UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

ESCUELA DE INGENIERIA ELECTRONICA

TEMA:

DISEÑO DE UN CONTROLADOR DIFUSO Y GENETICO PARA EL CONTROL DE DIRECCION DEL B.A.P. "CARVAJAL" DE LA MARINA DE GUERRA DEL PERU

Tesis

Para Optar el Titulo Profesional de

INGENIERO ELECTRONICO

Luis Campusano Quispe

LIMA – PERU

2007

Dedico éste trabajo a la memoria de mi Padre Rosas, que su espíritu, su bondad y su sentido del humor siga tocando mi vida.

A mi Madre y hermanos, quienes con intuición y generosidad me apoyaron siempre.

A mi esposa Lissete, y a mis amigos, fuente constante de motivación.

A mis profesores, porque ellos sembraron la semilla y fertilizaron mi inquietud de conocer.

AGRADECIMIENTOS

Agradezco a Dios nuestro señor por la oportunidad que he tenido de aprender, mejorar y de crecer junto a personas tan especiales para mí.

Agradecimiento especial para mi Profesor Asesor Ing. Julio Borjas Castañeda, por su amistad, paciencia y su constante apoyo durante el desarrollo de esta tesis. De igual forma deseo expresar mi agradecimiento al Jurado Calificador de esta tesis presidida Ing. Pedro Custodio Diez, Ing. Julio Casquero Zaidman , Ing. Wilbert Chavez Irazabal y el Ing. Julio Borjas Castañeda para la culminación de esta tesis.

Al personal del Centro de Investigación y Desarrollo (I & D) de la Marina de Guerra del Perú por su amistad y cooperación.

A mi familia por tener la paciencia de esperarme con tanta vehemencia.

A mis compañeros y amigos por compartir las angustias y gratificaciones, a todos ellos gracias.

Con mucho cariño, humildemente Luis Campusano Quispe

RESUMEN

La investigación se ha dividido en dos partes, la primera de generalidades y la segunda de diseño y simulación. En la parte de generalidades se revisan los puntos necesarios para comprender los temas de lógica difusa, y controlador genético. En la parte de diseño y simulación se realiza el diseño del controlador difuso derivativo para el control de dirección del un buque para este modelo y se simulan sus resultados, posteriormente se diseña el controlador genético proporcional derivativo, mediante controlador genético adaptativo con modelo de referencia, para ser aplicado al control de dirección de un buque.

Al final se llega a las conclusiones de que el controlador difuso es mucho rápido en el procesamiento pero menos exacto. El controlador genético es mucho mas lento en el procesamiento pero es mas exacto en su repuesta. Esto se demuestra en la simulación que se ha realizado con el software de programación MATLAB, el programa de simulación se encuentra en el apéndice.

INDICE

INDICE DE FIGURAS

PRIMERA PARTE

SEGUNDA PARTE

INDICE DE TABLAS

INTRODUCCION

En este trabajo, el controlador Difuso y controlador Genético, son diseñados para observar y analizar quien tiene mejor respuesta, para el control de dirección del B.A.P "CARVAJAL" basándonos en su modelo matemático no lineal.

Este tipo de controladores inteligentes evitan el tedioso cálculo que se tendría si se utilizara técnicas de control clásicas o avanzadas, obteniendo de esa forma un controlador inteligente y adaptable.

El contenido del escrito se ha dividido en dos partes, la primera de generalidades y la segunda de diseño y simulación. En la parte de generalidades se revisan los puntos necesarios para comprender los temas de lógica difusa, y controlador genético. Por otro lado, en la parte de diseño y simulación se realiza el diseño del controlador difuso derivativo para el control de dirección de un buque para este modelo y se simulan sus resultados, posteriormente se diseña el controlador genético proporcional derivativo, mediante el controlador genético adaptativo con modelo de referencia, para luego ser aplicado al control de dirección de un buque.

En la parte de simulación se presenta los resultados de la misma y en el apéndice figuran todos los programas realizados en Matlab utilizados en las simulaciones, de esta manera, se establece un nuevo procedimiento para el diseño de controladores inteligentes

PLANEAMIENTO DE LA INVESTIGACION

I.- Identificación del Problema

Muchas ventajas que son inherentes del control inteligente pueden ser atribuidas a su habilidad para permitir a los diseñadores fácilmente incorporar conocimiento heurísticos de como controlar mejor un sistema en un controlador. Este conocimiento podría venir de un operador humano que manualmente ha realizado la tarea de control, o de un ingeniero de control que ha hecho análisis matemático, basado en simulaciones, o experimentos de la planta y eligiendo el mejor controlador

Los sistemas de control difuso se encuentra en un creciente desarrollo especialmente en la industria, campo en el que existe desde hace tiempo multitud de aplicaciones de estos sistemas en funcionamiento, desplazando poco a poco a las técnicas de control convencionales. Los algoritmos genéticos por su parte son sistemas evolutivos que parten de la teoría darwiniana acerca de la evolución de la especie, donde se encuentra la mayor parte de su aplicación es en las ciencias computacionales, pero desde hace un tiempo también en el control de procesos industriales.

La presente investigación propone el diseño y simulación de estos algoritmos para poder controlar la dirección del B.A.P "Carvajal" del cual se ha obtenido el modelado no lineal del sistema.

II.- Descripción del Problema

El modelo matemático del buque es no lineal con variaciones considerables de los parámetros para distintas maniobras del cambio del rumbo, e incluye las limitaciones del ángulo y velocidad del timón. Estos comportamiento no lineal hace difícil el ajuste de los parámetros del controlador para todo el rango de posibles maniobras de cambio de rumbo. Es por ello que se diseñan estos tipos de controladores inteligentes, los resultados obtenidos se comprueban con los controladores convencionales clásicos, evidenciando así la ventaja de los controladores inteligente en el control de procesos no lineales. El objetivo del diseño es que el buque realice la maniobra de cambio de rumbo de forma rápida pero sin oscilaciones, simulando al buque a plena carga .

III.- Hipótesis

Es posible diseñar un sistema de control difuso, y genético para el control de dirección del BAP "Carvajal"

IV.- Objetivos

a.- Objetivo General

Controlar la dirección de rumbo del B.A.P "Carvajal" usando controladores difusos, y genéticos.

b.- Objetivos Específicos

- Proponer a la Marina de Guerra el uso de estos Controladores
- Demostrar que los controladores inteligentes o técnicas del Soft computing poseen mayor rendimiento que los controladores convencionales en los sistemas no lineales

V.- Metodología

Para el desarrollo de la presente investigación de utilizara el software de Fuzzytech y Matlab V7.0, trabajando sobre los Toolbox de Fuzzy Logic y el Toolbox de Algoritmos genéticos cabe notar que toda la simulación del sistema se hará bajo esta plataforma

Variable de Estudio

- Variables Independientes

  Angulo del timón del buque y dirección del buque

- Variables dependientes

  Definición de las reglas del controlador difuso, modelo de referencia del controlador adaptivo, modelado no lineal del buque, universo del algoritmo genético.

GENERALIDADES

PRIMERA PARTE:

1. LOGICA DIFUSA

La lógica difusa es una lógica multivalorada capaz de capturar informaciones vagas, en general descritas en un lenguaje natural y convertirlas en un formato numérico, de fácil manipulación para las computadoras de hoy en día. La lógica difusa puede ser definida como una lógica que soporta modos de raciocinio que son aproximados, en vez que exactos, como estamos acostumbrados a trabajar.

Los sistemas basados en lógica difusa tienen potencia en el modelamiento matemático no lineal mediante el uso de variables lingüísticas y una serie de condiciones previamente definidas, pero tiene inexactitud en los términos de salida, si se busca una aproximación muy buena se debe usar la lógica difusa.

1.1. Fundamentos De La Lógica Difusa

La lógica difusa asocia incertidumbre a la estructura de un conjunto de datos. Los elementos de un conjunto difuso son pares ordenados que indican el valor del elemento y su grado de pertenencia.

Para un conjunto difuso, $A = \{(x, u_A(x)) / x \in X\}$ se tiene que el elemento x pertenece al conjunto A con un grado de pertenencia $u_A(x)$, que puede variar entre 0 y 1. Por lo tanto, una variable puede ser caracterizada por diferentes valores lingüísticos, cada uno de los cuales representa un conjunto difuso. Por ejemplo, la velocidad puede ser caracterizada por valores lingüísticos como "Bajo", "Medio" y "Alto", que representan "una velocidad aproximadamente menor que 40 km/h", "una velocidad cercana a 55 km/h" y "una velocidad sobre 70 km/h aprox." respectivamente. Estos términos se asocian a conjuntos difusos con funciones de pertenencia como las mostradas en siguiente figura 1.1.

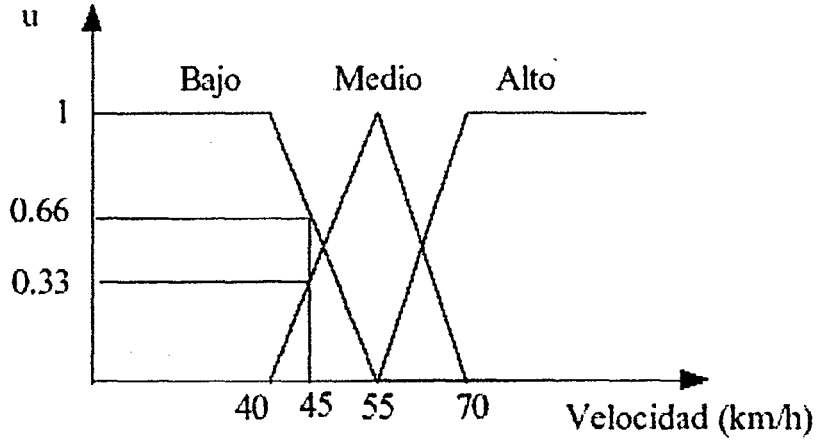

Figura 1.1 Función de Membresía para la Velocidad

Por lo tanto, si la velocidad es 45 km/h, existen grados de pertenencia 0.6, 0.3 y 0 a los conjuntos difusos "Bajo", "Medio" y "Alto" respectivamente.

1.2. Operaciones Básicas de Lógica Difusa

Dados dos conjuntos difusos A y B en el mismo universo X, con funciones de pertenencia uA y uB respectivamente, se pueden definir las siguientes operaciones básicas:

1.2.1 Unión. La función de pertenencia de la unión de A y B se define como:

$$u_{A\cup B} = \max\{(u_A(x), u_B(x))\}$$

1.2.2 Intersección. La función de pertenencia de la intersección de A y B es:

$$u_{A\cap B} = \min\{(u_A(x), u_B(x))\}$$

1.2.3 Complemento. La función de pertenencia del complemento de A se define como:

$$u_{\bar{A}}(x) = 1 - u_A(x)$$

1.2.4 Producto cartesiano. Dados los conjuntos difusos A1,..., An con universos X1,..., Xn respectivamente, se define el producto cartesiano como un conjunto difuso en X1,...,Xn con la siguiente función de pertenencia:

$$u_{A1....An}(x_1,..., x_n) = \min\{u_{A1}(x_1),..., u_{An}(x_n)\}$$

Según Mamdani (1974)

$$u_{A1....An}(x_1,..., x_n) = u_{A1}(x_1),..., u_{An}(x_n)$$

Según Larsen (1980).

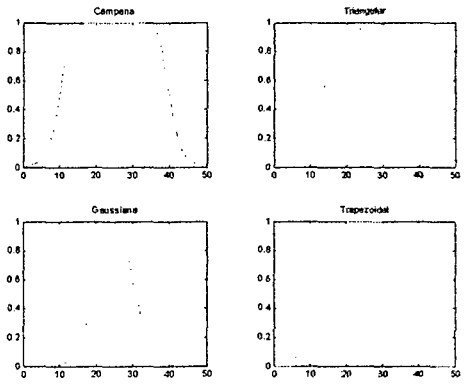

Figura 1.2. Operaciones Difusas

1.3. Razonamiento Difuso

Nosotros necesitamos reglas de inferencia para el funcionamiento de un razonamiento difuso. Las reglas de inferencia usadas en el razonamiento difuso son llamadas "reglas difusas IF-THEN."

$$u_{A1....An}(x_1,...,x_n) = \min\{u_{A1}(x_1),...,u_{An}(x_n)\}$$

$$u_{A1....An}(x_1,...,x_n) = u_{A1}(x_1),...,u_{An}(x_n) \qquad A = \{(x, u_A(x)) / x \in X\}$$

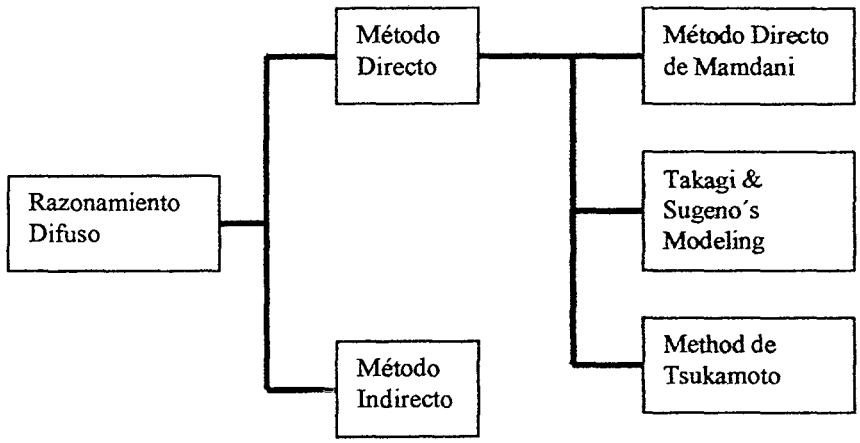

Figura1.3. Clasificación del Razonamiento Difuso

El más popular de los métodos directos es el propuesto por Mamdani. El método directo de Mamdani tiene una estructura simple de operación min-max y es muy popular en las aplicaciones .

Demos una primera mirada al **Método Directo de Mamdani**:

IF x is A and y is B Then z is C

Donde A, B y C son conjuntos difusos, If es llamado como premisa (x y, son llamadas como variable de premisa) y Then es llamado consecuencia (variables de la consecuencia).

Como por ejemplo:

IF	la temperatura de la habitación es "un poco alta"
AND	la humedad es "ligeramente alta"
THEN	incrementar el aire acondicionado a "alto."

Entonces encontrando la correspondencia:

X: temperatura de la habitación

Y: humedad

Z: configuración del aire acondicionado

A: poca alta

B: ligeramente alta

C: alta

Donde A, B y C son los conjuntos difusos. La temperatura de la habitación es medida en Celsius, la humedad en porcentaje y nosotros asumimos que la configuración del aire acondicionado es de 0 a 10.

Si reemplazamos los conjuntos difusos por números difusos entonces redescribiríamos de la siguiente manera:

IF	x es "aproximadamente 20 grados"
AND	y es "aproximadamente 5 %"
THEN	z es "aproximadamente 8"

Ahora demos una mirada al método de **Takagi & Sugeno's Modeling**, esto sistemas de inferencia utilizan funciones lineales en la parte de la consecuencia:

IF x is A and y is B THEN $z = ax + by + c$

Donde A, B y C, son los parámetros de las funciones lineales de la consecuencia y z es la ecuación lineal de la consecuencia.

IF	x es "aproximadamente 20 grados"
AND	y es "aproximadamente 80 %"
THEN	la configuración del aire acondicionado es:
	= temperatura de la habitación * 0.2 + humedad * 0.05.

Redescribiendo:

IF	x es "aproximadamente 20 grados"
AND	y es "aproximadamente 80 %"
THEN	la configuración del aire acondicionado es:
	$z = 0.2x + 0.05y$.

En general es difícil determinar las ecuaciones lineales para la parte de la consecuencia "empíricamente". En este tipo de razonamiento, nosotros asumimos la aplicación de técnicas de modelamiento usando la data de entrada-salida para obtener reglas.

Otro razonamiento difuso usado es el **Método Tsukamoto** o el de **Consecuencia Simple**, que se describe de la siguiente manera:

IF x is A and y is B THEN z = c

Donde C es un valor real. El valor real puede ser considerado como un conjunto especial difuso sin vaguedad, y a veces llamado "difuso singleton". Es un punto de vista diferente, solo el término constante existe en la ecuación lineal para el método simplificado. Este método siguiendo el ejemplo anterior puede ser descrito como:

IF	la temperatura de la habitación es "un poco alta"
AND	la humedad es "ligeramente alta"
THEN	el aire acondicionado = 8

Este método es menos usado por ser menos transparente.

1.4 Mecanismo de Razonamiento Difuso

En esta parte vamos a describir el esqueleto del mecanismo de razonamiento difuso. En la lógica convencional el razonamiento es basado en: "modus ponens" (deducción) y "modus tollens" (inducción), estas dos son complementarias.

Modus Ponens:

Premisa 1: A $\longrightarrow$ B

Premisa 2: A

Consecuencia: B

Premisa 1: IF x es A $\longrightarrow$ B

Premisa 2: x es A

Consecuencia: y es B

Modus Tollens:

Premisa 1: A $\longrightarrow$ B

Premisa 2: not A

Consecuencia: not B

En la práctica el Modus Ponens (GMP), seria de la siguiente manera:

Premisa 1: IF la temperatura es menor a 10°C THEN pon mas alto el calentador

Premisa 2: La temperatura es 5°C

Consecuencia: Poner mas alto el Calentador

1.4.1 Método Directo de Mamdani's

Mecanismo de Razonamiento

Para un primer vistazo al mecanismo de inferencia, realizaremos un ejemplo simple con dos entradas y una salida

Regla 1: IF x is A_1 and y is B_1 Then z is C_1

Regla 2: IF x is A_2 and y is B_2 Then z is C_2

Donde A_1, A_2, B_1, B_2, C_1 y C_2 son los conjuntos difusos, la figura 1.4 muestra el proceso de razonamiento.

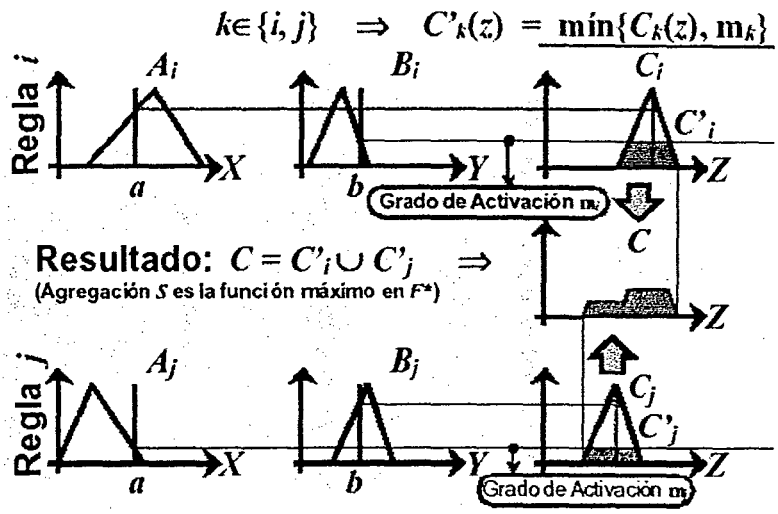

Figura 1.4. Proceso de Razonamiento por el Método de Mandami

Ahora suponemos x_0 e y_0, van a ser las entradas de las variables lingüísticas x e y respectivamente. El proceso de razonamiento es el siguiente:

Paso 1: Medir la adaptabilidad de la premisas de reglas para una entrada dada

Medir la adaptabilidad de cada regla para la entrada (x_0, y_0) de la siguiente manera:

Adaptabilidad de la regla 1: $W_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0)$

Adaptabilidad de la regla 2: $W_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0)$

Utilizamos el operador mínimo representado por $\wedge$.

Paso 2: Para la adaptabilidad de cada regla precede la conclusión de cada regla.

Conclusión de la regla 1: $\mu_{c1}(x_0) = W_1 \wedge \mu_{c1}(z)$

Conclusión de la regla 2: $\mu_{c2}(x_0) = W_2 \wedge \mu_{c2}(z)$

También usa el operador mínimo, pero diferentes implicaciones.

Paso 3: Agrega una conclusión individual para obtener una conclusión conjunta.

Conclusión final: $\mu_c(z) = \mu_{c1}(z) \wedge \mu_{c2}(z)$

En el paso 3 obtenemos la salida de un proceso de sistema difuso, la salida es un numero difuso, pero para la implementación de sistemas de control de procesos necesitamos un valor numérico natural y no difuso entonces, a esta operación se denomina "defuzzificación".

1.5. Métodos de Defuzzificación:

1.5.1. Defuzzificación de Máxima Membresía: Conocido como el método de altura (Fig 1.5), la salida crisp es la máxima altura del conjunto difuso final.

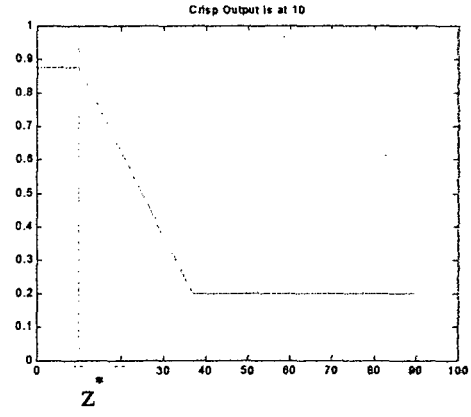

Figura 1.5 Defuzzificación de Máxima Membresía

1.5.2. **Defuzzificación por Centro de Área:** También llamado método de la centroide o de centro de gravedad (Fig. 1.6) , es el método mas usado y mas potente.

$$z^* = \frac{\int \mu_z z dz}{\int \mu_z}$$

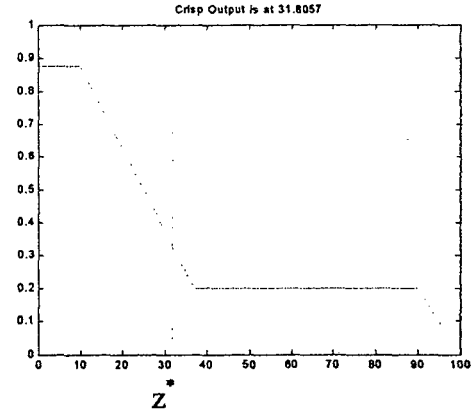

Figura 1.6. Defuzzificación por Centro de Área

1.5.3. **Defuzzificación por Máximo Promedio Eficaz:** Como su nombre lo indica utiliza el valor de la media eficaz del máximo, (Fig. 1.7).

$$z^* = \frac{\sum \mu_z z}{\sum \mu_z}$$

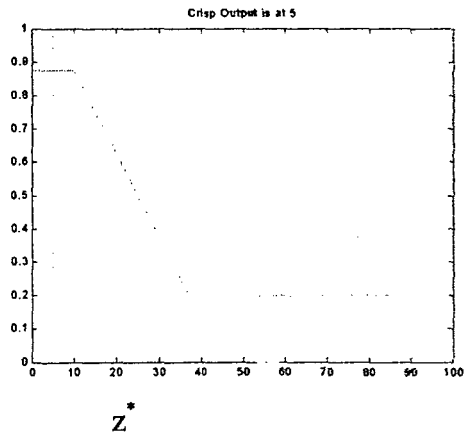

Figura 1.7 Defuzzificación por Máximo Promedio Eficaz

1.5.4. **Defuzzificación por Máximo del Medio:** Igual que el anterior, pero calcula la media de los máximos, (Fig. 1.8).

$$z^* = \frac{a+b}{2}$$

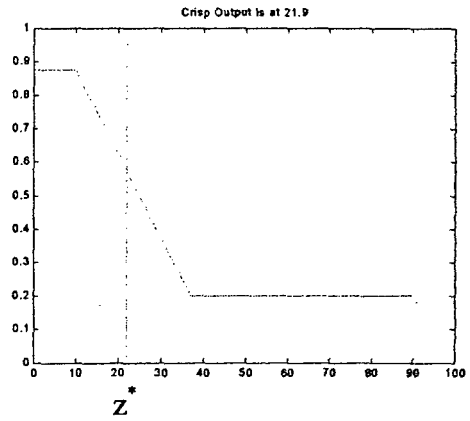

Figura 1.8. Defuzzificación por Máximo del Medio

1.5.5. **Defuzzificación Primer o Ultimo Máximo:** Este método toma el primer máximo o el ultimo máximo del a salida, (Fig. 1.9).

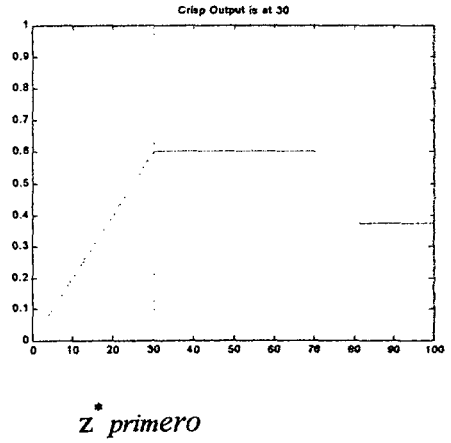

Figura 1.9 Defuzzificación del Primer o Ultimo Máximo

2. ALGORITMOS GENETICOS

Los Algoritmos Genéticos son procedimientos adaptativos para la búsqueda de soluciones en espacios complejos inspirados en la evolución biológica_ con patrones de operaciones basados en el principio darwiniano de reproducción y supervivencia de los individuos que mejor se adaptan al entorno en el que viven.

En este capitulo se presenta un estudio sobre los Algoritmos Genéticos que incluye los fundamentos básicos de los mismos, algunas de las extensiones que se han presentado a lo largo de las dos décadas de desarrollo de los mismos, aplicaciones en distintos campos y una introducción a otros

Algoritmos de Evolución que junto con los Algoritmos Genéticos confluyen en lo que hoy se denomina Computación Evolutiva .

2.1. Introducción

Un sistema biológico eficiente desarrolla estrategias exitosas de adaptación para lograr su supervivencia y propagación. La teoría de la evolución postula que las mejores en una especie se logran por la evolución de las mismas a lo largo de cientos y miles de generaciones, donde los individuos que sobreviven son los más aptos y los menos aptos desaparecen.

Un algoritmo genético, en adelante AG, simula la evolución de una población de individuos, mediante un proceso iterativo aplicado sobre un conjunto de estructuras. Cada estructura esta compuesta de características que definen la aptitud del individuo en un entorno. La población de estructuras evoluciona de generación en generación mediante la recombinación de sus integrantes, la mutación de algunas características elegidas al azar, y la selección de aquellas estructuras que se mostraron mas aptas. El material genético, sobrevive a cada generación, combinándose y ampliando su presencia en la población, en la medida en que las estructuras que lo contienen manifiesten aptitud relativamente buena.

Desde el punto de vista de la resolución de problemas, un individuo representa una solución posible a un problema dado. La aptitud del mismo es una medida de cuan buena es la solución para dicho problema. El objetivo del AG es entonces buscar una "buena" solución al problema.

2.1.1. Definición

Previo a la aplicación de un AG para resolver un problema, es necesario definir dos elementos: la representación de las soluciones candidatas en individuos de una población, y la construcción de la función que medirá la aptitud de cada uno de los individuos

La generación de la población inicial se realiza habitualmente creando individuos en forma aleatoria. Luego, cada generación se crea a partir de la generación anterior tras la aplicación de tres operadores básicos: selección, cruza y mutación. El proceso iterativo prosigue hasta que el algoritmo cumple con la condición de parada. Esta condición puede ser: algún individuo presenta una aptitud suficientemente buena, se alcanza una cota máxima de evaluaciones o generaciones, toda la población esta dominada por el mismo individuo, o se cumple alguna condición específica del problema.

Los operadores considerados fundamentales para un AG son la selección y la cruza. La mutación habitualmente es considerada un operador secundario, si bien cumple un rol necesario para la diversificación de la población.

La selección es el proceso por el cual sobreviven los mejores individuos de acuerdo a su aptitud. Esta aptitud esta representada por una función f que, intuitivamente, se puede pensar como alguna medida de utilidad o beneficio que se pretende maximizar. Este operador se implementa asignando a cada individuo una cantidad de copias de si mismo acorde a su aptitud de modo que n individuo con buena aptitud tendrá más probabilidad de participar de la cruza que uno poco apto.

"La ***selección*** provee la abstracción del mecanismo de selección natural".

Algunos de los individuos seleccionados sobreviven directamente en la próxima generación, y otros participan de la cruza contribuyendo en forma indirecta con el aporte de su material genético. La proporción de la población que es reemplazada en cada generación se denomina ***salto generacional***.

La cruza toma dos individuos sobrevivientes, recombina alguna de sus características – material genético – y produce dos nuevos individuos que pertenecen a la próxima generación (La cruza tradicional produce dos hijos, sin embargo no es la única alternativa. Otra posibilidad utilizada habitualmente consiste en generar un único hijo, o inclusive más).

"La ***cruza*** abstrae la reproducción sexual en los sistemas naturales".

La mezcla de código genético provista por la cruza brinda un método de búsqueda altamente eficiente en el espacio de estructuras. No se trata de un proceso totalmente estocástico ya que la selección impone un sesgo en la búsqueda que realiza la cruza.

Esto ocurre porque los mejores individuos tienden a dominar la población. Para garantizar que ninguna característica se pierda irremediablemente de la población se utiliza el operador de mutación. La mutación toma unos pocos individuos generados por la cruza y altera algunas características tomadas al azar.

La mutación altera aleatoriamenté una o más características de un individuo existente.

La mutación no solo evita que se pierdan alelos, sino que también contribuye a la creación de nuevos individuos.

El ciclo de ejecución básico de un AG es simple. Este comienza con la generación aleatoria de una población inicial de estructuras. Luego se evalúa la aptitud de cada una aplicando la función f definida a tal fin. Si alguna estructura, o la evolución misma, cumple con la condición de parada, el algoritmo termina y el resultado es la mejor estructura presente en la población.

De lo contrario, se seleccionan las estructuras mas aptas asignando a estas una cantidad de copias proporcional a su aptitud. Parte de esta población intermedia se cruza para generara nuevas estructuras que heredan el material genético de sus padres.

Por ultimo, algunas características de la población son elegidas para mutar, alterando así las estructuras que lo contienen. La población recién creada es evaluada y el ciclo vuelve a comenzar.

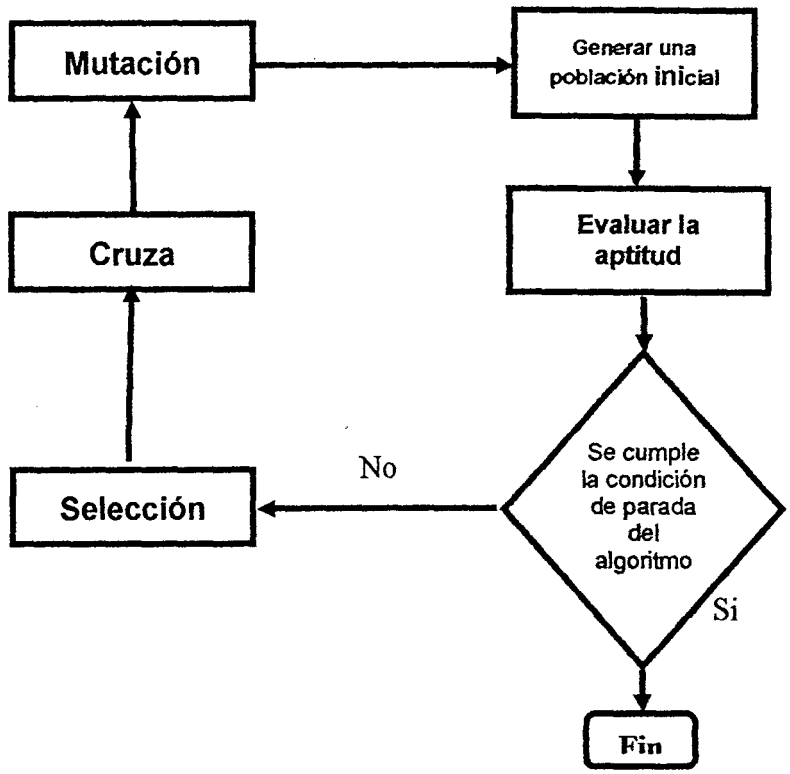

Figura. 2.1 Algoritmo Genético Simple

Ejemplo Sencillo:

Sea una población inicial de cuatro individuos de longitud l = 5, donde cada característica (gen) puede tomar los valores (alelos) X o Y:

XYYXY

YYXXX

XYXXX

YXXYY

La selección puede ser implementada de varias formas. Una de ellas es simular una ruleta donde cada individuo de la población tenga asignada una ranura. El tamaño de esa ranura será proporcional a la aptitud del individuo respectivo. El método consiste en lanzar una "bolilla imaginaria" tantas veces como individuos pretenda seleccionar. Cada individuo obtiene una copia por vez que la bolilla cae en su ranura. Obviamente, aquellos individuos con una ranura más grande tendrán mayor oportunidad de ser elegidos. Este método es conocido como selección por ruleta .

En la tabla 1 se muestra una posible asignación de aptitudes para los individuos, junto con la cantidad de copias esperadas en base a esta aptitud, y la cantidad de copias asignadas realmente.

Nº	**Individuo**	**Aptitud**	**Porcentaje**	**Copias Esperadas**	**Copias Asignadas**
1	XYYXY	169	14.4	0	1
2	YYXXX	576	49.2	2	2
3	XYXXX	64	5.5	0	0
4	YXXYY	361	30.9	2	1
Total		1170	100.0	4	4

Tabla 1 Asignación de Actitudes para los Individuos

Luego de la selección se aplica el operador de cruza. En este ejemplo se utilizara la cruza en un punto que se implementa de la siguiente forma. Se toma un par de individuos previamente seleccionados y se elige en forma aleatoria una posición k entre 1 y la longitud del individuo menos 1 [1, l-1]. Dos nuevos individuos son creados intercambiando genes entre la posición k +1 y l inclusive. Por ejemplo, eligiendo las estructuras 1 y 2, con k =3 se tiene:

XYY|XY XYY|XX

YYX|XX YYX|XY

Luego se aplica el operador de mutación, que consiste en alterar el valor de algunos genes dentro de los individuos de la población. Por ejemplo:

XYYXX XYYYX

La población resultante es nuevamente evaluada y si la condición de parada no se cumple, se continúa aplicando los operadores a la generación obtenida.

2.1.2. Sistemas Naturales y Sistemas Artificiales

Existe una correspondencia entre la terminología usada en la genética natural y la artificial. En este punto es necesario aclarar los términos para evitar confusiones.

En la naturaleza, las características de un organismo están determinadas por uno o más cromosomas. En los sistemas genéticos artificiales, cada individuo se representa con una única estructura, y ambos términos se consideran equivalentes al del cromosoma. Un cromosoma esta compuesto por genes, cada uno de los cuales puede tomar un valor llamado alelo. La posición de un gen dentro de un cromosoma se denomina locus. Por ejemplo, en el gen dentro de un cromosoma se denomina locus. Por ejemplo, en el gen que representa el color de los ojos de un animal, su locus podría ser la posición 10, y el valor de su alelo, ojos marrones. En la genética artificial las estructuras están compuestas de características que pueden tomar diferentes valores. Las características pueden estar ubicadas en diferentes posiciones de un individuo.

Es importante destacar los conceptos de genotipo y fenotipo. El conjunto de genes de un individuo se llama genotipo. El fenotipo es el resultado de la interacción del genotipo con su entorno. Desde el punto de vista de la genética artificial, el fenotipo se corresponde con la aptitud del individuo. Lo expuesto queda resumido en la tabla 2.

Sistemas Naturales	Algoritmos Genéticos
cromosoma	Individuo o estructura
gen	Característica o atributo
alelo	Valor
locus	Posición en la estructura
genotipo	Conjunto de genes o estructura
fenotipo	Aptitud del individuo

Tabla 2 Sistemas Naturales y Algoritmo Genéticos

A lo largo del presente trabajo se usara la terminología natural y la artificial en forma indistinta.

2.1.3. Aplicaciones

Los algoritmos genéticos son atractivos por varias razones:

1. Pueden resolver en forma rápida y eficiente problemas complejos con características como:

   a. Alta cardinalidad del espacio de búsqueda
   b. Alta dimensionalidad de la función de aptitud

c. Función de aptitud no lineales.

2. Utilizan muy poca información específica del problema, dado que solo requiere la posibilidad de proponer una solución y asignar una evaluación.

3. Son extensibles. En los problemas reales es muy difícil anticipar todas las dificultades, sin embargo es muy fácil modificar o incorporar conocimiento en los operadores genéticos.

4. Pueden ser utilizados para realizar una primera búsqueda global, para luego continuar con algún método de búsqueda local.

Los Algoritmos Genéticos tienen básicamente dos campos de aplicación: los problemas de optimización y la simulación de ambientes donde el objetivo es maximizar los beneficios acumulados a través del tiempo. Los problemas de optimización son los mas habituales como ser: la maximización de eficiencia o calidad, minimización de riesgos, costos o tiempo, en proyectos técnicos, económicos o científicos.

Algunos ejemplos de utilización de AG para problemas de búsqueda y optimización son: coloreo de grafos, diseño de circuitos VLSI, optimización del tamaño de enlaces en una red de comunicaciones,

problema de viajante, optimización de funciones, procesamiento de imágenes, reconocimiento de patrones, sistemas clasificadores y programación genética. En particular, los sistemas clasificadores utilizan AG para explorar el espacio de reglas de producción de un sistema de aprendizaje, y la programación genética realiza una búsqueda sobre un espacio de programas de computación escritos habitualmente en LISP.

2.1.3.1 Operadores Básicos: Variantes

El propósito de cada uno de los tres operadores básicos esta bien definido: elegir los individuos mas aptos (selección), recombinar algunos de ellos para producir nuevos individuos (cruza), y alterar características de algunos para garantizar diversidad (mutación). Con estos objetivos presentes, se han desarrollado gran cantidad de variantes para cada uno de los operadores desde la presentación de AG por Holland en 1975. Cada variante, o método, tiene características particulares que afectaran el comportamiento del AG. Una elección adecuada para cada operador puede influir decisivamente en la eficacia y eficiencia del proceso de búsqueda.

2.2. METODOS DE SELECCIÓN

Los individuos elegidos con el operador de selección aportaran sus genes a la generación siguiente, por lo tanto, es deseable que el operador de selección, elija los mejores individuos de la población actual. Existen dos tipos de métodos usados comúnmente: los proporcionales y los basados en el orden .

Los métodos proporcionales de selección eligen individuos teniendo en cuenta el peso de su aptitud respecto del resto de la población. En cambio, los métodos basados en el orden confeccionan una tabla ordenada de individuos en base a su aptitud, seleccionando cada uno de acuerdo a su ubicación es esta tabla de posiciones. A continuación se describen los métodos de selección más comunes:

2.2.1. SELECCIÓN POR RULETA

Este método consiste en construir una ruleta particionada en ranuras de igual tamaño, las cuales se numeran. A cada individuo de la población se le asigna una cantidad de ranuras proporcional a su aptitud. La manecilla de la ruleta se gira, y con probabilidad uniforme se elige una ranura seleccionándose aquel individuo dueño de la misma. El proceso se repite hasta completar la cantidad de individuos

deseados. Este método de selección otorga mayor probabilidad de contribuir a la siguiente generación a los individuos con mayor aptitud. Como se trata de un método probabilístico, el número obtenido de copias para un individuo puede ser muy distante del esperado.

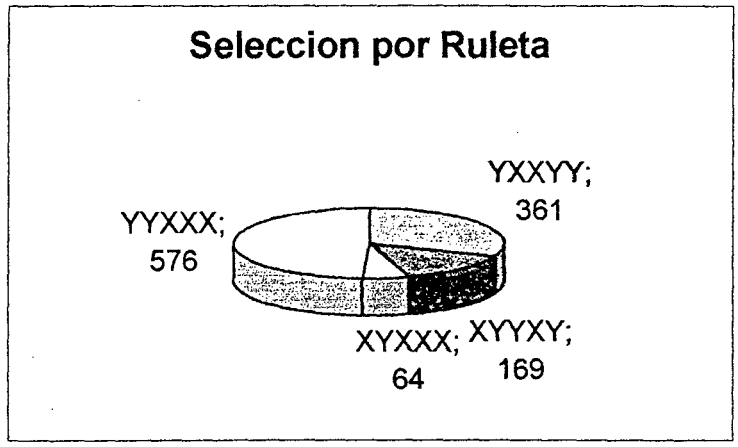

Figura 2.2 Selección por Ruleta

2.2.2. SELECCIÓN POR CONTROL DEL NUMERO ESPERADO

La selección por ruleta intenta asignar una cantidad de copias adecuada a los mejores individuos, pero no lo garantiza. Como se menciono anteriormente, puede haber una gran distancia entre el número esperado de copias y la cantidad asignada. Para reducir este error estocástico, De Jong y Spears [1992] diseñaron el modelo del valor esperado. Se define una cantidad de copias c, para un individuo i, tal que:

$$c_i = \frac{f_i}{\bar{f}}$$

Donde f_i es la aptitud del i-esimo individuo y f es la aptitud promedio de la población. Cada individuo recibe una cantidad de copias igual a la parte entera de e_i mas una copia adicional con probabilidad igual a la parte fraccionarla de c_i. Esto garantiza que la cantidad de hijos de cualquier individuo es por lo menos la parte del número esperado de hijos.

N°	**Individuo**	**Aptitud**	c_i	**Copias Esperadas**	**Copias Asignadas**
1	XYYXY	169	0.58	≥ 0	0+1
2	YYXXX	576	1.97	≥ 1	1+1
3	XYXXX	64	0.22	≥ 0	0+0
4	YXXYY	361	1.23	≥ 1	1+0
Total		1.170	4		4

Tabla. 3 Selección con Control sobre el Número Esperado

2.2.3. SELECCIÓN ELITISTA

Los dos métodos anteriores no garantizan la preservación de los mejores individuos, ya que estos pueden ser reemplazados por sus hijos durante la cruza. La selección elitista soluciona este problema, común a la mayoría de los métodos de selección.

El método elitista preserva los mejores m individuos de la generación actual, incluyéndolos directamente en la siguiente. Se busca mejorar la búsqueda local a expensas de la perspectiva global. También provoca que la aptitud del mejor individuo de la población mejore o a lo sumo se mantenga, pero nunca decrezca. La selección elitista siempre es utilizada en combinación con otras variantes de selección.

2.2.4. SELECCIÓN POR RANKING

Un problema habitual de los métodos proporcionales es la posibilidad de que súper-individuos dominen rápidamente la población, antes de que se haya podido realizar una exploración suficientemente amplia del espacio de búsqueda. Este problema se denomina convergencia prematura. Una de las alternativas para solucionarlo consiste en utilizar un método de selección que tenga en cuenta cuales son los mejores individuos, pero que no considere el peso de su aptitud. Otra alternativa, mucho más difundidas, consiste en el escalado de la función de aptitud. Este método de selección

tiene dos desventajas frente a los métodos proporcionales. La primera es que la evolución de la población frecuentemente es más lenta. La segunda desventaja es que no existe una teoría sólida que lo soporte, como existe para los proporcionales.

En este método cada individuo recibe una cantidad de copias que solo depende de su ubicación dentro de la tabla 4. Para esto, la población se ordena en forma descendente por la aptitud de cada individuo en una tabla de posiciones, y solamente los primeros m individuos reciben copias a asignar se distribuye en forma lineal.

Una posible implementación de este método es la siguiente. Sea n el tamaño de la población. Se define un mapeo lineal de tal forma que la peor estructura recibe un numero esperado de copias $R_{min} \in (0,1)$ y la mejor recibe un numero esperado de $2-R_{min}$. Al individuo ubicado en la posición i de la tabla 4 se le asigna la siguiente cantidad de copias:

$$R_{\min} + 2\frac{(n-i)(1-R_{min})}{n-1}$$

Aplicando este método al ejemplo visto, se tiene:

N°	Individuo	Aptitud	Copias Esperadas	Copias Asignadas
2	YYXXX	576	1.25	2
4	YXXYY	361	1.08	1
1	XYYXY	169	0.91	1
3	XYXXX	64	0.75	0
Total		1170	4	4

Tabla 4. Selección con Control sobre el Número Esperado

2.3. METODOS DE CRUZA

Es importante el rol que juega el operador de cruza en el diseño e implementación de sistemas adaptativos robustos. En la mayoría de los AG los individuos son representados por estructuras de longitud fija y la recombinación es implementada por el operador de cruza. Este opera sobre pares de estructuras (padres) para producir nuevas estructuras (hijos) por el intercambio de segmentos entre los padres.

Cada estructura participa de la cruza con probabilidad P_c. La cantidad de hijos generados depende de la implementación. La elección mas aceptada consiste en

generar dos hijos, siendo uno el complemento del otro. Sin embargo, también hay variantes donde la cruza genera un único hijo o incluso más de dos.

2.3.1. CRUZA SIMPLE

La cruza simple, o en un punto, elige al azar uno de los l-1 posibles puntos de cruza. Luego de esta elección se intercambian los segmentos de cromosoma separados por este punto. Suponiendo las siguientes estructuras de longitud l = 8, y eligiendo 3 como el punto de cruza se tiene:

XYX|XYYYX	XYX|YYXXY
YYX|YYXXY	YYX|XYYYX

2.3.2. CRUZA MULTIPUNTO

En este caso, el cromosoma es considerado un anillo, y se eligen n puntos de cruza en forma aleatoria. Si la cantidad de puntos de cruza es par, se intercambian las pociones de cromosomas definidas entre cada par de puntos consecutivos, si es impar se asume un punto de cruza adicional en la posición cero y se procede de igual modo.

Para ilustrar esta variante, supongamos dos estructuras de longitud l = 8, con n = 4 puntos de cruza. Intercambiando los segmentos de la posición 2 a 4 y 6 a 7, se tiene:

X|YXX|Y|YY|X	X|YXY|Y|XX|X
Y|YXY|Y|XX|Y	Y|YXX|Y|YY|Y

La cruza simple en realidad es un caso particular de la cruza multipunto cuando n = 1. Se debe mencionar, también, la cruza de dos puntos por la cantidad de estudios e implementaciones de que ha sido objeto.

2.3.3. CRUZA BINOMIAL

Para generar un cromosoma hijo por cruza binomial, se define la probabilidad Po como la probabilidad de que el Alelo de cualquier posición del descendiente se herede del padre, y $1-P_0$ como la probabilidad de que lo herede de la madre. En este caso se puede construir un único hijo por cada aplicación del operador, o bien generar un segundo hijo como complemento del primero.

Cuando existe igual probabilidad de heredar del padre como de la madre, $P_0 = 0,5$, la cruza se denomina uniforme. Para estructuras de longitud l la cruza uniforme implica un promedio de ½ puntos de cruza.

2.4. METODOS DE MUTACION

La mutación permite mantener diversidad en la población disminuyendo el riesgo de convergencia prematura. A la vez cumple una función de exploración ya que este operador brinda la posibilidad de generar cualquier estructura valida dentro del espacio de búsqueda.

Durante la aplicación del operador de mutación, cada gen es mutado con una probabilidad P_0, generalmente baja. La probabilidad puede ser constante durante toda la búsqueda genética o bien adaptativa. En el primer caso no se comparte información, en cambio en el segundo caso, se utilizan frecuentemente estadísticas de la población para adaptarla velocidad de mutación. Se describen varias implementaciones de este operador.

2.4.1. MUTACION SIMPLE

La mutación simple elige en forma aleatoria un gen, el cual se muta con cierta probabilidad, habitualmente muy baja. La probabilidad de mutación se mantiene constante durante las sucesivas generaciones. Debido a esto, es bastante dificil que la probabilidad elegida sea adecuada en todo momento, y por lo tanto, el operador de mutación no es bien explotado. Por otra parte, es dificil determinar la forma de adaptar el grado de mutación así que generalmente se utiliza este método debido a su sencillez.

2.4.2. MUTACION ADAPTATIVA POR CONVERGENCIA

En la mutación adaptativa por convergencia , la probabilidad de mutación varia en base a información proveniente de la búsqueda genética. Esta información esta dada por el grado de convergencia de la población (P_0 se puede ver, también, como la probabilidad de intercambiar alelos . Esta variante de mutación tiene la ventaja de aprovechar la información histórica para orientar la búsqueda, aumentando la mutación cuando la población se hace muy homogénea y disminuyéndola cuando hay demasiada diversidad. Generalmente, se comienza con probabilidades bajas y se aumenta o disminuye en función de la evolución de la población.

2.4.3. MUTACION ADAPTATIVA POR TEMPERATURA

El concepto de mutación adaptativa por temperatura deriva del método Simulated Annealing. Este tipo de mutación no utiliza información genética de la población, por lo tanto, es independiente de las características de la misma.

La probabilidad de mutación, P_m, depende del tiempo o cantidad de generaciones; por lo tanto $P_m = P_m(t)$. El rango de valores que puede tomar esta acotado por valores mínimo y máximo:

$$P_m \min \geq p_m(t) \geq P_m \max$$

Partiendo de un valor inicial P_m (0), la actualización se realiza hasta alcanzar un valor final, de la siguiente manera:

$$p_m(t+1) = p_m(t)+1$$

Dependiendo del signo de λ y de los valores inicial y final de P_m (t), hay dos variantes del método de mutación adaptativa por temperatura.

2.5. RESOLUCION DE PROBLEMAS CON AG

El objetivo es proponer un método informal de diseño de un AG para resolver un problema arbitrario. Para esto, se analizan dos aspectos claves para el éxito del algoritmo: lograr una representación apropiada para los cromosomas, y definir una función de aptitud con características que faciliten la búsqueda. Estos aspectos son los únicos estrechamente relacionados con el problema en si, puesto que una vez definidos ambos parámetros, el AG es absolutamente independiente del problema.

La sección se divide en tres partes. La primera parte trata sobre el problema de la representación de cromosomas, la segunda sobre la elección de la función de aptitud, y la tercera plantea el método de diseño propuesto.

2.5.1. EL PROBLEMA DE LA REPRESENTACION

Para la resolución de problemas mediante AG es crítico realizar una adecuada elección de la semántica de una estructura. Esto es, como representar una solución al problema en un individuo que pueda ser procesado por un AG. Los factores a tener en cuenta para la representación de los individuos son dos:

- La codificación de cada parámetro en un gen.
- La distribución de los genes dentro del cromosoma.

Ambos factores influyen en el tamaño del espacio de búsqueda, y en la construcción de bloques constituyentes funcionales al AG.

2.5.1.1. CODIFICACION DE PARAMETROS

Algoritmos Genéticos no presenta restricciones en cuanto a la forma que deben adoptar los genes. Estos pueden representarse como cadenas binarias, números enteros o reales, o estructuras más complejas. Se debe tener en cuenta que la representación debe elegirse conjuntamente con operadores genéticos adecuados para manejarla.

Una buena codificación es aquella que permite que los operadores genéticos exploten adecuadamente las similitudes entre individuos. Y estas, a su vez, deben reflejarse como similitudes en la codificación. A continuación se describen los métodos básicos de codificación

2.5.1.2. DISTRIBUCION DE LOS GENES DENTRO DEL CROMOSOMA

El segundo factor a tener en cuenta para la representación de los cromosomas es la distribución de los genes dentro del mismo. Como la cruza multipunto con pocos puntos de cruza genera bloques constituyentes de corta longitud definida, es conveniente que los genes que estén correlacionados entre si se ubiquen en posiciones adyacentes dentro del cromosoma. Si no se conoce el problema lo suficiente como para determinar esta correlación, se puede utilizar el operador de inversión .

El operador de inversión realiza permutaciones de un mismo cromosoma. Los genes pueden cambiar de posición dentro del cromosoma, para lo cual es necesario redefinir su estructura de tal forma que esto refleje la posición de cada gen. Para esto, cada gen es un par ordenado compuesto por el alelo y su ubicación. Por ejemplo, el cromosoma genérico de longitud 5:

$$(a_1, a_2, a_3, a_4, a_5)$$

con este nuevo concepto pasa a ser:

$$((1, a_1),(2, a_2),(3, a_3),(4, a_4),(5, a_5))$$

El símbolo "no interesa" - * - no tiene asociado una posición, ya que ocupa todas las posiciones no definidas. Por ejemplo, los siguientes esquemas son equivalentes:

((1, X), *, *, (4, Y)) y (*, *, (4, Y), (1, X))

Para aplicar el operador de inversión se elige un cromosoma al azar, y se determinan dos puntos de corte, x_1 y x_2 tal que $x_1 < x_2$. La nueva estructura se genera invirtiendo el segmento que comienza a la derecha de la posición x_1 y termina a la izquierda de la posición x_2.

Sea el cromosoma $u_{A1....A_n}(x_1,...,x_n) = u_{A1}(x_1),...,u_{An}(x_n)$ de longitud l,

Donde : $A_i = (i,a)$:

$$(A_1,.....,A_j,A_{j+1},A_{j+2},.....,A_{k-2},A_{k-1},A_k,.....,A_l)$$

Y eligiendo los puntos de corte j y k, con j < k, se tiene el siguiente resultado luego de aplicar el operador de inversión:

$$(A_1,.....,A_j,A_{k-1},A_{k-2},.....,A_{j+2},A_{j+1},A_k,.....,A_l)$$

El operador de inversión puede acercar genes que previamente estaban muy alejados, por ejemplo: A_j, y A_{k-1} o A_{j+1} y A_k. También produce el efecto contrario, alejar genes que estaban muy cercanos entre si. Como propiedad se puede mencionar que:

Cualquier posible permutación del cromosoma puede ser producida por una secuencia apropiada de inversiones.

Un cromosoma afectado por el operador de inversión no altera su significado, dado que cada gen es independiente de su posición y conserva su alelo. El operador preserva la posición lógica del gen dentro del cromosoma pero altera su distribución física. El efecto de la inversión consiste en generar permutaciones de los esquemas ya existentes, con distintas longitudes definidas. Como consecuencia de esto, la representación evoluciona de generación en generación hacia un estado que permite aprovechar el sesgo de la cruza.

2.5.2. ELECCION DE LA FUNCION DE APTITUD

En los últimos años se han investigado las características de los problemas que son de difícil resolución a través de AG. Actualmente se puede identificar con bastante certeza aquellos problemas extremadamente difíciles para AG, con lo cual es posible restringir la aplicación de estos a problemas de dificultad acotada. Se discutirán algunas características

habitualmente consideradas críticas de la superficie de aptitud que pueden influir considerablemente en la eficiencia de un AG.

Un problema es dificultoso para AG cuando la superficie de aptitud presenta alguna de las siguientes características:

- ***Aislamiento***: la solución deseada esta aislada y su cuenca de atracción es muy pequeña.
- ***Decepción***: el AG es atraído por subóptimos locales y alejado del óptimo global.
- ***Multimodalidad***: existencia de mas de un optimo local.
- ***Ruido***: la función objetivo posee pequeños errores respecto de la función que se quiere optimizar.

Una elección adecuada de la función de aptitud a menudo elimina la presencia de algunas de estas características La mayor parte de las investigaciones se han centrado en los problemas que combinan aislamiento y decepción.

2.5.3. METODOLOGIA DE DISEÑO EN AG

Los Algoritmos Genéticos conforman un paradigma apto para la resolución de problemas, con una característica peculiar: no necesita conocimiento específico del problema. Esto significa dos cosas: no exige condiciones y la función objetivo, y una vez definidos la representación del cromosoma y la función de aptitud, no es necesario ningún conocimiento adicional para la implementación del algoritmo. Sin embargo, la función de aptitud y la estructura del cromosoma, son el nexo clave del AG con el problema a resolver.

Se propone a continuación una metodología informal que describe los pasos a seguir para la resolución de un problema arbitrario con AG. Estos pueden separarse en dos fases: una dependiente del problema y otra independiente.

Los pasos a seguir son:

Fase Dependiente del Problema

- Análisis del Problema
- Diseño del Cromosoma
- Elección de la Función de Aptitud

Fase Independiente del Problema

- Elección de los Operadores Genéticos
- Implementación del AG
- Análisis de Resultados

2.5.3.1 FASE DEPENDIENTE DEL PROBLEMA

Es necesario analizar y comprender el problema para definir cada etapa del algoritmo en forma adecuada y de esta manera lograr un diseño final que permita una solución genética eficaz y eficiente. En una primera iteración en el método de diseño el conocimiento del problema probablemente sea pobre. A medida que se ensayan soluciones y analizan resultados este conocimiento crecerá y permitirá mejorar las nuevas soluciones. Las mejoras se logran rediseñando alguna etapa de acuerdo a la experiencia con otros problemas o con intentos previos de solución al problema actual.

Partiendo de un conocimiento dado del problema se define la representación del cromosoma, es decir la codificación de los parámetros y la distribución de los genes dentro del mismo. Si no se puede determinar una adecuada distribución de los genes, se debe analizar la conveniencia de utilizar el operador de inversión.

Dada la función objetivo a optimizar y el diseño del cromosoma, se define la función de aptitud que puede coincidir o no con la función objetivo. Esta definición depende del conocimiento que se tenga del problema, por lo cual, probablemente, sea necesario revisar el análisis del mismo.

2.5.3.2 FASE INDEPENDIENTE DEL PROBLEMA

Dada la representación del cromosoma y la función de aptitud, es necesario definir los operadores genéticos a utilizar. Es decir, elegir un método de selección que provea una presión adecuada, un método de cruza que permita construir bloques constituyentes funcionales al problema, y un método de mutación que garantice un mínimo de diversidad en la población.

Si bien la elección de los operadores se puede realizar utilizando conocimiento del problema, este conocimiento previo no es una condición necesaria. Sin embargo se pueden adaptar o crear nuevos operadores de tal forma que dependan del problema. Estas son las razones por las cuales la elección de operadores se considera dentro de la fase independiente del problema.

Con los operadores genéticos definidos, queda por implementar el algoritmo. El programa resultante debe ser suficientemente flexible como para poder cambiar la representación del cromosoma, la función de

aptitud, o los operadores genéticos sin grandes modificaciones. El análisis de los resultados obtenidos probablemente llevara a retroceder a alguna etapa anterior, donde se pueda mejorar la decisión realizada y volver a avanzar en la resolución.

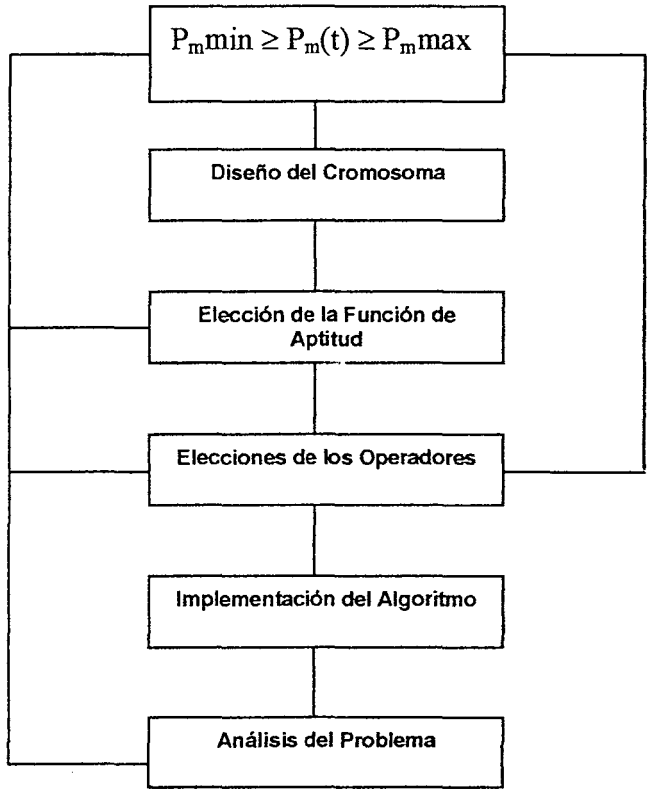

Figura 2.3 Representación del Cromosoma

2.6. ALGORITMOS EVOLUTIVOS

Los Algoritmos Genéticos pertenecen a una clase mas genérica de algoritmos, los AE simulan la denominados Algoritmos Evolutivos, en adelante AE. La evolución de una población de individuos mediante los procesos de selección, recombinación y mutación. Estos procesos dependen de la aptitud de cada individuo definida en base a su entorno. Los distintos tipos de AE se diferencian entre si de acuerdo a la importancia que cada uno le asigna a estos procesos, el diseño de los mismos, y la representación de los individuos.

Los AE son un concepto general adaptable para la resolución de problemas, y no una colección de algoritmos relacionados y listos para ser usados. Son especialmente adecuados para resolver problemas difíciles de optimización. Los AE, en general, datan de 1950, pero en las últimas décadas emergieron las tres variantes más importantes: Programación Evolutiva (PE), Estrategia Evolutiva (EE) y Algoritmos Genéticos (AG).
Estas variantes implementan algoritmos evolutivos de diferente manera.

Los A.G se pueden, a su vez, clasificar en dos categorías: Secuenciales (AGS) y Paralelos (AGP). El AGS es el A.G tradicional, estudiando ampliamente en las primeras secciones. El AGP se deriva de la posibilidad de implementar el algoritmo en arquitecturas paralelas o distribuidas.

La figura muestra una clasificaron de las variantes mencionadas de AE.

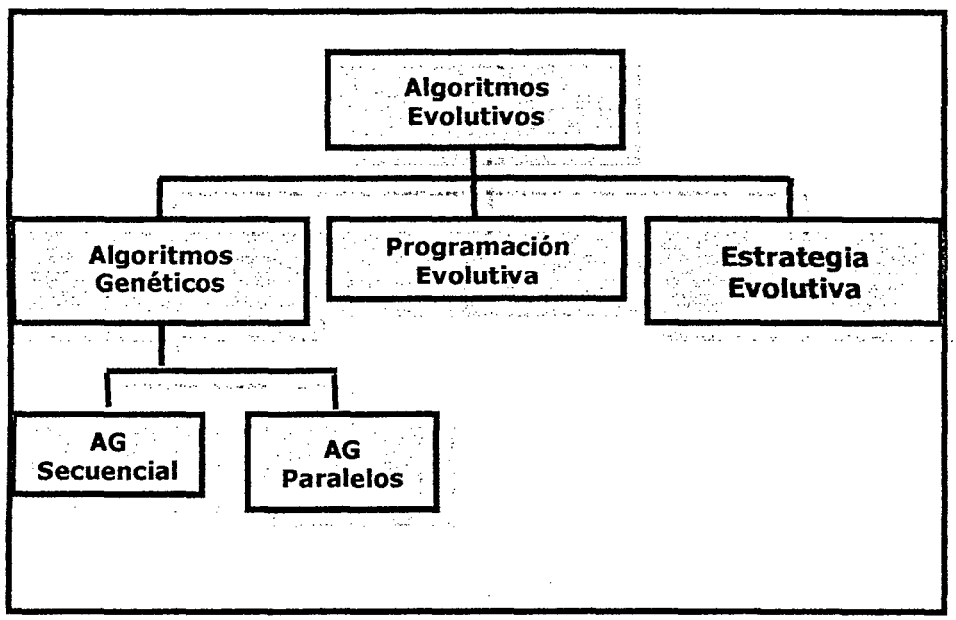

Figura 2.4 Clasificación de las Variantes Mencionadas del A.G

La estructura de un AE genérico se observa en la figura de abajo. Este algoritmo mantiene una población de estructuras que renueva de acuerdo a las reglas de selección, recombinación y mutación. A cada individuo de la población se le asigna una medida de su aptitud en el entorno. La selección enfoca su atención en los individuos de mayor aptitud para explotar la información provista por los mismos. La recombinación y mutación perturban estos individuos ampliando el espacio de exploración.

```
t ← 0
generar población inicial P(t)
evaluar P(t)
Hasta (condición de parada)
      t ← t + 1
      seleccionar P(t)
      recombinar P(t)
      mutar P(t)
      evaluar P(t)
      supervivencia P(t)
Fin Hasta
```

En la tabla 5 se resumen las características distintivas de los distintos Algoritmos Evolutivos.

	Representación	**Selección**	**Cruza**	**Mutación**
Programación Evolutiva	Dependiente del problema	Determinística	No utiliza	Operador Principal
Estrategia Evolutiva	Dependiente del problema	Ranking	Operador secundario	Operador principal
Algoritmos Genéticos	Independiente del problema	Estocástica	Operador principal	Operador secundario

Tabla 5 Características de los Distintos A.G

2.6.1. PROGRAMACION EVOLUTIVA

La PE fue desarrollada por Fogel en 1966. La representación de los individuos en PE esta adaptada al dominio del problema. Por ejemplo: en problemas de optimización con valores reales, los individuos dentro de la población son vectores de números reales; para el problema del viajante se utilizan listas ordenadas, y grafos para aplicaciones con maquinas de estados finitos. PE es frecuentemente usado como un optimizador.

Los n individuos de la población son siempre seleccionados como padres, y luego son mutados generando n hijos. Estos hijos son evaluados para luego elegir n sobrevivientes del total de 2n individuos, usando una función probabilística basada en la aptitud de cada individuo. La PE no utiliza operador de recombinación.

La mutación esta basada en la representación usada, y frecuentemente, es adaptativa, es decir, en lugar de usar una probabilidad de mutación global, esta se codifica como información propia de cada individuo y evoluciona con el.

2.6.2. ESTRATEGIA EVOLUTIVA

En 1973 Rechenberg desarrollo EE, utilizando una población de tamaño uno, y operadores de selección y mutación. En 1981 se amplio el paradigma a poblaciones de tamaño mayor a uno, y se introdujo el operador de cruza. Debido a que inicialmente estos algoritmos se utilizaban para resolver problemas de optimización hidrodinámicas, los individuos se representan, habitualmente, con un vector de números reales.

Se utiliza selección por ranking, donde los n mejores individuos son seleccionados para ser padres. Estos padres producen hijos por medio del operador de cruza, incorporando perturbaciones a través de la mutación. El algoritmo permite generar un numero de hijos mayor a n.

El operador de mutación es la principal herramienta de exploración, y, al igual que PE, se utiliza frecuentemente mutación adaptativa. Se utiliza operador de cruza, pero como operador secundario que interviene en la adaptación de la mutación.

La supervivencia es Determinística y puede ser implementada en una de dos formas: los n mejores hijos sobreviven y reemplazan a sus padres en la próxima generación, o se eligen los n mejores entre hijos y padres.

Es importante destacar que: tanto PE como EE usualmente tienen como objetivo optimizar. Es decir, el interés se concentra en hallar la mejor solución tan rápidamente como sea posible

2.6.3. ALGORITMOS GENETICOS SECUENCIALES

Como se mencionó previamente, los AGS son los tradicionales desarrollados por Holland en 1975. Estos se distinguen dentro de los AE por las siguientes características.

La representación utilizada es generalmente binaria, lo cual permite independizarse del dominio del problema. La selección de padres se realiza en base a una función del dominio del problema. La selección de padres se realiza en base a una función probabilística que depende de la aptitud de cada individuo. A diferencia de PE, la cruza es considerada el operador principal de exploración mientras que mutación cumple un rol secundario.

2.6.4. ALGORITMOS GENETICOS PARALELOS

Los Algoritmos Genéticos Paralelos, AGP, han sido usados para resolver problemas dificultosos que necesitan una gran población, trasladándose esto en un gran costo computacional. La motivación básica de estudiar los

GP ha sido la de reducir el tiempo de procesamiento necesario para la búsqueda de una solución aceptable.

Actualmente, los Algoritmos Genéticos se están estudiando ampliamente. Los AGP son fáciles de paralelizar y muchas variantes del modelo básico han sido implementadas con muy buenos resultados en diferentes clases de problemas.

Con los algoritmos genéticos secuenciales, se debe elegir entre obtener un buen resultado y pagar un alto costo en tiempo de procesamiento o perder en la calidad del resultado a favor de encontrarlo en menor tiempo. En contraste, los Algoritmos Genéticos Paralelos pueden obtener un resultado de alta calidad y encontrarlo rápidamente porque, al usar maquinas paralelas, pueden ser procesados grandes características de los tres primeros.

2.6.4.1. GLOBAL

En esta clase de AGP, la evaluación de los individuos y la aplicación de los operadores genéticos son explícitamente paralelizados. Cada individuo tiene probabilidad de cruzarse con el resto de los individuos de la población. Por lo tanto, la semántica de los operadores genéticos no cambia.

La evaluación puede ser paralelizada asignando un subconjunto de individuos a cada procesador disponible. La comunicación únicamente ocurre al comienzo y al final de esta fase.

En un multiprocesador de memoria compartida, los individuos son almacenados en dicha memoria. Cada procesador puede leer el individuo asignado y escribir el resultado de la evaluación.

Mientras que en un multiprocesador de memoria distribuida, la población es almacenada en un procesador para simplificar la aplicación de los operadores genéticos. Este procesador servidor será el responsable de enviar los individuos a los otros procesadores (clientes) para su evaluación, recolectar los resultados, y aplicar los operadores genéticos para producir la próxima generación. Sin embargo, podría producirse un cuello de botella en el servidor mientras los clientes permanecen ociosos.

Otro de los problemas planteados es que no se puede asegurar que la paralelización de la aplicación de los operadores resulte en una mejora de la performance. Los operadores genéticos son muy simples y el tiempo de comunicación puede ser mayor que el tiempo de cálculo. Esto es especialmente cierto en maquinas de memoria distribuida, donde el overhead de comunicación puede ser considerable.

2.6.4.2. GRANO GRUESO

La característica más importante de esta categoría es que la población se divide en unas pocas subpoblaciones, estando relativamente aisladas unas de otras, y se introduce un nuevo operador, migración, que es utilizado para intercambiar individuos entre subpoblaciones. Los AGP Grano Grueso son el modelo más popular.

La población puede ser implementada con dos modelos distintos: island o stepping stone. La población, en ambos modelos, es particionada en pequeñas subpoblaciones. En el modelo island los individuos pueden migrar a cualquier otra subpoblación. Mientras que en el modelo stepping stone, la migración esta restringida a subpoblaciones vecinas.

Es importante destacar que la migración es controlada por muchos parámetros, la topología que define la conexión entre las subpoblaciones, una tasa de migración que controla cuantos individuos serán migrados, y un intervalo de migración que determina cuando se llevara a cabo la migración.

Una de las preguntas que se realiza con mayor frecuencia es: ¿Cuándo se debe migrar? Si la migración ocurre antes que los individuos a migrar hayan evolucionado lo suficiente entonces la influencia en la búsqueda

hacia la dirección correcta será muy pobre y se gastara mucho tiempo en comunicación.

Por otro lado, es importante determinar la topología de interconexión entre subpoblaciones porque determina cuan rápido (o lento) una buena solución será diseminada a las otras subpoblaciones. Si la topología tiene una conectividad densa, buenas soluciones llegaran rápidamente a todas las subpoblaciones y predominaran en la población. Sin embargo, si la topología esta débilmente conectada, las soluciones serán distribuidas en las subpoblaciones lentamente, permitiendo la aparición de individuos autóctonos potencialmente mejores.

Es de esperar que si las subpoblaciones están relativamente aisladas, cada una encuentre diferentes soluciones parciales del problema. De ahí se puede concluir que si soluciones parciales pueden ser combinadas para formar una mejor solución, entonces los AGP probablemente encuentren mejores soluciones que los AG secuenciales.

En general los AGP grano grueso son conocidos como AG Distribuidos dado que usualmente son implementados en computadoras de memoria distribuida.

2.6.4.3. GRANO FINO

La población es particionada en un gran número de pequeñas subpoblaciones. El caso ideal es que cada subpoblacion tenga un solo individuo para todo elemento de procesamiento disponible. Para estos casos donde el cálculo de la función de aptitud requiera de un proceso complejo, esta alternativa puede ser adecuada. Este modelo es apto para ser implementado en computadoras de arquitectura paralela.

Tanto en el método de Grano Grueso como en el de Grano Fino la selección y cruza ocurre dentro de cada subpoblación. Dado que el tamaño de cada subpoblación es mucho menor que el usado en AG secuénciales, es de esperar que los AGP converjan más rápidamente. Esta convergencia se atenúa con las migraciones de individuos entre subpoblaciones.

Es interesante destacar que, mientras el método Global es simplemente la implementación del AG secuencial en una maquina paralela, en los métodos de Grano Grueso y Grano Fino el paralelismo esta dado por la semántica del algoritmo, por lo que pueden implementarse tanto en maquinas secuénciales como paralelas.

3. MODELOS MATEMATICO NO LINEAL DEL B.A.P. CARVAJAL

En este capitulo se obtendrá el modelo no lineal del control de dirección del B.A.P. Carvajal que es un buque de carga en actividad de la Marina de Guerra del Perú.

Para mejorar la eficiencia del combustible y reducir el desgaste en componentes del barco, los sistemas de piloto automático han sido desarrollados e implementados para controlar el rumbo direccional del Buque. A menudo, los pilotos automáticos utilizan esquemas simples de control como el control PID.

Sin embargo, la capacidad para los ajustes manuales de los parámetros del controlador es añadir un compensador para actuar en los disturbios en el barco como viento y corrientes.

Una vez que los parámetros adecuados del controlador son encontrados manualmente, el controlador generalmente trabajará adecuadamente para variaciones pequeñas en las condiciones operativas.

Para variaciones grandes, sin embargo, los parámetros del piloto automático deben estar continuamente modificados. Tales ajustes continuos son necesarios porque la dinámica de un barco cambia, para el ejemplo, la velocidad, y la carga. También, es útil para cambiar los parámetros de ley de control de piloto automático cuando el barco está expuesto a los disturbios grandes resultantes de cambios en el viento, las ondas, coetáneo, y la profundidad de agua.

El ajuste manual de los parámetros del controlador es a menudo una carga en la tripulación. Además, el pobre ajustamiento puede resultar de un error humano. Como consecuencia, es de gran interés tener un método para ajustar automáticamente el controlador

3.1. Modelo No Lineal del Sistema de Dirección del B.A.P. "Carvajal".

Generalmente, la dinámica del buque es obtenida por leyes Newton que aplica de movimiento para el buque. Para buques muy grandes, el movimiento en el plano vertical puede estar desechado desde que el "balanceo" o "efecto de rebote" del buque es pequeño para embarcaciones grandes. El movimiento del barco está generalmente descrito por un sistema de coordenadas esto es fijado por el buque . Como se muestra en la figura 3.1.

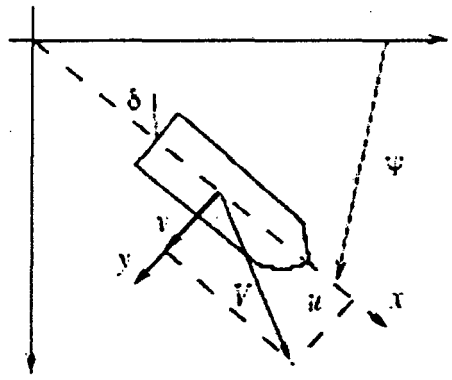

Figura 3.1. Buque de Carga.

El modelo del movimiento del buque es dado por:

$$\dddot{\psi}(t)+\left(\frac{1}{\tau_1}+\frac{1}{\tau_2}\right)\ddot{\psi}(t)+\left(\frac{1}{\tau_1\tau_2}\right)\dot{\psi}(t)=\frac{K}{\tau_1\tau_2}(\tau_3\dot{\delta}(t)+\delta(t)) \qquad (2.1)$$

Donde ψ es la cabecera del buque y δ es el ángulo del timón del buque, asumiendo condiciones iniciales cero la ecuación (2.1), puede redescribirse como:

$$\frac{\psi(s)}{\delta(s)} = \frac{K(s\tau_3 + 1)}{s(s\tau_1 + 1)(s\tau_2 + 2)} \qquad (2.2)$$

Donde K, τ_1, τ_2 y τ_3 son los parámetros constantes que están en función del buque así como **u** velocidad y **l** como longitud.

$$K = K_0\left(\frac{u}{l}\right) \qquad = 1,2,3 \qquad (2.3)$$
$$\tau_i = \tau_{i0}\left(\frac{l}{u}\right)$$

Donde de acuerdo con el buque Carvajal posee K_0 = -3.86, $\tau_{10}, \tau_{20}, \tau_{30}$, y l =161 metros. Asumimos que el buque viaja en dirección X a una velocidad de 5 m/s.

En dirección normal, un buque a menudo hace sólo desviaciones pequeñas de en curso de línea recta. Por consiguiente, el modelo de la ecuación (2.1) es obtenido por linealización de las ecuaciones de movimiento alrededor del ángulo del timón cero ($\delta = 0$).

Como consecuencia, el ángulo del timón no debería exceder aproximadamente 5 grados, de otra manera el modelo será inexacto.

Para nuestros propósitos, necesitamos un modelo adecuado para ángulos del timón que son mayores que 5 grados. Este modelo extendido es dado por:

$$\dddot{\psi}(t)+\left(\frac{1}{\tau_1}+\frac{1}{\tau_2}\right)\ddot{\psi}(t)+\left(\frac{1}{\tau_1\tau_2}\right)H(\dot{\psi}(t))=\frac{K}{\tau_1\tau_2}(\tau_3\dot{\delta}+\delta(t)) \qquad (2.4)$$

Donde $H(\dot{\psi})$ es una función no lineal de $\dot{\psi}$ (t). La función $H(\dot{\psi})$ puede ser encontrada de la relación entre δ y ψ en espacio de estado tal cual $\dddot{\psi}=\ddot{\psi}=\dot{\delta}=0$ y experimento conocido como el "test de espiral" aproxima $H(\dot{\psi})$ por:

$$H(\dot{\psi})=\bar{a}\dot{\psi}^3+\bar{b}\dot{\psi} \qquad (2.5)$$

Donde $\bar{a}$ y $\bar{b}$ son valores reales constantes, asimismo $\bar{a}$ es siempre positivo. Para nuestras simulaciones ambos valores serán los mismos.

3.2. Simulación del Sistema.

Cuando nosotros evaluaremos nuestros controladores, usaremos el modelo no lineal en la simulación. Note que al hacer esto nosotros necesitamos convertir el nth-orden de las ecuaciones diferenciales ordinarias no lineales representadas en el buque para la n primera-orden de las ecuaciones diferenciales ordinarias; Para la comodidad, dejamos.

$$a = \left(\frac{1}{\tau_1} + \frac{1}{\tau_2} \right)$$

$$b = \left(\frac{1}{\tau_1 \tau_2} \right)$$

$$c = \left(\frac{k\tau_3}{\tau_1 \tau_2} \right)$$

y

$$d = \left(\frac{k}{\tau_1 \tau_2} \right)$$

A nosotros no convendría tener el modelo de la forma:

$$\dot{x}(t) = F(x(t), \delta(t))$$

$$y(t) = G(x(t), \delta(t))$$

Donde $x(t) = [x_1(t), x_2(t), x_3(t)]^T$ y $F = [F_1, F_2, F_3]^T$, para usar en un programa de simulación no lineal. Nosotros necesitamos escoger $\dot{x}_i$ de tal manera F_i depende solo de x_i y δ para $i = 1, 2, 3$. Nosotros tenemos:

$$\dddot{\psi}(t) = -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + c\dot{\delta}(t) + d\delta(t)$$

Escogiendo:

$$\dot{x}_3(t) = \dddot{\psi}(t) - c\dot{\delta}(t)$$

Entonces F_3 no dependerá de $c\dot{\delta}(t)$ y

$$\dot{x}_3(t) = \ddot{\psi}(t) - c\delta(t)$$

Escogiendo $\dot{x}_2(t) = \ddot{\psi}(t)$ entonces $x_2(t) = \dot{\psi}(t)$. Finalmente, escogemos

$$x_1(t) = \psi(t).$$

Esto nos da:

$$\dot{x}_1(t) = x_2(t) = F_1(x(t), \delta(t))$$
$$\dot{x}_2(t) = x_3(t) + c\dot{\delta}(t) = F_2(x(t), \delta(t))$$
$$\dot{x}_3(t) = -a\ddot{\psi}(t) - bH(\dot{\psi}(t)) + d\dot{\delta}(t)$$

Pero, $\ddot{\psi}(t) = x_3(t) + c\dot{\delta}(t)$, $\dot{\psi}(t) = x_2(t)$ y $H(x_2) = x_2^3(t) + x_2(t)$ entonces

$$\dot{x}_3(t) = -a(x_3(t) + c\dot{\delta}(t)) - b(x_2^3(t) + x_2(t)) + d\dot{\delta}(t) = F_3(x(t), \delta(t))$$

Esto proporciona las ecuaciones correctas para la simulación. Después, suponga que las condiciones iniciales son $\dot{\psi}(0) = \ddot{\psi}(0) = \dddot{\psi}(0) = 0$. Esto implica que $x_1(0) = x_2(0) = 0$ y $x_3(0) = \ddot{\psi}(0) - c\dot{\delta}(0) \vee x_3(0) = -c\dot{\delta}(0)$. Para una implementación de tiempo discreto, nosotros simplemente discretizamos las ecuaciones diferenciales.

SEGUNDA PARTE:

DISEÑO Y SIMULACION

1. CONTROLADOR DIFUSO PARA EL CONTROL DE DIRECCIÓN DEL B.A.P. "CARVAJAL"

En esta parte del presente trabajo se diseña un controlador difuso proporcional derivativo, para el control de dirección del Buque de la Armada Peruana Carvajal, el modelo como se analizo en el capitulo anterior es no lineal.

Para criterios de simulación y para obtener los resultados mas reales posibles, se supone que el control esta implementado discretamente y la planta es un sistema continuo.

1.1. Diseño del Controlador Difuso

Hay muchos tipos diferentes de controladores difusos que podríamos examinar para el caso de múltiples entradas y una salida (MISO). Nosotros construiremos dos entradas "controlador difuso proporcional y derivativo" (como es algunas veces llamado). Este controlador, mostrado en la Figura 4.1, es similar a nuestro controlador SISO con la adición de la segunda entrada de/dt. Para nuestra aplicación, las funciones de membresía en los universos de discursos y valores lingüístico NMG, NG, NM, NP, NMP, ZE, PMP, PP, PM, PG, PMG, cuyo significado es negativamente muy grande, negativamente grande, negativamente mediano, negativamente pequeño y negativamente muy pequeño, etc.

El ancho de los conjunto lingüísticos es de 0.4 y el universo discurso es de [-1 1], para la tres variables que intervienen en el proceso que serán llamadas A, B y D (error, derivada del error y salida).

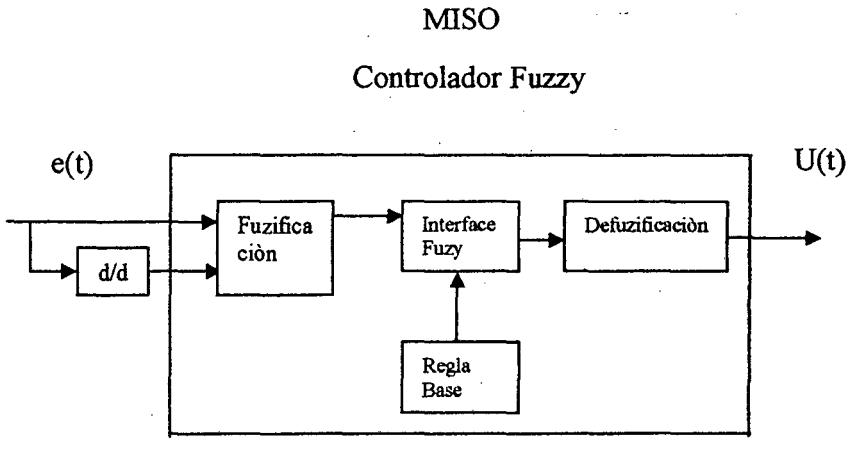

Figura 4.1. Controlador Difuso MISO

Por consiguiente ahora hay tres parámetros para cambiar al controlador difuso: A, B, y D, existen 11 funciones de membresía en cada universo de entrada de discurso, hay 121 reglas posibles que serán agregadas a la base de reglas. Una regla típica asumirá la forma:

IF **e** is NMG AND **de/dt** is NMG THEN **u** is NMG.

El juego completo de reglas es mostrado en forma tabulada en Figura 4.2. En la Figura 4.2 las premisas para la entrada **e** están representados por los valores lingüísticos encontrado en la parte superior de la fila, las premisas de entrada de/dt representan los valores lingüísticos en lado izquierdo de la columna, y el valor lingüístico representa al consecuente pues cada una de las 121 reglas puede ser encontrada en la intersección de la fila y la columna de las premisa apropiada. La parte sombreada de la Figura 4.2 es la representación de la regla citada anteriormente " IF **e** es NMG y **de/dt** es NMG ENTONCES **u** es NMG ". Lo demás del controlador difuso MISO es similar al controlador difuso SISO (La fuzzificación singleton, el producto para la premisa y la implicación difusa y la defuzzificación de centroide son usados).

La interpretación de las reglas para nuestro controlador difuso se codificara numéricamente para nuestro programa.

Negativamente muy grande	-1
Negativamente grande	-0.8
Negativamente mediano	-0.6
Negativamente pequeño	-0.4
Negativamente muy pequeño	-0.2
Zero	0
Positivamente muy pequeño	0.2
Positivamente pequeño	0.4
Positivamente mediano	0.6
Positivamente grande	0.8
Positivamente muy grande	1

Entonces nuestra base de reglas según un controlador PD difuso seria:

```
Reglas = [  1.0  1.0  1.0  1.0  1.0  1.0  0.8  0.6  0.3  0.1  0.0;
            1.0  1.0  1.0  1.0  1.0  0.8  0.6  0.3  0.1  0.0 -0.1;
            1.0  1.0  1.0  1.0  0.8  0.6  0.3  0.1  0.0 -0.1 -0.3;
            1.0  1.0  1.0  0.8  0.6  0.3  0.1  0.0 -0.1 -0.3 -0.6;
            1.0  1.0  0.8  0.6  0.3  0.1  0.0 -0.1 -0.3 -0.6 -0.8;
            1.0  0.8  0.6  0.3  0.1  0.0 -0.1 -0.3 -0.6 -0.8 -1.0;
            0.8  0.6  0.3  0.1  0.0 -0.1 -0.3 -0.6 -0.8 -1.0 -1.0;
            0.6  0.3  0.1  0.0 -0.1 -0.3 -0.6 -0.8 -1.0 -1.0 -1.0;
            0.3  0.1  0.0 -0.1 -0.3 -0.6 -0.8 -1.0 -1.0 -1.0 -1.0;
            0.1  0.0 -0.1 -0.3 -0.6 -0.8 -1.0 -1.0 -1.0 -1.0 -1.0;
            0.0 -0.1 -0.3 -0.6 -0.8 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0]
```

Figura 4.2 Reglas del contador Difuso

Se selecciona un controlador difuso proporcional derivativo (a veces llamado controlador difuso MIMO), este tipo de controladores utilizan técnicas de control convencional para adaptarlas a los sistemas difusos (sistemas híbridos).

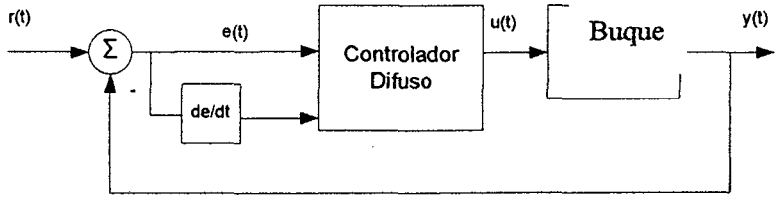

Figura 4.3. Diagrama de Bloques del Sistema Difuso MIMO.

Donde la señal de entrada al controlador difuso es:

e(t) = Error = Dirección deseada – Dirección medida.

de/dt = c (t) = Cambio de error.

La señal de control es:

u(t) = Timón del buque.

Y la variable a controlar es:

y(t) = Dirección de salida.

La planta en la figura 4.3 tiene una entrada u(kT) y una salida y(kT). Las entradas del controlador difuso son generados vía alguna función de salida de la planta y(kT) y la entrada de referencia r(kT). Las entradas del controlador difuso son el error e(kT) = r(kT) – y(kT) y el cambio de error.

$$c(kT) = \frac{e(kT) - e(kT - T)}{T}$$

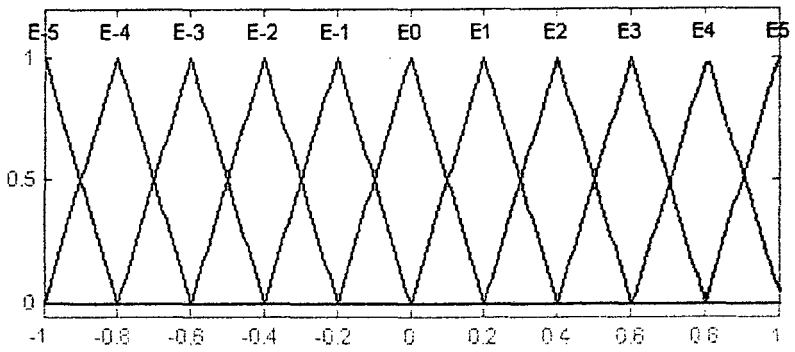

Figura 4.4. Funciones de Membresía

Las funciones de membresía de nuestro controlador difuso se muestran en la figura 4.4, las mismas funciones son para el cambio de error y para la salida del controlador. El tipo de inferencia difusa es de tipo Mamdani y la defusificación es de tipo COA.

1.2. Sintonización de las Funciones de Membresía.

De los muchos métodos existentes nosotros usaremos el de sintonización vía escalamiento del universo discurso ya que nos permite dar un mejor resultado de una forma muy simple.

Supongamos que para un sistema de control clásico tengamos una respuesta que sea inaceptable entonces nosotros tendríamos que cambiar la ganancia proporciona, integral o derivativa como se muestra en la figura 4.5. Entonces nosotros también en nuestro control difuso cambiaremos la ganancia proporcional y derivativa obteniendo el siguiente efecto.

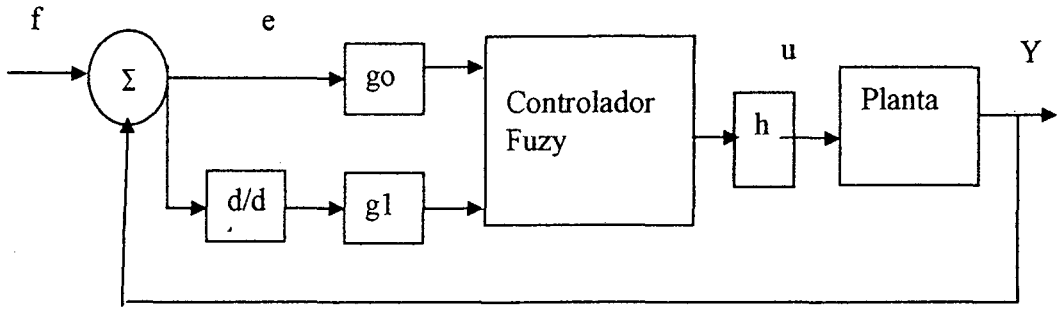

Figura 4.5 Controlador Difuso con Ganancias Escalonadas g0, g1, h

Si tenemos g0, g1 y h = 1 corresponde al mismo controlador difuso sin cambios, el mismo efecto para las ganancias de entrada tienen g0 y g1.

- Si g1 = 1, entonces no tiene efecto en la función de membresía.
- Si g1 < 1, la función de membresía se ensancha uniformemente, esto cambia el significado de la lingüística, por ejemplo "positivamente grande" caracteriza a la función de membresía que representa a los números grandes.
- Si g1 > 1, la función de membresía se contrae uniformemente esto cambia el significado de la lingüística, por ejemplo "positivamente grande" caracteriza a la función de membresía que representa a los números pequeños.

El efecto de la ganancia escalonamiento a la salida:

- Si h = 1, entonces no tiene efecto la función de membresía de salida.
- Si h < 1, la función de membresía se contrae uniformemente esto cambia el significado de la cuantificación lingüística a números pequeños.
- Si h > 1, la función de membresía se ensancha uniformemente, esto cambia el significado de la cuantificación lingüística a números grandes.

Es importante darse cuenta de que las ganancias de escalada no son los únicos parámetros que pueden ser sintonizados para mejorar el desempeño del sistema difuso de control. Ciertamente, algunas veces el caso para una base de reglas dadas y las funciones de membresía usted no puede lograr el desempeño deseado afinando sólo las ganancias de escalada. A menudo, lo que es necesario es una consideración ponderada de cómo especificar reglas adicionales o mejorar la función de membresía.

El problema con esto es que hay a menudo demasiados parámetros para afinar (por ejemplo, las formas de funciones de membresías, posicionamiento, número y tipo de reglas) y a menudo no hay una conexión cristalina entre los objetivos del diseño, mejor tiempo de levantamiento y una justificación razonada y un método que debería estar acostumbrado a afinar estos parámetros.

1.3. Simulación del Controlador Difuso.

En este capitulo mostraremos todos los resultados obtenidos del diseño del controlador difuso.

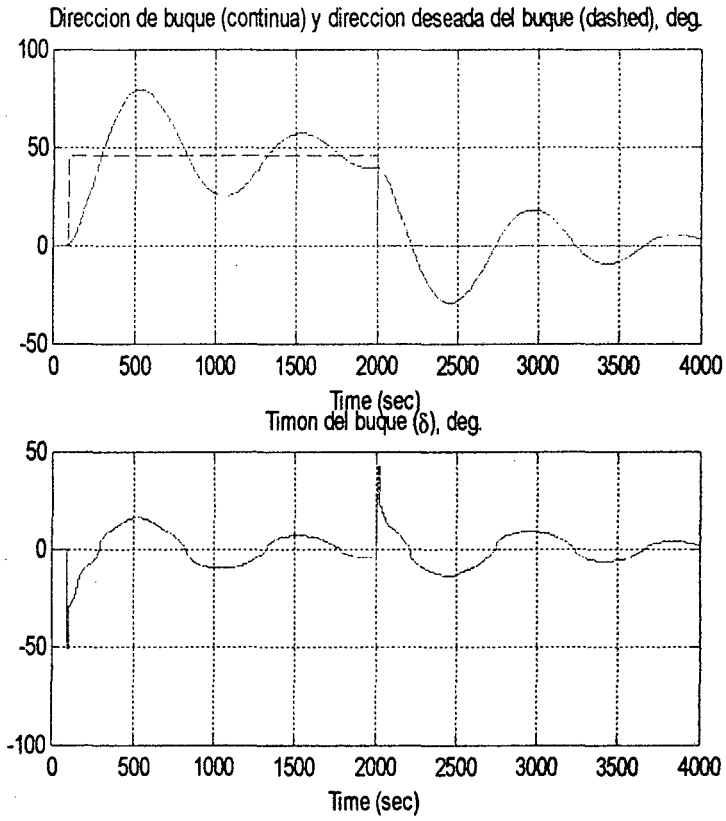

Figura 4.6 Simulación a la Salida del Controlador y la Señal del Controlador del Sistema Difuso sin Sintonización

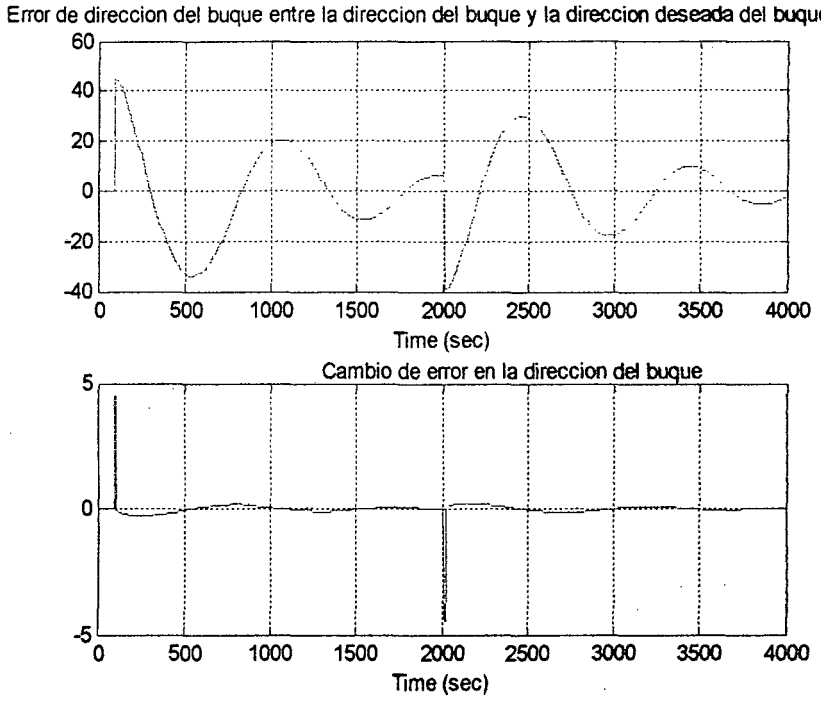

Figura 4.7. Simulación de la Señal de Error y del Cambio de Error del Controlador Difuso sin Sintonización

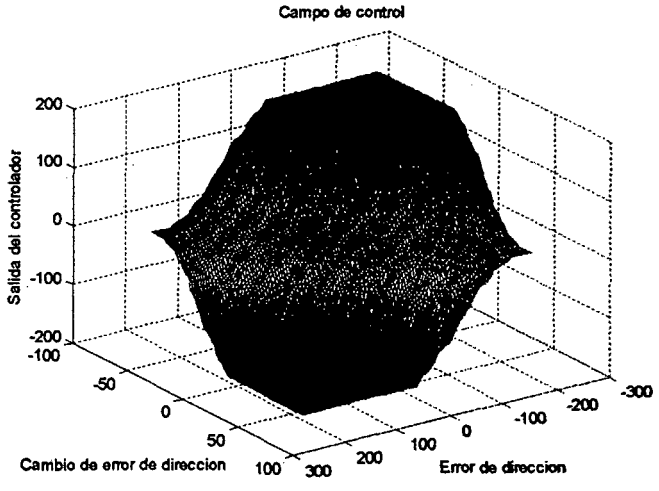

Figura 4.8. Campo del Controlador Difuso sin Sintonización

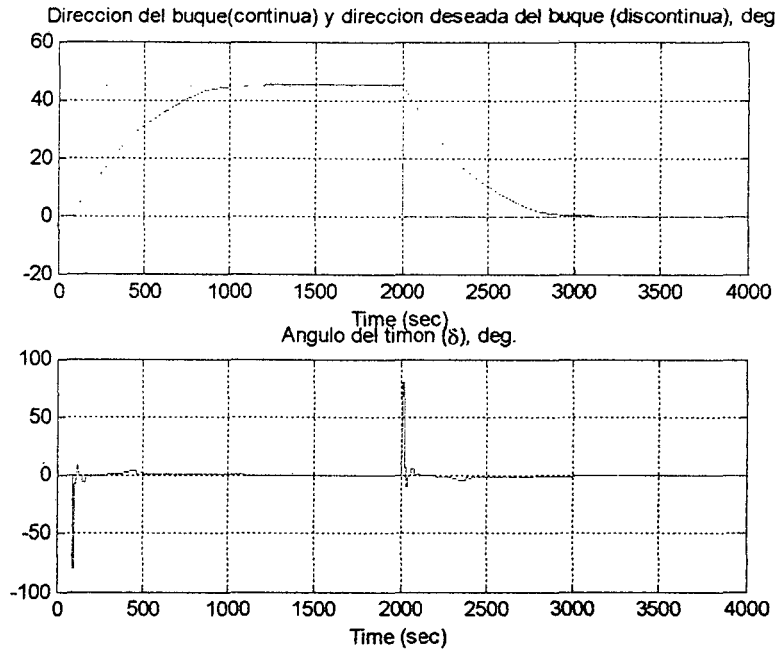

Figura 4.9. Simulación a la Salida del Controlador y la Señal del Controlador del Sistema Difuso Sintonizado

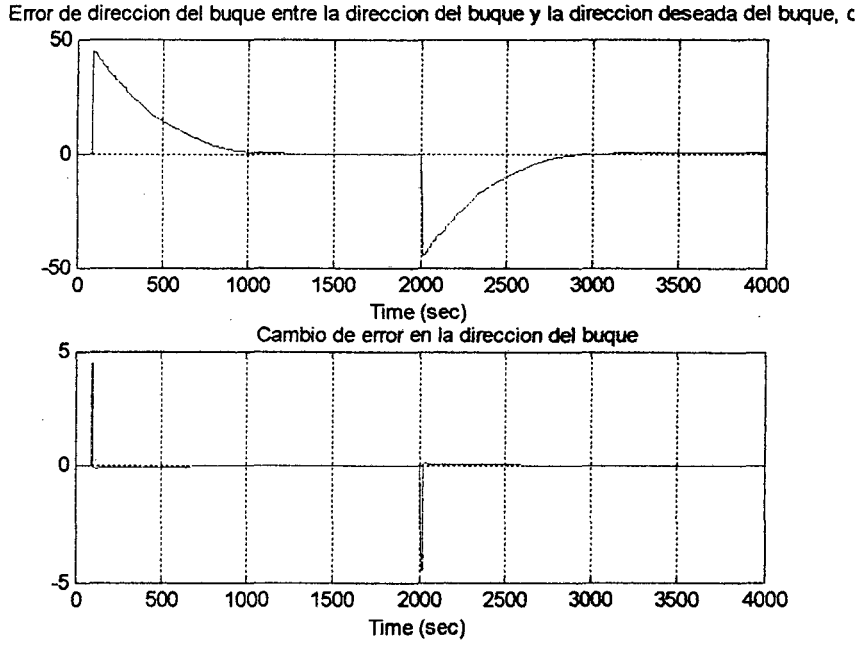

Figura 4.10. Simulación de la Señal de Error y del Cambio de Error del Controlador Difuso Sintonizado

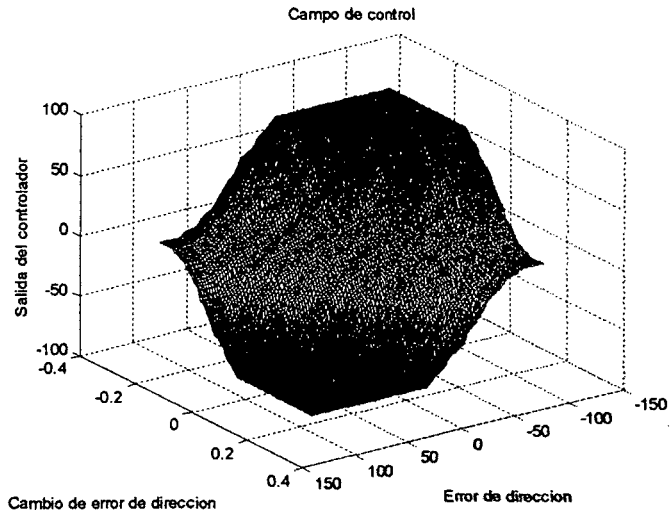

Figura 4.11. Campo del Controlador Difuso Sintonizado

2. CONTROLADOR GENETICO PARA EL CONTROL DE DIRECCIÓN DEL B.A.P. "CARVAJAL"

En esta parte del presente trabajo se diseña un controlador genético proporcional derivativo, para el control de dirección del Buque de la Armada Peruana "Carvajal", el modelo como se analizo en el capitulo anterior es no lineal.

Para criterios de simulación y para obtener los resultados más reales posibles, se supone que el control esta implementado discretamente y la planta es un sistema continuo.

2.1. Diseño del Controlador Genético.

El presente diseño es un controlador genético adaptativo con modelo de referencia. GMRAC que es usado para hallar las ganancias del controlador proporcional derivativo. El GMRAC se muestra en la figura 5.1 usa el modelo de la planta y el algoritmo genético para evolucionar los parámetros del controlador para minimizar el error entre la dirección del buque de guerra y la salida del modelo de referencia.

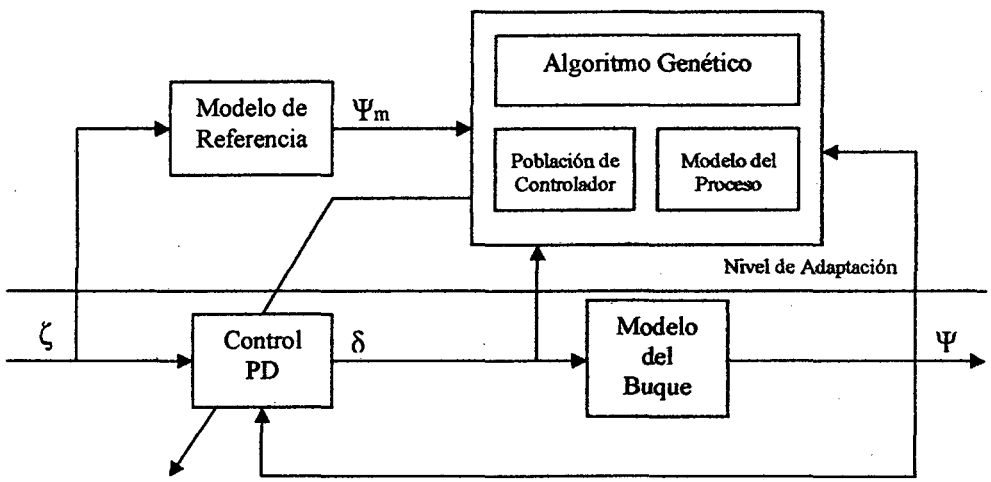

Figura 5.1 GMRAC para la Dirección del Buque

El algoritmo genético (AG) usa el principio de evolución y genética para seleccionar los parámetros del controlador adaptativo. Para nuestra simulación, el conjunto del controlador es usado como la población del algoritmo genético, y el potencial de cada controlador es evaluado para el buque.

Primero, los miembros de la población son definidos, en este caso el controlador PD. Cada controlador individual es definido por diez dígitos o "cromosomas", como se explico en el segundo capitulo. Los primeros cuatro dígitos describe la ganancia proporcional y los últimos seis dígitos describen la ganancia derivativa, las ganancias proporcional y derivativa están definidas entre:

$$-5 < p \leq 0, \qquad -500 < d \leq 0.$$

Por ejemplo, un posible cromosoma es [1234123456]. Este podría trasladarse en ganancia proporcional de p = -1.234, y una ganancia derivativa d = -123.456. Notar que el signo negativo es empleado, entonces el punto decimal en el derivativo es a los tres dígitos y en el proporcional es al primer digito, se diseña de esta manera y con estos limites ya que fueron los que dieron mejor resultado y se acomodaron a las especificaciones de diseño.

Durante cada paso de tiempo, cada miembro de la población es evaluada en como se puede minimizar el error entre la predicción de la repuesta del sistema en lazo cerrado (usando el modelo del buque) y la salida predicha del modelo de referencia.

El siguiente pseudo-código nos precisa la definición de adaptabilidad del AG y la operación del GMRAC:

1. Computar el error y la derivada del error entre la dirección del buque y la entrada de referencia como:

$$e(t) = \zeta(t) - (t)$$

$$\dot{e}(t) = \frac{de}{dt}$$

2. Predecir la salida del modelo de referencia, en el futuro usando una aproximación de primer orden

$$\hat{\psi}_m(t + NT) = \psi_m(t) + NT\dot{\psi}(t)$$

$$\dot{\psi}_m(t) = \frac{\psi_m(t) - \psi_m(t - T)}{T}$$

3. Computar el error entre la dirección del buque de carga y la salida del modelo de referencia.

$$\in (t) = \psi_m(t) - \psi(t)$$

4. Suponer que el i^{th} controlador candidato $C_i = (p_i, d_i)$. para cada controlador candidato C_i hace lo siguiente:

- Determinar la entrada del ángulo del timón usando las ganancias p_i,d_i en el cromosoma del controlador candidato.

$$\delta_i = p_i e(t) + d_i \dot{e}(t)$$

- Inicializar en modelo del buque e tiempo discreto con muestras pasadas del buque.

$$\hat{\psi}_i(k+j) = \psi(t - jT).....,j = 0,1,2$$

$$\delta_i(k-j) = \delta(t - jT).....,j = 0,1,2$$

- Asumiendo que el ángulo del timón δ_i quedan constantes para la siguiente N muestras $\delta_i(k+1) = ... = \delta_i(k+N) = \delta_i$, predice la salida del modelo buque, $\hat{\psi}_i(k+N)$, N es el paso en el futuro.
- Usando la salida del modelo del buque, estima el error entre la dirección del buque y la salida del modelo de referencia NT segundos en el futuro.

$$\hat{\epsilon}_i(t+NT) = \hat{\psi}_m(t+NT) - \hat{\psi}_i(k+N)$$

- Estimar la derivada del error entre la salida del modelo de del buque y la salida del modelo de referencia, usando la aproximación de primer orden.

$$\dot{\epsilon}_i(t) = \frac{\hat{\epsilon}_i(t+NT) - \epsilon(t)}{NT}$$

- Asignar la adaptabilidad, J_i a cada candidato del controlador , C_i:

$$J_i = \epsilon(t) + \beta \dot{\epsilon}_i(t)$$

$$j_i = \frac{\alpha}{j_i^2 + \alpha}$$

5. Repetir el paso 4 para cada miembro de la población.
6. Maximizar el controlador adecuado para transformarse en el controlador usado en el siguiente paso de tiempo.

El valor escogido por α se sitúa superior al limite de la adaptabilidad de J, que debe ser escogida con cuidado, si es muy pequeña α, entonces una diferencia larga de los valores de J que podrían existir en la población (un pequeño cambio en $\bar{J}_i$ podría causar un gran cambio en J_i). En general unos pequeños individuos podrían ser seleccionados para reproducirse en la siguiente generación.

Como sea si α es cambiado a muy largo, entonces todos los miembros de la población podrían aproximadamente igual a los valores de adaptación, y por lo tanto aproximadamente igual oportunidad a reproducirse, por esta razón se hace un compromiso entero de "supervivencia de la adaptabilidad" natural de los algoritmos genéticos. En general α es seleccionado alrededor de una magnitud menor que el promedio del valor de $\bar{J}_i$. Esto aparece al valor de compromiso razonable.

El valor para β es cambiado basado en el performance deseado del sistema y la habilidad de controlar la placa. Heurística menté hablando, β define como rápidamente nosotros podríamos como el error entre la dirección del buque carga y el modelo de referencia llega a cero. Al ver esto, la función de adaptabilidad es maximizada cuando $\bar{J}_i^2$ es minimizado, el cual corresponde cuando $\bar{J}_i = 0$, observando la ecuación para

$$\bar{J}_i = \in(t) + \beta \dot{\in}_i(t),$$

la adaptabilidad es maximizada cuando

$$\dot{\in}_i(t) = \frac{-\in(t)}{\beta}.$$

Por ejemplo si $\in(t) = 1.0$ entonces nosotros tendríamos $\dot{\in}_i(t) = -0.2$, entonces el error seria impulsado a cero, en aproximadamente β = 5 segundos. En este ejemplo el mejor controlador es el que produce $\dot{\in}_i(t)$ cercano a -0.2. Cuando nosotros cambiamos β, lo deseado es que sea lo mas pequeño posible, pero no

es deseable producir oscilaciones, no puede restringirse fisicamente tal como la saturación de la entrada del timón y la dinámica lenta de los sistemas son ignorados.

Una mirada a la ventana de tiempo, N se selecciono a 10 muestras (NT = 5 segundos), como un compromiso de conflictos de interés. En general es bueno hacer N largo, porque a menudo el flujo de la entra de señal no esta fácilmente apreciable a la salida del sistema. Como sea el error entre el modelo predicho de la dirección del buque y como se comporta actualmente el incremento del buque tal como N incremente. En general si se tiene que encontrar una ventana de tiempo de 10 – 20 muestras en tiempo discreto se trabajará muy bien.

Una vez que en cada controlador se le asigna una adaptabilidad de $\bar{J}_i$, el algoritmo genético usa el proceso de selección de la rueda de la ruleta, el cual se escoge el controlador que podría reproducirse en la próxima generación. Los individuos seleccionados para reproducirse son los parientes de la próxima generación. En el proceso de selección de la rueda de la ruleta, la probabilidad de una reproducción individual en la siguiente generación es proporcional a la adaptabilidad de ese individuo. Siendo mas específico, la probabilidad Pp_i es el i^{th} miembro de la población podría ser seleccionado como el J^{th} miembro de la combinación de parientes es

$$P_{pi} = \frac{J_i}{\sum_{k=1}^{N} J_k}$$

Notar que algunos individuos podrían probablemente convertirse mas parientes que otros (indicando que ellos tendrían mas de una descendencia) mientras otros no podrían ser seleccionados del todo. En esta manera los individuos inadecuados son generalmente removidos de la población mientras que la población adecuada se multiplica.

Cada pariente de la siguiente generación tiene que ser seleccionada. Cada par de familias tienen una probabilidad, p_c de perecer "crossover", en el cual algunos dígitos en la familia de cromosomas son intercambiados con otros dígitos de otros cromosomas de familias. Esto es muy común por la selección en locación de un cromosoma (the crossover site) e intercambiando todos los dígitos pasados de ese punto en el cromosoma con los dígitos en la misma locación en el apareo del cromosoma. Como sea el crossover es hecho diferente en todo nuestros algoritmos genéticos en esta investigación. Aquí una de dos familias tiene que ser seleccionadas por crossover, cada digito en el cromosoma tiene 0.5 de probabilidad de comenzar a intercambiar para el dígito en la misma locación de apareo del cromosoma. El digito del crossover se mantiene en su posición original en el cromosoma, y solo intercambiados con el digito en la misma posición del cromosoma. Por ejemplo, si dos familias de cromosomas, [1111111111] y [3333333333], experimentan un "crossover", el resultado cromosomas "hijo" podrían ser [1133313111] y [3311131333]. Notar que el método de crossover, no es convencional, pero nosotros tenemos encontrado que es mas efectivo que el método tradicional cuando mas de dos características

son codificadas en los cromosomas porque permita la posibilidad de dos características o rasgos no adyacentes en un cromosoma a quedarse junto después del crossover. El crossover es una forma de búsqueda local en los parámetros *p,d* de búsqueda espacial. La probabilidad de crossover, *Pc* fué determinado a 0.9.

Después del crossover, cada digito en cada cromosoma hijo tuvo la probabilidad, *Pm*, de mutación. Si un digito es seleccionado para mutar, entonces ese digito es reemplazado por una nueva selección aleatoria del digito. Por ejemplo, un cromosoma [1111111111] puede ser mutada a [111511111]. Notar que un digito puede ser mutado a su valor original, en efecto no comienza a mutar para todo. La probabilidad de mutación, P_m, fue determinado a 0.1 y debería tomarse en cuenta la posibilidad de estas falsas mutaciones. Mutación es una forma de búsqueda global de parámetros *p,d* búsquedas espaciales.

Después de la mutación, el cromosoma hijo llega a ser la siguiente generación del controlador y el proceso es repetido al siguiente paso de tiempo. Una excepción a este proceso es el operador de elitismo. Los pasos son selección, crossover y mutación y simplemente asentar el apropiado controlador de la generación previa en la siguiente generación sin modificación. El elitismo es usado en todos los AGs en esta investigación, los resultados de el GMRAC es mostrado en la siguiente sección. El muy buen performance del controlador cuando el buque de carga tuvo una velocidad de 5 o 7 m/s.

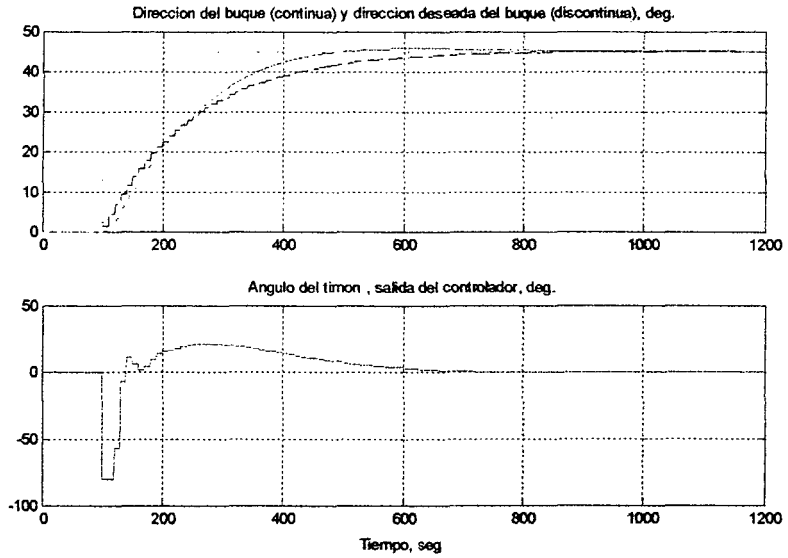

Figura 5.2. (a) Dirección del Buque y Dirección deseada del Buque (b) Angulo del Timón

La figura 5.2, muestra en (a) la comparación entre el rumbo deseado o set point versus el modelo del buque y el modelo de referencia, nótese que el modelo de referencia (línea negra punteada), sigue casi idénticamente al modelo real del sistema aportando así el aprendizaje del sistema. La figura 5.3, muestra el error entre la dirección del buque y el modelo de referencia, llegando a tender a cero que es lo deseado, aproximadamente a los 1000 segundos.

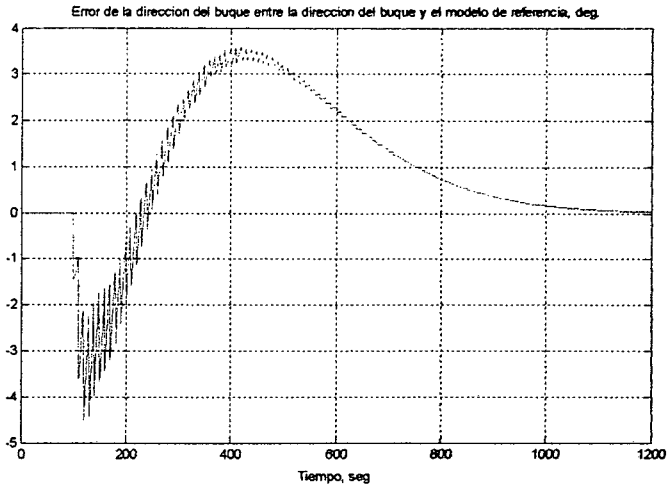

Figura 5.3 Error entre la Dirección del Buque y el Modelo de Referencia

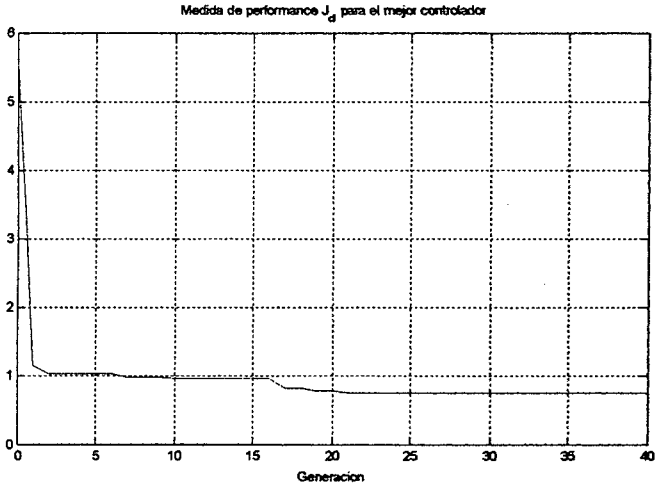

Figura 5.4 Medida del Performance para el mejor Controlador

La figura 5.4 muestra en una generación de 40, el performance hallado es de 0.75, el algoritmo genético paro la búsqueda espacial en este valor, nótese que el valor de J tiene que ser pequeña, normalmente entre cero y uno.

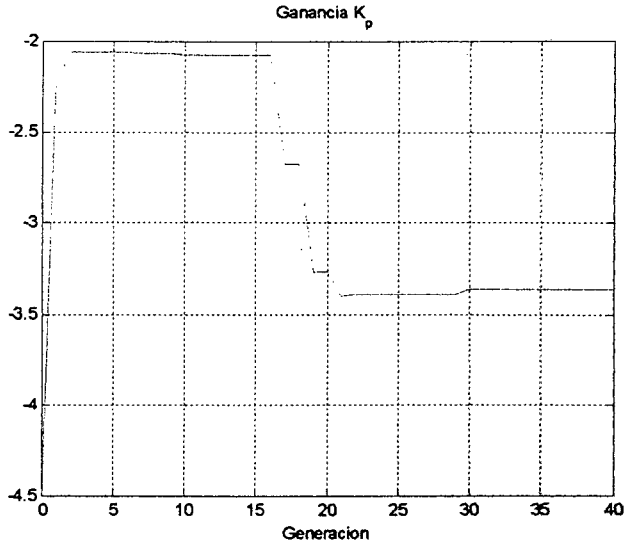

Figura 5.5 Ganancia Kp Encontrada por el AG

Las figuras 5.5 y 5.6 muestras los valores encontrados por el algoritmo genético para Kp = -3.364 y Kd = -500, estos valores son los de mayor performance para este modelo.

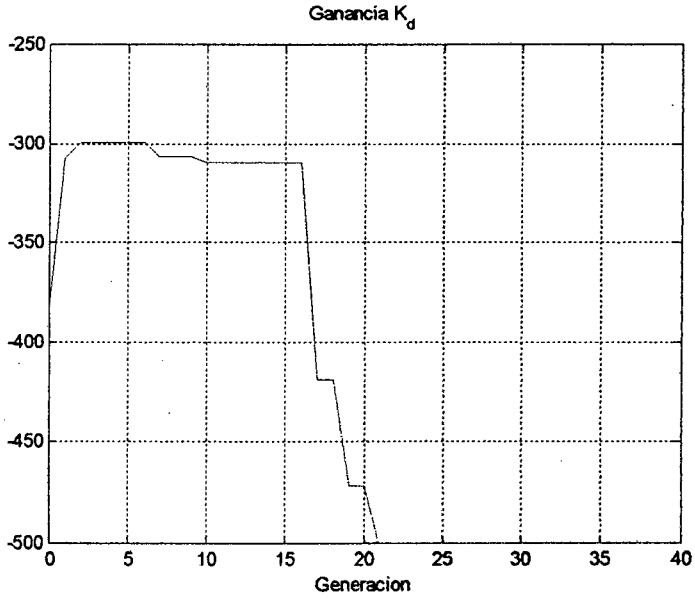

Figura 5.6 Ganancia Kd Encontrada por el AG

3.- COSTO Y BENEFICIOS DEL SISTEMA

3.1.- COSTO PARA REALIZAR LA SIMULACION

- La realización del diseño del controlador ocasiona un gasto horas – hombres por el tiempo empleado en el diseño, análisis, simulación del sistema y por las visitas a las Unidades Navales de la Marina de Guerra del Perú, las horas utilizadas diariamente para realización de la presente tesis es de cuatro horas por un tiempo de 90 días con un costo por día de S/ 40.00 nuevos soles entonces el costo total seria de S/. 3,600.00 nuevos soles

- El costo por capacitación en el curso de programación de MATLAB realizados en la universidad nacional del callao es de S/.120.00 nuevos soles, el software

para desarrollar el diseño, análisis y simulación del sistema, se utilizo el de Universidad Nacional del Callao por el costo que se asumiría si se adquiriera en forma personal dicho software cuyo precio es de $10,000.00 dólares americanos.

- La presente Investigación se realizo con el alquiler de una computadora Pentium IV con acceso a Internet por un tiempo aproximado de 60 días a un costo de S/ 1.00 la hora, se uso 4 horas diarias en esta estación se realizo el diseño, análisis y simulación del sistema, entonces el costo total por alquiler de computadora es de S/. 240.00 nuevos soles

- Tiempo para el desarrollo del proyecto, se estima un periodo de 3 meses para desarrollo del sistema.

  A continuación se detalla el cronograma de trabajo para realizar el proyecto

Item	**Descripción del trabajo a realizar**	**Tiempo**
1	Estudio preliminar del Sistema	40 días
2	Desarrollo del software	30 días
3	Validación software	10 días
4	Depuración de software	10 días

- El gasto que se produjo para la realización de la presenté investigación es de :
  - Gasto horas - hombres S/. 3,600.00
  - Gasto por capacitación S/. 120.00
  - Gasto por alquiler computadora y Internet S/. 240.00

  Total S/. 3,960.00

3.2.- COSTO ESTIMADO DE INSTALACION DEL DISEÑO

- Compra o adquisición de una computadora industrial P IV para la instalación del software de programación cuyo costo es de S/. 15,000.00 dólares americanos.

- Compra o adquisición del software de programación para la instalación de "MATLAB 7.0" para realizar el programa de diseño del control difuso y genético para controlar la dirección del buque B.A.P "CARVAJAL" cuyo costo es de S/. 30,000.00 dólares americanos.

- Diseño e instalación de las interfaces para enlazar la PC con los diversos parámetros que controlan la dirección del Buque cuyo costo estimado es de /. 6,000.00.

- Costo de la mano de obra para instalación del sistema en el buque B.A.P "CARVAJAL" es de S/. 6,000.00 Dólares americanos

- El gasto que se estima para instalación del diseño de la presenté investigación es de :

➢ Computadora Industrial P IV	S/.1,5000.00
➢ Software "MATLAB"	S/. 30,000.00
➢ Interfaces del Sistema	S/. 6,000.00
➢ Mano obra	S/. 6,000.00
Total	S/. 57,000.00

COSTO TOTAL :

➢ Costo para Realizar la Simulación	S/. 3,960.00
➢ Costo de Instalación del Diseño	S/. 57,000.00
Total	S/. 60,960.00

3.3.- BENEFICIOS

A continuación voy a mencionar los beneficios mas importantes:

- Reducción del personal para el control del buque, en un control manual se utiliza 6 personas, para un control automático inteligente solamente se utiliza solamente 1 persona, el sueldo de cada uno es de S/. 1,500.00 nuevos soles mensuales, entonces el ahorro es de S/. 6,000.00 nuevos soles

- Actualmente este proceso es manual, lo cual conlleva a errores y demora en la corrección del rumbo del buque, con el sistema a implementar se logrará una mayor precisión y control en la dirección del buque ya que el procesamiento,

corrección serán en forma automática y el tiempo en llegar a su destino será menor en comparación con el proceso manual por lo tanto se ahorrara combustible (petróleo) en un 5%, en un mes de navegación se usa Aproximadamente 35 barriles de petróleo, cada barril cuesta 420.00 nuevos soles, entonces el ahorro será aproximadamente de S/. 735.00 nuevos soles por navegación, también ocasiona una reducción en la contaminación del medio ambiente.

- El beneficio estimado que se produciría con la presente investigación es el siguiente:
  - Ahorro en personal S/. 7,500.00
  - Ahorro en combustible S/. 735.00
  - **Total** S/. 8,035.00

- B/C = BENEFICIO / COSTO > 1

  B/C = 8,035 / 60,960 > 0.13

  Los ingresos son menores que los egresos, la inversión solo es una vez y partir del octavo mes viene el ahorro por lo tanto el proyecto es rentable, es una inversión a mediano plazo.

CONCLUSIONES

1. En el diseño del controlador difuso para el control de dirección del buque observamos que las ganancias escalonadas influyen exitosamente en la respuesta del sistema, disminuyendo las oscilaciones, dando estabilidad al sistema.
2. Nos apoyamos en el controlador clásico Proporcional Derivativo, para que de acuerdo a la respuesta del escalón unitario, diseñar las reglas del controlador difuso dando buenos resultados.
3. Se utilizo 121 reglas, haciendo un controlador difuso bien selectivo, ya que al hacer 11 conjuntos difusos tenemos un mayor control en cada segmento de la salida del proceso.
4. El controlador genético es mucho mas preciso y menos oscilante que el controlador difuso, pero su tiempo de respuesta es mas lento que el controlador difuso

RECOMENDACIONES

1. El controlador genético no es recomendable para trabajar en MCU o en sistemas que contengan poca memoria y velocidad, debido al mayor tiempo que requiere para la búsqueda del mejor resultado y en evaluar toda una población.
2. Propone la implementación en el BAP "Carvajal", debido a que actualmente no cuenta con un autopiloto inteligente.
3. Proponer un trabajo de investigación para la mejora de este algoritmo y trabajar solo con el resultado dado por el AG (el kp y kd), y compararlo con un PID convencional que posea los mismos valores.
4. Cambiar la ecuación y el orden del modelo de referencia, para comparar los resultados con los trabajados, y así saber si el sistema trabaja mejor con otro modelo.
5. Trabajar el controlador difuso con menos conjuntos difusos y modificar solo las reglas que interactúan con el sistema, y así mejorar el controlador.
6. El caso de fuzzy logic, se recomienda comenzar con ganancias escalonadas de bajo valor y poco a poco ir aumentándolo hasta llegar al valor deseado

BIBLIOGRAFÍA

1. An Introduction to Nonlinear Analysis of Fuzzy Control Systems – David F. Jenkins and Kevin M. Passino.
2. An Introduction to Fuzzy Logic for Practical Applications – Tak Niimura.
3. Fuzzy Control – Kevin M. Passino, Stephen Yurokvich.
4. Fuzzy Logic Toolbox – Roger Jang.
5. Intelligent Control Systems Using Soft Computing Methodologies – Ali Zilouchian, Mo Jamshidi.
6. Redes Neuronales y Sistemas Difusos – Bonifacio Martín del Brío, Alfredo Sanz Molina.
7. J. Layne and K. Passino, "Fuzzy model reference learning control for cargo ship steering," IEEE Control Systems Magazine, vol. 13, pp. 23–34, Dec. 1993.
8. J. Layne and K. Passino, "Fuzzy model reference learning control," in 1st IEEE Conf. onControl Applications, (Dayton, OH), pp. 686–691, Sept. 1992.
9. J. Layne and K. Passino, "Fuzzy model reference learning control," International Journal of Intelligent and Fuzzy Systems, vol. 4, no. 1, pp. 33–47, 1996.
10. J. Layne, K. Passino, and S. Yurkovich, "Fuzzy learning control for anti-skid braking systems," IEEE Trans. on Control Systems Technology, vol. 1, pp. 122–129, June 1993.
11. W. Kwong, K. Passino, E. Laukonen, and S. Yurkovich, "Expert supervision of fuzzy learning systems for fault tolerant aircraft control," Special Issue on Fuzzy Logic in Engineering Applications, Proceedings of the Inst. of Electrical and Electronics Engineers (IEEE), vol. 83, pp. 466–483, March 1995.

12. V. Moudgal, W. Kwong, K. Passino, and S. Yurkovich, "Fuzzy learning control for a flexiblelink robot," IEEE Transactions on Fuzzy Systems, vol. 3, pp. 199–210, May 1995.

13. W. Lennon and K. Passino, "Intelligent control for brake systems," in IEEE Int. Symp. On Intelligent Control, (Monterey, CA), pp. 499–504, Aug. 1995.

APENDICE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sistema de Control Fuzzy para un Buque
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Este programa simula un sistema de control Fuzzy   para un Buque
%Tiene un controlador Fuzzy con dos entradas, el error de
%Dirección del barco (E) y el cambio de error (C). La salida
%del controlador  Fuzzy es la entrada de timón (delta). Queremos
%que la dirección del buque (psi) traquee la referencia de entrada
%de la dirección (psi_r).  Simulamos un  buque como un
%sistema continuo de tiempo que es controlado por un controlador
%Fuzzy que es implementado en una computadora digital con un
%intervalo de muestreo de T.
%Este programa se puede utilizar para ilustrar:
%- Cómo codificar a un controlador Fuzzy  (para dos entradas y una
% salida, ilustrando algunos aproximaciones para  simplificar los
%cómputos, para funciones triangulares de un conjunto variables
%, y  la defuzificacion utilizando el centro de gravedad).
%- Cómo colocar la entrada y las ganancias de salida de un
%controlador Fuzzy.
%- Cómo cambios de condiciones en la planta ("sin carga" y "con carga")
%puede afectar el desempeño.
%-Cómo ruido del sensor (ruido del sensor de cabeceo),
%perturbación de la planta (viento que golpea el lado del Buque),
%y las condiciones en que opera la planta  (la velocidad del
%Buque) puede afectar el desempeño.
%- Como una  elección  impropia de la escalas de ganancias
%puede tener como resultado las oscilaciones ( ciclos  límite).
%-   Cómo una  elección impropia de la escala de ganancias (o regla
%base) puede  tener como resultado un sistema inestable.
%- La forma de la no linealidad aplicado por el controlador fuzzy
%hasta el ploteo del mapa de la entrada-salida del controlador
%Fuzzy. .

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear


ell=350;                  % Longitud del buque  (en metros)
u=5;                      % Velocidad nominal (en m/seg)
%u=3;
abar=1;                   % Parámetros no linealidad
bbar=1;


K_0=5.88;
```

```
tau_10=-16.91;
tau_20=0.45;
tau_30=1.43;


K=K_0*(u/ell);
tau_1=tau_10*(ell/u);
tau_2=tau_20*(ell/u);
tau_3=tau_30*(ell/u);



nume=11;
numc=11;
g1=1/pi;,g2=200;,g0=8*pi/18;

we=0.2*(1/g1);

wc=0.2*(1/g2);

base=0.4*g0;


ce=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g1);


cc=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]*(1/g2);


Rules=

[1     1     1     1     1     1     0.8   0.6   0.3   0.1   0;
1     1     1     1     1     0.8   0.6   0.3   0.1   0     -0.1;
1     1     1     1     0.8   0.6   0.3   0.1   0     -0.1  -0.3;
1     1     1     0.8   0.6   0.3   0.1   0     -0.1  -0.3  -0.6;
1     1     0.8   0.6   0.3   0.1   0     -0.1  -0.3  -0.6  -0.8;
1     0.8   0.6   0.3   0.1   0     -0.1  -0.3  -0.6  -0.8  -1;
0.8   0.6   0.3   0.1   0     -0.1  -0.3  -0.6  -0.8  -1    -1;
0.6   0.3   0.1   0     -0.1  -0.3  -0.6  -0.8  -1    -1    -1;
0.3   0.1   0     -0.1  -0.3  -0.6  -0.8  -1    -1    -1    -1;
0.1   0     -0.1  -0.3  -0.6  -0.8  -1    -1    -1    -1    -1;
0     -0.1  -0.3  -0.6  -0.8  -1    -1    -1    -1    -1    -1] * g0;


flag1=input('\n Ud. desea simuar el \n sistema control fuzy\n para el buque?  \n (escribir
1 para SI y  0 para NO) ');

if flag1==1,
```

```
t=0;
index=1;
tstop=4000;
step=1;
T=10;
counter=10;

eold=0;

x=[0;0;0];

x(1)=0;

x(2)=0;

psi_r_old=0;

while t <= tstop

if t<100, psi_r(index)=0; end
if t>=100, psi_r(index)=45*(pi/180); end
if t>2000, psi_r(index)=0; end
%
s(index)=0;

psi(index)=x(1)+s(index);

if counter == 10,

counter=0;

c_count=0;,e_count=0;

e(index)=psi_r(index)-psi(index);


c(index)=(e(index)-eold)/T;

eold=e(index);


        if e(index)<=ce(1)

      mfe=[1 0 0 0 0 0 0 0 0 0 0];

        e_count=e_count+1;,e_int=1;

        elseif e(index)>=ce(nume)

        mfe=[0 0 0 0 0 0 0 0 0 0 1];
```

```
        e_count=e_count+1;,e_int=nume;


else

          for i=1:nume
                if e(index)<=ce(i)
                 mfe(i)=max([0 1+(e(index)-ce(i))/we]);


                        if mfe(i)~=0


                         e_count=e_count+1;
                         e_int=i;

                        end
                else
                 mfe(i)=max([0,1+(ce(i)-e(index))/we]);


                        if mfe(i)~=0
                         e_count=e_count+1;
                         e_int=i;

                        end
                end
               end
    end


    if c(index)<=cc(1)
  mfc=[1 0 0 0 0 0 0 0 0 0 0];
     c_count=c_count+1;
     c_int=1;
    elseif c(index)>=cc(numc)

     mfc=[0 0 0 0 0 0 0 0 0 0 1];
     c_count=c_count+1;
     c_int=numc;
    else
            for i=1:numc
             if c(index)<=cc(i)
              mfc(i)=max([0,1+(c(index)-cc(i))/wc]);
                        if mfc(i)~=0
                         c_count=c_count+1;
                         c_int=i;

  end
                else
```

```
                mfc(i)=max([0,1+(cc(i)-c(index))/wc]);
                    if mfc(i)~=0
                     c_count=c_count+1;
                     c_int=i;

                    end
                end
                end
        end


num=0;
den=0;
        for k=(e_int-e_count+1):e_int

                for l=(c_int-c_count+1):c_int

                 prem=min([mfe(k) mfc(l)]);


       num=num+rules(k,l)*base*(prem-(prem)^2/2);
                 den=den+base*(prem-(prem)^2/2);

    end
        end

  delta(index)=num/den;

else

e(index)=e(index-1);
c(index)=c(index-1);
delta(index)=delta(index-1);

end


if t = 0, x(3)=-(K*tau_3/(tau_1*tau_2))*delta(index); end

        time(index)=t;

if delta(index) >= 80*(pi/180), delta(index)=80*(pi/180); end
if delta(index) <= -80*(pi/180), delta(index)=-80*(pi/180); end

%Se da inicio la simulación de la planta, calculando datos

F=[ x(2) ;
   x(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(x(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
     (1/(tau_1*tau_2))*(abar*x(2)^3 + bbar*x(2)) + (K/(tau_1*tau_2))*delta(index) ];
```

```
        k1=step*F;
        xnew=x+k1/2;

F=[ xnew(2) ;
   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

        k2=step*F;
        xnew=x+k2/2;

F=[ xnew(2) ;
   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

        k3=step*F;
        xnew=x+k3;

F=[ xnew(2) ;
   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

        k4=step*F;
        x=x+(1/6)*(k1+2*k2+2*k3+k4);


t=t+step;
index=index+1;
counter=counter+1;

end

psi_r=psi_r*(180/pi);
psi=psi*(180/pi);
delta=delta*(180/pi);
e=e*(180/pi);
c=c*(180/pi);

figure(1)
clf
subplot(211)
plot(time,psi,'k-',time,psi_r,'k--')
grid on
xlabel('Tiempo (sec)')
```

```
title('dirección del buque (continua ) and dirección deseada del buque (discontinua),
deg.')
subplot(212)
plot(time,delta,'k-')
grid on
xlabel('Time (sec)')
title('Rudder angle (\delta), deg.')
zoom

figure(2)
clf
subplot(211)
plot(time,e,'k-')
grid on
xlabel('Tiempo (sec)')
title('error de dirección del buque entre la dirección del buque y dirección deseada del
buque, deg.')
subplot(212)
plot(time,c,'k-')
grid on
xlabel('Tiempo (sec)')
title('cambio de error en la dirección del buque, deg./sec')
zoom

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Siguiente, proporciona un ploteo de la superficie del controlador fuzzy
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

flag2=input('\n Ud. desea ver un trazo no lineal \n ejecutado por el   fuzzy \n
controlador? \n (escriba  1 para si y 0 para no) ');

if flag2==1,


e_input=(-(1/g1)-0.2*(1/g1)):(1/100)*(((1/g1)+0.2*(1/g1))-(-(1/g1)-...
   0.2*(1/g1))):((1/g1)+0.2*(1/g1));
ce_input=(-(1/g2)-0.2*(1/g2)):(1/100)*(((1/g2)+0.2*(1/g2))-(-(1/g2)-...
   0.2*(1/g2))):((1/g2)+0.2*(1/g2));
for jj=1:length(e_input)
        for ii=1:length(ce_input)

c_count=0;,e_count=0;
        if e_input(jj)<=ce(1)
      mfe=[1 0 0 0 0 0 0 0 0 0 0];
         e_count=e_count+1;,e_int=1;
        elseif e_input(jj)>=ce(nume)
         mfe=[0 0 0 0 0 0 0 0 0 0 1];
         e_count=e_count+1;,e_int=nume;
```

```
    else
      for i=1:nume
            if e_input(jj)<=ce(i)
             mfe(i)=max([0 1+(e_input(jj)-ce(i))/we]);


           if mfe(i)~=0
                  e_count=e_count+1;
                  e_int=i;
                 end
            else
             mfe(i)=max([0,1+(ce(i)-e_input(jj))/we]);
                                                          %
                  if mfe(i)~=0
                   e_count=e_count+1;
                   e_int=i;
                  end
            end
           end
    end


     if ce_input(ii)<=cc(1)
    mfc=[1 0 0 0 0 0 0 0 0 0 0];
     c_count=c_count+1;
     c_int=1;
     elseif ce_input(ii)>=cc(numc)

     mfc=[0 0 0 0 0 0 0 0 0 0 1];
     c_count=c_count+1;
     c_int=numc;
     else
           for i=1:numc
            if ce_input(ii)<=cc(i)
             mfc(i)=max([0,1+(ce_input(ii)-cc(i))/wc]);
                  if mfc(i)~=0
                   c_count=c_count+1;
                   c_int=i;
                  end
            else
             mfc(i)=max([0,1+(cc(i)-ce_input(ii))/wc]);
                  if mfc(i)~=0
                   c_count=c_count+1;
                   c_int=i;
                  end
            end
           end
    end

num=0;
```

```
den=0;
        for k=(e_int-e_count+1):e_int

                for l=(c_int-c_count+1):c_int

                  prem=min([mfe(k) mfc(l)]);

                  num=num+rules(k,l)*base*(prem-(prem)^2/2);
                  den=den+base*(prem-(prem)^2/2);

                end
        end

  delta_output(ii,jj)=num/den;

        end
end

delta_output=delta_output*(180/pi);
e_input=e_input*(180/pi);
ce_input=ce_input*(180/pi);

%
figure(3)
clf
surf(e_input,ce_input,delta_output);
view(145,30);
colormap(white);
xlabel('Error dirección (e), deg.');
ylabel('Cambio de error de dirección  (c), deg.');
zlabel('Salida controlador fuzy (\delta), deg.');
title('Trazo del controlador fuzy entre entrada y salida');

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fin de Programa %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Control Proporcional – Derivativo para regular la Dirección de un Buque
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear                                   % Borrar todas las variables de la memoria

% inicializar parametros del buque

ell=350;                                % Longitud del buque  (en metros)
u=5;                                    % Velocidad nominal (en m/seg)
abar=1;                                 % Parámetros no linealidad
bbar=1;

% Define el modelo de referencia (utilizamos la función de transferencia de primer
% orden k_r/(s+a_r)):

a_r=1/150;
k_r=1/150;

% numero de etapas de evolución
Nevsteps=40;

% tamaño de la población
S=4;

Kpmin=-5;
Kpmax=0;

Kdmin=-500;
Kdmax=-0;

KpKd=0*ones(2,S,Nevsteps);
for ss=1:S

        KpKd(1,ss,1)=Kpmin*rand;
        KpKd(2,ss,1)=Kdmin*rand;
end



Jcl=0*ones(S,Nevsteps);


w1=1;
w2=0.01;
```

```
beta1=0.5;
beta2=50;


pm=1;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inicio del lazo evolutivo del diseño
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for k=1:Nevsteps+1


        for ss=1:S


% Simula al controlador que regula la dirección del buque

% Inicializamos la simulación:

t=0;                    % Resetear el tiempo a cero
index=1;                % Índice del tiempo.
tstop=1200;             % Tiempo de parada de la simulación (en segundos) –
                        % Normalmente 20000
step=1;                 % Tamaño de paso integración
T=10;
counter=10;
eold=0;
cold=0;
psi_r_old=0;
ymold=0;

x=[0;0;0];
x(1)=0;
x(2)=0;




psi_r=0*ones(1,tstop+1);
psi=0*ones(1,tstop+1);
e=0*ones(1,tstop+1);
c=0*ones(1,tstop+1);
s=0*ones(1,tstop+1);
```

```
w=0*ones(1,tstop+1);
delta=0*ones(1,tstop+1);
ym=0*ones(1,tstop+1);

while t <= tstop


if t>=0, psi_r(index)=0; end
if t>=100, psi_r(index)=45*(pi/180); end
if t>=1500, psi_r(index)=0; end
if t>=3000, psi_r(index)=45*(pi/180); end
if t>=4500, psi_r(index)=0; end
if t>=6000, psi_r(index)=45*(pi/180); end
if t>=7500, psi_r(index)=0; end
if t>=9000, psi_r(index)=45*(pi/180); end
if t>=10500, psi_r(index)=0; end
if t>=12000, psi_r(index)=45*(pi/180); end
if t>=13500, psi_r(index)=0; end
if t>=15000, psi_r(index)=45*(pi/180); end
if t>=16500, psi_r(index)=0; end
if t>=18000, psi_r(index)=45*(pi/180); end
if t>=19500, psi_r(index)=0; end


s(index)=0;

psi(index)=x(1)+s(index);

if counter == 10,

counter=0;

ym(index)=(1/(2+a_r*T))*((2-a_r*T)*ymold+...
                         k_r*T*(psi_r(index)+psi_r_old));

ymold=ym(index);
psi_r_old=psi_r(index);


e(index)=psi_r(index)-psi(index);     % computa el error
c(index)=(e(index)-eold)/T;           % colocar el valor de  c

eold=e(index);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Un controlador proporcional - derivativo:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta(index)=KpKd(1,ss,k)*e(index)+KpKd(2,ss,k)*c(index);

else
e(index)=e(index-1);
c(index)=c(index-1);
delta(index)=delta(index-1);

ym(index)=ym(index-1);

end

        time(index)=t;

if delta(index) >= 80*(pi/180), delta(index)=80*(pi/180); end
if delta(index) <= -80*(pi/180), delta(index)=-80*(pi/180); end

u=5;

if flag==0,

K_0=0.83;
tau_10=-2.88;
tau_20=0.38;
tau_30=1.07;

else

K_0=5.88;                    % Estos son los parámetros bajo condiciones de "peso"
tau_10=-16.91;
tau_20=0.45;
tau_30=1.43;

end

%Los siguientes parámetros se utilizan en la definición del modelo del
%Buque:

K=K_0*(u/ell);
tau_1=tau_10*(ell/u);
tau_2=tau_20*(ell/u);
tau_3=tau_30*(ell/u);
```

```
% Después, viene la planta:
% Ahora, para el primer paso, nosotros ponemos la condición
% inicial para el tercer estado x(3).

if t==0, x(3)=-(K*tau_3/(tau_1*tau_2))*delta(index); end

% Siguiente, utilizamos las fórmulas para aplicar el método de Runge-Kutta

F=[ x(2) ;

   x(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(x(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*x(2)^3 + bbar*x(2)) + (K/(tau_1*tau_2))*delta(index) ];

         k1=step*F;
         xnew=x+k1/2;

F=[ xnew(2) ;

   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

         k2=step*F;
         xnew=x+k2/2;

F=[ xnew(2) ;

   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

         k3=step*F;
         xnew=x+k3;

F=[ xnew(2) ;

   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

         k4=step*F;
         x=x+(1/6)*(k1+2*k2+2*k3+k4); % calculando el siguiente estado


t=t+step;
index=index+1;
```

```
        counter=counter+1;

    end .


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calcula cuán bien es el  controlador en términos de  desviación del modelo de
% Referencia y el control de energía
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Jcl(ss,k)=w1*(psi-ym)*(psi-ym)'+w2*delta*delta';

end                     % La evaluación final de cada controlador en la población


[Jbest(k),bestone(k)]=min(Jcl(:,k));
Kpbest(k)=KpKd(1,bestone(k),k);
Kdbest(k)=KpKd(2,bestone(k),k);


KpKd(:,bestone(k),k+1)=KpKd(:,bestone(k),k); e

for ss=1:S
    if ss ~= bestone(k)

        KpKd(:,ss,k+1)=KpKd(:,bestone(k),k+1)+[beta1*randn; beta2*randn];


        if KpKd(1,ss,k+1)<Kpmin, KpKd(1,ss,k+1)=Kpmin; end
        if KpKd(1,ss,k+1)>Kpmax, KpKd(1,ss,k+1)=Kpmax; end
        if KpKd(2,ss,k+1)<Kdmin, KpKd(2,ss,k+1)=Kdmin; end
        if KpKd(2,ss,k+1)>Kdmax, KpKd(2,ss,k+1)=Kdmax; end

    end
end


if pm>rand,
        if bestone(k) ~= 1

            KpKd(1,1,k+1)=Kpmin*rand;
            KpKd(2,1,k+1)=Kdmin*rand;
        else
            KpKd(1,S,k+1)=Kpmin*rand;
            KpKd(2,S,k+1)=Kdmin*rand;
        end
end


end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Después, nosotros proporcionamos la grafica de la entrada y salida del buque junto
%con la referencia de la dirección, sobre esto queremos traquear para mejor diseño
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


flag=0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simule el controlador que regule la dirección del Buque
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Después, inicializamos la simulación

t=0;

index=1;

tstop=1200;

step=1;

T=10;

counter=10;

eold=0;

cold=0;

psi_r_old=0;
ymold=0;

x=[0;0;0];
x(1)=0;
x(2)=0;


psi_r=0*ones(1,tstop+1);
psi=0*ones(1,tstop+1);
e=0*ones(1,tstop+1);
c=0*ones(1,tstop+1);
s=0*ones(1,tstop+1);
w=0*ones(1,tstop+1);
delta=0*ones(1,tstop+1);
ym=0*ones(1,tstop+1);

while t <= tstop
```

```
if t>=0, psi_r(index)=0; end
if t>=100, psi_r(index)=45*(pi/180); end
if t>=1500, psi_r(index)=0; end
if t>=3000, psi_r(index)=45*(pi/180); end
if t>=4500, psi_r(index)=0; end
if t>=6000, psi_r(index)=45*(pi/180); end
if t>=7500, psi_r(index)=0; end
if t>=9000, psi_r(index)=45*(pi/180); end
if t>=10500, psi_r(index)=0; end
if t>=12000, psi_r(index)=45*(pi/180); end
if t>=13500, psi_r(index)=0; end
if t>=15000, psi_r(index)=45*(pi/180); end
if t>=16500, psi_r(index)=0; end
if t>=18000, psi_r(index)=45*(pi/180); end
if t>=19500, psi_r(index)=0; end

s(index)=0;

psi(index)=x(1)+s(index);

if counter == 10,  r

counter=0;



ym(index)=(1/(2+a_r*T))*((2-a_r*T)*ymold+...
                         k_r*T*(psi_r(index)+psi_r_old));

ymold=ym(index);
psi_r_old=psi_r(index);



e(index)=psi_r(index)-psi(index);
c(index)=(e(index)-eold)/T;

eold=e(index);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Un controlador proporcional y derivativo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


delta(index)=KpKd(1,bestone(Nevsteps),Nevsteps)*e(index)+KpKd(2,bestone(Nevstep
s),Nevsteps)*c(index);
else

e(index)=e(index-1);
c(index)=c(index-1);
delta(index)=delta(index-1);

ym(index)=ym(index-1);

end

        time(index)=t;


if delta(index) >= 80*(pi/180), delta(index)=80*(pi/180); end
if delta(index) <= -80*(pi/180), delta(index)=-80*(pi/180); end


u=5;

%end


if flag==0,
K_0=0.83;
tau_10=-2.88;
tau_20=0.38;
tau_30=1.07;

else

K_0=5.88;
tau_10=-16.91;
tau_20=0.45;
tau_30=1.43;

end


K=K_0*(u/ell);
tau_1=tau_10*(ell/u);
tau_2=tau_20*(ell/u);
tau_3=tau_30*(ell/u);
```

```
if t==0, x(3)=-(K*tau_3/(tau_1*tau_2))*delta(index); end

F=[ x(2) ;
   x(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(x(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*x(2)^3 + bbar*x(2)) + (K/(tau_1*tau_2))*delta(index) ];

        k1=step*F;
        xnew=x+k1/2;

F=[ xnew(2) ;
   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

        k2=step*F;
        xnew=x+k2/2;

F=[ xnew(2) ;
   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

        k3=step*F;
        xnew=x+k3;

F=[ xnew(2) ;
   xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index) ;
   -((1/tau_1)+(1/tau_2))*(xnew(3)+ (K*tau_3/(tau_1*tau_2))*delta(index))-...
      (1/(tau_1*tau_2))*(abar*xnew(2)^3 + bbar*xnew(2)) +
(K/(tau_1*tau_2))*delta(index) ];

        k4=step*F;
        x=x+(1/6)*(k1+2*k2+2*k3+k4); % Calculated next state


t=t+step;                    % Incrementa el tiempo
index=index+1;               % Incrementa el termino incluye un índice de modo que .
                             % índice =1  tiempo = 0

counter=counter+1;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Después, proporcionamos la grafica de  los datos de la simulación
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% First, we convert from rad. to degrees
psi_r=psi_r*(180/pi);
psi=psi*(180/pi);
delta=delta*(180/pi);
e=e*(180/pi);
c=c*(180/pi);
ym=ym*(180/pi);

% después , proporcionamos los ploteos

figure(1)
clf
subplot(211)
plot(time,psi,'k-',time,ym,'k--',time,psi_r,'k-.')
zoom
grid on
title('direccion del buque (continua) y direccion deseada del buque (discontinua), deg.')
subplot(212)
plot(time,delta,'k-')
zoom
grid on
title('ángulo del timón, salida del controlador (entrada del buque), deg.')
xlabel('tiempo, sec')

figure(2)
clf
plot(time,psi-ym,'k-')
zoom
grid on
title('eroor direccion del buque entre direccion del buque y modelo de referencia , deg.')
xlabel('Time, sec')

figure(3)
clf
plot(0:Nevsteps,Jbest,'k-')
zoom
grid on
title('Medida de performance')
xlabel('Generacion')

figure(4)
clf
plot(0:Nevsteps,Kpbest,'k-')
zoom
grid on
title('Ganancia Kp ')
```

```
xlabel('Generacion')


figure(5)
clf
plot(0:Nevsteps,Kdbest,'k-')
zoom
grid on
title(`Ganancia Kd ')
xlabel('Generacion')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fin de programa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```