IMPLEMENTACIÓN EN DSP DEL ALGORITMO HÍBRIDO PARTICLE SWARM OPTIMIZATION (CONVERGENCIA GARANTIZADA) Y EL SIMPLEX (PSO+SX)

JULIÁN COTE

CAMILO MONCADA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍA FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2010

IMPLEMENTACIÓN EN DSP DEL ALGORITMO HÍBRIDO PARTICLE SWARM OPTIMIZATION (CONVERGENCIA GARANTIZADA) Y EL SIMPLEX (PSO+SX)

JULIÁN COTE
CAMILO MONCADA

Trabajo de grado para optar por el título de
INGENIERO ELECTRÓNICO

Director:
CARLOS RODRIGO CORREA CELY

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍA FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2010

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

TITULO: **IMPLEMENTACIÓN EN DSP DEL ALGORITMO HÍBRIDO PARTICLE SWARM OPTIMIZATION (CONVERGENCIA GARANTIZADA) Y EL SIMPLEX (PSO+SX)**[*].

AUTOR(ES): **JULIÁN COTE, CAMILO MONCADA & RODRIGO CORREA[*].**

Palabras Clave: PSO, simplex, DSP, optimización, híbrido.

DESCRIPCIÓN:

La optimización por enjambre de partículas (PSO) es una técnica de optimización aleatoria basada en poblaciones e inspirada en el comportamiento social de los animales como las bandadas de aves o en cardúmenes. En el presente artículo se describe el desarrollo y la implementación del código en lenguaje C del método de optimización por enjambre de partículas PSO de convergencia garantizada, combinado en topología alternante o paralela con el método simplex Nelder Mead, en un procesador digital de señales (DSP). En el procesador digital de señales y en un computador de escritorio se realizan comparaciones de desempeño, de precisión y de tiempo con las funciones de prueba que se usan convencionalmente en la evaluación de algoritmos de optimización, tales como la función esfera, la función Zakharov, la función Rastrigin y la función Rosenbrock. De esta manera, la programación en lenguaje C en el procesador digital de señales (DSP) del algoritmo se realizó para verificar que es posible implementar una técnica avanzada de optimización que tiene una gran precisión, en un dispositivo que puede ser transportable, de tal manera que se aproveche la diferencia en su tamaño, en su flexibilidad y en su bajo costo con respecto al computador de escritorio.

[*] Trabajo de Grado.

[*] Facultad de Ingeniería Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Rodrigo Correa.

ABSTRACT

TITLE: **IMPLEMENTATION ON DSP OF THE HYBRID ALGORITHM PARTICLE SWARM OPTIMIZATION (GUARANTEED CONVERGENCE) AND SIMPLEX (PSO+SX)*.**

AUTHOR(S): **JULIÁN COTE, CAMILO MONCADA ****.

KEYWORDS: PSO, simplex, DSP, optimization, hybrid.

DESCRIPTION:

The particle swarm optimization (PSO) is a method of random optimization. This technique is based on particle populations and its algorithm is inspired in the social behavior of animals that can be flocks and schools of fish too. This article describes the development and the implementation of the Particle Swarm Optimization method PSO with a guaranteed convergence, that it is mixed in alternant or parallel topology with the Nelder Mead Simplex, on a digital signal processor (DSP). The performance comparisons, precision comparisons and time comparisons are made on the digital signal processor DSP and on the desktop computer with the test functions conventionally used in the evaluation of optimization's algorithms, such as, the sphere function, the Zakharov function, the Rastrigin function and the Rosenbrock function. So it, the programming of the Particle Swarm Optimization (PSO) algorithm in C language on the digital signal processor (DSP) and in the desktop computer is made to verify that it is possible to implement an advanced optimization technique that posses a relatively great precision, on a little device that can be transportable. So it, its size difference, flexibility difference and low cost difference can be usable if it is compared with a desktop computer.

* Degree Work.
** Faculty of Physicomechanical Engineering. School of electric, electronic and telecommunications. Project director: Rodrigo Correa.

INTRODUCCIÓN

Particle swarm optimization (PSO) es una técnica de optimización aleatoria basada en poblaciones, desarrollada por Eberhart y Kennedy en 1995 [3], e inspirada en el comportamiento social de las bandadas de animales como aves o en cardúmenes.

Actualmente el PSO no se utiliza como en 1995, sino que está siendo combinado con técnicas de búsqueda para mejorar su rendimiento y velocidad, [1, 4, 7, 11]. En este artículo se combinó con el método de búsqueda simplex en una topología alternante y se implementó en una tarjeta DSP con el fin de determinar los tiempos de ejecución del algoritmo y además demostrar su utilidad en dispositivos fácilmente transportables.

Los algoritmos evolutivos están siendo programados en diferentes plataformas de desarrollo y el único artículo afín encontrado es el realizado por Parvis Palangpour, et al., en el que se utiliza el algoritmo PSO como estabilizador para el sistema de generación eléctrica, con el propósito de reducir la desviación de la velocidad en dos de los cuatro generadores de energía. Lograron implementar satisfactoriamente el PSO en un DSP (Texas Instruments) como sintonizador para el sistema estabilizador de potencia simulado en tiempo real, reduciendo así las oscilaciones en la velocidad de los generadores y demostrando así la viabilidad del PSO como controlador en un proceso real, [10].

Este artículo muestra la implementación del híbrido Particle Swarm Optimization y simplex (PSO+SX) de topología alternante en un DSP (Motorola), evaluando su desempeño mediante funciones de prueba comúnmente utilizadas para ello. Se obtuvieron resultados favorables en cuanto a la precisión, convergencia y tiempo de ejecución, indicando la posibilidad de utilizar el PSO+SX alternante como

sistema de optimización portable y preciso mediante de un DSP, en procesos donde el uso de un computador es complicado por su ubicación o costo.

El artículo está organizado de la siguiente manera: los métodos de optimización PSO, simplex, PSO+SX, el DSP, así como El PSO mejorado se describe en la sección 3. Cuatro funciones de prueba típicas para algoritmos evolutivos se pueden apreciar en la sección 3. En la sección 4 se detallan los parámetros para la evaluación del desempeño del PSO mostrado en la sección 2 así como tablas y gráficas correspondientes al algoritmo tanto para el computador como para el DSP. Finalmente, se establecen unas conclusiones en la sección 5.

1. FUNDAMENTOS

1.1 MÉTODO SIMPLEX

El simplex es un método de optimización determinístico caracterizado por tener un proceso iterativo, que permite ir mejorando la solución en cada paso. Genera un triángulo, en el caso de dos dimensiones, donde sus vértices son aleatorios en la iteración inicial y cada uno corresponde a una posible solución del problema. El objetivo de cada iteración es trasladar o rotar el triángulo mediante el reemplazo de uno de sus vértices por un nuevo punto que lo acerque hacia el mínimo/máximo de la función objetivo. Con el propósito de cambiar el vértice, se establecen los puntos Best (B), Good (G) y Worst (W) de acuerdo con el valor de la función objetivo y se generan nuevos posibles puntos teniendo en cuenta unas reglas definidas, [19]. En el método simplex Nelder Mead de 1965, las tres reglas que se exponen son la reflexión, la expansión y la contracción del simplex, [9].

1.2 MÉTODO PSO

En este método, se define una población de posibles soluciones (partículas) que se desenvuelven dentro del espacio de búsqueda para encontrar la mejor solución a un problema. Cada partícula explora el dominio de la función objetivo teniendo presente su dirección anterior, la dirección de la mejor posición en que ella ha estado y la dirección de la partícula mejor ubicada. Existen dos ecuaciones que definen el PSO, (1) y (2), obteniendo una velocidad y posición final que serán apropiadas para mover la partícula,

$$V_1 = \left(W.V_0 + r_1C_1(L_{BEST} - X_0) + r_2C_2(G_{BEST} - X_0)\right) * 0{,}729 \qquad (1)$$

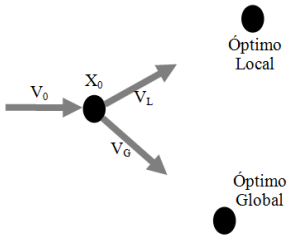
$$X_1 = X_0 + V_1 \qquad (2)$$

de donde:

$V_1 \rightarrow Velocidad\ de\ la\ partícula\ en\ la\ siguiente\ iteración$
$V_0 \rightarrow Velocidad\ de\ la\ partícula\ en\ la\ iteraciónactual$
$X_1 \rightarrow Posición\ de\ la\ partícula\ en\ la\ siguiente\ iteración$
$X_0 \rightarrow Posición\ de\ la\ partícula\ en\ la\ iteraciónactual$
$W \rightarrow Factor\ de\ inercia$
$L_{BEST} \rightarrow Mejor\ posición\ de\ la\ partícula$
$G_{BEST} \rightarrow Mejor\ posición\ de\ todas\ las\ partículas$
$C_1 \rightarrow Coeficiente\ de\ confianza\ relacionado\ al\ L_{BEST};\ C_1 = C$
$C_2 \rightarrow Coeficiente\ de\ confianza\ relacionado\ al\ G_{BEST};\ C_2 = C$
$r_{1,2} \rightarrow Numeros\ pseudoaleatorios\ entre\ 0\ y\ 1$

En la figura 1 se observa la partícula P con una ubicación X_0 y se resalta la tendencia de la partícula a ir hacia las mejores posiciones del espacio de búsqueda sin perder su velocidad anterior.

Figura 1. Velocidad de la partícula P ubicada en una posición X_0.



Fuente: Autores

La ecuación 2 se obtiene acumulando la posición anterior de la partícula con la velocidad que se encontró en la ecuación 1. El factor 0,729 se definió para garantizar la convergencia del algoritmo, [17, 18]. Al pasar por determinada

cantidad de iteraciones, las partículas tenderán al óptimo global y así a la solución del problema de optimización.

1.3 HÍBRIDO PSO CON SIMPLEX (PSO+SX)

Existen dos formas de combinar estos métodos:

1.3.1 Primer híbrido PSO+SX en topología secuencial. El simplex optimiza en el espacio de búsqueda las posibles soluciones de la función objetivo para luego de una cantidad “n” de iteraciones, entregarlas al PSO para que este continúe el proceso partiendo no de posiciones aleatorias sino de unas previamente depuradas; luego de una serie de “m” iteraciones, el PSO retorna los resultados de nuevo al simplex para continuar optimizando la función objetivo. Villareal y Osma, compararon el PSO+SX secuencial con un PSO+SX de evolución paramétrica y los validaron con funciones de prueba con el fin de establecer sus ventajas y desventajas, [20].

1.3.2 Segundo híbrido PSO+SX de topología alternante. Otro híbrido posible del PSO+SX se denomina de topología alternante y su diferencia radica en que cada método trabaja en un espacio de búsqueda totalmente diferente e independiente el uno del otro. El espacio de búsqueda del PSO es el determinado habitualmente por el problema de optimización que consta de las D dimensiones de la función objetivo; por otra parte, las dimensiones del espacio de búsqueda del simplex son establecidas por los parámetros del método PSO y su función es optimizarlos para que el PSO mejore en su desempeño. Posteriormente hay un intercambio de información entre ambos algoritmos para la realimentación de cada uno de ellos.

1.3.3 Desarrollo del algoritmo PSO+SX y su implementación en el DSP. A diferencia de [10], en el que se implementa el PSO original en un DSP, en esta

investigación se desarrolló un algoritmo híbrido de Particle Swarm Optimization (de convergencia garantizada) con simplex, combinados de forma alternante.

Para elaborar el algoritmo, en lenguaje C, se definen las matrices necesarias para el desarrollo del algoritmo, S (Puntos *G, B y W* del Simplex) y B (puntos para las reglas del Simplex) para el método Simplex. Para el PSO se emplean las matrices X (X_i-coordenadas de las partículas del PSO junto con su respectivo valor en la función objetivo), L (L_{Best} - óptimo local de cada partícula con su valor evaluado por la función objetivo) y V (V_0 - velocidad de cada partícula de la iteración anterior) y el vector G (G_{Best} - mejores coordenadas halladas por el PSO junto con su valor en la función objetivo).

La matriz S tienes tres dimensiones ya que en ella están los parámetros de número de partículas (Part), factor de inercia (W) y las constantes de confianza (C), que son valores requeridos por el PSO. En el momento en que se ordene la matriz S respecto a la columna descrita por el PSO se deben aplicar las reglas del simplex como se nombran en [19], teniendo en cuenta que todas las compresiones son en un factor de ½ y las expansiones son en un factor de 2.

Como criterio de parada del proceso de optimización PSO+SX implementado en el DSP se recurrió a:

- Determinar un número fijo de iteraciones que cuando el simplex realice la última iteración automáticamente se termina el proceso.

- Otra forma de finalizar la búsqueda: Cada cinco iteraciones, el método revisará si el valor de la posición $S_{0,3}$ (solución) es menor que la precisión buscada (épsilon) y si esto se cumple, se almacena en un vector de cinco posiciones y se calcula la varianza de los datos. Por otro lado, si ésta es menor que épsilon

elevado al cuadrado, el método se termina ya que no evolucionó en el tiempo y además cumplió la precisión deseada.

Al finalizar el método, la primera fila de la matriz S será la mejor solución encontrada. En la tabla 1 se definen los rangos para los experimentos realizados con las funciones de prueba.

Tabla 1. Parámetros definidos

Parámetros / Dimensiones	***Número de Partículas (Part)***	***Factor de Inercia (W)***	***Coeficientes de confianza (C)***	***Máximas iteraciones del simplex***	***Máximas iteraciones del PSO***
2	*10 ~ 25*	*0,6 ~ 0,98*	*1,5 ~ 3,5*	*50*	*40*
5	*10 ~ 40*	*0,6 ~ 0,98*	*1,5 ~ 3,5*	*60*	*50*

Fuente: Autores

1.3.4 DSP (Digital Signal Processor). *Un DSP es un procesador que se especializa en la ejecución de operaciones de manera digital. En comparación con los microprocesadores, tienen un mejor desempeño de cómputo. El DSP utilizado para implementar el algoritmo de PSO+SX se muestra como una herramienta realmente eficiente ya que está diseñado para las operaciones matemáticas sencillas (acumulación y ponderación), repetitivas (lazos) y de acceso a memoria.*

Como lo que se busca es aplicar el PSO+SX en un sistema de control real, el algoritmo requiere que sea ejecutado de forma continua (en línea); el DSP tiene la ventaja de trabajar sobre señales reales y además de ejecutar el método de optimización, logrando que estos dos procesos se ejecuten en un mismo dispositivo.

DSP PC56F8357PY60. El MOTOROLA DSP PC56F8357PY60 es de la familia de FREESCALE y tiene una frecuencia de 60 MHz, 256 KB de memoria flash de programa, 4KB de RAM de programa, 8 KB de flash de datos, 16 KB de RAM de

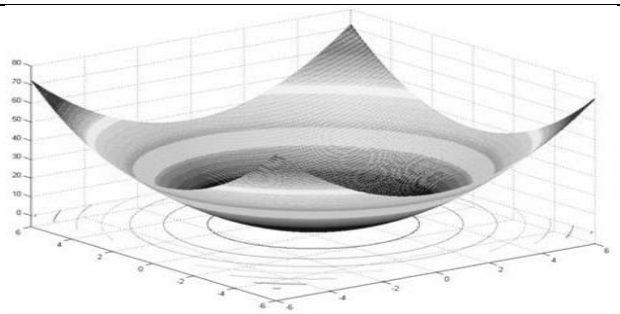
datos, 16 KB de flash de arranque, el formato de manejo de datos es de punto fijo de 16 bits y ejecuta hasta 60 millones de instrucciones por segundo Además, posee periféricos internos para funciones de tiempo, conversión análogo-digital, pines para propósito general y otros módulos de comunicación [15]. La tarjeta DSP es programada por un software para el desarrollo de aplicaciones, ya que es reprogramable y a su vez podría llegar a configurarse de nuevo en caso de ser necesario. Teniendo en cuenta que el DSP es marca MOTOROLA, se utiliza el software *CODEWARRIOR DEVELOPMENT STUDIO* para la serie 56800E. Sin embargo, se recomienda utilizar un DSP con formato de manejo de datos de punto flotante de 32 bits, que su frecuencia sea mayor de 400 MHz, que tenga en lo posible una memoria caché para el procesador y que las memorias tengan una mayor capacidad que el DSP utilizado.

2. FUNCIONES DE PRUEBA

Se evaluó el desempeño del algoritmo Particle Swarm Optimization + Simplex (PSO+SX) con cuatro funciones típicas para valorar algoritmos de optimización [8]. Al realizar experimentos, se debe tener presente el dominio de búsqueda, la posición del mínimo global y su valor en la función objetivo, entre otras características que se resaltan en las figuras 2 - 5.

2.1 FUNCIÓN ESFERA

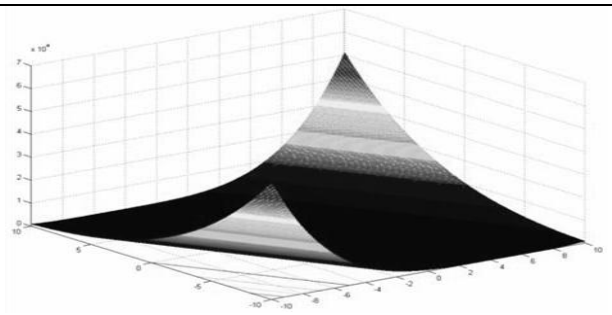
Figura. 2. Parámetros e imagen de la función esfera

Ecuación y gráfica de la función para dos dimensiones			
$f(x) = \sum_{i=1}^{p} x_i^2$ 			
Dominio de búsqueda	**Cantidad de mínimos locales**	**Posición del mínimo global**	**f(x) evaluada en el mínimo global**
$-5{,}12 \leq x_i \leq 5{,}12 \; i = 1,2,3,\ldots,n$	Ninguno	$x^* = (0,\ldots,0)$	$f(x^*) = 0$

Fuente: Test Functions for Unconstrained Global Optimization [8]

2.2 FUNCIÓN ZAKHAROV

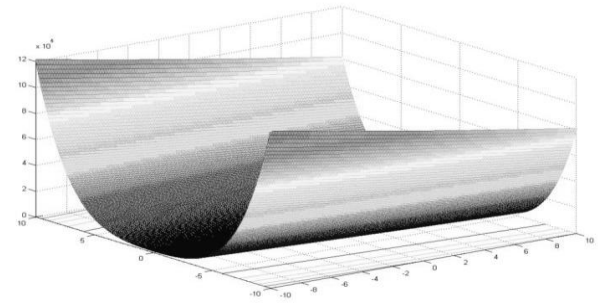
Figura. 3. Parámetros e imagen de la función Zakharov

Ecuación y gráfica de la función para dos dimensiones			
$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$ 			
Dominio de búsqueda	**Cantidad de mínimos locales**	**Posición del mínimo global**	**f(x) evaluada en el mínimo global**
$-5 \leq x_i \leq 10 \quad i = 1,2,3,\dots,n$	Ninguno	$x^* = (0,\dots,0)$	$f(x^*) = 0$

Fuente: Test Functions for Unconstrained Global Optimization [8]

2.3 FUNCIÓN ROSENBROCK

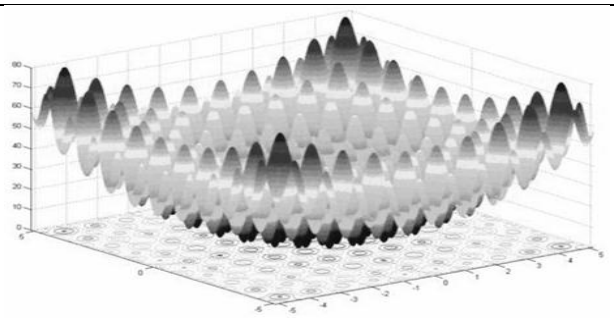
Figura. 4. Parámetros e imagen de la función Rosenbrock

Ecuación y gráfica de la función para dos dimensiones			
$f(x) = \sum_{i=1}^{p-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$ 			
Dominio de búsqueda	**Cantidad de mínimos locales**	**Posición del mínimo global**	**f(x) evaluada en el mínimo global**
$-5 \leq x_i \leq 10 \quad i = 1,2,3,\dots,n$	Ninguno	$x^* = (1,\dots,1)$	$f(x^*) = 0$

Fuente: Test Functions for Unconstrained Global Optimization [8]

2.4 FUNCIÓN RASTRIGIN

Figura. 5. Parámetros e imagen de la función Rastrigin

Ecuación y gráfica de la función para dos dimensiones			
$f(x) = 10p + \sum_{i=1}^{p} ({x_i}^2 - 10\cos(2\pi x_i))$ 			
Dominio de búsqueda	**Cantidad de mínimos locales**	**Posición del mínimo global**	**f(x) evaluada en el mínimo global**
$-5{,}12 \leq x_i \leq 5{,}12 \quad i = 1,2,3,\dots,n$	Varios	$x^* = (0,\dots,0)$	$f(x^*) = 0$

Fuente: Test Functions for Unconstrained Global Optimization [8]

3. EXPERIMENTOS

Con el propósito de comparar resultados en cuanto a precisión se refiere, se decidió realizar las mismas pruebas en un computador y en el DSP.

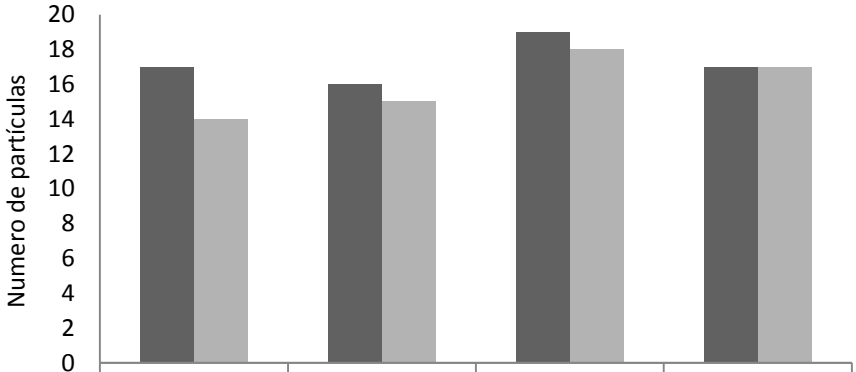
Computador. Se desarrolló el algoritmo en lenguaje C del PSO+SX con el fin de evaluar su desempeño en un computador personal, teniendo una RAM de 4 GB DDR2, un procesador INTEL CORE 2 DUO E 8400 de 64 bits, de punto flotante, 24 GFLOPS, una memoria cache L1 de 64 KB, una L2 de 6 MB, una frecuencia del bus del sistema frontal de 1.333 MHz y una frecuencia de reloj de 3 GHz. Para la programación del algoritmo se utilizó el programa DEV CPP, que es un compilador de código C y C++. Para calcular el tiempo de ejecución del algoritmo se utilizó la función *clock*, que arroja el valor del reloj en ese instante, y para variar los datos de cada prueba, se uso la función *srand*(*semilla*), la cual con cada cambio en el valor de la semilla, el grupo de números aleatorios que crea el compilador es diferente.

DSP. El DSP seleccionado es de marca MOTOROLA, referencia PC56F8357PY60 y una frecuencia de 60 MHz. Se utilizó el programa CODEWARRIOR para la compilación del código en C del algoritmo de PSO. Para medir el tiempo de ejecución del programa, se implementó un anexo al código original que no modifica significativamente el tiempo de ejecución. Mediante la herramienta de Processor Expert®, se utilizó el método de “TimeDate”, el cual configura los *Timers* para obtener un reloj en tiempo real con una resolución de centésimas de segundo.

3.1 RESULTADOS Y ANÁLISIS

Con el fin de evaluar el desempeño del algoritmo desarrollado, se muestran las figuras 6-15, que corresponden a las comparaciones de los parámetros y respuestas del PSO en el PC y en el DSP. En la figura 6 se ilustra el contraste entre el número de partículas utilizadas por cada uno de los dispositivos. Se puede apreciar que el DSP utiliza una cantidad de partículas similar a las requeridas por el computador para hallar el óptimo en las funciones de prueba ejecutadas.

Figura 6. Numero de partículas promedio para dos dimensiones.

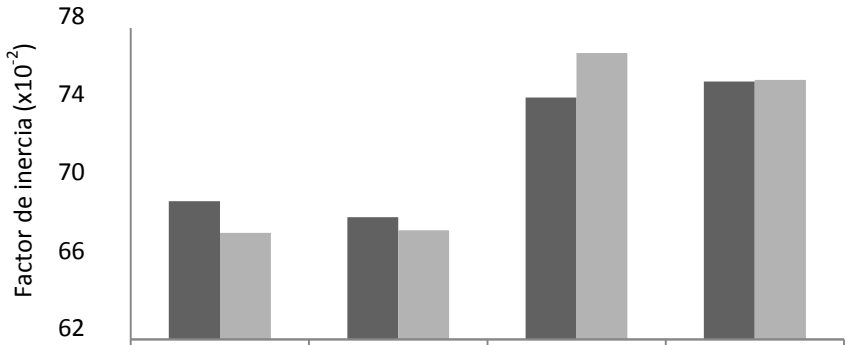


	ESFERA	ZAKHAROV	ROSENBROCK	RASTRIGIN
PC	17,56 ± 5,68	16,90 ± 6,31	19,62 ± 4,37	17,71 ± 4,37
DSP	14,81 ± 3,75	15,09 ± 4,24	18,62 ± 4,64	17,28 ± 4,72

Fuente: Autores

El factor de inercia promedio calculado para todas las funciones de prueba, se observa en la figura 7, y al igual que el número de partículas, no hay una diferencia apreciable en la ponderación de la velocidad para cada función. Sin embargo, a medida que el problema se torna más complejo, el valor de W para el PSO debe aumentar para compensar el efecto de las nuevas pruebas.

Figura 7. Factor de inercia promedio para dos dimensiones.

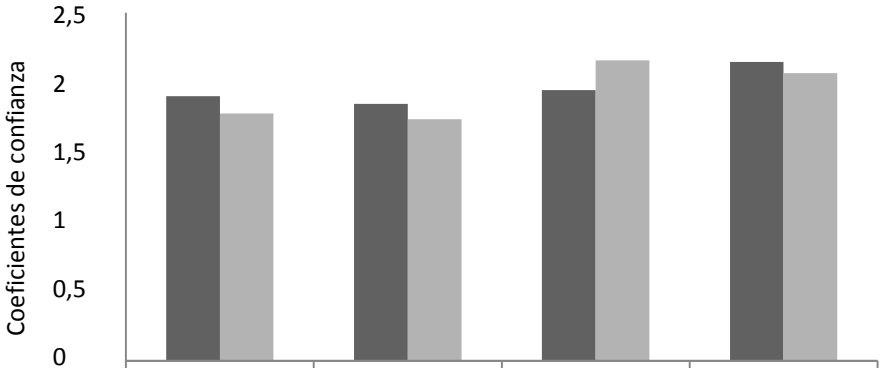


	ESFERA	ZAKHAROV	ROSENBROCK	RASTRIGIN
PC	0,68 ± 0,122	0,67 ± 0,098	0,74 ± 0,118	0,74 ± 0,132
DSP	0,66 ± 0,091	0,66 ± 0,104	0,76 ± 0,113	0,75 ± 0,114

Fuente: Autores

Se presenta la figura 8 para resaltar el comportamiento de los coeficientes de confianza (C) frente a las diferentes funciones objetivo, habiendo definido el rango de dichas constantes oscila entre 1,5 y 3,5 el valor se mantiene en valores cercanos a dos, pudiendo reducir el valor máximo del rango para una búsqueda más directa.

Figura 8. Coeficientes de confianza promedio para dos dimensiones.

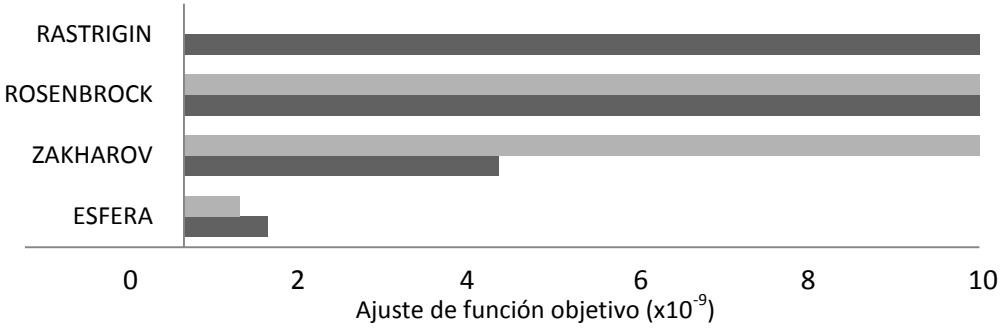


	ESFERA	ZAKHAROV	ROSENBROCK	RASTRIGIN
PC	1,89 ± 0,56	1,73 ± 0,39	1,94 ± 0,58	2,14 ± 0,53
DSP	1,77 ± 0,38	1,84 ± 0,37	2,15 ± 0,59	2,07 ± 0,48

Fuente: Autores

En la figura 9, se puede apreciar que la precisión entre el DSP y el computador se mantuvo en un valor cercano para las funciones esfera, Zakharov y Rosenbrock, pero al momento de evaluar el desempeño de la función Rastrigin, se nota una diferencia considerable, ya que el DSP tuvo valores idénticos a los teóricos. Sin embargo, si se observan los datos obtenidos para esta prueba, ningún dato se ajusta a las coordenadas ideales, cuando se ejecutó en el DSP y éste producía una respuesta inferior a $1x10^{-4}$, automáticamente se aproximaba a cero, mientras que en el computador sucedía el mismo fenómeno, pero cuando la respuesta era inferior a $1x10^{-6}$. Con la función Rastrigin, a pesar de llevar a cabo siempre todas las iteraciones, en ninguna de las pruebas se logró un resultado menor al épsilon establecido.

Figura 9. Ajuste de función objetivo promedio para dos dimensiones.

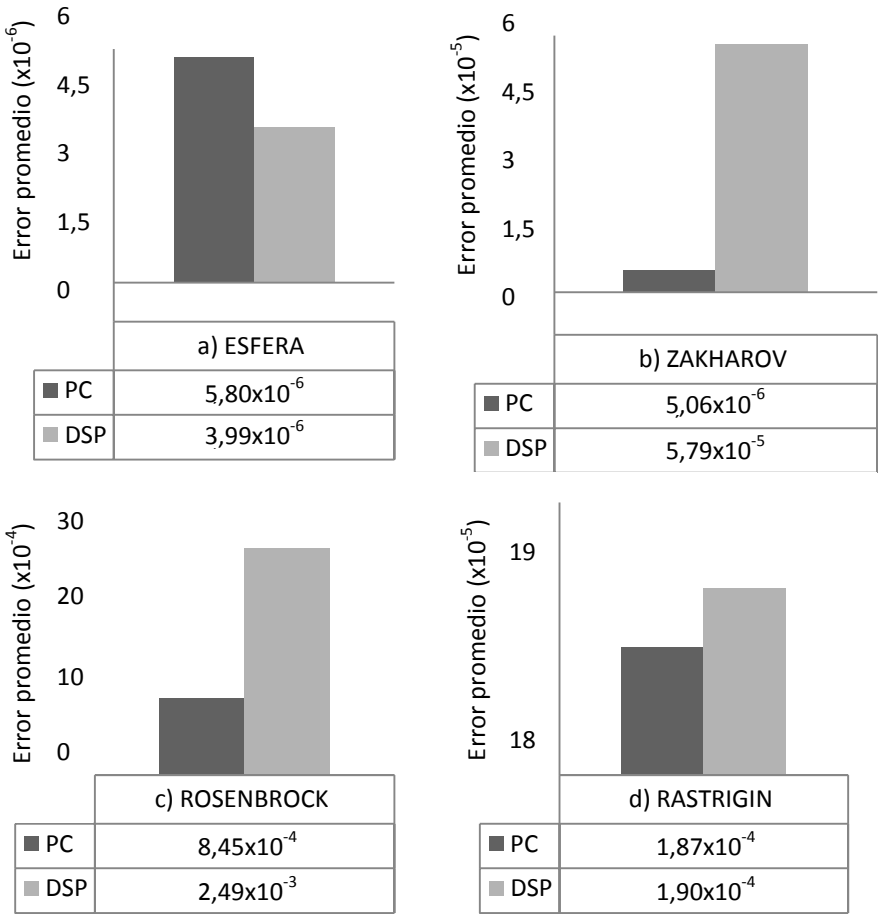


	ESFERA	ZAKHAROV	ROSENBROCK	RASTRIGIN
DSP	$7{,}01\times10^{-10}$	$5{,}14\times10^{-8}$	$2{,}33\times10^{-5}$	$0{,}00\times10^{0}$
PC	$1{,}05\times10^{-9}$	$3{,}95\times10^{-9}$	$1{,}77\times10^{-5}$	$3{,}23\times10^{-4}$

Fuente: Autores

Para las funciones de prueba esfera, Zakharov y Rastrigin, en los dos dispositivos se encontraron valores de coordenadas con una proximidad similar a las coordenadas ideales tal y como se observa en la figura 10. A diferencia de lo anterior, para la función Rosenbrock (figura 10-b), la implementación en el computador, arroja resultados más cercanos al error establecido que en el DSP.

Figura 10. Error promedio en a) esfera, b) Zakharov, c) Rosenbrock y d) Rastrigin.

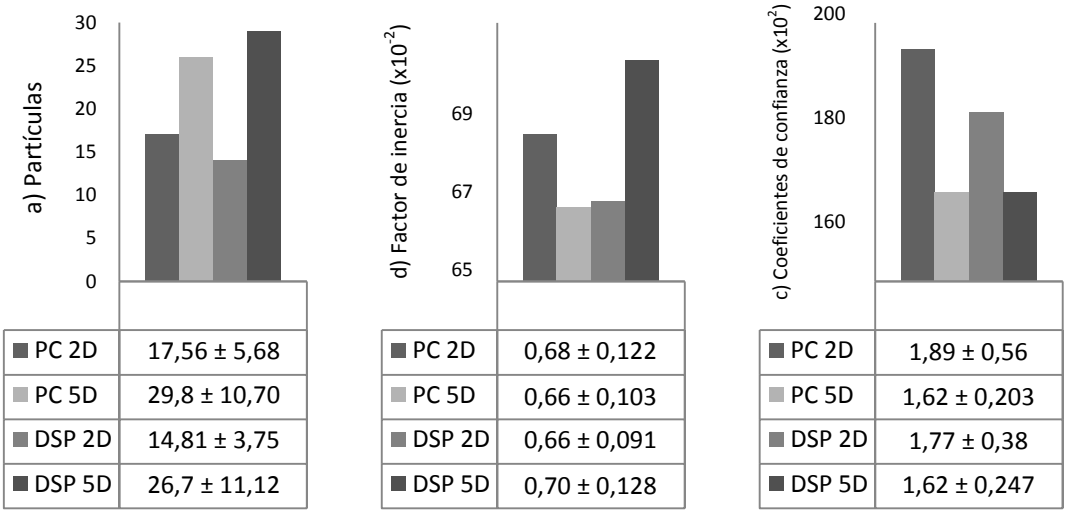


Fuente: Autores

En la figura 11, se muestran los resultados del análisis de parámetros del PSO entre una misma función para dos cantidades de dimensiones diferentes, logrando resultados esperados como el incremento en el número de partículas para el problema con un espacio de búsqueda superior. Se puede apreciar que trabajando con un valor máximo de hasta 25 partículas, el problema en dos dimensiones obtiene la respuesta para un valor de error establecido. No obstante, en el caso de cinco dimensiones, la función no alcanza a converger; se optó por cambiar este

límite para que el método pudiera arrojar valores de precisión aceptable (error≤$1x10^{-10}$).

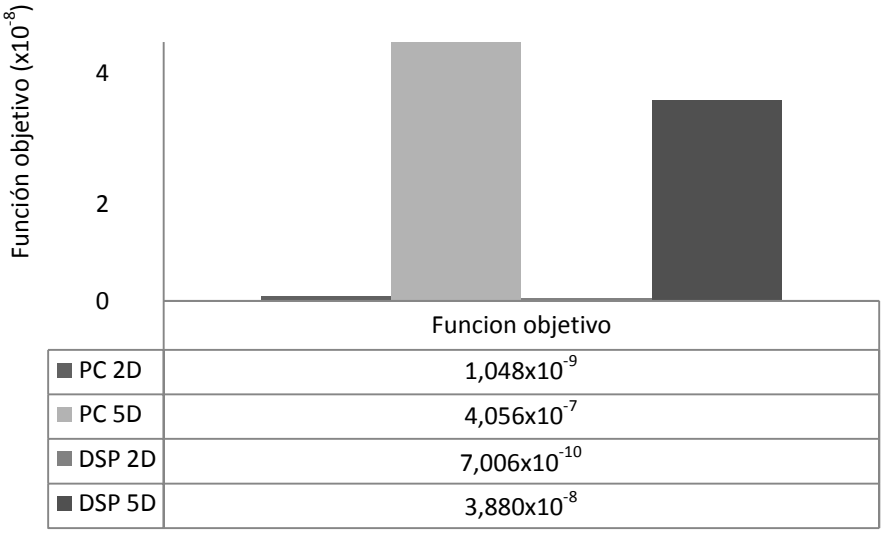
Figura 11. Análisis de parámetros promedio para pruebas de 2D y 5D de la función esfera. a) Partículas, b) factor de inercia y c) coeficientes de confianza.



Fuente: Autores

En la figura 12, se detalla la pérdida de precisión al momento de incrementar el nivel de exigencia. Cuando el problema tiene más dimensiones, el método realiza más iteraciones con el propósito de cumplir la precisión. Para 2D, el número de iteraciones promedio que realiza es de 22 en el DSP y 28 en el computador, mientras que para 5D, en el DSP se elevan a 42 y a 30 en el computador.

Figura 12. Análisis de la función objetivo de dos y cinco dimensiones de la función esfera.



	Funcion objetivo
PC 2D	$1{,}048 \times 10^{-9}$
PC 5D	$4{,}056 \times 10^{-7}$
DSP 2D	$7{,}006 \times 10^{-10}$
DSP 5D	$3{,}880 \times 10^{-8}$

Fuente: Autores

En la figura 13, se evidencia que en el computador, la respuesta se encuentra más cercana al óptimo global que en el DSP. En cinco dimensiones, a pesar que la función objetivo está más alejada comparada con la de dos dimensiones, el valor de sus coordenadas no demuestra estar en un rango fuera de lo aceptable. También se aprecia que en el computador, la respuesta es más cercana a la precisión deseada que en el DSP para cinco dimensiones con iguales parámetros.

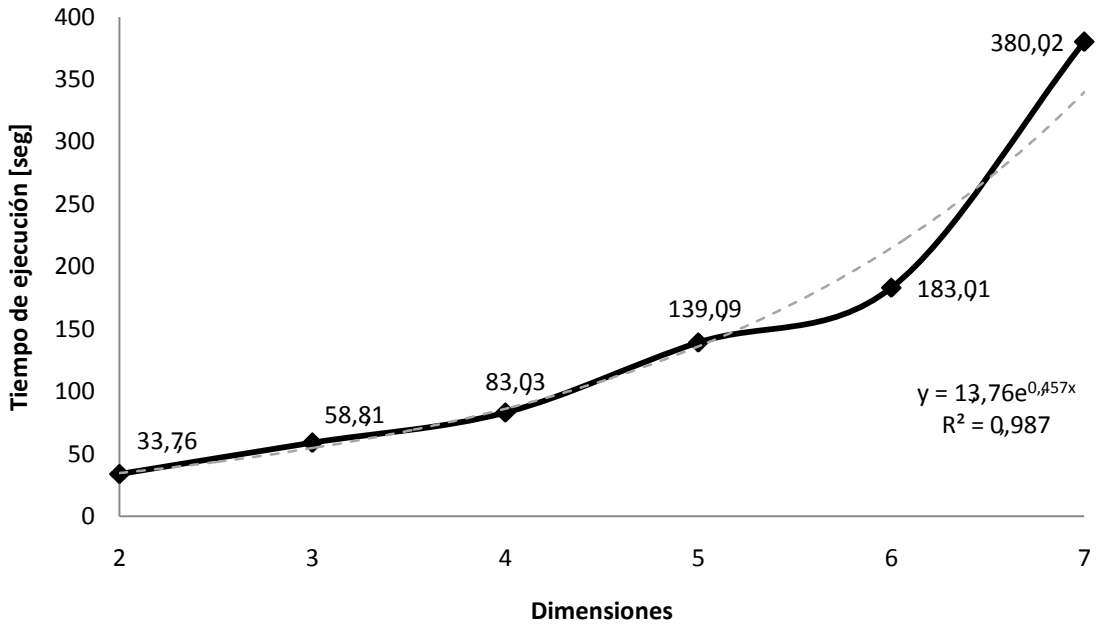
Figura 13. Desviación estándar de la función objetivo de dos y cinco dimensiones de la función esfera.

	Desviación estándar
PC 2D	$5{,}80\text{x}10^{-6}$
PC 5D	$3{,}75\text{x}10^{-5}$
DSP 2D	$4{,}00\text{x}10^{-6}$
DSP 5D	$1{,}36\text{x}10^{-4}$

Fuente: Autores

La figura 14 muestra el incremento del tiempo de ejecución del algoritmo PSO+SX de acuerdo al número de dimensiones para la función esfera. Para todos los experimentos, el error permitido fue de $1\text{x}10^{-6}$, el rango de partículas de 15 ~ 40 y el número máximo de iteraciones del PSO fue de 50 mientras que del simplex de 60. A medida que se incrementó el número de dimensiones, el tiempo que empleó el DSP aumentó de forma exponencial, tal y como lo demuestra la interpolación mostrada en la figura por la línea punteada.

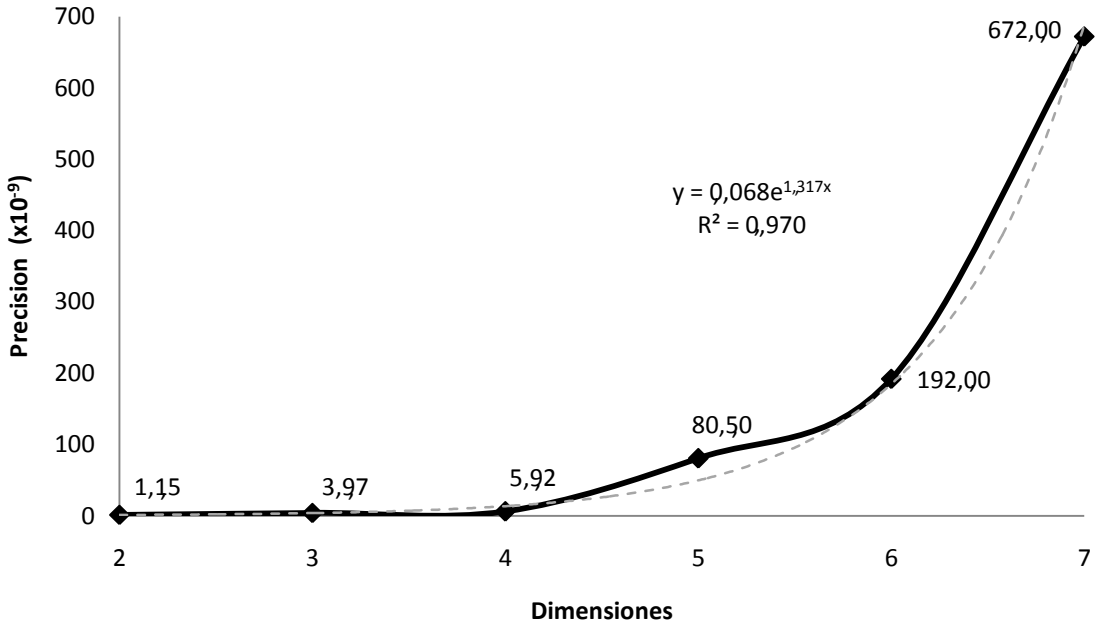
Figura 14. Tiempo de ejecución para la función esfera de dos a siete dimensiones en el DSP.



Fuente: Autores

En la figura 15, se aprecia la pérdida de precisión del método PSO+SX conforme el número de dimensiones del problema aumenta, sin embargo, los datos arrojados siguen siendo menores que el error permitido ($1x10^{-6}$). Nuevamente, el comportamiento de los resultados describe una tendencia exponencial (línea punteada).

Figura 15. Precisión para la función esfera de dos a siete dimensiones en el DSP.



Fuente: Autores

Los análisis descritos en las figuras 14 y 15, se exponen como una ayuda para estimar si la implementación en el DSP del algoritmo PSO+SX, es viable en una aplicación específica.

3.2 EVALUACIÓN DEL DESEMPEÑO

La tabla 2 y la figura 16, representan la evaluación del desempeño de los algoritmos, junto con una relación de tiempos para visualizar los resultados obtenidos durante la experimentación, obteniéndose las siguientes conclusiones:

- Es evidente que las respuestas en el computador con respecto al tiempo son mucho mejores comparadas con el DSP.

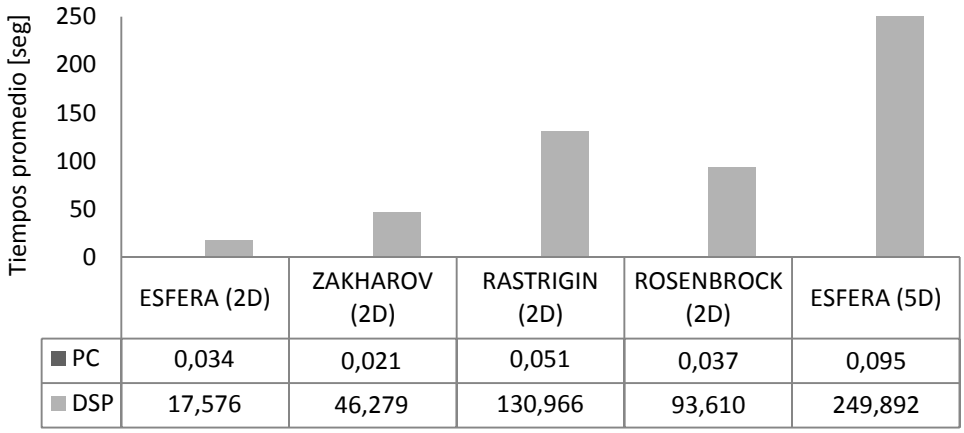
- Excluyendo los datos obtenidos para la esfera en dos dimensiones, el DSP es un dispositivo más de 2000 veces lento que el computador para realizar este tipo de algoritmo.

Tabla 2. Evaluación del desempeño

Función	Hardware	No. de pruebas realizadas	Tiempo promedio [seg]	Relación de tiempos
Esfera (2D)	DSP	32	17,576	523,21
	PC	32	0,0335	
Zakharov (2D)	DSP	32	46,279	2247,25
	PC	32	0,0206	
Rastrigin (2D)	DSP	32	130,966	2550,77
	PC	32	0,0513	
Rosenbrock (2D)	DSP	32	93,610	2515,14
	PC	32	0,0372	
Esfera (5D)	DSP	20	249,892	2642,97
	PC	20	0,0945	
Rastrigin (5D)	DSP	4	1684,795	3475,95
	PC	10	0,4847	

Fuente: Autores

Figura 16. Resultados de tiempos promedio



Fuente: Autores

4. CONCLUSIONES

Se logró implementar el algoritmo del híbrido Particle Swarm Optimization + Simplex (PSO+SX) en un DSP, obteniendo resultados con una precisión apreciable. Sin embargo, los tiempos de ejecución del algoritmo en el DSP seleccionado resultaron considerablemente elevados, debido a la diferencia entre el formato de manejo de datos entre ambos dispositivos, en velocidad de lectura y escritura de datos entre los procesadores y a que el DSP no tiene memoria caché dentro del procesador.

Con las pruebas realizadas del PSO+SX, se verificó con base en comparaciones con un computador, que en el DSP seleccionado los datos obtenidos cumplieron con la precisión establecida para la mayoría de las funciones de prueba.

Ya que el DSP seleccionado obtiene respuestas precisas y dentro del error permitido, se puede aprovechar el tamaño del mismo como un dispositivo portable, para situaciones que requieran, por ejemplo, de una apreciable capacidad de cómputo de gran precisión.

REFERENCIAS

[1] Yong Zhao, Xueying An y Wencai Luo, “Hybrid Particle Swarm Optimization based on parallel Collaboration”, IEEE, International Conference on Intelligent Computation Technology and Automation, 2008, pp 65 – 70.

[2] Bo Li y Ren Yue Xiao, “The Particle Swarm Optimization Algorithm: How to select the number of iteration”, Kaohsiung, IEEE, 26-28 Nov, 2007, pp 191 - 196.

[3] J. Kennedy y R. C. Eberhart, “Particle Swarm Optimization”, Perth, WA, Australia, IEEE Nov/Dic 1995, vol.4, pp 1942 – 1948.

[4] Jianping Wen, Xiaolan Wu, Kuo Jiang y Binggang Cao, “Particle Swarm Algorithm based on normal cloud”, Hong Kong IEEE, 1-6 Jun, 2008, pp 1492 - 1496.

[5] Zhigang lian, Fan Zhu, Zailin Guan y Xinyu Shao, “The Analysis of Particle Swarm Optimization Algorithm’s Convergence”, China, IEEE, 25-2 Jun, 2008, pp 623 – 626.

[6] Maurice Clerc, “Particle Swarm Optimization”, Ed. ISTE USA, Estados Unidos, 2006.

[7] Xuyuan Li, Hualong Xu y Zhaogang Cheng, “One immune Simplex Particle Swarm Optimization and its Application”, IEEE, Fourth International Conference on Natural Computation, China, 18-20 Oct, 2008, pp 331 – 335.

[8] Test Functions for Unconstrained Global Optimization, 2009, En internet: “http://www-optima.amp.i.kyoto-

u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm" (Fecha de revisión: Jun 7/10).

[**9]** J. A. Nelder y R. Mead, "A simplex method for function minimization", Computer Journal, 1965, vol. 7, pp 308-313.

[10] Parvis Palangpour, Pinaki Mitra, Swakshar Ray y Ganesh K. Venayagamoorthy, "DSP-Based PSO Implementation for Online Optimization of Power System Stabilizers", IEEE, NASA/ESA Conference on Adaptive Hardware and Systems, Noordwijk, 22-25 Jun, 2008, pp 379 – 384.

[11] Shu-Kai S. Fan, Yun-Chia Liang y Erwie Zahara, "A genetic algorithm and a Particle Swarm optimizer hibrized with Nelder-Mead simplex search", Computers & Industrial Engineering, Taiwan, Ago, 2005, Vol. 50, pp 401 - 425.

[12] Ruidiaz, Yair de Jesús y Tijaro, Omar, "Analizador de Espectros portátil utilizando la familia DSP5680X de Motorola", trabajo de grado, Universidad Industrial de Santander, Colombia, 2005

[13] Weiskamp, Keith, "Advanced Turbo C Programming", Academic Press, INC., 1988.

[14] Steven Torres, "Using Processor Expert to Develop a Software Real-Time Clock" Freescale Semiconductor, Application Note, Rev. 1, Enero, 2006.

[15] 56F8357 Data Sheet, Freescale Semiconductor, Rev 15, Enero, 2007.

[16] Ceballos, Francisco, "Enciclopedia del Lenguaje C++", Ed. Alfaomega, México, 2004.

[17] Maurice Clerc y James Kennedy, “The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space”, IEEE, Transactions on Evolutionary Computation, 2002, Vol. 1, pp 58 - 73.

[18] Riccardo Poli y David Broomhead, “Exact Analysys of the Sampling Distribution for the canonical Particle Swarm Optimiser and its Convergence during Stagnation”, Genetic and Evolutionary Computation Conference, Ant Colony optimization and swarm intelligence, England, 2007, pp 134 – 141.

[19] John H. Mathews y Kurtis K. Fink, “Numerical Methods using Matlab”, Cuarta edición, Capítulo 8, USA, 2004.

[20] Osma, Jaime y Villarreal, Mónica, “Comparación del desempeño del algoritmo de Optimización PSOSX (PE) frente al PSOSX (S)”, trabajo de grado, Universidad Industrial de Santander, Bucaramanga, 2009.