SINTONIZACION DE UN CONTROLADOR PID EN UN PLC HACIENDO USO DE INTELIGENCIA DE ENJAMBRES.

ARTURO DUQUE MARIN

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE OPERACIONES Y SISTEMAS
PROGRAMA DE INGENIERIA MECATRÓNICA
SANTIAGO DE CALI
2015

SINTONIZACION DE UN CONTROLADOR PID EN UN PLC HACIENDO USO DE INTELIGENCIA DE ENJAMBRES.

ARTURO DUQUE MARIN

Pasantía Institucional para optar el título de Ingeniero Mecatrónico

Director
PhD. JESÚS ALFONSO LÓPEZ SOTELO
Ingeniero Electricista

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE OPERACIONES Y SISTEMAS
PROGRAMA DE INGENIERIA MECATRÓNICA
SANTIAGO DE CALI
2015

Nota de Aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar el título de ingeniero Mecatrónico

JUAN CARLOS MENA

Jurado

Santiago de Cali, 18 de Marzo de 2015

AGRADECIMIENTOS

Es importante darle gracias a la vida por brindarme la oportunidad de realizar mis estudios profesionales. Agradecer a mis padre y familia por su apoyo y a la empresa que me permitió desarrollar este proyecto; Robotek LTDA. Cabe resaltar también el apoyo y ayuda recibida por parte Jesus Alfonso Lopez director del programa de ingeniería Mecatrónica quien dirigió este proyecto.

CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

CPU: corresponde a la unidad central de procesamiento, encargada de los cálculos matemáticos y demás funciones relacionadas con el manejo de datos.

INTELIGENCIA ENJAMBRES: es una rama de la Inteligencia artificial que estudia el comportamiento colectivo de los sistemas descentralizados, auto-organizados, naturales o artificiales. El concepto se emplea en los trabajos sobre inteligencia artificial. La expresión fue introducida por Gerardo Beni y Wang Jing en 1989, en el contexto de los sistemas robóticos móviles[1].

MATLAB: abreviatura de MATrix LABoratory, "laboratorio de matrices") Es un lenguaje de alto nivel e interactivo usado por millones de ingenieros y científicos alrededor del mundo. Permite la exploración y visualización de ideas así como la colaboración entre disciplinas, incluidas el procesamiento de imágenes y señales, comunicaciones, control y finanzas computacionales[2].

OPC: OPC es un estándar para la conectividad de datos usado en la industria para facilitar la comunicación entre sistemas de control industrial y aplicaciones de diferentes características, permitiendo el envió y la recepción de datos sin que sea necesaria la compatibilidad entre los drivers de comunicación de los equipos.

PID: es un esquema de control que se compone de una acción proporcional al error, una derivativa y una acción integral.

PLC: es la abreviación de "Programable Logic Controller", es decir, controlador lógico programable, dispositivo ampliamente usado en la industria.

PSO: forma abreviada para expresarse acerca de la optimización por enjambre de partículas, en inglés "Particle Swarm Optmization".

---

[1] BENI G., WANG J, Swarm intelligence in cellular robotic systems. En: Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Agosto 1989, p 40-45.

[2] MATLAB [en linea] The Lenguaje of Technical Computing, MatLab Corporation, 2014 [Consultado 30 de Enero de 2015]. Disponible en internet: http://www.mathworks.com/products/matlab/

RSLINX: corresponde al software necesario para establecer la comunicación entre el PLC y el software de programación RsLogix5000. Dentro de sus funciones adicionales se encuentra la creación de servidores OPC.

RSLOGIX5000: software requerido para la programación de cualquier PLC de la firma Allen Bradley en específico aquellos de la familia CompacLogix

ST: abreviación de "Structured Text" es un lenguaje de programación de alto nivel que se puede usar para la programación de un PLC, su estructura es similar a la de PASCAL.

RESUMEN

Los avances de la electrónica han permitido el desarrollo de sistemas más rápidos en el procesamiento de datos y eficientes en el consumo de energía, permitiendo la implementación de algoritmos complejos que requieren de gran cantidad de recursos computacionales, como capacidad de memoria o capacidad de procesamiento.

Considerando esto, hoy en día es posible implementar algoritmos complejos; como aquellos usados en la inteligencia artificial, en microcontroladores y PLCs que actualmente ofrecen las condiciones requeridas para un buen funcionamiento de los mismos.

En este trabajo de grado, se implementara en un PLC Allen Bradley un algoritmo de inteligencia de enjambres cuya función es la de determinar el modelo matemático de un sistema o proceso, esto con el fin de calcular los parámetros de un controlador PID que ejercerá la acción de control más adecuada para lograr del sistema la respuesta deseada.

La prueba del funcionamiento del algoritmo en el PLC se hará usando modelos matemáticos de sistemas dinámicos en matlab, desde donde se establecerá comunicación con el PLC. El PLC enviara las señales de control y desde matlab se envían los datos correspondientes a la salida de la planta. Para esto se usara el estándar de comunicación OPC.

PALABRAS CLAVE: Automatización, PLC, Inteligencia Artificial, PID, Sintonización, Inteligencia de Enjambres, OPC.

INTRODUCCION

No son muchas las técnicas de control usadas en los procesos industriales e implementados en los PLCs, las razones principales se deben a que muchas de estas técnicas tienen un alto grado de complejidad y requieren de un alto poder computacional. Es por esta razón que la estrategia de control más usada en la industria es el PID, la explicación a su constante uso es la simplicidad del mismo y su efectividad.

Aun con sus grandes ventajas para el control de los procesos industriales, los controladores PID en muchas ocasiones no son sintonizados de tal forma que alcancen su máximo potencial, ya que la sintonización del mismo requiere de un amplio conocimiento en el comportamiento del sistema, así como de conceptos teóricos relacionados con el tiempo de muestreo, el tiempo muerto o el orden del sistema. De esta manera, se hace entonces necesaria la presencia de un técnico o ingeniero con gran conocimiento y experticia en el tema para obtener del PID su mejor rendimiento. Considerando esto, en la industria se han desarrollado herramientas para sintonizar los controladores PID, consiste en un Auto Tune que define los parámetros del controlador y permite su óptimo funcionamiento así como la robustez del mismo.

En la industria, el control de los procesos se lleva a cabo haciendo uso de PLCs, que son sistemas computacionales que contienen un microprocesador el cual es programado para la ejecución de un algoritmo de control. Los algoritmos de control implementados en los PLCs poseen la cualidad de ser simples y efectivos como el PID, pero al momento de enfrentarlos a procesos multivariables o no lineales, estos algoritmos dejan de ser efectivos y se hace necesaria la implementación de técnicas de control más complicadas que permitan controlar este tipo de sistemas o procesos.

Gracias a los grandes avances en la electrónica, especialmente en los microprocesadores, los PLCs cuentan hoy en día con una gran capacidad de cómputo y procesamiento lo que los hace ideales para la implementación de algoritmos complejos. Así, algoritmos de inteligencia computacional o de inteligencia artificial pueden ser implementados hoy en día en un PLC y de esta manera ofrecen más y mejores soluciones a los problemas de control que se presentan en la actualidad.

Entonces, es posible encontrar que estos algoritmos inteligentes implementados en los PLCs, se usen con frecuencia para la sintonización de los parámetros de

una estrategia de control simple como un PID, así como para la modelación matemática de un sistema. De esta manera se pueden diseñar sistemas de control que se adapten a los cambios dinámicos de la planta y generar las acciones de control adecuadas.

Una de las técnicas de inteligencia computacional recibe el nombre de inteligencia de enjambres, siendo esta un área de la ingeniería y la inteligencia computacional relativamente nueva que ofrece muy buenas soluciones a problemas de aproximación y optimización y con la cual se pretende trabajar en este proyecto.

Siguiendo esta línea, el propósito de este proyecto es entonces implementar en un PLC de la compañía Rockwell Automation (PLC Allen Bradley) un algoritmo basado en inteligencia de enjambres que permita la sintonización de un controlador PID.

1. ANTECEDENTES

Entre los años 50 y 60 se desarrollaron un gran número de ideas útiles en la teoría del control adaptativo, estas ideas ocasionan el surgimiento de nuevas y mejores técnicas de control que permitieron la manipulación de sistemas con dinámicas complejas y no lineales. Todos estos avances se llevan a cabo en el periodo posterior a la segunda guerra mundial donde empieza a germinar la necesidad de controlar aeronaves, misiles y otro tipo de dispositivos, cuya dinámica no podía ser modelada y a su vez controlada con las técnicas establecidas hasta el momento[3].

De igual forma, se han documentado varios ejercicios de sintonización de controladores PID en un PLC, donde hacen uso de un ordenador para la implementación del algoritmo de sintonización, mientras en el PLC se encuentra el PID, Así, se tiene un trabajo publicado en 2011 donde se sintoniza un PID en un PLC haciendo uso de un ordenador, donde se identifica a la planta con un algoritmo basado en mínimos cuadrados recursivos[4].

Se han documentado también gran cantidad de proyectos donde se implementan técnicas de lógica difusa en los PLC, uno de los más interesantes, propone la lógica difusa en un PLC para el control de un Vehículo no Tripulado, el cual es usado para el transporte de materiales en un compañía[5].

Por otro lado, se han desarrollado técnicas de control, que implican el uso de algoritmos de inteligencia computacional que brindan robustez y eficacia al momento de manipular el comportamiento de diversos sistemas y plantas. De esta manera, la inteligencia computacional empieza a abrirse cabida en el mundo de la industria. Así, en el año 2008 se documenta la implementación de un algoritmo de optimización de partículas (PSO) en un PLC, donde el propósito es encontrar de manera óptima los pesos sinápticos de una neurona y controlar de esta manera un sistema de potencia eléctrica[6], esto permite vislumbrar como los PLC ofrecen

[3] ÅSTRÖM J. Karl., Adaptative Control Around 1960. En: Decision and Control, Proceedings of the 34th IEEE Conference, Junio, 1996, vol 3, p. 44-49

[4] RIVERA Manuel Manyari, Integrated Online Auto-tuning and Digital Implementation of PID Controllers in Industrial Processes. En: Control and Automation (ICCA), Agosto 2011, vol 9. p 30-34

[5] YAHYAEI Mehdi, Increasing the Flexibility and Intelligence of Material Handling through the Factory by Integrated Fuzzy Logic Controller with Programmable Logic Controller. En: 14th IEEE International Conference on Fuzzy Systems, 2005

[6] PARROT C, G.K Venayagamoorthy, Implementation of neuroidentifiers trained by PSO on a PLC platform for a multimachine power system. En: Swarm Intelligence Symposium, Junio 2008, vol 1, p 1-6.

ahora una alternativa para la implementación de complejos algoritmos donde es posible aplicar la inteligencia artificial para el control de procesos.

Adicional a esto, se han implementado estrategias de control PID auto-sintonizados, haciendo uso de inteligencia de enjambres, estas técnicas de control son implementadas en FPGAs, obteniendo muy buenos resultados en el rendimiento y robustez en la estrategia de control[7,8]. El problema radica en la falta de robustez a nivel de hardware que estos dispositivos ofrecen.

A nivel local se tienen aplicaciones de algoritmos inteligentes enfocados en diferentes áreas, tales como la optimización de parámetros en controladores, reconocimiento de patrones y organización de datos. La lógica difusa también ha tenido cabida en estos proyectos, en el año 2011, Diego Alexander Mazuera desarrolla una librería de lógica difusa para los PLC siemens[9].

De igual manera, en la Universidad Autónoma de Occidente se han implementado algoritmos genéticos en diferentes proyectos de investigación, liderados por el profesor Jesus Alfonso Lopez Sotelo, Director de programa de ingeniería Mecatrónica. Algunos de estos proyectos se caracterizan por la implementación de algoritmos inteligentes y algoritmos genéticos en PLCs, cuya función es en la mayoría de los casos es optimizar el método para encontrar los parámetros de un PID o los parámetros que definen el comportamiento de una planta.

[7] HUANG Yourui, QU Liguo, TIAN Yiming, Self-Tuning PID Controller Based on Quantum Swarm Evolution Algorithm. En: Natural Computation Fourth International Conference on, Junio 2008, Vol:6, p 401-404

[8] YAJUAN Chen, QUINGHAI Wu, Design of PID controller based on PSO algorithm and FPGA. En: Intelligent Control and Information Processing (ICICIP), 2011,p 1102-1105.

[9] MAZUERA Diego Alexander, Diseño de una Librería Fuzzy para PLC, Trabajo de grado Ingeniero Mecatronico, Santiago de Cali, Universidad Autónoma de Occidente, Universidad Autonoma de Occidente, 2011, 111 p.

2. MARCO TEORICO

2.1 AUTÓMATAS PROGRAMABLES (PLCS).

Los controladores lógicos programables o PLCs, son dispositivos electrónicos que tienen la capacidad de almacenar, estructurar y procesar la información que reciben a través de sus entradas o sus programas y generar una nueva información en sus salidas, permitiendo el funcionamiento automático de una secuencia o proceso industrial[10].

Se han diseñado de tal forma que soporten ambientes industriales, donde el ruido electrónico, las vibraciones y el aire no son propicios para el uso de un dispositivo electrónico cualquiera, adicional a esto, los PLCs ofrecen hoy en día un gran poder computacional lo que los hace aptos para la implementación de algoritmos o esquemas de control complejos.

Los autómatas programables cuentan con dos grandes grupos de componentes que son la CPU (Unidad central de procesamiento) y las unidades de entrada y salida. La unidad central de procesamiento comprende esencialmente dos componentes que son el procesador y la memoria, de esta manera, los elementos que permiten el intercambio de información entre el proceso y las acciones que brinda el controlador son los módulos de entrada y salida[11].

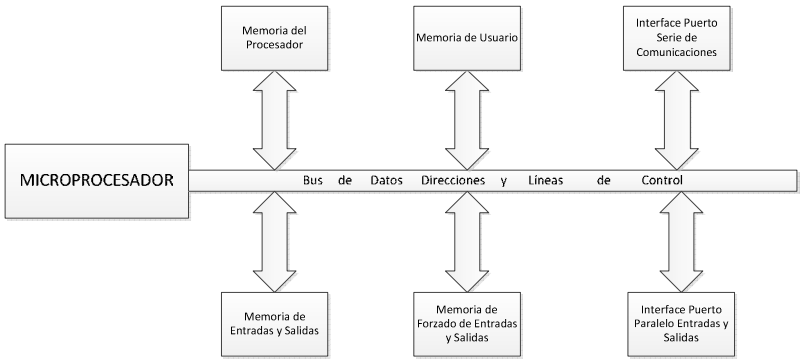
Comúnmente, los procesadores de los PLC se encentran rodeados de circuitos integrados que generalmente son memorias en donde el fabricante ha guardado el firmware del equipo para que él mismo lleve a cabo tareas de[11]:

- Adquisición y Actualización de Estados de las señales de entrada y salida.
- Vigilancia y diagnóstico del funcionamiento de un equipo.
- comunicaciones con los periféricos.

La figura 1 muestra un ejemplo de la organización más común en los procesadores de los PLCs.

[10] FLOWER LEIVA Luis, Controles y Automatismos, 10 Ed. Bogotá: Panamericana Formas e Impresos, 2008, p 10-20.
[11] MAYOL BADÍA Albert, Autómatas Programables. Mora de Toledo: Marcombo S.A., 1988, p 123

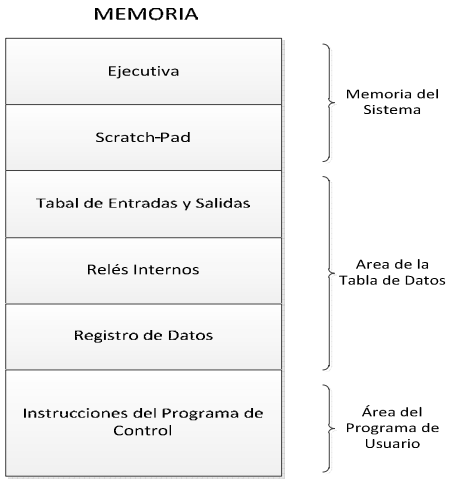
Figura 1. Esquema de microprocesadores en un PLC.



Fuente: MAYOL BADÍA Albert, Autómatas Programables. Toledo: Marcombo S.A., 1988.

Las instrucciones que el procesador ejecuta de acuerdo a los estados de la memoria, de las entradas o las salidas se llevan a cabo de manera secuencial, es decir instrucción por instrucción. Adicional a esto, la memoria de los PLC se encuentra organizada en áreas de trabajo, lo que condiciona el espacio de memoria al cual accede el procesador para generar, guardar o ejecutar los programas de control que en él se escriben. La figura 2 muestra un esquema de la organización de la memoria en un PLC discriminando de esta manera tres sectores que se diferencian entre sí por su nivel de acceso y funcionalidad al momento de poner en funcionamiento un programa.

Figura 2. Organización de un microprocesador en un PLC



Fuente: MAYOL BADÍA Albert, Autómatas Programables. Toledo: Marcombo S.A., 1988

Siguiendo este esquema, encontramos entonces que gracias a los avances electrónicos, es posible entonces ampliar el poder de procesamiento así como la capacidad de memoria del PLC, lo que se traduce en programas más grandes y complejos que permiten la implementación de estrategias de control no tradicionales.

Los PLC, para ejecutar la tarea que se desea deben ser programados de tal forma que a través de códigos o símbolos diseñemos un esquema de funcionamiento para que el PLC funcione. Existen diversos tipos de lenguajes para programar un PLC, así, estos lenguajes pueden ser visuales o escritos, donde los visuales hacen referencia a la programación haciendo uso de esquemas o símbolos gráficos, en su lugar, los escritos requieren que se dicte o escriba la sentencia que el PLC debe ejecutar.

De esta manera, los fabricantes de PLCs han desarrollado gran variedad de lenguajes para la programación de los mismos, a continuación una figura con la clasificación de los mismos y ejemplos para cada una de las clasificaciones:

Tabla 1. Tipos de lenguaje que se pueden tener en un PLC

Lenguaje	Características	Ejemplos*	Tipo	Nivel
Listas	Lista de Instrucciones	IL AWL STL IL/ST	Escrito	Bajo
Plano	Diagrama Eléctrico	LADDER LD KOP	Visual	Alto
Diagrama de Bloques Funcionales	Diagrama Lógico	FBD FBS FUD		
Organigrama de Bloques Secuenciales	Diagrama Algorítmico	AS SFC PETRI GRAFCET		
Otros	Lenguajes Usados en Otras Áreas de la computación	BASIC C	Escrito	
* Los nombres fueron asignados por el fabricante				

Fuente: Guía de Manejo para PLC, Autómatas Programables. SENA (Servicio nacional de Aprendizaje). Santiago de Cali, 2005.

Se puede profundizar en el tema acudiendo a los libros propuestos en la bibliografía.

2.2 ESTRATEGIA DE CONTROL PID

Las habilidades del controlador Proporcional-Integral (PI) y Proporcional-Integral-Derivativo (PID) para compensar la mayoría de los procesos de la industria, han permitido su amplia aceptación en aplicaciones de la industria como en muchas otras áreas[12]. Aun con su amplia aceptación, estudios acerca del estado del arte sobre las prácticas con controladores PID, han demostrado que en la mayoría de las aplicaciones no se obtiene una buena sintonización, lo que disminuye considerablemente el rendimiento del controlador haciéndose necesario encontrar nuevas técnicas o métodos para sintonizar un PID.

Siguiendo ahora con el enfoque teórico, encontramos el modelo en tiempo continuo para un controlador PID expresado en el dominio Laplaciano por:

$$U(S) = G_c(S) * E(S)$$

Ecuación 1

Con

$$G_c(S) = K_c * \left(1 + \frac{1}{T_i S} + T_d S\right)$$

Ecuación 2

La introducción de los términos integral y derivativo en el esquema del controlador permite que la planta alcance el valor de consigna establecido por el operador así como la anticipación ante cambios de la salida deseada respectivamente. Así, se puede entonces exponer cada uno de los componentes que forman el esquema de control de un PID.

2.2.1 Acción Proporcional. Para este esquema de PID, se tiene la siguiente ecuación que la representa:

$$U(t) = Ke(t) + u_b$$

Ecuación 3.

[12] O'DWYER Aidan, Handbook of PI and PID Controller Tuning Rules, 2 Ed,Londres: Imperial College Press, 2006

Donde se tiene que la acción proporcional hace referencia únicamente a un valor K proporcional al error que se obtiene de la realimentación. Algunas de las propiedades que tiene este tipo de lazo de control son: la disminución de los efectos del ruido o las perturbaciones sobre el comportamiento del sistema, la no disminución hasta cero del error del sistema en estado estacionario, la estabilización del sistema cuando el valor de K es muy grande.

2.2.2 Acción Integral. La función principal de la acción integral es asegurarse de que la salida del proceso coincide con el punto de consigna en estado estacionario[13], la ecuación que define la acción integral es:

$$u(t) = \left(\frac{1}{T_i}\int e(\tau)d\tau\right)$$

Ecuación 4.

Como se había mencionado, la acción integral contribuye a que el error del lazo de control alcance un valor de cero, pero al igual que la acción proporcional, puede ocasionar que el sistema se vuelva inestable.

2.2.3 Acción Derivativa. La acción derivativa se encarga de mejorar la estabilidad del lazo de control, adicional a esto, actúa como un lazo que permite predecir el comportamiento de la planta haciendo que la acción integral y proporcional se anticipe a cambios en el proceso. La formulación matemática para este esquema es:

$$u(t) = T_d \frac{de(t)}{dx}$$

Ecuación 5.

[13] ÅSTROM J. Karl., Control System Design [en línea]: Lecture Notes for ME 155A. Santa Barbara, California, 2002 [consultado 30 de Enero de 2014]. Disponible en Internet:http://neutron.ing.ucv.ve/eiefile/Control%20I/Astrom_notas.pdf

Así, conociendo los componentes de un controlador PID, se pueden definir diferentes esquemas donde la configuración e interacción de cada una de las partes antes mencionada es diferente, lo que permite obtener características de funcionamiento distintas, los esquemas más comunes se definen a continuación:

$$C(s) = K_c * \left(1 + \frac{1}{T_i S} + T_d S\right)$$

Ecuación 6.

La ecuación 6 denota el esquema tradicional de un controlador PID.

$$C(s) = K_c * \left(1 + \frac{1}{T_i S}\right)(1 + S T_d)$$

Ecuación 7.

Se conoce como un esquema de PID en serie, donde la acción derivativa influye en la acción integral.

$$C(s) = K + \frac{k_i}{S} + S k_d$$

Ecuación 8.

Este último esquema hace referencia a una configuración en paralelo de cada una de las partes del controlador PID.

2.2.4 WindUp. El WindUp corresponde a una situación no deseada en los esquemas de control PID. Consiste en el incremento excesivo de la acción integral cuando la planta empieza a comportarse en lazo abierto al momento de presentarse una limitación en el actuador de la planta debido a una acción de control que supera los límites de funcionamiento del mismo.

Estos valores elevados en la acción de control y que saturan al actuador son consecuencia en muchas ocasiones de cambios muy bruscos en el punto de consigna, esto provoca un valor elevado del error y por tanto una acción de control elevada que busque eliminar de manera rápida este error.

2.3 INTELIGENCIA DE ENJAMBRES

La inteligencia de enjambres es una disciplina de inteligencia artificial que se encarga del diseño de sistemas multi-agentes con aplicaciones en optimización, robótica, entre otras. En lugar de ser sofisticados controladores que manejan el comportamiento global de un proceso, se cuenta con pequeñas entidades no sofisticadas que cooperan para obtener el comportamiento deseado. La inspiración para el diseño de estos sistemas proviene del comportamiento colectivo de insectos como las hormigas, abejas, termitas así como del comportamiento de las aves o cardúmenes de peces[14]. Así, el término de inteligencia de enjambres surge en el año 89 en un trabajo sobre sistemas celulares robóticos[15].

La inteligencia de enjambres permite entonces la solución de problemas complejos haciendo uso de pequeños agentes que interactúan entre sí para encontrar la mejor solución. Estos algoritmos, reúnen entonces características de búsqueda aleatoria y selección de los individuos mejor adaptados[16]. De esta manera, se puede definir un esquema general acerca del funcionamiento de estos algoritmos:

Procesan simultáneamente, no una solución al problema, sino todo un conjunto de ellas. Estos algoritmos trabajan con alguna forma de representación de soluciones al problema, que se denominan individuos, el conjunto de estos individuos forma lo que se conoce entonces como población.

La composición de la población se va modificando a lo largo de las iteraciones del algoritmo, a lo que se le denomina generaciones. De generación en generación, además de variar el número de copias de un mismo individuo, también pueden aparecer nuevos individuos generador mediante operaciones de transformación.
Cada generación incluye un proceso de selección que da mayor probabilidad de permanecer en la población y participar en las operaciones de reproducción a los mejores individuos.

[14] BLUM Christian, MERKLE Daniel, Swarm Intelligence, Introduction and Applications, Berlin, Springer, 2008, 283 p
[15] BENI G., WANG J., Swarm intelligence in cellular robotic systems. En: Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Agosto 1989, p 40-45.
[16] ARAUJO Lourdes, CERVIGÓN Carlos, Algoritmos Evolutivos, Un Enfoque Práctico, Alfaomega RA-MA, 2009, 332 p

La inteligencia de enjambres cuenta con diversas técnicas que dependen de la manera en cómo se comportan los individuos de la población y cómo evolucionan o cambian las generaciones con cada iteración, así encontramos las siguientes técnicas:

2.3.1 Optimización por enjambre de partículas (PSO). Este tipo de técnica, se caracteriza por modelar el comportamiento de bandadas de aves y cardúmenes, donde cada individuo, que corresponde a una posible solución al problema, cambia de estado hasta encontrar un estado estable o ha llegado a un límite computacional.

En la optimización por enjambre de partículas, los individuos son partículas que se mueven a través de un espacio multidimensional, en donde cualquier cambio de estado o posición que haga una partícula es debido al comportamiento de las otras partículas, es decir, los cambios en una partícula son influenciados por la experiencia o conocimiento de sus vecinos[17].

El esquema básico para representar este tipo de algoritmos inicia con la definición de las partículas, que están constituidas por un vector D que denota el número de variables del problema, una posición x, y una velocidad que se encarga de definir el comportamiento de la partícula[18], como se muestra a continuación:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

Ecuación 9

El factor velocidad integra la componente cognitiva de la partícula así como la componente social de la misma. El factor de conocimiento hace referencia a la distancia de la partícula con respecto al mejor estado o posición, y la parte social hace se encarga de la relación que se establece con las otras partículas del enjambre[17].

[17] ENGELBRECHT Andries, Computational Intelligence An Introduction, 2 Ed, Londres: Wisley, 2007, 630 p

[18] CETINA DOMINGUEZ Omar, Una Adaptación del Comportamiento de la Abeja Exploradora en el Algoritmo de la Colonia Artificial de Abejas para Resolver Problemas de Optimización con Restricciones. Trabajo de Grado de Maestría en Ingenieria, Laboratorio Nacional de Informática Avanzada, Facultad de Ingeniería, 2003, 48 p

Así, a lo largo de su desarrollo, se han planteado dos algoritmos de PSO, que se diferencian por la forma en como cada partícula interactúa con el enjambre, tenemos pues un algoritmo denominado *gBest* y *lBest,* que se ilustran a continuación:

- **Algoritmo del mejor global o gBest**: Para este algoritmo se define un único líder, este líder es seguido por cada una de las partículas y se establece una relación entre todas las partículas del enjambre, así, la componente social de la velocidad, se encuentra influencia por todos los componentes del enjambre. La figura 4, ilustra el algoritmo que define este tipo de estrategia dentro del PSO.

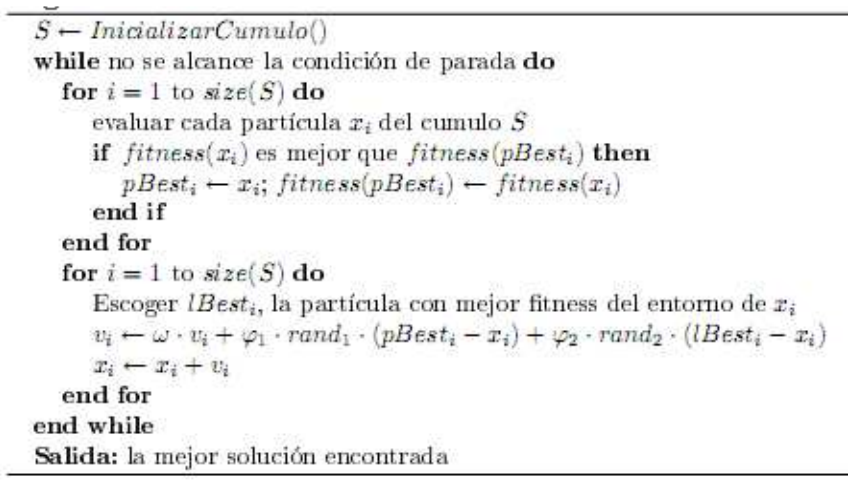
Figura 3 Algoritmo del mejor Global o gBest

$S \leftarrow InicializarCumulo()$
while no se alcance la condición de parada **do**
  for $i = 1$ to $size(S)$ **do**
    evaluar cada partícula x_i del cúmulo S
    if $fitness(x_i)$ es mejor que $fitness(pBest_i)$ **then**
      $pBest_i \leftarrow x_i$; $fitness(pBest_i) \leftarrow fitness(x_i)$
    end if
    if $fitness(pBest_i)$ es mejor que $fitness(gBest)$ **then**
      $gBest \leftarrow pBest_i$; $fitness(gBest) \leftarrow fitness(pBest_i)$
    end if
  end for
  for $i = 1$ to $size(S)$ **do**
    $v_i \leftarrow \omega \cdot v_i + \varphi_1 \cdot rand_1 \cdot (pBest_i - x_i) + \varphi_2 \cdot rand_2 \cdot (gBest - x_i)$
    $x_i \leftarrow x_i + v_i$
  end for
end while
Salida: la mejor solución encontrada

Fuente: GARCÍA J. M., Algoritmos Basados en Cúmulos de Partículas para la resolución de Problemas Complejos, [en línea]: Networking and Emerging Optimization, 2006, [consultado 30 de Enero de 2014]. Disponible en Internet: http://neo.lcc.uma.es/staff/jmgn/doc/Memoria_PFC_JMGN.pdf

- **Algoritmo del mejor Local o lBest:** Para este algoritmo se crean subgrupos que definen vecindades entre las partículas, así, cada partícula tiene unos vecinos diferentes y se define un conocimiento local para cada una de esas vecindades. A continuación se muestra el algoritmo de esta estrategia:

Figura 4 Algoritmo del mejor Local lBest



Fuente: GARCÍA J. M., Algoritmos Basados en Cúmulos de Partículas para la resolución de Problemas Complejos, [en línea]: Networking and Emerging Optimization, 2006, [consultado 30 de Enero de 2014]. Disponible en Internet: http://neo.lcc.uma.es/staff/jmgn/doc/Memoria_PFC_JMGN.pdf

2.3.2 Optimización por Colonia de Hormigas (ASO). Trata de emular el comportamiento cooperativo de las hormigas, donde se tiene un grupo de algoritmos que pueden ofrecer una solución al problema que se presenta. Se basa en el comportamiento de las hormigas para seguir el rastro de las feromonas que otras hormigas dejan, así, eligen el camino o la dirección que contenga la mayor concentración de feromonas y que eventualmente conducirá a la mejor solución.

El principal algoritmo, y del cual se derivan todos los demás, es el Ant System (AS). Un Sistema de Colonia de Hormiga (ACS) artificial es un sistema basado en agentes, el cual simula el comportamiento natural de las hormigas y desarrolla mecanismos de cooperación y aprendizaje. El ACS fue propuesto por Marco Dorigo en 1996; investigador Belga, como una nueva heurística para resolver problemas de optimización combinatoria. Así, el principal y primero de los algoritmos que surge de este tipo de técnica de inteligencia de enjambres se describirá en los siguientes párrafos.

Este algoritmo, denominado sistema de hormiga (Ant System AS), consiste en la definición de una función que representa la feromona que dejan las hormigas al momento de salir en busca de comida. de igual manera se define una función que determina la probabilidad de que una hormiga se desplace de un lugar a otro, dependiendo esto de la feromona, de esta manera, en cada iteración, el valor de la feromona se actualiza y por tanto las probabilidades de que la hormiga se mueva a un estado diferente varían, permitiendo esto la optimización del procesos de búsqueda de la solución al problema[19].

La ecuación que representa la feromona se muestra a continuación:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Ecuación 10.

[19] DORIGO Marco, CARO Gianni, Ant colony optimization: a new meta-heuristic. En: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, Agosto 1999, vol 2

Donde $\Delta\tau_{ij}^{k}$, representa la cantidad de feromona dejada por la hormiga k sobre el camino que recorre, el valor ρ representa la razón de evaporación de la feromona, dependiendo esto del nivel de error o cumplimiento del objetivo que la hormiga logra cuando recorre un camino. En la construcción de la solución, la hormiga se mueve de un lugar i a un lugar j siguiendo la siguiente función probabilística:

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k} \left[\tau_{il}(t)\right]^{\alpha} \cdot \left[\eta_{lj}\right]^{\beta}} \qquad \text{con} \quad j \in N_i^k$$

Ecuación 11

2.3.3. Algoritmos basados en modelos sociales de las abejas (BHO).

Las abejas ofrecen un gran número de comportamientos sociales que pueden ser tenidos en cuenta para su modelación computacional, uno de estos comportamientos consiste en la búsqueda de alimento, donde de forma óptima, las abejas buscan y comunican posibles lugares que sirvan para la obtención del alimento.

El algoritmo fue desarrollado por Kevin Passino y Nicanor Quijano, como se mencionó anteriormente, este simula el comportamiento de las abejas al momento de llevar a cabo una actividad específica, como buscar comida, por tanto, modela las danzas y los criterios de decisión de las abejas para salir en busca de alimento en un espacio de búsqueda.

De esta manera, al igual que las colmenas de abejas, el algoritmo define unos agentes que se encargan de explorar el espacio, adquirir experiencia y transmitir información respecto a los mejores lugares o fuentes de alimento. La danza que realizan las abejas es definida en el algoritmo a través de una función de beneficio que determina que tan buena es la fuente de alimento que los exploradores han encontrado. Una vez encontrado un valor de beneficio o provecho que determina la posible mejor solución al problema, la abeja que encontró tal estado o lugar vuelve a la colmena y transmite el mensaje a las demás abejas de la colmena, se envían entonces; de manera aleatoria, abejas exploradoras a ese lugar con el propósito de explotar ese espacio y definir la solución más óptima [Quijano].

Las principales funciones que definen el comportamiento de las abejas son:

$$TA^i = \begin{cases} \frac{1}{Prof_i} \cdot a & si\ 1^{era}\ vez\ en\ la\ Fuente \\ TA^i_{ant} + \xi. \end{cases}$$

Ecuación 12

Que se conoce como la tendencia de abandono, y determina si una abeja que se encuentra explorando el espacio, debe continuar o no por el camino en el cual se encuentra. Como se observa en la función, esta tendencia de abandono depende de la función de beneficio o provecho. De igual manera, se define una función que regula las danzas, equivalentes a la transmisión de la información de las abejas a la colmena, esta función permite entonces determinar si una fuente o posible solución debe ser explotada por las abejas de la colmena.

$$P_u = \frac{Prof(u)}{\sum_{k \in J} Prof(k)}$$

Ecuación 13

2.4 OPC (OLE FOR PROCESS CONTROL)

OPC es un estándar para la conectividad de datos usado en la industria para facilitar la comunicación entre sistemas de control industrial y aplicaciones de diferentes características, permitiendo el envió y la recepción de datos sin que sea necesaria la compatibilidad entre los drivers de comunicación de los equipos.

De esta manera, OPC es un conjunto de interfaces estándar basado en tecnología OLE/COM de Microsoft. La aplicación de la interface estándar OPC hace posible la interoperabilidad entre las aplicaciones de automatización/control, sistemas/dispositivos de campo etc[20].

Este estándar surge después de una serie de mejoras a diferentes protocolos y aplicaciones desarrolladas por Microsoft. Después de mostrar en Windows 3.0 un mecanismo estándar para el intercambio de datos entre aplicaciones; DDE o

[20] Diccionario de Makintrop OPC[en línea], Definition OPC Standard [Consultado el 20 de Febrero de 2015]. Disponible en internet: http://www.matrikonopc.es/resources/dictionary.aspx.

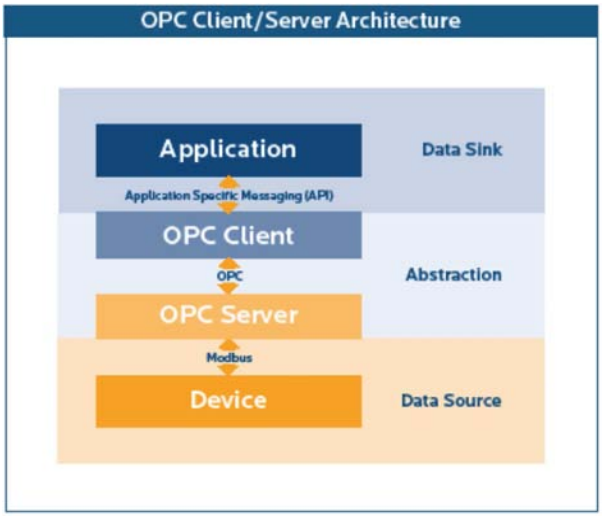
intercambio dinámico de datos, se empezó a desarrollar y a estandarizar lo que hoy se conoce como OPC.

OPC establece una línea entre proveedores de hardware y software. Facilita un mecanismo para proporcionar los datos de una fuente y comunicar esos datos a cualquier aplicación del cliente. Un vendedor puede desarrollar un servidor optimizado para comunicarse con la fuente de datos, y mantener el mecanismo de acceso a los datos de la fuente o dispositivo. Incorporando al servidor una interfaz OPC, permite a cualquier cliente el acceso a sus dispositivos[21].

2.4.1 Arquitectura OPC. El funcionamiento de este estándar de comunicación se basa en la utilización de dos componentes fundamentales, que son un Cliente OPC y un Servidor OPC, entre estos dos componentes se da el intercambio de datos. La fuente de datos no se comunica directamente con la aplicación que hará uso de estos, así, el dispositivo o fuente de datos; por ejemplo un PLC, se comunica con el servidor OPC así, el cliente OPC puede acceder a los datos que se encuentran en el servidor y pasarlos a la aplicación; que puede ser una interfaz o sistema de visualización en un computador, sin la necesidad de tener compatibilidad entre los protocolos de comunicación de la fuente y la aplicación. La figura 6 permite comprender mejor la arquitectura del estándar OPC.

[21] Pérez, Fede. Escuela Técnica Superior, Bilbao, España. OPC Conceptos Fundamentales. 2004

Figura 5 Arquitectura del Estándar OPC.



Fuente: DAREK Kominek, OPC: ¿De qué se trata y cómo funciona?; Guía para entender la Tecnología OPC; Alberta; Canadá, 2003, p 8.

2.4.2 Componentes del Estándar OPC. El estándar OPC hace uso de un servidor y un cliente, que permiten el intercambio de datos entre la fuente y el destino, a continuación se describe estos componentes:

- **Servidor OPC:** Un Servidor OPC es una aplicación de software. Un driver "estandarizado" desarrollado específicamente para cumplir con una o más especificaciones OPC.

La palabra "Server" en "OPC Server" no hace referencia en absoluto al ordenador donde este software se estará ejecutando. Hace referencia a la relación con el Cliente OPC. Los Servidores OPC son conectores que se pueden asimilar a traductores entre el mundo OPC y los protocolos nativos de una Fuente de Datos.

OPC es bidireccional, esto es, los Servidores OPC pueden leer de y escribir en una Fuente de Datos. La relación Servidor OPC/Cliente OPC es de tipo maestro/esclavo, lo que significa que un Servidor OPC sólo transferirá datos de/a una Fuente de Datos si un Cliente OPC así se lo pide.

- **Cliente OPC:** Un Cliente OPC es una pieza de software creada para comunicar con Servidores OPC. Utiliza mensajería definida por una especificación concreta de la OPC Foundation.

Conceptualmente Un Cliente OPC representa un destino de datos. Inician y controlan la comunicación con Servidores OPC basados en las peticiones recibidas desde la aplicación en la que están embebidos. Los Clientes OPC traducen las peticiones de comunicación provenientes de una aplicación dada en la petición OPC equivalente y la envían al Servidor OPC adecuado para que la procese. A cambio, cuando los datos OPC vuelven del Servidor OPC, el Cliente OPC los traduce al formato nativo de la aplicación para que ésta pueda trabajar de forma adecuada con los datos.

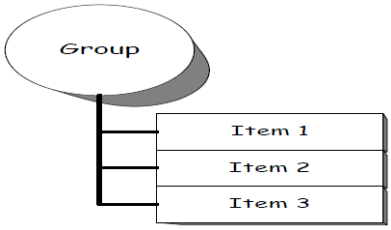
Técnicamente: Los Clientes OPC son módulos de software utilizados por una aplicación para permitirla comunicarse con cualquier Servidor OPC compatible visible en la red. Típicamente, los Clientes OPC están embebidos en aplicaciones como HMIs, SCADAs, graficadores, Historiadores o generadores de informes, convirtiéndolos en aplicaciones compatibles OPC[22].

2.4.3 Grupos e Ítems en OPC. El estándar OPC usa para la transferencia de datos elementos que se denominan grupos e ítems, esto permite tener la información organizada en el servidor y facilitar el acceso a la misma, así, en el servidor se tienen grupos de datos, dentro de los cuales tenemos ítems que corresponden a los datos que se usaran en la comunicación.

Los Ítems OPC representan conexiones a las fuentes de datos dentro de un servidor. Desde el punto de vista de una interfaz, un Ítem OPC, no es accesible como un objeto por parte de un Cliente OPC. Por lo tanto, no hay ninguna interfaz externa definida para un Ítem OPC. Todos los accesos a Ítems OPC se hacen a través de un objeto del Grupo OPC que contiene al Ítem OPC[21]. La siguiente figura ilustra de manera más clara la jerarquización de los datos en el estándar OPC.

[22] DAREK Kominek, OPC: ¿De qué se trata y cómo funciona?; Guía para entender la Tecnología OPC; Alberta; Canadá, 2003, p 8.

Figura 6. Organización de datos en el Estándar OPC



Fuente: DAREK Kominek, OPC: ¿De qué se trata y cómo funciona?; Guía para entender la Tecnología OPC; Alberta; Canadá, 2003, p 8.

2.4.4 OPC y Matlab. Dentro de las diferentes herramientas que provee Matlab podemos encontrar el Toolbox correspondiente al manejo de un cliente OPC, con este cliente se puede entonces establecer una comunicación para el intercambio de datos con cualquier servidor OPC que se encuentre en el equipo u ordenador en donde se esté ejecutando Matlab, permitiendo la lectura, escritura y demás acciones con los datos que se quieren manejar.

2.5 LENGUAJES DE PROGRAMACIÓN PARA PLC

Los lenguajes que se pueden usar para programar un PLC tienen diferentes formas. Para facilitar el manejo de los mismos y permitir que ingenieros con poco conocimiento en programación de PLC's pudiesen entender y así mismo diseñar sus propios programas se creó el lenguaje de programación escalera o Ladder. Este lenguaje emula la lógica que se usaba con contactores eléctricos en los sistemas de control industrial épocas atrás, lo que permitió que los técnicos de la industria pudieran adaptarse a esta nueva forma de hacer control.

Cada fabricante de PLC's ha creado y definido la manera en que son programados sus respectivos equipos, sin embargo se creó un estándar con el propósito de unificar la forma en como son programados los PLC's sin importar el fabricante. Este estándar se denomina IEC 1131-3 y provee; además de normas con respecto a los lenguajes de programación, diferentes procedimientos y manuales que encierran toda la vida útil del PLC. Es decir, definiciones generales del PLC, requerimientos eléctricos y mecánicos para el funcionamiento de los mismos, lenguajes de programación, guías para la selección, instalación y mantenimiento de PLC's entre otros aspectos.

Así, en este estándar se mencionan 5 lenguajes que sirven para programar un PLC, que son:

- Diagramas Ladder
- Funciones graficas secuenciales
- Diagramas con bloques de funciones
- Lenguaje en texto estructurado
- Lista de funciones

2.5.1 Lenguaje de Programación Ladder. Es un lenguaje de programación muy popular en los autómatas programables debido a que es fácil de entender y de usar, la lógica usada por este lenguaje se fundamenta en los esquemas eléctricos de control clásicos permitiendo que técnicos e ingenieros que no conozcan mucho acerca de programación puedan entender los programas en el PLC.

2.5.2 Funciones Graficas Secuenciales. Las funciones graficas secuenciales se asemejan en gran medida a los diagramas de flujo de un proceso, este usa pasos y transiciones para ejecutar una acción u operación en especial y así controlar un sistema.

2.5.3 Diagramas con Bloques de Funciones. Es un lenguaje de programación grafico orientado al uso de compuertas lógicas AND, OR, NOT y las combinaciones que con ellas se obtienen.

2.5.4 Lenguaje de Texto Estructurado. El lenguaje estructurado usado en el PLC se asemeja en gran medida al lenguaje de programación PASCAL. Donde los programas se escriben con una serie sentencias e instrucciones separadas por un punto y coma y donde cada sentencia o instrucción ejecuta un algoritmo para una tarea en especial[23].

2.5.5 Lista de Funciones Se caracteriza por hacer uso de funciones o instrucciones simples, se puede entender como un lenguaje de tipo ladder solo que descrito por medio de texto. Una lista de instrucciones se genera de tal forma que cada función o instrucción se define en una nueva línea, así, cada instrucción consta de un operador seguido de operadores[23].

[23] BOLTON W, Programmable Logic Controllers, 5 Ed. Londres: Newunes, 2009.

3. OBJETIVOS

3.1 OBJETIVO GENERAL.

Desarrollar una estrategia de sintonización de un PID basado en inteligencia de enjambres para ser implementada en un PLC Allen Bradley.

3.2 OBJETIVOS ESPECÍFICOS

- Identificar la arquitectura y caracterizar el funcionamiento del PLC a usar en el proyecto.

- Diseñar la estrategia de sintonización del PID usando inteligencia de enjambres.

- Validar la estrategia de sintonización del PID diseñado con inteligencia de enjambres mediante simulación.

- Comprobar el funcionamiento del algoritmo en el PLC haciendo uso de modelos matemáticos de plantas y usando el estándar de comunicación OPC entre el PLC y Matlab.

4. PLANTEAMIENTO DEL PROBLEMA

Los grandes avances tecnológicos y el desarrollo de nuevos productos han contribuido a la implementación de nuevas técnicas de producción así como el uso de sistemas o plantas con procesos caracterizados por dinámicas complejas o no lineales, bajo estas condiciones, las técnicas de control implementadas hasta el momento empiezan a ser poco eficaces, no brindando robustez y eficiencia.

De igual forma, en la industria la mayoría de los lazos de control de los procesos son PID[24], esto gracias a su buen desempeño y confiabilidad al momento de realizar control. A pesar de esto, la dinámica y la complejidad de los sistemas a controlar ha aumentado y el clásico PID ya no puede brindar la misma funcionalidad.

Debido a lo anterior, se han desarrollado nuevas técnicas de control así como algoritmos que permiten la modelación y manipulación de sistemas con dinámicas variables, no lineales o muy complejas. Estas nuevas técnicas ofrecen una gran versatilidad al momento de implementar controladores en los procesos industriales de la actualidad y conllevan a un aumento en la robustez y eficacia en el control.

Así, podemos nombrar los algoritmos basados en inteligencia de enjambres, que ofrecen gran variedad de soluciones en problemas de optimización y control de procesos, este tipo de técnica permite entonces la estimación de los parámetros de una planta así como la sintonización de controladores, planteando a partir de estas ideas un esquema de control adaptativo basado en algoritmos genéticos.

Surge a partir de este punto la idea de implementar un sistema basado en inteligencia de enjambres que permita la sintonización de un PID, con el propósito de mantener las buenas características de control de esta estrategia, adicionándole con la inteligencia de enjambres, la versatilidad para ser implementado en sistemas con dinámicas complejas, agregándole la cualidad de robustez ante cambios repentinos del proceso.

El problema de estos nuevos modelos y algoritmos de control radica entonces en la necesidad de un equipo con alta capacidad computacional para poder ejecutar las líneas de código del algoritmo, ciñéndonos entonces al uso de un ordenador.

[24] ÅSTRÖM J. Karl, HÄGGLUN Tore, Control PID Avanzado, 2 Ed. Madrid: Pearson Prentice Hall, 2009, 488 p.

Pero gracias a los avances en la tecnología de los microprocesadores, es factible entonces encontrar dispositivos industriales capaces de brindar gran poder de cómputo, con la cualidad adicional de ser diseñados para soportar ambientes industriales.

Tal es el caso de los PLC, sistemas programables ampliamente usados en la industria gracias a su amplia robustez y facilidad de programación. Actualmente estos sistemas cuentan con microprocesadores que permiten la implementación de técnicas de control en tiempo real así como una gran variedad de técnicas de control.

A partir de este punto, se plantea entonces la siguiente pregunta, que concentra el interés de esta investigación:
¿Es posible implementar una técnica de inteligencia de enjambres para la sintonización de un controlador PID en un PLC (Controlador Lógico Programable)?

5. DESARROLLO DEL PROYECTO

El tema central de este proyecto es el desarrollo de un algoritmo para ser usado en un PLC, por lo tanto la metodología de diseño se centrara en el software a desarrollar. Aun así, se consideraran los diferentes aspectos físicos que podrían influir en el correcto funcionamiento del algoritmo, tales como la capacidad de memoria y de procesamiento del PLC a usar.

5.1 IDENTIFICACIÓN DE NECESIDADES

Las necesidades se proponen de acuerdo a los requerimientos que se deducen del planteamiento del problema y el análisis realizado por el grupo de diseño. Se tienen entonces los siguientes puntos:

- Que el algoritmo entregue la respuesta de manera rápida
- Que el algoritmo sea Robusto
- Que el algoritmo sea confiable
- Que tenga buena capacidad de memoria
- Que pueda conectarse con otros periféricos o sistemas
- De bajo costo
- Que pueda ser usada fácilmente por el operario
- El algoritmo pueda ser reutilizado
- Lenguaje de programación estándar
- Que use algoritmos de inteligencia de enjambre
- Que logre controlar una planta
- El esquema del controlador debe ser un PID

5.2 REQUERIMIENTOS DEL PROBLEMA

Considerando las anteriores necesidades se les ha asignado un valor que identifica su importancia, tomando el 1 como el nivel más bajo de importancia y el 5 como el mayor nivel de importancia.

Tabla 2. Necesidades del Problema

N°	Necesidades	Importancia
1	Respuesta Rápida del Algoritmo	5
2	El sistema es Robusto	4
3	Algoritmo Confiable	4
4	Buena Capacidad de Memoria	4
5	Conexión con otros Periféricos	3
6	De bajo Costo	3
7	De Fácil Uso	4
8	Algoritmo Reutilizable	4
9	Lenguaje de Programación Estándar	4
10	Uso de Inteligencia de Enjambres	5
11	Que Logre Controlar una Planta	5
12	El Controlador debe ser un PID	5

5.3 ESPECIFICACIONES TECNICAS

Considerando las necesidades planteadas en el numeral anterior, podemos elaborar la siguiente tabla y definir los requerimientos para este proyecto así como su nivel de importancia y la forma en que se medirán. El nivel de importancia se califica con una escala de 1 a 5 siendo 1 la menor puntuación y 5 la mayor.

Tabla 3. Requerimientos Técnicos

N°	Necesidad	Requerimiento	Importancia	Unidad
1	1	Velocidad de Procesamiento	4	Kb/s
2	2,3	Robustez	4	Subjetivo
3	4	Capacidad de Almacenamiento	4	Kb
4	5	Manejo de Periféricos	3	Modularidad (Binario)
5	6	Económico	3	Pesos
6	7	Interfaz intuitiva	4	Subjetivo
7	8,9	Software Ergonómico	5	IEC 631131-3
8	10,11	Técnica de Identificación por Enjambres	5	Lista
9	12	Controlador	5	Tipo

5.4 ANALISIS DE LA CALIDAD Y CASA DE LA CALIDAD

La casa de la calidad nos permite definir las necesidades del problema que más importancia tienen así como los parámetros funcionales que permitirán la elaboración de un producto con buenas características de calidad y que cumplirán con los requerimientos del problema.

En la Tabla 3 se tiene la casa de la calidad donde se observa la correlación que se hizo entre las necesidades y los requerimientos funcionales, lo que permitió encontrar las necesidades con más relevancia en el proyecto así como los requerimientos más importantes para cumplir con las exigencias del problema y desarrollar un producto de calidad.

De acuerdo a esto, se tiene que las necesidades más importantes son:

- Respuesta rápida del algoritmo
- Lenguaje de programación estándar
- Uso de inteligencia de enjambres
- Que logre control sobre una planta
- El controlador debe ser un PID

Para poder suplir estas necesidades, la casa de la calidad nos muestra que los requerimientos funcionales sobre los que se debe concentrar el esfuerzo de diseño son:

- Robustez
- Software ergonómico
- Técnica de identificación por enjambres
- Tipo de Controlador

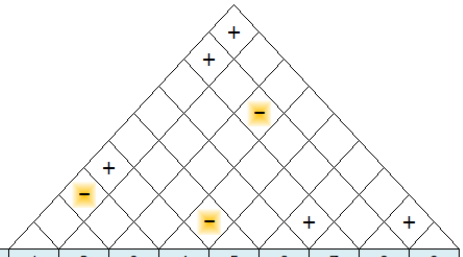
Analizando los anteriores requerimientos es posible determinar que están direccionados al cumplimiento de las necesidades con más importancia dentro del problema y si se estudia su relación con el objetivo general y los objetivos específicos de este proyecto es posible concluir que el cumplimento de estos llevara al alcance exitoso de estos objetivos.

Tabla 4. Casa de La Calidad

Relacion	
Fuerte	●
Moderada	○
Debil	▽

Direccion de Mejora	
Maximizar	▲
Objetivo	◇
Minimizar	▼

Correlacion	
Positivo	+
Negativo	−
No Correlacion	



Columna #	1	2	3	4	5	6	7	8	9
Direccion de Mejora	◇	▲	◇	▲	◇	▲	◇	◇	◇

Fila #	Grafica de Pesos	Peso Relativo	Importancia	Maxima Relacion	Requerimientos del Cliente \ Requerimientos Funcionales	Velocidad de Procesamiento	Robustez	Capacidad de Almacenamiento	Manejo de Perifericos	Economico	Interfaz Intuitiva	Software Ergonomico	Tecnicas de Identificacion por Enjambres	Tipo de Controlador
1		10%	5	9	Respuesta Rapida del Algoritmo	●	○		○	○			●	●
2		8%	4	9	El sistema es Robusto	○	●	○					●	●
3		8%	4	9	Algoritmo Confiable	○	●				○		●	●
4		6%	3	9	Buena Capacidad de Memoria	○	○	●	▽				●	▽
5		6%	3	9	Conexión con otros Perifericos	○			●	○		●		
6		8%	4	9	De bajo Costo	○			○	●				
7		8%	4	9	De Facil Uso				●		●	●		
8		8%	4	9	Algoritmo Reutilizable				○			●		
9		10%	5	9	Lenguaje de Programacion Estandar							●		
10		10%	5	9	Uso de Inteligencia de Enjambres		○						●	
11		10%	5	3	Que Logre Controlar una Planta		○							○
12		10%	5	9	El Controlador debe ser un PID		●					○		●
					Unidad de Medida	Kb/s	Subjetivo	Kb	Modularidad(Binario)	Pesos	Subjetivo	Norma IEC 631131-3	Particulas, Abejas, Hormigas	PID
					Maxima Relacion	9	9	9	9	9	9	9	9	9
					Importancia Tecnica	194	335	76,5	206	118	94,1	312	371	353
					Peso Relativo	9%	16%	4%	10%	6%	5%	15%	18%	17%
					Grafica de Peso									

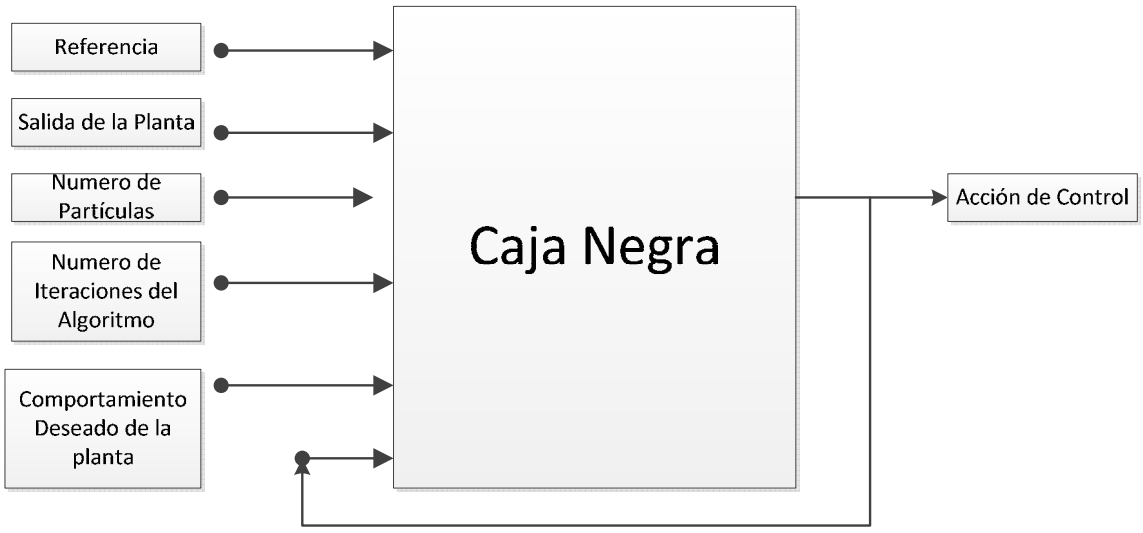
6. GENERACIÓN Y SELECCIÓN DE CONCEPTOS

En esta etapa del proyecto se lleva a cabo un desarrollo conceptual del producto o sistema a diseñar. Considerando los parámetros y necesidades más relevantes que se definieron en el numeral anterior.se puede elaborar un diseño que ejecute las tareas necesarias y a su vez cumpla con los requerimientos que conllevan a la obtención de un producto de calidad.

6.1 CAJA NEGRA

Para comprender de manera más clara lo que debe hacer el sistema a diseñar se plantea el siguiente esquema, llamado caja negra, en donde se relacionan las entradas que el sistema requiere y las salidas que se tendrán. En la figura 7 se muestra este diagrama.

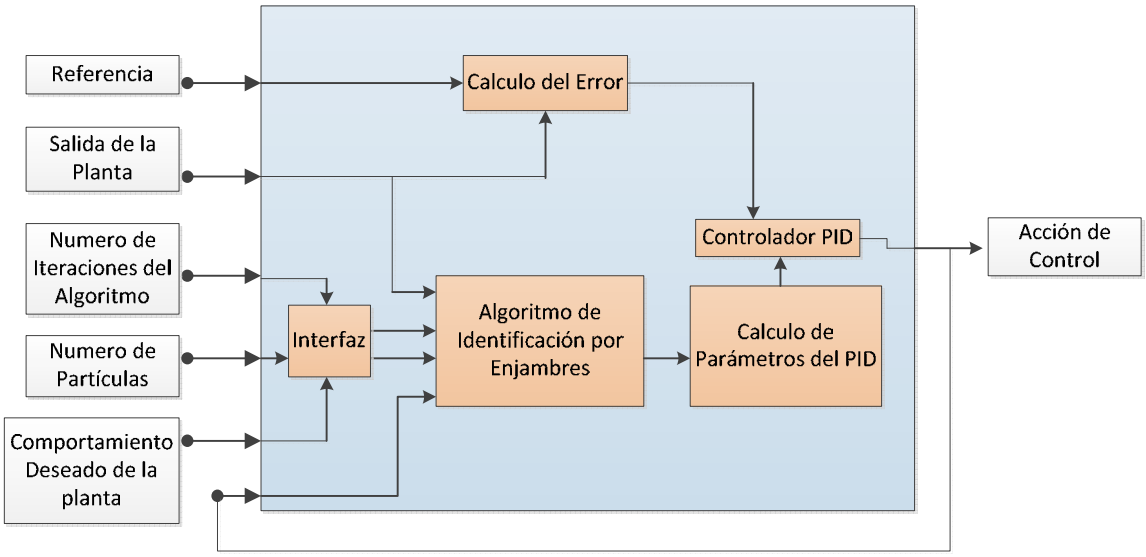
Figura 7. Diagrama de Caja Negra del Sistema



6.2 DESCOMPOSICIÓN FUNCIONAL

Identificadas las entradas y salidas que el sistema debe tener, se plantea ahora un nuevo esquema basado en la anterior caja negra. En este nuevo esquema se evidencian los subsistemas que se necesitan para lograr la salida deseada y se muestra el tratamiento que debe recibir cada señal de entrada al sistema.

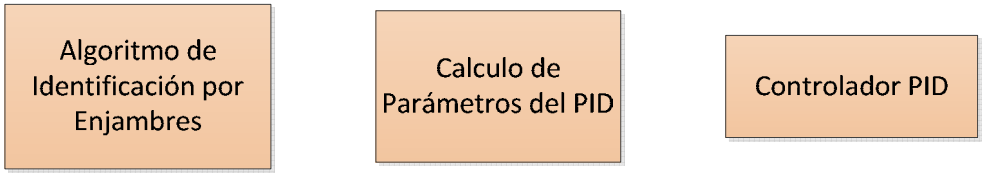
Figura 8. Descomposición Funcional del Sistema



La figura 8 muestra entonces la descomposición funcional del sistema. Así, centrándose en el diseño de cada una de estas partes y siguiendo las pautas que se obtuvieron haciendo el análisis de la calidad se puede obtener una solución al problema que cumpla con todas las necesidades del cliente y satisfaga al mismo tiempo los requerimientos que le dan calidad al sistema.

De esta manera, considerando la conclusión obtenida con el análisis de la calidad se tiene la siguiente rama crítica que define los bloques funcionales sobre los cuales se debe centrar el esfuerzo de diseño. La figura 9 la muestra.

Figura 9. Rama Crítica.



6.3 GENERACIÓN DE CONCEPTOS PARA LAS SUB-FUNCIONES

Considerando la descomposición funcional que se hizo, se hace posible el planteamiento de los métodos o algoritmos que se usaran en cada uno de los bloques descritos en el numeral anterior. Aquí se formulan las diferentes opciones que se tienen para resolver el problema y así posteriormente seleccionar la más conveniente de todas teniendo en cuenta las necesidades del cliente y los criterios de calidad.

6.3.1 Calculo de Error. Para esta función no es provechoso invertir tiempo de diseño ya que consiste únicamente en la diferencia de la referencia o valor deseado con la salida de la planta, algorítmicamente representa una resta entre estos dos valores.

6.3.2 Interfaz. La interfaz será desarrollada haciendo uso de Matlab y su herramienta para el desarrollo de entornos de interacción hombre máquina, GUIDE. Se hará la interfaz bajo estos parámetros debido a que la comunicación entre el PLC y el PC usara la herramienta de OPC contenida en Matlab y por tanto es más provechoso usar el GUIDE de Matlab para esta interacción.

6.3.3. Algoritmo de Identificación por Enjambres. Para la identificación del sistema haciendo uso de inteligencia de enjambres se puede usar:

- Optimización por enjambre de partículas
- Modelos sociales basados en abejas
- Optimización por colonias de hormigas

6.3.4 Calculo de Parámetros de PID. El cálculo de los parámetros de un controlador PID se puede hacer usando dos métodos:

- Analítico
- Experimental

6.4 ANÁLISIS y COMBINACIÓN DE CONCEPTOS

Teniendo cada uno de los posibles métodos o funciones a usar en los subsistemas descritos anteriormente se pueden obtener las siguientes combinaciones:

Tabla 5. Combinación de Conceptos

Algoritmo de Enjambres	Calculo de Parámetros del PID	Controlador
Optimización por enjambre de Partículas	Analítico	Controlador PID
	Experimental	
Optimización por Colonia de Hormigas	Analítico	
	Experimental	
Modelos sociales basados en abejas	Analítico	
	Experimental	

Al observar la tabla 5 se encuentra que solo se tienen tres subsistemas (Algoritmo de Identificación, Calculo de Parámetros, Controlador) de los 5 que se plantearon en la descomposición funcional, esto debido a que estos subsistemas hacen parte de la rama crítica y son aquellos sobre los cuales se debe central el esfuerzo de diseño, adicional a esto, los subsistemas de cálculo de error e interfaz no brindan más opciones para su implementación que las expuestas en la descomposición funcional que se realizó.

Analizando las diferentes combinaciones es posible descartar una serie de ellas. Así, las combinaciones que contienen en su estructura el uso de métodos experimentales para el cálculo de los parámetros del controlador se deben descartar, ya que estos métodos se usan cuando no se conoce el modelo matemático de la planta y para este proyecto el modelo de la planta será determinado haciendo uso de inteligencia de enjambres y por tanto se requiere de un método analítico para poder sintonizar el controlador PID. Retirando las combinaciones con el método de cálculo de parámetros experimental tenemos:

Tabla 6. Conceptos a Seleccionar

Algoritmo de Enjambres	Calculo de Parámetros del PID	Controlador
Optimización por enjambre de Partículas	Analítico	Controlador PID
Optimización por Colonia de Hormigas	Analítico	
Modelos sociales basados en abejas	Analítico	

6.5 SELECCIÓN DE CONCEPTOS

Para la selección del mejor concepto se hizo uso de criterios absolutos[25], basándose en la experiencia y conocimientos de las personas encargadas de direccionar este proyecto se definió la combinación más adecuada, considerando también el estado de arte de las diferentes implementaciones realizadas y de los estudios expuestos en los diferentes libros propuestos en la bibliografía. Con estos recursos se decidió que la combinación a implementar es:

Tabla 7. Combinación Seleccionada

Algoritmo de Enjambres	Calculo de Parámetros del PID	Controlador
Optimización por enjambre de Partículas	Analítico	Controlador PID

El algoritmo de optimización por enjambre de partículas se seleccionó gracias a que es uno de los más simples en su estructura algorítmica, esta característica le

[25] NAVAS Andres Felipe. Diseño Mecatronico, Santiago de Cali, Universidad Autónoma de Occidente, Junio 2009.

da ventajas sobre los otros ya que este proyecto está pensado para sistemas electrónicos sin muchas capacidades de procesamiento y almacenamiento.

De igual forma, de acuerdo a la experiencia en la implementación de los diferentes algoritmos propuestos en la generación de conceptos, la optimización por enjambre de partículas ofrece un menor tiempo de ejecución y alcanza a ofrecer buenos resultados cuando sus parámetros de funcionamiento se limitan, brindando muy buenas aproximaciones al modelo de la planta.

El cálculo de los parámetros del PID se hará de forma analítica ya que es la manera más coherente considerando que se obtendrá el modelo matemático del sistema a controlar y por tanto existe la posibilidad de calcular estos parámetros por asignación de polos, es decir, proponer un comportamiento deseado para obtener una ecuación e igualar esta ecuación con el modelo matemático encontrado por el algoritmo de inteligencia de enjambres y así definir los parámetros que permiten en la planta el comportamiento deseado. El método experimental se usa principalmente en casos donde se tiene conocimiento del modelo de la planta y no es este el caso.

7. ESPECIFICACIONES FINALES

Luego de seleccionar la combinación más adecuada se tienen los siguientes elementos y subsistemas que conformaran este proyecto

7.1 ESPECIFICACIONES DE HARDWARE

El PLC que se usara en este proyecto es de la compañía Allen Bradley y fue puesto a disposición por Robotek Ltda. Debido a esto no se hizo un análisis exhaustivo del PLC a utilizar, sin embargo, al estudiar las características de este equipo es posible encontrar que brinda las condiciones necesarias para poder ejecutar el algoritmo sin inconvenientes.

La familia del PLC es la denominada familia Logix5000 y su referencia hace parte de los compact-Logix, familia cuya principal característica es su integración de módulos de entrada y salida y CPU en un mismo bloque, sin embargo dentro de esta familia se encuentran también sistemas modulares para ser ampliados con otros bloques funcionales como entradas y salidas tanto análogas como digitales o módulos de comunicación para la integración de computadoras u otros PLC.

Para este proyecto se usara un PLC CompactLogix L32E cuyas características técnicas se muestran a continuación[26]:

- Configuración modular y rápida expansión, posibles a través de módulos de interfaz
- Amplitud modular a través de módulos digitales, analógicos, simulación y módulos de función que permiten la comunicación con otros tipos de módulos
- Procesamiento de fórmulas matemáticas
- Configuración / Programación rápida y sencilla con ayuda del software RsLogix5000
- El PLC a usar cuenta con un módulo de salidas análogas de la familia 1769
- Capacidad de procesamiento de hasta 100 programas / Tarea
- Unidad de procesamiento CompactLogix L32E
- Capacidad de ampliación de hasta 16 módulos
- Capacidad de almacenamiento de 750 Kb

[26] Las características técnicas expuestas en este documento se han extraído de la siguiente fuente: "1769 CompactLogix Controllers User Manual"

7.2 ESPECIFICACIONES DEL LENGUAJE DE PROGRAMACIÓN

Para el desarrollo del proyecto se hará uso de texto estructurado (ST; structured text), aunque el entorno de programación (RsLogix5000) permite el desarrollo de algoritmos en los diferentes lenguajes usados para los PLC expuestos en el marco teórico de este documento se escogió este lenguaje ya que facilita la implementación de algoritmos complejos y extensos. Este lenguaje está basado en PASCAL y su estructura permite el uso de sub-funciones y llamadas a otros algoritmos durante la ejecución permitiendo el desarrollo de un algoritmo modular.

7.3 ENTORNO DE PROGRAMACIÓN

El PLC Allen Bradley que se usara en este proyecto debe ser programado haciendo uso de un entorno llamado RsLogix5000 que permite descargar y cargar los programas en el PLC así como la gestión en la comunicación del PLC con el computador. De igual forma, este software permite el monitoreo de las variables y el comportamiento del algoritmo que se está ejecutando en el PLC así como la visualización del porcentaje de CPU que está siendo utilizado por el algoritmo.

7.4 ALGORITMO DE IDENTIFICACIÓN

De acuerdo al análisis realizado para la selección de conceptos se tiene que el algoritmo a usar para este proyecto es "Optimización por Enjambre de Partículas". Como se mencionó previamente este algoritmo brinda ciertas ventajas sobre los expuestos en el marco teórico de este proyecto, así, los requerimientos del cliente se cumplen haciendo uso de este algoritmo.

7.5 CALCULO DE LOS PARÁMETROS DEL PID

Se usara el método de asignación de polos de tal forma que se plantee el comportamiento deseado de la planta, obteniendo así una ecuación que represente este comportamiento y se pueda igualar con la ecuación de la planta estimada por el algoritmo de optimización por enjambre de partículas.

8. DISEÑO DEL SOFTWARE

Definidos los conceptos y la estructura sobre la cual se desarrollara el proyecto se procede entonces con la parte de la implementación, en esta etapa del documento se expone el diseño y la estructura general del algoritmo escrito en el PLC.

8.1 DESCRIPCIÓN DEL ALGORITMO

El propósito del algoritmo es el de estimar el modelo matemático de un sistema, el cual se desea controlar. La estimación de este modelo se hará haciendo uso de un algoritmo basado en inteligencia de enjambres denominado optimización por enjambre de partículas, así, se realiza una búsqueda de los parámetros de la planta en un espacio bidimensional que corresponde a los números reales.

Al ser implementado en un PLC este algoritmo arroja los parámetros de un modelo discreto de la planta, de esta manera, si el sistema a estimar es estable los parámetros encontrados deben estar al interior del circulo unitario en el dominio discreto, de lo contrario estarán afuera de este.

Como el objetivo de estimar el modelo de la planta es poder sintonizar un controlador PID, el algoritmo contiene en su estructura líneas de código que se encargan de estimar los parámetros Kp, Ki y Kd. Estos valores se calculan haciendo uso de una técnica analítica algebraica que consiste en igualar la ecuación característica del sistema a controlar con una ecuación que describa el comportamiento deseado para la planta, al igualar estas dos ecuaciones se puede encontrar el valor que deben tener las variables Ki, Kp y Kd.

El algoritmo de estimación hará uso de dos vectores de 5 muestras tomados de la entrada y la salida de la planta, de esta manera podrá encontrar los parámetros que mejor describan el comportamiento del sistema. El modelo que se encontrara corresponde al descrito en la ecuación 14.

$$Y(K) = a * Y(K-1) + b * U(K-1)$$

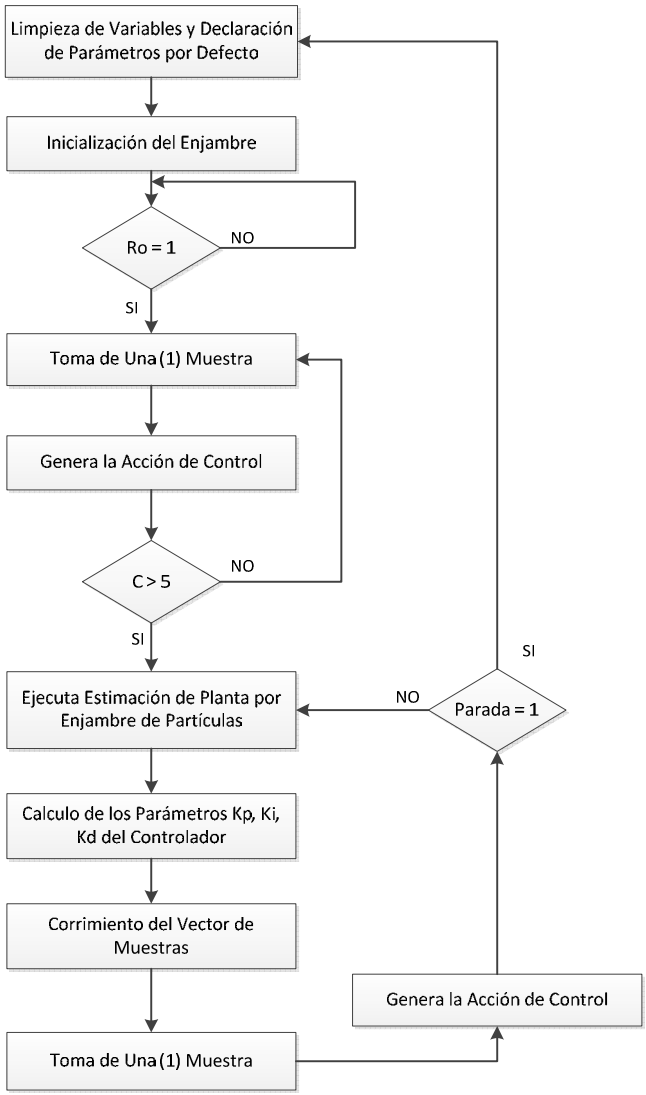
Ecuación 14

Los valores a y b corresponden a los valores que debe arrojar el algoritmo al momento de realizar la estimación. Es importante resaltar que el algoritmo estima

un sistema de primer orden y debido a esto el controlador PID pasa a ser un controlador PI ya que si se usa un esquema PID con un sistema de primer orden se obtendrán tres incógnitas con dos ecuaciones, lo que no permite la obtención de un sistema de ecuaciones con una única solución.

8.2 DIAGRAMA DE FLUJO

A continuación se muestra el diagrama de flujo del algoritmo donde se consideran cada una de las etapas que lo componen así como las variables que se usaran en el mismo.



8.3 ALGORITMO Y SUS VARIABLES

Esta sección busca dar más claridad al funcionamiento del algoritmo. Como surge la necesidad de usar números aleatorios para el correcto funcionamiento del algoritmo y considerando la falta de funciones para la obtención de los mismos, se creó para este proyecto un generador de números pseudoaleatorios que se expone en esta sección. A continuación se describen y definen las variables que se usan en el algoritmo:

Tabla 8. Variables del Algoritmo

VARIABLE	TIPO	DESCRIPCIÓN	
Swarm_s	Entero	Variable de tipo real que define la cantidad de partículas a usar en el algoritmo	Variables Necesarias para el Enjambre
Swarm_P	Real	Es un Vector de 2 Filas y 50 Columnas que guarda las posiciones de las partículas	
Swarm_PB	Real	Vector que guarda las mejores posiciones de cada partícula, tiene las mismas dimensiones de Swarm_P	
Swarm_GB	Real	Vector que guarda las mejores posiciones de todo el enjambre, es un vector de 2 posiciones	
Swarm_V	Real	Vector de 2 Filas por 50 Columnas donde se almacenan las velocidades calculadas para cada partícula	
Random	Real	Almacena el valor enviado por la sub-función RandG	
Iteraciones	Entero	Corresponde al número de veces que se desea ejecutar la estimación de parámetros con el enjambre	
F_P	Real	Guarda el fitness calculado en cada ciclo para cada partícula	
F_PB	Real	Almacena el fitness calculado para las mejores posiciones personales de cada partícula	
F_GB	Real	Guarda el fitness para las mejores posiciones de todo el enjambre	
Error	Real	Almacena el error cuadrático medio calculado en la función Fitness	
Ye	Real	Almacena el valor calculado con el modelo estimado por el enjambre de partículas	

Tabla 8. (Continuación)

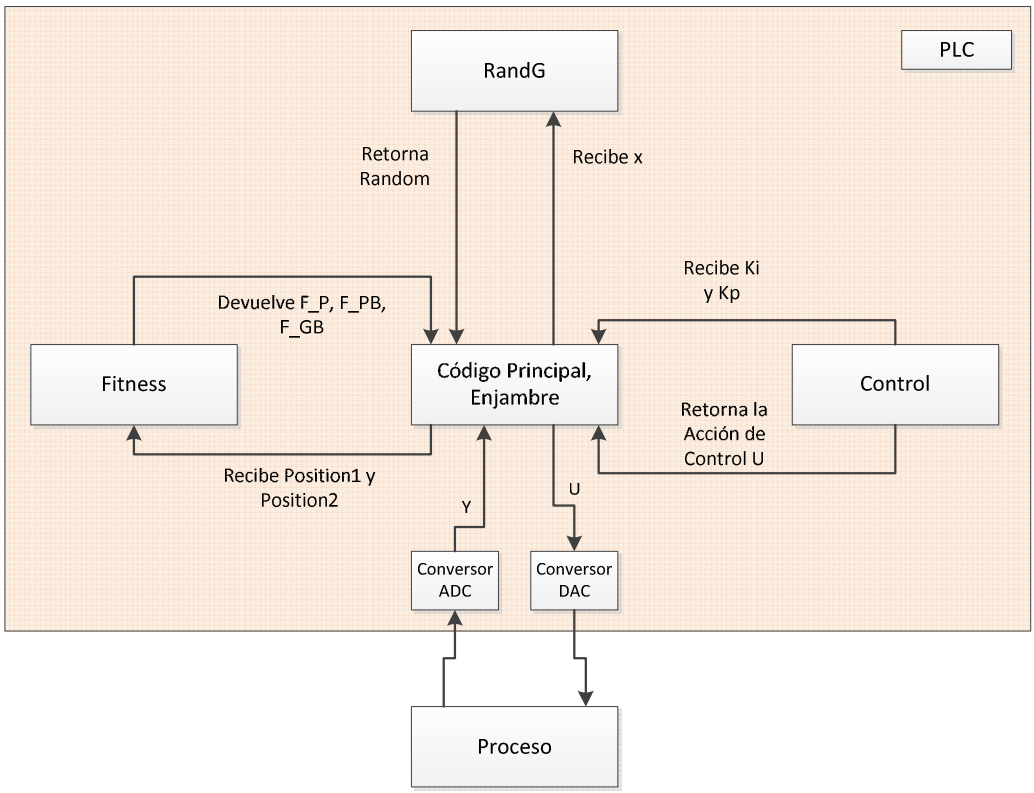
a	Entero	Variables para el cálculo interno de los números Pseudoaleatorios	**Variables para la Generación de Números Aleatorios**
m	Entero		
q	Entero		
r	Entero		
rand	Real	Número real pseudoaleatorio entregado por la sub-función RandG	
Semilla	Entero	Valor que indica el numero en donde se inicia el cálculo de números pseudoaletorios	
Tiempo	Entero	Vector de 7 datos que almacena el tiempo del sistema obtenido con la función GSV de RSLogix	
x	Entero	Almacena el valor de los segundos del sistema	
U	Real	Variables del Procesos, Entrada y Salida de la Planta	**Proceso**
Y	Real		

En la anterior tabla se muestran únicamente las variables que afectan el comportamiento del enjambre y del controlador durante su ejecución en el PLC, no se consideran a acá las variables necesarias para el manejo de los valores enteros que entrega el ADC ni de aquellos necesarios para la conversión hacia el DAC, esto debido a que en esta sección se pretende únicamente exponer el desarrollo código sobre el PLC.

8.4 ESTRUCTURA DEL ALGORITMO

Teniendo en cuenta la necesidad de hacer el algoritmo modular y reutilizable, se desarrollaron sub-funciones para ser llamadas desde el código principal. Con el propósito de dar claridad en la estructura del mismo se ha creado el diagrama expuesto en la figura 10.

Figura 10. Estructura del Algoritmo en el PLC



Las sub-funciones presentes en el algoritmo son:

- **RandG:** Encargada de generar números aleatorios para el funcionamiento del enjambre, recibe la variable x que representa la semilla y retorna un número aleatorio.

- **Fitness:** es de gran importancia para el enjambre ya que calcula el error existente entre el modelo estimado y el comportamiento real de la planta.

- **Control:** Es el encargado de generar la acción de control, necesaria para lograr el comportamiento deseado en la planta.

9. ALGORITMO EN EL PLC

En esta sección se muestra la forma en que se implementó el algoritmo en el PLC, así, se explica cada uno de los bloques que conforman el diagrama de flujo de la sección anterior y se muestra el código desarrollado en el PLC

9.1 LIMPIEZA DE VARIABLES Y PARÁMETROS

Para el correcto funcionamiento del algoritmo es conveniente y recomendable el reinicio de todas las variables (acumuladores, contadores, parámetros de importancia) de tal manera que no se presenten comportamientos no deseados. En la siguiente figura se muestra el código implementado en el PLC para este propósito.

Figura 11. Código para Limpiar e Iniciar Variables

```
IF (Limp = 1) THEN

    FOR i:=0 TO (Swarm_s-1) DO    // Ciclo para limpiar las posiciones del enjambre

        Swarm_P[0,i] := 0;
        Swarm_P[1,i] := 0;
        Swarm_PB[0,i] := 0;
        Swarm_PB[1,i] := 0;
        Swarm_V[0,i] := 0;
        Swarm_V[1,i] := 0;

    END_FOR;

    // Reseteo de los fitness, Error y de los mejores globales
    F_P := 0;
    F_PB := 0;
    F_GB := 0;
    Error := 0;
    Swarm_GB[0] := 0;
    Swarm_GB[1] := 0;

    // Reseteo de variables para el controlador
    C := 0;
    DY := 0;
    Y := 0;
    Uant := 0;
    EI := 0;
    U := 0;

    // Ki y Kp Iniciales
    Ki := 0;
    Kp := 1;

    //Limpieza de Contadores y Variables Ciclos
    c := 0;
    i := 0;
    J := 0;

    Limp := 0;

END_IF;
```

9.2 INICIALIZACIÓN DEL ENJAMBRE

Para que el sistema funcione de la manera adecuada se hace necesario inicializar el enjambre, si no se inicializa, al momento de calcular los parámetros del controlador se obtendrán valores indeterminados y la acción de control calculada no será la adecuada y se ocasionara un mal funcionamiento en la planta.

Figura 12. Código para Inicializar el Enjambre

```
GSV(WallClockTime,,LocalDateTime,tiempo[0]);    // Toma del tiempo del sistema
x := tiempo[6] MOD 100;
Rng.Semilla := x;                               // Inicializacion de la semilla del
                                                // generador de numeros aleatorios

// Ciclo para definir las posiciones iniciales de las particulas
FOR i:=0 TO (Swarm_s-1) DO
    RandG(Rng,Random);
    Swarm_P[0,i] := Random;
    RandG(Rng,Random);
    Swarm_P[1,i] := Random;
END_FOR;

COP(Swarm_P[0,0],Swarm_PB[0,0],100);            // Instruccion para llenar los Swarm_PB

RandG(Rng,Random);                              // Inicializacion de los mejores globales
Swarm_GB[0] := 10*Random;
RandG(Rng,Random);
Swarm_GB[1] := 10*Random;

R [:=] 0;
```

9.3 ENJAMBRE DE PARTÍCULAS

En esta etapa del código el algoritmo se encarga de estimar el modelo matemático de la planta, como se mencionó en los numerales anteriores se estima siempre un modelo de primer orden. El enjambre hace uso de dos sub-funciones que son Fitness y RandG, la segunda se usa también al momento de inicializar el enjambre. El código para la optimización por enjambre de partículas; debido a su extensión, se muestra en la Anexo A.

9.4 CALCULO DE PARAMETROS DEL CONTROLADOR

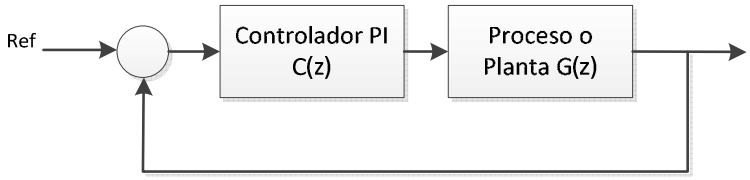
El cálculo de parámetros del controlador solo consta de dos líneas de código, este cálculo se debe realizar después de estimar los valores del modelo matemático de la planta, de lo contrario podrían obtenerse valores para Ki y Kp causantes de comportamientos no deseados en la acción de control y por tanto de la planta. La figura 13 muestra el código usado para el cálculo de estos parámetros.

Figura 13. Calculo de Parametros Controlador.

```
Kp := (-1.844 + 1 + Swarm_GB[0])/ Swarm_GB[1];
Ki := ((0.863 - Swarm_GB[0]) / Swarm_GB[1]) + Kp;
```

9.4.1. Análisis de Ecuaciones para los Parámetros. Para comprender de manera clara porque se usan las ecuaciones ilustradas en la figura 13 se expone el desarrollo matemático que se realizó para llegar a ellas.

Figura 14. Esquema Básico de Control



En la figura 14 se muestra un esquema básico de control en lazo cerrado, ahora bien, si consideramos cada bloque por separado se tienen las siguientes funciones de transferencia:

$$C(z) = \frac{(Kp)Z - (Kp\ -\ Ki)}{Z - 1}$$

Ecuación 15

$$G(z) = \frac{b}{Z - a}$$

Ecuación 16

Considerando el lazo cerrado y la retroalimentación negativa se obtiene la ecuación 17 que muestra el denominador de la función de transferencia en lazo cerrado de la figura 14.

$$1 + \frac{Kp * b * Z - (Kp - Ki) * b}{Z^{\wedge 2} - (1 + a) * Z + a}$$

Ecuación 17

Resolviendo se obtiene:

$$Z^{\wedge 2} + [-(1 + a) + Kp * b] * Z + [a - (Kp - Ki) * b]$$

Ecuación 18

Ahora, considerando una ecuación deseada de segundo orden; ecuación 19, podemos igualar la ecuación 18 con esta y encontrar los valores de Kp y Ki, esto es posible ya que el proceso G(z) puede ser identificado con el algoritmo de inteligencia de enjambres, lo que se traduce en el conocimiento de las variables a y b de la ecuación 18. De esta manera obtenemos las ecuaciones para Kp y Ki y por tanto se hace posible la sintonización del controlador PI.

$$Z^{\wedge 2} + Pd1 * Z + Pd2$$

Ecuación 19

$$Kp = \frac{Pd1 + 1 + a}{b}$$

Ecuación 20

$$Ki = \frac{Pd2 - a}{b} + Kp$$

Ecuación 21

9.5 ACCIÓN DE CONTROL

La acción de control se genera con las líneas de código mostradas en la figura 15.

Figura 15. Código para Calcular la Acción de Control

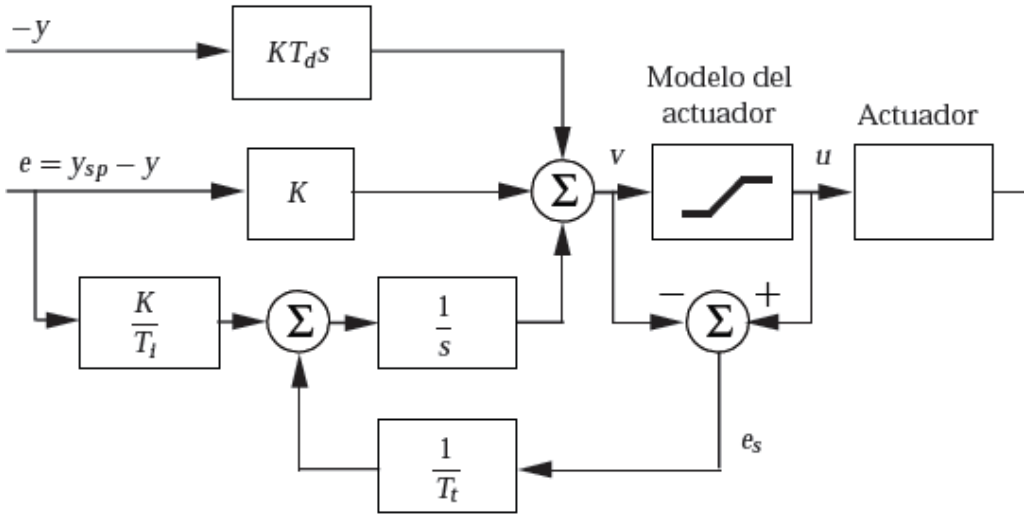
```
SBR();

DY := Ref - Y;
AC := (1 + 0.1)*ACant - 0.1*Uant + DY*Kp + (Ki - Kp)*EI;
ACant := AC;

/// MODELO DEL ACTUADOR PARA EL ANTI-WINDUP
U := AC;
    IF (U > 10) THEN
        U :=  10;
    END_IF;

    IF (U < -10) THEN
        U := -10;
    END_IF;
Uant := U;
///// FIN DEL MODELO DEL ACTUADOR

EI := DY;

RET();
```

Para el cálculo de la acción de control se adiciono un esquema Anti-Windup para evitar la saturación de la acción integral del PI y así lograr el comportamiento deseado del sistema.

La ecuación que define el comportamiento de U se encuentra al definir la ecuación en diferencias para un controlador PI. Usando la ecuación 15 es posible llegar a la expresión de U mostrada en la figura 15.

9.5.1 Anti-WindUp. Como se expuso en el marco teórico de este proyecto, el WindUp es un problema que se presenta en los esquemas de control PID por una saturación del actuador que lleva a un crecimiento exagerado y no deseado de la acción integral. Para el desarrollo de este proyecto se implementó un esquema de Anti-WindUp y así evitar comportamientos no deseados, este esquema se muestra en la Figura 16.

Figura 16. Controlador PID con mecanismo Anti-WindUp



Fuente: Åström, J., Karl, Hägglun, Tore, "Control PID Avanzado", Pearson Prentice Hall, Edición 2, Cap3, Sección 3.5, Pág. 83, 2009

El esquema anterior se puede entonces implementar con una ecuación en diferencias como se muestra a continuación:

$$V = eKp + (Ki - Kp)e(k-1) + \left(1 + \frac{1}{T_t}\right)V(k-1) - \frac{1}{T_t}U(k-1)$$

Ecuación 22

9.6 SUB-FUNCIÓN PARA GENERAR NÚMEROS ALEATORIOS

Dentro de las funciones provistas por el software de programación del PLC no se encuentra alguna que permita la generación de números aleatorios, debido a que estos números ayudan a generar una buena búsqueda en el espacio son de gran importancia para el algoritmo, así, se hizo necesario el desarrollo de un código que permitiera la obtención de estos valores. En la siguiente figura se muestra el código implementado.

Figura 17. Código para Obtener Números Aleatorios

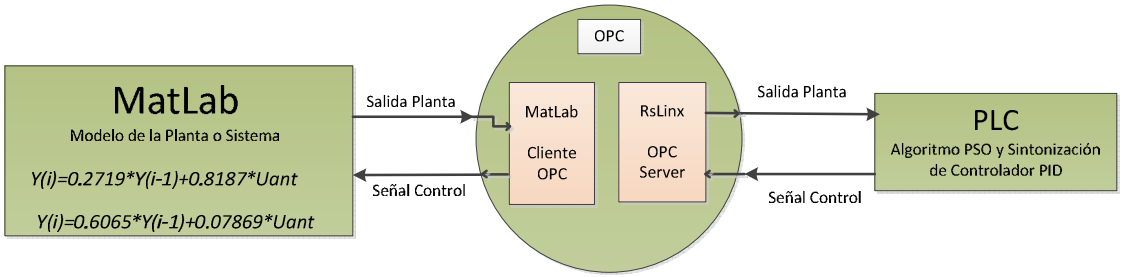
```
IF (Semilla = 0) THEN
    Semilla := 1;
END_IF;

Semilla := a*(Semilla MOD q) - r*Semilla/q;

IF (Semilla < 0) THEN
    Semilla := Semilla + m;
END_IF;

rand := Semilla/m;
```

En el Anexo B se explica a profundidad sobre el funcionamiento del código y su comportamiento.

10. PRUEBA DEL ALGORITMO

Para hacer las pruebas del algoritmo se hizo uso de MatLab y la herramienta de comunicación OPC. El programa de comunicación RsLinx que comunica el PLC con el programa RsLogix 5000 en el ordenador permite la creación de un servidor OPC, por otro lado MatLab tiene un toolbox que permite conectarlo a este servidor de tal manera que se pueda establecer un intercambio de datos entre el PLC y MatLab. La figura 18 muestra el esquema de comunicación.

Figura 18. Esquema de Comunicación entre MatLab y el PLC



Se usó este esquema para probar el funcionamiento el algoritmo debido a la ausencia de un módulo de entradas análogas en el PLC. Dentro de este esquema, MatLab provee el comportamiento de una planta y gracias a OPC se puede interactuar con el PLC de tal forma que se hace posible analizar el comportamiento del algoritmo.

10.1 ALGORITMO COMUNICACIÓN MATLAB Y PLC

La interacción entre el programa desarrollado en MatLab y el algoritmo en el PLC se lleva a cabo haciendo uso de una serie de variables que coordinan el intercambio de datos entre ellos. Esta interacción se debe hacer de esta manera ya que así no se pierden datos debido a la rapidez de ejecución del código en MatLab.

En la Anexo C se muestra el diagrama de flujo que explica el comportamiento del código en MatLab y la forma en que interactúa con el código en el PLC.

10.2 SISTEMAS A CONTROLAR

Para llevar a cabo las pruebas de estimación y del controlador en el PLC se usaron dos modelos de plantas. Estos dos modelos corresponden a dos sistemas existentes en el laboratorio de automática y control en la universidad Autónoma de Occidente y se obtuvieron haciendo uso de la herramienta de identificación de MatLab. Los dos modelos se muestran a continuación:

$$Y(K) = 0{,}8187 * Y(K-1) + 0{,}2719 * U(K-1)$$

Este primer modelo corresponde a un motor de la mesa X-Y del laboratorio, la otra planta que se identificó corresponde al tanque de nivel y se muestra a continuación:

$$Y(K) = 0{,}07869 * Y(K-1) + 0{,}6065 * U(K-1)$$

Gracias al uso de estos modelos se pudo comprobar el correcto funcionamiento del algoritmo PSO y de la sintonización del controlador en el PLC.

10.3 RESULTADOS DE LA PRUEBA

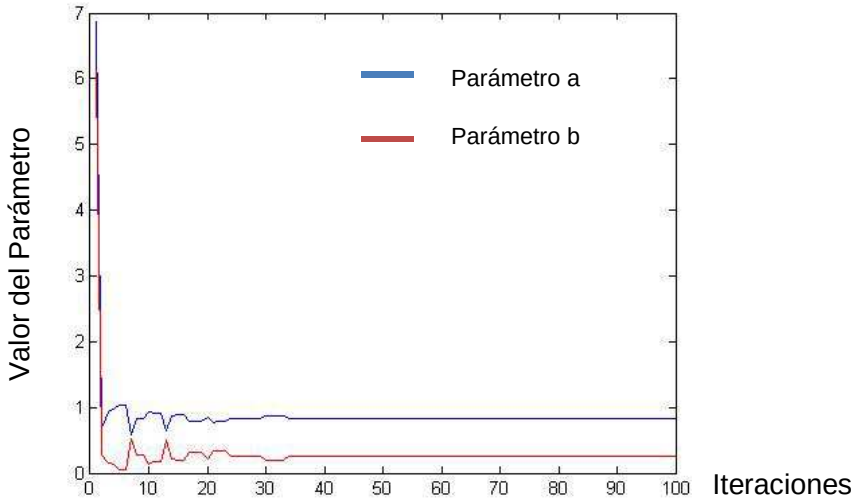
Para el desarrollo del algoritmo se siguió un procedimiento que buscara su implementación de manera gradual y así poder detectar fallas y depurar el código sin mayor dificultad. Considerando esto, se inició la implementación del algoritmo haciendo pruebas únicamente con el enjambre para luego implementar el controlador y la sintonización del mismo.

10.3.1 Algoritmo de Identificación. El algoritmo de identificación es la optimización por enjambre de partículas cuyo funcionamiento ya se ha expuesto en este documento al igual que su código en el PLC. Con las pruebas realizadas con este algoritmo se definieron sus parámetros de funcionamiento y las condiciones bajo las cuales funciona de manera adecuada en el PLC.

Al hacer pruebas con el número de partículas y el número de iteraciones bajo las cuales se obtenía un buen desempeño se encontró que usando 25 partículas y 12 iteraciones del algoritmo se estimaba un muy buen modelo de la planta con el error más pequeño entre todas las pruebas realizadas. El Anexo D muestra una tabla con los resultados de la prueba. Para efectos de la estimación y de las pruebas posteriores se tomó el modelo del motor expuesto anteriormente.

Después de definir estos parámetros de comportamiento se obtuvieron graficas que muestran el comportamiento de los parámetros de la planta estimados. La figura 19 muestra esta gráfica.

Figura 19. Comportamiento Parámetros Estimados.



La anterior grafica muestra como los parámetros estimados logran estabilizarse en un valor. La traza de color azul representa el comportamiento de la estimación del parámetro a y la traza roja representa el comportamiento del parámetro b que componen la ecuación del modelo de una planta de primer orden que se muestra a continuación para recordarlo.

$$Y(K) = a * Y(K-1) + b * U(K-1)$$

10.3.2 Respuesta de Control y Comportamiento del Algoritmo. Una vez encontrado el funcionamiento óptimo del algoritmo de estimación se hicieron pruebas con el algoritmo completo para estudiar de esta manera su comportamiento, en el Anexo E se encuentra el código usado en Matlab para hacer las pruebas de funcionamiento.

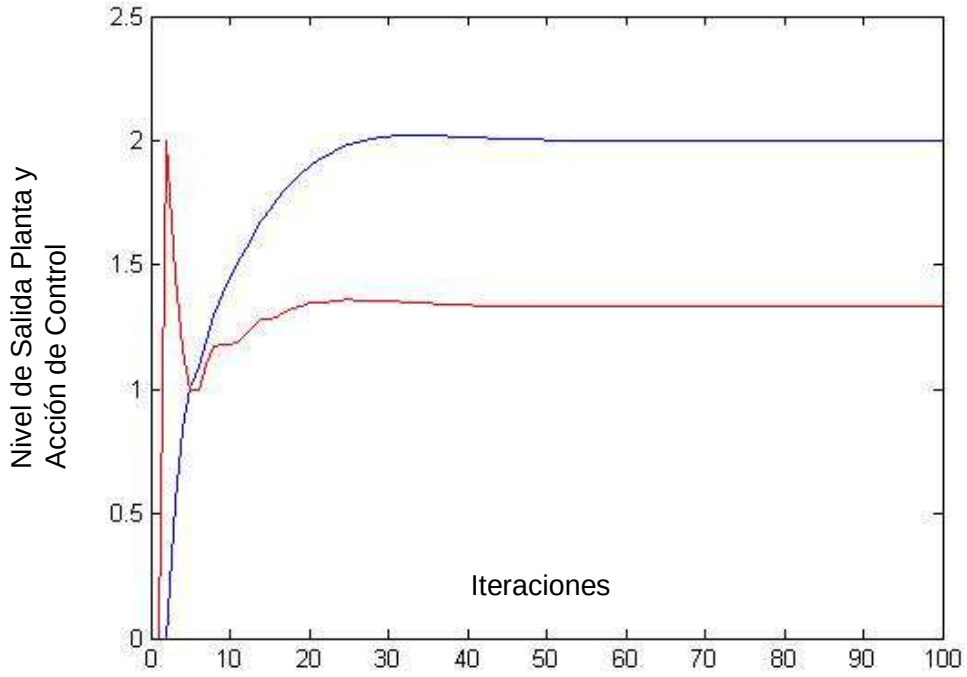
En estas pruebas se obtienen graficas del comportamiento del controlador así como de la respuesta de la planta cuando se definen diferentes referencias, de igual manera se estudia el algoritmo cuando se presenta un cambio de planta mientras este está funcionando y se compara el comportamiento del algoritmo cuando no se tiene el esquema anti-WindUp.

Para las pruebas acá mostradas se usó la siguiente ecuación característica que representa el comportamiento deseado en el proceso.

Tabla 9. Comportamiento Deseado del Proceso

Características Deseadas	Ecuación característica
$M_p = 2\%$ $t_s = 3\ Seg$	$Z^{\wedge 2} + 1.74 * Z + 0.7661$

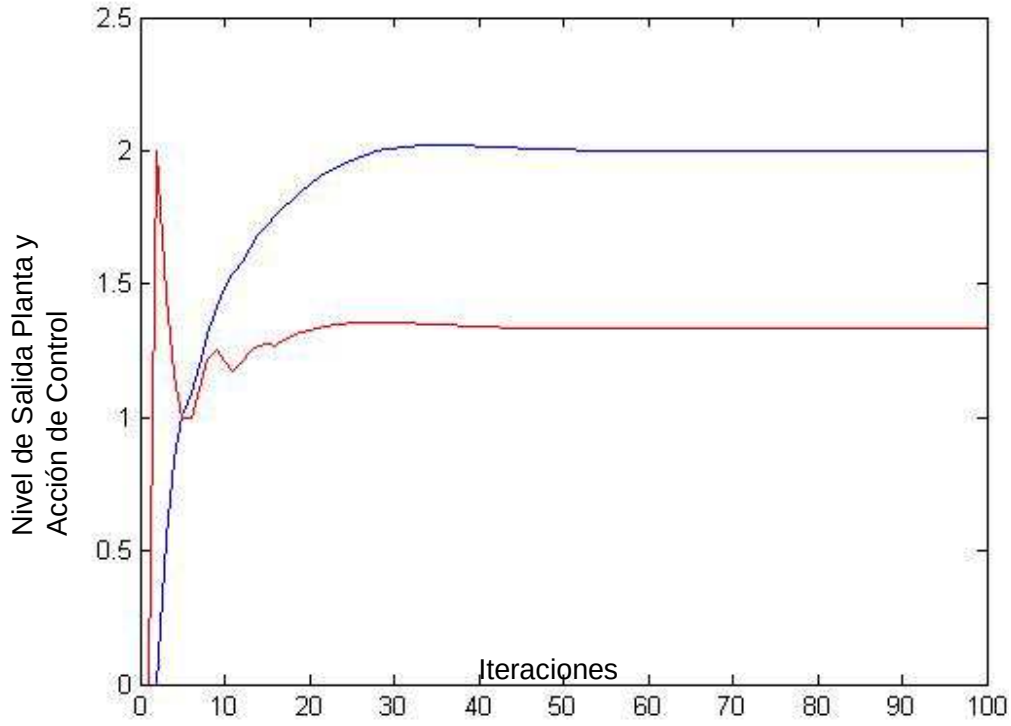
Figura 20. Salida de la Planta y Acción de Control sin Anti-WindUp



La anterior grafica ilustra el comportamiento de la planta y la acción de control generada desde el PLC cuando no se tiene Anti-WindUp, es posible observar que la acción de control oscila al inicio de la estimación y del control, ya que el algoritmo PSO aún no ha encontrado los mejores parámetros que definen el comportamiento de la planta. Sin embargo, el algoritmo logra controlar el sistema hasta encontrar la referencia definida que es 2.

Para comparar el comportamiento del algoritmo con el esquema Anti-WindUp se presente la siguiente gráfica, usando los mismos parámetros deseados.

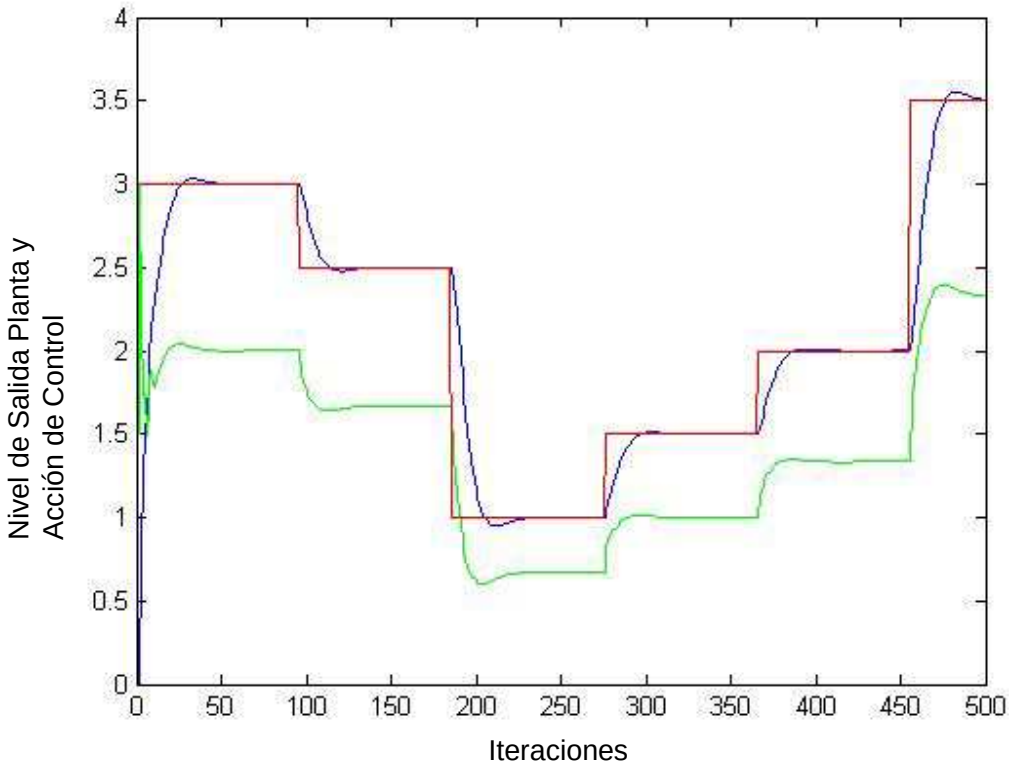
Figura 21. Salida de la Planta y Acción de Control con Anti-WindUp



Es posible determinar que el controlador se comporta de manera similar a cuando no presenta la estructura Anti-WindUp, no se establece por tanto una diferencia notoria entre ambos comportamientos, sin embargo a lo largo de las pruebas posteriores se seguirá usando la estructura Anti-WindUp.

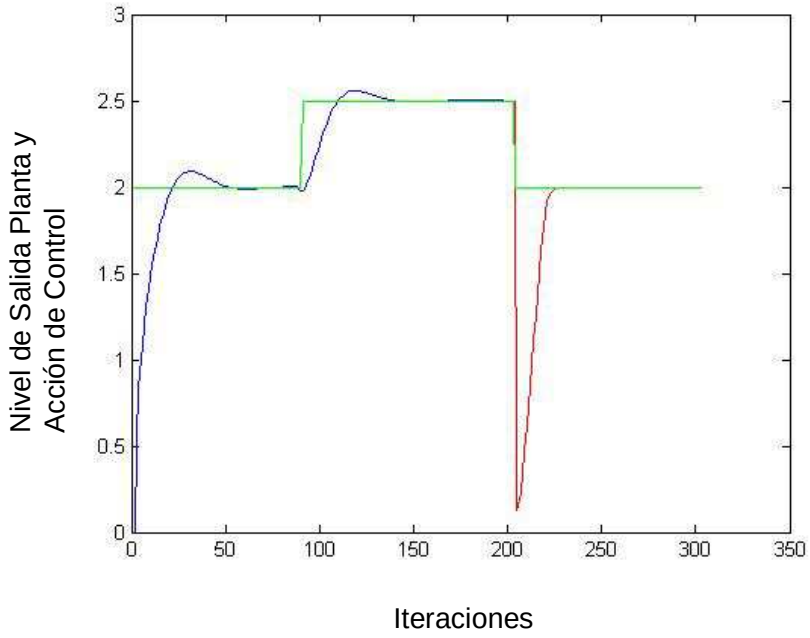
Para poder saber si el controlador responde bien a cambios en la referencia, se hicieron pruebas de tal forma que la referencia se cambiaba durante la ejecución del algoritmo.

Figura 22. Comportamiento Cambio de Referencias con Anti-WindUp.



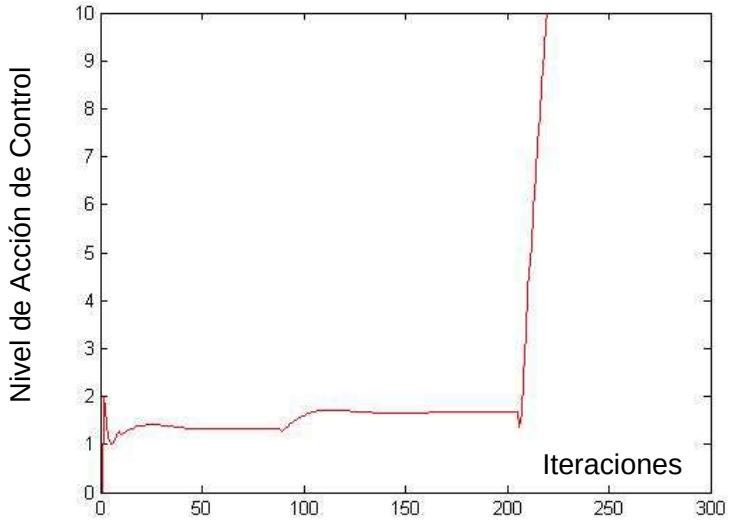
La traza verde representa la acción de control y la traza azul corresponde con el comportamiento de la planta, las referencias que se usaron fueron 3, 2.5, 1, 1.5, 2, 3.5 y se observan con la traza de color rojo. Como se nota en la gráfica, el algoritmo logra controlar la planta sin problemas y seguir las referencias deseadas. Ahora bien, con el propósito de comprobar si el algoritmo de estimación y el algoritmo de control funcionan de manera adecuada ante cambios repentinos del sistema o cambios del sistema, se agregaron líneas de código en matlab para emular una situación donde se hace un cambio entre las plantas.

Figura 23. Comportamiento ante un Cambio de Planta



La figura 23 muestra el comportamiento de las dos plantas. En esta prueba, en la muestra número (200) doscientos se genera un cambio de planta, este cambio ocasiona el comportamiento observado en la gráfica, aun con el cambio de planta se logra estimar de manera acertada los parámetros del nuevo sistema y generar la acción de control necesaria para llevar el sistema hasta la referencia deseada, para esta prueba, la referencia fue de 2 y 2.5 para el primer sistema y de 2 para el segundo sistema.

Figura 24. Acción de Control ante Cambio de Planta.

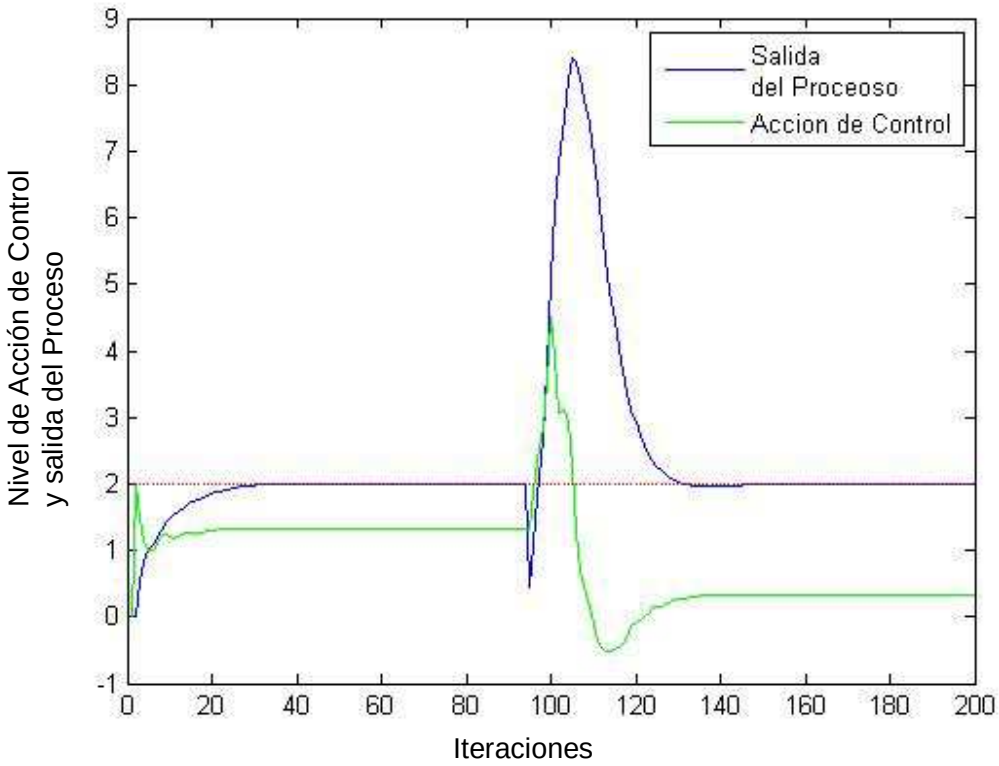


La anterior grafica muestra la acción de control necesaria para ambos sistemas, se observa que no sufre muchas fluctuaciones y que el cambio más brusco se presentó al momento de cambiar el proceso.

Para someter el algoritmo a más pruebas se hizo uso de otro modelo de proceso y así estudiar el comportamiento del algoritmo ante grandes perturbaciones. El modelo matemático del nuevo proceso se muestra a continuación:

$$Y(K) = 0{,}9429 * Y(K-1) + 0{,}3428 * U(K-1)$$

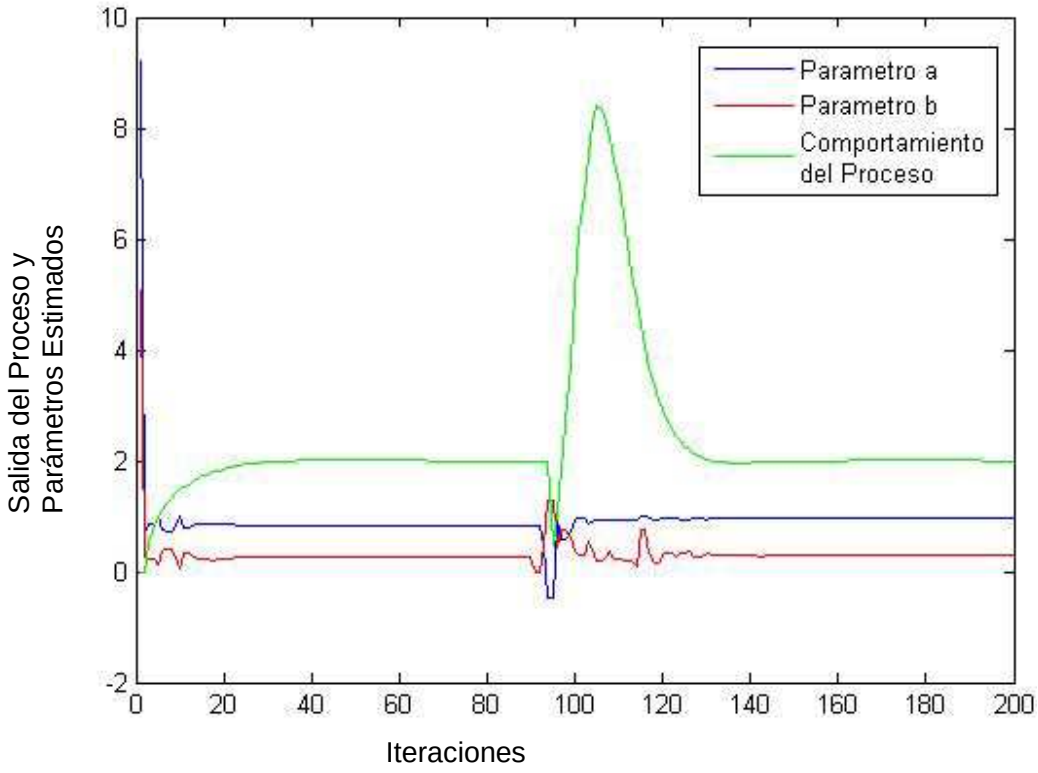
Figura 25. Proceso y Parámetros Estimados Cambio de Planta



Al observar la figura 25 se puede encontrar que al momento de cambiar los procesos; muestra (90) noventa, el algoritmo genera un aumento en el valor de la acción de control, lo que se traduce en una un aumento en la salida del proceso, sin embargo, logra estabilizar el proceso en el punto de consigna definido que es 2.

Para comprender de manera más clara el comportamiento descrito en la figura 25 se tiene la siguiente gráfica.

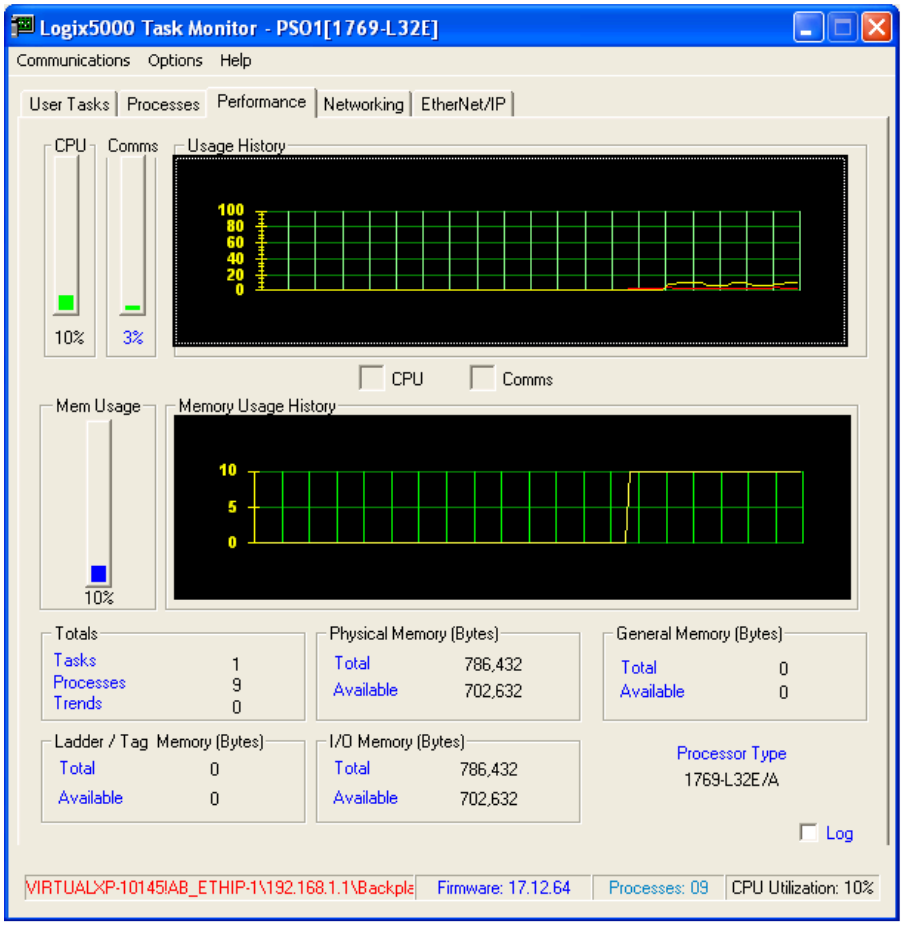
Figura 26. Parámetros Estimados y su Efecto sobre el Proceso



Es posible observar que en el cambio de planta los parámetros estimados se mueven y oscilan de manera brusca, esta posiblemente es la causa del comportamiento no deseado en la salida del proceso.

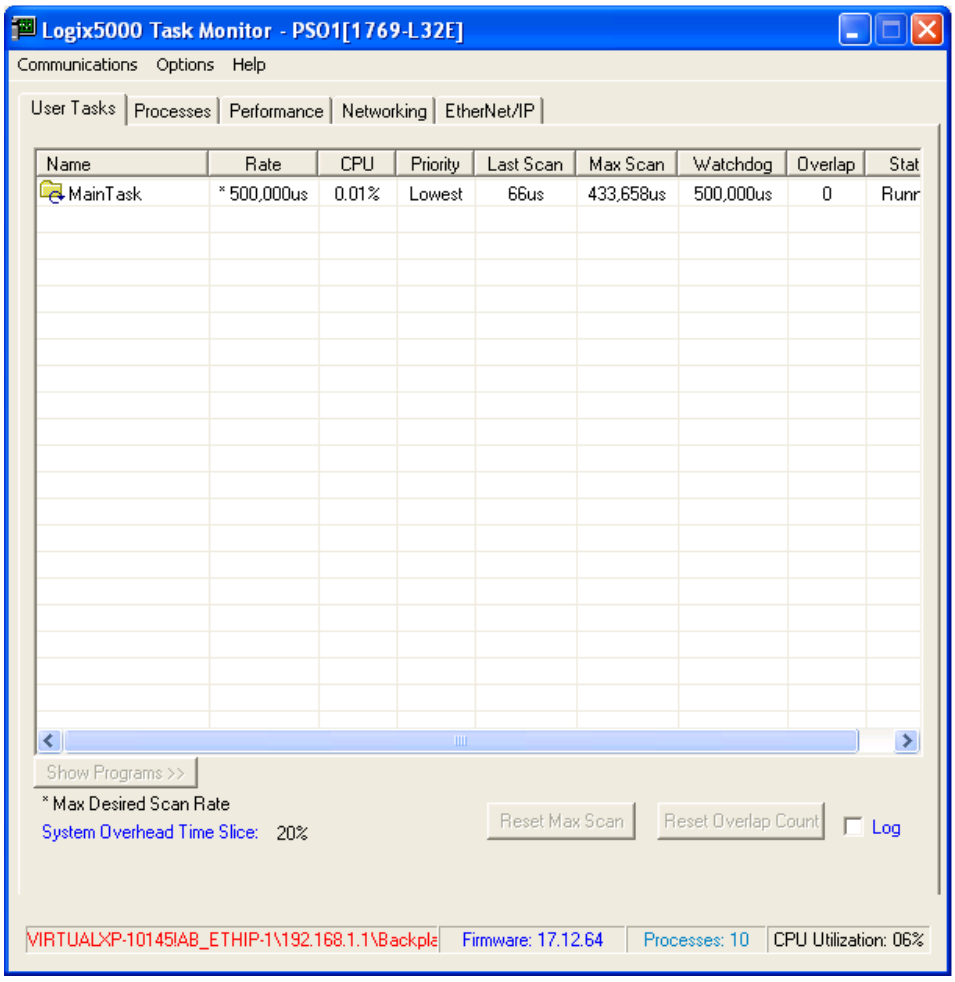
10.3.3 Comportamiento del PLC. Considerando las necesidades de hardware para la ejecución del algoritmo en el PLC se tomaron imágenes de los recursos consumidos por el programa mientras este está funcionando. Esta información la brinda el software RsLogix 5000.

Figura 27. Consumo de Memoria y Comunicación PLC



La anterior grafica muestra que a pesar del gran número de datos que requiere el algoritmo, este solo consume el 10 % del total de la memoria física, lo que muestra las buenas capacidades del PLC usado en esta práctica. De igual forma, se usa únicamente el 10 % del total de la unidad de procesamiento central.

Figura 28. Ejecución Tarea Principal PLC



La anterior figura muestra el tiempo de ejecución del código main del algoritmo, la columna denominada last scan indica el tiempo consumido durante la ejecución del código desde el inicio hasta el final, como se ve, el tiempo es muy pequeño, 66us .

11. ANÁLISIS COMPARATIVO DE CONTROLADORES

Buscando medir el desempeño del esquema de control y estudiar mejor su comportamiento, se desarrolla en este capítulo un análisis comparativo entre un controlador PI convencional sin la auto-sintonización y el esquema implementado en este proyecto.

Para realizar este análisis comparativo se tomaran dos casos para su estudio, el comportamiento del algoritmo y del controlador cuando se tienen cambios en el punto de consigna deseado y cuando se presenta un cambio en la planta a controlar.

De igual manera, se usaran los siguientes indicadores para estudiar el comportamiento de ambos esquemas y comparar su desempeño:

11.1 INDICADORES DE DESEMPEÑO

11.1.1 Índice de Desempeño de Error. Este se encarga de evaluar la diferencia entre el valor deseado y el valor obtenido en la salida. Se definen dos índices, el promedio del error al cuadrado (I1) y el promedio del error absoluto (I2)[27].

$$I1 = \frac{1}{N}\sum_{K=0}^{N}(r(k) - y(k))^2$$

$$I2 = \frac{1}{N}\sum_{K=0}^{N}|r(k) - y(k)|$$

[27] LOPEZ Jesus Alfonso, CASTAÑEDA Oscar, POLANCO ARISTIZABAL German, Análisis Comparativo De Técnicas De Control Convencional E Inteligente Con Los Sistemas De Articulación Flexible Y Bola Biga. En: II Congreso Internacional de Ingeniería Mecatrónica, Octubre 2013 Bogotá, Vol 2.

11.1.2 Índice de Desempeño de Esfuerzo de Control. Compara la diferencia entre una muestra anterior de la señal de control y la actual (I3). Permite conocer que tan bruscos son los cambios de la señal de control[27].

$$I3 = \frac{1}{N}\sum_{K=0}^{N}|u(k)|$$

11.1.3 Índice suavidad esfuerzo de control. Compara la diferencia entre una muestra anterior de la señal de control y la actual (I4). Permite conocer que tan bruscos son los cambios de la señal de control.[28]

$$I4 = \frac{1}{N-1}\sum_{K=0}^{N}\big(u(k)-u(k-1)\big)^2$$

11.2 ANÁLISIS ANTE CAMBIOS EN LA REFERENCIA

Para esta prueba se tomó el siguiente modelo matemático para el proceso:

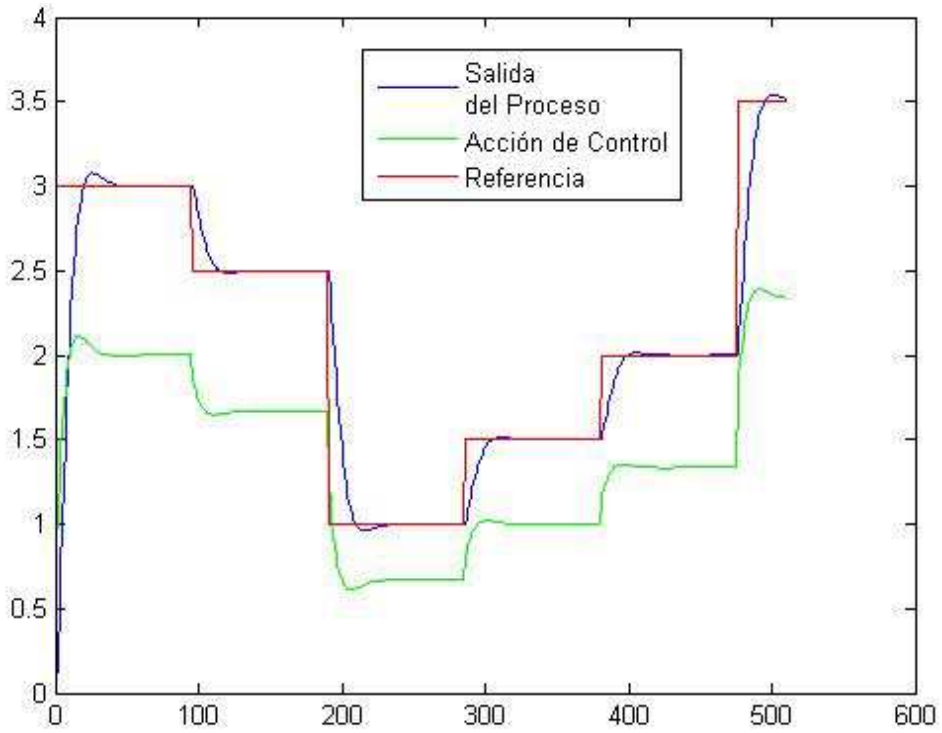
$$Y(K) = 0{,}8187 * Y(K-1) + 0{,}2719 * U(K-1)$$

11.2.1 Controlador PI sin Estimación de Parámetros PSO. Para esta prueba se calcularon los parámetros para el controlador PI, el cálculo de Ki y Kp se llevó a cabo a través de métodos algebraicos encontrando que para los parámetros deseados expuestos en la Tabla 9 se deben usar los siguientes valores:

$$Ki = 0{,}0959$$
$$Kp = 0{,}2894$$

[28] LOPEZ Jesus Alfonso, CASTAÑEDA Oscar, POLANCO ARISTIZABAL German, Análisis Comparativo De Técnicas De Control Convencional E Inteligente Con Los Sistemas De Articulación Flexible Y Bola Biga. En: II Congreso Internacional de Ingeniería Mecatrónica, Octubre 2013 Bogotá, Vol 2.

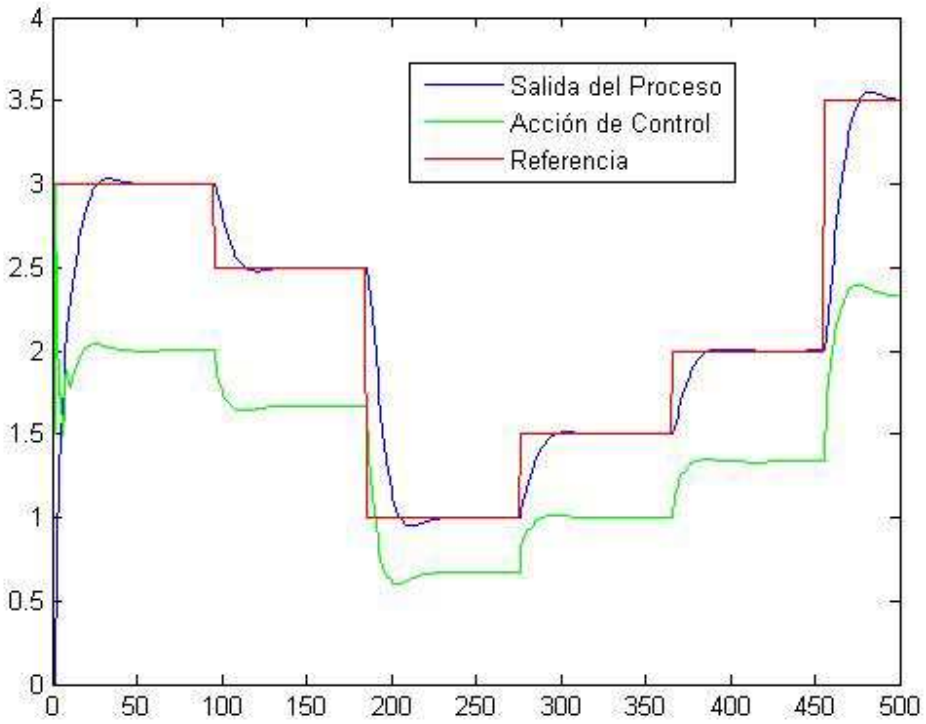
Figura 29. Controlador PI sin Estimación Cambio de Referencia



Se observa que el controlador funciona de manera adecuada y logra seguir cada una de las referencias asignadas mostrando un buen desempeño.

11.2.2 Algoritmo de Estimación PSO y controlador. Para esta prueba se usaron los mismos parámetros deseados y se evaluó el comportamiento bajo las mismas referencias. El siguiente grafico muestra el resultado:

Figura 30. Algoritmo PSO y Controlador ante Cambios de Referencia.



Se puede ver que el comportamiento es bueno y que el algoritmo logra estimar y controlar el sistema aun cuando se presentan cambios en el punto de consigna.

11.2.3 Indicadores para cambios de referencia. Una vez hechos los respectivos ensayos se pueden obtener los siguientes indicadores, organizados en la Tabla 10.

Tabla 10. Comparación Indicadores de Desempeño (Cambio de Referencias)

INDICE	ALGORITMO PSO Implementación	CONTROLADOR PI Implementación
I1	0,1258	0,1321
I2	0,1175	0,1127
I3	1,4222	1,3957
I4	0,0209	0,0015

Con los indicadores podemos observar que ambos ofrecen buenos resultados en lo que respecto al error y el punto de consigna deseado. Sin embargo el algoritmo PSO con el controlador parece ofrecer mejor rendimiento en cuanto al erro. Por otro lado, el controlador PI por si solo brinda un mejor comportamiento en la acción de control, mostrando ser más suave en sus cambios, y un valor promedio menor que el usado por el algoritmo PSO.

11.3 ANÁLISIS ANTE UN CAMBIO EN LA PLANTA

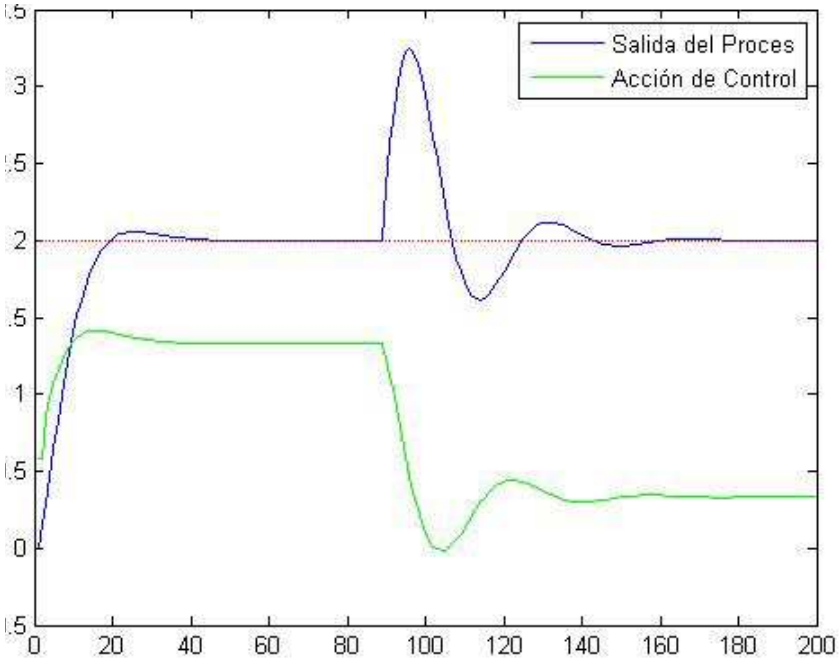
Usando el mismo modelo expuesto en la sección 11.2, se llevó a cabo un cambio durante la ejecución de este modelo y el expuesto a continuación:

$$Y(K) = 0{,}9429 * Y(K-1) + 0{,}3428 * U(K-1)$$

De esta manera, se obtienen los resultados expuestos a continuación.

11.3.1 Controlador PI sin estimación de Parámetros por PSO. Se dejaron fijos los parámetros de los controlador y no cambiaron con respecto a los usados en la sección anterior, para esta prueba se tiene la siguiente gráfica.

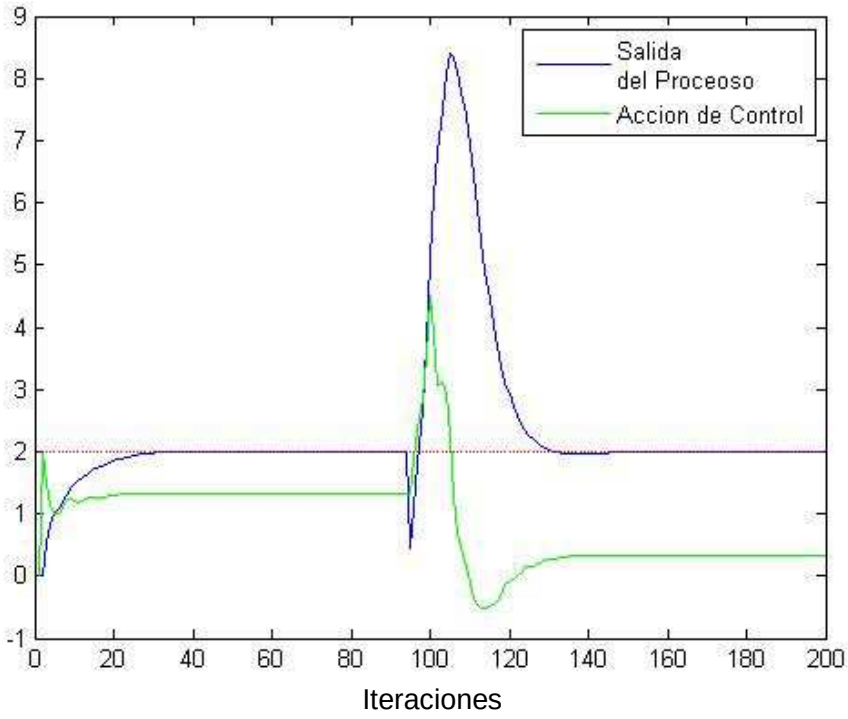
Figura 31. Comportamiento Controlador PI Cambio de Planta



El control funciona bien aun cuando el proceso ha cambiado, presenta oscilaciones cuando se da este cambio y tarde en llegar a la referencia, aun así lo logra.

11.3.2 Algoritmo de Estimación PSO y Controlador. La siguiente grafica ilustra lo que sucede con el algoritmo PSO y el controlador cuando se genera un cambio de proceso durante la ejecución del mismo.

Figura 32. Comportamiento Algoritmo PSO ante un Cambio de Planta



El comportamiento es parecido al visto con el controlador sin la estimación de parámetros, sin embargo para este caso se tiene un sobre pico más grande y el proceso logra llegar a la referencia sin oscilación alguna.

11.3.3 Indicadores ante un cambio de planta. La Tabla 11 muestra la comparación de desempeño entre ambos esquemas usando los indicadores mencionados anteriormente.

Tabla 11. Comparación Indicadores de Desempeño (Cambio de Planta)

INDICE	ALGORITMO PSO Implementación	CONTROLADOR PI Implementación
I1	2,045	0,1688
I2	0,5179	0,1756
I3	0,912	0,7728
I4	0,0487	7,92E-04

El sobre pico que se observa en la figura 31 hace que los indicadores se vean por debajo de los mostrados por el controlador PI, si bien los valores de la tabla 11 difieren considerablemente entre ellos, favoreciendo el comportamiento del controlador PI sin la estimación de parámetros, es posible decir que el algoritmo PSO permite la estimación rápido del modelo del proceso y lleva a cabo el control del mismo sin inconveniente alguno.

12. CONCLUSIONES

Con el progreso de la electrónica se han podido obtener sistemas de gran capacidad computacional, esto da la posibilidad de desarrollar algoritmos más grandes y potentes para la solución de diversos problemas. Anteriormente era difícil encontrar un PLC con tantas capacidades como las de hoy en día, estas nuevas y mejores capacidades han permitido sin duda alguna el desarrollo de este proyecto y su buen funcionamiento.

Dentro del ámbito de control industrial el PLC es el instrumento electrónico por excelencia y la necesidad de desarrollar programas y algoritmos cada vez más versátiles y útiles para diversas plantas o sistemas es más demandante, el uso de algoritmos de inteligencia de enjambres puede brindar solución a problemas de sintonización de controladores así como la oportunidad de controlar sistemas no lineales o con dinámicas muy complejas.

Al momento de la implementación y de las pruebas realizadas se hizo posible observar que estos algoritmos pueden brindar robustez al control de una planta; ya que en las pruebas respondió bien ante cambios y diferentes condiciones y que aun con la gran cantidad de variables que necesita y del aparente alto requerimiento en el procesamiento, el PLC respondió de la mejor manera y no presento inconveniente alguno para ejecutar el algoritmo completo.

Los indicadores de desempeño usados para analizar y comparar el comportamiento del algoritmo implementado en este proyecto dan pie para continuar con la investigación acerca de la implementación de estos sistemas a nivel hardware y encontrar cabida en los diversos campos de la industria, aun cuando el PI mostro mejores características de desempeño no se puede descartar la idea de que el rendimiento del algoritmo se puede mejorar considerablemente y que se pueden desarrollar pruebas en donde el comportamiento de este ofrezca un mejor desempeño que el actual siendo superior a alguna otra técnica de control clásica.

Este proyecto no solo comprobó la posibilidad que existe de implementar complejos algoritmos en un PLC sino también las vastas capacidades que pueden ofrecer al momento de ejecutar los mismos.

BIBLIOGRAFÍA

ARAUJO Lourdes, CERVIGÓN Carlos, Algoritmos Evolutivos, Un Enfoque Práctico, Alfaomega RA-MA, 2009, 332 p.

ÅSTRÖM J. Karl., Adaptative Control Around 1960. En: Decision and Control, Proceedings of the 34th IEEE Conference, Junio, 1996, vol 3, p. 44-49

ÅSTROM J. Karl., Control System Design [en línea]: Lecture Notes for ME 155A. Santa Barbara, California, 2002 [consultado 30 de Enero de 2014]. Disponible en Internet: http://neutron.ing.ucv.ve/eiefile/Control%20I/Astrom_notas.pdf.

ÅSTRÖM J. Karl, HÄGGLUN Tore, Control PID Avanzado, Edicion 2. Madrid: Pearson Prentice Hall, 2009, 488 p.

BENI G., WANG J., Swarm intelligence in cellular robotic systems. En: Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Agosto 1989, p 40-45.

BLUM Christian, MERKLE Daniel, Swarm Intelligence, Introduction and Applications, Berlin, Springer, 2008, 283 p

BOLTON W, Programmable Logic Controllers, 5 Ed. Londres: Newunes, 2009.

CETINA DOMINGUEZ Omar, Una Adaptación del Comportamiento de la Abeja Exploradora en el Algoritmo de la Colonia Artificial de Abejas para Resolver Problemas de Optimización con Restricciones. Trabajo de Grado de Maestría en Ingenieria, Laboratorio Nacional de Informática Avanzada, Facultad de Ingeniería, 2003, 48 p.

Diccionario de Makintrop OPC [en línea], Definition OPC Standard [Consultado el 20 de Febrero de 2015]. Disponible en internet: http://www.matrikonopc.es/resources/dictionary.aspx.

DORIGO Marco, CARO Gianni, Ant colony optimization: a new meta-heuristic. En: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, Agosto 1999, vol 2.

ENGELBRECHT Andries, Computational Intelligence An Introduction, 2 Ed, Londres: Wisley, 2007, 630 p.

FLOWER LEIVA Luis, Controles y Automatismos, 10 Ed. Bogotá: Panamericana Formas e Impresos, 2008.

GARCÍA J. M., Algoritmos Basados en Cúmulos de Partículas para la resolución de Problemas Complejos, [en línea]: Networking and Emerging Optimization, 2006, [consultado 30 de Enero de 2014]. Disponible en Internet: http://neo.lcc.uma.es/staff/jmgn/doc/Memoria_PFC_JMGN.pdf

GOLBERG David Edward, Genetic Algorithms in Search, Optimization and Machine Learning, Londres: Adison-Wesly, 1989, 432 p.

HOEGER Herbert, Generación de Números Aleatorios, [en linea]: Universidad de los Andes, 2014, [Consultado 15 de Febrero de 2014]. Disponible en internet: http://webdelprofesor.ula.ve/ingenieria/hhoeger/simulacion/PARTE4.pdf

HUANG Yourui, QU Liguo, TIAN Yiming, Self-Tuning PID Controller Based on Quantum Swarm Evolution Algorithm. En: Natural Computation Fourth International Conference on, Junio 2008, Vol:6, p 401-404

LOPEZ Jesus Alfonso, CASTAÑEDA Oscar, POLANCO ARISTIZABAL German, Análisis Comparativo De Técnicas De Control Convencional E Inteligente Con Los Sistemas De Articulación Flexible Y Bola Biga. En: II Congreso Internacional de Ingeniería Mecatrónica, Octubre 2013 Bogotá, Vol 2

MAYOL BADÍA Albert, Autómatas Programables. Mora de Toledo: Marcombo S.A., 1988, 123 p.

MAZUERA Diego Alexander, Diseño de una Librería Fuzzy para PLC, Trabajo de grado Ingeniero Mecatronico, Santiago de Cali, Universidad Autónoma de Occidente, Universidad Autonoma de Occidente, 2011, 111 p.

NAVAS Andres Felipe. Diseño Mecatronico, Santiago de Cali, Universidad Autónoma de Occidente, Junio 2009.

O'DWYER Aidan, Handbook of PI and PID Controller Tuning Rules, 2 Ed,Londres: Imperial College Press, 2006.

PARROT C, G.K Venayagamoorthy, Implementation of neuroidentifiers trained by PSO on a PLC platform for a multimachine power system. En: Swarm Intelligence Symposium, Junio 2008, vol 1, p 1-6.

QUIJANO Nicanor, PASSINO Kevin, Honey Bee Social Foraging Algorithms for Resource Allocation, En: American Control Conference, Part I: Algorithm and Theory, 2007.

RIVERA Manuel Manyari, Integrated Online Auto-tuning and Digital Implementation of PID Controllers in Industrial Processes. En: Control and Automation (ICCA), Agosto 2011, vol 9. p 30-34

Guía de Manejo para PLC, Autómatas Programables. SENA (Servicio nacional de Aprendizaje). Santiago de Cali, 2005.

YAHYAEI Mehdi, Increasing the Flexibility and Intelligence of Material Handling through the Factory by Integrated Fuzzy Logic Controller with Programmable Logic Controller. En: 14th IEEE International Conference on Fuzzy Systems, 2005.

YAJUAN Chen, QUINGHAI Wu, Design of PID controller based on PSO algorithm and FPGA. En: Intelligent Control and Information Processing (ICICIP), 2011,p 1102-1105.

ZHU Yan-Fei, TANG Xiong-Ming, Overview of swarm intelligence. En: Computer Application and System Modeling (ICCASM), 2010, Vol 9, p 400-403.

ANEXOS

Anexo 1. Algoritmo de optimización por enjambre de partículas en el PLC.

```
    GSV(WallClockTime,,LocalDateTime,tiempo[0]);
    x := tiempo[6] MOD 100;
    Rng.Semilla := x;


    FOR J:=1 TO Iteraciones DO

        FOR i:=0 TO (Swarm_s-1) DO  // Actualizacion de las mejores posiciones del enjambre

            JSR(Fitness,2,Swarm_P[0,i],Swarm_P[1,i],F_P);       // Calculo Fitness Actual
            JSR(Fitness,2,Swarm_PB[0,i],Swarm_PB[1,i],F_PB);    // Calculo Fitness Mejor Personal
            JSR(Fitness,2,Swarm_GB[0],Swarm_GB[1],F_GB);        // Calculo Fitness Mejor Global

            IF (F_P < F_PB) THEN
                Swarm_PB[0,i] := Swarm_P[0,i];
                Swarm_PB[1,i] := Swarm_P[1,i];
            END_IF;

            IF (F_PB < F_GB) THEN
                Swarm_GB[0] := Swarm_PB[0,i];
                Swarm_GB[1] := Swarm_PB[1,i];
            END_IF;
        END_FOR;

        FOR i:=0 TO (Swarm_s-1) DO

            // VELOCIDAD 1: Codigo para calcular la velocidad de las particulas
            RandG(Rng,Random);
            a1 := Random;
            RandG(Rng,Random);
            a2 := Random;
            Swarm_V[0,i] := 0.5*Swarm_V[0,i] + (1*a1*(Swarm_PB[0,i]-Swarm_P[0,i])) + (3*a2*(Swarm_GB[0]-Swarm_P[0,i]));

            // VELOCIDAD 2: Codigo para calcular la velocidad de las particulas
            RandG(Rng,Random);
            a1 := Random;
            RandG(Rng,Random);
            a2 := Random;
            Swarm_V[1,i] := 0.5*Swarm_V[1,i] + (1*a1*(Swarm_PB[1,i]-Swarm_P[1,i])) + (3*a2*(Swarm_GB[1]-Swarm_P[1,i]));


            // ACTUALIZACION: Se actualizan las posiciones e las particulas.
            Swarm_P[0,i] := Swarm_P[0,i] + Swarm_V[0,i];
            Swarm_P[1,i] := Swarm_P[1,i] + Swarm_V[1,i];
        END_FOR;

    END_FOR;

    Kp := (-1.844 + 1 + Swarm_GB[0])/ Swarm_GB[1];
    Ki := ((0.863 - Swarm_GB[0]) / Swarm_GB[1]) + Kp;

    R1 := 0;
    R2 := 1;
END_IF;
```

Anexo 2. Números Aleatorios

Un paso clave en simulación es tener rutinas que generen variables aleatorias con distribuciones específicas: exponencial, normal, etc. Esto es hecho en dos fases. La primera consiste en generar una secuencia de números aleatorios distribuidos uniformemente entre 0 y 1. Luego esta secuencia es transformada para obtener los valores aleatorios de las distribuciones deseadas. La primera fase es la que nos concierne ahora[29].

Las propiedades deseadas del generador son las siguientes:

1. Deben ser eficientes computacionalmente: dado que típicamente se requieren varios miles de números aleatorios por corrida, el tiempo de procesador requerido para generarlos debe ser pequeño
2. El periodo debe ser largo: periodos cortos limitan la longitud aprovechable de una corrida de simulación porque el reciclaje resulta en una repetición de secuencias de eventos
3. Los valores sucesivos deben ser independientes y uniformemente distribuidos en (0,1): la correlación entre números sucesivos debe ser pequeña y si es significante indica dependencia

Las primeras dos propiedades son relativamente fáciles de implementar. La tercera requiere un conjunto de pruebas estadísticas[29]. El generador usado en este algoritmo es del tipo Congruencial-Lineal.

Generadores Congruenciales Lineales. En 1951, D. H. Lehmer descubrió que residuos de potencias sucesivas de un número tienen buenas propiedades aleatorias[29]:

$$x_n = a^n \mod m$$

[29] HOEGER Herbert, Generación de Números Aleatorios, [en linea]: Universidad de los Andes, 2014, [Consultado 15 de Febrero de 2014]. Disponible en internet: http://webdelprofesor.ula.ve/ingenieria/hhoeger/simulacion/PARTE4.pdf

Los parámetros a y m son llamados multiplicador y modulo respectivamente. Muchos de los generadores actuales son generalizaciones de la propuesta de Lehmer y tienen la siguiente forma[29]:

$$x_n = (ax_{n-1} + b) \mod m$$

Entre los resultados de los estudios realizados con estos generadores se tiene que:

- El modulo m debe ser grande. Dado que los x están entre 0 y m-1, el periodo nunca puede ser mayor que m.

- Para que el computo de mod m sea eficiente, m debe ser una potencia de 2, es decir, 2k. En este caso mod m puede ser obtenido truncando el resultado y tomando en k bits a la derecha.

Dentro de lo desarrollado hasta el momento en el PLC, se tiene una buena base para la implementación del enjambre en el PLC, el primer obstáculo que se encuentra es la falta de una instrucción que permita la generación de números aleatorios para poder inicializar las posiciones de las partículas del enjambre y para el cálculo de las velocidades. Por tanto, se hace necesario desarrollar el algoritmo en el PLC que permita obtener estos números.

Según Herber Hoeger[30] se exponen valores para el diseño de un generador de números aleatorios que cumple las características necesarias para obtener una serie con buenas propiedades pseudo aleatorias, los valores se muestran a continuación.

$$a = 16807$$

$$m = 2147483647$$

$$q = m \text{ div } a = 2147483647 \text{ div } 16807 = 127773$$

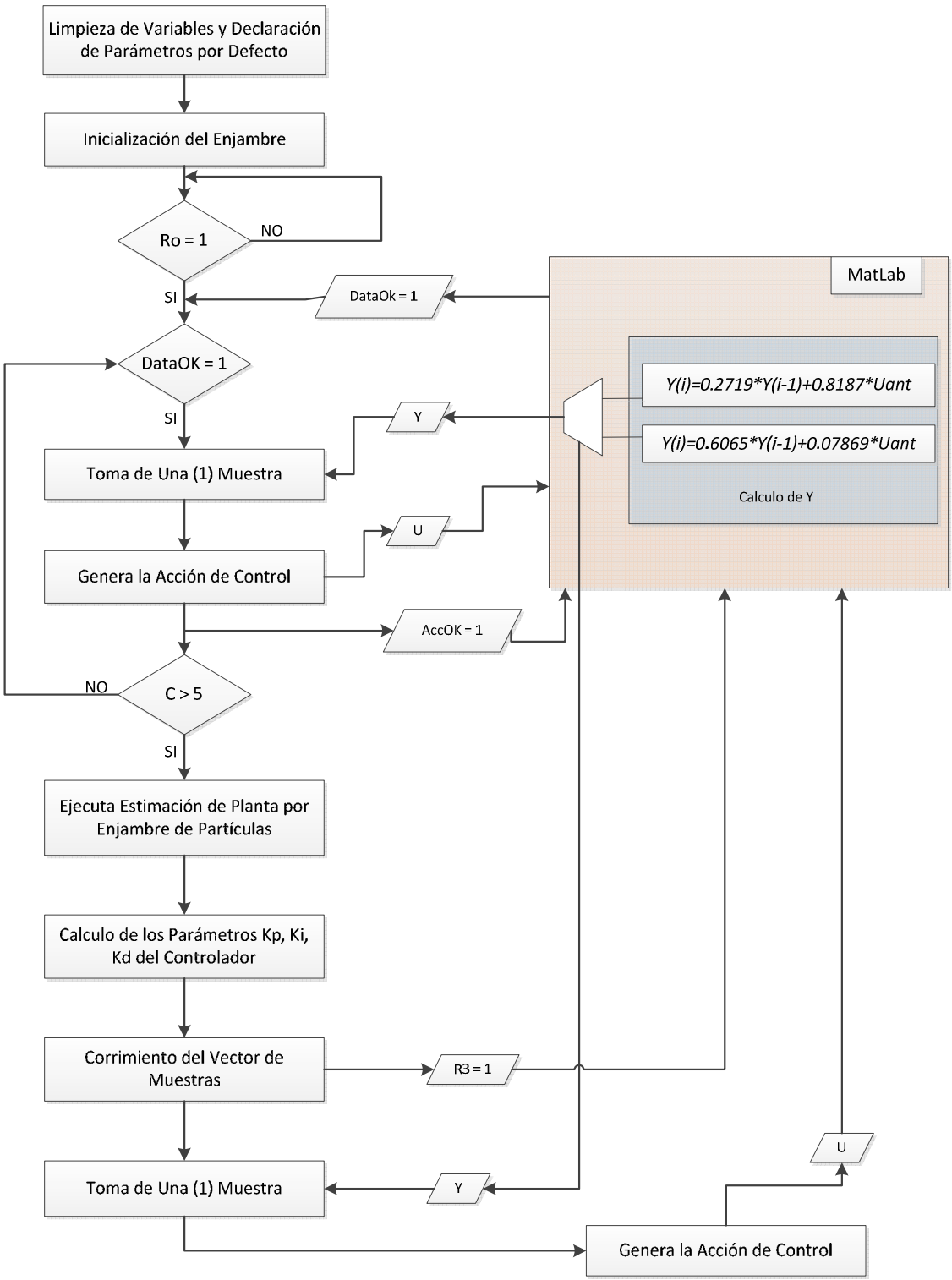
$$r = m \bmod a = 2147483647 \bmod 16807 = 2836.$$

[30] HOEGER Herbert, Generación de Números Aleatorios, [en linea]: Universidad de los Andes, 2014, [Consultado 15 de Febrero de 2014]. Disponible en internet: http://webdelprofesor.ula.ve/ingenieria/hhoeger/simulacion/PARTE4.pdf

El código en el PLC es:

```
IF (Semilla = 0) THEN
    Semilla := 1;
END_IF;

Semilla := a*(Semilla MOD q) - r*Semilla/q;

IF (Semilla < 0) THEN
    Semilla := Semilla + m;
END_IF;

rand := Semilla/m;
```

Anexo 3. Diagrama de Flujo MatLab y PLC



Anexo 4. Resultados Pruebas de Estimación en el PLC

Los resultados obtenidos son los siguientes:

Parámetros del enjambre:

N° Partículas	Iteraciones
50	2

Swarm_GB	{...}
Swarm_GB[0]	0.7278685
Swarm_GB[1]	0.3023308

F_GB	1.27784570e-003
F_P	0.17889646
F_PB	0.15222602

El error global es pequeño, y el valor de los parámetros encontrados está cercano a los esperados.

Parámetros del enjambre:

N° Partículas	Iteraciones
50	4

Swarm_GB	{...}
Swarm_GB[0]	0.78925645
Swarm_GB[1]	0.28979975

F_GB	6.17437836e-005
F_P	0.20963399
F_PB	0.014073446

El error disminuyo, y los parámetros estimados se acercan más a los esperados.

Parámetros del enjambre:

N° Partículas	Iteraciones
50	6

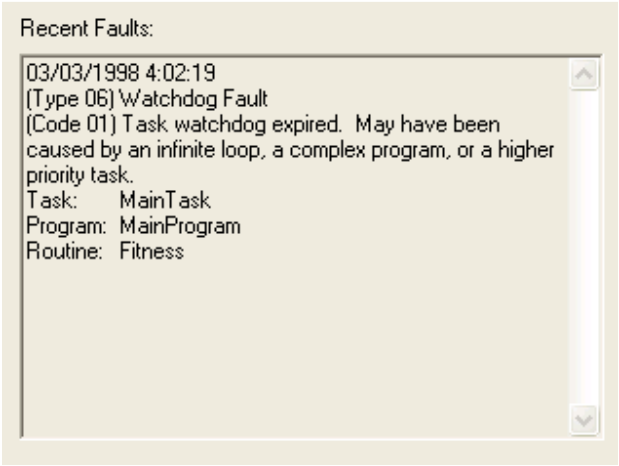
Swarm_GB	{...}
Swarm_GB[0]	0.80986273
Swarm_GB[1]	0.2765277

F_GB	5.59343880e-006
F_P	0.40088454
F_PB	0.033372715

Se logró notar que es mejor reiniciar el enjambre cada vez que este vaya a trabajar, con este número de iteraciones se observa que el error disminuye un poco más y los parámetros se aproximan cada vez más a los deseados.

Parámetros del enjambre:

N° Partículas	Iteraciones
50	8



Al definir estos parámetros, el PLC no responde de manera adecuada y arroja el anterior error, que se debe a la duración del ciclo for que define las iteraciones que hace el enjambre para llegar al mejor global. La duración de este ciclo es mayor al tiempo de scan del PLC por lo que se produce este error.

Ahora, se probara el enjambre aumentando el número de partículas con cada ensayo:

PRUEBA 2

Parámetros del enjambre:

N° Partículas	Iteraciones
10	10

Swarm_GB	{...}
Swarm_GB[0]	0.6059818
Swarm_GB[1]	0.42872113

F_GB	5.19405957e-003
F_P	0.009298223
F_PB	0.008345848

Parámetros del enjambre:

N° Partículas	Iteraciones
20	10

Swarm_GB	{...}
Swarm_GB[0]	0.87560046
Swarm_GB[1]	0.24341121

F_GB	2.45008094e-004
F_P	8.98955856e-004
F_PB	1.00259425e-003

Parámetros del enjambre:

N° Partículas	Iteraciones
25	10

Swarm_GB	{...}
Swarm_GB[0]	0.8165684
Swarm_GB[1]	0.2722199

F_GB	1.57701900e-006
F_P	3.13685462e-003
F_PB	2.41931600e-004

Parámetros del enjambre:

N° Partículas	Iteraciones
30	10

Swarm_GB	{...}
Swarm_GB[0]	0.77896905
Swarm_GB[1]	0.26768732

F_GB	1.25325122e-003
F_P	0.016581714
F_PB	1.99723663e-003

Se observa que el mejor comportamiento se obtiene cuando se usan 25 partículas y 10 iteraciones, por tanto se usaran estos parámetros para el algoritmo.

Anexo 5. Código en MatLab

```
Var_c = true;
v1 = zeros(1,500);
v2 = zeros(1,500);
Yd = zeros(1,200);
AC = zeros(1,200);
Yant = 0;
Um = 0;
p = 1;
c = 1;
ciclo = true;
%%
while(ciclo)

    %%% ecuacion que se simulara; Plant = (0.2719*Uant) + (0.8187*Yant);
    if (Var_c)
        for i=1:5

            % CALCULO DEL VALOR A ENVIAR AL PLC
            f = (0.2719*Um) + (0.8187*Yant);
            Yant = f;

         % ALMACENAMIENTO DE DATOS
            Yd(p) = f;
         % ENVIO DE DATOS AL PLC
            write(Y,f);        % Dato de la planta
            write(DatOK,1);         % Activacion Accion de Control en el PLC

         % Lectura de Datos del PLC
            Acon = true;
            while(Acon)

                Read(AccOK);
                if (AccOK.value == 1)
                    read(U);
                    Um = U.value;
```

```
                    AC(p) = Um;
                    Acon = false;
                    write(AccOK,0);
                end
            end

             p = p + 1;
             pause(0.5)
        end

        write(R1,1);

    end

    Var_c = false;
    ciclo2 = true;


    while(ciclo2)
            % Espera para la terminacion del corrimiento en el vector del PLC
            read(R3);
            read(R1);

            if (R3.value == 1 && R1.value == 0)

                write(R3,0);
                Read(U);
                Um = U.value;
                AC(p) = Um;
                write(AccOK,0);

                write(AccOK,0);

             % Calculo del nuevo dato de la planta
                if (c < 45)
                    f = (0.2719*Um) + (0.8187*Yant);
                    Yant = f;
                else
                    f = 0.07869*Um + 0.6065*Yant;
                    Yant = f;
                end

%                     f = (0.2719*Um) + (0.8187*Yant);
%                     Yant = f;

             % Envio del dato a la ultima posicion del vector en el PLC
                write(Y,f);
                write(DatOK,1);

%              % Almacenamiento de los datos de la planta
                Yd(p) = f;
                p = p + 1;

             % Terminacion del ciclo de espera
                ciclo2 = false;

            end
    end

    write(R1,1);
```

```
94 -        v1(c) = GB1.value;
95 -        v2(c) = GB2.value;
96
97          %c = input('desea continuar ?');
98 -        c = c + 1;
99
100 -       if (c == 45)
101 -          Yant = 0;
102 -          disp('Otra Planta');
103 -       end
104
105 -       if (c > 150)
106 -           ciclo = false;
107 -       end
108
109
110 -        pause(0.5);
111 -   end
112 -   disp('task finished');
113 -   write(R3,0);
114
115
```