. A., & Sierra, J. M. G. (2012). Automática marina: una revisión desde el punto de vista del control. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, *9*(3), 205-218.
23. López, R., Santos, M., Polo, O., & Esteban, S. (2002). Experimenting a fuzzy controller on a fast ferry. In *Control Applications, 2002. Proceedings of the 2002 International Conference on* (Vol. 2, pp. 1082-1087). IEEE.
24. Pathan, D.M., Unar, M.A., & Memon, Z.A. (2012). Fuzzy Logic Trajectory Tracking Controller for a Tanker. *Mehran University Research Journal of Engineering & Technology* (Vol. 31).
25. Rekdalsbakken, W., & Steve, A. (2007). Simulation on of Intelligent Ship Autopilots.

26. González, F. J. V., Rodríguez, T. M. R., García, E. L., Pérez, E. M., & López, J. L. N. Aplicación del Control QFT Robusto a la Maniobra de Cambio de Rumbo de un Buque.
27. Antonic, R., Valcic, M., & Tomas, V. (2012, April). Ship speed prediction in real sea environment using advanced technologies. In *Proceedings of the 11th WSEAS international conference on Applied Computer and Applied Computational Science* (pp. 17-22). World Scientific and Engineering Academy and Society (WSEAS).
28. Santos, M., López, R., & de la Cruz, J. M. (2010). Modelo predictivo neuro-borroso de la aceleración de cabeceo de un buque de alta velocidad. *RIAII*,*2*(3), 39-47.

9. Acrónimos y definiciones

PID *Proporcional Integral Derivativo*
GDL *Grados De Libertad*
IA *Inteligencia Artificial*
AG *Algoritmo genético*
RNA *Red Neuronal Artificial*
SVM *Support Vector Machine*
ZMF *Z-shaped Membership Function*
SMF *S-shaped Membership Function*
DGPS *Differential Global Positioning System*
GUI (*del inglés) Graphical User Interface (Interfaz Gráfico de Usuario)*
IMU (*del inglés) Inertial Measurement Unit (Unidad de Medición Inercial)*
MEMS (*del inglés) Microelectromechanical Systems (Sistemas Microelectromecánicos)*
Giroscopio: *Dispositivo mecánico que sirve para medir, mantener o cambiar la orientación en el espacio de algún aparato o vehículo.*
Soft computing: *Término empleado en las ciencias computacionales que engloba diversas técnicas (redes neuronales, lógica difusa, algoritmos evolutivos, etc.) empleadas para solucionar problemas que manejan información incompleta, con incertidumbre e inexacta.*
Sobrelongación: *Término utilizado para expresar cuando una señal o función excede su valor objetivo (también llamado "sobrepico").*
Feed-forward (o prealimentación): *Describe un tipo de sistema que reacciona a los cambios en su entorno, normalmente para mantener algún estado concreto del sistema.*
Eslora: *Dimensión de un barco tomada a su largo (de proa a popa).*
Manga: *Medida del barco en el sentido transversal (de estribor a babor).*
Calado: *Distancia vertical entre un punto de la línea de flotación y la línea base o quilla (espesor del casco incluido).*

I. Anexo 1: Optimización AG

En el presente anexo se muestran los resultados de las pruebas realizadas al tratar de optimizar los diferentes aspectos configurables del algoritmo genético diseñado.

Análisis del tiempo de convergencia del algoritmo genético

A continuación se van a mostrar los resultados obtenidos a la hora de analizar los parámetros que optimizan la relación entre el tiempo de convergencia del algoritmo y la obtención de unos resultados que puedan considerarse aceptables.

Para esto, fijando una función objetivo y un tipo de rumbo concretos, se han realizado diversas pruebas modificando los parámetros correspondientes a la cantidad de generaciones y número de individuos por población (ya que éstos son los parámetros que mayor efecto tienen en el tiempo de ejecución del algoritmo).

Como ya se ha comentado, los algoritmos genéticos tienen una componente aleatoria que hace posible la obtención de resultados finales distintos al repetir una ejecución con los mismos parámetros iníciales. Es por esto que se han realizado diversas repeticiones para cada conjunto de parámetros, analizando de este modo las variaciones que surgen en las respectivas simulaciones.

Las pruebas se han realizado para la siguiente relación de parámetros (aplicando 5 repeticiones por prueba para poder comprobar si existe mucha variación de una simulación a otra, o si por el contrario los resultados se mantienen más o menos en un rango similar):

- 20 generaciones: 20, 40, 60 y 80 individuos por población.
- 30 generaciones: 20, 40, 60 y 80 individuos por población.
- 40 generaciones: 20, 40, 60 y 80 individuos por población.
- 50 generaciones: 20, 40, 60 y 80 individuos por población.

Nota: Las siguientes simulaciones se han realizado para unas probabilidades de cruce y mutación de 1 y 0.01 respectivamente, y una función de ajuste con forma escalón de 5º y tiempo 3 segundos:

20 generaciones:

Error	K_p	K_i	K_d
49.1074	9.7070	0.3533	9.7710
45.4154	9.7674	0.2489	9.9448
51.1723	9.8663	0.4946	9.9989
42.6237	9.8174	0.0175	9.3860
42.4077	9.8329	0.0124	9.6733

Tabla I.1. 20 generaciones / 20 individuos

Un aspecto a tener en cuenta a partir de estos resultados es el siguiente: La diferencia entre el peor (error=51.1723) y mejor (error=42.4077) individuo de la tabla anterior puede no parecer muy grande, ya que se puede considerar que los parámetros PID obtenidos son bastante similares (salvo la componente integral):

- k_{p_mejor}=9.8329 / k_{p_peor}=9.8663
- k_{i_mejor}=0.0124 / k_{i_peor}=0.4946
- k_{d_mejor}=9.6733 / k_{d_peor}=9.9989

Si analizamos los resultados que se obtienen aplicando estos dos controladores a una trayectoria recta, con un cambio de rumbo puntual de 30º, se puede observar que la diferencia de uno a otro es bastante notable. Las siguientes figuras muestran la comparativa entre los resultados deseados, los obtenidos con el controlador PID clásico, y con el controlador PID optimizado mediante AG para los casos de mejor y peor individuo obtenidos en la tabla anterior:

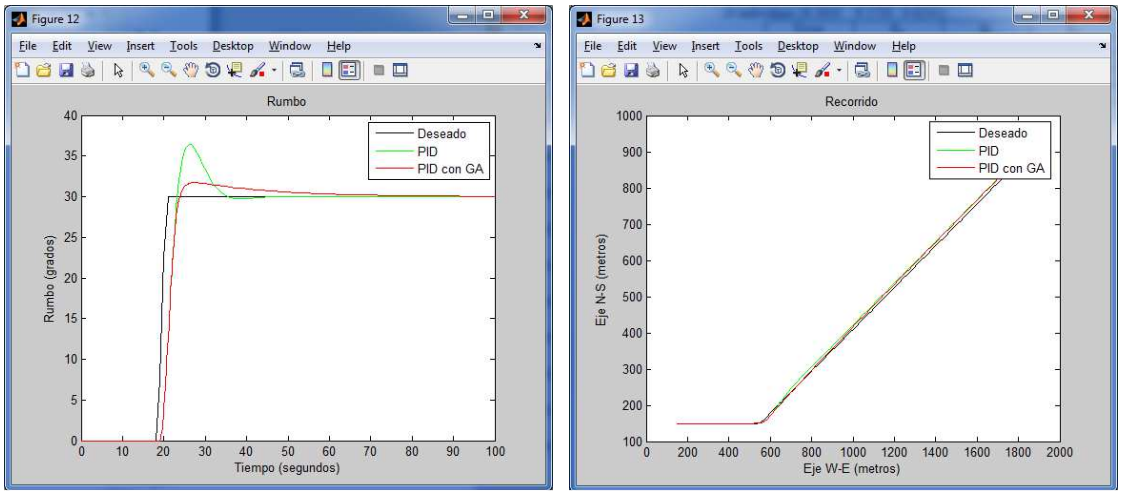

Figura I.1. Resultados para controlador PID optimizado con el peor individuo obtenido de la simulación (Error=51.1723, k_p=9.8663, k_i=0.4946, k_d=9.9989)

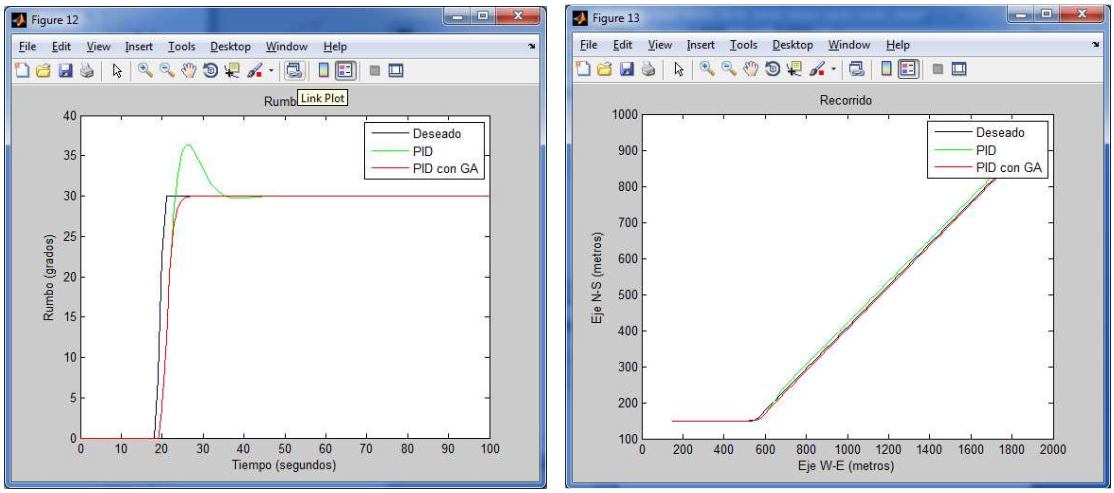

Figura I.2. Resultados para controlador PID optimizado con el mejor individuo obtenido de la simulación (Error=42.4077, k_p=9.8329, k_i=0.0124, k_d=9.6733)

Se observa que la diferencia en los resultados entre los dos controladores es considerable. Al utilizar el peor individuo, tanto el sobrepico como el tiempo para alcanzar un valor estable son mayores. Esto implica, tal y como se puede observar en las gráficas de la derecha, que al utilizar el mejor individuo el barco seguirá un recorrido más parecido al deseado.

Por otro lado, tal y como era de esperar, la variación de los resultados entre las diferentes repeticiones es bastante grande (ver tabla I.1). Esto se debe a que al algoritmo no le da tiempo a converger a un resultado aceptable por utilizar una relación relativamente pequeña de individuos por población y cantidad de generaciones. A medida que aumenten estos parámetros se podrá observar que las diferencias entre las distintas repeticiones irán disminuyendo.

A continuación se muestran los resultados obtenidos para el resto de simulaciones variando el número de individuos por población:

Error	K_p	K_i	K_d
41.2811	9.9791	0.0371	9.8494
44.9526	9.7256	0.2085	9.9194
42.1219	9.9424	0.1089	9.9864
46.1556	9.7369	0.2880	9.9883
43.7116	9.9934	0.2060	9.9419

Tabla I.2. 20 generaciones / 40 individuos

Error	K_p	K_i	K_d
43.4640	9.9750	0.1681	9.8259
41.4997	9.9793	0.0682	9.9172
42.0506	9.9535	0.1105	9.9980
44.6963	9.7721	0.0536	9.1931
43.9419	9.8047	0.1839	9.9930

Tabla I.3. 20 generaciones / 60 individuos

Error	K_p	K_i	K_d
41.4622	9.9713	0.0560	9.8698
42.4509	9.9317	0.1327	9.9994
42.8827	9.9091	0.1284	9.8285
40.8795	9.9983	0.0124	9.6759
41.3722	9.9753	0.0384	9.8823

Tabla I.4. 20 generaciones / 80 individuos

30 generaciones:

Error	K_p	K_i	K_d
41.3655	9.9701	0.0440	9.8012
42.4966	9.9044	0.1175	9.9631
44.7449	9.9287	0.1268	9.4564
41.5323	9.9415	0.0338	9.6752
42.0557	9.8846	0.0393	9.7716

Tabla I.5. 30 generaciones / 20 individuos

Error	K_p	K_i	K_d
43.0836	9.9395	0.1688	9.9543
41.5156	9.9880	0.0445	9.9965
40.9825	9.9997	0.0254	9.7840
40.9760	9.9977	0.0199	9.7890
41.2317	9.9962	0.0487	9.9002

Tabla I.6. 30 generaciones / 40 individuos

Error	K_p	K_i	K_d
40.9544	9.9861	0.0103	9.6567
40.8571	9.9985	0.0108	9.7157
42.3695	9.8247	0.0252	9.5531
41.6685	9.9857	0.0710	9.7620
40.8724	9.9989	0.0132	9.7148

Tabal I.7. 30 generaciones / 60 individuos

Error	K_p	K_i	K_d
41.1150	9.9947	0.0207	9.8466
40.8604	9.9983	0.0108	9.7201
41.0094	9.9805	0.0103	9.7195
40.8937	9.9957	0.0117	9.6691
40.9507	9.9902	0.0114	9.7407

Tabla I.8. 30 generaciones / 80 individuos

40 generaciones:

Error	K_p	K_i	K_d
41.0106	9.9994	0.0129	9.98178
44.3256	9.5892	0.0198	9.2742
41.1498	9.9864	0.0146	9.8207
42.9417	9.9968	0.1752	9.9619
47.3038	9.3756	0.0100	8.7512

Tabla I.9. 40 generaciones / 20 individuos

Error	K_p	K_i	K_d
40.8687	10.0000	0.0109	9.6537
40.9901	9.9988	0.0257	9.7803
40.8634	10.0000	0.0114	9.6651
40.9094	9.9938	0.0105	9.6494
40.9964	9.9907	0.0175	9.6716

Tabla I.10. 40 generaciones / 40 individuos

Error	K_p	K_i	K_d
40.8744	9.9989	0.0123	9.7349
40.8365	10.0000	0.0102	9.6930
40.8529	9.9989	0.0108	9.7143
40.8687	9.9973	0.0108	9.7191
40.8388	9.9998	0.0103	9.6965

Tabla I.11. 40 generaciones / 60 individuos

Error	K_p	K_i	K_d
40.8658	9.9998	0.0105	9.7448
40.9391	9.9997	0.0199	9.7697
40.8972	9.9925	0.0100	9.7087
40.8412	9.9997	0.0103	9.6878
40.8773	9.9986	0.0134	9.7204

Tabla I.12. 40 generaciones / 80 individuos

50 generaciones:

Error	K_p	K_i	K_d
41.3583	9.9579	0.0274	9.7851
40.8430	9.9995	0.0100	9.6770
41.6216	9.9050	0.0143	9.6335
42.1694	9.9748	0.0505	9.5259
41.5163	9.9212	0.0129	9.6854

Tabla I.13. 50 generaciones / 20 individuos

Error	K_p	K_i	K_d
40.8406	9.9998	0.0105	9.7044
40.8587	10.0000	0.0126	9.7003
40.8349	10.0000	0.0100	9.6953
40.8420	9.9998	0.0106	9.6933
40.8505	9.9998	0.0114	9.7157

Tabla I.14. 50 generaciones / 40 individuos

Error	K_p	K_i	K_d
40.8349	10.0000	0.0100	9.6948
40.8559	10.0000	0.0115	9.7277
40.8826	9.9992	0.0144	9.7268
40.8361	10.0000	0.0100	9.6870
40.8355	10.0000	0.0100	9.7044

Tabla I.15. 50 generaciones / 60 individuos

Error	K_p	K_i	K_d
40.8803	9.9997	0.0124	9.7485
40.8349	10.0000	0.0100	9.6994
40.8708	9.9985	0.0121	9.7241
40.8349	10.0000	0.0100	9.6963
40.8507	9.9998	0.0111	9.6803

Tabla I.16. 50 generaciones / 80 individuos

A partir de los datos de las tablas anteriores, te observa claramente que a medida que aumenta el nº de individuos y generaciones, todas las repeticiones realizadas tienden a converger a los mismos valores (aproximadamente: Error=40.8, k_p=10, k_i=0.01, k_d=9.7).

Además, se ve que a medida que aumenta la relación entre generación e individuos por población, la diferencia entre las soluciones que aporta el algoritmo al ejecutarlo de forma repetitiva son prácticamente despreciables (en la tabla I.16., la diferencia entre en mejor y peor individuo es de aproximadamente el 0.1%).

Análisis del tiempo de cómputo:

Haciendo uso de las funciones *tic* y *toc* de Matlab, se ha podido calcular el tiempo de simulación para los diferentes tipos de pruebas. Como era de esperar, el tiempo de simulación está principalmente condicionado por la cantidad de individuos sobre los que se deben realizar las diversas operaciones del algoritmo. Se ha calculado que de forma aproximada, la simulación necesitará 0.02 segundos por cada individuo que se vaya a procesar. Esto implica que conseguiremos una relación más o menos lineal entre el tiempo de simulación y la cantidad de individuos.

La siguiente gráfica muestra la relación del tiempo de simulación para cada uno de los casos anteriormente analizados:

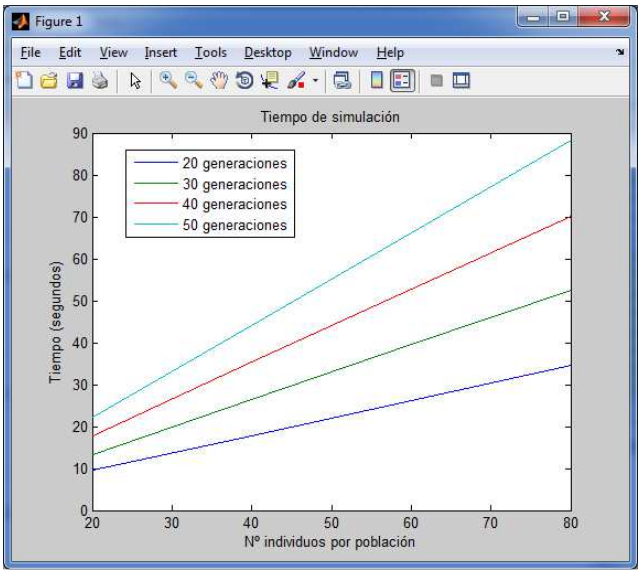

Figura I.3. Tiempos de simulación

A la hora de determinar qué tiempo de simulación se considera adecuado, es necesario fijar también el error de la solución que se considera aceptable (así como la variación que se puede obtener en las diferentes repeticiones al ejecutar del algoritmo). En este caso concreto, se desea conseguir un controlador que minimice al máximo el sobrepico, y se ha observado que eso se consigue en los resultados que dan un error inferior a un valor de 42 en el error ITAE de la función objetivo.

De este modo, analizando los resultados de las tablas mostradas anteriormente, las alternativas que se considerará que ofrecen resultados aceptables en sus diferentes repeticiones son las siguientes:

- 50 generaciones / 40 individuos (o más)
- 40 generaciones / 40 individuos (o más)
- 30 generaciones / 80 individuos

De estas alternativas, la que implica un menor tiempo de simulación sería la opción de un algoritmo con 40 generaciones y 40 individuos por población (con un tiempo aproximado de 35 segundos).

Convergencia al mejor individuo

En relación al apartado anterior, el código del algoritmo genético se ha programado para que genere unas gráficas donde muestra la variación a lo largo de las generaciones tanto del error del mejor individuo, como del error medio de toda la población.

Esta representación sirve para tener una idea acerca de cómo converge el algoritmo hacia el mejor individuo con el paso de las diferentes iteraciones realizadas en la simulación:

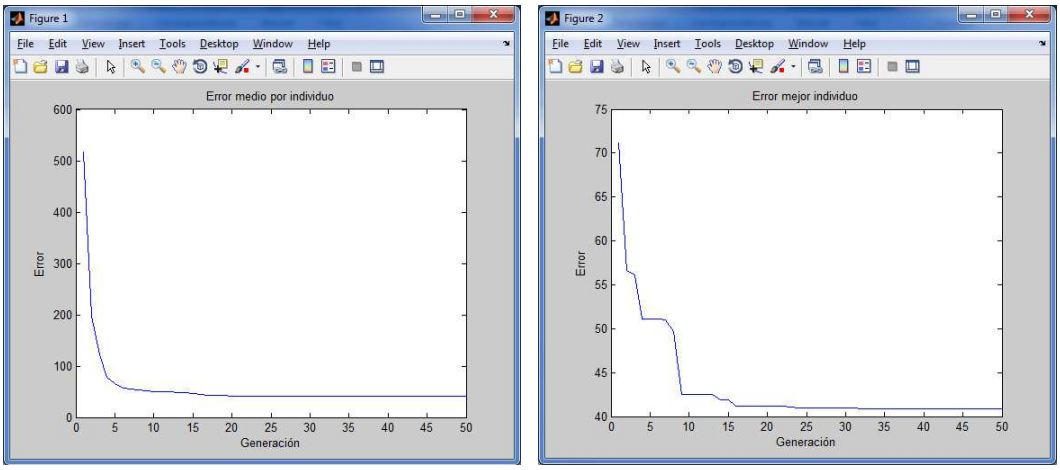

Figura I.4. Evolución para población de 80 individuos

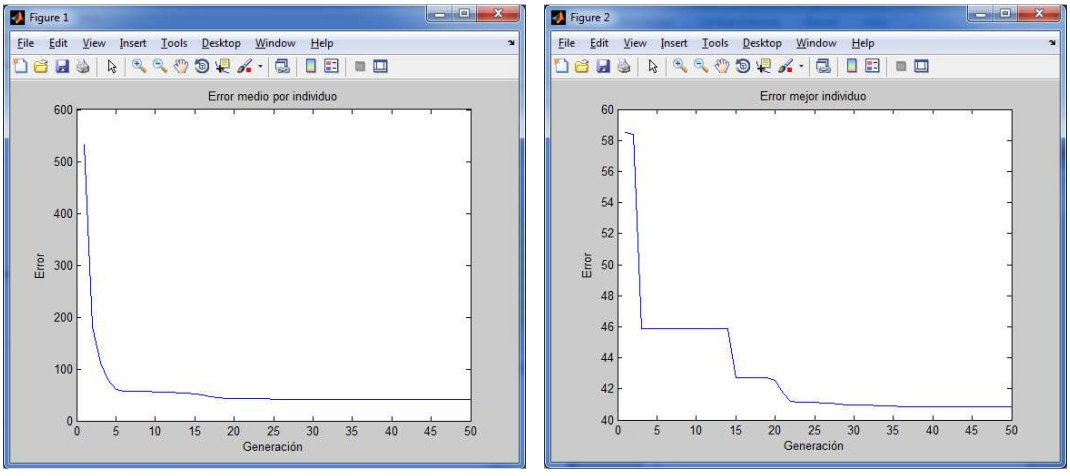

Figura I.5. Evolución para población de 60 individuos

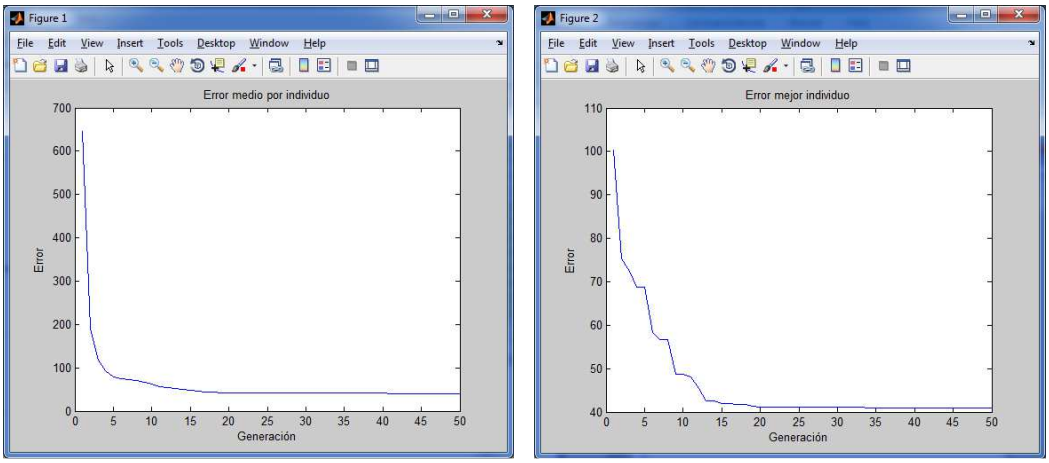

Figura I.6. Evolución para población de 40 individuos

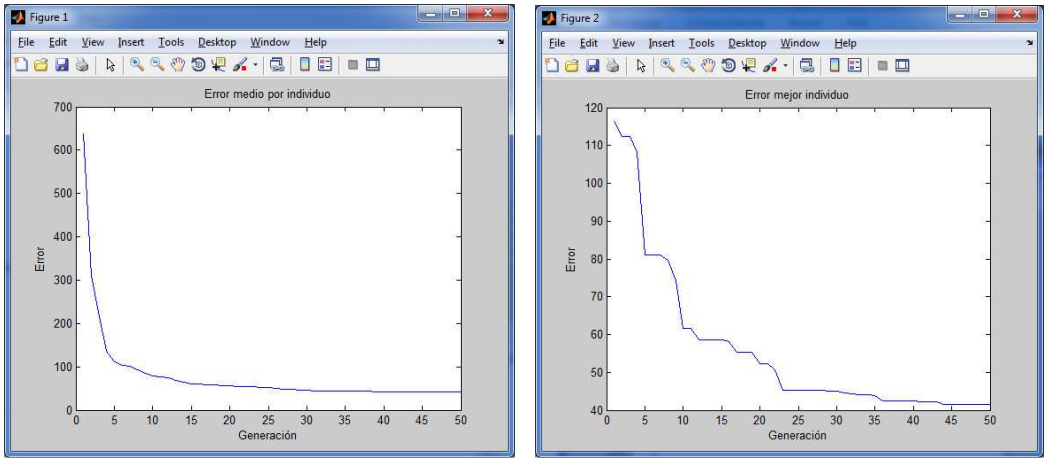

Figura I.7. Evolución para población de 20 individuos

Las gráficas del error del mejor individuo son bastante aleatorias, ya que dependen de cada caso concreto de simulación. Sin embargo, la analizar la evolución del error medio por individuo, se observa que en todos los casos se sigue una línea similar.

Como es lógico, a medida que aumenta el nº de individuos de una población, la facilidad para obtener mejores individuos (soluciones con menor error) es mayor, y por lo tanto el algoritmo tiende a converger más rápido a una solución buena.

Aplicando un zoom a las gráficas anteriores, se puede observar la diferencia entre el caso de una población de 20 individuos, y una población de 60. Se puede comprobar que en el caso de una población mayor (grafica de la derecha), se llega antes a alcanzar los valores aceptables, y por lo tanto en las últimas iteraciones del bucle la curva tiende a mantenerse más constante:

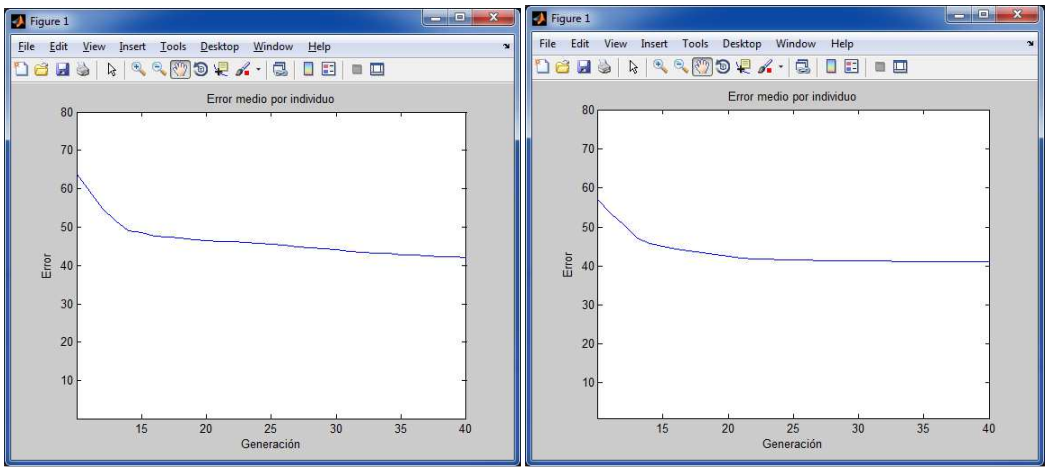

Figura I.8. Error medio en las últimas 30 iteraciones para poblaciones de 20 (derecha) y 60 (izquierda) individuos

Efecto probabilidades de cruce y mutación

El programa permite modificar los valores aplicables a las probabilidades de cruce y mutación. Mientras que el operador de mutación tiene como objetivo aumentar la probabilidad de exploración del espacio de búsqueda, disminuyendo el riesgo de estancamiento del algoritmo en óptimos locales, el operador de cruce permite modificar la cantidad de nuevos individuos que se irán creando en las sucesivas iteraciones del algoritmo.

A partir de los datos fijados en el anterior apartado se han realizado una serie de simulaciones para analizar el efecto de las probabilidades de cruce y mutación:

Error	K_p	K_i	K_d
40.8687	10.0000	0.0109	9.6537
40.9901	9.9988	0.0257	9.7803
40.8634	10.0000	0.0114	9.6651
41.0171	9.9925	0.0106	9.5899
40.9278	9.9980	0.0185	9.7181

Tabla I.17. P_{cruce}=1 y $P_{mutación}$=0.01

Error	K_p	K_i	K_d
40.9012	9.9988	0.0100	9.7642
41.3400	9.9979	0.0589	9.9733
40.9270	9.9998	0.0137	9.6393
41.5436	9.9957	0.0893	9.9930
41.7828	9.9852	0.1030	9.9610

Tabla I.18. P_{cruce}=0.75 y $P_{mutación}$=0.01

Error	K_p	K_i	K_d
41.2239	9.9585	0.0121	9.7261
42.0911	9.9421	0.1031	9.9460
41.6294	9.9200	0.0237	9.7348
43.8843	9.7791	0.0643	9.3713
42.3946	9.8341	0.0335	9.6838

Tabla I.19. P_{cruce}=0.5 y $P_{mutación}$=0.01

Error	K_p	K_i	K_d
41.2689	9.9808	0.0411	9.8332
40.9823	9.9950	0.0201	9.7718
40.9094	9.9938	0.0105	9.6494
41.0532	9.9739	0.0115	9.6645

Tabla I.20. P_{cruce}=1 y $P_{mutación}$=0.1

Error	K_p	K_i	K_d
46.8474	9.3936	0.0949	9.5980
45.3863	9.6279	0.0449	9.0954
44.3963	9.8131	0.0809	9.3419
45.5568	9.6250	0.0914	9.2759
46.8300	9.9669	0.1827	9.4114

Tabla I.21. P_{cruce}=1 y $P_{mutación}$=0.3

Error	K_p	K_i	K_d
60.3648	8.2444	0.2884	9.4855
53.5260	9.0645	0.1350	8.5419
60.3694	9.2913	0.7239	9.7683
48.7657	9.5282	0.3498	9.8835
51.5816	8.9271	0.0728	8.6271

Tabla I.22. P_{cruce}=1 y $P_{mutación}$=0.5

Error	K_p	K_i	K_d
40.9099	9.9947	0.0137	9.6973
40.9204	9.9973	0.0172	9.7245
41.0815	9.9924	0.0268	9.6934
41.0482	9.9960	0.0297	9.7415

Tabla I.23. P_{cruce}=0.9 y $P_{mutación}$=0.05

Error	K_p	K_i	K_d
40.8686	9.9992	0.0127	9.7232
40.9584	9.9956	0.0199	9.7386
40.9241	9.9994	0.0188	9.7526
40.9443	9.9977	0.0179	9.6822

Tabla I.24. P_{cruce}=1 y $P_{mutación}$=0.05

El hecho de almacenar en un vector el mejor individuo de cada población, y realizar la inserción en cada iteración utilizando únicamente los mejores individuos entre los padres y los hijos creados, minimiza el posible efecto negativo que tendría utilizar valores elevados de probabilidades de mutación. Si se utilizase un método de reinserción en el que la nueva generación se formaría a partir de los hijos creados mediante la recombinación de los padres de la generación anterior, probabilidades de mutación elevadas tenderían a modificar demasiado los individuos, haciendo que el algoritmo tardase más en obtener resultados aceptables.