UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

ÁREA ACADÉMICA DE INGENIERÍA

DISEÑO DE UN ESTABILIZADOR PARA EL PÉNDULO INVERTIDO OPTIMIZADO POR ALGORITMOS GENÉTICOS.

T E S I S

PARA OBTENER EL TÍTULO DE
Maestro en Ciencias en Ingeniería de Manufactura
P R E S E N T A :

Ing. Antonio Rodríguez Cruz

DIRECTOR DE TESIS
Dr. Norberto Hernández Romero

CODIRECTOR DE TESIS

Dr. Juan Carlos Seck Tuoh Mora

Noviembre 2014.

Agradecimientos

En primer lugar quiero agradecer a Dios por la oportunidad que me brinda día con día de mejorar como ser humano.

A mis padres Antonio Rodríguez Ugalde y Dolores Cruz Vivanco por su amor, educación, comprensión y tiempo.

A mis dos hijos Joshua y Ángel que son mi motor para seguir adelante.

A mis amigos, Patricio Ordaz y Carlos Alberto Rodríguez por su valiosa amistad.

A mi asesor el Doctor Norberto Hernández Romero por su gran apoyo y paciencia.

Y a ti, Mónica gracias por llegar a mi vida y convertirte en una de las personas más importantes en ella.

Resumen

Este trabajo reporta el modelado, simulación y control por retroalimentación de estados para el péndulo invertido en 2 puntos de equilibrio inestables utilizando algoritmos evolutivos del tipo AGC (*"Continuos Genetic Algorithms"*). El modelo se encuentra definido por ecuaciones diferenciales no lineales ordinarias (EDNLO), estas ecuaciones son necesarias para evaluar el comportamiento dinámico de un robot, el cual es linealizado para obtener un sistema en variables de estado, con la finalidad de determinar las ganancias adecuadas que ayuden a controlar al péndulo. Posteriormente se desarrolla una estrategia de sintonización del control por retroalimentación de estados usando los AGC, los cuales solo utilizan una entrada y una salida del sistema dinámico del péndulo y conforme el algoritmo evoluciona en el tiempo determina las ganancias. Esta técnica evita linealizar sistemas dinámicos, determinar raíces y desarrollar matemática compleja debido a que, como se mencionó anteriormente, solo utiliza una entrada y una salida del sistema dinámico del péndulo. Finalmente se analizan los resultados obtenidos para desarrollar una metodología formal y consistente para que la implementación de los AGC sea favorable en distintas aplicaciones. Cabe mencionar que este trabajo solo se realizó a nivel simulación utilizando Matlab.

Abstract

This work reports the modeling, simulation, and control by state feedback for the inverted pendulum in 2 unstable points of balance, using evolutionary algorithms of the AGC type (*"Continuous Genetic Algorithms"*). The model is defined by nonlinear differentials equations (EDNLO). These equations are necessary to test the dynamic behavior of a robot, which is linearized to obtain a system in variables of state for determining the suitable gains to stabilize the pendulum. Then, a strategy of tuning the control by state feedback using AGC is developed. The AGC only uses one input and one output of the dynamic system and the algorithm determines the gains as it evolves in time. This technique avoids the linearization of dynamic systems and the use of complex roots, due to, it only uses one input and one output as described above. Finally the results obtained are analyzed to develop a formal and consistent methodology to get a favorable implementation of the AGC in different applications. This work has been developed with computer simulation in Matlab.

Contenido

3 Optimización del control de retroalimentación de estados utilizando Algoritmos Genéticos.

4 Pruebas y resultados.

5 Conclusiones

Apéndices

Lista de Figuras

Lista de Tablas

Antecedentes

El péndulo invertido es un servo mecanismo que consta de un carro en el cual se encuentra montado un péndulo que puede girar libremente. El carro está controlado por un servomotor y su principal función es la de aplicar fuerzas al péndulo, este sistema es conocido por ser uno de los problemas más importantes y clásicos de la teoría de control ya que al desarrollar el modelo matemático se determina un control inestable y no lineal. A menudo, es utilizado como ejemplo académico, principalmente porque de él se desarrolla un sistema de control más accesible en comparación a otros sistemas y por otro lado, permite mostrar las principales diferencias de un sistema en lazo abierto y de su estabilización a lazo cerrado. Pese a existir diferentes técnicas a la hora de diseñar el regulador óptimo capaz de estabilizar el péndulo, algunos métodos resultan ser una mejor opción que otras [7].

Dentro de la teoría del control clásico, uno de los sistemas más comunes para analizar es el péndulo invertido, esto es, debido a que permite mostrar de manera no tan compleja las diferencias que existen entre el control de lazo abierto y el control de lazo cerrado [8]. Hoy en día existen diferentes sistemas mecánicos que representan al péndulo invertido, algunos son el péndulo invertido de doble articulación en un plano inclinado o el péndulo invertido de tipo rotacional. Pero el sistema mecánico más conocido es el compuesto por un carro el cual tiene montado un péndulo que puede girar libremente. Dicho sistema puede emular la dinámica de un objeto volador como un misil y es muy utilizado para realizar una gran variedad de simulaciones probando diferentes técnicas de control [2].

Para el levantamiento del péndulo se tienen algunos antecedentes como el reportado por Mori (Moriet al., 1976) que implica un control en tiempo mínimo, los de Aström (Aström and Furuta, 1996) y Yoshida (Yoshida, 1999) que plantean el levantamiento del péndulo como un sistema subactuado y emplea conceptos de energía. Como antecedentes del control en la posición

vertical tenemos los reportados por Verde (Verde et al., 1996), que consisten de una ubicación de polos con compensación de fricción estática [4, 5].
Dentro de los sistemas automatizados entra lo que es la teoría de sistemas de control, la cual se encarga de analizar y diseñar sistemas de controlabilidad que brinden comportamientos deseados utilizando el concepto de retroalimentación que consiste en evaluar las variables de interés del sistema y con dicha evaluación controlar el sistema. En nuestros días el control tiene aplicaciones en un gran número de disciplinas como en la industria aeronáutica, en química, robótica, la industria automotriz, no ingenieriles, entre muchas otras. Las ventajas que genera dentro de la industria son inmensas como la reducción de consumo de energía, mayores y mejores niveles de seguridad y reducción de la contaminación [2].

El punto de inicio en el análisis de un sistema de control es representarlo matemáticamente, es decir, generar su modelo matemático por lo general en un conjunto de ecuaciones diferenciales. En este tipo de modelos existen los lineales y no lineales. En los cuales existe una gran cantidad de técnicas de diseño. Algunos ejemplos de los sistemas no lineales son la estructura de un ala de avión, manipuladores de robots, misiles guiados, entre otros [3].

Por otra parte, los algoritmos genéticos fueron desarrollados en la década de los años 60´s por el Ingeniero John Holland imitando la genética y la selección natural basados en la búsqueda aleatoria para encontrar mediante una población establecida la mejor solución deseada. John Holland inicio sus investigaciones acerca de los algoritmos genéticos en la Universidad de Michigan en Ann Harbor dentro del grupo "*logic of conputers"*. John se basaba en el libro del biólogo evolucionista A. Fisher titulado "*La teoría genética de la selección natural"* de donde concluyo generar programas adecuadamente adaptados para un fin específico. Estas ideas en principio mezcladas con el aprendizaje obtenido en el curso titulado "*Teoría de sistemas adaptativos"* que impartía a la par con sus investigaciones, fueron las bases de lo que hoy conocemos como los algoritmos genéticos. Lo que Holland consiguió en sus primeras investigaciones fueron dos puntos clave: Imitar la adaptación que tienen los seres vivos en un medio ambiente y diseñar

sistemas programables que mantuviesen los mecanismos importantes de los sistemas naturales.

David Goldberg fue uno de los fundadores de los algoritmos genético. Al igual que Holland intentó y logró aplicar los algoritmos genéticos en la industria. Aunque al principio Holland creía equivocadamente que los algoritmos genéticos no funcionarían en esta área debido a la complejidad de los sistemas, sin embargo, Goldberg logró aplicarlos en el control de flujo de un sistema de tuberías de gas. Demostrando así la versatilidad de dichos algoritmos [1].

Al igual que Goldberg, Kenneth De Jong demostró con sus investigaciones plasmadas en su tesis que los algoritmos genéticos podían desenvolverse en una gran variedad de funciones de pruebas. Con el paso del tiempo estos algoritmos han sido una opción viable a la optimización de sistemas esto debido a que trabajan con funciones no lineales, lineales, discretas, no suaves, probabilísticas, etc. Se aplican desde la predicción en la bolsa y la planificación de la cartera de valores, bioquímica y biología molecular, diseño de horarios en aeropuertos y líneas de montaje hasta ingeniería aeroespacial y diseño de microchips [10].

Objetivos

Objetivo General

Diseñar una estrategia de sintonización para un estabilizador por retroalimentación de estados del péndulo invertido mediante algoritmos genéticos continuos para generar estabilidad en puntos cercanos al estado de equilibrio.

Objetivos Específicos

1. Modelar y simular el sistema de EDNLO del péndulo invertido usando Matlab®
2. Realizar la sintonización de un control por retroalimentación de estados del péndulo invertido en 2 puntos de equilibrio inestable., utilizando Matlab®.
3. Realizar pruebas de desempeño de los Algoritmos GA tomando en cuenta:
   - Tamaño de la población.
   - Tiempo de respuesta.
   - Número de iteraciones.
4. Optimizar un control retroalimentado usando un algoritmo genético continuo AGC.

Justificaciones

Al utilizar las técnicas de control lineal se desarrolla un control por retroalimentación de estados, el cual logra que los puntos inestables se hagan estables. Sin embargo existen otras alternativas que no son sensibles a los parámetros del modelo matemático y a través de las señales de entrada y salida del sistema es posible determinar las ganancias del control por retroalimentación de estados, por lo que se desarrolla un estabilizador para el péndulo utilizando AGC ya que actualmente la metodología para determinar la ubicación de polos es compleja y para lograr lo anterior se necesita desarrollar el modelo matemático que represente al péndulo invertido y linealizarlo para proponer la ubicación de los polos y así obtener las ganancias adecuadas del sistema; para esto puede utilizar el comando "acker" del software Matlab®. Por lo que el propósito de diseñar el estabilizador utilizando AGC es la de generar las ganancias de la retroalimentación adecuadas para controlar el péndulo sin la necesidad de linealizar el modelo que lo representa debido a que dichas ganancias se determinan utilizando solo el modelo no lineal. Estas ganancias se generan a partir de un rango establecido previamente dentro del código del AGC.

Con dicha metodología se pueden determinar ganancias adecuadas para estabilizar otros sistemas no lineales.

Estado del Arte

En diferentes artículos se han desarrollado una gran variedad de técnicas para lograr la estabilización y el control del péndulo invertido, estos métodos van desde el control clásico utilizando ecuaciones de Euler-LaGrange, PID, control difuso, entre otros.

Por ejemplo en los reguladores PID se encuentra un desarrollo matemático de acuerdo al criterio de Routh del cual se logra llegar a un control por medio de un proporcional integral derivativo, el cual es el más estable para el control debido a que modera el sobre impulso del integrador y adelanta el tiempo de estabilización del derivativo para ajustar la señal rápidamente, es decir, fusiona ambas y genera una señal de control con un pequeño sobre impulso y también llegando más rápido a la estabilidad deseada. Esto ayuda de gran manera debido a que al evitar un sobre impulso, se evita en la implementación dañar el motor [2].

Algunos otros métodos utilizan diferentes resoluciones matemáticas como por ejemplo la incursión del principio de invariancia de Lasalle para obtener algún análisis de convergencia, también están las propiedades elásticas del péndulo juntos con la teoría de forma normal y el método Melnikov pueden utilizarse para el análisis de una nueva clase de movimientos del sistema en el caso de las grandes perturbaciones [11].

El péndulo invertido se ha implementado de diferentes maneras, tal es el caso del péndulo giratorio sobre el eje horizontal, el cual consta de un brazo que gira en el plano horizontal, uno de sus extremo es montado sobre el eje de un motor y un péndulo es montado sobre el otro extremo [16]. El objetivo en este tipo de péndulo es el de hallar una ley de control al momento de rotación de salida del motor tal que el movimiento del péndulo invertido sea estabilizado sobre su eje vertical [17].

Muchas técnicas de control como el método de control PD, el de compensación adaptativa de fricción, el control robusto dinámico y/o el método de compensación basado en el modelo de fricción, para los sistemas subactuados de n grados de libertad se caracterizan porque aplican técnicas de control convencionales lo cual conlleva a una gran implementación de ecuaciones y desarrollos matemáticos complejos [3]; por lo que el AG es una opción de optimización muy viable, ya que al explorar simultáneamente un gran número de posibles soluciones, la probabilidad de encontrar un óptimo global aumenta considerablemente, por lo que tiene fuerte solidez y es aplicado en diferentes áreas [6].

Dentro de los algoritmos evolutivos se encuentran los algoritmos genéticos binarios (AGB), los algoritmos genéticos continuos (AGC), algoritmos de optimización por enjambre de partículas (PSO), estrategias de evolución, programación evolutiva. Cada uno de ellos tiene características específicas pero el concepto básico de todos es el de buscar una solución generando posibles soluciones simultáneamente. Por ejemplo, él PSO se basa en la inteligencia de un enjambre de partículas para solucionar un problema que requiera N variables, se posiciona un conjunto de partículas, inicialmente, de manera aleatoria en el espacio de búsqueda N-dimensional, reposicionándose de forma iterativa y siguiendo un comportamiento determinado. Cada posición que toman las partículas corresponde a una posible solución [1].

En cuanto a los algoritmos genéticos se refiere, se tiene una buena cantidad de ejemplos en donde son de gran utilidad, como por ejemplo en la industria del cine, mejora de aerodinámica, diseño automatizado, teoría de juegos, predicción, optimización de estructuras moleculares, buscar ejemplos. Es por tal motivo que un AG se desea implementar en un sistema sub-actuado, lo que se realizará en este trabajo.

Hipótesis

Mediante algoritmos de búsqueda aleatoria tal como el AG, es posible encontrar las ganancias de retroalimentación de estados para lograr puntos de equilibrio en modelos dinámicos como el péndulo invertido, usando únicamente la entrada y salida del sistema. El diseño de una metodología de un estabilizador para un control por retroalimentación de estados del péndulo invertido validará el desempeño de los algoritmos evolutivos del tipo AGC para generar ganancias en dicho control del diseño desarrollado.

Capítulo 1

Modelo matemático del péndulo invertido.

1.1 Introducción.

El péndulo invertido como se mencionó anteriormente es un servo mecanismo que se compone de dos elementos fundamentales: el primero es un bloque el cual puede moverse libremente en la horizontal. El segundo elemento es un péndulo simple o una barra unida al bloque por medio de algún elemento como un balero para que dicho eslabón pueda girar libremente sobre su eje, dicho bloque es controlado por un servomotor el cual se encarga de aplicar la fuerza necesaria en la dirección correcta para mantener en una posición deseada al péndulo. Si se considera al péndulo separado del bloque, este tiene dos puntos de equilibrio: uno estable, abajo; y otro inestable, arriba. La finalidad es desarrollar el modelo matemático del sistema, linealizarlo y simular el comportamiento del péndulo. El objetivo del control es cambiar la dinámica del sistema para que en la posición vertical, arriba, se tenga un punto de equilibrio estable. En otras palabras, la idea es encontrar la fuerza que ha de aplicarse al bloque para que el péndulo se mantenga en su posición, incluso si se le perturba con un impulso [6].

El comportamiento físico de un sistema puede ser representado matemáticamente por algún tipo de sistema de ecuaciones o por un modelo matemático. En este capítulo se desarrolla el modelo matemático del péndulo invertido, el cual se compone de un bloque con una masa m_1 y un péndulo de longitud l, masa m_2 y un ángulo θ de inclinación. Ambos tienen posiciones tanto en x como en y. La figura 1.1 muestra el péndulo invertido.

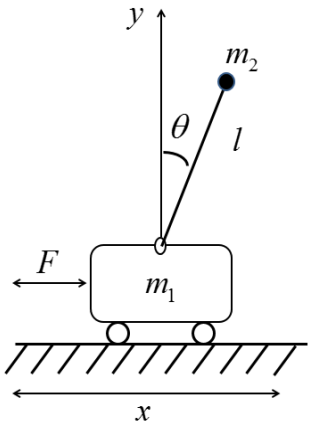

Figura 1.1. Esquema del péndulo invertido.

Donde:

$m_1 =$ Masa del bloque (Kg).

$m_2 =$ Masa del péndulo (Kg).

$l =$ Longitud del péndulo (m).

$\theta =$ Ángulo del péndulo respecto a la vertical (rad).

$F =$ Fuerza aplicada al bloque (N).

$x =$ Coordenadas de posición del bloque (m).

$y =$ Coordenadas de posición del péndulo (m).

$\rho_1(x,y) =$ Posiciones del carro tanto horizontal como vertical.

$\rho_2(x,y) =$ Posición del péndulo.

1.2 Modelado con Euler-Lagrange.

Uno de los métodos que se utiliza para determinar el modelo matemático de un sistema es el método de Euler-Lagrange, el cual nos puede ayudar a encontrar el modelo matemático del sistema que se analiza en ese momento; dicho modelo tiene una serie de fórmulas, las cuales determinan las posiciones, velocidades y aceleraciones del sistema.

El método de Euler-Lagrange es representado con la ecuación 1.1.

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_1}\right\} - \frac{\partial L}{\partial q_1} = Q \tag{1.1}$$

Dónde:

q_i = Posición articular.

$\dot{q}_i = \frac{d}{dt} q_i$ = Velocidad.

$L(q,\dot{q})$ = Lagrangiano.

Q = Fuerza generalizada.

En donde el Lagrangiano se representa por la ecuación 1.2.

$$L(q,\dot{q}) = T(q,\dot{q}) - P(q) \tag{1.2}$$

Dónde:

$T(q,\dot{q})$ = Energía cinética.

$P(q)$ = Energía potencial.

La energía cinética está representada por la ecuación 1.3.

$$T(q,\dot{q}) = \frac{1}{2}\sum_{i-1}^{n} m_i v_i^2 \tag{1.3}$$

m_i = Número de masas del sistema.

v_i^2 = Número de velocidades de las masas respectivas.

Mientras la energía potencial se representa por la ecuación 1.4.

$$P(q) = g \sum_{i-1}^{n} m_i h_i \tag{1.4}$$

En donde

h_i = Altura de cada elemento del sistema.

g = Índice de gravedad.

Para determinar los grados de libertad de un sistema se utiliza:

$$GdL = n_g - n_r$$

Donde

n_g = Número de coordenadas generalizadas.

n_r = Número de restricciones del sistema.

Para determinar el modelo matemático del péndulo invertido primero se necesita obtener la energía cinética y potencial (una vez obtenidas, se determina el Lagrangiano y por último se desarrolla el modelo matemático del sistema). Por lo tanto para llegar a dichas energías primero se deben hallar las posiciones del carro y del péndulo por separado, para poder determinar con estos valores las velocidades. Por lo que:

Se determinan las posiciones en los ejes x é y.

$\rho_1(x,y)$ = Posiciones del carro en x é y.

$\rho_2(x,y)$ = Posiciones del péndulo en x é y.

La posición del carro se genera únicamente en el eje x, por lo que en el eje y es cero. Esto es porque el carrito únicamente puede avanzar o retroceder, por lo tanto:

$$\rho_1(x,y)=\begin{bmatrix}\rho_1(x)\\ \rho_2(y)\end{bmatrix}=\begin{bmatrix}x\\ 0\end{bmatrix}$$

En la posición del péndulo con respecto al eje x se tiene un ángulo θ de longitud l más la posición del carro en ese eje. En la posición del péndulo con respecto al eje y tiene un ángulo θ con respecto a dicho eje por la longitud l.

$$\rho_2(x,y)=\begin{bmatrix}\rho_2(x)\\ \rho_2(y)\end{bmatrix}=\begin{bmatrix}lsen\theta+x\\ l\cos\theta\end{bmatrix}$$

Una vez que se determinan las ecuaciones representantes de las posiciones de cada elemento, lo siguiente es determinar las velocidades del bloque y del péndulo por separado, partiendo de las posiciones, por lo tanto se derivan ambas:

$$v_1=\frac{d}{dt}\rho_1(x,y)=\begin{bmatrix}\frac{d}{dt}\rho_1(x)\\ \frac{d}{dt}\rho_1(y)\end{bmatrix}=\begin{bmatrix}\dot{x}\\ 0\end{bmatrix}$$

$$v_2=\frac{d}{dt}\rho_2(x,y)=\begin{bmatrix}\frac{d}{dt}\rho_2(x)\\ \frac{d}{dt}\rho_2(y)\end{bmatrix}=\begin{bmatrix}l\cos\theta\dot{\theta}+\dot{x}\\ -lsen\theta\dot{\theta}\end{bmatrix}$$

Pero V_1 y V_2 son vectores por lo tanto se aplica:

$$v_1=\langle v_1,v_2\rangle=\left\langle\begin{bmatrix}\dot{x}\\ 0\end{bmatrix},\begin{bmatrix}l\cos\theta\dot{\theta}+\dot{x}\\ -lsen\theta\dot{\theta}\end{bmatrix}\right\rangle$$

Por consiguiente se tiene:

$$v_1^2 = (\dot{x})^2 + (0) = \dot{x}^2$$

$$v_2^2 = (l\cos\dot{\theta}\theta + \dot{x})^2 + (-lsen\theta\dot{\theta})^2 = l^2\dot{\theta}^2 + 2l\dot{x}\dot{\theta}\cos\theta + \dot{x}^2$$

Para sustituir a coordenadas articulares se debe encontrar GdL, por lo que:

$$GdL = n_g - n_r$$

Donde

$n_g =$ Número de coordenadas generalizadas. En este caso se tienen 2 coordenadas (x,y) en el péndulo y una coordenada en el carrito (x), por lo que:

$n_g = 3$.

$n_r =$ Número de restricciones. Que en el péndulo invertido solo se tiene como restricción al péndulo. Por lo que:

$n_r = 1$

$$GdL = 3 - 1 = 2$$

Sustituyendo a coordenadas articulares se tiene:

$$n = 2 \quad \therefore$$

$$q_1 = x \longrightarrow \dot{q}_1 = \dot{x}$$

$$q_2 = \theta \longrightarrow \dot{q}_2 = \dot{\theta}$$

Al sustituir en las velocidades se tiene:

$$v_1^2 = \dot{q}_1^2$$

$$v_2^2 = l^2 \dot{q}_2^2 + 2l \dot{q}_1 \dot{q}_2 \cos q_2 + \dot{q}_1^2$$

Una vez que se obtienen las velocidades se determinan las energías cinética y potencial de las ecuaciones 1.3 y 1.4.

$$T(q,\dot{q}) = \frac{1}{2}\sum_{i=1}^{2} m_i v_i^2 \tag{1.3}$$

$$= \frac{1}{2} m_1 \dot{q}_1^2 + \frac{1}{2} m_2 (l^2 \dot{q}_2^2 + 2l \cos q_2 \dot{q}_1 \dot{q}_2 + \dot{q}_1^2)$$

$$= \frac{1}{2} \dot{q}_2^2 (m_1 + m_2) + \frac{1}{2} m_2 l^2 \dot{q}_2^2 + \frac{1}{2} m_2 l^2 \dot{q}_2^2 + m_2 l \cos q_2 \dot{q}_1 \dot{q}_2$$

La energía potencial:

$$P(q) = g\sum_{i=1}^{2} m_i h_i \tag{1.4}$$

Sustituyendo se tiene:

$$P(q) = g h_1 m_1 + g m_2 l \cos q_2$$

Donde:

$h_1 = 0 \longrightarrow$ Altura del carrito.

$h_2 = l \cos q_2 \longrightarrow$ Altura del péndulo.

$$P(q) = gm_2 l \cos q_2$$

Ahora se sustituye la energía cinética y potencial en la ecuación 1.2 (Lagrangiano) por lo que se obtiene:

$$L(q,\dot{q}) = T(q,\dot{q}) - P(q) \tag{1.2}$$

$$= \frac{1}{2}\dot{q}_1^2(m_1 + m_2) + \frac{1}{2}m_2 l^2 \dot{q}_2^2 + m_2 l \cos q_2 \dot{q}_1 \dot{q}_2 - glm_2 \cos q_2$$

En donde la ecuación 1.1 de Euler-Lagrange establece que:

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_i}\right\} - \frac{\partial L}{\partial q_i} = Q \tag{1.1}$$

Entonces para $i = 1$ se tiene lo siguiente:

$$\frac{\partial L}{\partial \dot{q}_1} = \frac{1}{2}\dot{q}_1^2(m_1 + m_2) + m_2 l \cos q_2 \dot{q}_1 \dot{q}_2 \quad \therefore$$

$$\frac{\partial L}{\partial \dot{q}_1} = \dot{q}_1(m_1 + m_2) + m_2 l \cos q_2 \dot{q}_2$$

Al derivar con respecto al tiempo se obtiene:

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_1}\right\} = \ddot{q}_1(m_1 + m_2) + lm_2 \cos q_2 \ddot{q}_2 - \dot{q}_2^2 lm_2 sen q_2$$

Una vez que se determinó el primer elemento de la fórmula de Euler-Lagrange se obtiene el segundo elemento de dicha ecuación $\therefore$

$$\frac{\partial L}{\partial q_1} = 0$$

Donde

$$Q_1 = F_1$$

Ahora se determina para $i = 2 \therefore$

$$\frac{\partial L}{\partial \dot{q}_2} = m_2 l^2 \dot{q}_2 + m_2 l \cos q_2 \dot{q}_1$$

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_2}\right\} = l^2 m_2 \ddot{q}_2 + m_2 l \cos q_2 \ddot{q}_1 - \dot{q}_1 l m_2 sen q_2 \dot{q}_2$$

$$\frac{\partial L}{\partial q_2} = -m_2 l sen q_2 \dot{q}_1 \dot{q}_2 + g m_2 l sen q_2$$

$$Q_2 = \tau \quad \therefore$$

Las dos ecuaciones de Euler-Lagrange que se obtienen son:

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_1}\right\} - \frac{\partial L}{\partial q_1} = Q_1$$

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_2}\right\} - \frac{\partial L}{\partial q_2} = Q_2$$

Ahora se sustituye en las ecuaciones anteriores por lo que:

$$\ddot{q}_1(m_1 + m_2) + l m_2 \cos q_2 \ddot{q}_2 - l m_2 sen q_2 \dot{q}_2^2 = F_1$$

$$m_2 l^2 \ddot{q}_2 + m_2 l \cos q_2 \ddot{q}_1 - g m_2 l sen q_2 = \tau$$

Ya que se encontraron las 2 ecuaciones que representan al sistema, lo siguiente es sustituirlas en la forma general de un robot. Dicha forma se encuentra representada en la ecuación 1.5:

$$D(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=F \tag{1.5}$$

Donde:

$D(q)\ddot{q}=$ Fricción.

$C(q,\dot{q})\dot{q}=$ Coriolis.

$G(q)=$ Gravedad.

$F=$ Fuerzas.

$$q=\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \qquad \dot{q}=\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \qquad \ddot{q}=\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \qquad F=\begin{bmatrix} F_1 \\ 0 \end{bmatrix}$$

De la ecuación general de un robot se despeja el término más alto el cual es $\ddot{q}$ y se obtiene lo siguiente:

$$\ddot{q}=D^{-1}(q)F-D^{-1}(q)C(q,\dot{q})\dot{q}-D^{-1}(q)G(q)$$

Donde

$$D(q)=\begin{bmatrix} m_1+m_2 & m_2 l\cos q_2 \\ m_2 l\cos q_2 & m_2 l^2 \end{bmatrix}$$

Pero

$$D^{-1}=\frac{1}{det(A)}adj(A) \quad \therefore$$

$$D^{-1}(q)=\begin{bmatrix} \dfrac{1}{m_1+m_2-m_2\cos^2 q_2} & -\dfrac{\cos q_2}{l(m_1+m_2-m_2\cos^2 q_2} \\ -\dfrac{\cos q_2}{l(m_1+m_2-m_2\cos^2 q_2} & \dfrac{m_1+m_2}{m_2 l^2(m_1+m_2-m_2\cos^2 q_2} \end{bmatrix}$$

$$C(q,\dot{q}) = \begin{bmatrix} 0 & -lm_2 senq_2\,\dot{q}_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} 0 \\ -gm_2 lsenq_2 \end{bmatrix} 2$$

$$F = \begin{bmatrix} F_1 \\ 0 \end{bmatrix}$$

$$\ddot{q} = \begin{bmatrix} \dfrac{1}{m_1 + m_2 - m_2 \cos^2 q_2} & -\dfrac{\cos q_2}{l(m_1 + m_2 - m_2 \cos^2 q_2} \\ -\dfrac{\cos q_2}{l(m_1 + m_2 - m_2 \cos^2 q_2)} & \dfrac{m_1 + m_2}{m_2 l^2 (m_1 + m_2 - m_2 \cos^2 q_2)} \end{bmatrix} \begin{bmatrix} F_1 \\ 0 \end{bmatrix} - \ldots\ldots$$

$$- \begin{bmatrix} \dfrac{1}{m_1 + m_2 - m_2 \cos^2 q_2} & -\dfrac{\cos q_2}{l(m_1 + m_2 - m_2 \cos^2 q_2} \\ -\dfrac{\cos q_2}{l(m_1 + m_2 - m_2 \cos^2 q_2)} & \dfrac{m_1 + m_2}{m_2 l^2 (m_1 + m_2 - m_2 \cos^2 q_2} \end{bmatrix} \begin{bmatrix} -lm_2 senq_2\,\dot{q}_2^2 \\ 0 \end{bmatrix} - \ldots$$

$$- \begin{bmatrix} \dfrac{1}{m_1 + m_2 - m_2 \cos^2 q_2} & -\dfrac{\cos q_2}{l(m_1 + m_2 - m_2 \cos^2 q_2} \\ -\dfrac{\cos q_2}{l(m_1 + m_2 - m_2 \cos^2 q_2} & \dfrac{m_1 + m_2}{m_2 l^2 (m_1 + m_2 - m_2 \cos^2 q_2} \end{bmatrix} \begin{bmatrix} 0 \\ -gm_2 lsenq_2 \end{bmatrix}$$

Finalmente se llega a:

$$\ddot{q} = \begin{bmatrix} \dfrac{1}{m_1 + m_2 sen^2 q_2} \left[m_2 senq_2 (l\,\dot{q}_2^2 - g\cos q_2) + F_1 \right] \\ \dfrac{1}{l(m_1 + m_2 sen^2 q_2)} \left[gsenq_2 (m_1 + m_2) - F_1 \cos q_2 - lm_2 senq_2 \cos q_2\,\dot{q}_2^2 \right] \end{bmatrix}$$

1.3 Simulación del péndulo invertido en lazo abierto.

Después de obtener cada uno de los elementos de la forma general de un robot, se procede a simular el modelo matemático en Matlab® para lazo abierto, los parámetros del sistema se muestran en la tabla 1.1. Los parámetros m_1, m_2 y l se proponen mientras que g es el valor de la gravedad en m/s^2.

PARÁMETROS DE SIMULACIÓN		
Parámetro	**Valor**	**Unidad**
m_1	3.2	Kg
m_2	0.32	Kg
l	0.44	M
g	9.81	m/s

Tabla 1.1 Parámetros de simulación.

El código del sistema en lazo abierto se realiza en Matlab® con condiciones iniciales 0, 0, 0, 0. Estas condiciones iniciales se refieren a las posiciones y velocidades del sistema (del péndulo y del carrito) las cuales están ordenadas como:

- Posición 1: Posición del carrito en el eje x.
- Posición 2: Posición del péndulo la cual se encuentra establecida en grados.
- Posición 3: Velocidad del carrito establecida en m/s.
- Posición 4: Velocidad del péndulo.

Se toman todas las C. I. como cero porque se simula que el bloque se encuentra en la posición del eje x en el origen, en estado de reposo y el péndulo se encuentra 90° o lineal con respecto a la vertical sin movimiento alguno; por lo que idealmente la simulación debe de generar una gráfica de comportamiento lineal, es decir, los resultados de los movimientos y posiciones no deben variar.

Los programas que muestran lo anterior son el programa 1 y programa 2 en él apéndice A.

Al realizar la simulación con la condiciones iniciales 0, 0, 0, 0 el sistema se encuentra estable, debido a que este se encuentra en reposo, es decir, el bloque y el péndulo permanecen sin movimiento.

Una vez que se comprueba que el sistema es estable en lazo abierto con las C. I. 0, 0, 0, 0. Ahora se cambia la posición del péndulo a 1° de inclinación, por lo que las condiciones iniciales se cambian a 0, 0.01745, 0, 0. El programa 3 del apéndice A es el que representa los cambios.

Al graficar con las C. I. anteriores se obtiene la gráfica de la figura 1.2, la cual muestra cada uno de los comportamientos del sistema, por ejemplo; la posición del carro (azul marino) varía muy poco al igual que su velocidad (rojo), esto es debido a los impulsos que le genera el movimiento del péndulo, por otro lado, la posición del péndulo (verde) al no estar a 90° comienza a variar, es decir, el péndulo comienza a caer llegando inclusive a dar casi dos vueltas, logra detenerse por un periodo de tiempo muy pequeño y comienza de nueva cuenta a moverse pero ahora en sentido contrario al del inicio. Con respecto a la velocidad del péndulo (azul claro), esta se eleva a su punto máximo cuando el péndulo casi alcanza a realizar la segunda vuelta y llega a cero en el mismo momento en el que el péndulo logra detenerse por un pequeño lapso de tiempo, lo cual hace al sistema inestable.

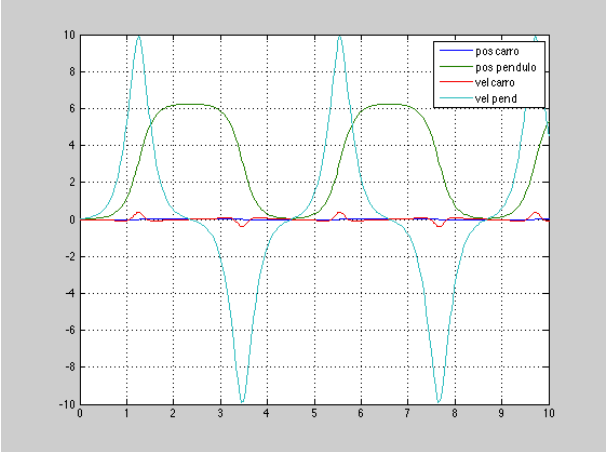

Figura 1.2 Gráfica del sistema en lazo abierto con C.I. Pos. Carro=0, Pos. Péndulo=0.01745, Vel. Carro=0, Vel. Péndulo=0.

Como se menciona anteriormente, con todas las condiciones iniciales en cero, el sistema permanece estable, esto es porque el péndulo se mantiene totalmente alineado hacia arriba. Por otro lado, al inclinar un poco el péndulo, este comenzará a oscilar.

1.4 Diseño de un sistema linealizado.

Ahora se desea obtener una retroalimentación del sistema para estabilizar al péndulo en algún punto cercano al más alto, por lo tanto se necesita linealizar el modelo matemático para determinar las ganancias del sistema en lazo cerrado, por lo que:

Sí

$$\ddot{q} = \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \quad \therefore$$

En cada elemento se tiene el siguiente resultado:

$$\ddot{q}_1 = \frac{1}{m_1 + m_2 sen^2 q_2}\left[m_2 sen q_2 (l \dot{q}_2^2 - g \cos q_2) + F_1 \right]$$

$$\ddot{q}_2 = \frac{1}{l(m_1 + m_2 sen^2 q_2)}\left[gsen(m_1 + m_2) - F_1 \cos q_2 - lm_2 senq_2 \cos q_2 \dot{q}_2^2 \right]$$

Al realizar los cambios de variable se obtienen las ecuaciones 1.6 a la 1.9.

$$\dot{x}_1 = x_2 \tag{1.6}$$

$$\dot{x}_2 = \frac{1}{m_1 + m_2 sen^2 q_2}\left[m_2 senq_2(l\dot{q}_2^2 - g\cos q_2) + F_1 \right] \tag{1.7}$$

$$\dot{x}_3 = x_4 \tag{1.8}$$

$$\dot{x}_4 = \frac{1}{l(m_1 + m_2 sen^2 q_2)}\left[gsenq_2(m_1 + m_2) - F_1 \cos q_2 - lm_2 senq_2 \dot{q}_2^2 \right] \tag{1.9}$$

Para linealizar el sistema se necesita representarlo de la siguiente manera:

$$\dot{x} = Ax + Bu$$

En donde la expresión anterior puede representarse por medio de:

$$= \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} & \frac{\partial h_1}{\partial x_4} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} & \frac{\partial h_2}{\partial x_4} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} & \frac{\partial h_3}{\partial x_4} \\ \frac{\partial h_4}{\partial x_1} & \frac{\partial h_4}{\partial x_2} & \frac{\partial h_4}{\partial x_3} & \frac{\partial h_4}{\partial x_4} \end{bmatrix} \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial h_1}{\partial f_1} \\ \frac{\partial h_2}{\partial f_1} \\ \frac{\partial h_3}{\partial f_1} \\ \frac{\partial h_4}{\partial f_1} \end{bmatrix}_{C.I.=0,0,0,0.}$$

Al sustituir las condiciones iniciales se obtiene lo siguiente:

$$= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{m_2 g}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g(m_1+m_2)}{lm_1} & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ -\frac{1}{lm_1} \end{bmatrix}$$

Sustituyendo valores se tiene:

$$= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -0.98 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 24.5 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.3125 \\ 0 \\ -0.7102 \end{bmatrix} \qquad (1.10)$$

Por medio de Matlab® se obtienen las ganancias del sistema utilizando el comando ***acker*** proponiendo los polos siguientes:

$$P = \begin{bmatrix} -2+0.5j & -2-0.5j & -1+j & -1-j \end{bmatrix}$$

$$A = \begin{bmatrix} 0,1,0,0; & 0,0,-0.98,0; & 0,0,0,1; & 0,0,24.5,0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0; & 0.3125; & 0; & -0.7102 \end{bmatrix}$$

$$K = acker(A,B,P)$$

$$K = -1.2212 \quad -2.3706 \quad -55.0995 \quad -9.4914$$

1.5 Simulación del sistema en lazo cerrado.

Una vez que se obtienen las ganancias, lo siguiente es generar el código en Matlab® para simular el comportamiento del sistema con retroalimentación de estados suponiendo alguna perturbación en dicho sistema. El código del compilador del sistema retroalimentado es el programa 4 del apéndice A.

Al analizar el código del programa 4 del apéndice se observa la adición del comando ***acker,*** para obtener las ganancias del sistema e indica que la posición angular del péndulo se graficará de forma punteada. También se observa que la matriz A es la ecuación 1.10 desarrollada a partir del Jacobiano del sistema.

El código con los parámetros del sistema se encuentra en el apéndice A como programa 5. A diferencia del código en lazo abierto, se observa en el programa 5, la adición de las ganancias en la línea f1. La gráfica de la figura 1.3 muestra el comportamiento del sistema con retroalimentación de estados utilizando C. I. 0, 0, 0, 0.

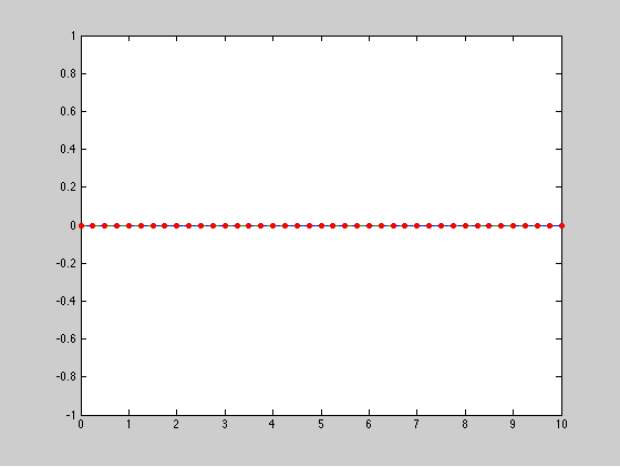

Figura 1.3 Gráfica del sistema en lazo cerrado con C.I. Pos. Carro=0, Vel. Carro=0, Pos. Péndulo=0, Vel. Péndulo=0.

Ahora se agrega una perturbación al sistema, es decir, se inclina un grado al péndulo por lo que el código del compilador se representa como el programa 6 del apéndice A.

La gráfica de la figura 1.4 muestra el comportamiento del péndulo con una inclinación de un grado:

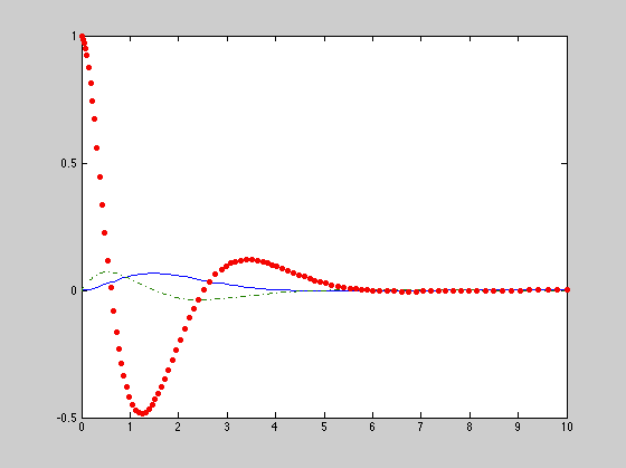

Figura 1.4 Gráfica del sistema en lazo cerrado con C.I. Pos. Carro=0, Vel. Carro=0, Pos. Péndulo=0.01745, Vel. Péndulo=0.

Al realizar pruebas con la simulación del sistema se observa que a partir de los 50 grados el sistema se hace inestable, es decir, el péndulo no logra estabilizarse en la parte más alta. Lo anterior puede mostrarse en la gráfica de la figura 1.5 y el programa 7 del apéndice.

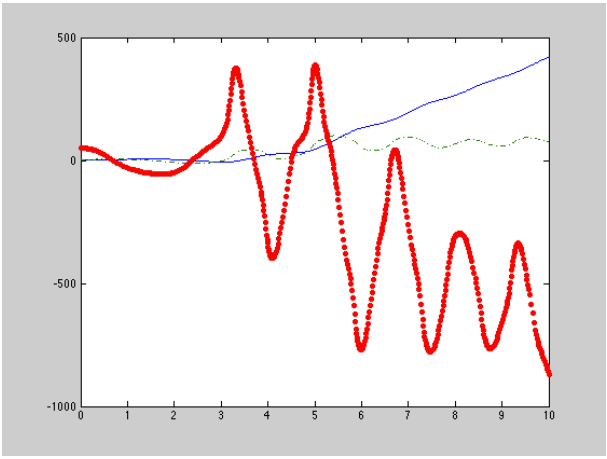

Figura 1.5 Gráfica del sistema en lazo cerrado con C.I. Pos. Carro=0, Vel. Carro=0, Pos. Péndulo=0.8726, Vel. Péndulo=0.

En un sistema real, el sistema es estable con pequeñas perturbaciones u/o inclinaciones las cuales no deben exceder los 10 grados, esto es debido al par que se necesita en el motor para estabilizar el sistema en el punto más alto tomando como una C.I. más grande a los 10 grados de inclinación del péndulo.

1.6 Conclusiones del capítulo.

En este capítulo se desarrolló el modelo matemático del sistema del péndulo invertido para poder generar a partir de dicho modelo, un control del mismo, el cual se realizó de manera exitosa. Una vez realizado dicho control, se obtuvieron las ganancias adecuadas para el sistema, las cuales serán de gran ayuda para los capítulos siguientes en donde se van a generar ganancias aleatoriamente por medio del algoritmo genético continuo (AGC) que se aplicará al sistema a partir de un rango establecido. Dicho rango se basará en las ganancias que se obtuvieron en este capítulo.

Capítulo 2

Algoritmos Genéticos.

2.1 Introducción a los algoritmos genéticos.

Un algoritmo genético o AG es una técnica de programación que sirve para optimizar sistemas, dicho algoritmo aplica los principios de la genética para la resolución de problemas. Los elementos que integran a un algoritmo genético son: individuos y operadores (la selección basada en la población, reproducción o cruce, la mutación, evaluación, etc.). Los AG fueron desarrollados por John Holland en el transcurso de los años 1960 a 1970 en la Universidad de Michigan en Ann Arbor dentro del grupo "logic of computer. Durante ese tiempo, Holland impartió el curso titulado "Teoría de sistemas adaptativos" el cual trataba de imitar los procesos adaptativos de los sistemas naturales. Finalmente el trabajo de Holland fue plasmado en su libro "***Adaptation in Natural and Artificial Systems"*** *y* fue el primero en desarrollar una base teórica para los algoritmos genéticos.

En 1975 De Jong demostró la utilidad de los algoritmos genéticos en su libro "***Analysis of the behavior of a class of genetic adaptive systems"***. Finalmente en 1989 el ingeniero industrial David Goldberg y alumno de Holland popularizo los algoritmos genéticos al solucionar un problema que implicaba el control de transmisión de un gaseoducto utilizando los algoritmos genéticos. En 1989 Goldberg publicó el libro "***Genetic Algorithms in Search, Optimization, and Machine Learning"***. A partir de ese momento se han generado diversos métodos de los AG para un gran número de aplicaciones. Algunas características de estos son:

- La optimización de variables continuas o discretas.
- Maneja un gran número de variables.
- Optimiza variables de costo sumamente complejas.

- Proporciona no solamente una solución única, sino una lista de variables óptimas para el sistema.
- Puede codificar las variables de modo que la optimización sea hecha con las variables codificadas.

Sin embargo, los AG no siempre son el mejor modo de solucionar un problema. Por ejemplo, los métodos tradicionales determinan rápidamente la solución de una función mientras que los AG tardan en analizar la función costo de una población inicial. Para algunos problemas relativamente sencillos se pueden aplicar otros métodos los cuales determinan la solución más rápido que los AG. La figura 2.1 representa un panorama general de las reglas de un AG.

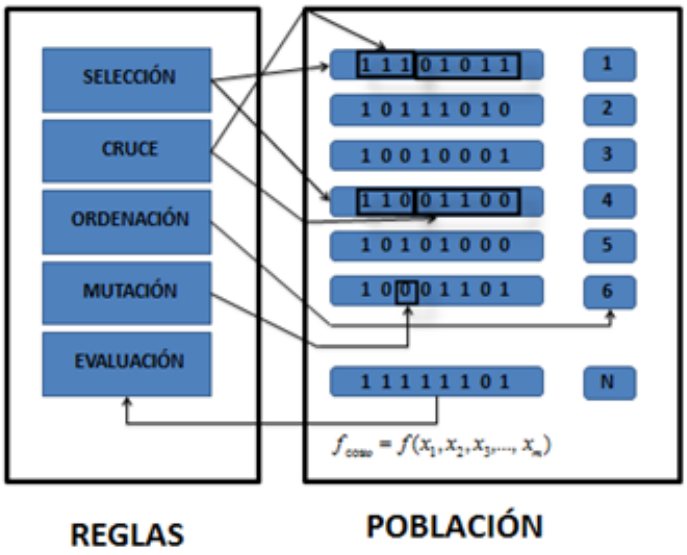

Figura 2.1. Reglas de un AG.

Para entender de mejor manera se observa la figura 2.2. La cual muestra una comparativa entre la evolución biológica y un binario de AG.

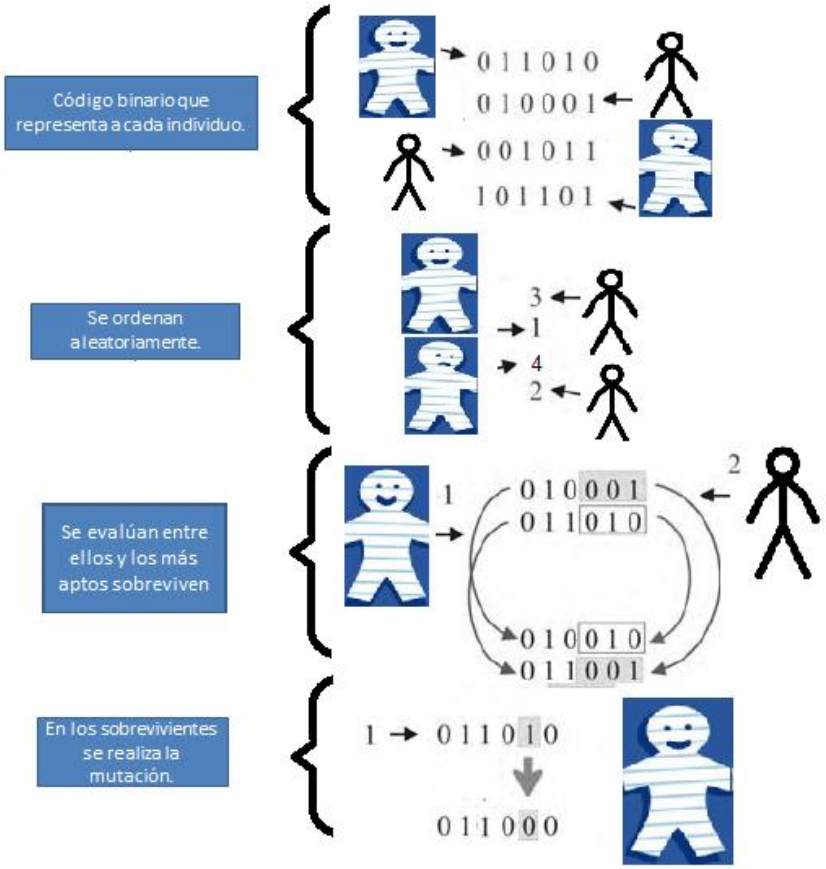

Figura 2.2. Ejemplo de un AG.

Como puede observarse, se tiene a cuatro individuos los cuales también son representados de manera binaria o por cromosomas (es decir, cada uno tiene una combinación binaria de 6 dígitos). Cada fila de números binarios representa las características de cada individuo en la población. Los rasgos específicos de cada individuo son codificados en la secuencia binaria asociada con cada individuo. Si por ejemplo, se desea obtener una nueva raza de individuos altos, se selecciona al individuo más alto para obtener descendencia de este. Por lo que se determina una media de estatura para los individuos más altos, que son los que sobreviven y los más bajos son eliminados.

De esta población se toman dos individuos al azar para crear dos nuevos individuos o hijos. Los nuevos individuos o hijos tienen una alta probabilidad de ser altos porque sus antecedentes o padres tienen esas características. Por lo tanto, las nuevas secuencias binarias (cromosomas) de los individuos

contienen las partes de las secuencias binarias de ambos padres. Estos nuevos individuos sustituyen a los individuos desechados que no eran altos.

2.2 Componentes de un AG.

Los AG al igual que la evolución biológica se compone de diversos elementos, estos elementos son: Los individuos y los operadores; que en este caso son: La selección de individuos, el apareamiento o cruce, la mutación y la evaluación. El diagrama de flujo de la figura 2.3 indica la secuencia que se lleva en un AG. Cada uno de estos operadores tiene funciones específicas a realizar.

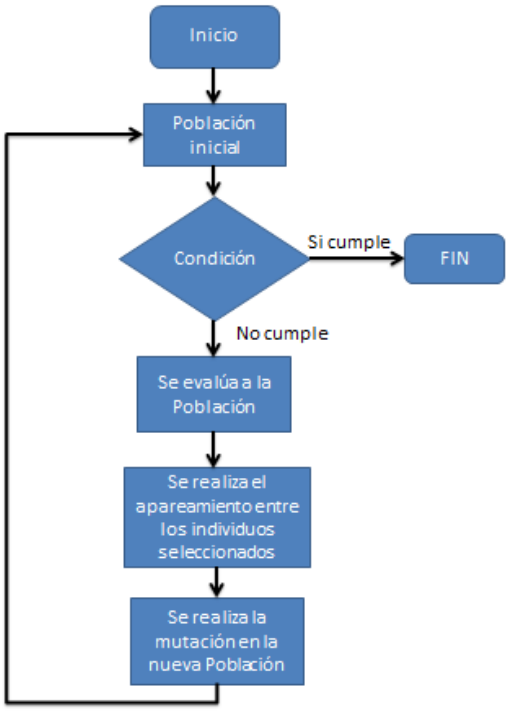

Figura 2.3. Estructura de los AG.

La selección es uno de los operadores más importantes debido a que se encarga de transmitir el código genético de los individuos más aptos a las

generaciones siguientes. Existen diferentes tipos de selección: Torneo, ruleta, elitismo, proporcional, etc. Una vez que se realiza la selección de individuos se ordenan para después pasar al apareamiento.

Una vez que se tienen seleccionados a los individuos, se pasa al operador cruce, el cual se encarga de tomar dos individuos llamados padres y aparearlos para obtener 2 nuevos individuos llamados hijos. Estos hijos se obtienen dividiendo de manera aleatoria en dos partes a cada individuo llamado padre y uniendo una de las dos partes de cada individuo padre con la de otro individuo padre, esto se observa en la figura 2.4. El punto de cruce (que es el punto en donde se divide a cada individuo) se genera de manera aleatoria entre 1 y $m-1$, donde m es la longitud del cromosoma (el tamaño de cada individuo).

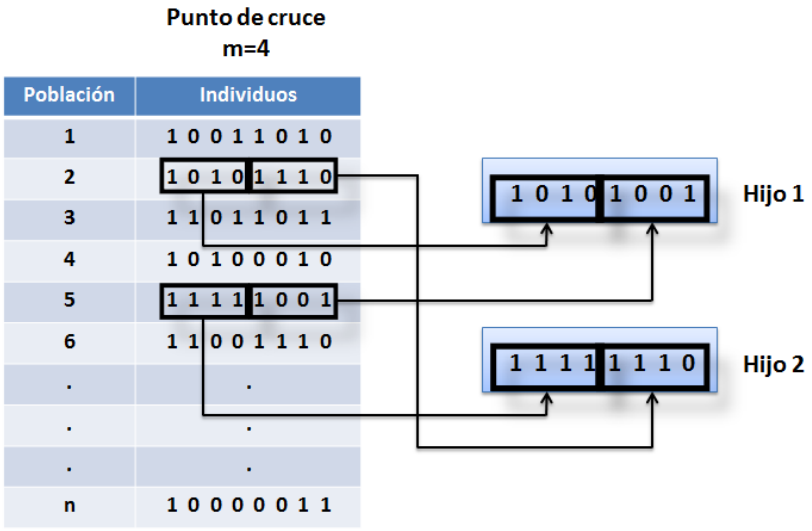

Figura 2.4. Obtención de dos hijos por medio del punto de cruce.

Una vez que se obtienen nuevos individuos a partir del apareamiento (unir partes de un individuo con las partes de otro individuo) lo que sigue es alterar alguna de las partes del individuo. Es decir, si un individuo es representado por la cantidad binaria 1 0 1 **0** 1 0 0 1 y se realiza la mutación en la posición 5, lo que se alteró o modificó es dicha posición. Lo que da como resultado 1 0 1 **1** 1 0 0 1 (como se observa, dicha posición fue alterada).

El elitismo se refiere a la sobrevivencia de los mejores individuos, desechando o eliminando al resto. En esta parte del proceso, se genera el mismo número de individuos que los que se eliminan.

En la parte de la evaluación, lo que se realiza es analizar si la última generación de individuos logro llegar o acercarse al objetivo establecido.

Como se puede observar en la figura 2.1 la evolución ocurre en base a los cromosomas al igual que el apareamiento y la mutación. Una de las ventajas de los AG es su optimización global, es decir, exploran globalmente áreas desconocidas dentro del espacio de búsqueda y explotan el conocimiento obtenido para obtener un resultado bastante aceptable. Otra de sus ventajas es que son independientes a cualquier problema, esto los hace un algoritmo robusto.

Dentro de los AG se encuentran los algoritmos genéticos binarios (AGB) y los algoritmos genéticos continuos (AGC).

2.3 Algoritmos Genéticos Binarios (AGB).

En los algoritmos genéticos binarios o AGB se comienza definiendo las variables de optimización con la función costo y se finaliza realizando la convergencia hacia el valor óptimo deseado. Una de las partes importantes en este tipo de algoritmo es la codificación y decodificación de las variables, es decir, convertir los valores continuos a valores binarios, para realizar los procedimientos pertinentes en valores binarios y una vez realizados convertir dichos valores de nueva cuenta a valores continuos.

Función costo

La función costo es una de las partes importantes dentro del código que representa al AGB, esto debido a que genera una salida con un número determinado de variables de entrada (un cromosoma). La función costo puede ser una función matemática, un juego, un experimento, etc. El objetivo es

modificar la salida de alguna manera para encontrar los valores apropiados para las variables de entrada. Por ejemplo, cuando se va a llenar un recipiente con agua tibia, el costo es la diferencia entre las temperaturas deseada y real del agua, las variables de entrada son las llaves de agua caliente y fría; En este caso la función costo es el resultado experimental de poner la mano en el agua. Lo que determina que nuestra función costo sea adecuada es seleccionar de manera correcta las variables que intervienen en dicha función.

Por lo mencionado anteriormente, el AG comienza por definir un cromosoma; si este tiene n variables por lo tanto dicho cromosoma se escribe como un vector de una fila de n elementos, esto se muestra en la ecuación 2.1.

$$Cromosoma = [p_1, p_2, p_3 ... p_n] \tag{2.1}$$

Si cada variable se convierte a sistema binario se tiene lo siguiente:

$$Cromosoma = \left[\underbrace{11100010}_{gen_1} \underbrace{10100001}_{gen_2} .... \underbrace{00011101}_{gen_n} \right]$$

En donde la función costo puede ser definida como se muestra en la ecuación 2.2:

$$FunciónCosto = f(cromosoma) = f(p_1, p_2, p_3 ... p_n) \tag{2.2}$$

Cuando se determina la función costo debe analizarse con especial cuidado, esto, debido a que puede complicarse por un número excesivo de variables. Por ejemplo, si se quisiera optimizar el ahorro de combustible de un automóvil se necesitarían variables como el tamaño del auto, el peso, el tamaño y tipo de motor. Pero el color del automóvil, los tipos de luces o el tipo de volante, son

variables con poco o ningún impacto acerca del consumo de gasolina del automóvil y no deben ser incluidas en dicha función. Por lo regular el número correcto de variables generan mejores resultados. Por lo general en los problemas de optimización se requiere de un número límite de variables.

Codificación y decodificación de variables

Los individuos generados en la población son representados por una serie de números binarios, los cuales son convertidos primero de decimal a binario y después convertidos de binario a decimal (valores continuos). A este proceso de conversión de decimal a binario y viceversa se le llama codificación y decodificación. Las variables representadas en sistema binario son convertidas en valores continuos. Entre más bits contenga cada individuo el resultado será más cercano al buscado.

Para la codificación de variables se utilizan las ecuaciones 2.3 y 2.4.

$$P_{norm} = \frac{P_n - P_{lo}}{P_{hi} - P_{lo}} \tag{2.3}$$

$$gene[m] = round\left\{ P_{norm} - 2^{-m} - \sum_{P=1}^{m-1} gene[p]2^{-p} \right\} \tag{2.4}$$

Para la decodificación se utilizan las ecuaciones 2.5 y 2.6.

$$P_{quant} = \sum_{m=1}^{N_{gene}} gene[m]2^{-m} + 2^{-(M+1)} \tag{2.5}$$

$$q_n = P_{quant}(P_{hi} - P_{lo}) + P_{lo} \tag{2.6}$$

Dónde:

P_{norm} = Variable normalizada $0 \le P_{norm} \le 1$.

P_n = Variable *n*.

P_{lo} = Valor mínimo de la variable.

P_{hi} = Valor máximo de la variable.

$gene[m]$ = Forma binaria de P_n.

$round\{\bullet\}$ = Redondea al valor entero más cercano.

P_{quant} = Forma cuantificada de P_{norm}.

q_n = Forma cuantificada de P_n.

Se tiene como ejemplo que la primera variable del cromosoma P_{norm} tiene límites entre $-5 \le P_{norm} \le 5$ y un gen de 4 bits que genera $2^{N_{gen}} = 16$ valores entre -5 y 5. La tabla 2.1 muestra los valores en binario y su ponderación.

Valor ponderado	Valor binario	Número de divisiones
-5	0000	0
-4.333	0001	1
-3.666	0010	2
-3	0011	3
-2.333	0100	4
-1.666	0101	5
-1	0110	6
-0.333	0111	7
0.333	1000	8

1	1001	9
1.666	1010	10
2.333	1011	11
3	1100	12
3.666	1101	13
4.333	1110	14
5	1111	15

Tabla .2 1 Decodificación del parámetro P_{norm}.

Para decodificar y cuantificar el parámetro P_{norm} se utilizan los siguientes comandos:

$$P_{quant} = bin2dec(num2str(gen[m]));$$

$$q_n = P_{quant} * ((P_{hi} - P_{lo})/(2^{N_{gen}} - 1)) + P_{lo};$$

Para los demás parámetros se obtiene un total de bits para el cromosoma de:

$$N_{bits} = N_{gen} - N_{var}$$

Dónde:

N_{bits} = Número de bits del cromosoma.
N_{gen} = Número de bits del gen.
N_{var} = Número de variables.

Población

El AG comienza con un grupo de cromosomas llamados población. La población se conforma de un número de cromosomas N_{pop} y es una matriz $N_{pop} \times N_{bits}$ la cual se llena aleatoriamente de ceros y unos; el comando para generar dicha matriz es:

$$pop = round(rand(N_{pop}, N_{bits}));$$

Dónde:

N_{pop} = Tamaño de la población.

La expresión anterior redondea los números al entero más cercano, que en este caso es 0 o 1.

Selección

La selección se basa en la selección natural de los individuos más aptos y con mejores cualidades para un objetivo específico. El decir, los más aptos se traduce en desechar a aquellos individuos que no contienen las cualidades deseadas, es decir, sólo lo mejor es seleccionado para seguir, mientras el resto es eliminado. Aquí se ordenan del mejor cromosoma al peor, para seleccionar a los más adecuados mientras los demás son desechados y sustituidos por la nueva generación. El número de cromosomas que son aptos para cada iteración está dada por:

$$N_{keep} = X_{rate} N_{pop}$$

Dónde:

N_{keep} = Número de población sobreviviente para aparearse.

X_{rate} = Tasa de selección.

N_{pop} = Número de la población.

Apareamiento

Una vez que se seleccionaron a los individuos más aptos, se procede al apareamiento o cruce; En este proceso exixten diferentes métodos de apareamiento, se analizarán algunos de ellos.

Apareamiento de arriba abajo.- En este procedimiento lo que se realiza es seleccionar el cromosoma más alto, es decir todos los números impares en fila serán los padres, mientras las filas pares serán las madres; por lo tanto, el apareamiento inicia con el padre 1 y la madre 1 (es decir, la fila 1 con la fila 2). Este tipo de cruce no modela bien la naturaleza del cruce, pero es una técnica aceptable para principiantes.

Apareamiento ponderado o ruleta.- En este tipo de apareamiento o cruce, los cromosomas son acoplados inversamente proporcional a su estatus, es decir, el cromosoma mejor ponderado tiene una probabilidad muy baja de acoplamiento, por el contrario, el cromosoma peor ponderado tiene una alta probabilidad de acoplamiento. Aquí un número aleatorio determina que cromosoma es seleccionado.

Apareamiento aleatorio.- Aquí se genera aleatoriamente el número de padres y madres, lo cual se logra con los comandos siguientes:

$$ma = ceil(N_{keep} * rand(1, N_{kepp}));$$

$$pa = ceil(N_{keep} * rand(1, N_{kepp}));$$

Hijos

Estos son el resultado del cruce de los padres (cromosomas) seleccionados de N_{keep} en donde los individuos seleccionados se ordenaron de acuerdo a la función costo. Para generar a los hijos, se seleccionan a los padres y también se establece el punto de cruce, el cual es fundamental en la creación de los hijos. Una vez que se establece el punto de cruce, de dos padres se obtienen dos hijos. Es decir, el Papá 1 y la Mamá 1 son divididos en dos partes (por lo general se dividen en dos partes pero no siempre es así, en algunas ocasiones se pueden partir en 3, es decir, pueden tener 2 puntos de cruce), el punto de cruce determina en que parte del código de los padres serán divididos. Una vez establecido el punto de cruce, el Hijo 1 se genera de la primera parte del Papá 1 y la segunda parte de la Mamá 1. Si por ejemplo, el punto de cruce es el 3 y

el tamaño del cromosoma es de 8 bits, el Hijo 1 se compone del 1er al 3er bit del Papá 1 y del 4to bit 8vo se compone del código de la mamá 1. La Figura 2.5 ejemplifica lo antes explicado.

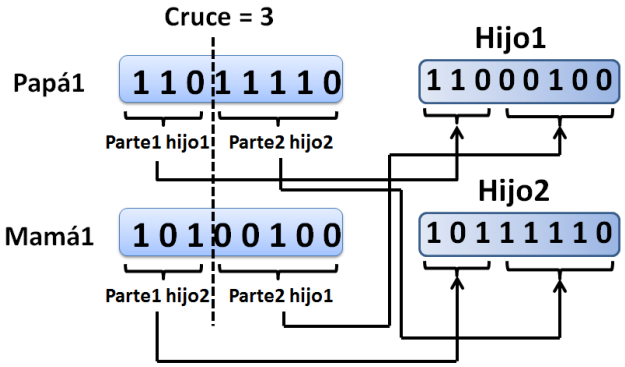

Figura 2.5 Cruce de padres para generar hijos.

Este procedimiento se realiza para los padres restantes. Una vez que se obtienen todos los hijos, se procede a realizar la mutación.

Mutación.

La mutación cambia el valor de un porcentaje de bits de la población, es decir, el bit que se elige cambia su valor, si es cero se cambia a uno y si es uno se cambia a cero, esto se realiza para evitar que la población caiga en un óptimo local. El único cromosoma que no es afectado por la mutación es el mejor ponderado de la población. El número de mutaciones en la población se determina con el siguiente comando:

$$num_{mut} = \mu ceil(N_{pop} - 1)N_{bits};$$

Dónde:

num_{mut} = Número de mutaciones.

μ = Porcentaje de mutación.

N_{pop} = Número de población.

N_{bits} = Número de bits del cromosoma.

Una vez terminada la mutación se realiza la evaluación para cada individuo con respecto a la función costo y se repite todo el procedimiento anterior hasta terminar el número de iteraciones establecida.

2.4 Algoritmos Genéticos Continuos (AGC).

Si se requiere de una mayor exactitud en las variables, menor tiempo de ejecución y menor consumo de memoria, se considera una mejor opción a los AGC ya que al utilizar valores de punto flotante en lugar del sistema binario no necesita codificar y decodificar lo que lo hace más rápido y más exacto. Ambos sistemas (AGB como los AGC) son muy similares en cuanto a su procedimiento solo cambian algunas partes que se mencionan a continuación.

Población.

El paso es similar al de los AGB, la población se conforma de un número de cromosomas N_{pop}, la matriz $N_{pop} \times N_{bits}$ se llena aleatoriamente de números continuos utilizando el comando siguiente:

$$P_{norm} = rand(N_{pop}, N_{var});$$

Para que cada cromosoma sea evaluado con la función costo son normalizados entre 0 y 1, para esto se utiliza el comando siguiente:

$$P = (P_{hi} - P_{lo})P_{norm} + P_{lo};$$

Dónde:

P_{hi} = Valor máximo del parámetro.

P_{lo} = Valor mínimo del parámetro.

P_{norm} = Valor normalizado del parámetro.

Hijos

El procedimiento de generar hijos es el mismo que el de los AGB la diferencia radica en la aplicación de fórmulas para obtenerlos, es decir, se selecciona a la mitad más apta mientras la otra mitad es desechada. La mitad sobreviviente es la que se utiliza para el apareamiento. Aquí en el apareamiento es en donde se puede aplicar diferentes métodos para generar hijos, como el cruce uniforme, métodos de mezcla, heurística de cruce entre otros. Se explicara él heurística de cruce.

En el método heurística de cruce se introduce aleatoriamente un número entre 0 y 1 llamado β para generar a los hijos por medio de los siguientes comandos:

$$P_{new1} = P_{mn} + \beta(P_{mn} - P_{pn})$$

$$P_{new2} = P_{mn} + \beta(P_{mn} - P_{pn})$$

Dónde:

P_{new1} = Primer hijo.

P_{new2} = Segundo hijo.

P_{mn} = Valor continuo de la madre.

P_{pn} = Valor continuo del padre.

β = Número aleatorio entre 0 y 1.

Cabe mencionar que si por alguna razón, alguno de los hijos sobrepasa el valor del intervalo máximo lo que se realiza es restar la parte $\beta(P_{mn} - P_{pn})$ para no exceder dicho intervalo.

Mutación

Al igual que en la mutación en los AGB, se realiza prácticamente lo mismo, utilizando

$$Num_{mut} = \mu N_{pop} N_{var}$$

Dónde:

Num_{mut} = Número de mutaciones.

μ = Tasa de mutación.

N_{pop} = Número de población.

N_{var} = Número de variables.

Al igual que en los AGB, después de realizar la mutación, se sigue con la repetición del procedimiento el número de iteraciones establecidas o hasta la obtención del valor deseado.

2.5 Conclusiones del capítulo.

En este capítulo se dio una explicación acerca de la estructura y funcionamiento de los AG, sus aplicaciones, fortalezas y debilidades; así como también se explicaron los comandos que utilizan y la diferencia entre un AGB y un AGC. Todo esto para entender cuál de los 2 AG es la mejor opción para implementarse en el código que representa el control del sistema del péndulo invertido.

En el capítulo siguiente (capítulo 3) se realizará la optimización del control de retroalimentación de estados implementando AG, es decir, se desea generar los parámetros adecuados utilizando AG para controlar el sistema.

Capítulo 3

Optimización del control de retroalimentación de estados utilizando AG.

3.1 Introducción.

En los capítulos anteriores se desarrolló el modelo matemático del péndulo invertido, el código en lazo abierto y en lazo cerrado para representarlo. Lo que se logró con todo lo anterior es linealizar el sistema, ya que para estabilizarlo se debe llegar a su modelo matemático en forma lineal.

Ahora, en este capítulo se acopla el AGC al modelo matemático linealizado y retroalimentado para poder obtener las ganancias adecuadas para estabilizar a dicho sistema.

El sistema se maneja bajo los rangos $-\frac{\pi}{18} \leq x_3 \leq \frac{\pi}{18}$. Debido a que el sistema trabajará entre $\pm 10^{\circ}$ de inclinación del péndulo.

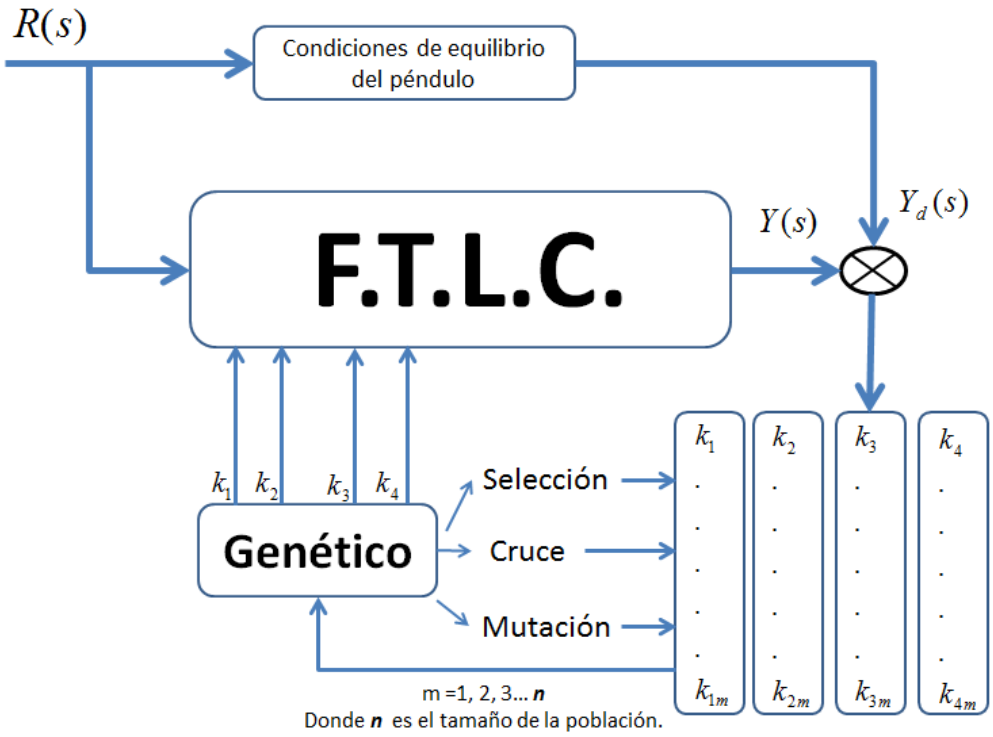

Figura 3.1 Diagrama de bloques que representa al sistema.

Como se observa en la figura 3.1 el AGC genera 4 ganancias aleatoriamente para estabilizar al sistema. Dichas ganancias se generan a partir de un rango establecido en cada una de ellas, es decir, con las ganancias obtenidas en el primer capítulo, se establecen los rangos para cada ganancia.

Como se analizó en el primer capítulo, el péndulo invertido es un sistema no lineal por lo que al llegar a la ecuación general de un robot se despeja el término más alto como se muestra en la siguiente expresión:

$$\ddot{q} = \begin{bmatrix} \dfrac{1}{m_1 + m_2 sen^2 q_2}\left[m_2 sen q_2 (l \dot{q}_2^2 - g\cos q_2) + F_1 \right] \\ \dfrac{1}{l(m_1 + m_2 sen^2 q_2)}\left[g sen q_2 (m_1 + m_2) - F_1 \cos q_2 - l m_2 sen q_2 \cos q_2 \, \dot{q}_2^2 \right] \end{bmatrix}$$

A partir de aquí se necesita linealizar el sistema y pasarlo a la forma $\dot{x} = Ax + Bu$ para obtener las ganancias adecuadas y estabilizar al sistema. Por lo que se necesita realizar un cambio de variable, el cual está representado en las ecuaciones 1.6 a la 1.9 del primer capítulo.

Una vez realizado el cambio de variable, lo siguiente es linealizar con la siguiente expresión:

$$= \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} & \frac{\partial h_1}{\partial x_4} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} & \frac{\partial h_2}{\partial x_4} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} & \frac{\partial h_3}{\partial x_4} \\ \frac{\partial h_4}{\partial x_1} & \frac{\partial h_4}{\partial x_2} & \frac{\partial h_4}{\partial x_3} & \frac{\partial h_4}{\partial x_4} \end{bmatrix} \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} \frac{\partial h_1}{\partial f_1} \\ \frac{\partial h_2}{\partial f_1} \\ \frac{\partial h_3}{\partial f_1} \\ \frac{\partial h_4}{\partial f_1} \end{bmatrix}_{C.I.=0,0,0,0.}$$

Al sustituir con los parámetros establecidos anteriormente se obtiene la ecuación 1.10.

$$= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -0.98 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 24.5 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.3125 \\ 0 \\ -0.7102 \end{bmatrix} \qquad (1.10)$$

Una vez que se llega a esta parte, se proponen los polos siguientes para obtener las ganancias adecuadas del sistema:

$$P = \begin{bmatrix} -2 + 0.5j & -2 - 0.5j & -1 + j & -1 - j \end{bmatrix}$$

Con los polos propuestos se obtienen las ganancias que se encontraron en el capítulo 1, las cuales son:

$$K = -1.2212 \quad -2.3706 \quad -55.0995 \quad -9.4914$$

Una vez que se obtiene lo mencionado anteriormente, el siguiente paso es realizar el acoplamiento de los algoritmos genéticos en la retroalimentación de estados, para que dicho algoritmo determine las mejores ganancias del sistema. El objetivo de este trabajo es realizar una metodología utilizando Algoritmos Genéticos para determinar el óptimo en la retroalimentación de estados.

3.2 Código del AG.

La función del código generado en AG es la de generar un determinado número de ganancias en un intervalo establecido y así poder determinar las mejores ganancias para el sistema. El programa 8 del apéndice B muestra el código del AGC a utilizar.

Dicho programa se encuentra comentado para realizar una breve explicación en cada una de las líneas. En las líneas iniciales se establecen los parámetros como el número de individuos, el número de evoluciones, el número de ganancias, los rangos mínimos y máximos de las ganancias. Después se genera aleatoriamente una matriz para la población de tamaño $N \times O$ (número de individuos por número de ganancias), dicha población se escala en los rangos máximos y mínimos de los parámetros establecidos para posteriormente generar las ganancias con respecto a la función costo el número determinado de evoluciones, las cuales se grafican. Una vez graficadas, cada uno de los individuos se ordenan y se eliminan entre ellos; Una vez que se tienen a los individuos sobrevivientes se genera una nueva generación de individuos a partir de los que sobrevivieron para sustituir a los individuos eliminados.

Con la nueva generación se realiza el mismo procedimiento hasta que se termina el número de evoluciones. Con lo cual, se obtienen las ganancias mejor ponderadas y su gráfica correspondiente.
El código descrito con anterioridad nombrado ***"agprincipal"*** llama a tres programas más, los cuales son ***"parametros"***, ***"cruce"*** y ***"mutación"***. Estos códigos se encuentran en el apéndice como programa 9, programa 10 y programa 11 respectivamente.

En el programa llamado ***"parámetros.m*** localizado en el apéndice se generan los parámetros del sistema físico y el modelo matemático ubicado por partes en las posiciones correspondientes para posteriormente ser llamado desde el código principal "***agprincipal"*** y ejecutarse en la sentencia "**ode45"**.

En el programa del apéndice llamado ***"cruce.m"*** se realiza el apareamiento de individuos, es decir, por cada dos individuos se crean dos hijos; dicho procedimiento de generar nuevos individuos se realiza dentro del ciclo "***for"***.
Dicho código es llamado por el programa principal ***"agprincipal"*** en la sentencia ***"P_Nueva=Cruce"***.

El programa de mutación se realiza para no caer en una solución local. El código agprincipal llama al código Mutación en la sentencia ***"P_Nueva = Mutacion"***.

3.3 Implementación del Código de retroalimentación de estados utilizando AGC.

Una vez que se tienen los códigos de Algoritmos Genéticos para la retroalimentación de estados del sistema, lo siguiente es ejecutarlos para verificar su comportamiento. Para correr el sistema se utilizan los parámetros de la tabla 3.1. Las ganancias son las obtenidas en el capítulo 1; Estas son:

$$K = -1.2212 \quad -2.3706 \quad -55.0995 \quad -9.4914$$

Ganancias obtenidas			
K1	**K2**	**K3**	**K4**
-1.2212	-2.3706	-55.0995	-9.4914
Parámetros propuestos			
Max. Min. K1	**Max. Min. K2**	**Max. Min. K3**	**Max. Min. K4**
0 a -2.5	-1 a -4	-60 a -50	-5 a -10
Núm. Evo.	**Núm. Individuos**	**T. Simulación**	**Núm. Ganancias**
10	20	10	4

Tabla 3.1 Parámetros de la simulación con 20 individuos.

Para conocer el comportamiento del algoritmo genético se realizó la simulación 20 veces (esto es debido a que con 20 simulaciones son suficientes para saber si el rango de ganancias es adecuado o no ya que no genera dudas como las simulaciones cortas y evita pérdida de tiempo por exceso de las mismas) con los parámetros anteriores. En las figuras 3.2 y 3.3 se muestran las 4 mejores corridas de dicha prueba.

Simulación 1. Simulación 2.

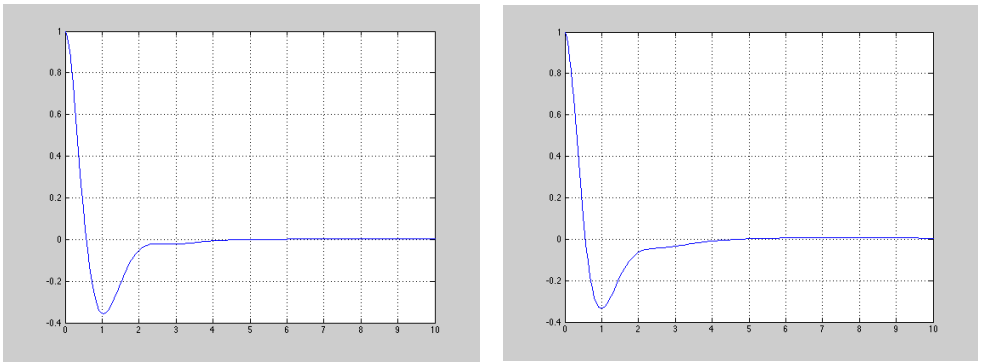

Figura 3.2 Gráficas que representan el comportamiento del péndulo en las simulaciones 1 y 2.

Simulación 5. Simulación 14.

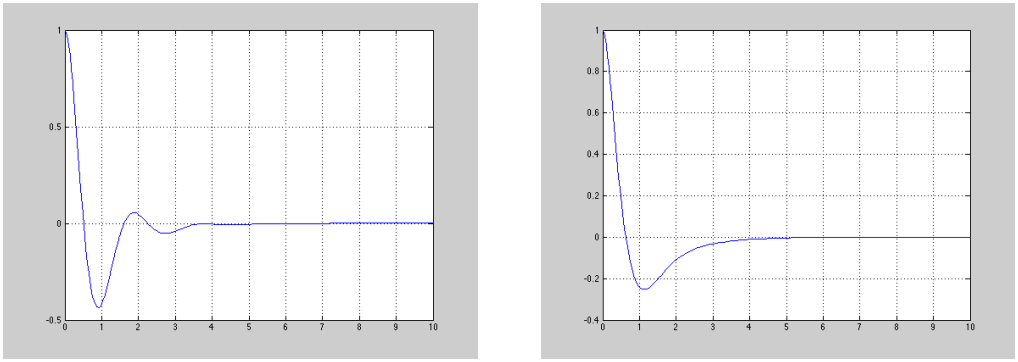

Figura 3.3 Gráficas que representan el comportamiento del péndulo en las simulaciones 5 y 14.

Al correr 20 veces la simulación con los parámetros establecidos, se obtienen diferentes gráficas, las cuales indican que el sistema logra estabilizarse con las ganancias que el programa del Algoritmo Genético proporciona al sistema, es decir, dichas ganancias se encuentran dentro de los rangos máximo y mínimo establecidos. Las ganancias obtenidas en las 20 simulaciones se muestran en la tabla 3.2.

Valores de ganancias en las 20 simulaciones.				
Núm. simulación.	K1	K2	K3	K4
1	**-0.1380**	**-1.7383**	**-55.5357**	**-9.1120**
2	**-0.2640**	**-1.4943**	**-57.9433**	**-7.8376**
3	-0.2546	-1.1018	-57.6702	-9.7201
4	-0.0893	-1.0121	-59.9970	-9.8085
5	**-0.0147**	**-1.4120**	**-56.3069**	**-8.1308**
6	-0.0753	-1.0703	-55.5174	-9.1038
7	-0.4141	-1.6184	-59.5618	-9.7832
8	-0.3614	-1.0017	-59.9268	-9.9519
9	-0.0711	-1.1258	-59.8798	-9.7456
10	-0.0021	-1.0865	-57.8614	-7.2425
11	-0.0032	-1.2842	-55.8349	-8.2876
12	-0.2909	-1.0658	-59.8430	-8.3993
13	-0.0040	-1.2794	-59.0338	-8.8186
14	**-0.0009**	**-1.2195**	**-59.3745**	**-8.7421**
15	-0.0870	-1.0001	-57.3885	-8.1798
16	-0.1751	-1.0591	-58.3819	-8.0737
17	-0.2437	-1.6115	-58.7089	-9.6381
18	-0.0120	-1.1231	-57.8958	-8.5402
19	-.10969	-.1.0096	-58.5506	-9.0351
20	-0.4682	-1.0882	-59.4614	-8.4396

Tabla 3.2 Ganancias de la simulación con 20 individuos.

Para observar el comportamiento del AGC se realiza la misma simulación pero ahora cambiando la cantidad de individuos de la población de 20 a 500.

Ganancias obtenidas			
K1	**K2**	**K3**	**K4**
-1.2212	-2.3706	-55.0995	-9.4914
Parámetros propuestos			
Max. Min. K1	**Max. Min. K2**	**Max. Min. K3**	**Max. Min. K4**
0 a -2.5	-1 a -4	-60 a -50	-5 a -10
Núm. Evo.	**Núm. Individuos**	**T. Simulación**	**Núm. Ganancias**
10	500	10	4

Tabla 3.3 Parámetros de la simulación con 500 individuos.

Para conocer el comportamiento del algoritmo genético se realizó la simulación 20 veces con los parámetros anteriores. En las figuras 3.4 y 3.5 se muestran las 4 mejores corridas de dicha prueba.

Simulación 1. Simulación 2.

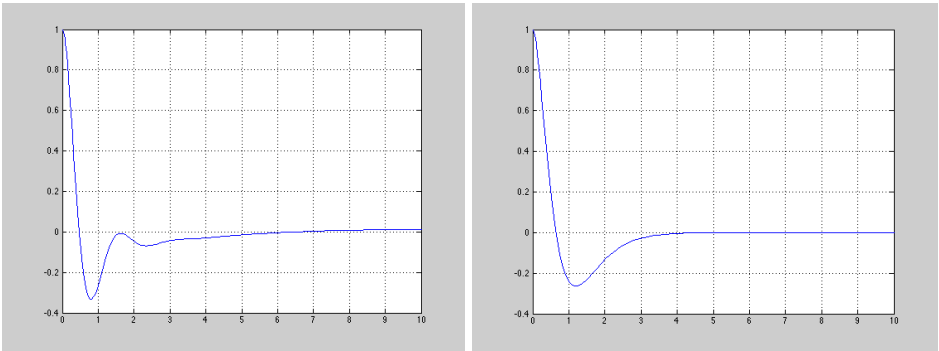

Figura 3.4 Gráficas que representan el comportamiento del péndulo en las simulaciones 1 y 2.

Simulación 3. Simulación 9.

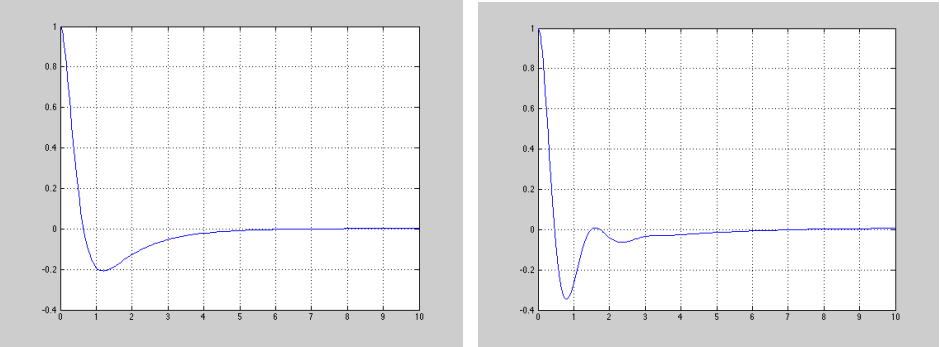

Figura 3.5 Gráficas que representan el comportamiento del péndulo en las simulaciones 3 y 9.

Al correr 20 veces la simulación con los parámetros establecidos, se obtienen diferentes gráficas, las cuales indican que el sistema logra estabilizarse con las ganancias que el programa del Algoritmo Genético proporciona al sistema, es decir, dichas ganancias se encuentran dentro de los rangos máximo y mínimo establecidos. Las ganancias obtenidas en las 20 simulaciones se muestran en la tabla 3.4.

Valores de ganancias en las 20 simulaciones.				
Núm. simulación.	K1	K2	K3	K4
1	**-0.0676**	**-1.1282**	**-58.8328**	**-6.2282**
2	**-0.0315**	**-1.0127**	**-59.6405**	**-6.1861**
3	**-0.0083**	**-1.0020**	**-59.2006**	**-9.8521**
4	-0.0079	-1.1162	-59.5186	-9.4794
5	-0.2222	-1.2171	-59.0581	-6.5711
6	-0.0656	-1.2615	-63.1884	-10.4332
7	-0.0974	-1.3409	-62.8093	-11.0045
8	-0.0069	-1.1307	-63.3364	-11.1000
9	**-0.0485**	**-1.2626**	**-59.6407**	**-6.4102**
10	-0.0029	-1.3495	-60.2825	-6.3786

11	-0.0126	-1.0604	-58.6400	-5.9183
12	-0.0092	-1.0582	-58.6956	-9.9980
13	-0.0440	-1.2947	-63.5507	-11.1603
14	-0.0160	-1.0216	-61.1422	-6.0627
15	-.0005	-1.2456	-59.7947	-10.0259
16	-0.2177	-1.0497	-58.9875	-6.3362
17	-0.0548	-1.1868	-62.6671	-10.8965
18	-0.0988	-1.2356	-60.3056	-7.4289
19	-0.0045	-1.1971	-57.6296	-6.7936
20	-0.0789	-1.0134	-59.7945	-5.8643

Tabla 3.4 Ganancias de la simulación con 500 individuos.

Al analizar las ganancias de las simulaciones, se observa que todas se encuentran dentro del rango establecido y pueden estabilizar al sistema inclusive con una perturbación de 10° de inclinación en el péndulo.

3.4 Conclusiones del capítulo.

En el capítulo 3 se implementó el código del AGC a un sistema de control de retroalimentación de estados que representa al péndulo invertido. Dicho código puede generar una serie de ganancias las cuales se encontrarán siempre dentro de un rango establecido. Al igual que el rango de las ganancias, la población puede establecerse. Al analizar las corridas se observa que todas las simulaciones generan estabilidad al sistema y también que las corridas con un mayor número de individuos en la población genera una mejor estabilidad a dicho sistema.

Capítulo 4

Pruebas y resultados

Como se menciona anteriormente el objetivo de este trabajo es desarrollar una estrategia para determinar las ganancias de un sistema utilizando los algoritmos genéticos continuos; Para desarrollarla se utiliza como ejemplo el sistema del péndulo invertido con una inclinación en el péndulo de 1°, del cual ya se tienen las ganancias definidas. La ventaja de esta estrategia es que no es necesario conocer los parámetros del sistema.

Como se menciona en el capítulo anterior, utilizando los algoritmos genéticos no es necesario linealizar el sistema para determinar sus ganancias. Se puede comenzar proponiendo un rango bastante amplio en todos los rangos de las ganancias y dependiendo de los resultados se podrá determinar si los rangos se acotan o no. Por otra parte, también se propone correr la simulación por lo menos 20 veces (en el capítulo anterior se explica la razón del número de corridas) para obtener un número amplio de datos.

4.1 Prueba con un rango de ganancias de 0 a -100.

En el capítulo 3, se analizó el comportamiento de un genético con el sistema dinámico que representa al péndulo invertido, dicho análisis inicio con un rango amplio en las ganancias; en esta prueba se propone comenzar de la misma manera, iniciar con un amplio rango en las ganancias, entre 0 y -100, esto porque como se analizó anteriormente, se parte de un rango muy amplio

en dichas ganancias y dependiendo de los resultados se decide hacia donde se dirigen dichos rangos y también si estos se reducen o no. Al igual que los rangos, se manejaron diferentes tamaños de población o individuos en las simulaciones anteriores, lo cual se propone aplicar en esta prueba, comenzando con 20 individuos, lo cual es un número relativamente pequeño. Los parámetros para las primeras 20 corridas se muestran en la tabla 4.1.

Población: 20 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-100 a 0	-100 a 0	-100 a 0	-100 a 0
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.1 Parámetros de la primera simulación.

Con los datos anteriores no se obtienen datos concretos para generar las ganancias que estabilicen al sistema. El siguiente paso es aumentar la población. Primero a 100 individuos y posteriormente a 1000 para tener más opciones de exploración.

Población: 100 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-100 a 0	-100 a 0	-100 a 0	-100 a 0
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.2 Parámetros de la segunda simulación.

Población: 1000 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-100 a 0	-100 a 0	-100 a 0	-100 a 0
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.3 Parámetros de la tercera simulación.

Con los parámetros anteriores se observa que el sistema no converge, esto es porque los parámetros establecidos se encuentran muy abiertos. Lo siguiente es acotar los rangos de las ganancias.

4.2 Prueba con un rango de ganancias de 0 a -50.

La siguiente simulación se genera con los rangos de las ganancias reducidos de 0 a -50 para determinar las ganancias adecuadas para determinar en cual región o regiones se localizan dichas ganancias que logren estabilizar el sistema. Los parámetros completos se muestran en la tabla 4.4.

Población: 20 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-50 a 0	-50 a 0	-50 a 0	-50 a 0
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.4 Parámetros de la cuarta simulación.

Corrida	Comportamiento del sistema	Ganancias			
		K1	K2	K3	K4
1	Inestable.	-4.0939	-30.7690	-33.1839	-37.4656
2	Inestable.	-5.8581	-1.5361	-49.4819	-38.6690
3	Inestable.	-1.5979	-2.6513	-47.2367	-20.3667
4	Inestable.	-1.9046	-8.5653	-38.8199	-38.1509
5	Inestable.	-0.3777	-12.4659	-48.0261	-27.3913
6	Inestable.	-3.2208	-44.9764	-49.2807	-47.2447
7	Inestable.	-8.5496	-1.9613	-20.9130	-26.4254
8	Inestable.	-1.441	-34.9841	-49.9974	-43.4035
9	Inestable.	-7.8872	-19.9373	-43.1829	-37.8415
10	Inestable.	-0.2196	-8.7687	-43.7947	-16.4810
11	Inestable.	-3.3953	-2.3996	-49.3654	-12.1981
12	Inestable.	-0.7176	-2.4414	-48.7321	-18.2438
13	Inestable.	-6.2897	-17.3670	-49.6747	-27.2952
14	Inestable.	-7.0922	-0.9461	-30.7506	-26.9319
15	Inestable.	-7.3593	-5.0294	-34.6681	-43.4166
16	Inestable.	-1.7219	-5.2806	-34.9564	-26.0983
17	Inestable.	-3.9069	-22.5793	-35.7415	-25.7019
18	Inestable.	-5.3778	-39.4797	-47.9620	-48.4339
19	Inestable.	-2.9682	-29.1584	-44.3998	-43.4550
20	Inestable.	-0.4838	-24.4300	-49.1704	-34.4243

Tabla 4.5 Datos y ganancias de la cuarta simulación.

Corrida 1. Corrida 2.

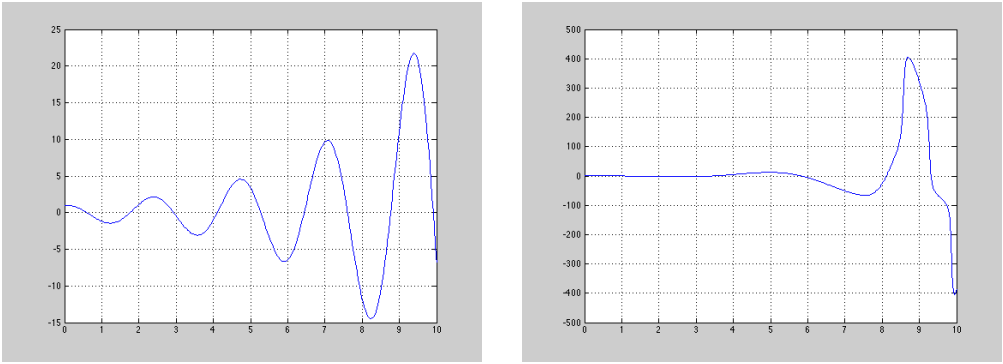

Figura 4.1 Gráficas que representan el comportamiento inestable del péndulo en las corridas 1 y 2.

Corrida 19. Corrida 20.

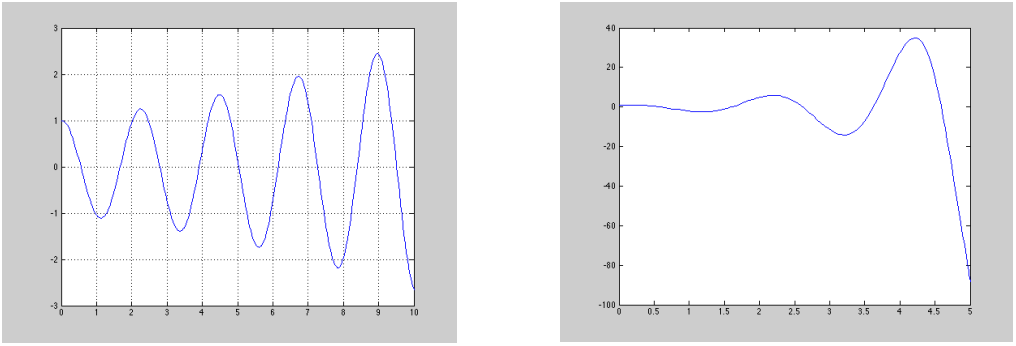

Figura 4.2 Gráficas que representan el comportamiento inestable del péndulo en las corridas 19 y 20.

En las corridas anteriores (1, 2 19 y 20) se observa que las simulaciones con ese rango de valores para las ganancias y ese número de individuos aun no estabilizan al sistema. Por lo que se propone como siguiente paso incrementar el número de población, primero a 100 y después a 1000 individuos, lo cual se observa en las gráficas siguientes.

Población: 100 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-50 a 0	-50 a 0	-50 a 0	-50 a 0
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.6 Parámetros de la quinta simulación.

Corrida	**Gráfica**	**Ganancias**			
		K1	**K2**	**K3**	**K4**
1	Inestable.	-0.7001	-0.1933	-41.4646	-8.7343
2	Inestable.	-4.27	-10.2643	-41.103	-49.2389
3	Inestable.	-2.4966	-5.7703	-44.4631	-30.2817
4	Inestable.	-3.7581	-1.8688	-42.6191	-18.5345
5	Inestable.	-0.9359	-3.0082	-38.3311	-24.5303
6	**Estable.**	**-2.9537**	**-27.8695**	**-49.6863**	**-36.4215**
7	Inestable.	-3.927	-26.2693	-45.7832	-43.5521
8	Inestable.	-11.1206	-2.2382	-35.1741	-42.4605
9	Inestable.	-1.7984	-3.5327	-49.6035	-20.022
10	Inestable.	-0.373	-2.801	-47.8695	-27.4098
11	Inestable.	-2.6653	-23.6023	-46.384	-38.3544
12	Inestable.	-2.9443	-27.6279	-48.5764	-37.4035
13	Inestable.	-6.3979	-5.3635	-35.7651	-42.4483
14	Inestable.	-8.056	-5.8562	-49.8742	-39.7467
15	Inestable.	-5.6053	-12.6105	-37.9792	-48.7481
16	**Estable.**	**-0.9812**	**-2.286**	**-48.4771**	**-16.9118**
17	Inestable.	-0.2901	-0.9356	-42.4567	-11.099
18	Inestable.	-1.1521	-18.4766	-48.1714	-27.4389
19	**Estable.**	**-2.495**	**-2.0688**	**-45.3209**	**-9.0193**
20	Inestable.	-0.5834	-2.5448	-47.0291	-22.5423

Tabla 4.7 Datos y ganancias de la quinta simulación.

Corrida 6. Corrida 16.

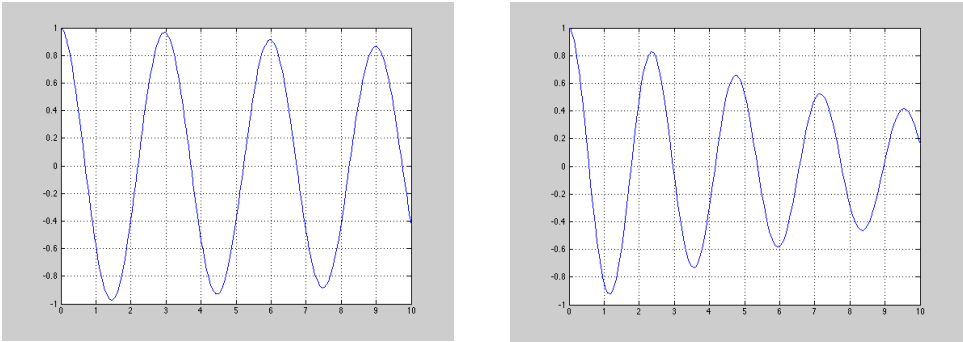

Figura 4.3 Gráficas que representan el comportamiento inestable y estable del péndulo en las corridas 6 y 16.

Corrida 19.

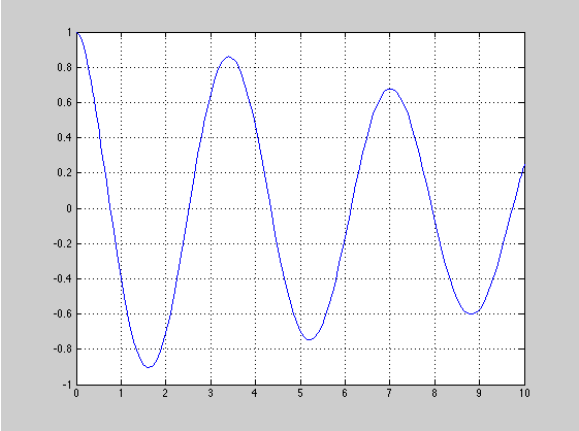

Figura 4.4 Gráfica que representa el comportamiento estable del péndulo en la corrida 19.

Se observa que en las corridas 6, 16 y 19 el sistema logra estabilizarse con ese rango específico de valores en las ganancias. Lo siguiente es aumentar la población de individuos para obtener más ganancias que estabilicen al sistema. Por lo que esta se aumenta a 1000 individuos.

Población: 1000 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-50 a 0	-50 a 0	-50 a 0	-50 a 0
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.8 Parámetros de la sexta simulación.

Corrida	**Comportamiento de la Gráfica**	**Ganancias**			
		K1	**K2**	**K3**	**K4**
1	Inestable	-0.5808	-3.3948	-48.5766	-16.2432
2	Inestable	-2.7611	-25.9767	-49.4942	-34.9382
3	**Estable**	**-0.8616**	**-1.3408**	**-42.8509**	**-5.3827**
4	**Estable**	**-1.5434**	**-2.7121**	**-46.9296**	**-10.5351**
5	Inestable	-0.5178	-0.3753	-48.9977	-11.8232
6	Inestable	-2.0238	-2.0038	-45.247	-18.4954
7	Inestable	-4.4378	-0.5232	-42.617	-11.3324
8	Inestable	-0.597	-0.8604	-45.1236	-12.2885
9	Inestable	-1.2366	-24.7689	-48.3943	-35.2857
10	Inestable	-4.5904	-0.9096	-48.2212	-14.448
11	Inestable	-0.1188	-0.0819	-39.3145	-0.1445
12	Inestable	-0.2372	-6.4248	-37.3246	-33.1131
13	Inestable	-1.5802	-1.9744	-43.2155	-9.1465
14	**Estable**	**-3.0978**	**-0.4847**	**-46.8615**	**-11.8241**
15	Inestable	-5.2997	-2.2193	-35.5979	-26.625
16	Inestable	-1.4269	-1.0665	-42.6078	-13.4074
17	Inestable	-1.6444	-0.5554	-42.1832	-8.0175
18	Inestable	-1.1275	-0.5308	-46.9694	-8.3688
19	Inestable	-2.7293	-1.3127	-47.2168	-13.201
20	Inestable	-2.1691	-1.5151	-40.098	-11.1889

Tabla 4.9 Datos y ganancias de la sexta simulación.

Corrida 3. Corrida 4.

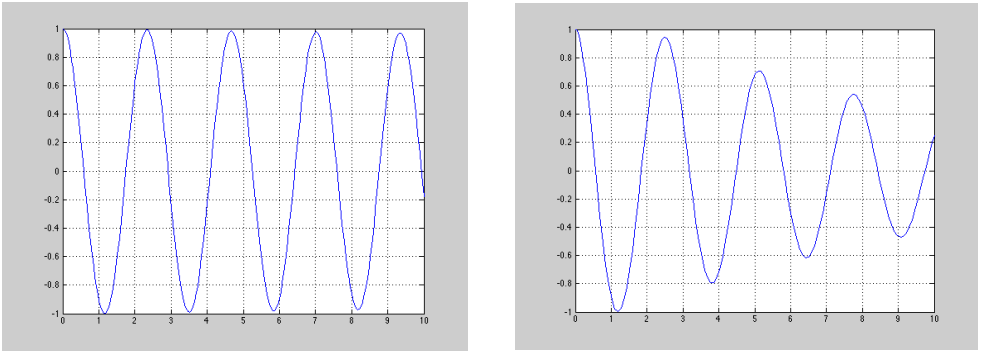

Figura 4.5 Gráficas que representan el comportamiento inestable y estable del péndulo en las corridas 3 y 4.

Corrida 14.

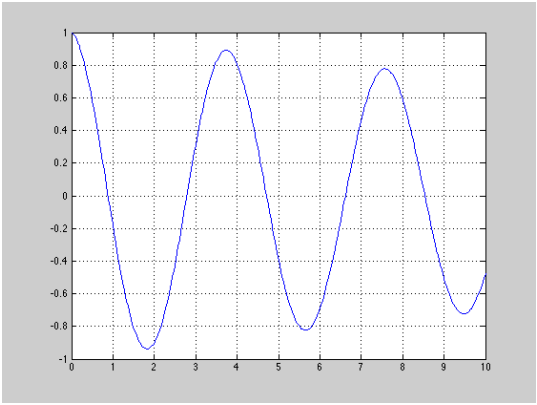

Figura 4.6 Gráfica que representa el comportamiento estable del péndulo en la corrida 14.

Al igual que en la simulación anterior de 100 individuos, la actual genera 3 corridas con ganancias que estabilizan al sistema. Estas simulaciones son la 3, 4 y 14.

4.3 Prueba con rangos específicos en cada ganancia.

Al observar las corridas anteriores se puede deducir que en la corrida 16 con los 100 individuos y en la corrida 4 con los 1000 individuos se generan valores cercanos a la estabilidad del sistema. Por lo que se pueden tomar como base los parámetros de dichas corridas para generar los rangos máximos y mínimos.

La corrida 16 de 100 individuos de población tiene como ganancias:-0.9812, -2.2860, -48.4771, -16.9118.

La corrida 4 de 1000 individuos de población tiene como ganancias: -1.5434, -2.7121, -46.9296, -10.5351.

Como se puede observar en la ganancia k1 se tienen valores de -0.9812 y -1.5434; Por lo que se propone un rango entre 0 y -3.

Para los rangos máximo y mínimo de la ganancia k2 se proponen entre 0 y -5, esto es debido a que las ganancias recopiladas son -2.2860 y -2.7121.

Los rangos máximo y mínimo de la ganancia k3 se proponen mucho más abiertos en comparación con los demás rangos, esto es debido a que las ganancias que se obtuvieron son -48.4771 y -46.9296. Lo cual indica que esta ganancia debe ser la más baja. Dichos rangos se encuentran entre -30 y -70.

Para la última ganancia, se proponen los rangos máximo y mínimo entre -5 y -20. Ya que las ganancias obtenidas son -16.9118 y -10.5351.

Los parámetros completos se muestran en la tabla 4.10.

Población: 20 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-3 a 0	-5 a 0	-70 a -30	-20 a -5
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.10 Parámetros de la séptima simulación.

Corrida	**Gráfica**	**Ganancias**			
		K1	**K2**	**K3**	**K4**
1	Inestable.	-0.0261	-0.5146	-66.9919	-6.8232
2	Estable.	-0.1220	-0.6	-69.2659	-8.5684
3	Estable.	-0.3861	-1.1719	-66.747	-8.7865
4	Estable.	-0.3123	-1.2122	-67.2921	-10.5636
5	**Estable.**	**-0.004**	**-0.0256**	**-61.7538**	**-5.9328**
6	**Estable.**	**-0.1199**	**-1.405**	**-58.0517**	**-10.4512**
7	Estable.	-0.3742	-0.4383	-61.4279	-10.187
8	Estable.	-0.2997	-0.08	-60.1527	-10.229
9	Estable.	-0.007	-0.3287	-56.9736	-7.3954
10	Estable.	-0.0341	-0.6658	-57.7872	-8.7395
11	Estable.	-0.1123	-0.0224	-64.9648	-10.4285
12	Estable.	-0.2622	-0.434	-68.3676	-12.1511
13	Estable.	-0.0286	-1.3596	-64.4703	-9.9262
14	**Estable.**	**-0.9036**	**-1.0738**	**-67.1727**	**-6.9671**
15	**Estable.**	**-0.0852**	**-0.0766**	**-60.5124**	**-7.3766**
16	Estable.	-0.8837	-0.2821	-64.2377	-8.9431
17	Estable.	-0.8857	-0.5861	-65.6601	-11.5608
18	Estable.	-1.0371	-0.634	-66.9117	-11.5751
19	Estable.	-0.1687	-0.4047	-57.8705	-6.3395
20	Estable.	-0.1789	-2.1045	-69.2316	-9.7332

Tabla 4.11 Datos y ganancias de la séptima simulación.

Corrida 5. Corrida 6.

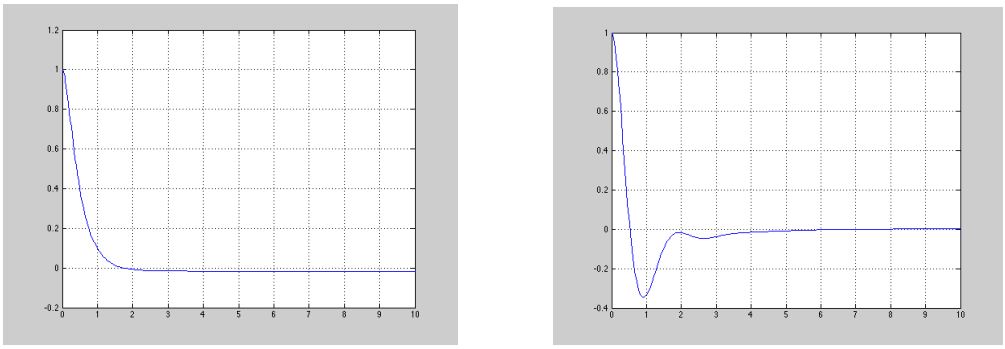

Figura 4.7 Gráficas que representan el comportamiento estable del péndulo en las corridas 5 y 6.

Corrida 14. Corrida 15.

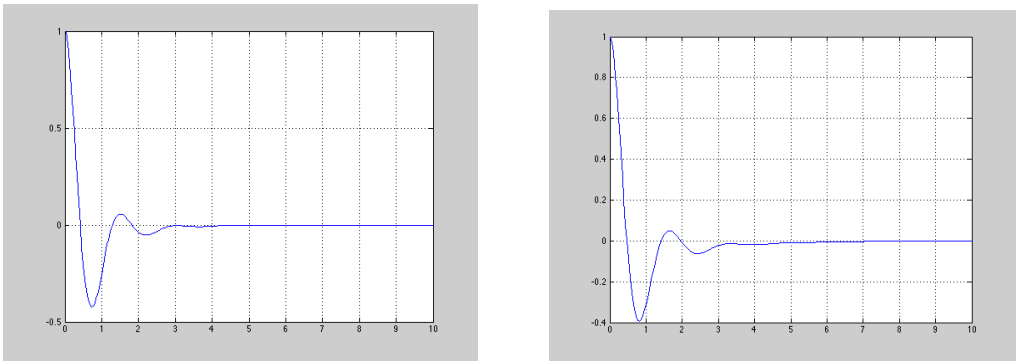

Figura 4.8 Gráficas que representan el comportamiento estable del péndulo en las corridas 14 y 15.

En las corridas 5, 6, 14 y 15 se obtienen las ganancias que estabilizan de mejor manera al sistema. Estas corridas son las que se muestran en las figuras 4.7 y 4.8.

Población: 100 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-3 a 0	-5 a 0	-70 a -30	-20 a -5
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.12 Parámetros de la octava simulación.

Corrida	**Comportamiento de la Gráfica**	**Ganancias**			
		K1	**K2**	**K3**	**K4**
1	Estable	-0.1111	-0.3531	-62.6831	-6.4054
2	**Estable**	**-0.2314**	**-0.26**	**-57.2638**	**-7.3658**
3	**Estable**	**-0.0529**	**-0.0751**	**-62.508**	**-5.5188**
4	Estable	-0.3534	-0.0343	-66.4461	-11.5559
5	Estable	-0.3381	-0.2738	-63.4663	-10.9077
6	Estable	-0.0131	-0.5669	-61.3275	-10.7598
7	Estable	-0.2815	-0.0113	-59.8218	-5.6549
8	Estable	-0.1776	-0.7058	-61.5892	-10.8617
9	Estable	-0.2471	-0.0075	-58.082	-5.0425
10	Estable	-0.8037	-0.0554	-67.9416	-11.5662
11	Estable	-0.178	-0.2857	-59.1488	-7.8616
12	Estable	-0.5479	-0.7129	-63.56	-7.7044
13	Estable	-0.2706	-0.5551	-67.6999	-11.6628
14	Estable	-0.2167	-0.4471	-58.92	-5.9529
15	**Estable**	**-0.4571**	**-0.5915**	**-67.3677**	**-11.2129**
16	Estable	-0.0706	-0.533	-65.4148	-7.9772
17	Estable	-0.1972	-0.5456	-64.3179	-10.8835
18	**Estable**	**-0.1702**	**-0.3242**	**-66.8394**	**-11.4831**
19	Estable	-0.0164	-0.0409	-60.1991	-6.7186
20	Estable	-0.1074	-0.0445	-65.8394	-10.908

Tabla 4.13 Datos y ganancias de la octava simulación.

Corrida 2. Corrida 3.

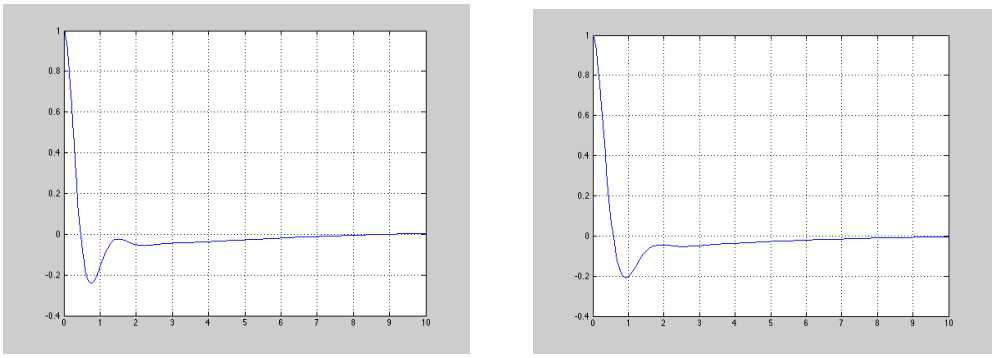

Figura 4.9 Gráficas que representan el comportamiento estable del péndulo en las corridas 2 y 3.

Corrida 15. Corrida 18.

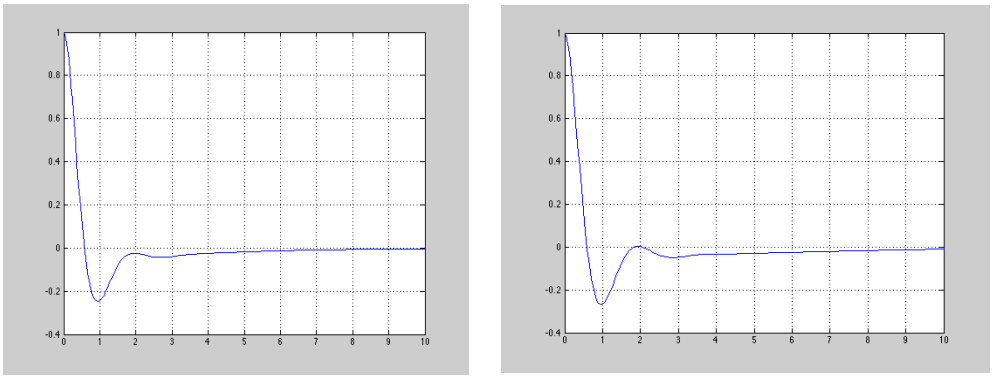

Figura 4.10 Gráficas que representan el comportamiento estable del péndulo en las corridas 15 y 18.

Al igual que la corrida de 20 individuos en la de 100 se seleccionan a las simulaciones que representa el mejor comportamiento del péndulo.

Población: 1000 individuos.			
Mín. y Máx. K1	**Mín. y Máx. K2**	**Mín. y Máx. K3**	**Mín. y Máx. K4**
-3 a 0	-5 a 0	-70 a -30	-20 a -5
Evoluciones	**Mutación**	**Tiempo de simulación**	**Número de ganancias**
10	4%	10	4

Tabla 4.14 Parámetros de la novena simulación.

Corrida	**Comportamiento de la Gráfica**	**Ganancias** **K1**	**K2**	**K3**	**K4**
1	**Estable**	**-0.2794**	**-0.1328**	**-61.5662**	**-6.1997**
2	**Estable**	**-0.0119**	**-0.2618**	**-61.292**	**-6.2122**
3	**Estable**	**-0.1407**	**-0.5237**	**-68.2856**	**-11.7864**
4	Estable	-0.5043	-0.521	-68.745	-11.8428
5	Estable	-0.2784	-0.5377	-67.6526	-11.7059
6	Estable	-0.1504	-0.1763	-69.2085	-11.5757
7	Estable	-0.8916	-0.0146	-68.9378	-11.5026
8	**Estable**	**-0.005**	**-0.1158**	**-59.2488**	**-6.2691**
9	Estable	-0.2452	-0.1058	-59.2027	-7.6666
10	Estable	-0.0137	-0.015	-60.6106	-6.4942
11	Estable	-0.042	-0.1858	-60.2721	-5.981
12	Estable	-0.1339	-0.1516	-61.9756	-7.3012
13	Estable	-0.1696	-0.5301	-68.9529	-11.949
14	Estable	-0.0875	-0.0056	-58.9245	-5.5489
15	Estable	-0.1973	-0.6577	-68.1976	-11.8204
16	Estable	-0.0292	-0.1681	-66.1057	-11.3286
17	Estable	-0.2952	-0.2279	-69.201	-11.6488
18	Estable	-0.0194	-0.0771	-60.7745	-5.7324
19	Estable	-0.1471	-0.0934	-60.4033	-6.9148
20	Estable	-0.0692	-1.2405	-68.399	-12.0922

Tabla 4.15 Datos y ganancias de la novena simulación.

Corrida 1. Corrida 2.

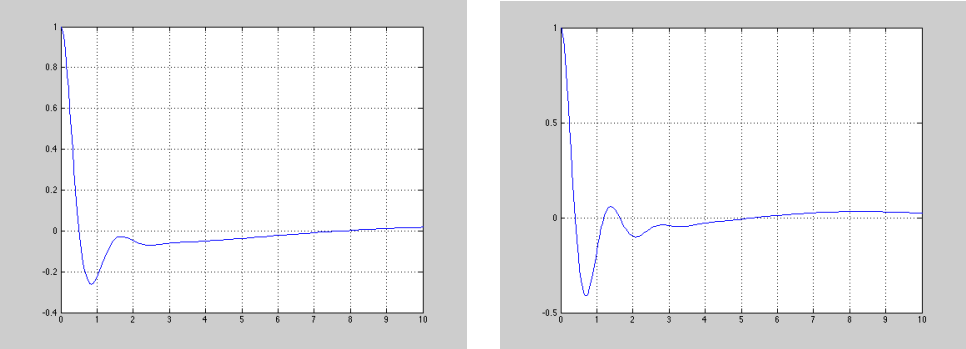

Figura 4.11 Gráficas que representan el comportamiento estable del péndulo en las corridas 1 y 2.

Corrida 3. Corrida 8.

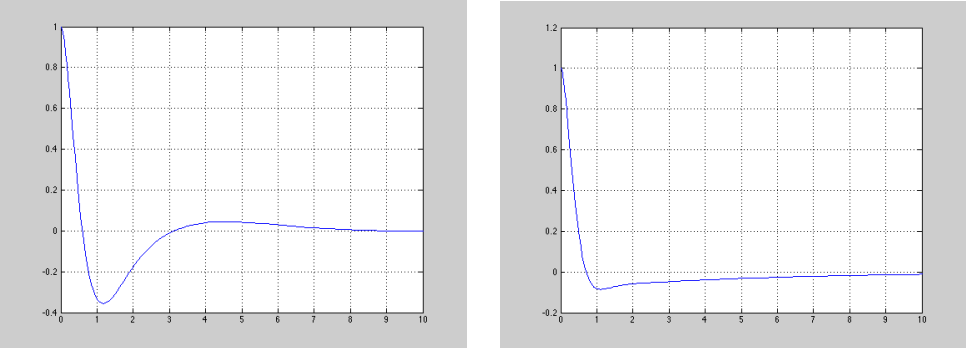

Figura 4.12 Gráficas que representan el comportamiento estable del péndulo en las corridas 3 y 8.

Al observar en las tres corridas diferentes (la de 20, 100 y 1000 individuos) se analiza que algunas corridas estabilizan de mejor manera al sistema que otras. Como por ejemplo en la corrida de 20 individuos (tabla 4.11) se tienen las simulaciones 5, 6, 14 y 15 (figuras 4.7 y 4.8).

En la corrida siguiente de 100 individuos (Tabla 4.13) se tienen las simulaciones 2, 3, 15 y 18 (figuras 4.9 y 4.10).

En la corrida de 1000 individuos (tabla 4.1) se tienen las simulaciones 1, 2, 3 y 8 (figura 4.11 y 4.12). Por lo que se propone utilizar las ganancias de las corridas más estables y generar un valor promedio de cada ganancia para determinar una ganancia promedio.

Al realizar lo anterior obtenemos las ganancias siguientes:

$$K = -0.2051 \quad -0.4054 \quad -62.6551 \quad -8.0646$$

4.4 Conclusiones del capítulo.

En este capítulo se desarrolló una metodología para determinar las ganancias adecuadas para estabilizar al sistema; dicha metodología se basa en utilizar en un principio un rango de ganancias muy amplio e ir reduciéndolo para ir encontrando ganancias que lleguen a estabilizar al sistema a la par de ir aumentando el número de individuos en la población.

Se observó que al aumentar el número de individuos en la población, aumenta la posibilidad de generar ganancias que estabilizan al sistema. Así como también al reducir el tiempo de simulación ayuda de manera significativa a encontrar dichas ganancias.

Capítulo 5

Conclusiones

5.1 Conclusiones.

Al analizar el capítulo anterior se puede desarrollar una metodología para determinar ganancias que estabilicen a diferentes sistemas en dado caso que no se tuviesen dichas ganancias.

Se propone seguir los siguientes pasos para lograr encontrar las ganancias de algún sistema de control que usen algoritmos evolutivos:

1.- Comenzar con un rango amplio entre el valor de las ganancias máximo y mínimo; el cual se propone entre 10 y -100.

2.- Empezar con un número relativamente bajo de individuos; 20 o 40 sería un buen número para iniciar dicha búsqueda.

3.- Las evoluciones se pueden concentrar entre 10 y 20.

4.- Comenzar con un 4% del porcentaje de mutación.

5.- El tiempo de simulación que se encuentre entre 1.5 y 10.

6.- Simular con los parámetros anteriores al menos 20 veces el programa.

7.- En cada simulación guardar gráficas y datos. En ocasiones la gráfica final no se genera debido a que el ODE45 queda ciclado al buscar una solución que crece hacia infinito. Las ganancias generadas deben ser almacenadas en una tabla.

8.- Una vez realizadas todas las simulaciones y guardado todos los datos, analizar cual o cuales son los datos que generan estabilidad en el sistema. Si alguna o algunas de las simulaciones estabiliza al sistema, basarse en dichas

ganancias para acotar los rangos máximos y mínimos de las ganancias y repetir los pasos anteriores.

9.- Si no se obtuvo en ninguna simulación la estabilidad en el sistema, lo siguiente es aumentar la población a 100 individuos y repetir los pasos anteriores. Si aun así no se obtiene al menos alguna corrida estable, se repiten los pasos anteriores pero ahora con una población aumentada a 1000 individuos.

10.- Si el resultado sigue siendo negativo, lo siguiente es acotar los rangos máximos y mínimos de las ganancias, es decir, reducir entre 0 y -50. Repetir los pasos anteriores a partir del 2.

10.- Si se obtuvieron algunas ganancias pero el sistema no es totalmente estable, lo siguiente es aumentar la población a 100 individuos. Repetir todos los pasos con la acotación entre 0 y -50. Una vez terminadas las simulaciones volver a realizar todos los pasos pero con 1000 individuos.

11.- Al acotar los rangos y aumentar la población, el sistema tiende a generar ganancias y en algunas corridas también genera una gráfica que representa la estabilidad en el sistema. Por lo que se toman dichos datos y a partir de estos se generan las acotaciones pertinentes para cada ganancia.

5.2 Aportaciones.

Los algoritmos evolutivos pueden implementarse en un sistema dinámico o ser utilizados en diferentes aplicaciones como en el control difuso, en la retroalimentación de estados, en un PCO, en la evolución diferencial, etc. Esto debido a su flexibilidad para adaptarse fácilmente a diferentes tipos de control y el ejemplo más claro es la implementación de un AGC al sistema de control del péndulo invertido explicado en este trabajo.

5.3 Trabajos futuros.

Como trabajos futuros se puede utilizar el mismo sistema dinámico (péndulo invertido) pero ahora llevando el bloque o carro de un punto a otro controlando al péndulo en el punto más alto utilizando un sistema de control como un PID, un control difuso o un control neuronal. Por lo que las ganancias del sistema aumentarían a 7; 4 ganancias para el péndulo y 3 ganancias del control PID. Una aplicación de los AGC puede ser en la optimización por enjambre de partículas. Otra de las aplicaciones sería la de implementar físicamente el sintonizador en el péndulo invertido.

Referencias.

[1] Randy L. Haupt. *Practical Genetic Algorithms* Wiley Intercience. Second edition 2004.

[2] José R. Espinoza C. *Apuntes. Introducción al Análisis de Sistemas No-Lineales* 543 703. Universidad de Concepción, Chile. 9na edición. Agosto 2009.

[3] Shankar Sastry. *Nonlinear systems, Analysis, Stability and Control.* Springer, 1999.

[4] Aström, K. J. and K. Furuta. *Swinging a pendulum by energy control.* 19th Congress of IFAC. Vol G.

[5] Yoshida K. *Swing-up control of an inverted pendulum by energy-based methods.* American Control Conference. San Diego California.

[6] P. J. Fleming. *Genetic Algorithms in Control Systems Engineering.* University of Sheffield, UK. PB_Fleming_Purshouse.pdf.2005.

[7] Katsuhiko Ogata. *Ingeniería de Control Moderna.* Tercera edición Pearson educación.

[8] Benjamín Kuo. *Sistemas de Control Automático.* Séptima edición. Prentice Hall. 1996.

[9] Goldberg D.E. *Genetic ALgorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Boston, MA, USA, segunda edición, 1992.

[10] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

[11] V. S. Sorokin. *Analysis of motion of inverted pendulum with vibrating suspension axis at low-frequency excitation as an illustration of a new approach for solving equations without explicit small parameter*. University of Denmark. Article available online: March 2014.

[12] M. Bettayeb, C.Boussalem, R.Mansouri, U.M.Al-Saggaf. *Stabilization of an inverted pendulum-cart system by fractional PI-state feedback.* University of Sharjah, UAE/King Abdulaziz University, KSA. University Mouloud Mammeri of Tizi-Ouzou, Algeria. King Abdulaziz University, KSA. Article available online: December 2013.

[13] Jia-Jun Wang. *Simulation studies of inverted pendulum based on PID controllers.* Institute of Automation, Hangzhou Dianzi University, Hangzhou, China. Article available online: August 2010.

[14] Manuel F. Pérez-Polo, Manuel Pérez Molina, Javier Gil Chica, José A. Berna Galiano. *Stability and chaotic behavior of a PID controlled inverted pendulum subjected to harmonic base excitations by using the normal form theory.* Departamento de Física, Universidad de Alicante, Escuela Politécnica Superior, Campus de San Vicente, Alicante, España. Article 2014.

[15] Ya Xuan Zhang, Zhong-Jie Han, Gen Qi Xu. *Expansion of solution of an inverterd pendulum system with time delay*. College of Science, Civil Aviation University, Tianjin University. China. Article available 2011.

[16] Rogelio Lozano, Isabelle Fantoni, Dan J. Block. *Stabilization of the inverted pendulum around its homoclinic orbit.* College of Engineering, University of Illinois at urbana-Champaign. Urbana USA. Article available 2000.

[17] Y. Yavin. *Control of a rotary inverted pendulum*. Laboratory of decision and control. Department of Electrical and Electronic Engineering. University of Pretoria. Pretoria South Africa. Article accepted october 1997.

[18] Fuyan Cheng, Guomin Zhong, Youshan Li, Zhengming Xu. *Fuzzy control of a double-inverted pendulum.* Hebei institute of Mechano-Electric Engineering, Shi Jia Zhuang, People´s Republic of China, Yanshan University. Article accepted 1996.

[19] Zongli Lin, Ali Saberi, Michael Gutmann, Jacov A. Shamash. *Linear controller for an inverted pendulum having restricted travel: A high-and-low gain approach.* Automatica, Vol. 32, No. 6, pp. 933-937, 1996.

[20] S. Yurkovich, M. Widjaja. *Fuzzy controller synthesis for an inverted pendulum system.* Department of Electrical Engineering. The Ohio State University, Columbus OH, USA. Article accepted 1996.

Apéndice A

Programación del modelo matemático del péndulo invertido utilizando Matlab®.

En este trabajo se usó Matlab® para simular el control del sistema, así como el acoplamiento del AGC a dicho control. Matlab® tiene la facilidad de simular el comportamiento del péndulo invertido con diferentes condiciones iniciales propuestas.

Los programas que van del 1 al 7 representan el modelo matemático del péndulo invertido en lazo abierto y con retroalimentación de variables de estado; Con diferentes condiciones iniciales fuera del punto de equilibrio para observar el comportamiento dinámico de dicho sistema.

Programa 1.

```
clc;
clear all;
close all;

global dx

dx=zeros(4,1);
ti=0;
tf=10;
[t,x]=ode45('pendulo_carro',[ti,tf],[0,0,0,0])
plot(t,x)
legend('pos carro', 'pos pendulo', 'vel carro', 'vel pend')
grid
```

Programa 1.- Código realizado en Matlab del compilador del sistema en lazo abierto con C. I. 0, 0, 0, 0.

Programa 2.

```
function dx = pendulo_carro(t,x)
global dx

%%%Parámetros%%%
m1=3.2;
m2=0.32;
l=0.44;
g=9.81;
f1=0;
tau=0;

q1=x(1);
q2=x(2);
q1p=x(3);
q2p=x(4);
qp=[q1p;q2p];

%%Parámetros de inercias
D=[m1+m2,m2*l*cos(q2);m2*l*cos(q2),m2*l*l];

%%Parámetros de Coriolis
C=[0,-m2*l*sin(q2)*q2p;0,0];

%%Parámetros de gravedad
G=[0;-g*m2*l*sin(q2)];

%%%Fuerzas centrífugas
F=[f1;tau];

%%Despeje de qpp de la formula gral del robot
qpp=inv(D)*(F-C*qp-G)

dx(1,1)=q1p;
dx(2,1)=q2p;
dx(3,1)=qpp(1);
dx(4,1)=qpp(2);
```

Programa 2.- Código del sistema en lazo abierto.

El programa siguiente (programa 3) tiene como condición 1° de inclinación en la posición del péndulo (es decir, pi/180 = 0.01745).

Programa 3.

```
clc;
clear all;
close all;

global dx

dx=zeros(4,1);
ti=0;
tf=10;
[t,x]=ode45('pendulo_carro',[ti,tf],[0, 0.01745, 0, 0])
plot(t,x)
legend('pos carro', 'pos pendulo', 'vel carro', 'vel pend')
grid
```

Programa 3.- Código realizado en Matlab del compilador del sistema en lazo abierto con C. I. 0, 0.01745, 0, 0.

Programa 4.

```
clear all
clc;
m1=3.2;
m2=0.32;
l=0.44;
g=9.81;

%Sistema linealizado
A = [0 1 0 0; 0 0 -m2*g/m1 0; 0 0 0 1; 0 0 (m1+m2)*g/(l*m1) 0];
B = [0; 1/m1; 0; -1/(l*m1)];
P = [-2+0.5*i -2-0.5*i -1+i -1-i];

%Cálculo de ganancias de retroalimentacion
K  = place(A,B,P);
[T,X] = ode45(@rigid,[0 10],[0 0 0 0],[],K);
plot(T,X(:,1),'-',T,X(:,2),'-.',T,X(:,3)*180/pi,'.')
```

Programa 4.- Código realizado en Matlab del compilador del sistema en lazo cerrado con C. I. 0, 0, 0, 0.

Programa 5.

```
function dx = rigid(t,x,K)
dx = zeros(4,1);
%Parámetros
m1=3.2;
m2=0.32;
l=0.44;
g=9.81;

f1 = -(K(1,1)*x(1)+K(1,2)*x(2)+K(1,3)*x(3)+K(1,4)*x(4));

dx(1)=x(2);
dx(2)=(1/(m1+m2*sin(x(3))^2))*(m2*sin(x(3))*(l*x(4)^2-
g*cos(x(3)))+f1);
dx(3)=x(4);
dx(4)=(1/(l*(m1+m2*sin(x(3))^2)))*(-m2*l*x(4)^2*sin(x(3))…
*cos(x(3))+(m1+m2)*g*sin(x(3))-f1*cos(x(3)));

% x1p = posición lineal
% x2p = velocidad lineal
% x3p = posición angular
% x4p = velocidad angular
end
```

Programa 5.- Código del sistema en lazo cerrado.

Programa 6.

```
clear all
clc;
m1=3.2;
m2=0.32;
l=0.44;
g=9.81;

%Sistema linealizado
A = [0 1 0 0; 0 0 -m2*g/m1 0; 0 0 0 1; 0 0 (m1+m2)*g/(l*m1) 0];
B = [0; 1/m1; 0; -1/(l*m1)];
P = [-2+0.5*i -2-0.5*i -1+i -1-i];

%Cálculo de ganancias de retroalimentacion
K  = place(A,B,P);

[T,X] = ode45(@rigid,[0 10],[0 0 0.01745 0],[],K);
plot(T,X(:,1),'-',T,X(:,2),'-.',T,X(:,3)*180/pi,'.')
```

Programa 6.- Código realizado en Matlab del compilador del sistema en lazo cerrado con C. I. 0, 0, 0.01745, 0.

En el siguiente programa se simula el sistema con una inclinación de 50°, lo cual generará en el sistema inestabilidad.

Programa 7.

```
clear all
clc;
m1=3.2;
m2=0.32;
l=0.44;
g=9.81;

%Sistema linealizado
A = [0 1 0 0; 0 0 -m2*g/m1 0; 0 0 0 1; 0 0 (m1+m2)*g/(l*m1) 0];
B = [0; 1/m1; 0; -1/(l*m1)];
P = [-2+0.5*i -2-0.5*i -1+i -1-i];

%Cálculo de ganancias de retroalimentacion
K  = place(A,B,P);

[T,X] = ode45(@rigid,[0 10],[0 0 0.8726 0],[],K);
plot(T,X(:,1),'-',T,X(:,2),'-.',T,X(:,3)*180/pi,'.')
```

Programa 7.- Código realizado en Matlab del compilador del sistema en lazo cerrado con C. I. 0, 0, 0.8726, 0.

Apéndice B

Programación del AGC y su acoplamiento al modelo matemático del péndulo invertido utilizando Matlab®.

Los programas que van del 8 al 12, son los que representan a los AG implementados en el sistema del péndulo invertido, es decir, dichos programas generarán una serie de ganancias aleatorias dentro del rango establecido para determinar si son adecuadas para estabilizar al péndulo en su punto más alto.

Programa 8.

```
clear all;
clc

%Parámetros del AG

N = 20;        %Número de individuos
O = 4;         %Numero de ganancias
NumEvo = 10;    %Numero de evoluciones
Mut    = 4;    %Porcentaje de mutación 7%

%Valores máximo y mínimo de las ganancias
FACTOR = 1.0;

pmin(1) = -2.5;    %Ganancia uno
pmin(2) = -4;      %Ganancia dos
pmin(3) = -65;     %Ganancia tres
pmin(4) = -15;     %Ganancia cuatro

pmax(1) = 0;       %Ganancia uno
pmax(2) = -1;      %Ganancia dos
pmax(3) = -40;     %Ganancia tres
pmax(4) = -5;      %Ganancia cuatro

%Intervalo de simulación
tspan=0:0.05:2;


%Se inicializa la población de parámetros.
P = rand(N,O);
```

```
%Se escala respecto al valor máximo y mínimo.
for i = 1: O
   P(:,i) = (pmax(i) - pmin(i))*P(:,i) + pmin(i);
end

%Función costo
contador = 0;
while(contador < NumEvo )

    %Se determina la función costo
    for i = 1:N

        k1 = P(i,1);
        k2 = P(i,2);
        k3 = P(i,3);
        k4 = P(i,4);
        K=[k1 k2 k3 k4];

    [T,X] = ode45(@parametros,[0 10],[0 0 0.01745 0],[],K);
       Y  = X(:,3);
       Costo(i) = sum((Y).^2);
       plot(T,X(:,3)*180/pi);
       grid on;
       hold on;
    end
       figure;


    %Se ordena de menor a mayor en función del costo,
    %en ind se guarda el índice
    [Costo_y, indx]=sort(Costo);

    %Se ordena la población en función del índice
    P = P(indx,:);

    %Se selecciona la mitad de la población con buen índice
    %de desempeño.
    P = P(1:N/2,:);


    %Se generan los índices para seleccionar de forma aleatoria
    %a las parejas.
    [F C]=size(P);

    t = randperm(F);
    idx= reshape(t,F/2,2);
    beta = rand(F/2);

    %Se realiza el cruce.
    for i = 1:F/2
        P_Nueva = Cruce(idx(i,:),beta(i),P,pmin,pmax);
    end

    %Se realiza la mutación.
    P_Nueva = Mutacion(P_Nueva, Mut, pmin,pmax);
```

```
  P = [P;P_Nueva];

contador = contador +1
end

%Se determina la función costo
    for i = 1:N
        k1 = P(i,1);
        k2 = P(i,2);
        k3 = P(i,3);
        k4 = P(i,4);

        [T,X] = ode45(@parametros,[0 10],[0 0 0.01745 0],[],K);
        Y  = X(:,3);
       Costo(i) = sum((Y).^2);
       plot(T,X(:,3)*180/pi);
    end

    %Se ordena de menor a mayor en función del costo
    %en ind se guarda el índice
    [Costo_y, indx]=sort(Costo);

    %Se ordena la población en función del índice
    P = P(indx,:);

        k1 = P(1,1)
        k2 = P(1,2)
        k3 = P(1,3)
        k4 = P(1,4)

      [t,Xg] = ode45(@parametros,[0 10],[0 0 0.01745 0],[],K);
      Yg  = Xg(:,3);
      plot(T,Xg(:,3)*180/pi);
      grid on;
```

Programa 8.- Código realizado en Matlab llamado “agprincipal.m”

Programa 9.

```
 function dx = parametros(t,x,K)
dx = zeros(4,1);

%Parámetros
m1=3.2;
m2=0.32;
l=0.44;
g=9.81;

f1 = -(K(1,1)*x(1)+K(1,2)*x(2)+K(1,3)*x(3)+K(1,4)*x(4));

dx(1)=x(2);
```

```
dx(2)=(1/(m1+m2*sin(x(3))^2))*(m2*sin(x(3))*(l*x(4)^2-
g*cos(x(3)))+f1);
dx(3)=x(4);
dx(4)=(1/(l*(m1+m2*sin(x(3))^2)))*(-
m2*l*x(4)^2*sin(x(3))*cos(x(3))+(m1+m2)*g*sin(x(3))-
f1*cos(x(3)));

% x1p = posición lineal
% x2p = velocidad lineal
% x3p = posición angular
% x4p = velocidad angular
```

Programa 9.- Código realizado en Matlab llamado “parametros.m”

Programa 10.

```
function [ P_Nueva ] = Cruce( idx,beta,P_Nueva,pmin,pmax )
[F C]=size(P_Nueva);

 for i = 1:C

        aux1 = P_Nueva(idx(1,1),i) +
(beta)*(P_Nueva(idx(1,1),i)-P_Nueva(idx(1,2),i));
        aux2 = P_Nueva(idx(1,2),i) -
(beta)*(P_Nueva(idx(1,1),i)-P_Nueva(idx(1,2),i));

        while((aux1 > pmax(i)) || aux1 < pmin(i))
            beta = rand();
            aux1 = P_Nueva(idx(1,1),i) +
(beta)*(P_Nueva(idx(1,1),i)-P_Nueva(idx(1,2),i));
        end

        while( (aux2 > pmax(i)) || aux2 < pmin(i))
            beta = rand();
            aux2 = P_Nueva(idx(1,2),i) -
(beta)*(P_Nueva(idx(1,1),i)-P_Nueva(idx(1,2),i));
        end

        P_Nueva(idx(1,1),i) = aux1;
        P_Nueva(idx(1,2),i) = aux2;

 end
end
```

Programa 10.- Código realizado en Matlab llamado “cruce.m”

Programa 11.

```
function [ P_Nueva ] = Mutacion( P_Nueva,GM,pmin,pmax)

[R C] = size(P_Nueva);

NumMut = round(GM*R*C/100.0);
for i = 1: NumMut
    ix = ceil(C*rand);
    iy = ceil(R*rand);

    P_Nueva(iy,ix) = pmin(ix) + rand*(pmax(ix)-pmin(ix));

end

end
```

Programa 11.- Código realizado en Matlab llamado “mutacion.m”