UNIVERSIDAD NACIONAL DEL CALLAO

ESCUELA DE POSGRADO

SECCIÓN DE POSGRADO DE LA FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

"ALGORITMOS GENÉTICOS PARA LA OPTIMIZACIÓN DEL CONTROLADOR PID APLICADO AL SISTEMA PELOTA Y ARO"

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS DE LA ELECTRÓNICA
MENCIÓN: CONTROL Y AUTOMATIZACIÓN

AUTOR: ING. ELMER JAVIER CÓRDOVA ZAPATA

CALLAO – PERU

2011

JURADO SUSTENTACIÓN DE TESIS DE MAESTRÍA

DR. JUAN HERBER GRADOS GAMARRA	Presidente
MG. NICANOR RAÚL BENITES SARAVIA	Secretario
MG. FRANCO IVÁN VÉLIZ LIZÁRRAGA	Miembro
DR. MARCELO NEMESIO DAMAS NIÑO	Miembro
MG. RICARDO RAÚL RODRÍGUEZ BUSTINZA	Asesor

Nº DE LIBRO: 01

Nº DE ACTA : 09

FECHA: junio 9, 2011.

RESOLUCIÓN DE LA SECCIÓN POSGRADO DE LA FIEE Nº 041-2011-DSPG-FIEE

DEDICATORIA

La presente Tesis la dedico a mi madre por su apoyo permanente que me impulsó siempre a no renunciar nunca a mis sueños y aspiraciones personales y a mis hijos por ser el aliento constante de cada día de mi vida.

Un especial agradecimiento a los docentes de la Maestría en Ciencias de la Electrónica, mención en Control y Automatización, de la Universidad Nacional del Callao, que con la mística de una enseñanza exigente, reconocida por todos, me entregaron las herramientas del conocimiento, necesarias para mi desenvolvimiento profesional.

ÍNDICE

CAPÍTULO II

CAPÍTULO III

PRÓLOGO

Durante los últimos años muchas técnicas diferentes han sido desarrolladas para obtener los parámetros óptimos, proporcionales, integrales y derivativos de control para controladores PID. Este trabajo de tesis describe la aplicación de algoritmos genéticos a la sintonía óptima de los tres términos para los controladores clásicos PID para ser utilizada para regular procesos no lineales. El uso de algoritmos genéticos en este campo de optimización del controlador PID espera vencer las debilidades de otros enfoques convencionales en situaciones no lineales.

Los algoritmos genéticos son muy efectivos en encontrar soluciones óptimas a una variedad de problemas. Esta técnica innovadora resuelve los problemas complejos porque no impone muchas limitaciones de técnicas tradicionales. Los algoritmos genéticos son los métodos globales estocásticos de la búsqueda que son basados en la evolución biológica natural. Los algoritmos genéticos operan en una población inicial de soluciones de búsqueda que aplican el principio de la ley del más fuerte para producir mejores soluciones. En cada generación un nuevo conjunto de soluciones es creado y tendrá idealmente valores más ajustados a las soluciones. El ajuste se refiere a como una solución actúa en el dominio del problema. Este proceso lleva a la evolución de poblaciones de individuos que son convenidos al ambiente.

El presente trabajo de tesis se ha dividido en 5 capítulos como serán

detallados a continuación:

Capítulo I: Se presenta el planteamiento inicial de la investigación, identificación del problema, formulación del problema, objetivos de la investigación, justificación, limitaciones, facilidades y la hipótesis de partida.

Capítulo II: Se describe el modelado del sistema pelota y aro, los controladores PID y los algoritmos genéticos.

Capítulo III: Se presenta la metodología de la investigación, relación entre las variables de la investigación, operacionalización de variables, tipo de investigación, diseño de la investigación, etapas de la investigación, población y muestra, técnicas e instrumentos de recolección de datos y procedimiento estadístico y análisis de datos.

Capítulo IV: Este capítulo se presentan los resultados parciales y finales de las simulaciones del controlador PID óptimo convencional y del controlador PID basado en algoritmos genéticos.

Capítulo V: Se presentan los resultados de la simulación, midiendo diferentes entes respuestas basadas en los índices de performances (IAE, MSE, ITAE y ISE) los que logran ser funciones de evaluación, se incorporan otras funciones que determinan el flujo: población inicial, medidas de ajustes, selección, mutación, cruce y soluciones óptimas.

Además esta tesis presenta una serie de conclusiones y recomendaciones obtenidas en base a los resultados de las simulaciones, así mismo se presenta la bibliografía requerida. Finalmente en el anexo A se encuentra la matriz de consistencia y en el anexo B se listan los códigos de los programas en Matlab.

RESUMEN

El proyecto de Tesis de Maestría tiene como finalidad crear un controlador PID óptimo para el sistema pelota y aro que consiste en una pelota de acero que está libre rodando dentro de un aro circular. Este sistema ilustra la dinámica compleja de líquido salpicado, es decir la manera como se comporta el líquido en un contenedor móvil. El aparato de la pelota y el aro es difícil de controlar utilizando con técnicas tradicionales, se requiere el diseño de controlador que optimice parámetros de un controlador de PID porque los parámetros de sistema cambian constantemente. Se propuso el estudio de optimización del controlador PID por ser èste un controlador ampliamente usado en la industria. Para afinar de nuevo el controlador del sistema es necesario tomar el sistema fuera de línea. El propósito del controlador PID es encontrar como resultado un mejor desempeño general del sistema. El controlador será sintonizado utilizando inteligencia artificial, básicamente, los algoritmos genéticos. La aplicación de algoritmos genéticos a la optimización del controlador PID es de gran importancia ya que se considera como una técnica de sintonía para procesos que son difíciles de sintonizar.

Los algoritmos genéticos es un método global de búsqueda estocástica que imita el proceso de la evolución natural. Los algoritmos genéticos han demostrado que son capaces de dar soluciones en dominios complejos sin

experimentar las dificultades que pueden asociarse con la dimensión alta u óptima como suelen ocurrir con las técnicas basadas en gradiente descendente. Utilizar algoritmos genéticos para realizar la sintonía del controlador tiene como resultado un controlador óptimo que puede ser evaluado cada vez. Para el estudio de los algoritmos genéticos se decidió crear una función objetivo de optimización que evalùe las ganancias en forma óptima del controlador PID basado en los sistemas de control de error global. Se realizarán pruebas de sintonía usando métodos convencionales como son las prácticas estándar de ajuste, dada por Ziegler Nichols, y técnicas de optimización como son, IAE (Integral Absolute Error), ITAE (Integral Time Absolute Error), para el diseño de controladores PID. Se realizarán pruebas de identificación basadas en algoritmos recursivos basadas en el Error Cuadrático Medio (MSE) la que produce un desempeño más eficaz en controladores PID, en comparación con las técnicas convencionales. El sistema pelota y aro puede ser sometido a pruebas de algoritmos recursivos de tal forma que se alcance su función de transferencia. Un estimador recursivo de mínimos cuadrados es creado para estimar el sistema en línea y proporcionar la estimación más precisa del sistema para que el algoritmo genético diseñe un controlador PID más óptimo.

ABSTRACT

The Mastery Thesis project has as purpose to create a controller PID optimum for the system ball and hoop that consists of a ball of steel that is free rolling inside of a hoop to circulate. This system illustrates the dynamics complex of liquid sprinkled, that is to say the way as behaves the liquid in a mobile tenant. The apparatus of the ball and the hoop is difficult to control with traditional techniques, is required of a design of controller that optimize parameters of a controller of PID because the parameters of system change constantly. The study of optimization of the controller was proposed PID by being this a controller extensively used in the industry. To tune up again the controller of the system is necessary for take the system off-line. The purpose of the controller PID is to find as a result the best general performance of the system. The controller will be tuned in utilizing artificial intelligence, basically, the genetic algorithms. The application of genetic algorithms to the optimization of the controller PID is of great importance since is considered like a technique of tuning for processes that are difficult to tune in.

The genetic algorithms are a global method of stochastic search that imitates the process of the natural evolution. The genetic algorithms have shown that they are capable of giving solutions in complex controls without experiencing the difficulties that can be associated with the high or optimum dimension as are used to occurring with the techniques based on descending gradient. Utilizing genetic algorithms to carry out the tuning of the controller results in an optimum

controller that can be evaluated each time. For the study of the genetic algorithms was decided to create an objective function of optimization that evaluate the profits in optimum form of the controller PID based on the global error control systems. Conventional methods using tuning tests will be carried out as are the standard practices of adjustment, given by Ziegler Nichols, and techniques of optimization like are, IAE (Integral Absolute Error), ITAE (Integral Swindle Absolute Error), for the design of controllers PID. Tests of identification based on algorithms will be carried out recursive based on the Medium Quadratic Error (MSE) the one that produces a more efficient performance in controllers PID, in comparison with the techniques Conventional. The system ball and hoop can be submitted recursive algorithms-proof in such a way that be reached its function of transfer. An estimator recursive estimator of square minimums is created to estimate the system on line and to provide the estimation more I need of the system so that the genetic algorithm design a controller PID more optimum.

CAPÍTULO I

PLANTEAMIENTO INICIAL DE LA INVESTIGACIÓN

El sistema pelota y aro (PYA) es una analogía para el movimiento de un líquido de una nave cilíndrica, esta analogía se muestra en la Figura 1.1, que nos muestra dos posiciones, del lado izquierdo ilustra la agitación en un cilindro, mientras que hacia la derecha enseña como la pelota rueda análogamente dentro del aro.

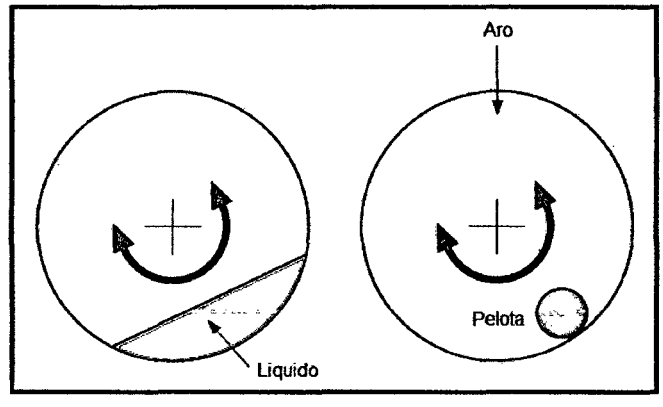


Figura 1.1: Sección transversal a través del cilindro (izquierda) y pelota rodando dentro del aro (derecha).

Para entender este sistema, consideremos un péndulo cilíndrico que se muestra en la Figura 1.2 y comparamos con el tazón de agua. El radio R del cilindro determina la frecuencia de oscilación y la fricción en el rodado de la pelota determina el rango de caída de las oscilaciones. Lo mismo es verdadero si la pelota se reemplaza por un líquido donde la frecuencia de agitación es dada por

el radio del contenedor mientras que la viscosidad del líquido determinará el rango de caída de las oscilaciones.

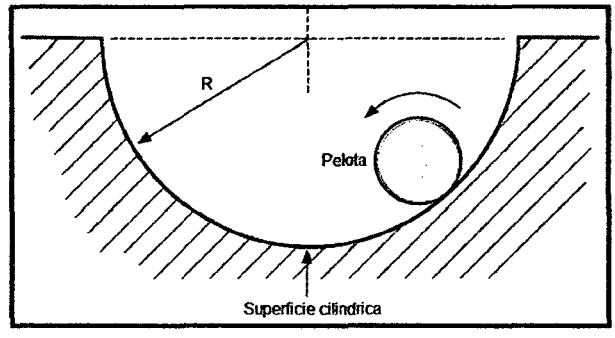


Figura 1.2: Péndulo cilíndrico.

La pelota y el aro simulan el problema práctico, sin embargo, es también importante para estudios prácticos de laboratorio por lo que se permite examinar oscilaciones en sistemas dinámicos en general y demostrar como la teoría de control puede cambiar estas oscilaciones en forma muy interesante. El profesor Wellstead tuvo la idea de diseñar el sistema PYA mientras trabajaba hace algunos años en estrategias de control para reducir la agitación de líquidos en sistemas dinámicos. El diseño del sistema PYA es una muestra tecnológica para demostrar a los ingenieros algunos de los métodos de control para reducir las oscilaciones en los líquidos en los tanques.

1.1. Identificación del problema

¿Qué podemos hacer para controlar la velocidad del aro y controlar la posición de la bola?

Se busca obtener la respuesta del sistema debido a una entrada escalón de la velocidad del aro con un controlador PID empleando técnicas de sintonía preliminar como son 3 valores distintos propios del controlador PID. A partir de los resultados, el algoritmo genético seleccionará los parámetros que permitan obtener mejores resultados, observando el error estacionario y la constante de tiempo de cada caso.

En forma análoga podemos emplear procedimientos para el control de la posición de la bola, buscaremos observar el comportamiento en estado estacionario de la bola, con Vref = 0 y condiciones iniciales $\Psi(0) = 0$ y $\Psi'(0) = 0$. Para esto se implementa un controlador de tipo PID y se hacen pruebas con distintos valores previamente optimizados.

1.2. Formulación del problema

1.2.1 Problema general: "Controlar la velocidad del aro y la posición de la bola del sistema pelota y aro".

El problema consiste en controlar la velocidad del aro y la posición de la bola del sistema pelota y aro. El control se realiza mediante un computador provisto de convertidores análogo digital, digital análogo y software que permite simular controladores de la familia PID.

1.2.2 Problema específico: "Utilizar los algoritmos genéticos para optimizar el controlador PID aplicado al sistema pelota y aro".

El sistema de trabajo consiste en contrastar el desempeño de varios conjuntos de parámetros para controladores PID aplicados al sistema PyA mediante técnicas de inteligencia artificial como son los algoritmos genéticos, a fin de determinar cuál es el mejor de ellos, teniendo presente parámetros como saturación de actuación, tiempo de respuesta y estabilidad entre otros.

1.3. Objetivos de la investigación

Los objetivos del trabajo de tesis están ligados a plantear la hipótesis, dar a conocer los resultados, planteamientos, proposiciones o respuestas en torno al problema que le ocupa.

La búsqueda de nuevos paradigmas nos conduce a obtener conocimientos y solucionar problemas científicos, filosóficos o empírico-técnicos, y se desarrolla mediante un proceso. En otras palabras daremos soluciones concretas a problemas de carácter científico.

La investigación nos ayuda a mejorar el estudio porque nos permite establecer contacto con la realidad a fin de que la conozcamos mejor. Constituye un estímulo para la actividad intelectual creadora. Ayuda a desarrollar una curiosidad creciente acerca de la solución de problemas, además, contribuye al progreso de la lectura crítica.

1.3.1 Objetivo general

El proyecto de Tesis tiene como objetivo diseñar un controlador PID óptimo para el sistema pelota y aro que consiste en una pelota de acero que está libre rodando por dentro de un aro circular. Este sistema ilustra la dinámica compleja de líquido salpicado, es decir la manera como se comporta el líquido en un contenedor móvil. El aparato de la pelota y el aro es difícil de controlar utilizando con técnicas tradicionales, se requiere de un diseño de controlador que optimice parámetros de un controlador PID porque los parámetros de sistema cambian constantemente. Se propuso el estudio de optimización del controlador PID por ser este un controlador ampliamente usado en la industria.

1.3.2 Objetivos específicos

a.- Realizar el modelamiento matemático del sistema pelota y aro.

b- Especificar requerimientos de diseño para el buen control.

c.- Usar técnicas de optimización que lleven a un buen desempeño del control.

d.- Diseñar el controlador y validar su desempeño por medio de la simulación.

1.4. Justificación

La ejecución del presente trabajo, es un aporte científico que se desarrolla en la Sección de Posgrado de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional del Callao. El trabajo de tesis permitirá ampliar conocimientos de técnicas emergentes de inteligencia artificial, asimismo, serán complementadas con herramientas que llevarán a cabo las simulaciones.

Para demostrar las técnicas expuestas se ha escogido un tema de un sistema complejo que medirá indicadores importantes en la resolución de

Id. Exemplar: 39026

sintonías finas provenientes del estudio de las técnicas de optimización basadas en los algoritmos genéticos.

1.5. Limitaciones y facilidades

La limitación es que no se cuenta con el dispositivo pelota y aro, en ese sentido en el análisis se usan los parámetros definidos previamente por el fabricante Technical Teaching Equipment for Engineering, y son los siguientes:

a.- Unidad independiente y compacta.

b.- Muestra los problemas de control de velocidad y posición de un cuerpo móvil o líquido en un contenedor.

c.- Expone el control básico de posición o velocidad, y de estudios avanzados de líquido en naves cilíndricas.

Se debe mencionar que es posible trabajar con datos obtenidos a partir de las simulaciones que el fabricante nos proporciona los parámetros pudiendo ser èstos emulados para posible desarrollo de datos para simulaciones que serán revalidados con los algoritmos de optimización e identificación del modelo.

1.6. Hipótesis de partida

En función de las interrogantes planteadas del problema, así como de los objetivos generales y específicos que se persigue, el siguiente trabajo de tesis plantea la siguiente hipótesis: **"Es posible optimizar los parámetros del controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos"**.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes del estudio

A continuación se presentan algunos de los principales aportes e investigaciones desarrolladas sobre controladores PID aplicados al sistema pelota y aro, tomadas del ámbito internacional.

- En el trabajo de Ian Griffin, de la Universidad DCU de Irlanda, "On-line PID Controller Tuning using Genetic Algorithms", el objetivo de este proyecto fue crear un controlador PID para el sistema pelota y aro, el cual es sintonizado en línea, usando algoritmos genéticos. El sistema pelota y aro es notoriamente difícil de controlar óptimamente usando un controlador PID porque los parámetros del sistema son constantemente cambiantes. Esta es la razón por lo que una estrategia en línea fue aplicada. Los algoritmos genéticos son efectivos para encontrar valores óptimos de los parámetros del controlador PID.
- En el trabajo de S. Kanthalaskshmi, del Comba PSG College of Technology de la India, "Genetic Algorithm Based Self Tuning". El algoritmo genético es usado para dos tareas básicas del Self Tune Regulator (STR), identificación del sistema y sintonización del PID. El controlador tiene la capacidad de

autosintonizar sus parámetros mientras las propiedades dinámicas de la planta cambian, de una forma óptima. La eficiencia del sistema pelota y aro, el cual es difícil de control óptimamente usando un controlador PID porque los parámetros del sistema cambian constantemente, es presentada. Entonces el algoritmo genético propuesto basado en control adaptativo óptimo es diseñado para sintonizar el controlador PID. Perturbaciones son aplicadas al sistema para verificar la robustez del sistema propuesto. El resultado refleja que el esquema propuesto mejora la eficiencia del proceso en términos de especificaciones en el dominio del tiempo, robustez al cambio de parámetros y estabilidad óptima.

2.2 Bases teóricas

2.2.1 El sistema pelota y aro

El modelo dinámico del sistema PYA mostrado en la Figura 2.1, es análogo al sistema de inclinación/agitación que ocurre cuando se transporta líquidos, ya sea dentro de contenedores o en forma de combustible. El movimiento de la pelota en la parte interior de la periferia del aro reproduce las oscilaciones de los fluidos en tanques o contenedores durante el transporte de los mismos.

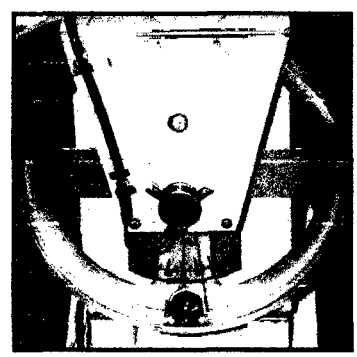


Figura 2.1: El sistema de pelota y aro.

El sistema pelota y aro es también adecuado para demostrar algunos conceptos básicos de la dinámica de sistemas y teoría de control, como ejemplo los ceros del sistema y los sistemas de fase mínima. El sistema presenta las siguientes características:

- Un aro puede rotar con una pelota de acero en su periferia interior.
- Un motor servo, puede manejar el aro y controlar el ángulo de inclinación mediante su torque τ (t).
- Un sensor del ángulo del aro θ nos dará la medida de dicho ángulo.

2.2.2 Ecuaciones Dinámicas del Sistema de Pelota y Aro

La dinámica del sistema de pelota y aro es complicada y difícil de obtenerla utilizando procedimientos estándar. Un método para hacer la deducción màs sencilla es utilizando un método de la posición angular de variación. La Figura 2.2 muestra el nuevo sistema de pelota y aro pero incluyendo etiquetas para las variables. El aro es montado en el eje del motor el cual es modelado como una fuente de torque τ (t).

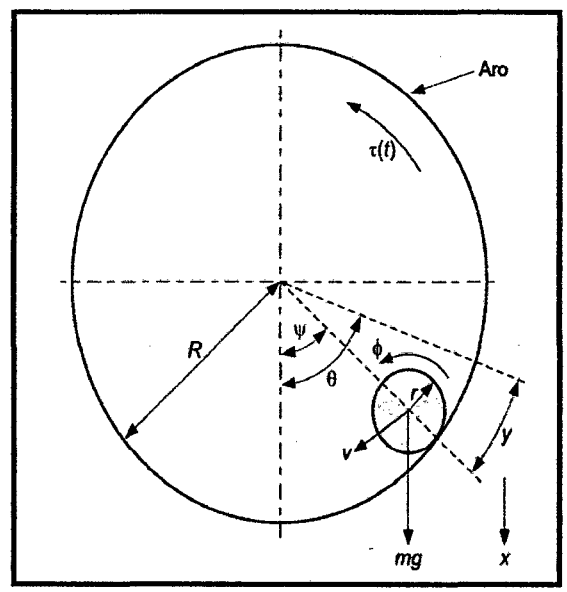


Figura 2.2: Representación esquemática del sistema pelota y aro

Donde las variables más importantes son:

- El radio del aro: R
- El radio de la pelota: r
- La masa de la pelota: m
- El ángulo de la pelota respecto a la vertical (inclinación/agitación): ψ
- La posición de la pelota en el aro: y
- El torque de entrada al aro: τ (t)

El comportamiento dinámico del sistema será completamente representado por las ecuaciones de movimiento del aro **θ**, y la posición **y** de la pelota en la periferia interior del aro. Por esta razón utilizaremos estas variables como las coordenadas generalizadas en las ecuaciones de Lagrange. Ninguna de estas coordenadas es limitada, de tal forma que las diferenciales son respectivamente δθ y δy, [5].

Entonces las ecuaciones de Lagrange para dichas coordenadas son, para θ.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\delta L}{\delta \theta} + \frac{\delta J}{\delta \dot{\theta}} = \tau_\theta(t) \tag{2.1}$$

Mientras para la coordenada y es:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}}\right) - \frac{\delta L}{\delta y} + \frac{\delta J}{\delta \dot{y}} = F_y(t) \tag{2.2}$$

Donde L es el Lagraniano, J es el contenido del sistema, τ_θ (t) es el torque generalizado externo con respecto de θ y $F_y(t)$ es la fuerza generalizada externa con respecto a la coordenada y.

Ahora necesitamos algunas ecuaciones para relacionar las coordenadas generalizadas con otras variables en el sistema. En particular la rotación angular de la pelota ø, es dada por:

$$\phi = \frac{y}{r} \tag{2.3}$$

La velocidad traslacional de la pelota v es dada también por:

$$v = (R - r)\dot{\psi} \tag{2.4}$$

Donde r es el radio verdadero sobre el cual rueda la pelota. Esto quiere decir que el radio verdadero será menor que el radio ideal de la pelota r_b.

El ángulo de inclinación ψ se relaciona con las coordenadas generalizadas por medio de:

$$\psi = \left(\theta - \frac{y}{R}\right) \tag{2.5}$$

El Lagraniano del sistema L, se construye completamente de las energías cinemáticas U, que están asociadas con la rotación del aro, la rotación de la pelota y la traslación de su centro de masa. Entonces el sistema Lagraniano puede ser expresado como:

$$L = U = \frac{1}{2}\left(I_a(\dot{\theta})^2 + I_b(\dot{\phi})^2 + mv^2\right) \tag{2.6}$$

Reescribiendo la ecuación en términos de las coordenadas generalizadas resulta:

$$L = \frac{1}{2}\left(I_a\left(\dot{\theta}\right)^2 + I_b\left(\frac{\dot{y}}{r}\right)^2 + m\left((R-r)\left(\dot{\theta} - \frac{\dot{y}}{R}\right)\right)\right) \tag{2.7}$$

Donde:

- I_a es el momento de inercia del aro
- I_b es el momento de inercia de la pelota
- m es la masa de la pelota

Además el contenido del sistema J, se ha asociado con la función de la fricción en el rodado de la pelota, representada por el coeficiente b_b y el cuerpo del motor que se representa mediante un coeficiente de fricción rotacional b_m lo que resulta en:

$$J = \frac{1}{2}\left(b_b\left(\frac{\dot{y}}{r}\right)^2 + b_m\left(\dot{\theta}\right)^2\right) \tag{2.8}$$

Las entradas generalizadas son dos: primero la variable θ, el torque generalizado $\tau_\theta(t)$ es dado por la suma del torque de entrada del motor y la gravedad que actúa sobre la pelota como sigue:

$$\tau_\theta(t) = \tau(t) + mg\frac{\delta x}{\delta \theta} \tag{2.9}$$

Para la variable y, la F_y generalizada es dada por:

$$F_y = mg\frac{\delta x}{\delta y} \tag{2.10}$$

Donde x es el desplazamiento vertical de la pelota que se mide positivamente hacia abajo como se define a continuación:

$$x = -(R-r)\left(1-\cos\left(\theta-\frac{y}{R}\right)\right) \tag{2.11}$$

De aquí (2.9) resulta:

$$\tau_\theta(t) = \tau(t) - mg\,(R-r)\sin\left(\theta-\frac{y}{R}\right) \tag{2.12}$$

Además (2.10) viene dada por.

$$F_y(t) = mg\left(\frac{R-r}{R}\right)\sin\left(\theta-\frac{y}{R}\right) \tag{2.13}$$

Las ecuaciones de movimiento siguen la forma general de la ecuación de Lagrange, de tal manera que para θ tenemos:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right) - \frac{\delta L}{\delta\theta} + \frac{\delta J}{\delta\dot{\theta}} &= \tau_\theta(t) \qquad (2.14)\\ \left(I_a + m(R-r)^2\right)\ddot{\theta} + b_m\dot{\theta} - \frac{m(R-r)^2}{R}\ddot{y} &= \tau(t) - mg(R-r)\sin\left(\theta-\frac{y}{R}\right) \end{aligned}$$

Para la variable y:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial\dot{y}}\right) - \frac{\delta L}{\delta y} + \frac{\delta J}{\delta\dot{y}} &= F_y(t) \qquad (2.15)\\ \left(\frac{I_b}{r^2} + \frac{m(R-r)^2}{R^2}\right)\ddot{y} + \frac{b_b}{r^2}\dot{y} - \frac{m(R-r)^2}{R}\ddot{\theta} &= mg\frac{(R-r)}{R}\sin\left(\theta-\frac{y}{R}\right) \end{aligned}$$

Las ecuaciones (2.14) y (2.15) son las ecuaciones dinámicas del movimiento del sistema PYA. Ambos conjuntos de ecuaciones pueden ser combinados en una sola ecuación diferencial expresada en forma matricial que resulta en:

$$\begin{pmatrix} \tau(t) \\ 0 \end{pmatrix} = \begin{pmatrix} I_a + m(R-r)^2 & \dfrac{-m(R-r)^2}{R} \\ \dfrac{-m(R-r)^2}{R} & \dfrac{I_b}{r^2} + \dfrac{m(R-r)^2}{R^2} \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{y} \end{pmatrix} + \cdots$$

$$+ \begin{pmatrix} b_m & 0 \\ 0 & \dfrac{b_b}{r^2} \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{y} \end{pmatrix} + mg \begin{pmatrix} (R-r)\sin(\theta - \dfrac{y}{R}) \\ -\dfrac{R-r}{R}\sin(\theta - \dfrac{y}{R}) \end{pmatrix} \quad (2.16)$$

2.2.2 Ecuaciones de Modelo Diferencial Linealizado

Asumiendo que el ángulo de inclinación $\psi = \left(\theta - \frac{y}{R}\right)$

es relativamente pequeño, podemos reemplazar $\sin\left(\theta - \frac{y}{R}\right)$ por $\left(\theta - \frac{y}{R}\right)$ para definir una ecuación diferencial matricial de segundo orden de la forma:

$$\begin{pmatrix} \tau(t) \\ 0 \end{pmatrix} = \begin{pmatrix} I_a + m(R-r)^2 & \dfrac{-m(R-r)^2}{R} \\ \dfrac{-m(R-r)^2}{R} & \dfrac{I_b}{r^2} + \dfrac{m(R-r)^2}{R^2} \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{y} \end{pmatrix} + \cdots \quad (2.17)$$

$$+ \begin{pmatrix} b_m & 0 \\ 0 & \dfrac{b_b}{r^2} \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{y} \end{pmatrix} + mg \begin{pmatrix} (R-r) & -\dfrac{R-r}{R} \\ -\dfrac{R-r}{R} & \dfrac{R-r}{R^2} \end{pmatrix} \begin{pmatrix} \theta \\ y \end{pmatrix}$$

- **Modelo Lineal en Variables de Estado**

La ecuación (2.17) es una forma específica de la ecuación:

$$M\ddot{\mathbf{x}} + D\dot{\mathbf{x}} + K\mathbf{x} = Bu \quad (2.18)$$

Esta es la forma general usada frecuentemente en robótica para formular

claramente las ecuaciones diferenciales del sistema. Es un camino conveniente también para formular las ecuaciones del sistema en variables de estado. Específicamente si utilizamos las variables de estado, $x = (\theta, y)^T$, entonces:

$$x_1 = \begin{pmatrix} \theta & y \end{pmatrix}^T \quad x_2 = \begin{pmatrix} \dot{\theta} & \dot{y} \end{pmatrix}^T$$

La expresión (2.18) puede expresarse en ecuaciones de estado como sigue:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ M^{-1}B \end{pmatrix} u$$

$$y = C \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{2.19}$$

La elección de la matriz de salida C dependerá de los estados que se miden. El conjunto de ecuaciones (2.19) puede ser utilizado para diseñar el controlador. El resultado de las simulaciones para el sistema PYA en lazo abierto debido una entrada escalón unitario son mostradas en la Figura 2.3.

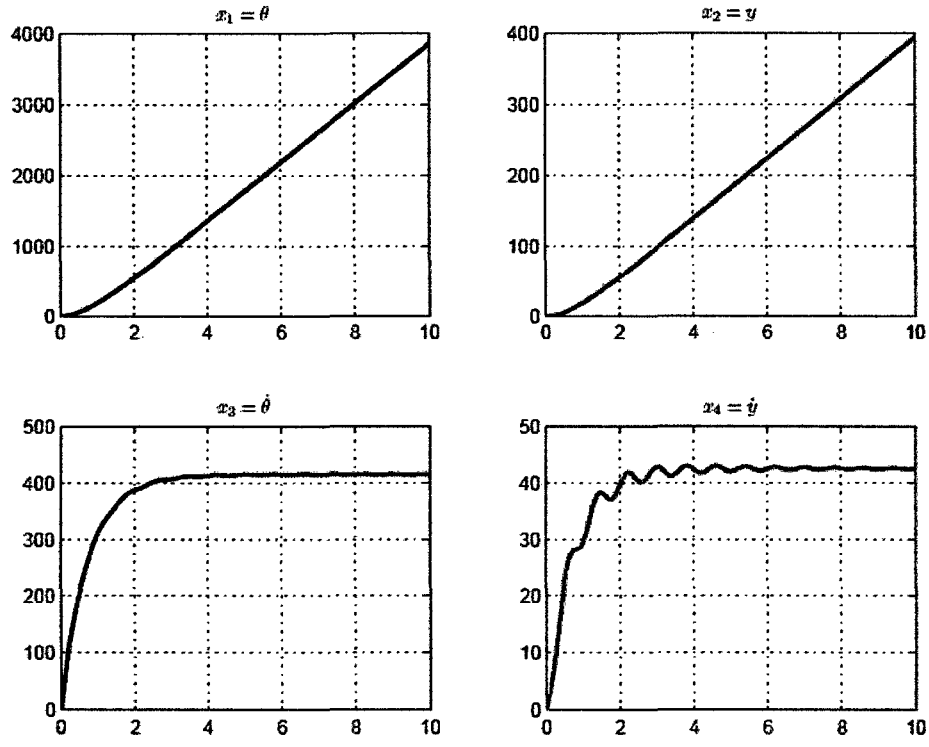


Figura 2.3: Evolución de los estados del sistema PYA en lazo abierto.

- **Aproximaciones y Sustituciones**

Debido a que el radio del aro es mucho más grande que el radio de la pelota R >> r, la ecuación (2.17) puede ser reescrita como sigue:

$$\begin{pmatrix} \tau(t) \\ 0 \end{pmatrix} = \begin{pmatrix} I_a + mR^2 & -mR \\ -mR & \frac{I_b}{r^2} + m \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{y} \end{pmatrix} + \cdots$$
$$+ \begin{pmatrix} b_m & 0 \\ 0 & \frac{b_b}{r^2} \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{y} \end{pmatrix} + mg \begin{pmatrix} R & -1 \\ -1 & \frac{1}{R} \end{pmatrix} \begin{pmatrix} \theta \\ y \end{pmatrix} \quad (2.20)$$

Considerando que el momento de inercia de la pelota solida es $I_b = \frac{2}{5} m r_b^2$ y que el radio de la pelota es distinto al radio de rodado, por lo que las ecuaciones

pueden ser escritas como sigue:

$$\begin{pmatrix} \tau(t) \\ 0 \end{pmatrix} = \begin{pmatrix} I_a + mR^2 & -mR \\ -mR & m\left(\frac{2r_b^2}{5r^2} + 1\right) \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{y} \end{pmatrix} + \cdots$$
$$+ \begin{pmatrix} b_m & 0 \\ 0 & \frac{b_b}{r^2} \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{y} \end{pmatrix} + mg \begin{pmatrix} R & -1 \\ -1 & \frac{1}{R} \end{pmatrix} \begin{pmatrix} \theta \\ y \end{pmatrix} \quad (2.21)$$

Las ecuaciones matriciales diferenciales pueden separarse en ocasiones. En este caso, si el torque inercial de la pelota de masa m es pequeño, comparado con el torque del aro y el torque del motor, entonces la ecuación (2.21) puede ser escrita como la conocida ecuación diferencial para un motor DC con una carga inercial I_a y fricción viscosa b_m:

$$I_a\ddot{\theta} + b_m\dot{\theta} = \tau(t) \quad (2.22)$$

Mientras que la segunda ecuación desde (2.21) es:

$$\left(\frac{2r_b^2}{5r^2} + 1\right)\ddot{y} + \frac{b_b}{mr^2}\dot{y} + \frac{g}{R}y = R\left(\ddot{\theta} + \frac{g}{R}\theta\right) \quad (2.23)$$

Las dos ecuaciones diferenciales que han sido separadas son muy útiles porque nos permiten escribir el modelo del sistema como dos funciones de transferencia en cascada, donde en la primera función de transferencia, representa el torque del motor que produce un ángulo del aro y en la segunda función de transferencia dicho ángulo del aro produce una posición para la pelota. Estas ideas son muy

útiles para diseñar un modelo de simulación y controladores requeridos.

En la Tabla 2.1 se indican las variables y parámetros con los que se describe el comportamiento dinámico de la bola y el aro.

Tabla 2.1: Variables y constantes del sistema

Símbolo	Descripción	Valor
$\tau(t)$	Torque del motor	[Nm]
$\theta(t)$	Posición angular del aro	[rad]
$\omega(t)$	Velocidad angular del aro	[rad/seg]
$\psi(t)$	Posición angular de la bola	[rad]
I_a	Momento de inercia del aro	$(1/2)MR^2$ [kg/m^2]
I_b	Momento de inercia de la bola	$(2/5)mr_b^2$ [kg/m^2]
b_m	Coeficiente de roce del motor	2.05×10^{-3} [Nm/seg]
b_b	Coeficiente de roce de la bola	1.92×10^{-6} [Nm/seg]
R	Radio del aro	0.1025 [m]
r	Radio de rotacion de la bola	7.5×10^{-3} [m]
r_b	Radio de la bola	9.5×10^{-3} [m]
M	Masa del aro	0.3 [Kg]
m	Masa de la bola	28.1×10^{-3} [kg]
g	Aceleración de la gravedad	9.8 [m/seg^2]

2.2.3 Controladores PID

- **Introducción**

Un controlador de retroalimentación (feedback) está diseñado para generar una salida que cause algún esfuerzo correctivo para que sea aplicado a un proceso, de tal manera que se lleva la salida de dicho proceso hacia un valor deseado conocido como Punto de Referencia (Setpoint.) El controlador usa un

actuador que afecta al proceso y un sensor para medir el resultado. Virtualmente todos los controladores de retroalimentación determinan su salida mediante la observación del error producido entre el punto de referencia y la medición de la variable del proceso. El error ocurre cuando el operador humano cambia el punto de referencia intencionalmente o cuando una perturbación o carga sobre el proceso, cambia la variable de proceso accidentalmente. Entonces la misión del controlador es eliminar el error automáticamente. Pero para que esto suceda, la elección del controlador a usar debe de darse siguiendo ciertas especificaciones de diseño, los cuales describen que debe hacer el sistema y como debe hacerlo.

Estas especificaciones son únicas para cada aplicación individual y con frecuencia incluyen especificaciones como estabilidad relativa, precisión en estado estable (error), respuesta transitoria, y características de respuesta en frecuencia. Esta última consideración ha sido históricamente desarrollada con una gran cantidad de herramientas gráficas tales como las trazas de Bode, la traza de Nyquist, la traza de ganancia–fase, y la carta de Nichols, pero debido al desarrollo y la disponibilidad de un software de computadora amigable, rápido y poderoso, la práctica de diseño ha cambiado rápidamente, tal es así que el diseñador puede ejecutar ràpidamente, un gran numero de diseños empleando especificaciones en el dominio del tiempo. Esto ha disminuido considerablemente la ventaja histórica del diseño en el dominio de la frecuencia, el cual está basado en la conveniencia de realizar el diseño gráfico en forma manual. Además es difícil, excepto para el diseñador experimentado, seleccionar un conjunto coherente de especificaciones en el dominio de la frecuencia que correspondan a requisitos de desempeño en el dominio del tiempo, [1].

Algo que si no ha cambiado, dentro del diseño de sistemas de control, el

uso permanente de los controladores Proporcional, Integral, Derivativo (PID), los cuales se utilizan debido a su robustez. En el campo de los sistemas para control de procesos, es un hecho bien conocido que los esquemas de control PID básicos y modificados han demostrado su utilidad para aportar un control satisfactorio, aunque tal vez no aportan un control óptimo en muchas situaciones específicas.

- **Interpretación en el Dominio del Tiempo**

Para entender el significado del control PID en el dominio del tiempo, es necesario definir ciertos conceptos que constituyen el cuerpo del controlador. Como mencionamos anteriormente, entre el controlador más popular podemos mencionar al controlador PID, a continuación, desarrollaremos algunas descripciones breves acerca de sus características más relevantes respecto de las actuaciones en un proceso.

- **Control Proporcional**

Es una técnica de control la cual multiplica la señal de error (la diferencia entre el punto de referencia y la variable de proceso) por una ganancia especificada Kp y es usada como una señal correctiva que ingresa al proceso. El resultado efectivo radica en exagerar el error y reaccionar inmediatamente para corregirlo. Kp no puede eliminar completamente los errores (a menos que el proceso tenga propiedades integrativas inherentes), ya que como el error siguiente se acerca a cero, la corrección proporcional desaparece. Esto produce cierta cantidad de error en estado estable, lo cual se puede apreciar en la Figura 2.4, donde existe un desplazamiento entre el valor deseado (punto de referencia) y el valor de la salida del sistema.

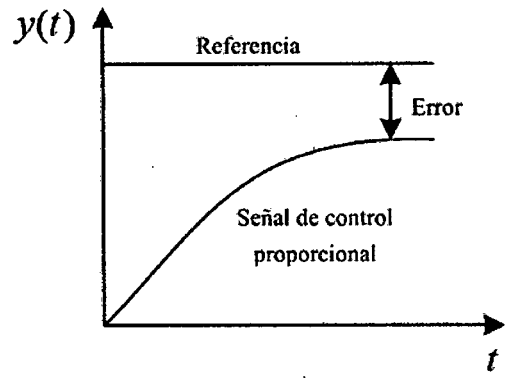


Figura 2.4: Efecto de la acción de control proporcional.

- **Control Integral**

Es una técnica la cual acumula la señal de error (ver Figura 2.5) sobre los tiempos t_1 y t_2, y multiplica dicha suma por una ganancia especificada K_i y usa este resultado como una señal correctiva sobre el proceso. Debido a que esta técnica actúa sobre errores pasados, el factor de corrección no se va a cero con el error siguiente, eliminando de esta manera el error en estado estable. La ganancia integral tiene un efecto negativo importante, el cual consiste en un factor desestabilizante para el lazo de control para valores grandes de *Ki*, o para valores usados sin una apropiada amortiguación, los cuales pueden causar severas oscilaciones.

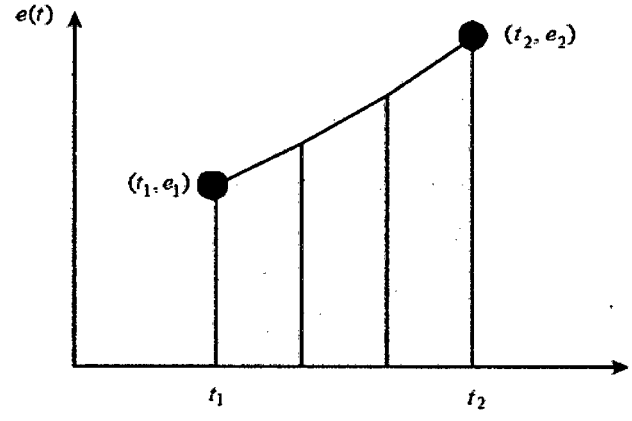


Figura 2.5: Integrando el error en el tiempo

La señal de error acumulado genera el efecto mostrado sobre la salida del sistema, tal y como se muestra en la Figura 2.6.

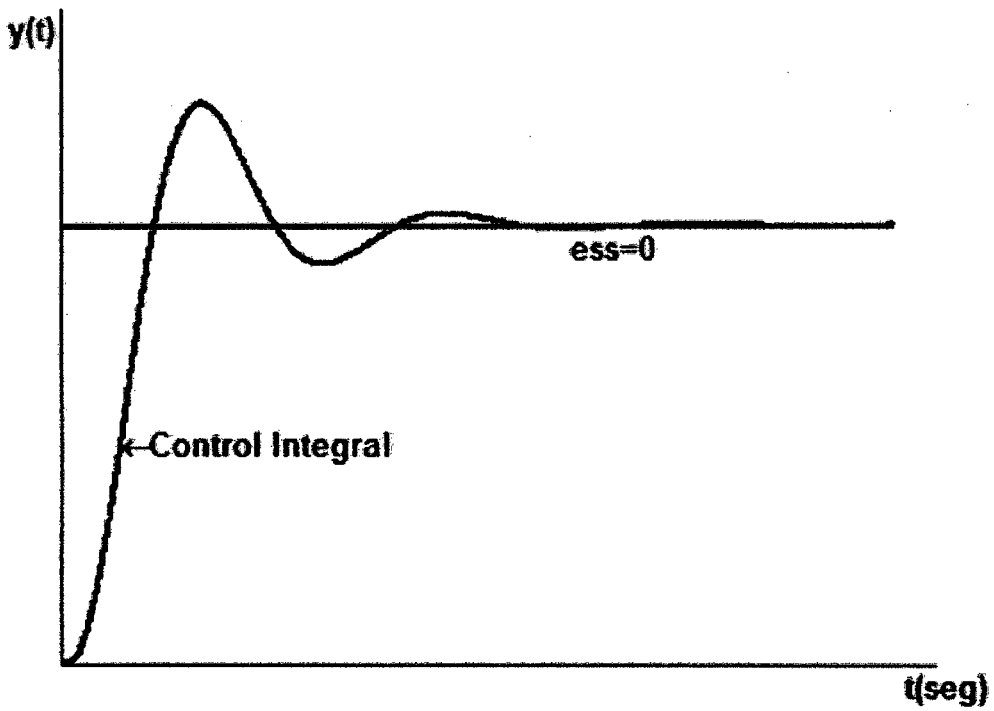


Figura 2.6: Efecto de la acción de control integral.

- **Control Derivativo**

Se basa en la multiplicación de la razón de cambio del error siguiente por una ganancia especificada K_d y usa este resultado como una señal correctiva sobre el proceso. Otra forma de ver el control derivativo es como un control anticipativo. Esto es, al conocer la pendiente el controlador puede anticipar la dirección del error y emplearla para controlar mejor el proceso.

En forma intuitiva, el control derivativo afecta el error en estado estable de un sistema solo si el error en estado estable varía con el tiempo. Si el error en estado estable de un sistema es constante con el tiempo, la derivada con

respecto al tiempo de este error es cero, y la porción derivativa del controlador no provee ninguna entrada al proceso, lo cual se muestra en la Figura 2.7 que nos indica que mientras el error existe y varíe con el tiempo, se podrá calcular la derivada de dicho error y de esta manera nos permite conocer hacia donde se dirige dicho error.

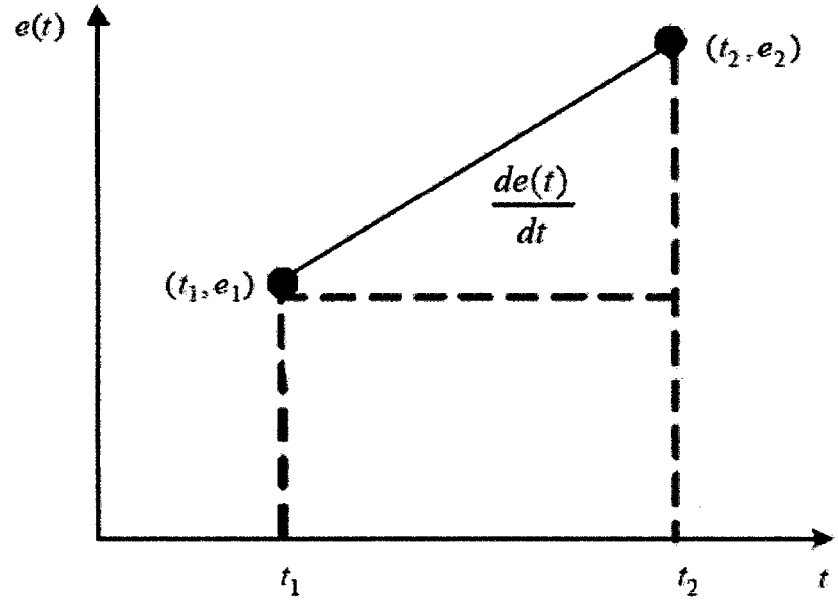


Figura 2.7: Efecto de la acción de control derivativa.

- **Control Proporcional Integral Derivativo**

El control PID de acuerdo a Astrom, la versión del algoritmo de control PID tiene la siguiente forma:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right] \tag{2.24}$$

$$e(t) = r(t) - y(t) \tag{2.25}$$

Donde **u** es la variable de control, **e** representa el error entre la señal de referencia r y el valor medido de la salida del proceso **y**. La ganancia proporcional K, la constante de tiempo integral Ti, y la constante de tiempo derivativo Td son los parámetros del controlador. Esta versión del algoritmo PID es también llamada el "controlador PID no interactivo ideal" o "ISA" de sus siglas en Inglés (Ideal noninteracting Algorithm).

Otras formas de algoritmos PID usados en la industria son los siguientes:

- **Controlador PID Interactivo:**

$$u(t) = K'\left[e(t) + \frac{1}{T_i'}\int_0^t e(\tau)d\tau\right]\left[1 + T_d'\frac{de(t)}{dt}\right] \tag{2.26}$$

- **Controlador Ideal PID Paralelo:**

$$u(t) = K_p e(t) + K_i\int_0^t e(\tau)d\tau + K_d\frac{de(t)}{dt} \tag{2.27}$$

Donde *Kp* es la ganancia proporcional, *Ki* es la ganancia Integral y *Kd* es la ganancia derivativa.

La simplificación de la ecuación (2.26) nos da la siguiente notación para los algoritmos PID:

$$u(t) = P + I + D \tag{2.28}$$

Donde, *P* es el término proporcional, *I* es el término integral y *D* es el término derivativo.

Se puede deducir una equivalencia entre los valores del algoritmo no interactivo y el algoritmo ideal paralelo de los controladores PID como se muestra a continuación:

$$K_p = K, \qquad K_i = \frac{K}{T_i}, \qquad K_d = KT_d \tag{2.29}$$

Todas estas versiones son vàlidas y usadas por distintos fabricantes de controladores, por ejemplo, Foxboro y Fisher usan las versiones no interactivos, mientras que Honeywell y Texas Instruments usan el algoritmo interactivo.

Como vemos de las ecuaciones anteriores, la mayoría de los controladores operan sobre una señal de error **e,** la cual es generada en línea (on-line) debido a la sustracción de la salida y del punto de referencia ***r***. Entonces el sistema de control resultante es mostrado en la Figura (2.8) donde ***z*** representa la salida del proceso, en tanto que ***l*** y ***n,*** la carga de perturbación y el ruido en la medición respectivamente, siendo estos últimos los que modifican tanto a la salida del controlador, como a la salida del proceso, generándose las señales $u' = u + l$, e $y = z + n$ respectivamente. A este sistema que se le conoce como "sistema con error retroalimentado".

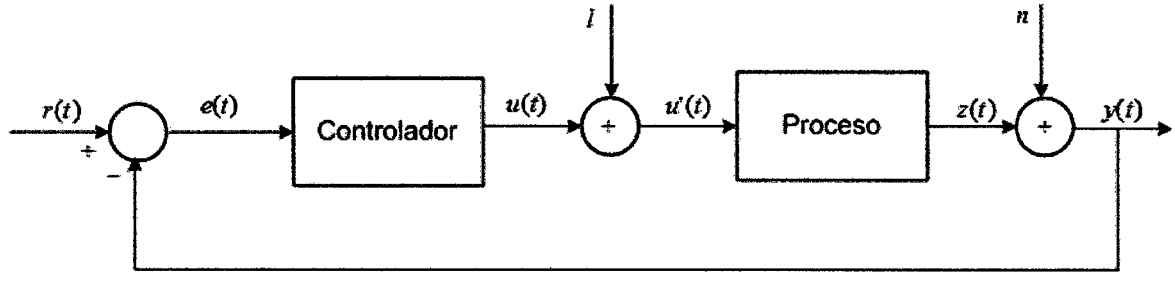


Figura 2.8: Sistema de control con error retroalimentado

En el resto de la tesis, entenderemos por diseño a la estructura usada en los términos *P*, *I* y *D* y por sintonización entenderemos a la selección de los valores numéricos de los parámetros en los términos *P*, *I* y *D*. Además, nosotros trabajaremos con la configuración del "controlador PID no interactivo", a la cual

modificaremos más adelante para dar mayor estabilidad y robustez al control. La elección de esta configuración es debido a los requerimientos de nuestro algoritmo, que como se verá mas adelante, se hará una selección de los mejores controladores, pudiéndose generar valores indeseados, ya que como se sabe por ejemplo si se requiere utilizar *Ti*, y solo tenemos disponibles los valores de *Kp*, *Ki* y *Kd*, la equivalencia *Ti = Kp/Ki* se puede dar, pero si ambos valores *Kp* y *Ki* fueron escogidos ceros, se produce un valor indefinido.

- **Reglas de Sintonización de los Controladores PID**

El proceso de seleccionar los parámetros del controlador que cumpla con las especificaciones de desempeño se conoce como sintonización del controlador. Hay sin embargo algunos métodos de sintonización en los cuales primero, un modelo simple del proceso es determinado por el mismo proceso a ser controlado, para luego escogerse los parámetros PID.

Debido a su uso muy difundido en la práctica, existen varios métodos para sintonización de controladores PID, muchos de los cuales datan de varias décadas atrás. Entre estos métodos "clásicos" podemos encontrar a los siguientes:

- Método de Reacción de la Curva de Ziegler-Nichols.
- Método de Oscilaciones Sostenidas de Ziegler-Nichols.
- Método de Cohen - Coon.

Las dos primeras reglas fueron sugeridas por Ziegler y Nichols para obtener un modelo aproximado de primer orden del sistema y luego sintonizar los controladores PID basándose tanto en las respuestas escalón experimentales, como en la estabilidad marginal cuando sólo se usa la acción de control

proporcional, respectivamente. En ambos métodos denominados reglas de sintonización de Z-N se pretende obtener un 25% de sobrepaso máximo en la respuesta $y(t)$ al escalón unitario, tal y como se observa en la Figura 2.9. Por otro lado, Cohen-Coon hizo una modificación a la segunda regla de Z-N, para que la respuesta tenga un mínimo desplazamiento y otras propiedades favorables.

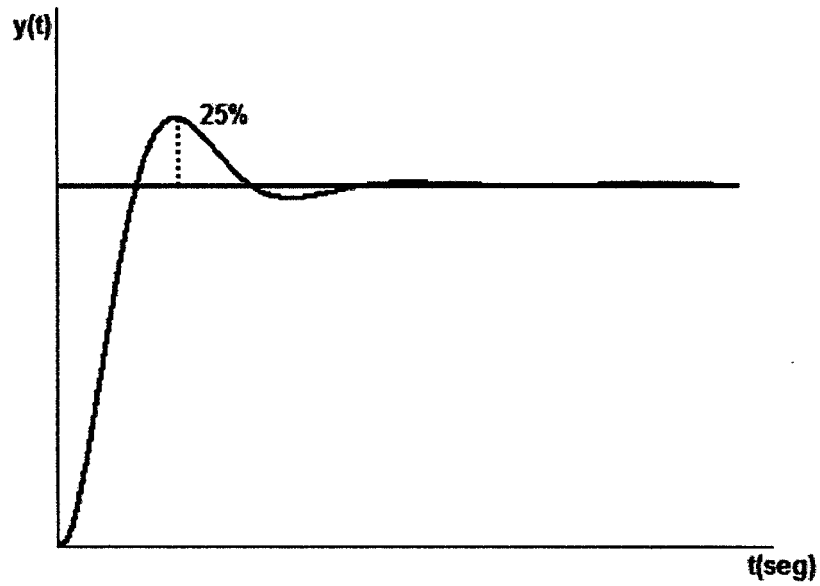


Figura 2.9: Curva de respuesta escalón unitario con 25% de sobrepaso máximo

Las reglas de Ziegler - Nichols, que se presentan a continuación, son muy convenientes cuando no se conocen los modelos matemáticos de las plantas. (Por supuesto, estas reglas se aplican también al diseño de sistemas de control con modelos conocidos.)

- **Método de Reacción de la Curva de Ziegler-Nichols**

En este primer método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental, es decir al sistema se le excita

mediante una entrada escalón unitario y luego se procede a medir la señal de salida, siendo ésta de tipo sobre amortiguado como se observa en la Figura 2.10.

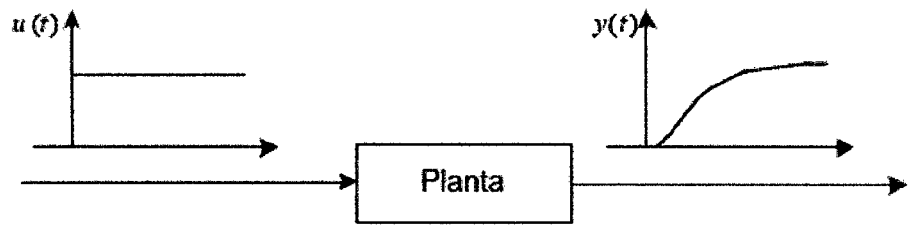


Figura 2.10. Respuesta escalón unitario de una planta.

Si la planta no contiene integradores ni polos complejos conjugados, la curva de respuesta escalón unitario puede tener la forma de *S*, como se observa en la Figura 2.11 (si la respuesta no exhibe una curva en forma de *S*, este método no es adecuado.) A esta curva también se le conoce como curva de reacción del proceso. Tales curvas de respuesta escalón se generan experimentalmente o a partir de una simulación dinámica del proceso.

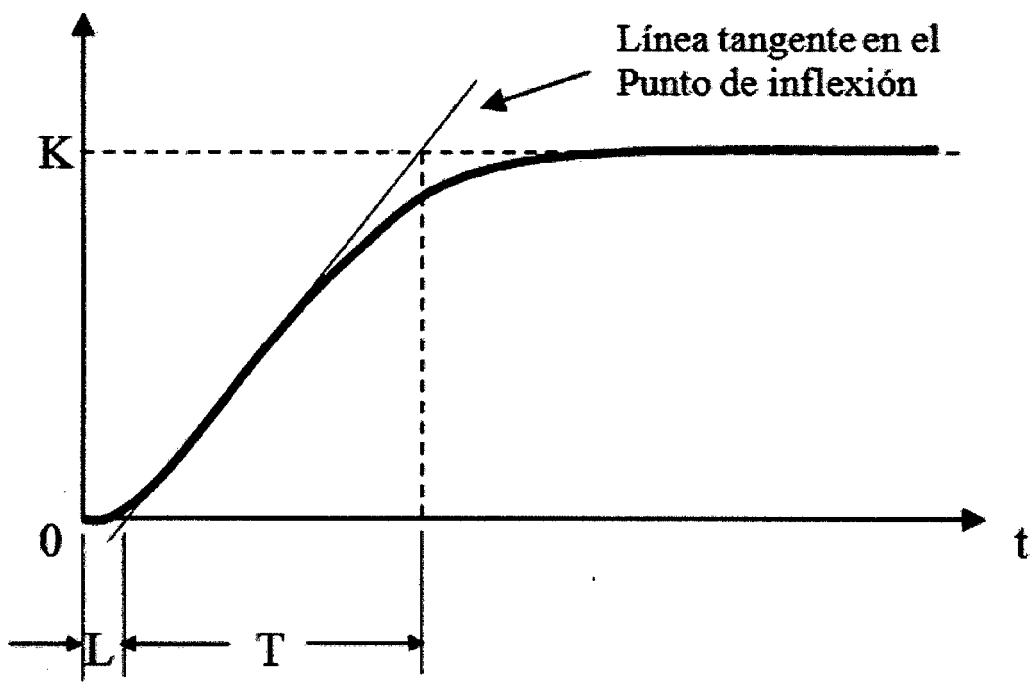


Figura 2.11: Curva de respuesta en forma de S

También en la Figura 2.11 se puede apreciar que la curva en forma de *S* se caracteriza por dos parámetros: el tiempo de retardo *L* y la constante de tiempo *T*. El tiempo de retardo y la constante de tiempo, se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de *S* y determinando las intersecciones de esta tangente con el eje del tiempo y la línea $y(t) = K$. En este caso, la función de transferencia de la planta $Y(s)/U(s)$ se aproxima mediante un sistema de primer orden con un retardo de transporte del modo siguiente:

$$\frac{Y(s)}{U(s)} = \frac{Ke^{-Ls}}{\tau s + 1} \tag{2.30}$$

Ziegler y Nichols sugirieron establecer los valores de *Kp*, *Ti* y *Td* de acuerdo con la fórmula que aparece en la Tabla 2.1.

Tabla 2.1: Regla de sintonización de Z -N basada en la respuesta escalón de la planta

Tipo de Controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Transformando la ecuación (2.24) al plano–s y reemplazando los valores dados en la tabla de arriba se obtiene el controlador sintonizado mediante el primer método de las reglas de Ziegler - Nichols:

$$G_c(s) = 0.6T\frac{\left(s+\frac{1}{L}\right)^2}{s} \tag{2.31}$$

Por tanto, el controlador PID tiene un polo en el origen y un cero doble en $s = -1/L$.

- **Método de Oscilaciones sostenidas de Ziegler-Nichols**

En el segundo método, primero establecemos $Ti = \infty$ y $Td = 0$. Usando sólo la acción de control proporcional (ver la Figura 2.12), se incrementa *Kp* de 0 a un valor crítico *Kcr* en donde la salida exhiba primero oscilaciones sostenidas.

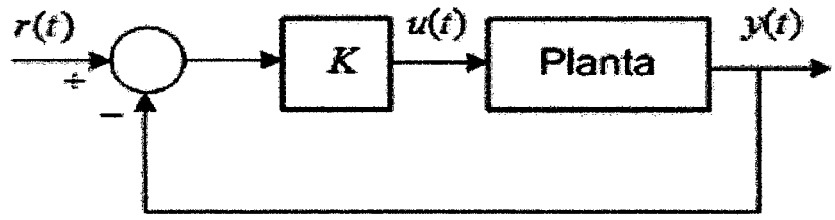


Figura 2.12: Sistema en lazo cerrado con controlador proporcional

Si la planta no aplica oscilaciones sostenidas para cualquier valor que pueda tomar *Kp* no se aplica este método. Por tanto, la ganancia crítica *Kcr* y el periodo Pcr correspondiente se determinan experimentalmente (ver la Figura 2.13).

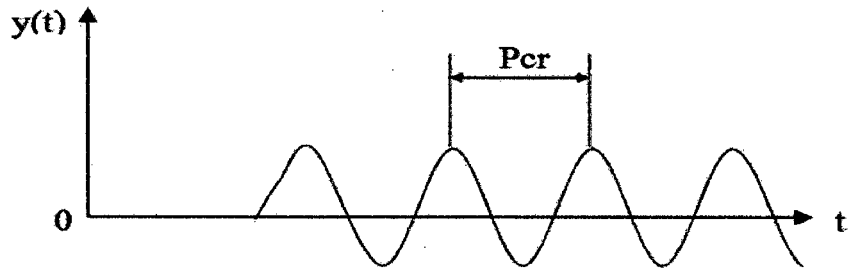


Figura 2.13: Oscilación sostenida con un periodo *Pcr*.

Ziegler - Nichols sugirieron que se establecieran los valores de los parámetros *Kp*, *Ti* y *Td* de acuerdo con la fórmula que aparece en la Tabla 2.2.

Tabla 2.2: Regla de sintonización de Z -N basada en el método de oscilaciones sostenidas (*Kcr* y *Pcr*)

Tipo de Controlador	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$0.82P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Reemplazando los valores de la Tabla 2.2 en la versión de la ecuación (2.24) en el plano–s, el controlador PID sintonizado mediante el segundo método de las reglas de Ziegler - Nichols produce:

$$G_c(s) = 0.075K_{cr}P_{cr}\frac{\left(s+\frac{4}{P_{cr}}\right)^2}{s} \tag{2.32}$$

Por tanto, el controlador PID tiene un polo en el origen y cero doble en $s = -4/P_{cr}$.

- **Implementación Digital**

Una vez obtenido los valores de *Kp*, *Ki* y *Kd* directa o indirectamente a través de sus equivalentes obtenidos del algoritmo PID no entrelazado, el siguiente paso es discretizar el controlador PID continuo, para poder implementarlo en circuitos digitales, los cuales por su gran versatilidad y velocidad son los que hoy en día se usan mayormente para periodos de muestreo pequeños. La ecuación (2.27) puede ser reemplazada por discretización a través de ecuaciones en diferencias. El término derivativo es simplemente reemplazado por una diferencia de primer orden hacia atrás y el término integral puede ser aproximada por una integración rectangular (suma) o una integración trapezoidal. En nuestro caso aplicando una integración rectangular nos da lo siguiente:

$$u(k) = K_p e(k) + K_i T \sum_{i=0}^{k-1} e(i) + \frac{K_d}{T}(e(k) - e(k-1)) \tag{2.33}$$

Donde T es el periodo de muestreo, el cual debe de ser lo más pequeño posible. A esta ecuación también se le conoce como un algoritmo de control no recursivo. Para la formación de la suma todos los errores pasados $e(k)$ tienen que guardarse. Debido al valor producido de la variable manipulada $u(k)$, a este algoritmo también se le conoce como un "algoritmo de posición".

Sin embargo, los algoritmos recursivos son más convenientes para programar. Estos algoritmos se caracterizan por el cálculo de la variable manipulada actual $u(k)$ a través del valor de la variable manipulada previa $u(k-1)$ y de los términos correctivos.

$$u(k-1) = K_p e(k-1) + K_i T \sum_{i=0}^{k-2} e(i) + \frac{K_d}{T}\big(e(k-1) - e(k-2)\big) \tag{2.34}$$

Para esto es necesario sustraer $u(k-1)$ de la ecuación (2.33) del valor $u(k)$ de (2.34), con lo cual se obtiene el algoritmo recursivo PID en tiempo discreto.

$$u(k) - u(k-1) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{2.35}$$

Con los parámetros siguientes:

$$\begin{aligned} q_0 &= K_p + \frac{K_d}{T} \\ q_1 &= -K_p - 2\frac{K_d}{T} + K_i T \\ q_2 &= \frac{K_d}{T} \end{aligned} \tag{2.36}$$

Al cambio actual en la variable manipulada, $\Delta u(k) = u(k) - u(k-1)$, se le conoce como el "algoritmo de velocidad". Para el caso en que la integración continua sea reemplazada por una integración aproximada trapezoidal entonces se obtiene la siguiente ecuación para la variable manipulada:

$$u(k) = K_p e(k) + K_i T\left(\frac{e(0) + e(k)}{2} + \sum_{i=1}^{k-1} e(i)\right) + \frac{K_d}{T}\Big(e(k-1) - e(k-2)\Big) \tag{2.37}$$

Luego de la correspondiente sustracción de $u(k-1)$, se obtiene otra relación recursiva para el algoritmo de control PID.

$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{2.38}$$

Con los siguientes parámetros:

$$q_0 = K_p + \frac{K_d}{T} + \frac{K_i T}{2}$$

$$q_1 = -K_p - 2\frac{K_d}{T} + \frac{K_i T}{2}$$
$$q_2 = \frac{K_d}{T} \qquad (2.39)$$

- **Optimización de los Parámetros PID**

Cuando se implementa los controladores PID, generalmente se trabaja con actuadores que tienen un rango de trabajo establecido, razón por la cual no se puede aumentar ni disminuir indefinidamente la señal de control. Es por eso que se incluye el término saturador, el cual limita la señal de control de acuerdo a los valores máximos y mínimos que puede admitir el actuador. Además se realizan algunos cambios de acciones proporcionales y acciones derivativas que mejoran el desempeño del algoritmo de control. Estas consideraciones serán tratadas a continuación.

- **Término Derivativo y uso de Filtros**

En la mayoría de los controladores PID propuestos en muchos libros básicos de Control, la acción derivativa opera en la señal de error producida por la diferencia entre la referencia y la salida del proceso. Sin embargo por cambios abruptos de la señal de referencia, el término derivativo también sufre cambios drásticos, lo cual desestabiliza la acción de control. Es por eso que se introduce una nueva técnica, en la cual, la acción derivativa opera solo en la señal de salida del proceso y no en la señal de referencia para prevenir lo dicho anteriormente.

Por otro lado, una modificación importante es filtrar la acción derivativa con un filtro de primer o segundo orden, para eliminar o disminuir lo máximo posible el ruido derivativo. Esta característica limita la medición de alta frecuencia de la

amplificación del ruido en la salida del controlador. Entonces la señal de control tendrá menos ruido y la ganancia de alta frecuencia permanecerá dentro de los límites apropiados. A continuación se muestra un término derivativo con un filtro de primer orden:

$$\frac{T_d}{N}\frac{dD}{dt} + D = -KT_d\frac{dy_p}{dt} \tag{2.40}$$

Para aproximar el término derivativo en la ecuación (2.40) se tiene las siguientes opciones:

- Método de la Diferencia hacia adelante:

$$\begin{aligned} \tau\frac{D(k+1)-D(k)}{T} + D(k) &= K_{ds}\frac{y(k+1)-y(k)}{T} \\ D(k) &= (1-\frac{h}{\tau})D(k) + \frac{K_{ds}}{\tau}(y(k+1)-y(k)) \end{aligned} \tag{2.41}$$

- Método de la Diferencia hacia atrás:

$$\begin{aligned} \tau\frac{D(k)-D(k-1)}{T} + D(k) &= K_{ds}\frac{y(k)-y(k-1)}{T} \\ D(k) &= \frac{\tau}{\tau+T}D(k-1) + K_{ds}T(y(k)-y(k-1)) \end{aligned} \tag{2.42}$$

Este último método es siempre estable, motivo por el cual será usado por nosotros para el desarrollo del esquema final del controlador que usaremos, mientras que la ecuación (2.41) es inestable para pequeños periodos de muestreo *T*.

- **Modificación de la Acción Proporcional K_p**

Un concepto adicional y que solo será mencionado es la modificación de la acción proporcional. Como se sabe, el sistema con lazo de retroalimentación propuesto en la ecuación (2.24) trabaja solamente sobre el error de retroalimentación (un grado de libertad), sin embargo, si separa la salida del proceso de la señal de referencia en el término proporcional de la ley de control (dos grados de libertad), la flexibilidad de diseño se incrementa y por lo tanto se mejora el rendimiento del controlador. La modificación deseada se consigue parcialmente si modificamos el error *e*, en la parte proporcional de la ecuación (2.24).

$$e_p = \beta r - y \tag{2.43}$$

Donde β es un parámetro constante de diseño. La respuesta transitoria del lazo cerrado se puede mejorar manipulando el valor de β. Más grados de libertad para el sistema de control, significa que se necesitan más parámetros a ser sintonizados. En muchos controladores PID convencionales β no es sujeto del proceso de sintonización. Si queremos hacerlo como los otros parámetros *Kp*, *Kd* y *Ki*, nosotros deberíamos de ser capaces de sintonizar cuatro parámetros en vez de tres, lo cual no es muy fácil con métodos convencionales.

Para concluir esta sección, es necesario resaltar que en nuestra investigación se utilizará la configuración del controlador PID no entrelazado ideal en su forma modificada tal y como se muestra en la ecuación (2.44), con lo cual se asegura una buena estabilidad del sistema, así como robustez para afrontar los grandes cambios en la señal de referencia, debido a la inclusión de nuevos parámetros que optimizan dicha configuración.

$$P = K_p e$$

$$I(k+1) = I(k) + \frac{K_p T}{T_i} e + \frac{1}{T_i}(v(k) - u(k))$$

$$D(k+1) = \left(\frac{\tau}{\tau + T}\right) D(k) + K_p T_d T(y(k) - y(k-1)) \quad (2.44)$$

- **Optimizando el Controlador PID Digital**

Uno de los métodos de optimización en el diseño del controlador PID es el método del gradiente descendente. En este método, se hace necesario utilizar la función de transferencia del controlador como:

$$G_c(z) = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \quad (2.45)$$

El minimizar la función de error es un problema que puede ser resuelto si los valores de los coeficientes del controlador (*q*0, *q*1, *q*2) son determinados en forma conveniente. Estas tres combinaciones de valores potenciales forman un espacio de tres dimensiones. La función del error formará algún contorno dentro del espacio. Este contorno tiene máximos, mínimos y gradientes que tienen como resultado una superficie continua.

La idea de este método de optimización es alcanzar los mínimos por el camino más corto. Para lograr que este camino sea más corto, se baja el gradiente que lleva a alcanzar los mínimos. Cuando el gradiente cambia punto a punto, asegura que el camino más escarpado todavía es utilizado, esto significa escoger nuevos cambios de dirección. De ahí la reducción de la función de error es lograda analizando la misma función de la misma función. En la siguiente

sección, se obtiene la función de transferencia de planta al minimizar de función de error.

Si el modelo matemático del proceso que será controlado existe, la optimización de los parámetros del controlador se realiza mediante un método que generalmente puede aplicarse para determinar los parámetros desconocidos *q*0, *q*1, . . . , *qv*. Se define una función de costo *J*, por ejemplo, aplicando un criterio de performance del control cuadrático resulta la variación del parámetro del controlador.

$$q^T = \begin{bmatrix} q_1 & q_2 & \cdots & q_v \end{bmatrix} \tag{2.46}$$

Determinado de tal manera que *J* asume un mínimo, mientras:

$$\frac{dJ}{dq} = 0 \tag{2.47}$$

- **Diseño numérico de un Controlador PID**

Un controlador digital puede ser implementado vía ecuaciones en diferencias. Por ejemplo, un controlador PID digital puede ser escrito de la forma:

$$u(k) = u(k-1) + K\left[\left(1 + \frac{T}{T_i} + \frac{T_d}{T}\right)e(k) - \left(1 + \frac{2T_d}{T}\right)e(k-1) + .. \right.$$
$$\left. \frac{T_d}{T}e(k-2)\right] \tag{2.48}$$

La señal de control también puede ser escrita como:

$$u(k) = u(k-1) + q_0\, e(k) + q_1\, e(k-1) + q_2\, e(k-2) \tag{2.49}$$

Donde $q0$, $q1$ y $q2$ son constantes. Para optimizar el controlador se debe escoger algunas especificaciones que ayudaran a escoger los coeficientes $\{q0, q1, q2\}$.

Un método de hacer esto y de una manera inteligente, es conseguir la función de transferencia de lazo cerrado se parezca a alguna función deseada de transferencia de lazo cerrado tan de cerca como sea posible. Se compara la salida de nuestro sistema controlado con el sistema deseado $e(k)$. Entonces se calcula J como:

$$J = \sum_{k} |e(k)|^2 \tag{2.50}$$

Si se diseña el controlador para minimizar J entonces la salida de nuestro sistema controlado se aproxima la salida deseada tan cerca como sea posible.

- **Función de minimización de 3 Dimensiones**

E diseño del controlador PID es un problema de optimización en tres variables. Se empieza con una suposición inicial acerca de los mejores valores de $q = \{q0, q1, q2\}$. Entonces, se aproxima la pendiente ∇Jq de J con respecto a q. Si se cambia los valores de q en la dirección de la pendiente negativa, se desciende bajo la superficie del error hacia un mínimo, [4].

El algoritmo descriptivo del criterio de la optimización numérica viene dado por:

a. Se inicia con una suposición $q = \{q0, q1, q2\}$

b. Se asigna al "tamaño de paso" a un valor pequeño. Se asume $\gamma = 10^{-3}$, y el gradiente con una "prueba de tamaño de pasos" $\delta q0 = \delta q1 = \delta q2$ a un valor menor que γ.

c. Estimar la pendiente en la dirección $q0$:

$$\frac{\partial J}{\partial q_0} \approx \frac{J(q_0 + \delta q_0, q_1, q_2) - J(q_0, q_1, q_2)}{\delta q_0}$$

d. Estimar la pendiente en la dirección $q1$:

$$\frac{\partial J}{\partial q_1} \approx \frac{J(q_0, q_1 + \delta q_1, q_2) - J(q_0, q_1, q_2)}{\delta q_1}$$

e. Estimar la pendiente en la dirección $q2$:

$$\frac{\partial J}{\partial q_2} \approx \frac{J(q_0, q_1, q_2 + \delta q_2) - J(q_0, q_1, q_2)}{\delta q_2}$$

f. Normalizar el gradiente para la mejor performance

$$\Delta = \sqrt{\left(\frac{\partial J}{\partial q_0}\right)^2 + \left(\frac{\partial J}{\partial q_1}\right)^2 + \left(\frac{\partial J}{\partial q_2}\right)^2}$$

g. Actualizando $q0$, $q1$ y $q2$

$$q_0 \leftarrow q_0 - \frac{\gamma}{\Delta}\left(\frac{\partial J}{\partial q_0}\right)$$

$$q_1 \leftarrow q_1 - \frac{\gamma}{\Delta}\left(\frac{\partial J}{\partial q_1}\right)$$

$$q_2 \leftarrow q_2 - \frac{\gamma}{\Delta}\left(\frac{\partial J}{\partial q_2}\right)$$

h. Repetir 3 hasta la convergencia

Notar que cada evaluación de *J*(*x*, *y*, *z*) requiere que se evalúe la performance del sistema para el controlador definido por los parámetros *{x, y, z}*. Se requiere cuatro evaluaciones separadas por iteración del lazo anterior (*J*(*q*0, *q*1, *q*2) necesita ser evaluado sólo una vez). En este caso el criterio de diseño para la respuesta al escalón, necesita cuatro simulaciones de las respuestas al escalón por iteración.

- **Aplicación del Algoritmo PID Óptimo**

El objetivo es hacer que la respuesta al escalón de la planta *G*(*s*) sea digitalizada:

$$G(s) = \frac{1}{s(s+a)(s+b)}$$

Además la respuesta del sistema de control debe parecerse al escalón unitario ideal 1(*k*) tan estrechamente como sea posible. La discretización de *G*(*s*) por medio del retenedor de orden cero:

$$G(z) = (1 - z^{-1})Z\left\{\frac{G(s)}{s}\right\}$$

Si se reemplaza los coeficientes a = 1 y b = 5, se obtiene la siguiente función de transferencia:

$$G(z) = \frac{b_1 z^2 + b_2 z + b_3}{a_1 z^3 + a_2 z^2 + a_3\, z + a_4}$$

Siendo los coeficientes:

$$\begin{aligned}
b_1 &= \left(25\,\mathrm{e}^{4T} - 24\,\mathrm{e}^{5T} + 20\,T\,\mathrm{e}^{5T} - 1\right) \\
b_2 &= \left(24\,\mathrm{e}^{5T} - \frac{24}{\mathrm{e}^{T}} - 26\,\mathrm{e}^{4T} - 20\,T - 20\,T\,\mathrm{e}^{4T} + 26\right) \\
b_3 &= \frac{24}{\mathrm{e}^{T}} + \mathrm{e}^{4T} + \frac{20\,T}{\mathrm{e}^{T}} - 25 \\
a_1 &= 100\,\mathrm{e}^{5T} \\
a_2 &= -\left(100\,\mathrm{e}^{4T} + 100\,\mathrm{e}^{5T} + 100\right) \\
a_3 &= -\left(-\frac{100}{\mathrm{e}^{T}} - 100\,\mathrm{e}^{4T} - 100\right) \\
a_4 &= -\frac{100}{\mathrm{e}^{T}}
\end{aligned}$$

Se considera el periodo de muestreo de T = 0.0025 segundos, por lo tanto la función de transferencia de la planta es:

$$G(z) = \frac{0.0237\,z^2 + 0.0821\,z + 0.01759}{164.8721\,z^3 - 414.0545\,z^2 + 339.6662\,z - 90.4837}$$

Luego, la función de transferencia del error viene dada por:

$$E(z) = \frac{1}{Q(z)} R(z)$$

Siendo Q(z) = 1 + Gc(z)G(z), además Q(z) tiene la forma:

$$Q(z) = \frac{D(z)}{C(z)} = \frac{d_0 + d_1 z^{-1} + d_2 z^{-2} + d_3 z^{-3} + d_4 z^{-4} + d_5 z^{-5}}{c_0 + c_1 z^{-1} + c_2 z^{-2} + c_3 z^{-3} + c_4 z^{-4}}$$

Reescribiendo.

$$(d_0+d_1 z^{-1}+d_2 z^{-2}+d_3 z^{-3}+d_4 z^{-4}+d_5 z^{-5})E(z) = (c_0+c_1 z^{-1}+c_2 z^{-2}+c_3 z^{-3}+c_4 z^{-4})R(z)$$

De esta relación se obtiene la ecuación en diferencias:

$$\begin{aligned} e(k) &= -\frac{d_1}{d_0}e(k-1) - \frac{d_2}{d_0}e(k-2) - \frac{d_3}{d_0}e(k-3) - \frac{d_4}{d_0}e(k-4) + \frac{d_5}{d_0}e(k-5) + \ldots \\ & \frac{c_0}{d_0}r(k) + \frac{c_1}{d_0}r(k-1) + \frac{c_2}{d_0}r(k-2) + \frac{c_3}{d_0}r(k-3) + \frac{c_4}{d_0}r(k-4) \end{aligned}$$

Donde los coeficientes vienen dados por:

$$\begin{aligned} d_0 &= a1 \\ d_1 &= q_0\, b_1 + (a_2 - a_1) \\ d_2 &= q_0\, b_2 + q_1\, b_1 + (a_3 - a_2) \end{aligned}$$

$$d_3 = q_0\, b_3 + q_1\, b_2 + q_2\, b_1 + (a_4 - a_3)$$

$$d_4 = q_1\, b_3 + q_2\, b_2 - a_4$$

$$d_5 = q_2\, b_3$$

$$c_0 = a_1$$

$$c_1 = (a_2 - a_1)$$

$$c_2 = (a_3 - a_2)$$

$$c_3 = (a_4 - a_3)$$

$$c_4 = -a_4$$

El método de optimización gradiente descendente busca mostrar la performance de la salida de la planta del sistema en lazo cerrado debido a una entrada al escalón unitario, tal como lo mostrado en la Figura 2.14.

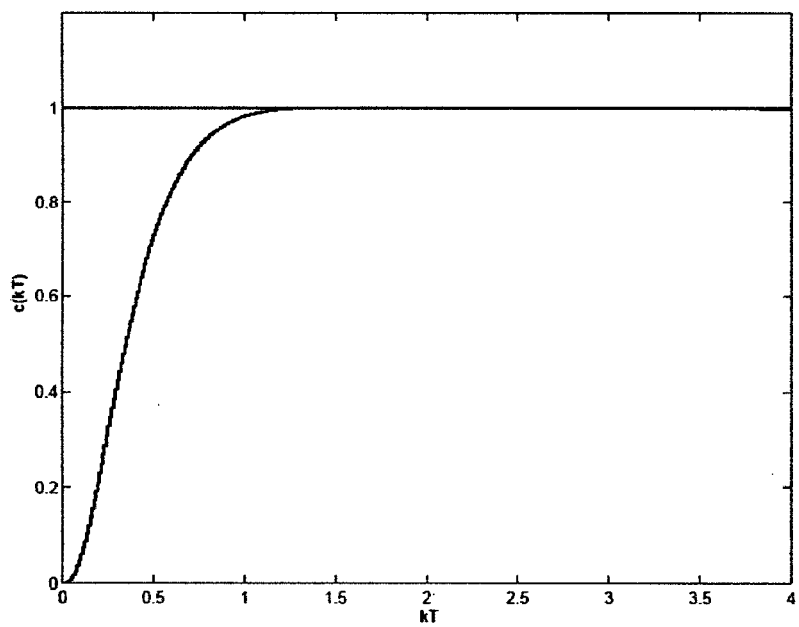


Figura 2.14: Optimización con el método de gradiente descendente

A continuación se tiene el ploteo de la señal de error del controlador optimizado, en este caso podemos decir que el error fue minimizado y la performance de su evolución se muestra en la Figura 2.15.

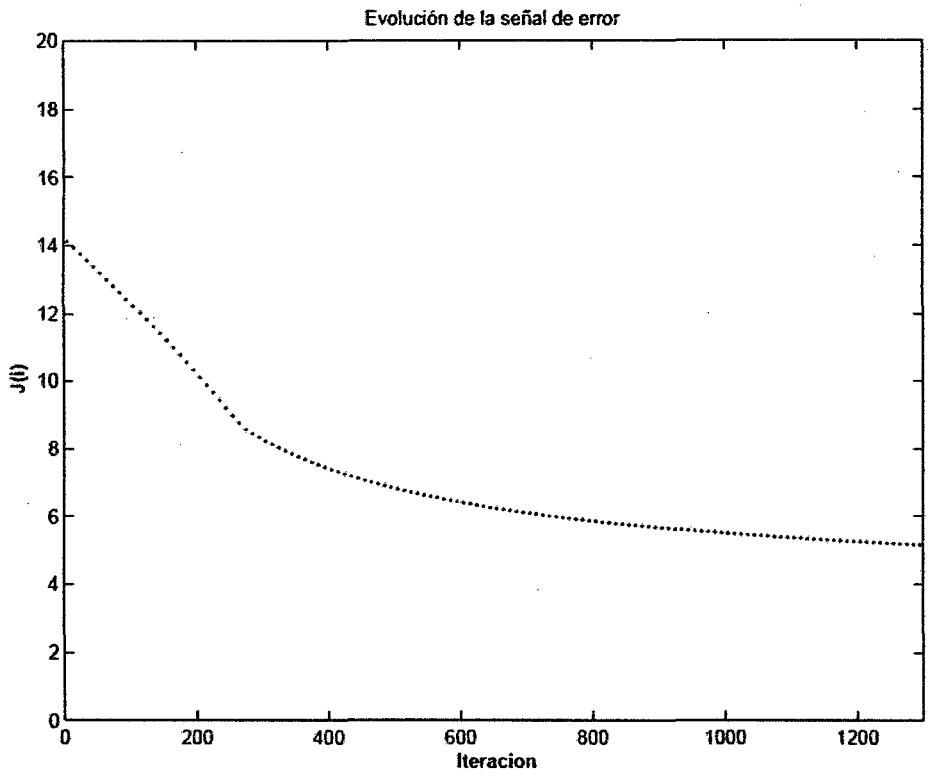


Figura 2.15: Curva de aprendizaje

Desde la Figura 2.15, el error inicial es reducido a cero, esto tomó aproximadamente 5000 iteraciones para que el error sea minimizado.

2.2.4. Algoritmos Genéticos

- **Introducción**

Aunque ya se conocía esta aproximación desde hace 30 años, recientemente ha despertado un gran interés en la búsqueda de algoritmos que presenten analogías con los procesos naturales. Esta tendencia se basa fundamentalmente en la observación de la destreza consumada que poseen los

organismos vivos en la resolución de problemas, los cuales manifiestan una versatilidad muy por encima de programas más refinados.

Así, los investigadores más pragmáticos consideran que, "en lugar de envidiar la eficacia de la evolución natural, debemos emularla". Podemos por tanto tratar de copiar a la naturaleza como una forma de resolución de aquellos problemas en los que no se puedan encontrar soluciones o que èstas no sean lo suficientemente satisfactorias.

Los algoritmos más conocidos que se basan en procesos naturales incluyen, entre otros, los denominados: Programación Evolutiva, Estrategias de Evolución, Algoritmos Genéticos, Enfriamiento Simulado, Sistemas Clasificadores y Redes Neuronales. En este estudio se centrarà en una subclase de estos algoritmos, aquellos que se basan en los principios de la evolución natural. Tales algoritmos mantienen una población de soluciones potenciales, y poseen algún proceso de selección basado en la adecuación de las soluciones a su entorno (supervivencia de los más aptos) y algunos operadores de recombinación.

Particularmente se habla de los Algoritmos Genéticos (AGs), los cuales representan una técnica paralela de búsqueda que emula las leyes de la evolución, para tratar de encontrar soluciones óptimas a problemas complejos de optimización, en nuestro caso hallar los controladores óptimos, y cuyo proceso de evolución de la búsqueda genética está basado en: la diversidad de la población y en la presión selectiva (obligar a que sólo los mejores individuos sobrevivan).

La estructura del algoritmo genético puede verse la Figura 2.16:

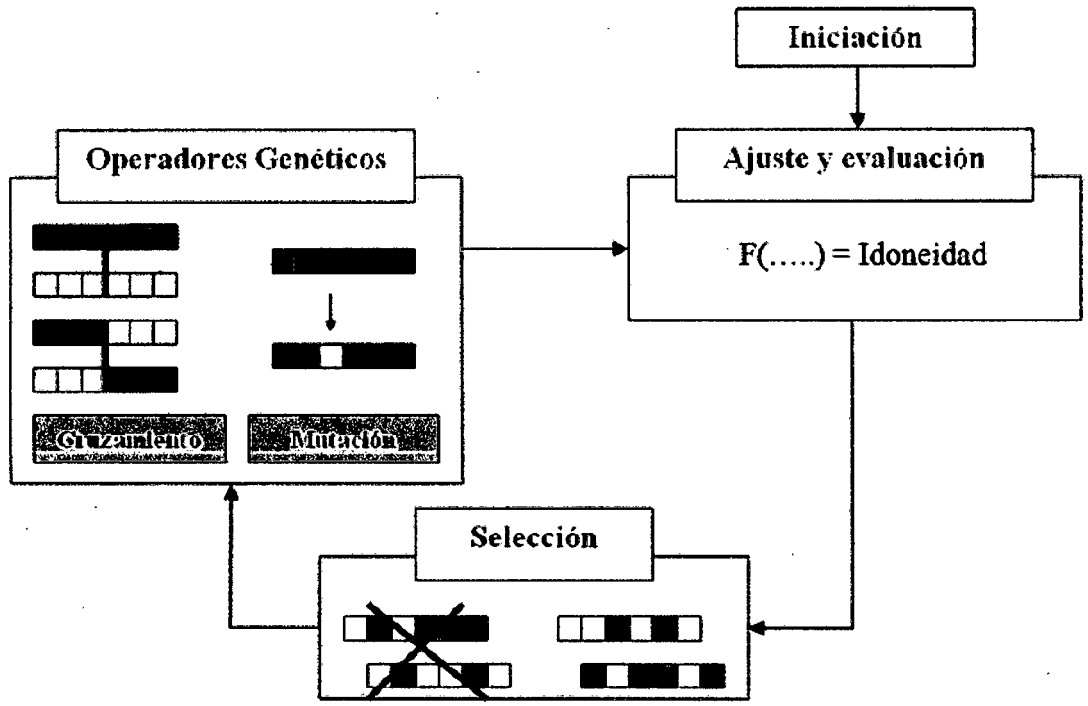


Figura 2.16: Estructura genérica y funcionamiento de un algoritmo genético

Se observa que el proceso de optimización comienza con la iniciación, consistente en la generación, regularmente aleatoria, de una población de individuos.

Luego se hace un ajuste y evaluación de dicha población de acuerdo a funciones de eficiencia o idoneidad. Una vez realizado este proceso, se seleccionan los mejores individuos, para lo cual existen diversas técnicas, como son: la selección por *Rueda de Ruleta* y la *Selección por Torneo*, el estudio se centra principalmente en la primera. Los individuos seleccionados entrarán en la formación de la próxima generación, no sin antes ser afectados por operadores genéticos: cruzamiento y mutación, los cuales simulan el proceso de reproducción de los seres vivos.

Para terminar esta introducción y antes de pasar a detallar cada uno de los puntos que conforman los AGs, es necesario definir algunos conceptos inherentes de la parte genética, los cuales son necesarios para entender el comportamiento de los AGs, y que serán presentados a continuación:

- **Genoma:**

Todos los parámetros que definen a todos y cada uno de los individuos de la población.

- **Genotipo:**

La parte del genoma que describe a un individuo concreto. En esta interpretación son los genes que describen un controlador concreto.

- **Gen:**

Cada uno de los parámetros que describen a un individuo. En esta interpretación los genes codifican parámetros de un controlador concreto, que en nuestro caso, puede ser representado por las ganancias *Kp*, *Ki* y *Kd* que constituyen los parámetros del controlador PID. Al conjunto de genes se le denomina también cromosoma.

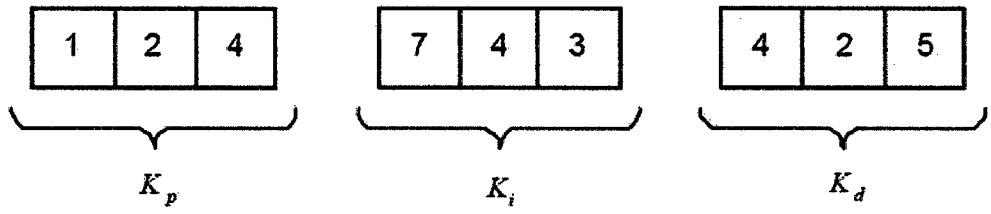
- **La Codificación**

El método más utilizado para la codificación de los genes es el uso de una cadena de longitud y secuencias fijas. Los parámetros del controlador que se ha decidido ajustar, según los criterios que se han expuesto antes, se almacenan en esta cadena como secuencias fijas de números. De esta manera, el número de genes (en nuestro caso parámetros) resulta fijo, y su valor se guarda siempre en una misma posición de la cadena.

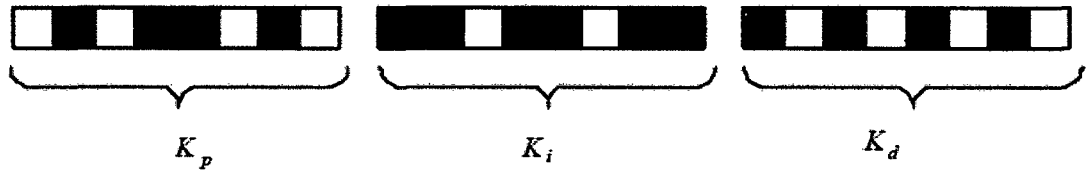
Por otro lado, también se decide cómo codificar estos números; son

habituales codificaciones binarias de 8 bits y de 16 bits, aunque también se hace uso de una codificación en punto fijo o punto flotante. Asimismo, puede emplearse una codificación basada en reservar cierto número de bits para cada gen, y codificar su valor mediante el número de unos presentes, con independencia de su posición. El tipo de codificación seleccionado para los genes influirá en cómo implementar operadores (como el de mutación), así como en la cantidad de memoria necesaria para almacenar el genoma.

Por ejemplo, como se ve en la Figura 2.17 los valores de los parámetros PID pueden codificarse de dos maneras distintas:



Codificación con números con punto flotante



Codificación binaria de 8 bits de anchura fija

Figura 2.17: Distintos métodos de codificación del cromosoma

En este punto puede surgir la pregunta de ¿usar números binarios o números con punto flotante?. Para responder esto es necesario conocer, que la representación binaria tradicionalmente usada en los AGs tiene algunas desventajas cuando se aplica a problemas de control multidimensionales y de

gran precisión numérica. Por ejemplo, para 100 variables con dominios en los rangos [-500, 500] donde la precisión requerida es de 6 dígitos después del punto decimal, la longitud de cada variable es 30, es decir se requieren 29 dígitos binarios que representan el número 500.000000 y un dígito que representa el signo, por lo tanto la longitud total del vector solución binario es de 3000, lo cual a su vez genera un espacio de búsqueda de cerca 23000. Para este tipo de problemas los AGs evolucionan muy pobremente.

Experimentos realizados por investigadores Koza y Michalewicz, indican que la representación en punto flotante es más rápido, más consistente de corrida a corrida, y provee una gran precisión (especialmente con grandes dominios, donde la codificación binaria requeriría, prohibitivamente, grandes representaciones.) Al mismo tiempo su desempeño puede aumentar por operadores especiales para alcanzar gran exactitud (aún más grande de lo que ofrece la representación binaria.)

Es por ello que sòlo se utiliza la codificación binaria para mostrar ejemplos sencillos, ya que en una implementación de control en tiempo real se podrá usar la codificación de punto flotante.

- **La Población**

Hay dos interpretaciones para el concepto de individuo y población en la optimización de controladores. La primera, habitualmente llamada Pitts, que es la más utilizada, y que se seguirá de aquí en adelante, entiende por población a un conjunto de individuos, en nuestro caso, cada individuo será un controlador PID completo.

La segunda interpretación, habitualmente llamada Michigan, consiste en considerar como individuo a las habilidades de un controlador, de forma que la población representa el conjunto de todas sus habilidades. Este tipo de interpretación se ha utilizado en el control de sistemas en tiempo real y mayormente en robótica industrial. Por ejemplo, en el caso de robótica, un individuo está formado por el conjunto de reglas que permiten al robot agarrar un objeto; la población está formada por individuos especializados en ciertas tareas, y durante la evolución se seleccionan los mejores para cada proceso.

Usualmente, los individuos de la primera población de un AG son generados por medio de un muestreo aleatorio uniforme dentro del espacio de las soluciones como se mostró anteriormente en la introducción. Sin embargo, algunas excepciones están contempladas según:

- Muestreo Aleatorio No Uniforme

Existen algunas codificaciones para las cuales un algoritmo rápido de muestreo aleatorio uniforme puede resultar bastante complicado, usándose entonces algoritmos de muestreo aleatorio no uniforme

- Muestreo Aleatorio con Información Híbrida

En ciertas aplicaciones prácticas puede resultar de gran importancia introducir en la población inicial buenas soluciones conocidas, obtenidas de algún otro método de resolución propio del problema, como pueden ser las reglas de sintonización de controladores PID mencionadas anteriormente, con lo cual combinado con el operador elitista, se garantiza que el AG no obtendrá peores soluciones que el método propio. Sin embargo, esta técnica debe implementarse con cuidado, ya que si estas soluciones introducidas en la población están muy por encima de la adecuación media de la población, es bastante probable que

ocurra convergencia prematura. Como solución al problema, estas buenas soluciones pueden introducirse cuando hayan transcurrido algunas generaciones.

- Iniciación Parcialmente Enumerativa

La motivación de este método es asegurar que todos los buenos esquemas necesarios para la solución están presentados en la población inicial, por lo que al menos un representante de estos esquemas es generado.

El caso más frecuente, no obstante, consiste en partir de una población aleatoria, pues como se ha explicado antes, los AGs son especialmente indicados para realizar la búsqueda global de soluciones óptimas en un gran espacio.

El tamaño de la población, es una de las elecciones más importantes que encara cualquier usuario de los AGs, y puede ser crítico en muchas aplicaciones. Si el tamaño de la población es muy pequeño, los AGs pueden converger rápidamente; en cambio, si es demasiado grande, los AGs desgastan mucho tiempo computacional en procesar a todos los individuos. Como se dijo en la introducción, son dos las consideraciones importantes en el proceso de evolución de la búsqueda genética: la diversidad de población y la presión selectiva.

Claramente ambos factores están influenciados por el tamaño de la población. Por ejemplo para aplicaciones del sistema pelota y aro, se puede determinar tamaños de población intermedio que va de los 50 a los 100 individuos, ya que poblaciones mayores consumen mucho tiempo, y en control lo que se requiere es disminuir dicho tiempo, y por ende disminuir el periodo de muestreo.

- **La Función de Evaluación**

La fase siguiente del proceso es la evaluación, en la cual se deja que cada

uno de los controladores que forman la población actúe controlando el sistema, normalmente mediante una simulación en el caso del diseño de sistemas de control fuera de línea, siendo evaluados mediante una función de eficiencia o idoneidad (fitness,). Para este caso, la etapa de la simulación es normalmente la que más tiempo requiere, pues se han de simular cada uno de los controladores de la población durante el tiempo necesario para evaluar su eficiencia, y en un número suficiente de situaciones de control. Para el caso del diseño de sistemas de control en línea, como se verá más adelante, la función de idoneidad se evalúa utilizando un modelo linealizado o aproximado de la planta, en un número de periodos de muestreo pequeño pero suficiente para poder predecir la dinámica de la planta que utiliza cada individuo. La definición de esta función de eficiencia es fundamental para el éxito del uso de los AGs para la resolución de un problema dado. A partir de ella, el diseñador decide qué comportamientos no se han de potenciar, cuáles si, y en qué medida. Una descripción más detallada del diseño de esta función se verá también más adelante.

Una práctica ampliamente aceptada para mantener unos niveles de competición apropiados durante el proceso de evolución, consiste en el escalado de la adecuación bruta (valor de la función idoneidad), con lo cual se previene el dominio de los superindividuos que pueden dar lugar a la convergencia prematura.

Entre los mecanismos de escalado más importantes se incluyen los siguientes:

- **Escalado Lineal**. Con este método la adecuación bruta de un f_t^i cromosoma es escalada usando una ecuación lineal de la forma:

$$f_i'^t = a\, f_i^t + b$$

Donde los coeficientes *a* y *b* son normalmente seleccionados de forma que:

- El valor de la adecuación bruto será igual al valor de la adecuación escalada.
- La mejor adecuación sea un múltiplo (usualmente 2) de la adecuación media.

El escalado lineal trabaja bastante bien excepto cuando aparecen valores de adecuación negativos, los cuales deben de ser tratados de otra manera.

- **Truncamiento Sigma**. Para tratar valores de adecuación negativos e incorporar información dependiente del problema, el truncamiento sigma calcula la nueva adecuación de la siguiente forma:

$$f_i'^t = f_i^t + (\bar{f}^t - c\sigma^t)$$

Donde σ^t es la desviación estándar de la población, y *c* es elegido como un múltiplo razonable de la desviación estándar de la población (entre 1 y 3.) Los resultados negativos ($f't < 0$) son fijados a cero.

- **Escalado de Potencia**. Con este método, la adecuación escalada se toma como alguna potencia específica de la adecuación bruta:

$$f_i'^t = f_i^{tk}$$

Para algún *k* cercano a 1.

- **Escalado Limitador**. A través de este método se fija un límite a la adecuación bruta que al principio del algoritmo puede iniciarse con valores grandes, (debido a la magnitud del error en estado transitorio por ejemplo. Tiene la siguiente forma:

$$f_i'^t = \frac{\alpha}{F(f_i) + \alpha}$$

Donde α determina el límite de una función de la adecuación bruta, generalmente dicha función es un factor cuadrático.

- **La Función de Selección**

La fase siguiente es la de selección, en la cual se simula el proceso de selección natural de los individuos en cada generación. En este sentido, se debe de seleccionar qué individuos han de transmitir su genotipo a la generación siguiente.

Dada una población de *M* individuos, son posibles diversas alternativas:

(*a*) *Sólo el mejor se selecciona.* De toda la población se elige al individuo que tiene una mejor evaluación, y solamente se emplea su genoma para crear la siguiente generación.

(*b*) *Sólo los mejores se seleccionan.* De toda la población se eligen los *n* individuos ($n \ll M$) que tienen una mejor evaluación, y para crear la siguiente generación solamente se utiliza su genoma.

(*c*) *Todos pueden seleccionarse.* Cada individuo de la población se selecciona con una probabilidad mayor cuanto mejor sea su evaluación.

La primera de las alternativas permite una rápida convergencia a la solución, pero puede estancarse en un mínimo local. Ello resulta especialmente probable en aquellos problemas en los que se debe de alcanzar una solución de compromiso entre varios objetivos contrapuestos, ya que si se selecciona sólo el mejor, el genoma se empobrece (en cuanto a variedad), y los individuos pueden tender a especializarse en conseguir parte de los objetivos, dando malos resultados en el resto. Este problema puede resolverse con el uso de la segunda alternativa y una adecuada elección de *n/M*. La última alternativa es adecuada para buscar soluciones a problemas con un gran espacio de búsqueda y para los que no se conoce una buena aproximación a la solución; pero cuenta con el inconveniente de requerir más tiempo de cálculo y poblaciones mayores, por lo que en algunos casos resulta inaplicable.

En este caso, se elige la segunda alternativa, en donde se seleccionan individuos que tengan mejor evaluación, para utilizarlos en la próxima generación. Para la adecuada elección de estos individuos existen diversos métodos, siendo los más importantes la selección por Rueda de Ruleta, y la selección por Torneo, los mismos que serán descritos a continuación.

- **Selección por Rueda de Ruleta**

En este método la probabilidad de un individuo para reproducirse en la siguiente generación es proporcional al valor de la función idoneidad evaluada de dicho individuo. Esto se puede apreciar mejor en el siguiente algoritmo de la selección por Ruleta:

- Se calcula el valor de la función idoneidad $eval(v_i)$ para cada uno de los cromosomas v_i ($i = 1, \ldots, tam\text{-}poblac$)

- Luego se encuentra la suma total de las funciones idoneidad, de toda la población.

$$F = \sum_{i=1}^{tam\_poblac} eval(v_i)$$

- Se calcula la probabilidad de selección p_i para cada cromosoma v_i (i =1, . . . , *tam–poblac*):

$$p_i = \frac{eval(v_i)}{F}$$

- Se calcula la probabilidad acumulativa q_i para cada cromosoma v_i (i =1, . . . , *tam–poblac*):

$$q_i = \sum_{j=1}^{i} p_j$$

El proceso de selección está basado en el giro de la Ruleta *tam–poblac* veces, cada vez se selecciona un solo cromosoma para una nueva población de la manera siguiente:

- Se genera un número aleatorio (float) r que se encuentre en el rango [0. . . 1].
- Si $r < q1$ entonces se selecciona el primer cromosoma ($v1$); de otra forma se selecciona la i-ésima cromosoma v_i ($2 \leq i \leq$ *tam–poblac*), tal que $q_{i-1} < r \leq q_i$.

Obviamente, algunos cromosomas serán seleccionados más que otros, esto está en concordancia con la segunda alternativa de selección, donde los

mejores cromosomas conseguirán más copias, el promedio todavía permanecerá, y los peores cromosomas morirán.

- **Selección por Torneo**

Este procedimiento estocástico de selección fue propuesto a Brindle por Wetzel en 1983. En este método (también llamado método de Ranking), se combina la idea de las categorías, en una manera interesante y eficiente. Este método (en una sola iteración) selecciona cierto número ***k*** de individuos y busca el mejor de este conjunto de *k* elementos para usarlo en la siguiente generación. Cuando dos toros luchan por el derecho de procrearse con una vaca determinada, la selección por torneo está ocurriendo. Este proceso es repetido *tam-poblac* número de veces. Está claro que grandes valores de ***k***, se incrementan la presión selectiva de este procedimiento; valores típicamente aceptados por muchas aplicaciones son $k = 2$ (también llamado tamaño del torneo).

Goldberg propuso una forma interesante de encontrar el par de individuos seleccionados para el torneo. Esto es, las probabilidades de selección son calculadas normalmente y un sucesivo par de individuos son sustraídos de la población usando la selección de Rueda de Ruleta. Después de haber sustraído el par de individuos, aquel que tenga mayor valor de idoneidad será declarado ganador, siendo insertado en la nueva población, mientras que el otro individuo es sacado definitivamente. Este proceso, continúa hasta que la nueva población esté completa.

Una consideración adicional para encontrar soluciones óptimas dentro del espacio de búsqueda, es el operador elitista, en el cual se fuerza al mejor individuo de la población en cada generación siguiente generación. Aunque este

modelo puede incrementar el problema de la convergencia prematura, en la mayoría de los casos mejora el rendimiento del AG.

- **Operadores de Cruzamiento y Mutación**

Los operadores que se utilizan realizan modificaciones sobre el genoma, simulando la evolución del genoma de los seres vivos causada por la reproducción sexual y otros factores, como las mutaciones. Los principales operadores utilizados en los AGs son el cruzamiento y la mutación.

- **Operador Cruzamiento**

El cruzamiento permite simular la combinación del genoma entre los individuos. Este puede realizarse sobre parejas de individuos o sobre toda la población. Cuando se realiza sobre parejas de individuos, mejor conocidos como padres y de acuerdo a la probabilidad de cruzamiento, al que llamaremos ***pc***, se copia el genoma en dos memorias; de forma aleatoria se selecciona un punto en esta memoria y se cruzan las dos mitades de cada copia de sus genotipos, como se puede apreciar en la Figura 2.18.

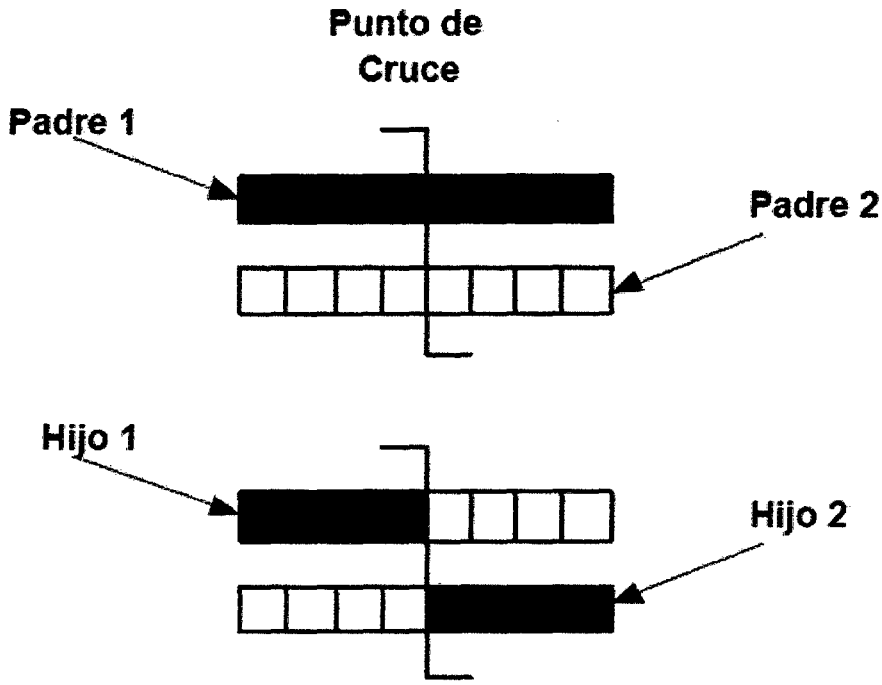


Figura 2.18: Cruzamiento del genoma entre individuos

El cruzamiento permite que las mejores cualidades de los padres se combinen en sus descendientes. En problemas en los que se pretende que el controlador haya de alcanzar simultáneamente varios objetivos, se espera que los descendientes en algún momento de la evolución puedan heredar de cada uno de los padres el conjunto de reglas que permiten lograr cada uno de los objetivos.

Una variante a la forma en que se cruzan los genomas de los individuos es que si se escogen por ejemplo 4 genes a cruzarse, estos se cruzan de forma aleatoria dentro del genoma. Esto se puede visualizar de la siguiente manera: si se tienen 2 cromosomas de longitud 10, [1111111111] y [3333333333], y se elige que se cruzan cuatros genes, esta operación se puede hacer de la siguiente manera, [1133313111] y [3311131333], lo cual representa un método de cruzamiento que no es estándar pero que ofrece buenos resultados y que ha demostrado que es más efectivo que el método tradicional de cruzamiento.

Por otro lado, se conoce, que durante la reproducción de los seres vivos se producen errores en la copia del genoma, a los que asignaremos una cierta probabilidad, que llamaremos *pm*. Este efecto introduce diversidad del genoma, y permite explorar nuevas soluciones a los problemas. A este fenómeno se le conoce como mutación y dentro de los AGs se modela a través del operador Mutación que se detalla a continuación.

- **Operador Mutación**

Este operador supone seleccionar un punto de mutación y luego modifica el parámetro almacenado con una probabilidad *pm*. Si la codificación es de tipo binario se modifica uno de los bits, como se puede apreciar en la Figura 2.19, mientras que si la codificación es de tipo punto fijo o punto flotante, se suele realizar un cambio aleatorio del gen que ha sido escogido para mutarse dentro de un rango limitado (por ejemplo de 0 a 9).

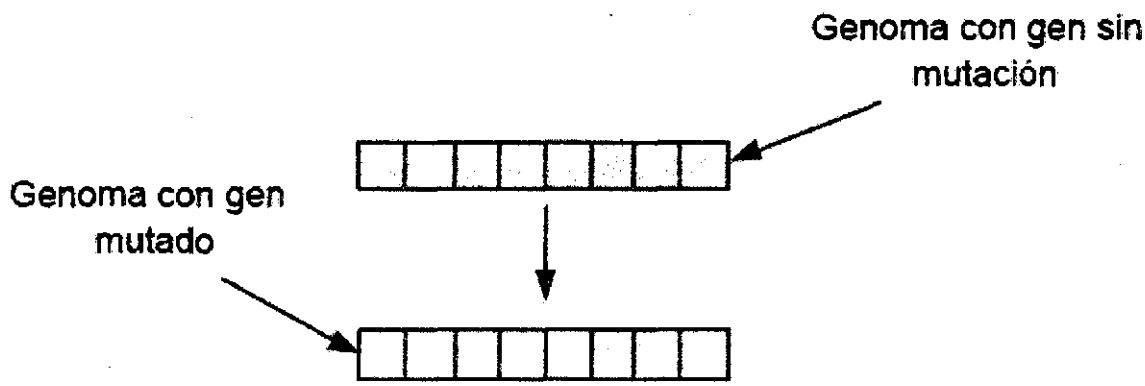


Figura 2.19: Operador mutación actuando sobre un punto del genoma

Según los trabajos realizados por De Jong's, la probabilidad de cruzamiento p_c se elige de tal manera, que se pueda llevar a cabo el máximo número de cruzamientos, es por eso que se elige un valor alto para p_c. Mientras que el valor de la probabilidad de mutación p_m, se elige en forma inversamente

proporcional al tamaño de la población, lo cual nos dice que a mayor número de individuos, menor es el efecto de las mutaciones.

- **Optimización de una Función Simple**

En esta sección veremos en acción las características básicas de los AGs utilizando como ejemplo la optimización de una función simple de una sola variable. Esta función está definida como:

$$f(x) = x\ \sin(10\pi x) + 1$$

Y su gráfica se muestra en la Figura 2.20.

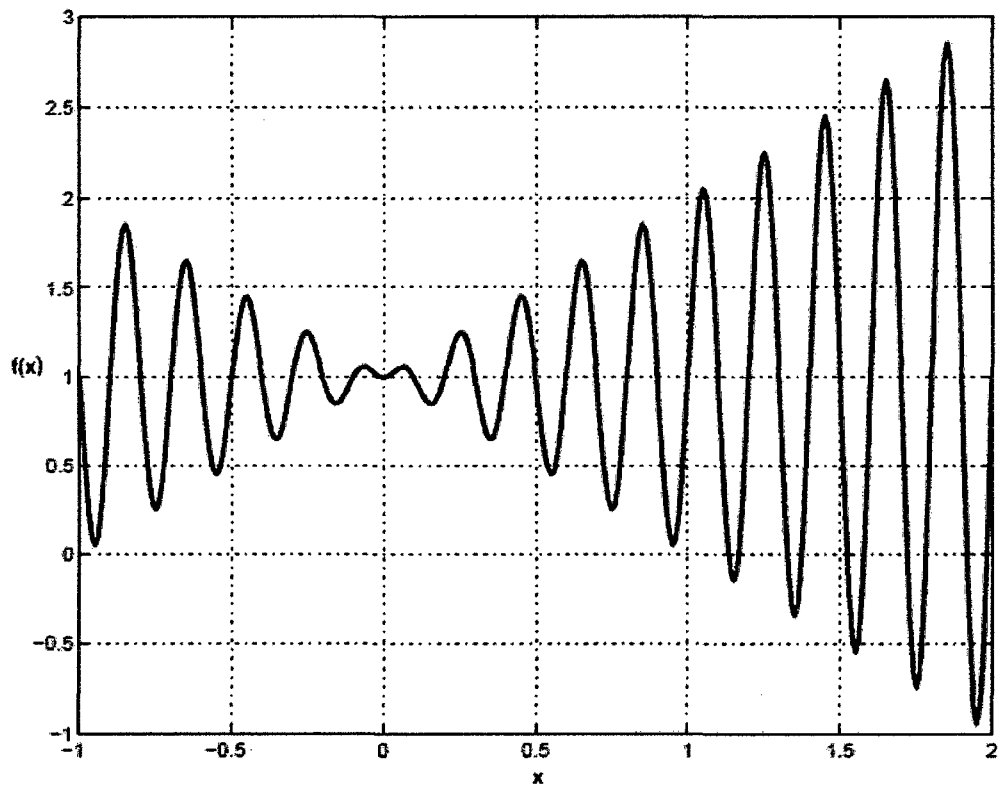


Figura 2.20: Gráfica de la función f(x)

El problema consiste en encontrar el valor de x dentro del rango [−1 . . . 2], que maximice la función f, por ejemplo encontrar un $x0$ tal que: $f(x0) \geq f(x)$ para todos los $x \in [-1 \ldots 2]$.

Se observa, que es relativamente fácil analizar la función f. Los ceros de la primera derivada f' podrían ser determinados:

$$f'(x) = \sin(10\pi x) + 10\pi x \cos(10\pi x) = 0$$

Dicha fórmula es equivalente a:

$$\tan(10\pi x) = -10\pi x$$

Es claro que las ecuaciones de arriba tienen un número infinito de soluciones.

$$x_i = \frac{2i-1}{20} + \epsilon_i \quad para,\ i = 1, 2, \ldots$$

$$x_0 = 0$$

$$x_i = \frac{2i-1}{20} + \epsilon_i \quad para,\ i = -1, -2, \ldots$$

Donde los términos, representan secuencias decrecientes de números reales (para i = 1, 2, . . . y i = −1,−2, . . .) cercanas a cero. Hay que notar también que f alcanza su máximo local para xi, si i es un entero impar, y su mínimo local para x_i, si i es un entero par (ver Figura 2.20).

Ya que el rango del problema es $x \in [-1 \ldots 2]$, la función alcanza su valor máximo para dicho rango en x_{19} =37/20 + ϵ_{19}, donde $f(x_{19})$ es ligeramente más grande que f(1.85) = 1.85 sin(18π +π/2) + 1 = 2.85. Se asume que se quiere

construir un AG para resolver el problema planteado arriba, por ejemplo hallar el máximo de dicha función *f*.

- **Representación**

Usaremos un vector binario como un cromosoma, para representar valores reales de la variable *x*. La longitud del vector, depende de la precisión requerida, el cual, en este ejemplo, son seis lugares después del punto decimal.

El dominio de la variable *x* tiene una longitud de 3, los requerimientos de la precisión implican que el rango [−1 . . . 2] será dividido en al menos $3 * 1000000$ partes igualmente espaciados. Esto significa que se requiere de 22 bits como un vector binario (cromosoma):

$$2097152 = 2^{21} < 3000000 \leq 2^{22} = 4194304$$

El mapeo de una palabra binaria < *b*21, *b*20, . . . , *b*0 > en un número real *x* del rango [−1 . . . 2] es sencillo y es completado en dos pasos:

- Convertir la palabra binaria < *b*21, *b*20, . . . , *b*0 > de la base 2 a la base 10:

$$(< b_{21}, b_{20}, \ldots, b_0 >)_2 = \left(\sum_{i=0}^{21} b_i 2^i \right)_{10} = x'$$

- Encontrar el correspondiente número real *x*:

$$x = -1 + x' \frac{3}{2^{22} - 1}$$

Donde −1 es el límite izquierdo del dominio y 3 es la longitud de dicho dominio. Por ejemplo, un cromosoma (1000101110101000111) representa el número 0.637197, debido a que:

$$x' = (10001011101010001 11)_2 = 2288967$$

y

$$x = -1 + 2288967\frac{3}{4194303} = 0.637197$$

Está claro que los cromosomas:

$$(00000000000000000000) \quad \text{y} \quad (11111111111111111111)$$

Representan los límites del dominio, −1 y 2 respectivamente.

- **Población Inicial**

El proceso de iniciación es muy simple: Nosotros creamos una población de cromosomas, donde cada cromosoma es un vector binario de 22 bits. Todos los bits para cada cromosoma son inicializados aleatoriamente.

- **Función de Evaluación**

La función idoneidad de evaluación eval para cada vector binario v es equivalente a la función *f*:

$$eval(v) = f(x)$$

Donde el cromosoma *v* representa el valor real de *x*.

Como se vio anteriormente, la función idoneidad de evaluación juega el papel del medio ambiente, evaluando soluciones potenciales en términos de su conveniencia. Por ejemplo, tres cromosomas:

$$v_1 = (10001011101101010001 11)$$

$$v_2 = (00000011100000000010000)$$

$$v_3 = (11100000001111111000101)$$

Corresponden a los valores $x1$ = 0.637197, $x2$ = −0.958973, y $x3$ = 1.627888, respectivamente. Consecuentemente, la función idoneidad los evaluará como sigue:

$$eval(v_1) = f(x_1) = 1.586345$$

$$eval(v_2) = f(x_2) = 0.078878$$

$$eval(v_3) = f(x_3) = 2.250650$$

Claramente, el cromosoma v_3, es el mejor de los tres cromosomas, ya que su función idoneidad retorna el valor más alto.

- **Operadores Genéticos**

Durante la fase de alteración de los AGs nosotros podemos usar dos operadores genéticos clásicos: mutación y cruzamiento. Como se mencionó anteriormente, la mutación altera uno o más genes con una probabilidad igual a la razón de mutación. Se asume que el quinto gen del cromosoma v_3, fue seleccionado para mutarse. Desde que el quinto gen en este cromosoma es 0, lo podemos cambiar a 1. Entonces el cromosoma v_3, después de la mutación será:

$$v_3' = (11101000011111111000101)$$

Este cromosoma representa el valor de $x'_3 = 1.721638$ y $f(x'_3) = -0.082257$. Esto significa que esta mutación en particular resulta en un significativo decrecimiento del valor del cromosoma v_3. Por otra parte, si el décimo gen fuese seleccionado para mutarse en el cromosoma v_3, entonces:

$$v''_3 = (11100000111111111000101)$$

El valor correspondiente de $x''_3 = 1.630818$ y $f(x''_3) = 2.343555$, que constituye un mejoramiento sobre el valor original de $f(x_3) = 2.250650$.

Seguidamente se ilustra el uso del operador cruzamiento sobre los cromosomas v_2 y v_3. Se asume que el punto de cruzamiento, fue (aleatoriamente) seleccionada después del 5to gen:

$$v_2 = (00000|01110000000010000)$$

$$v_3 = (11100|00000111111000101)$$

Los dos descendientes resultantes son:

$$v'_2 = (00000|00000111111000101)$$

$$v'_3 = (11100|01110000000010000)$$

La evaluación de estos descendientes da como resultado:

$$f(v'_1) = f(-0.998113) = 0.940865$$

$$f(v'_2) = f(1.666028) = 2.459245$$

Hay que notar que el segundo descendiente tiene una mejor evaluación que cualquiera de sus padres.

- **Parámetros**

Para este problema en particular y de acuerdo a los criterios de elección de

parámetros dados en las secciones anteriores, se usaron los siguientes paràmetros: tamaño de población = 50, probabilidad de cruzamiento p_c = 0.9, probabilidad de mutación p_m = 0.02. A continuación se presenta algunos resultados experimentales obtenidos para la resolución del problema planteado, en el software Matlab.

- **Resultados**

En la Tabla 2.3 se proporciona el número de generaciones, que conforme van evolucionando, se puede notar el mejoramiento del valor de la función idoneidad de evaluación.

El mejor cromosoma después de 150 generaciones fue:

$$v_{max} = (11110011010001000001 01)$$

El cual corresponde al valor x_{max} = 1.850773.

Como se esperaba, x_{max} = 1.85 + ϵ, y $f(x_{max})$ es ligeramente más grande que 2.85.

Tabla 2.3: Resultado de 150 generaciones

Número de Generaciones	Función de Evaluación
1	1.441942
6	2.250003
8	2.250283
9	2.250284
10	2.250363
12	2.328077
39	2.344251
40	2.345087
51	2.738930
99	2.849246
137	2.850217
145	2.850227

- **Algoritmo Genético Descriptivo con Matlab**

Los algoritmos genéticos corresponden a la clase de métodos estocásticos de búsqueda. Mientras la mayoría de estos métodos operan sobre una única solución, estos algoritmos operan en una población de soluciones. La idea básica, inspirada en los procesos evolutivos en biología, es que el contenido genético de una población contiene potencialmente la solución, o una solución mejor, a un problema dado de adaptación. Esta solución puede estar inactiva porque la

combinación genética adecuada está diseminada entre varios sujetos. Solo la asociación de genomas distintos puede llevar a la activación de la solución.

Crudamente, el mecanismo evolutivo procede así sobre una población, algunos individuos son seleccionados para la reproducción, con más oportunidades para los mejor adaptados al ambiente. Durante la reproducción, los nuevos individuos de la población resultan de modificaciones e intercambio genético de los padres. Una vez que se renueva la población, el proceso recomienza. Es decir que hay dos espacios donde opera la evolución. Por una parte, a nivel de los individuos físicos (fenotipo), que deben adaptarse para ser seleccionados. Y luego, a nivel de la información genética (genotipo), a través de los operadores que intercambian y varían la información genética.

La información genética está codificada en los cromosomas, que son secuencias de genes, cada uno de los cuales codifica una característica particular del individuo. El operador de mutación introduce cierta aleatoriedad en la búsqueda simplemente cambiando unos genes por otros, contribuyendo a una exploración "azarosa" en el espacio genético. El operador de crossover, en cambio, es una recombinación de la información durante la reproducción de los individuos seleccionados, [2], [6].

- **Los Algoritmos Genéticos contra Métodos Tradicionales**

Los algoritmos genéticos son substancialmente diferentes a las técnicas más tradicionales de búsqueda y la optimización. Las cinco principales diferencias son:

a. Los algoritmos genéticos buscan una población de puntos en paralelo, no de un único punto.

b. Los algoritmos genéticos no requieren hallar información ni otro conocimiento auxiliar; sólo una función objetiva y correspondientes niveles que influyan en la dirección de la búsqueda.

c. Los algoritmos genéticos utilizan reglas de transición probabilística, mas no reglas deterministas.

d. Los algoritmos genéticos trabajan en codificar un conjunto de parámetros no el conjunto de parámetros mismo (excepto cuando los individuos verdadero-valorados son utilizados).

e. Los algoritmos genéticos pueden proporcionar varias soluciones potenciales a un problema dado y la elección del final son dejados para el usuario. Inicializar la Población del Algoritmo Genético. El código siguiente es basado en el Toolbox de Algoritmos Genéticos de Matlab.

- **Población del Algoritmo Genético**

Para dar inicio al desenvolvimiento de un algoritmo genético con Matlab, éste debe ser primero entendido por algunas variables que van a significar un desempeño para describir el algoritmo genético, tales variables, se encargarán de optimizar las ganancias de un controlador de PID, a continuación se detallarà cada línea de código:

```
populationSize = 100;
variableBounds = [0 10;0 10;0 10];
evalFN = 'PID_objfun_MSE';
evalOps = [];
options = [1e-6 1];
```

initPop = initializega (populationSize, variableBounds, evalFN, ...

evalOps, options);

PopulationSize

La primera etapa consiste en escribir un algoritmo genético de tal modo que se cree una población. Esta orden define el tamaño de población. populationSize =100;

VariableBounds

En este proyecto de tesis se utiliza algoritmos genéticos para optimizar las ganancias de un controlador de PID, por ello habrá tres cadenas de ganancias asignadas a cada miembro de la población, estos miembros serán comprendidos de ganancias Proporcional (*P*), Integral (*I*) y derivativa (*D*) que será evaluada a través del algoritmo genético. Los tres términos son ingresados en el algoritmo genético a través de la declaración de una matriz llamada **variablebounds** de tres filas. El número de filas en la matriz de **variablebounds** representa el número de términos en cada miembro de la población. Las líneas debajo ilustran una población de cien miembros para ser inicializada con valores al azar seleccionado entre 0 y 10.

variableBounds=[0 10

0 10

0 10]

EvalFN

La función de evaluación es la declaración del nombre del archivo que contiene la función objetivo.

evalFN = 'PID_objfun_MSE';

Options

Aunque a menudo los ejemplos son código binario codificado, esto fue sólo para propósitos ilustrativos. Las cadenas binarias tienen dos principales inconvenientes:

- Toman más largo evaluar debido al hecho que tienen que ser convertidos desde o hacia código binario.
- Las cadenas binarias pierden precisión durante conversión.

A consecuencia de esto y el hecho que utilizan menos memoria, números reales (punto flotante) serán utilizados para codificar la población. Se tiene el orden de opciones, donde el término 1e-6 es la precisión flotante del punto y 1 es el término que indica que números verdaderos son utilizados.

Options = [1e-6 1];

Initialisega

Esta orden combina todos los términos anteriormente descrito y crea una población inicial de 100 miembros valorados verdaderos entre 0 y 10 con precisión decimal 6.

initPop = initializega (populationSize, variableBounds, evalFN,...

evalOps, options);

- **Inicializando el Algoritmo Genético**

Un algoritmo genético es inicializado de acuerdo con las líneas de código Matlab siguientes:

```
bounds = [-100 100;-100 100;-100 100];
evalFN = 'PID_objfun_MSE';
evalOps = [];
startPop = initPop;
opts = [1e-6 1 0];
termFN = 'maxGenTerm';
termOps = 100;
selectFN = 'normGeomSelect';
selectOps = 0.08;
xOverFNs = 'arithXover';
xOverOps = 4;
mutFNs = 'unifMutation';
mutOps = 8;
```

Bounds

Significa para el algoritmo genético una búsqueda dentro de un conjunto utilizando esta orden. Estos puede ser diferente a los iniciales de la población y ellos definen el espacio entero de la búsqueda para el algoritmo genético.

StartPop

Da inicio a la población del algoritmo genético, es definido como la población describió en la sección anterior, es decir "initPop".

Opts

Las opciones para el algoritmo genético consisten en la precisión de los valores es decir 1e-6, la declaración de valores codificados verdaderos, 1, y una petición para el progreso del algoritmo genético para ser demostrados, 1, o suprimidos, 0.

TermFN

Es la declaración de la función de terminación para el algoritmo genético. Es utilizada para terminar el algoritmo genético con criterio una vez que ha sido encontrado. En este proyecto, cada algoritmo genético será terminado cuando alcanza un cierto número de generaciones que utilizan la función de **maxGenTerm**. Este método de la terminación tiene en cuenta más control sobre el compilatiempo (es decir la cantidad de tiempo toma para el algoritmo genético para alcanzar su criterio de terminación) del algoritmo genético cuando comparado con otros criterios de terminación como por ejemplo el criterio de terminación de convergencia.

TermOps

Esta orden define las opciones, para la función de terminación. En este ejemplo las opciones de terminación son puestas a 100, que significa que el algoritmo genético reproducirá cien generaciones antes de terminar. Este número puede ser alterado para convenir mejor los criterios de convergencia del algoritmo genético es decir si converge rápidamente entonces las opciones de terminación deben ser reducidas.

SelectFN

Normalización de selección geométrica (normGeomSelect) es el proceso de selección primario que será utilizado en este proyecto de tesis. La caja de

herramientas proporciona otras dos funciones de selección, selección de Torneo y selección Rueda de Ruleta. La selección del torneo tiene un tiempo más largo de compilación que los demás y corre en conjunto con el tiempo del algoritmo genético, mientras que la opción de la rueda de la ruleta está fuera de lugar y es poco apropiada usarla debido a las razones mencionados en secciones anteriores.

SelectOps

Se usa la opción de **normGeomSelect**, es el único parámetro que tiene que ser declarado. Es la probabilidad de seleccionar el cromosoma más conveniente de cada generación, en este ejemplo, esta probabilidad es puesta a 0.05.

XOverFN

Es un procedimiento aritmético escogido como el procedimiento de paso. El paso único del punto es demasiado simplista y trabaja efectivamente en un cromosoma con tres alleles (Cada una de las posibles formas en las que existe un gen a consecuencia de una o más mutaciones). El paso heurístico fue desechado porque realiza el procedimiento de paso de varios tiempos y entonces escoge el mejor. Esto aumenta el tiempo de compilación del programa y es indeseable. El procedimiento aritmético del paso es utilizado específicamente para números de punto flotante y es la opción ideal de paso para el uso en este proyecto.

XOverOptions

Es el número de puntos de paso especificado. En el ejemplo mostrado, el número de puntos de pasos es puesto a cuatro.

MutFNs

El **multiNonUnifMutation**, o multioperario distribuido no uniforme de mutación, fue escogido para funcionar bien con múltiples variables.

MutOps

Es un operador de mutación que acepta tres opciones cuando se usa la función de **multiNonUnifMutation**. El primer parámetro es el número total de mutaciones, se puso normalmente con una probabilidad de alrededor de 0.1 %. El segundo parámetro es el número máximo de generaciones y el tercer parámetro es la forma de la distribución. Este último parámetro es puesto a un valor de dos, tres o cuatro donde el número refleja la variación de la distribución.

2.2.5 Glosario de términos

Este glosario ha sido elaborado con el fin de facilitar la consulta al lector de términos o conceptos y abreviaturas que aparecen a lo largo del texto en diversas ocasiones.

Controlador PID.-

Acción de control proporcional-integral-derivativa, esta acción combinada reúne las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con esta acción combinada se obtiene mediante:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(\tau)d\tau + K_p T_d \frac{de(t)}{dt}$$

y su función de transferencia resulta:

$$C_{PID}(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

Ziegler Nichols.-

Es un método para el ajuste de los parámetros de un controlador PID.

IAE (Integral Absolute Error).-

Es la integral del valor absoluto del error. Se emplea cuando el valor del error es mucho menor que 1.

ITAE (Integral Time Absolute Error).-

Es la integral del valor absoluto del error ponderado en el tiempo. Se emplea cuando el valor del error es mucho menor que 1.

MSE (Mean Square Error).-

Es el error medio cuadrático para la optimización de los parámetros de un controlador PID.

CAPÍTULO III

METODOLOGÍA

3.1 Relación entre las variables de la investigación

Se realizó la descripción del sistema a linealizar para determinar sus ecuaciones de estado, lo cual nos permitió determinar cuáles son las variables dependientes e independientes. Las relaciones de las variables que se presentan en el sistema de control del sistema pelota y aro se observaron en los diagramas de cuerpo libre y diagramas de bloques donde se aplicaron las leyes de Lagrange-Euler para la obtención de las ecuaciones dinámicas.

3.2 Operacionalización de las variables

Para demostrar y comprobar la hipótesis, la operacionalizamos, obteniéndose las variables y los indicadores que a continuación se indican:

a.- Variable X = Sistema pelota y aro.

Indicadores:

-Velocidad del aro X1.

-Posición de la bola X2.

b.- Variable Y = Algoritmos genéticos.

Indicadores:

-Codificación Y1.

-Población Y2.

-Función de evaluación Y3.

-Función de selección Y4.

-Operadores de cruzamiento Y5.

c.- Variable Z = Optimización del controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos.

Indicadores:

-IAE (Integral Absolute Error) Z1.

-ITAE (Integral Time Absolute Error) Z2.

-MSE(Error Cuadrático Medio) Z3.

3.3 Tipo de Investigación

El presente trabajo, es una investigación científica experimental, por lo que se utilizó la teoría de control PID optimizado, basado en el estudio de los algoritmos genéticos, el que nos permitió evaluar la performance satisfaciendo las consideraciones del mínimo sobrepaso y error en estado estacionario nulo.

Temporal.

El estudio es de tipo longitudinal.

Espacial.

El estudio se realizó en los ambientes de la Sección de Posgrado de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional del Callao.

3.4 Diseño de la investigación

3.4.1 Modelado del sistema pelota y aro

El aro es gobernado por un motor DC, como resultado, demuestra las mismas características de un motor DC, es decir tiene una ganancia y un tiempo

de retardo. La función de transferencia general para el sistema aro es muestra en la Figura 3.1.

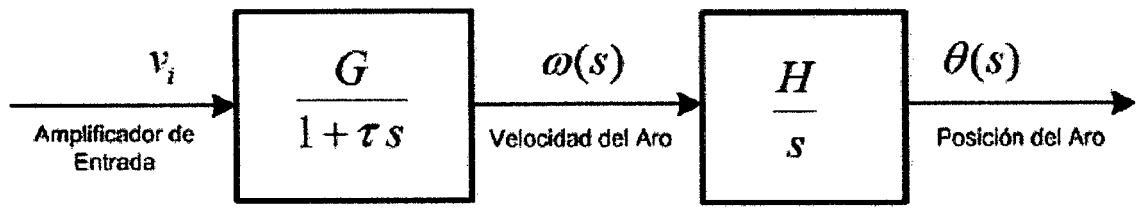


Figura 3.1: Función de transferencia relativo al voltaje de entrada y la posición del aro

La ganancia de estado constante, *G*, puede ser medida ploteando la velocidad ω constante del aro para un rango constante del voltaje de entrada del amplificador. Desde la ecuación dada en (2.22) podemos obtener la función de transferencia del sistema pelota y aro.

$$\frac{\omega(s)}{V_i(s)} = \frac{1}{I_a s + b_m}, \quad \frac{\theta(s)}{\omega(s)} = \frac{1}{s} \tag{3.1}$$

Los resultados de la simulación para este sistema en lazo abierto son mostrados en la Figura 3.2.

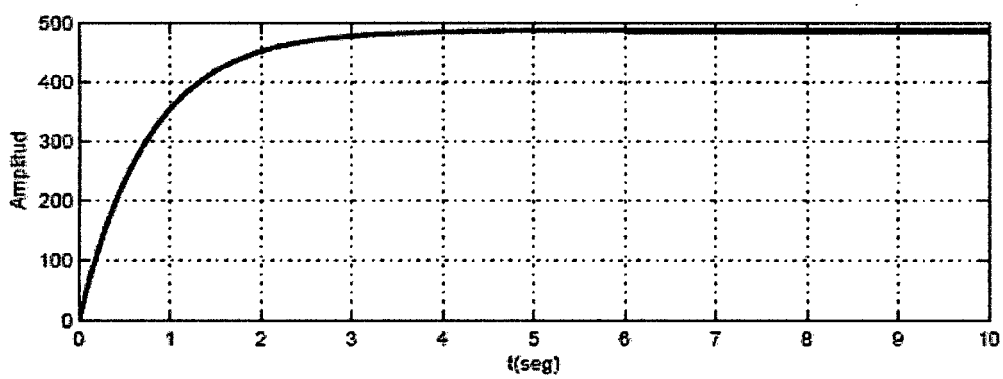


Figura 3.2: Cambio de la velocidad del motor en el tiempo

Si la ecuación (2.23) se expresa como una función de transferencia, esta

resulta:

$$\frac{Y(s)}{\Theta(s)} = R\frac{s^2 + \frac{g}{R}}{\left(\frac{2r_b^2}{5r^2} + 1\right)s^2 + \frac{b_b}{mr^2}s + \frac{g}{R}} \tag{3.2}$$

Entonces es claro que la función de transferencia relacionando el ángulo del aro $\theta(s)$, con la posición de la pelota en el aro $Y(s)$ tiene ceros puramente imaginarios en $s = \pm j\,g/R$ como se observa en la Figura 3.3.

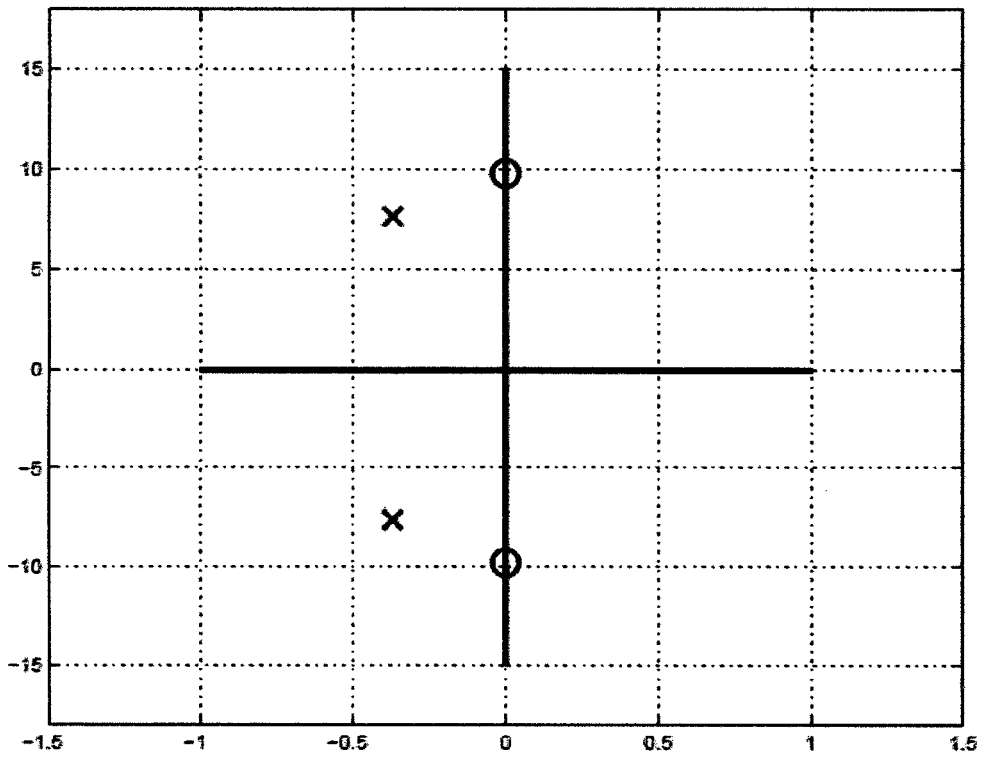


Figura 3.3: Polos y ceros del sistema

Estos son los llamados ceros de transmisión, lo que significa que cuando el ángulo del aro θ es una señal senoidal de frecuencia g/R rad/seg es aplicada a θ. Entonces genera una respuesta cero exacta en la posición de la pelota a la salida

$Y(s)$. En términos físicos esto corresponde al caso donde la pelota oscila dentro del aro exactamente a la misma frecuencia que el aro. Para un observador (parado) en el aro, la pelota no se mueve por lo que la pelota y el aro se mueven sincronizadamente. Por supuesto que la pelota se encuentra moviéndose cuando se observa desde fuera, pero si se observa desde un punto adecuado pareciera que la pelota es estacionaria.

Completando el sistema pelota y aro se puede afirmar que tanto la función de transferencia del sistema aro como el sistema pelota viene representados por el diagrama de bloques que se muestra en la Figura 3.4.

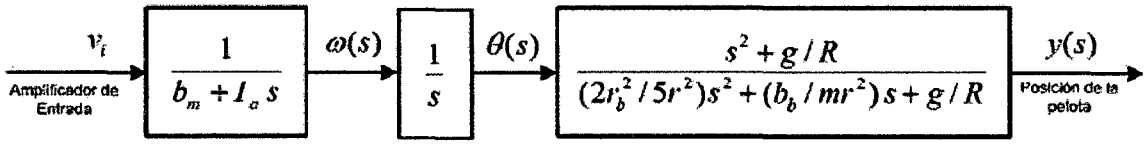


Figura 3.4: Función de transferencia del sistema pelota y aro

3.4.2 Algoritmo PID óptimo del sistema pelota y aro

La planta del proceso es dado por la función de transferencia siguiente:

$$P(s) = \frac{s^2 + 95.61}{0.001011s^4 + 0.001316s^3 + 0.1507s^2 + 0.196s}$$

Si discretizamos la planta mediante el retenedor de orden cero para un periodo de muestreo T = 0.1seg se obtiene:

$$P(z) = \frac{4.532z^3 - 0.7249z^2 - 0.3369z + 4.346}{z^4 - 2.564z^3 + 3.167z^2 - 2.481z + 0.878}$$

Asumiendo los parámetros, tiempo de simulación *tfinal* = 20, y = 0.01,

Δq = 0.001, las señal deseada es un escalón unitario del tamaño de *tk* (eje de tiempo digital), las condiciones iniciales de q_i = 0 para i = 1, 2, 3 y un máximo de 1000 iteraciones.

- **Algoritmo de Control en Matlab**

El algoritmo se implementa con Matlab desde las líneas de código siguientes:

```
q0=0; q1=0; q2=0;
maxiter=1000;
J=zeros([1 maxiter]);
for i=1:maxiter,
d=tf([q0+dq q1 q2],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J1=norm(des-s);
d=tf([q0 q1+dq q2],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J2=norm(des-s);
d=tf([q0 q1 q2+dq],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J3=norm(des-s);
d=tf([q0 q1 q2],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J(i)=norm(des-s)
```

```
if (mod(i-10,10)==0),
J(i)
end
p1=(J1-J(i))/dq;
p2=(J2-J(i))/dq;
p3=(J3-J(i))/dq;
Delta=sqrt(p1*p1+p2*p2+p3*p3);
q0=q0-(gamma/Delta)*p1;
q1=q1-(gamma/Delta)*p2;
q2=q2-(gamma/Delta)*p3;
end;
```

De este resultado se obtiene la función de transferencia del controlador PID digital dado por:

$$C(z) = \frac{0.09107z^2 - 0.1564z + 0.06954}{z^2 - z}$$

3.4.3 Algoritmo genético para evaluar el mejor controlador PID para el sistema pelota y aro

- **Función objetivo para el algoritmo genético**

Escribir una función objetivo es la parte más difícil de crear un algoritmo genético. En este proyecto, la función objetivo es requerida para evaluar el mejor controlador PID para el sistema pelota y aro. Una función objetivo podría ser creada para encontrar un controlador PID que da el más pequeño sobrepaso, y

que tenga un tiempo de subida rápido o un tiempo de establecimiento más rápido pero para combinar todos estos objetivos se decidió diseñar una función objetivo que minimice el error del sistema controlado.

Cada cromosoma en la población es pasado a la función objetivo uno a uno. El cromosoma entonces es evaluado y es asignado a un número, el número más grande será el de mejor ajuste. El algoritmo genético utiliza valor de ajuste del cromosoma para crear una nueva población que consiste en los miembros más convenientes.

Las siguientes líneas de código Matlab nos muestran tales evaluaciones:

```
function [x_pop,fx_val] = PID_objfun_MSE(x_pop)
global PID
global t
global planta
Kp=x_pop(2);
Ki=x_pop(3);
Kd=x_pop(1);
% Creando el controlador PID para valores actuales
pid_num=[Kd Kp Ki];
pid_den=[1 0];
pid_sys=tf(pid_num,pid_den);
```

Cada cromosoma consiste en tres cadenas separadas que constituyen los términos proporcional. (P), integral (I) y el término derivativo (D), definido por 3

filas que se declaran al crear la población. Cuando el cromosoma es evaluado, es separado en sus tres términos, entonces se crea el un controlador de PID según:

$$PID(s) = \frac{K_d s^2 + K_p s + K_i}{s}$$

El controlador PID nuevamente formado es colocado en un lazo de realimentación unitario en serie con la función de transferencia del sistema pelota y aro. Para reducir el tiempo de compilación del programa la función de transferencia del sistema PYA son definidos en otro archivo e importados como una variable global. El sistema controlado es dado para una entrada escalón unitario y el error es valorado utilizando un criterio apropiado de desempeño de error, es decir, ITAE (Integral of Timeweighted Absolute value of Error), ISE (Integral of Absolute value of Error), IAE (Integral of Absolute value of Error) o MSE (Mean Square Error). El cromosoma es asignado un valor general de ajuste de performance según la magnitud del error.

```
t =0:0.01:0.1;
y = step(PID,t);
for i=1:length(y)
error(i) = 1-y(i);
end
```

El código adicional fue agregado para asegurar que el algoritmo genético converge a un controlador que produce un sistema fijo. Las líneas de código Matlab que se lista debajo, valora los polos del sistema controlado y si son

encontrados inestable (polos en el semiplano derecho del plano-s), el error es asignado un valor grande para asegurarse de que el cromosoma no sea seleccionado de nuevo.

```
polos=pole(PID);
if poles(1)>0
MSE=10e300;
elseif polos(2)>0
MSE=10e300;
elseif polos(3)>0
MSE=10e300;
elseif polos(4)>0
MSE=10e300;
elseif polos(5)>0
MSE=10e300;
end
```

Para evaluar un algoritmo genético óptimo para el uso en este proyecto, varios algoritmos genéticos fueron creados y analizados como se mostrará en la próxima sección.

- **Desarrollo de Algoritmo Genético para la Sintonía del Controlador PID**

Para ayudar con el desarrollo de este proyecto de tesis en relación al control del sistema PYA podemos desarrollar la sintonía de las familias para el diseño del controlador PID que es diseñado utilizando los métodos convencionales. Un algoritmo genético fue creado para evaluar los coeficientes de PID del mismo sistema y los resultados de las dos técnicas fueron comparados.

Las siguientes líneas de código muestran la evolución del controlador PID convencional en Matlab. Se generan varias respuestas dentro de un rango convenido para cada parámetro de ganancias.

```
Kp=[0.1 0.01 0.001 0.002];
Ti=[100 500 1000 1500];
Td=[0.1 0.2 0.3 0.4];
for i=1:4 % Kp
for j=1:4 % Ti
for k=1:4 % Td
PID=tf(Kp(i)*[Td(k)*Ti(j) Ti(j) 1],[Ti(j) 0]);
S=series(PID,P);
F=S/(S+1);
t=0:0.1:15;
hold on
y=step(F,t);
plot(t,y,'k')
xlabel('t(seg)')
```

```
ylabel('Amplitud')
grid on
end
end
end
```

- **Índices de Optimización**

Varias funciones objetivos fueron escritas basadas en el criterio del desempeño de error. Cada función objetivo es fundamental ya que define el criterio específico de desempeño de error para ser aplicado. Para optimizar el desempeño de un sistema de control PID, las ganancias de PID del sistema son ajustadas para llevar al máximo o para minimizar un cierto índice de desempeño. El índice del desempeño es calculado sobre un intervalo de tiempo T, normalmente en la región de $0 \leq T \leq t_s$ donde t_s es el tiempo de establecimiento del sistema.

- **Integral de Tiempo multiplicado por Error Absoluto (ITAE)**

Se emplea cuando la magnitud del error es $\ll 1$ pero además perdura en el tiempo de forma que éste trabaja como un factor de peso variable.

$$I_{ITAE} = \int_0^T t|e(t)|dt$$

Las líneas de código Matlab nos ilustran el procedimiento para llevar a cabo el índice de desempeño ITAE.

```
t =0:0.01:0.1;
u=ones(size(t));
y =lsim(PID,u,t);
for i=1:length(y)
error(i) = (abs(1-y(i)))*t(i);
end
ITAE=sum(error);
fx_val=1/ITAE;
```

- **Integral de Magnitud de Error Absoluto (IAE)**

Se emplea con buenos resultados aún cuando la magnitud del error es ≪ 1.

$$I_{IAE} = \int_0^T |e(t)|dt$$

Las líneas de código Matlab nos ilustran el procedimiento para llevar a cabo el índice de desempeño IAE.

```
for i=1:length(y)
error(i) = 1-y(i);
end
IAE=sum(abs(error));
fx_val=1/IAE;
```

- **Integral Cuadrático de Error(ISE)**

Se emplea siempre que la magnitud del error no sea $\ll 1$.

$$I_{ISE} = \int_0^T e^2(t)dt$$

Las líneas de código Matlab nos ilustran el procedimiento para llevar a cabo el índice de desempeño ISE.

```
for i=1:length(y)
error(i) = 1-y(i);
end
ISE=error*error';
fx_val=1/ISE;
```

- **Error Cuadrático Medio(MSE)**

El MSE refleja todas las variaciones y desviaciones del valor deseado.

$$I_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(e(t))^2$$

Las líneas de código Matlab nos ilustran el procedimiento para llevar a cabo el índice de desempeño MSE.

```
for i=1:length(y)
error(i) = 1-y(i);
end
```

```
error_sq=error*error';
MSE=error_sq/max(size(error));
```

Un experimento fue emprendido para evaluar cuál de los cuatro criterios de desempeño produce los mejores resultados cuando se usa en conjunción con un algoritmo genético. Una función objetivo fue creada para cada criterio individual de desempeño que represente al control PID, como son: PID_objfun_ITAE.m, PID_objfun_IAE.m, PID_objfun_ISE.m, y PID_obfun_MSE.m que están listadas en el Anexo B.

La función para el algoritmo genético, Initial_PID_GA.m, fue utilizada para cada función objetivo. El algoritmo genético fue inicializado con una población de veinte y fue iterado para 100 generaciones. El número total de mutaciones fue puesto a tres y cada limite estaría comprendido entre ±50. Para asegurar que todos los algoritmos genéticos tuvieron la mismo condición inicial de tal forma que el algoritmo inicializa al mismo conjunto de valores.

3.5 Etapas de la Investigación

Las etapas de la investigación se concretan en los siguientes aspectos:

a.- Descripción experimento
b.- Hipótesis de investigación
c.- Variables
d.- Universo-muestra
e.- Instrumento de medición
f.- Diseño de la investigación
g.- Prueba piloto
h.- Limitaciones y alcances

3.6 Población y muestra

Debido a la naturaleza de esta investigación científica experimental no amerita determinar el tamaño de la muestra.

3.7 Técnicas e Instrumentos de recolección de datos

Debido a que no se ha determinado el tamaño de la muestra, no corresponde utilizar las técnicas e instrumentos para recolección de datos.

3.8 Procesamiento estadístico y análisis de datos

No corresponde utilizar procesamiento y análisis de datos.

CAPÍTULO IV

RESULTADOS

4.1 Resultados parciales

4.1.1 Resultados del Sistema de Control PID Digital Óptimo del sistema pelota y aro

Los resultados de la simulación del sistema, mediante el programa Matlab, se muestra en la Figura 4.1. Los coeficientes de optimización son q_1 = 0.0911, q_2=−0.1564 y q_3 = 0.0695.

La curva de performance, en donde se presenta la evolución del error es mostrada en la Figura 4.2. Podemos observar que han transcurrido 100 iteraciones y simultáneamente va midiendo la respuesta de salida, este algoritmo requiere de una precisión en los parámetros de sintonía para obtener las mejores respuestas, sin embargo se presenta como una alternativa para encontrar los coeficientes del controlador PID Digital, de ese modo podemos evaluar directamente en las simulaciones.

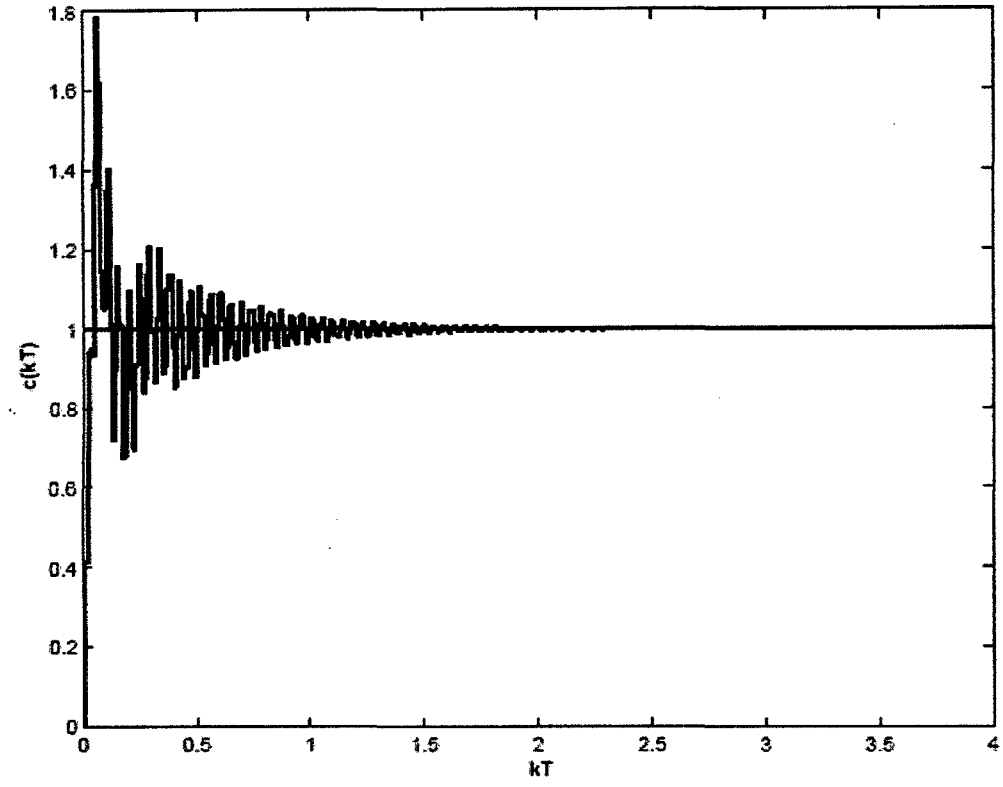


Figura 4.1: Respuesta del sistema de control debido a una entrada escalón unitario

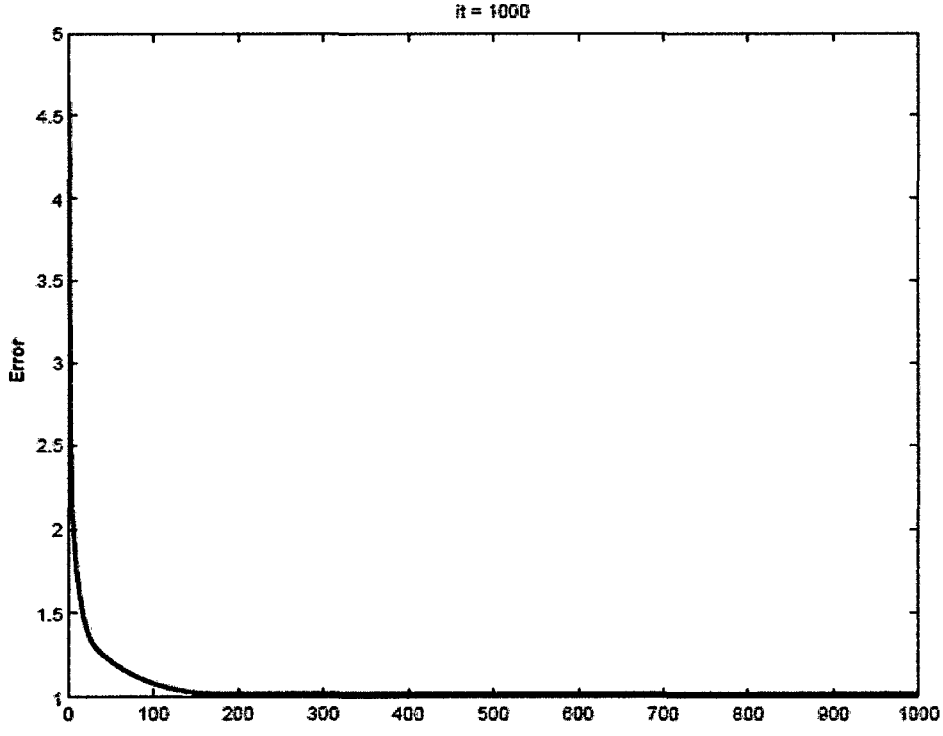


Figura 4.2: Curva de evolución del error

4.1.2 Resultados del Sistema de Control PID del sistema pelota y aro, sintonizado con algoritmos genéticos

La evolución de las respuestas del sistema de control es mostrado en la Figura 4.3, podemos observar diferentes tipos de respuestas para variadas consideraciones de diseño, en relación a sobrepaso, tiempo de subida y tiempo de establecimiento.

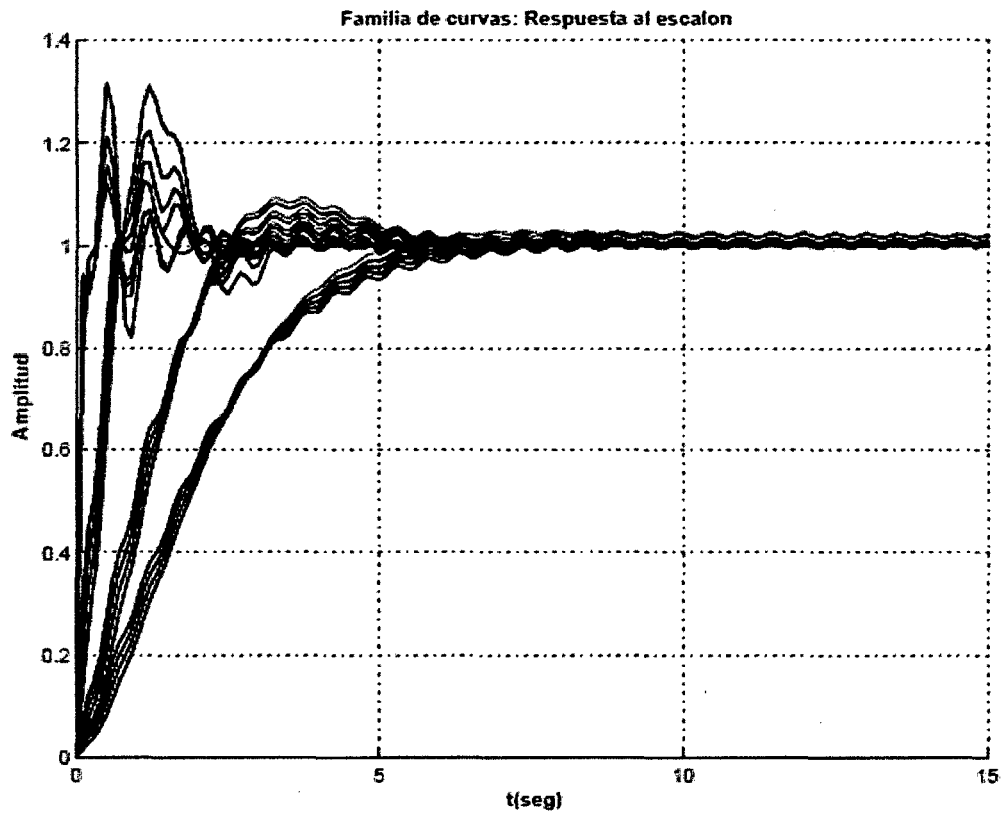


Figura 4.3: Evolución de las ganancias de las familias PID convencional

Escogiendo una de las repuestas que cumplan con las consideraciones de diseño, por ejemplo que no presente sobrepaso máximo, con mejores tiempos de respuestas se observa en la Figura 4.4. Podemos observar también que tal respuesta presenta alguna oscilación auto sostenida y tiempo de establecimiento

alto, que de alguna forma no es recomendable para el control del sistema pelota y aro que requiere de respuestas rápidas.

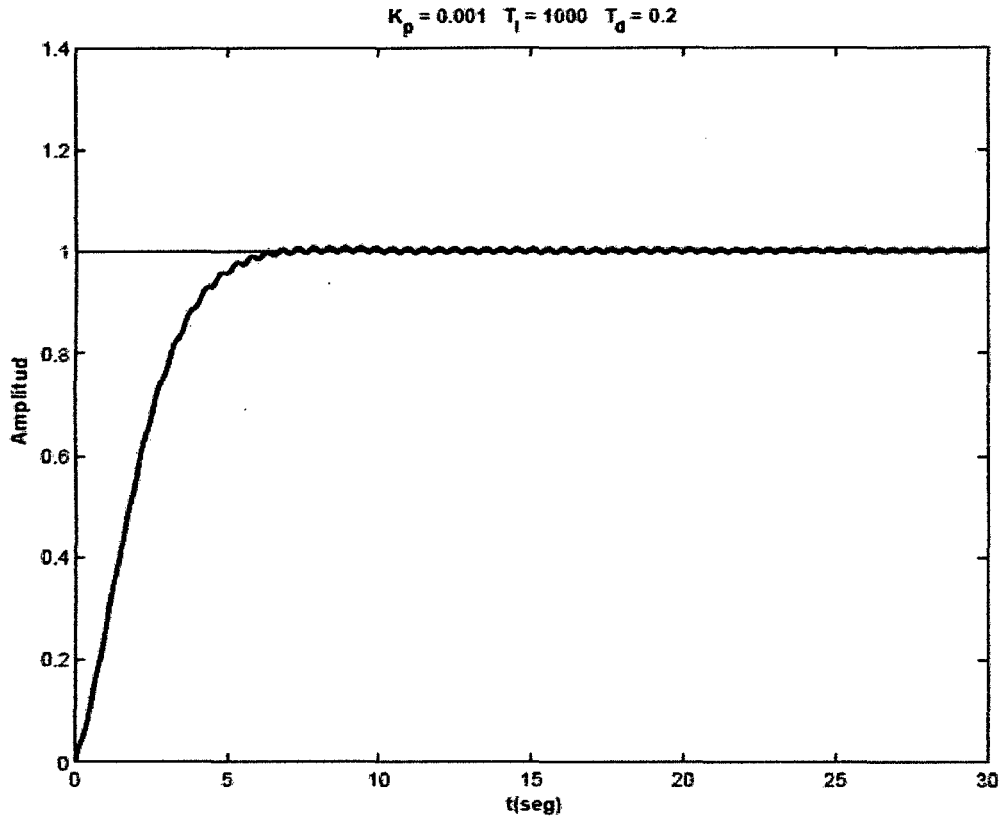


Figura 4.4: Evolución de la mejor performance del controlador PID convencional

4.2 Resultados finales

A continuación se presentan los resultados de las simulaciones de las medidas de performance del sistema de control PID basado en algoritmos genéticos y de un sistema de control PID convencional.

Las medidas de performance sedan por los resultados que se obtengan de las simulaciones del sistema basado en algoritmos genéticos. El código Matlab de medias de índices de performance se lista debajo:

```
t=0:0.0001:0.5;
u1=[ones(1,1250) zeros(1,1250)];
u2=0.5*[ones(1,1250) zeros(1,1251)];
u=[u1 u2];
y=lsim(PID,u,t);
plot(t,y,'r')
hold
plot(t,u,'b')
```

Se miden varios índices de performance para la respuesta del algoritmo genético en comparación con un algoritmo PID convencional, como lo demuestran las simulaciones de las Figuras 4.5–4.9.

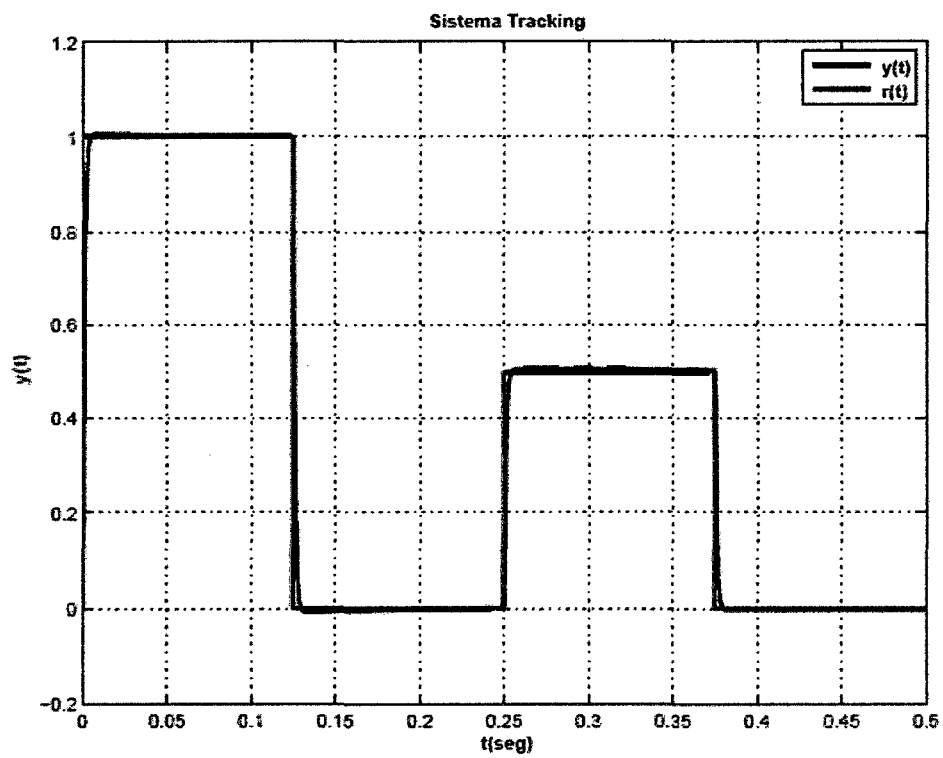


Figura 4.5: Sistema tracking para el índice de performance ITAE

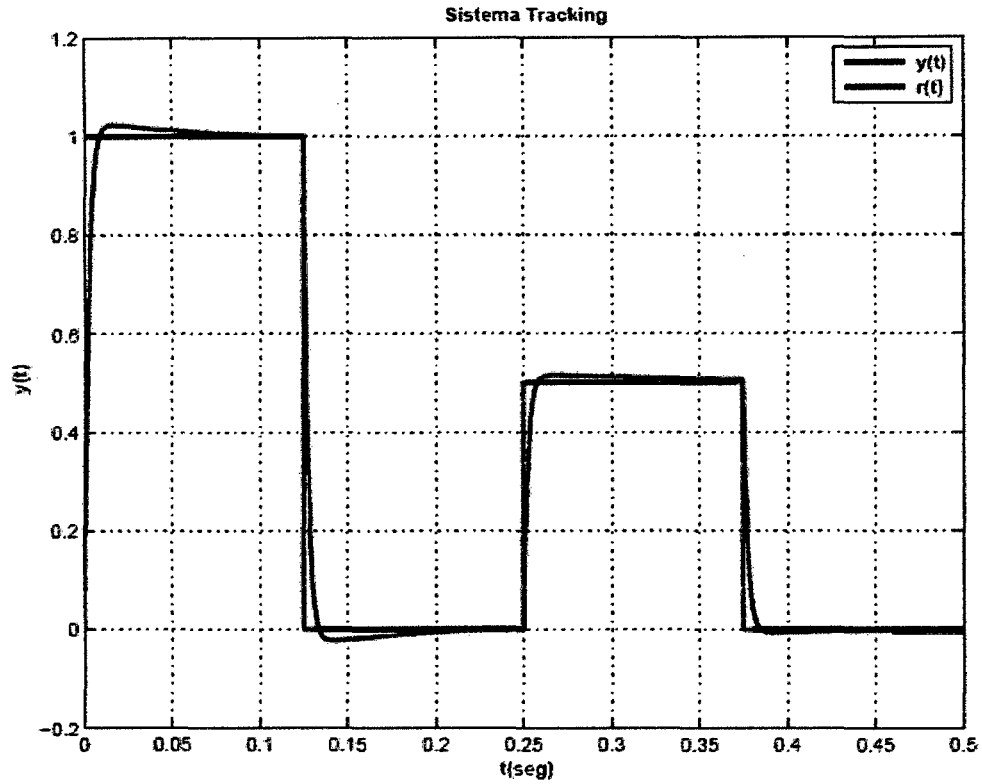


Figura 4.6: Sistema tracking para el índice de performance IAE

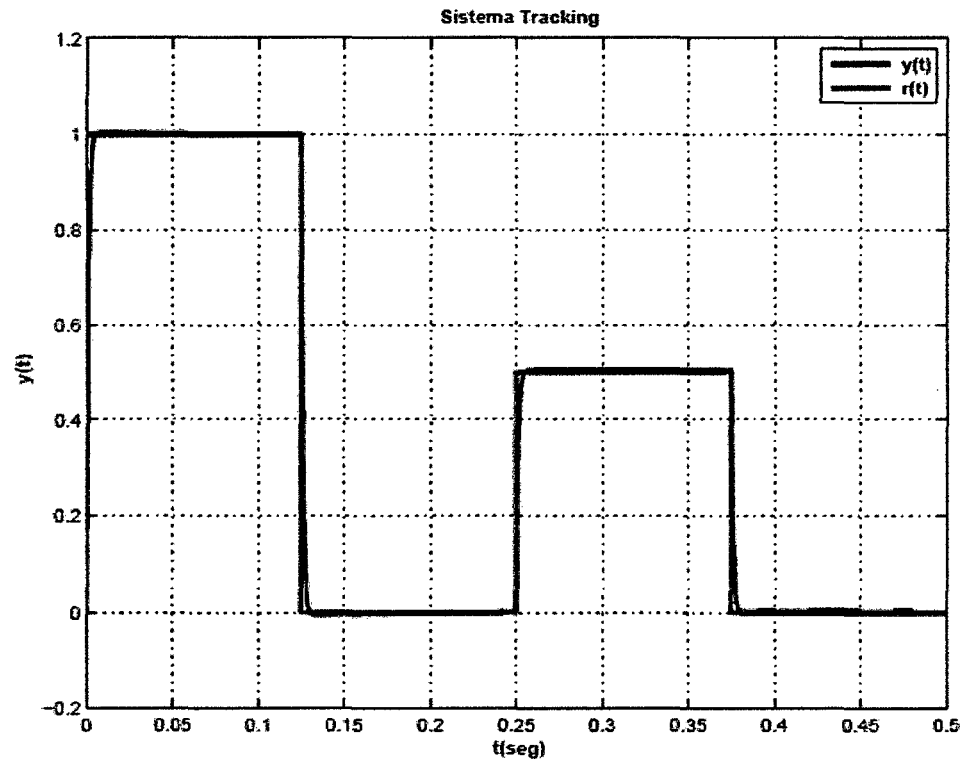


Figura 4.7: Sistema tracking para el índice de performance ISE

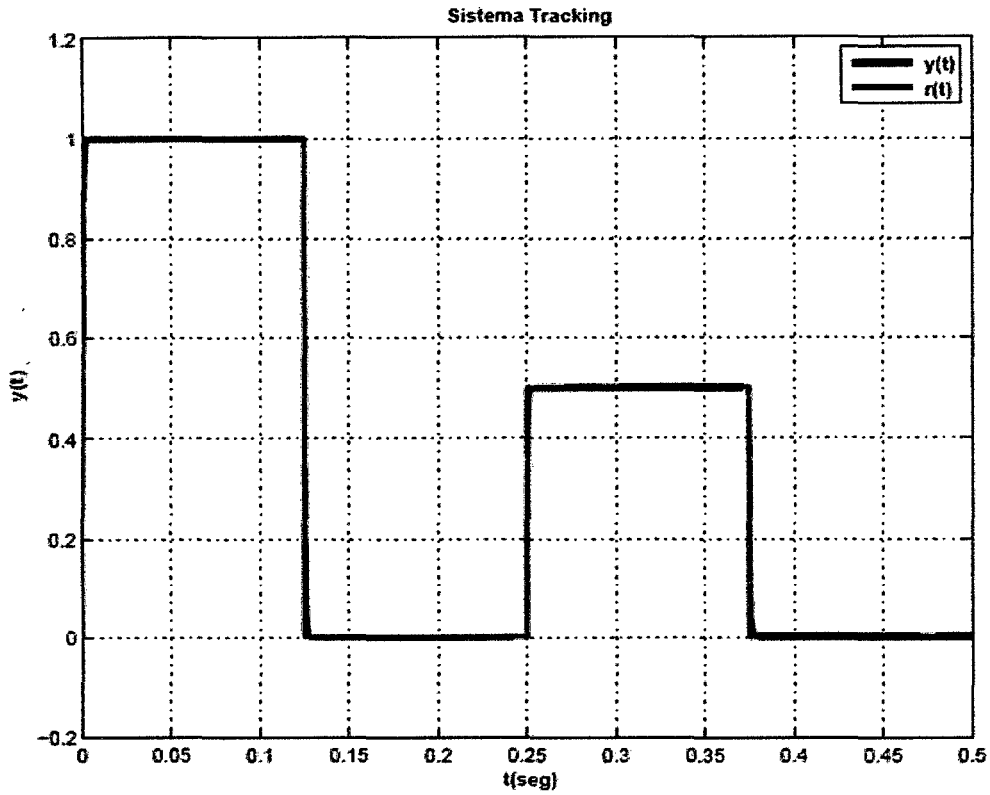


Figura 4.8: Sistema tracking para el índice de performance MSE

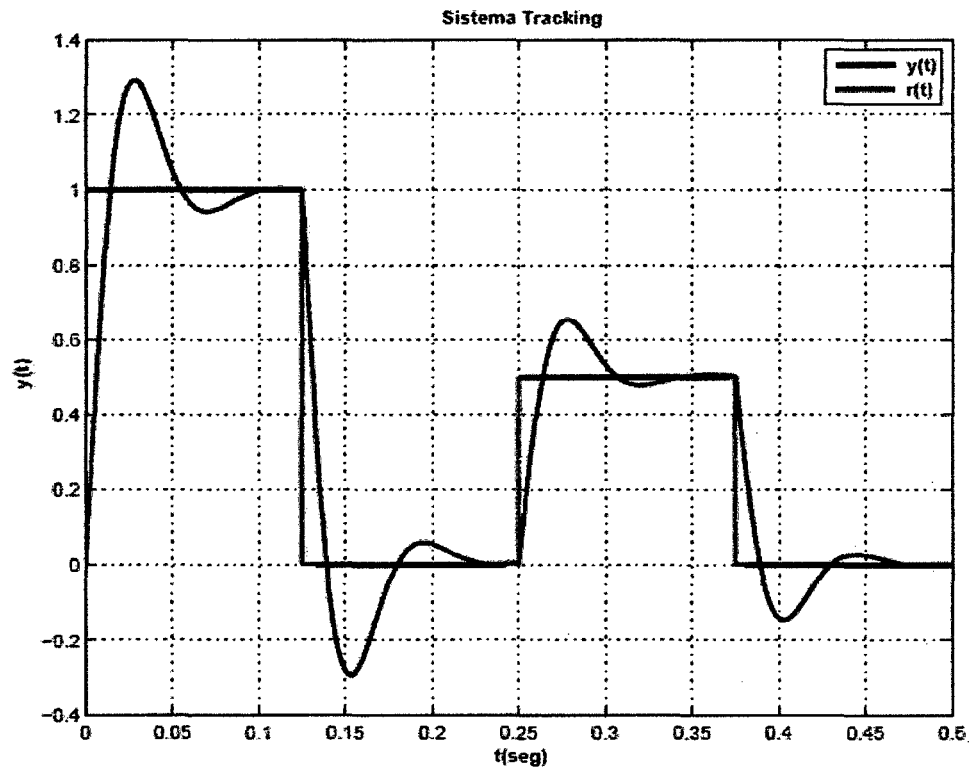


Figura 4.9: Sistema tracking para un PID convencional

CAPÍTULO V

DISCUSIÓN DE RESULTADOS

5.1. Contrastación de hipótesis con los resultados

De las simulaciones presentadas en la sección 4.2, se observa que los índices de performance ISE y MSE son funciones objetivos que trabajan casi idénticamente, teniendo un tiempo más pequeño de subida, esto significa que pueden mejorar el tiempo de establecimiento en comparación con otros controladores. Cada uno de los controladores genéticos PID supera a los métodos de diseño convencional, como son el de PID convencional de Ziegler Nichols. La función MSE fue escogida como el criterio primario de desempeño para el resto de este proyecto debido a su tiempo más pequeño de subida y más pequeño sobrepaso que cualquier otro método.

Esto es asociado con el hecho que MSE ha sido una medida demostrada de control y calidad durante muchos años lo que hace el criterio ideal de desempeño para este proyecto de tesis.

En tal sentido, se ha cumplido con la hipótesis planteada en el trabajo, el cual paso a anotar:

"Es posible optimizar los parámetros del controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos".

5.2. Contrastación de resultados con otros estudios similares

Al comparar los resultados obtenidos en el presente trabajo con respecto a los trabajos de Ian Griffin, de la Universidad DCU de Irlanda, "On-line PID Controller Tuning using Genetic Algorithms" y de S. Kanthalaskshmi, del Comba PSG College of Technology de la India, "Genetic Algorithm Based Self Tuning", mediante los algoritmos genéticos, la respuesta del lazo cerrado del sistema pelota y aro es óptima, cuando los algoritmos genéticos autosintonizan los parámetros del controlador PID.

CONCLUSIONES

- El buen diseño y la implementación de un algoritmo de control, implica conocer ciertas características del modelo del proceso, el mismo que será utilizado en el diseño del control. También se pudo observar que el uso de las ganancias PID involucra tener presente el aporte que genera cada una de esas ganancias, así como el efecto de la variación de las mismas. Además fue necesario conocer dentro de que rangos se encontraban dichas ganancias, lo cual no hubiese sido posible sin valernos de un método complementario que nos diera el punto de arranque, en este caso nos referimos a los resultados obtenidos usando la sintonía de controladores PID utilizando métodos convencionales.
- Como se vio en las simulaciones, nos permitieron conocer el rango óptimo de cada una de las ganancias PID, mientras que se observaba buenos resultados aplicando la normalización de selección geométrica. Por otra parte, los algoritmos genéticos dieron resultados bastante aceptables durante la simulación de perfiles de movimiento, como se mostró en las simulaciones, los cuales se usan frecuentemente en el control de posición de motores DC, lo que nos lleva a concluir que el algoritmo es muy eficiente para otro tipo de entradas que no sean escalón.

RECOMENDACIONES

- A partir de los resultados obtenidos con el control propuesto se recomienda implementar el sistema de control, con la garantía que el sistema pueda ser gobernado experimentalmente sin problemas.
- Se puede sugerir que para trabajos futuros de sistemas donde los parámetros cambian constantemente se utilicen algoritmos genéticos para la optimización de los controladores PID aplicados a estos sistemas.

REFERENCIALES

[1] C. Phillips, H. Nagle. ***"Control System Analysis and Design"***. Prentice Hall, 1984.

[2] Gene Franklin, J. Powell, A. Emami-Naeini. ***"Feedback Control of Dynamics System"***. Addison Wesley, 3era Ed, 1994.

[3] Goldberg, D. E. ***"Genetic Algorithms in Search, Optimization and Machine Learning"***. Addison-Wesley Longman Publishing Co.,Inc., Boston, MA, USA.

[4] Houck, C. R. and Kay, M. ***"A genetic algorithm for function optimization: A Matlab implementation"***. ACM Transactions on Mathematical Software, 1996.

[5] Wellstead, P. E., ***"The Ball and Hoop System"***, Automatica, Vol. 19, N°. 4, pp. 401-406, 1983

[6] Whitley, ***"A Genetic Algorithm Tutorial, Technical Report"***, CS-93-103, Dept. of Computer Science, Colorado State University, 1993

ANEXO A

Matriz de Consistencia

Título: "ALGORITMOS GENÉTICOS PARA LA OPTIMIZACIÓN DEL CONTROLADOR PID APLICADO AL SISTEMA PELOTA Y ARO"

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLES	MÉTODOS
1.Problema general : Controlar la velocidad del aro y la posición de la bola del sistema pelota y aro. **2.Problema específico:** Utilizar los algoritmos genéticos para optimizar el controlador PID aplicado al sistema pelota y aro.	**1.Objetivo general:** Optimizar un controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos. **2.Objetivos específicos:** 2.1 Realizar el modelamiento matemático del sistema pelota y aro. 2.2 Especificar requerimientos de diseño para el buen control. 2.3 Usar técnicas de optimización que lleven a un buen desempeño del control. 2.4 Diseñar el controlador y validar su desempeño por medio de la simulación.	Es posible optimizar los parámetros del controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos.	Las variables que van a ser relacionadas en la investigación son las siguientes: **Variable X** = Sistema pelota y aro. **Variable Y** = Algoritmos genéricos. **Variable Z** = Optimización del controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos.	La optimización del controlador PID aplicado al sistema pelota y aro, mediante los algoritmos genéticos, requiere los siguientes pasos: a.- Determinar las ecuaciones dinámicas del sistema aplicando Lagrange-Euler. b.- Realizar el modelamiento en variables espacio estado. c.- Analizar el modelo resultante del sistema en lazo abierto. d.- Diseño del algoritmo de control PID digital óptimo. e.- Diseño del algoritmo de optimización de parámetros mediante los algoritmos genéticos. f.- Simular los resultados probando su desempeño respecto a referencias arbitrarias. g.- Observaciones y conclusiones.

ANEXO B

PROGRAMAS EN MATLAB

B.1. Programa: Controlador PID Basado en Algoritmo Genético

Los programas de simulación basadas en algoritmos genéticos están escritos en código MATLAB. Se presentan los programas que fueron utilizados en las simulaciones realizadas en el capítulo 4.

- **Initial_PID_GA.m**
- **ga.m**
- **PID_objfun_IAE.m**
- **PID_objfun_ITAE.m**
- **PID_objfun_ISE.m**
- **PID_objfun_MSE.m**
- **arithXover.m**
- **initializega.m**
- **maxGenTerm.m**
- **normGeomSelect.m**

B.1.1. Programa Principal

```
%-----------------------------------------------------------------
global PID
global t
global planta
% -----------------------------------------------------------------
% Parametros del Sistema Pelota y Aro
% -----------------------------------------------------------------
bm=2.05e-3;bb=1.92e-6;R=0.1025;r=7.5e-3;rb=9.5e-3;M=0.3;m=28.1e-3;g=9.8;
```

```
Ia=(1/2)*M*R^2;Ib=(2/5)*m*rb^2;
% ------------------------------------------------------------------
% Planta
% ------------------------------------------------------------------
Gp=tf(1,conv([1 0],[Ia bm]));
Gy=tf([1 0 g/R],[2*rb^2/(5*r^2) bb/m*r^2 g/R]);
P=Gp*Gy;
planta=zpk(P);
% Inicializando el algoritmo Genetico
populationSize=100;
variableBounds=[0 10;0 10;0 10];
%evalFN='PID_objfun_MSE';
%evalFN='PID_objfun_ITAE';
%evalFN='PID_objfun_IAE';
evalFN='PID_objfun_ISE';
evalOps=[];
options=[1e-6 1];
initPop=initializega(populationSize,variableBounds,evalFN,evalOps,options);
% Seteando los parametros para el GA
bounds=[-100 100;-100 100;-100 100];
%evalFN='PID_objfun_MSE'; % Cambiar por una funcion relevante
evalFN='PID_objfun_ISE';
evalOps=[];
startPop=initPop;
opts=[1e-6 1 0];
```

```
termFN='maxGenTerm';
termOps=100;
selectFN='normGeomSelect';
selectOps=0.05;
xOverFNs='arithXover';
xOverOps=4;
mutFNs='unifMutation';
mutOps=8;
% Iteractua con el GA
[x,endPop,bPop,traceInfo]=ga(bounds,evalFN,evalOps,startPop,opts,termFN,...
termOps,selectFN,selectOps,xOverFNs,xOverOps,mutFNs,mutOps);
% Creando el controlador optimo PID desde GA
PID_GA=tf([x(1) x(2) x(3)],[1 0]);
S=PID_GA*planta;
PID=S/(S+1);
figure
subplot(2,3,[4 6])
t=0:0.0001:0.025;
u=ones(size(t));
y=lsim(PID,u,t);
plot(t,y,'r','LineWidth',2)
hold
plot(t,u,'b','LineWidth',2)
axis([0 0.025 0 1.2])
grid
```

```
ylabel('\bf y(t)');
xlabel('\bf t(seg)');
legend('\bf y(t)','\bf r(t)',4)
% ------------------------------------------------------------------
% Ploteando el Progreso de la mejor poblacion
% x(1)=Kd, x(2)=Kp, x(3)= Ki
% ------------------------------------------------------------------
subplot(2,3,1)
plot(bPop(:,3),'k','LineWidth',2)
title(['\bf K_p = ', num2str(x(2))])
ylabel('\bf Ganancia');
grid
subplot(2,3,2)
plot(bPop(:,3),'k','LineWidth',2)
title(['\bf K_i = ', num2str(x(3))]);
ylabel('\bf Ganancia');
grid
subplot(2,3,3)
plot(bPop(:,2),'k','LineWidth',2)
title(['\bf K_d = ', num2str(x(1))])
ylabel('\bf Ganancia');
%xlabel('\bf Generaciones')
grid
% ------------------------------------------------------------------
% Imprime Ganancias
```

```
% ------------------------------------------------------------------
disp('Gains Kp Ki Kd')
disp([x(2) x(3) x(1)])
% ------------------------------------------------------------------
figure
t=0:0.0001:0.5;
u1=[ones(1,1250) zeros(1,1250)];
u2=0.5*[ones(1,1250) zeros(1,1251)];
u=[u1 u2];
y=lsim(PID,u,t);
plot(t,y,'r','LineWidth',2)
hold
plot(t,u,'b','LineWidth',2)
%axis([0 0.025 0 1.2])
grid
ylabel('\bf y(t)');
xlabel('\bf t(seg)');
legend('\bf y(t)','\bf r(t)',1)
title('\bf Sistema Tracking')
%-----------------------------------------------------------------
```

B.1.2. Funciones Principales de Optimización

Función ga

```
%---------------------------------------------------------------
function [x,endPop,bPop,traceInfo] = ga(bounds,evalFN,evalOps,startPop,opts,...
termFN,termOps,selectFN,selectOps,xOverFNs,xOverOps,mutFNs,mutOps)
n=nargin;
if n<2 || n==6 || n==10 || n==12
disp('Argumentos Insuficientes')
end
if n<3 % Default evalua opts.
evalOps=[];
end
if n<5
opts = [1e-6 1 0];
end
if isempty(opts)
opts = [1e-6 1 0];
end
if any(evalFN<48)
if opts(2)==1
e1str=['x=c1; c1(xZomeLength)=', evalFN ';'];
e2str=['x=c2; c2(xZomeLength)=', evalFN ';'];
else
e1str=['x=b2f(endPop(j,:),bounds,bits); endPop(j,xZomeLength)=',evalFN ';'];
end
```

```
else
if opts(2)==1
e1str=['[c1(xZomeLength) c1]=' evalFN '(c1,[gen evalOps]);'];
e2str=['[c2(xZomeLength) c2]=' evalFN '(c2,[gen evalOps]);'];
else % Binario ga
e1str=['x=b2f(endPop(j,:),bounds,bits);[v x]=' evalFN ...
'(x,[gen evalOps]); endPop(j,:)=[f2b(x,bounds,bits) v];'];
end
end
if n<6
termOps=[100];
termFN='maxGenTerm';
end
if n<12 % mutacion
if opts(2)==1 % Float ga
mutFNs=['boundaryMutation multiNonUnifMutation nonUnifMutation unifMutation'];
mutOps=[4 0 0;6 termOps(1) 3;4 termOps(1) 3;4 0 0];
else
mutFNs=['binaryMutation'];
mutOps=[0.05];
end
end
if n<10 % Cruce
if opts(2)==1 % Float ga
xOverFNs=['arithXover heuristicXover simpleXover'];
```

```
xOverOps=[2 0;2 3;2 0];
else % Binario ga
xOverFNs=['simpleXover'];
xOverOps=[0.6];
end
end
if n<9 % rueda de ruleta
selectOps=[];
end
if n<8 % Default selecciona
selectFN=['normGeomSelect'];
selectOps=[0.08];
end
if n<6 % Default termina informacion
termOps=[100];
termFN='maxGenTerm';
end
if n<4 % No comienza poblacion
startPop=[];
end
if isempty(startPop)
startPop=initialize(80,bounds,evalFN,evalOps,opts(1:2));
end
if opts(2)==0 % binario
bits=calcbits(bounds,opts(1));
```

```
end
xZomeLength = size(startPop,2); % Longitud de xzome=numVars+fittness
numVar = xZomeLength-1; % Numero de variables
popSize = size(startPop,1); % Numero de individuos en la poblacion
endPop = zeros(popSize,xZomeLength); % Una matriz poblacion secundaria
c1 = zeros(1,xZomeLength); % Un individuo
c2 = zeros(1,xZomeLength); % Un individuo
numXOvers = size(xOverFNs,1); % Numero de operadores de cruce
numMuts = size(mutFNs,1); % Numero de operadores de mutacion
epsilon = opts(1); % Threshold para dos fittness diferentes
oval = max(startPop(:,xZomeLength));% Mejor valor para inio pob
bFoundIn = 1; % Numero de tiempos para mejor cambio
done = 0; % No simula evolucion
gen = 1; % Numero de generacion actual
collectTrace = (nargout>3); % Informacion de gen
floatGA = opts(2)==1; % Aplicacion de probabilidad ops
display = opts(3); % Visualiza progreso
while(~done)
% Modelo Elitista
[bval,bindx] = max(startPop(:,xZomeLength));
best = startPop(bindx,:);
if collectTrace
traceInfo(gen,1)=gen; % Generacion actual
traceInfo(gen,2)=startPop(bindx,xZomeLength); % Mejor fittness
traceInfo(gen,3)=mean(startPop(:,xZomeLength)); % media fittness
```

```
end
if ( (abs(bval - oval)>epsilon) || (gen==1)) % Si tenemos mejor sol
if display
fprintf(1,'\n%d %f\n',gen,bval); % Actualiza el display
end
if floatGA
bPop(bFoundIn,:)=[gen startPop(bindx,:)]; % Actualiza matriz bPop
else
bPop(bFoundIn,:)=[gen b2f(startPop(bindx,1:numVar),bounds,bits)...
startPop(bindx,xZomeLength)];
end
bFoundIn=bFoundIn+1; % Actualiza numero de cambios
oval=bval; % Actualiza mejor valor
else
if display
fprintf(1,'%d ',gen);
end
end
endPop = feval(selectFN,startPop,[gen selectOps]); % Selecciona
if floatGA % Corre con el modelo de parametros que son numeros ops
for i=1:numXOvers,
for j=1:xOverOps(i,1),
a = round(rand*(popSize-1)+1);
b = round(rand*(popSize-1)+1);
xN=deblank(xOverFNs(i,:)); % pone nombre a funcion cruce
```

```
[c1 c2] = feval(xN,endPop(a,:),endPop(b,:),bounds,[gen xOverOps(i,:)]);
if c1(1:numVar)==endPop(a,(1:numVar))
c1(xZomeLength)=endPop(a,xZomeLength);
elseif c1(1:numVar)==endPop(b,(1:numVar))
c1(xZomeLength)=endPop(b,xZomeLength);
else
end
if c2(1:numVar)==endPop(a,(1:numVar))
c2(xZomeLength)=endPop(a,xZomeLength);
elseif c2(1:numVar)==endPop(b,(1:numVar))
c2(xZomeLength)=endPop(b,xZomeLength);
else
end
endPop(a,:)=c1;
endPop(b,:)=c2;
end
end
for i=1:numMuts,
for j=1:mutOps(i,1),
a = round(rand*(popSize-1)+1);
if c1(1:numVar)==endPop(a,(1:numVar))
c1(xZomeLength)=endPop(a,xZomeLength);
else
end
endPop(a,:)=c1;
```

```
end
end
else
for i=1:numXOvers,
xN=deblank(xOverFNs(i,:));
cp=find(rand(popSize,1)<xOverOps(i,1)==1);
if rem(size(cp,1),2) cp=cp(1:(size(cp,1)-1));
end
cp=reshape(cp,size(cp,1)/2,2);
for j=1:size(cp,1)
a=cp(j,1); b=cp(j,2);
[endPop(a,:) endPop(b,:)] = feval(xN,endPop(a,:),endPop(b,:),...
bounds,[gen xOverOps(i,:)]);
end
end
for i=1:numMuts
mN=deblank(mutFNs(i,:));
for j=1:popSize
endPop(j,:) = feval(mN,endPop(j,:),bounds,[gen mutOps(i,:)]);
eval(e1str);
end
end
end
end
gen=gen+1;
done=feval(termFN,[gen termOps],bPop,endPop);
```

```
startPop=endPop;
[bval,bindx] = min(startPop(:,xZomeLength));
startPop(bindx,:) = best;
end
[bval,bindx] = max(startPop(:,xZomeLength));
if display
fprintf(1,'\n%d %f\n',gen,bval);
end
x=startPop(bindx,:);
if opts(2)==0 %binary
x=b2f(x,bounds,bits);
bPop(bFoundIn,:)=[gen b2f(startPop(bindx,1:numVar),bounds,bits)...
startPop(bindx,xZomeLength)];
else
bPop(bFoundIn,:)=[gen startPop(bindx,:)];
end
if collectTrace
traceInfo(gen,1)=gen;
traceInfo(gen,2)=startPop(bindx,xZomeLength);
traceInfo(gen,3)=mean(startPop(:,xZomeLength));
end
%---------------------------------------------------------
```

Función PID_objfun_MSE

```
%-----------------------------------------------------------------
function [x_pop,fx_val]=PID_objfun_MSE(x_pop)
global PID
global t
global planta
Kp=x_pop(2); Ki=x_pop(3); Kd=x_pop(1);
pid_den=[1 0];
pid_num=[Kd Kp Ki];
pid_sys=tf(pid_num,pid_den);
S=pid_sys*planta;
PID=S/(S+1);
t=0:0.01:0.1;
y = step(PID,t);
for i=1:length(y)
error(i) = 1-y(i);
end
error_sq = error*error';
MSE=error_sq/max(size(error));
fx_val=1/MSE;
%-----------------------------------------------------------------
```

Función PID_objfun_IAE

```
%-----------------------------------------------------------------
function [x_pop, fx_val]=PID_objfun_IAE(x_pop)
global PID
```

```
global t
global planta
Kp=x_pop(2); Ki=x_pop(3); Kd=x_pop(1);
pid_den=[1 0];
pid_num=[Kd Kp Ki];
pid_sys=tf(pid_num,pid_den);
S=pid_sys*planta;
PID=S/(1+S);
t =0:0.01:0.1;
u=ones(size(t));
y= lsim(PID,u,t);
for
i=1:length(y)
error(i) = 1-y(i);
end
IAE=sum(abs(error));
fx_val=1/IAE;
%-----------------------------------------------------------
```

Funcion PID_objfun_ITAE

```
%-----------------------------------------------------------
function [x_pop, fx_val]=PID_objfun_ITAE(x_pop)
global PID
global t
global planta
Kp=x_pop(2); Ki=x_pop(3); Kd=x_pop(1);
```

```
pid_den=[1 0];
pid_num=[Kd Kp Ki];
pid_sys=tf(pid_num,pid_den);
S=pid_sys*planta;
PID=S/(S+1);
t =0:0.01:0.1;
u=ones(size(t));
y =lsim(PID,u,t);
for i=1:length(y)
error(i) = (abs(1-y(i)))*t(i);
end
ITAE=sum(error);
fx_val=1/ITAE;
%-----------------------------------------------------------
```

Función PID_objfun_ISE

```
%-----------------------------------------------------------
function [x_pop, fx_val]=PID_objfun_ISE(x_pop)
global PID
global t
global planta
Kp=x_pop(2); Ki=x_pop(3); Kd=x_pop(1);
pid_den=[1 0];
pid_num=[Kd Kp Ki];
pid_sys=tf(pid_num,pid_den);
S=pid_sys*planta;
```

```
PID=S/(1+S);

t=0:0.01:0.1;

u=ones(size(t));

y=lsim(PID,u,t);

for i=1:Length(y)

error(i) = 1-y(i);

end

error=error*error';

ISE=sum(error);

fx_val=1/ISE;

%---------------------------------------------------------------
```

B.1.3. Funciones Complementarias

Función arithXover

```
%---------------------------------------------------------------

function [c1,c2] = arithXover(p1,p2,bounds,Ops)

a = rand;

c1 = p1*a + p2*(1-a);

c2 = p1*(1-a) + p2*a;

%---------------------------------------------------------------
```

Función initializega

```
%---------------------------------------------------------------

function [pop] = initializega(num, bounds, evalFN,evalOps,options)

if nargin<5

options=[1e-6 1];
```

```
end
if nargin<4
evalOps=[];
end
if any(evalFN<48)
if options(2)==1
estr=['x=pop(i,1); pop(i,xZomeLength)=', evalFN ';'];
else
estr=['x=b2f(pop(i,:),bounds,bits); pop(i,xZomeLength)=', evalFN ';'];
end
else
if options(2)==1
estr=['[ pop(i,:) pop(i,xZomeLength)]=' evalFN '(pop(i,:),[0 evalOps]);'];
else %Binario GA
estr=['x=b2f(pop(i,:),bounds,bits);[x v]=' evalFN ...
'(x,[0 evalOps]); pop(i,:)=[f2b(x,bounds,bits) v];'];
end
end
numVars = size(bounds,1); % Numero de variables
rng = (bounds(:,2)-bounds(:,1))'; % El rango de variables
if options(2)==1 % Float GA
xZomeLength = numVars+1; % longitud string es numVar + fit
pop = zeros(num,xZomeLength); % ubicando nueva poblacion
pop(:,1:numVars)=(ones(num,1)*rng).*(rand(num,numVars))+(ones(num,1)*bound
s(:,1)');
```

```
else
bits=calcbits(bounds,options(1));
xZomeLength = sum(bits)+1;
pop = round(rand(num,sum(bits)+1));
end
%---------------------------------------------------------------
```

Función maxGenTerm

```
%---------------------------------------------------------------
function [done] = maxGenTerm(ops,bPop,endPop)
currentGen = ops(1);
maxGen = ops(2);
done = currentGen >= maxGen;
%---------------------------------------------------------------
```

Función normGeomSelect

```
%---------------------------------------------------------------
function [newPop] = normGeomSelect(oldPop,options)
q=options(2); % Probabilidad de la mejor seleccion
e = size(oldPop,2);
n = size(oldPop,1);
newPop = zeros(n,e);
fit = zeros(n,1);
x=zeros(n,2);
x(:,1) =[n:-1:1]';
```

```
[y x(:,2)] = sort(oldPop(:,e));

r = q/(1-(1-q)^n); % Normaliza la distribucion

fit(x(:,2))=r*(1-q).^(x(:,1)-1);

fit = cumsum(fit);

rNums=sort(rand(n,1));

fitIn=1; newIn=1;

while newIn<=n

if(rNums(newIn)<fit(fitIn))

newPop(newIn,:) = oldPop(fitIn,:);

newIn = newIn+1;

else

fitIn = fitIn + 1;

end

end

%---------------------------------------------------------------
```

B.2. Programa: Diseño del Controlador PID Convencional

```
% ---------------------------------------------------------------

% Acá se presenta un trabajo en base a buscar poblaciones de individuos

% (Kp,Td,Ti) de tal modo podamos escoger dentro de esta amplia gama los mas

% idóneos.

% ---------------------------------------------------------------

% Parámetros del Sistema Pelota y Aro

% ---------------------------------------------------------------

bm=2.05e-3;bb=1.92e-6;R=0.1025;r=7.5e-3;rb=9.5e-3;M=0.3;m=28.1e-3;g=9.8;
```

```
la=(1/2)*M*R^2;lb=(2/5)*m*rb^2;
% ------------------------------------------------------------------
% Planta
% ------------------------------------------------------------------
Gp=tf(1,conv([1 0],[la bm])); Gy=tf([1 0 g/R],[2*rb^2/(5*r^2)
bb/m*r^2 g/R]); P=Gp*Gy; P=zpk(P);
% Zero/pole/gain:
% 988.7269 (s^2 + 95.61)
% ---------------------------------
% s (s+1.301) (s^2 + 5.989e-009s + 149)
% ------------------------------------------------------------------
% Poblacion de la Familias PID
% ------------------------------------------------------------------
figure
Kp=[0.1 0.01 0.001 0.002];
Ti=[100 500 1000 1500];
Td=[0.1 0.2 0.3 0.4];
t=0:0.1:15;
for i=1:4 % Kp
for j=1:4 % Ti
for k=1:4 % Td
PID=tf(Kp(i)*[Td(k)*Ti(j) Ti(j) 1],[Ti(j) 0]);
S=series(PID,P);
F=S/(S+1);
t=0:0.1:15;
```

```
hold on
y=step(F,t);
plot(t,y,'k')
title('\bf Familia de curvas: Respuesta al escalon')
xlabel('\bf t(seg)')
ylabel('\bf Amplitud')
grid on
%drawnow
end
end
end
% ------------------------------------------------------------------
% Eligiendo Parametros adecuados del Controlador PID
% ------------------------------------------------------------------
S=P;
F=S/(S+1);
F=minreal(F);
% Parametros de sintonia PID
Kp=.001; % Controla
Ti=1000;
Td=0.2;
PID=tf(Kp*[Td*Ti Ti 1],[Ti 0]);
PID=zpk(PID);
S=series(PID,P);
F=S/(S+1);
```

```
F=minreal(F);
figure t=0:0.01:30;
u=ones(size(t));
y=lsim(F,u,t);
plot(t,u,'b')
hold plot(t,y,'k','LineWidth',2)
title(['\bf K_p = ',num2str(Kp),' ','\bf T_i = ',num2str(Ti),' ',...
'\bf T_d = ',num2str(Td)])
xlabel('\bf t(seg)')
ylabel('\bf Amplitud')
% ------------------------------------------------------------------
```

B.3. Programa: Diseño del Controlador PID Óptimo

```
%-----------------------------------------------------------------
bm=2.05e-3; bb=1.92e-6; R=0.1025; r=7.5e-3; rb=9.5e-3; M=0.3;
m=28.1e-3; g=9.8; Ia=(1/2)*M*R^2; Ib=(2/5)*m*rb^2;
Gv=tf(1,[Ia bm]);
Gp=tf(1,conv([1 0],[Ia bm]));
Gy=tf([1 0 g/R],[2*rb^2/(5*r^2) bb/m*r^2 g/R]);
P=Gp*Gy;
Ts=0.1;
g=c2d(P,Ts,'zoh');
tfinal=20;
gamma=0.01; dq=0.001;
T=0:Ts:tfinal;
```

```
des = ones(size(T))';
q0=0; q1=0; q2=0;
maxiter=1000;
J=zeros([1 maxiter]);
for
i=1:maxiter,
d=tf([q0+dq q1 q2],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J1=norm(des-s);
d=tf([q0 q1+dq q2],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J2=norm(des-s);
d=tf([q0 q1 q2+dq],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J3=norm(des-s);
d=tf([q0 q1 q2],[1 -1 0],-1);
f=(d*g)/(1+d*g);
s=step(f,T);
J(i)=norm(des-s);
if (mod(i-10,10)==0),
J(i)
plot(J);
```

```
title(['it = ' ,num2str(i)])
end
p1=(J1-J(i))/dq;
p2=(J2-J(i))/dq;
p3=(J3-J(i))/dq;
Delta=sqrt(p1*p1+p2*p2+p3*p3);
q0=q0-(gamma/Delta)*p1;
q1=q1-(gamma/Delta)*p2;
q2=q2-(gamma/Delta)*p3;
end
q=[q0 q1 q2];
Cz=tf([q(1) q(2) q(3)],conv([1 -1],[1 0]),Ts);
S=Cz*g;
F=S/(S+1);
tk=0:Ts:tfinal;
uk=ones(size(tk));
yd=lsim(F,uk,tk);
stairs(tk,uk,'k')
stairs(tk,yd,'r')
xlabel('tiempo(kT)')
ylabel('c(kT)')
%-------------------------------------------------------------
```