PRUEBAS DE DESEMPEÑO DEL ALGORITMO PSO (*PARTICLE SWARM OPTIMIZATION*) UTILIZANDO UN DSP (*DIGITAL SIGNAL PROCESSOR*)

AUTOR:
DAVID MATAJIRA RUEDA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2015

PRUEBAS DE DESEMPEÑO DEL ALGORITMO PSO (*PARTICLE SWARM OPTIMIZATION*) UTILIZANDO UN DSP (*DIGITAL SIGNAL PROCESSOR*)

AUTOR:
DAVID MATAJIRA RUEDA

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRÓNICO

CARLOS RODRIGO CORREA CELY, PhD.
DIRECTOR TRABAJO DE GRADO

OSCAR JAVIER BEGAMBRE CARRILLO, PhD.
CODIRECTOR TRABAJO DE GRADO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2015

"Si buscas resultados distintos, no hagas siempre lo mismo".
Albert Einstein.

Agradecimientos

"***El azar no existe, Dios no juega a los dados***"

A mi familia… "*Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber*".

A Carlos Rodrigo Correa Cely, mi director de proyecto de grado… "*Yo no enseño a mis alumnos, solo les proporciono las condiciones en las que puedan aprender*".

A Oscar Javier Begambre Carrillo, mi codirector de proyecto de grado… "*La formulación de un problema, es más importante que su solución*".

A mis compañeros del Grupo de Investigación CEMOS (Control – Electrónica – Modelado - Simulación)… "*Tengo una pregunta que a veces me tortura: Estoy loco yo o los locos son los demás*".

A Daniel Alfonso Sierra Bueno, mi calificador… "*No entiendes realmente algo a menos que seas capaz de explicárselo a tu abuela*".

A Cesar Antonio Duarte Gualdron, mi calificador… "*En los momentos de crisis, sólo la imaginación es más importante que el conocimiento*".

Todas las frases son autoría de Albert Einstein. Gracias a su ingenio aprendí:

"*Si buscas resultados distintos, no hagas siempre lo mismo*"

TABLA DE CONTENIDO

INTRODUCCIÓN 18
1 Descripción del trabajo de grado 21
1.1 Objetivo general 21
1.2 Objetivos específicos 21
2 FUNDAMENTACIÓN TEÓRICA 23
2.1 Optimización por enjambre de partícula 24
2.2 Procesador de señales digitales 32
3 ESTADO DEL ARTE 35
3.1 Optimización por enjambre de partícula (PSO): 35
3.2 Procesador de señales digitales (DSP): 39
4 ANÁLISIS Y RESULTADOS 43
4.1 Seleccionar: 43
4.2 Efectuar: 46
4.2.1 Función Ackley: 49
4.2.2 Función Griewank: 50
4.2.3 Función Rastrigin: 51
4.2.4 Función Rosenbrock: 52
4.2.5 Función Schaffer F6: 53
4.3 Desarrollar: 57
4.4 Visualizar: 59
4.5 Verificar: 62
5 OBSERVACIONES Y CONCLUSIONES 87
6 RECOMENDACIONES 89
7 REFERENCIAS BIBLIOGRAFÍCAS 90

BIBLIOGRAFÍA.....93
ANEXOS.....96

LISTA DE FIGURAS

FIGURA 1. ALGORITMO BÁSICO DEL PSO.....28
FIGURA 2. DIAGRAMA DE FLUJO VERSIÓN ASÍNCRONA DE PSO.....30
FIGURA 3. DIAGRAMA DE FLUJO VERSIÓN SÍNCRONA DE PSO.....31
FIGURA 4. ARQUITECTURA VON NEUMANN 33
FIGURA 5. ARQUITECTURA HARVARD 34
FIGURA 6. DISTRIBUCIÓN DE PUBLICACIONES REALIZADAS SOBRE TÉCNICAS DE OPTIMIZACIÓN. TOMADA DE [13].....36
FIGURA 7. NUMERO DE PUBLICACIONES SOBRE PSO POR AÑO, ENTRE LOS AÑOS 2001 – 2011. TOMADA DE [13].....38
FIGURA 8. TOPOLOGÍA GLOBAL PARA EL PSO.....47
FIGURA 9. FUNCIÓN ACKLEY. 49
FIGURA 10. FUNCIÓN GRIEWANK.....50
FIGURA 11. FUNCIÓN RASTRIGIN. 51
FIGURA 12. FUNCIÓN ROSENBROCK. 52
FIGURA 13. FUNCIÓN SCHAFFER F6. 53
FIGURA 14. TARJETA DE DESARROLLO TMDS320C6711. TOMADA DE [6]. 59
FIGURA 15. COMPARATIVO DE LA PRECISIÓN DE LOS SISTEMAS PSO-PC Y PSO-DSP CON RESPECTO A LA DIFERENCIA DE ERROR EN LOS VALORES DE LA FUNCIÓN OBJETIVO.....64
FIGURA 16. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN ACKLEY. 65
FIGURA 17. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN GRIEWANK.....66
FIGURA 18. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN RASTRIGIN.....67
FIGURA 19. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN ROSENBROCK. 68

FIGURA 20. PROMEDIO DE LA DESVIACIÓN ESTÁNDAR DE LOS VALORES MÍNIMOS ENCONTRADOS.....69
FIGURA 21. COMPARATIVO DE LA RELACIÓN DE TIEMPOS DE PROCESAMIENTO PARA LAS FUNCIONES DE PRUEBA (TIEMPO PC / TIEMPO DSP).....72
FIGURA 22. PROMEDIO DE LA DESVIACIÓN ESTÁNDAR DE LOS TIEMPOS DE PROCESAMIENTO.....73
FIGURA 23. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN ACKLEY.....74
FIGURA 24. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN GRIEWANK. 74
FIGURA 25. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN RASTRIGIN. 75
FIGURA 26. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN ROSENBROCK.....75
FIGURA 27. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN SCHAFFER F6.....76
FIGURA 28. DIFERENCIA DE ITERACIONES REALIZADAS POR LOS SISTEMAS PSO-PC Y PSO-DSP.....76
FIGURA 29. PORCENTAJE DE LA DIFERENCIA DE ITERACIONES REALIZADAS POR LOS PROCESOS EN CADA UNA DE LAS FUNCIONES CON RESPECTO AL TOTAL DE ITERACIONES REALIZADAS (1000).....78
FIGURA 30. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN ACKLEY.....79
FIGURA 31. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN GRIEWANK.....79
FIGURA 32. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN RASTRIGIN.....80
FIGURA 33. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN ROSENBROCK. 80

FIGURA 34. COMPARATIVO ENTRE LAS DESVIACIONES DE LOS MÍNIMOS ENCONTRADOS ESTÁNDARES ENTRE PC-DSP PARA LA FUNCIÓN ACKLEY. 81
FIGURA 35. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS MÍNIMOS ENCONTRADOS ENTRE PC-DSP PARA LA FUNCIÓN GRIEWANK. 81
FIGURA 36. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS MÍNIMOS ENCONTRADOS ENTRE PC-DSP PARA LA FUNCIÓN RASTRIGIN. 82
FIGURA 37. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS MÍNIMOS ENCONTRADOS ENTRE PC-DSP PARA LA FUNCIÓN ROSENBROCK. 82
FIGURA 38. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN ACKLEY. 83
FIGURA 39. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN GRIEWANK. 84
FIGURA 40. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN RASTRIGIN. 84
FIGURA 41. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN ROSENBROCK. 85
FIGURA 42. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN SCHAFFER F6. 85
FIGURA 43. ITERACIONES POR SEGUNDO DEL PROCESO PSO-PC PARA TODAS LAS FUNCIONES. 86
FIGURA 44. ITERACIONES POR SEGUNDO DEL PROCESO PSO-DSP PARA TODAS LAS FUNCIONES. 86
FIGURA 45. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (2 DIMENSIONES).....96
FIGURA 46. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (5 DIMENSIONES).....97
FIGURA 47. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (10 DIMENSIONES).....97

FIGURA 48. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (25 DIMENSIONES).....98
FIGURA 49. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (50 DIMENSIONES).....98
FIGURA 50. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (100 DIMENSIONES).....99
FIGURA 51. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (2 DIMENSIONES).....100
FIGURA 52. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (5 DIMENSIONES).....100
FIGURA 53. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (10 DIMENSIONES).....101
FIGURA 54. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (25 DIMENSIONES).....101
FIGURA 55. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (50 DIMENSIONES).....102
FIGURA 56. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (100 DIMENSIONES).....102
FIGURA 57. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (2 DIMENSIONES).....103
FIGURA 58. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (5 DIMENSIONES).....103
FIGURA 59. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (10 DIMENSIONES).....104
FIGURA 60. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (25 DIMENSIONES).....104
FIGURA 61. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (50 DIMENSIONES).....105

FIGURA 62. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (100 DIMENSIONES).....105
FIGURA 63. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (2 DIMENSIONES).....106
FIGURA 64. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (5 DIMENSIONES).....106
FIGURA 65. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (10 DIMENSIONES).....107
FIGURA 66. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (25 DIMENSIONES).....107
FIGURA 67. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (50 DIMENSIONES).....108
FIGURA 68. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (100 DIMENSIONES).....108
FIGURA 69. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN SCHAFFER F6. (2 DIMENSIONES).....109

LISTA DE TABLAS

TABLA 1. COMPARATIVO ENTRE LOS DSP DE REFERENCIA TMS320C6711 DE TEXAS INSTRUMENTS INCORPORATED Y MC56F8357 FREESCALE SEMICONDUCTOR INCORPORATED. 44
TABLA 2. CARACTERÍSTICAS PC. 48
TABLA 3. CARACTERÍSTICAS DE LAS FUNCIONES ESTÁNDARES SELECCIONADAS. 54
TABLA 4. RESULTADOS PRUEBAS CON EL PC. 56
TABLA 5. RESULTADOS PRUEBAS CON EL DSP. 61
TABLA 6. DIFERENCIA DE ERROR ENTRE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP. 63
TABLA 7. RELACIÓN DE TIEMPOS DE PROCESAMIENTO ENTRE EL PC Y EL DSP. 70
TABLA 8. PORCENTAJE DE DIFERENCIA DE ERROR ENTRE LAS ITERACIONES REQUERIDAS POR EL PC Y EL DSP CON RESPECTO A LAS MÁXIMAS DEFINIDAS (1000). 77

LISTA DE ANEXOS

ANEXO A: PROCESAMIENTO FUNCIÓN ACKLEY. 96
ANEXO B: PROCESAMIENTO FUNCIÓN GRIEWANK. 100
ANEXO C: PROCESAMIENTO FUNCIÓN RASTRIGIN. 103
ANEXO D: PROCESAMIENTO FUNCIÓN ROSENBROCK. 106
ANEXO A: PROCESAMIENTO FUNCIÓN SCHAFFER F6. 109

RESUMEN

TÍTULO: Pruebas de desempeño del algoritmo PSO (*Particle Swarm Optimization*) utilizando un DSP (*Digital Signal Processor*).[*]

AUTOR: DAVID MATAJIRA RUEDA.[†]

PALABRAS CLAVE: PSO, optimización por enjambre de partícula, DSP, procesador de señales digitales, función objetivo, óptimo local, optimo global.

CONTENIDO: Este documento tiene como objetivo presentar las pruebas de desempeño del algoritmo de Optimización por Enjambre de Partícula implementado en un Procesador de Señales Digitales utilizando funciones estándares para analizar las características del proceso (tiempo de procesamiento, números de iteraciones, precisión de los valores óptimos (Mínimos y función objetivo), iteraciones por segundo, robustez, eficacia y eficiencia, etc). Se mostrarán tablas, gráficas y demás figuras, que permitirán evidenciar el rendimiento del PSO en un dispositivo programable que entre otras importantes características, permite dedicar el procesamiento a una tarea específica sin perder recursos en otras innecesarias. Para verificar la validez de este proceso se mostrarán los resultados obtenidos de maximizar y/o minimizar funciones estándar a través de esta estrategia, una vez analizados, serán comparados con los resultados de implementar el mismo algoritmo de PSO en un computador personal por medio de programación en C/C++, para finalmente analizar los datos de los dos procesos, que permitirán observar que la implementación del PSO por medio de un DSP es una opción factible y viable. Además, este documento proporcionará la caracterización de algunas de las variables que se encuentran relacionadas con el proceso y que de una u otra manera proveerán de ayuda a próximos proyectos.

---

[*] Proyecto de grado.

[†] Facultad de Ingenierías Físicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Carlos Rodrigo Correa. Co-director: Oscar Javier Begambre.

ABSTRACT

TITLE: Performance testing of the PSO (Particle Swarm Optimization) algorithm using a DSP (Digital Signal Processor).[‡]

AUTHOR: DAVID MATAJIRA RUEDA.[§]

KEYWORDS: PSO, Particle Swarm Optimization, DSP, digital signal processor, objective function, local optimum global optimum.

CONTENT: This study presents the performance testing of the algorithm Particle Swarm Optimization implemented on a Digital Signal Processor using standard functions (Benchmark Functions) to analyze the process characteristics (processing times, number of iterations, optimal values accuracy (minimums and objective function), iterations per seconds, sturdiness, errors, effectiveness and efficiency, etc). Tables, graphs and other figures show the performance of PSO on a programmable device that among other important features, allows devoting the processing to a specific task without losing resources in unnecessary taskes. In order to check the validity of this process, the results of maximizing and/or minimizing standard functions through this strategy are displayed. Once analyzed, they are compared with the results of implementing the same algorithm of PSO on a personal computer through C / C ++, programming. Finally, the analysis of data of the two processes, allows noting that the implementation of the PSO using a DSP is a feasible and viable option. In addition, this study provides the characterization of some of the variables which are related to the process and one way or another provide assistance to upcoming projects. This is a simple project to demonstrate and/or confirm the performance of the DSP with respect to a specific PC.

[‡] Undergraduate thesis.

[§] Facultad de Ingenierías Físicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Advisor: Carlos Rodrigo Correa. Co-advisor: Oscar Javier Begambre.

INTRODUCCIÓN

En la actualidad, los diferentes sistemas industriales son diseñados de forma tal que puedan optimizar, por ejemplo, la totalidad de la materia prima como táctica de preservación ambiental, la potencia eléctrica consumida por un dispositivo permitiendo un uso eficiente de energía, el tiempo de ejecución de procesos electrónicos o los recursos computacionales de un sistema. Los procesos de control de calidad de una u otra forma concluyen en la idea de la optimización de sistemas.

Los algoritmos de optimización, se presentan como una clara alternativa para la consecución de dicho objetivo; y ya que son implementados a través de la informática, proveen con efectividad soluciones satisfactorias, sin un esfuerzo considerable.

En los últimos años se han encontrado gran cantidad de estrategias de optimización de sistemas, basadas en programación, las cuales pretenden suministrar mayor eficiencia y efectividad frente a diversos problemas, y en variadas áreas de la ingeniería. Algunas de dichas estrategias se originan de modificaciones realizadas al algoritmo de optimización por enjambre de partículas (PSO), que aunque no es el único entre tales algoritmos posee gran importancia por una característica totalmente esencial y relevante como lo es su simplicidad.

Entre dichos algoritmos se encuentran principalmente los algoritmos genéticos, más conocidos como GA (de sus siglas en inglés para *Genetic Algorithm*) y los algoritmos evolutivos, más conocidos como EA (de sus siglas en inglés para *Evolutionary Algorithm*); y es en estos últimos donde la comunidad científica centra su atención debido a su simplicidad de forma y procesamiento.

Aunque existen diversas versiones del algoritmo PSO, y todas pretenden mejorar una determinada característica, terminan realizando un compromiso entre efectividad y rendimiento, y aunque a veces logran un buen balance entre estos, no es lo común, ya que según la aplicación y la versión que se utilice el comportamiento del algoritmo es en algunos casos impredecible, mostrando en ciertos procesos la exhibición de una gran efectividad.

La relación y/o compromiso existente entre rendimiento y efectividad parece ser un inconveniente en el mejoramiento y desarrollo de este algoritmo. Los algoritmos de optimización basados en métodos heurísticos han dominado considerablemente en la solución de diversos problemas multidimensionales, lineales o no-lineales, continuos o discretos, por su efectividad y fácil implementación en distintas áreas de la investigación y su popularidad aumenta cada vez que son aplicados a otros campos de la ciencia.

Actualmente, solo se tiene un referente de la ejecución de este algoritmo en otro dispositivo diferente a un computador personal multipropósito, y allí su comportamiento no fue caracterizado, por tanto, implementarlo en un procesador dedicado y caracterizar su comportamiento se presenta como una alternativa que en principio podría permitir la obtención de tiempos de procesamiento menores aprovechando así las cualidades propias del dispositivo como lo es por ejemplo, el procesamiento en tiempo real, para lo cual existen variadas aplicaciones del algoritmo en cuestión.

Los algoritmos de optimización y mucho más el de Optimización por Enjambre de Partículas (PSO) son constantemente modificados buscando el mejoramiento de alguna de sus características, dichas modificaciones se llevan a cabo sobre el seudocódigo original o sobre sus versiones (híbridos), pero se ha encontrado poca evidencia de trabajos realizados sobre estos bio-algoritmos adaptados a

dispositivos electrónicos de alto rendimiento diferentes a los equipos de computación normalmente usados, por tanto este proyecto surge como una prueba práctica al mejoramiento del algoritmo de optimización enfocándose no en su código sino en su ejecución y más específicamente en el lugar de ejecución.

Las modificaciones que son hechas a estos algoritmos muchas veces producen un aumento en la precisión o un aumento en el tiempo de procesamiento. Es allí donde se hace evidente la necesidad de implementarlo en un dispositivo de alto rendimiento y así comprobar que se pueden obtener tiempos de procesamiento menores con un alto grado de precisión y sobre todo de confiabilidad en los resultados que suministre.

El dispositivo electrónico que se plantea utilizar, posee características especiales entre muchos otros, que posiblemente permitirán que se puedan alcanzar las condiciones en las que el algoritmo sea en su totalidad eficiente y eficaz.

El desarrollo del presente proyecto se basa en cinco pasos o procedimientos relacionados y congruentes con los objetivos del mismo. Inicialmente, se seleccionara un dispositivo capaz de realizar el procesamiento necesario para el algoritmo del PSO, luego se escogerán cinco funciones estándar que se aplicaran al PSO siendo ejecutado en un computador personal (PC). A continuación se desarrollara la programación del PSO esta vez en el DSP seleccionado en pasos anteriores; nuevamente se aplicaran las funciones para adquirir datos que, finalmente, posibilitarán las respectivas comparaciones entre ambos procesos, determinando las diferencias y similitudes entre las dos plataformas planteadas.

Así, se mencionarán las ventajas y desventajas de la implementación del PSO tanto en el PC, como en el DSP.

1 DESCRIPCIÓN DEL TRABAJO DE GRADO

Con este proyecto de investigación se muestra la implementación del algoritmo PSO en un DSP, realizando pruebas de desempeño a través de funciones estándares que exhiben las características que dicho algoritmo posee en un procesador de propósito especial, como lo es el DSP. Para lo anterior se escogieron cinco funciones estándares para revisar el procesamiento que se obtienen en cada una de ellas, mediante el sistema propuesto y realizando modificaciones en los parámetros del algoritmo, entre otras. Lo mencionado, con el objetivo de demostrar que el sistema conformado por el algoritmo (PSO) y el dispositivo (DSP) permite una notable mejoría en el desempeño del mismo.

1.1 OBJETIVO GENERAL

Caracterizar la estrategia de implementación del algoritmo PSO (*Particle Swarm Optimization*) en un DSP (*Digital Signal Processor*).

1.2 OBJETIVOS ESPECÍFICOS

El cumplimiento del objetivo general del trabajo de grado comprende:

1. Seleccionar un dispositivo DSP adecuado que permita realizar el procesamiento del algoritmo con alta confiabilidad y rapidez.

2. Efectuar pruebas de desempeño del proceso a través de funciones estándares que se emplean para este propósito y, a partir de los resultados obtenidos, analizar su eficiencia.

3. Desarrollar la programación del algoritmo de optimización aprovechando las características del DSP.

4. Visualizar y tabular el proceso de optimización que realiza PSO desarrollado en el DSP, analizando sus características específicas.

5. Verificar el desempeño del método de optimización PSO ejecutado por medio de un DSP, obteniendo gráficas y tablas de su comportamiento, entre otras.

2 FUNDAMENTACIÓN TEÓRICA

De una forma estricta, la palabra optimizar significa buscar la mejor manera de realizar una actividad; dentro de las técnicas de optimización con algoritmos computacionales esta se define como la acción de buscar y encontrar la mejor solución o si no, la más aproximada o suficientemente satisfactoria para resolver un problema[1].

Dentro de la cotidianidad se encuentran problemas básicos que necesitan ser resueltos en forma óptima y que por su complejidad muchas veces no se requieren análisis extensos para poder solucionarlos, en otros casos cabe la posibilidad de utilizar ayuda de la informática para este fin, debido en gran parte por la cantidad de variables que se deben manipular; es ahí donde se fundamenta la importancia de los algoritmos computacionales[2].

En la clasificación de los bio-algoritmos de optimización (algoritmos basados en el comportamiento natural de algunos animales) se encuentra la optimización por enjambre de partículas, la cual es mucho más conocida como PSO (de sus siglas en inglés para *Particle Swarm Optimization*)[3].

La importancia de este tipo de algoritmo radica en la capacidad de exploración de espacios (lineales o no-lineales) que la mayoría de veces se tornan multidimensionales.

Durante la última década se han presentado modificaciones sobre la programación general de este algoritmo, las cuales han tratado de cierta forma de “optimizar” el algoritmo de optimización, proponiendo estrategias de búsqueda, realizando cambios (en algunos casos muy simples) a los parámetros predeterminados del

algoritmo original. Lo anterior con el fin de lograr maximizar algunas características y en otros casos de minimizarlas, como sucede con el tiempo de procesamiento[2].

Es así como surgió la idea de implementar este algoritmo por medio de un dispositivo que proporciona tiempos de procesamiento relativamente bajos y cuyo uso es cada vez más común en la ingeniería.

2.1 OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULA

El algoritmo de optimización por enjambre de partícula fue desarrollado originalmente por el sociólogo James Kennedy y por el ingeniero electrónico Russell Eberhart en el año de 1995; basándose en la idea de “sociedad” de ciertas especies naturales, las cuales desarrollan una actividad común, realimentando información de forma individual y colectiva [4][5].

Para poder llegar a un conocimiento simple sobre PSO se puede pensar en las posibles estrategias para encontrar los máximos y mínimos de una función sin recurrir al método diferencial[1]. Partiendo de lo anterior y haciendo énfasis en la analogía biológica se podría explicar el procesamiento de PSO como se describe a continuación: Un enjambre de abejas requiere encontrar alimento, el grupo en general no sabe en qué lugar se encuentra pero conoce de antemano la distancia a dicho lugar y analizan la mejor forma de llegar a esa región, por lo cual la mejor forma de hacerlo es seguir a la abeja que se encuentre más cerca de allí. De esta forma, cada abeja se comunica con sus compañeras y sabiendo cuál de ellas está más cerca, decide si debe o no seguir una ruta que la acerque más al alimento, las decisiones que toma influyen directamente en su dirección, sentido y velocidad, y por ende en su nueva posición[4].

Haciendo referencia directa sobre el problema matemático como tal, se puede decir, que una estrategia para lograr resolver el problema inicial, y que resalta por su simplicidad, es evaluar con cierta cantidad de iteraciones la función a maximizar (o minimizar) observando así sus valores críticos, comparándolos entre sí. Lo anterior, representa un método que se hace monótono y al extremo desgastante, pero la solución puede darse mucho más rápido, y esto es lo que facilita y agiliza este proceso cuando se implementa mediante programación computacional.

La función que se requiere optimizar se le llama normalmente función objetivo, las partículas (en el ejemplo, las abejas) se les llama normalmente agentes, y cada una de ellas representa una posible solución, estas son liberadas al azar (otras veces de forma definida, y en otras ocasiones con una combinación de estas dos) en el espacio de búsqueda y poseen características especiales tales como posición y velocidad, las cuales depende de sus valores inmediatamente anteriores y son las responsables del “movimiento” sobre el espacio [4].

Se hace presente también una función *fitness* (llamada así en esta y en muchas otras aplicaciones, y en cada una de ellas mantiene el mismo significado) la cual es la que posee la información sobre las mejores posiciones individual y colectiva, y con base a ella se puede evaluar y decidir cuál será la nueva posición que debe ocupar el agente dándole una velocidad. Es decir, permite evaluar y actualizar la velocidad e indirectamente la posición del agente[4][1].

Según sea la forma de actualizar estos datos, se encuentran en la literatura las versiones síncrona y asíncrona del algoritmo. Además, dependiendo de la influencia en la memoria de cada agente se pueden diferenciar dos tipos de algoritmos: Global y local[6].

Optimización local: En la optimización local se busca una mejor solución que se encuentra entre una de todas las posibles soluciones.

Optimización global: En la optimización global se busca una mejor solución que se encuentra entre todas las posibles soluciones.

Realizando las respectivas combinaciones entre estos cuatro factores se pondrán en evidencia las versiones más utilizadas en el campo de la ingeniería[7].

Debe notarse que en la mayoría de casos de aplicación se realiza previamente un estudio paramétrico con el fin de obtener el conjunto de valores (parámetros) que determinaran la configuración inicial (el arranque del algoritmo). Así pues, no sobra mencionar que es necesaria una sintonización particular a una aplicación específica[8][9].

Conceptualmente el algoritmo se presenta de manera muy simple y aun así existen una gran variedad de híbridos del mismo que se han desarrollado últimamente en varias comunidades científicas[2].

Dos ecuaciones pueden representar el comportamiento de las partículas y en ellas se hacen evidentes las propiedades que ya fueron mencionadas para estas:

$$v_{in}(k+1) = v_{in}(k) + c_{1.}r_1(k).(P_{in}(k) - x_{in}(k)) + c_2.r_2(k).(G_{in}(k) – x_{in}(k)) \quad (1)$$

$$x_{in}(k+1) = x_{in}(k) + v_{in}(k+1) \quad (2)$$

En donde $v_{in}(k)$ y $x_{in}(k)$ representan la velocidad y la posición de la k-esima iteración de la partícula i en la n-esima dimensión del espacio de búsqueda,

respectivamente. Los coeficientes c_1 y c_2 son las llamadas constantes de aceleración y representan la influencia del comportamiento individual y colectivo sobre su propio movimiento. Los términos $r_1(k)$ y $r_2(k)$ son números aleatorios cuyo valor se determina en el intervalo [0,1] y tienen como propósito simular la capacidad circunstancial del movimiento de la partícula. Los términos P y G representan los valores de mejor posición individual y mejor posición colectiva, correspondientemente, estos pueden y deben variar durante la ejecución del algoritmo[10].

En cada iteración se actualizan la velocidad y la posición de las partículas, esto se hace por medio de las dos ecuaciones anteriormente presentadas; al tener ya dispuestos los valores iniciales para la velocidad, la posición y los parámetros de las ecuaciones (1) y (2), se realiza la primera iteración y se procede a actualizar la velocidad, teniendo en cuenta su valor anterior y los valores globales y locales de los mejores óptimos hasta ese momento, luego se procede a actualizar la posición, a partir del valor de velocidad, calculado inmediatamente antes. Los términos P y G se utilizan para llevar a cabo una cierta medida sobre la calidad del movimiento realizado por las partículas, obviamente el proceso se ejecuta tras ciertas iteraciones hasta que los agentes encuentren una posición que no puede ser mejorada.

El procedimiento anterior, se podría visualizar mejor a partir del estudio básico del algoritmo, como se muestra a continuación[10], en la Figura 1, donde gBest se refiere al mejor optimo global y pBest al optimo local encontrados, se utiliza provisionalmente la letra k para denotar las iteraciones que debe realizar el proceso y p como el número de partículas a liberar para que procedan con su búsqueda en el espacio:

FIGURA 1. ALGORITMO BÁSICO DEL PSO.

```
k=0
Para p=1 hasta el número de partículas
        Inicializar x_i y v_i
Mientras condición de parada no se cumpla
        Hacer
                k=k+1
                        Para p=1 hasta el número de partículas
                                Evaluar x_i
                                        Si f(x_i) es mejor que f(pBest_i)
                                                Entonces
                                                        pBest_i=x_i
                                                        f(pBest_i)=f(x_i)
                        Para p=1 hasta el número de partículas
                                Evaluar x_i
                                        Si f(x_i) es mejor que f(gBest_i)
                                                Entonces
                                                        gBest_i=x_i
                                                        f(gBest_i)=f(x_i)
Para p=1 hasta el número de partículas
        Calcular v_i la velocidad de x_i de acuerdo a pBest_i y gBest_i
        Calcular la nueva posición x_i
Devolver la mejor solución
```

En este punto debe destacarse la importancia de los diferentes coeficientes que hacen parte de las ecuaciones descriptivas, como también de la limitación de la velocidad de las partículas, ya que con valores no adecuados el algoritmo puede divergir o en otros casos mostrar soluciones erróneas o simplemente locales sin haber explorado todo el espacio de búsqueda.

Ya que se ha mencionado otra característica del algoritmo, como lo es la velocidad máxima de las partículas, la cual debe ser suministrada por el usuario que implementa el algoritmo, debe decirse sobre esta que su valor es esencial ya que podrían presentarse una de las dos situaciones descritas a continuación:

- Si la velocidad es muy alta puede ser que las partículas ignoren la región de una posible solución global. Ejemplo: Si los agentes exploran a una alta velocidad puede ser que no vean regiones de soluciones potenciales[9].

- Si la velocidad es muy baja puede ser que las partículas exploren la zona de una posible solución local y queden sumergidas en ella sin explorar el resto del espacio. Ejemplo: Si los agentes exploran a una baja velocidad puede ser que no vean otra región que contenía una mayor y mejor solución[9].

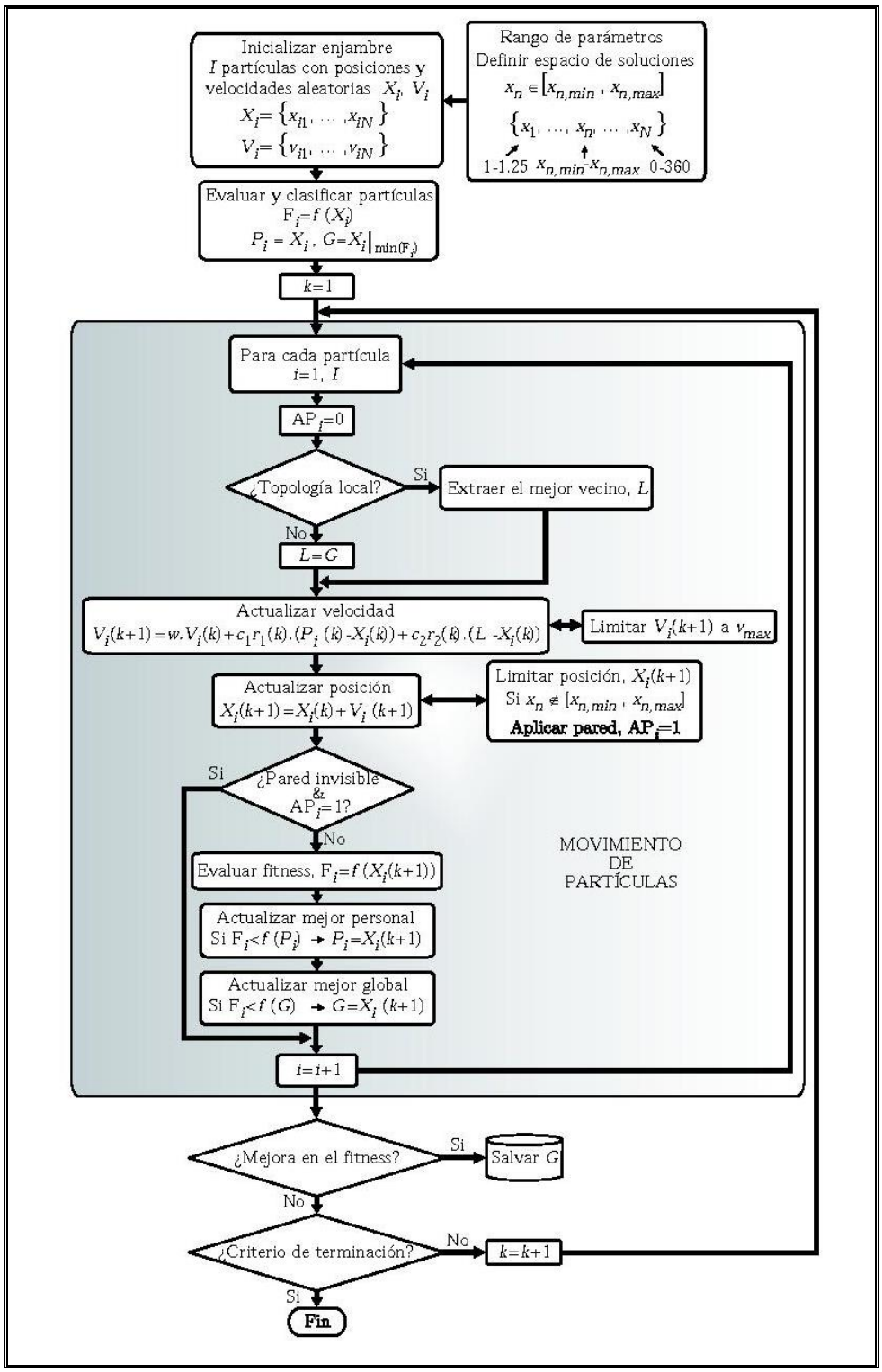
Además se podría pensar en prescindir de la velocidad fijándole un valor de cero; en tal caso ocurriría un fenómeno denominado explosión PSO, en el cual las partículas presentan un comportamiento oscilatorio y algunas veces inestable, haciendo que este método pierda por completo sus características de búsqueda y optimización[7].

A este punto ya se pueden presentar más detallamente los procedimientos que realizan tanto la versión de actualización asíncrona de PSO, como la síncrona del mismo.

En la versión asíncrona que se presenta en la Figura 2, la actualización de la mejor posición individual y de la mejor posición colectiva se hace de partícula a partícula, teniendo en cuenta la información suministrada en una iteración anterior.

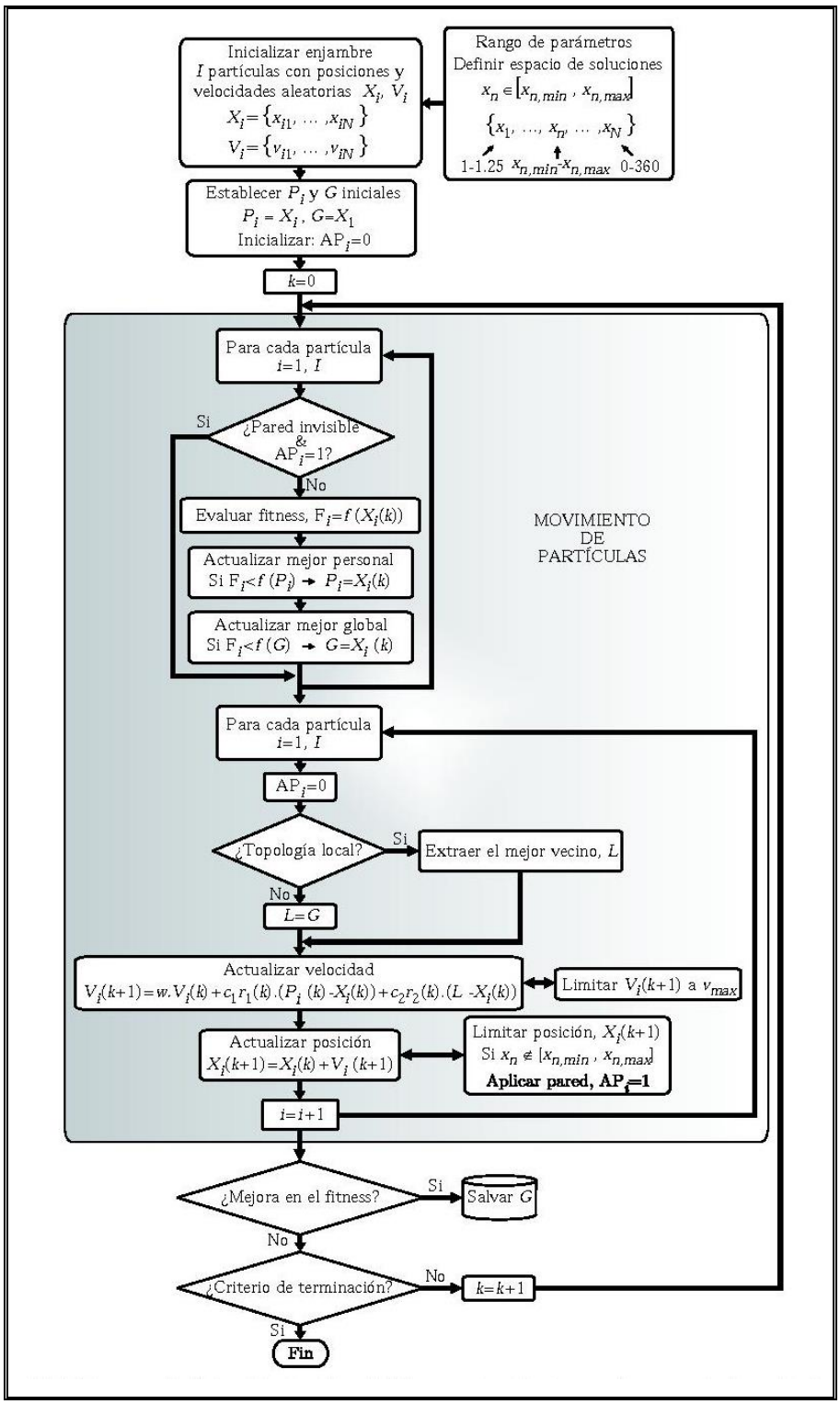
En contraste, en la versión síncrona que se presenta en la Figura 3, la actualización de la mejor posición individual y de la mejor posición colectiva de las partículas se hace al mismo tiempo, es decir, se evalúa el *fitness* de todas ellas, y se avanza a la nueva posición, esto es, que todas las partículas comparten exactamente la misma información.

FIGURA 2. DIAGRAMA DE FLUJO VERSIÓN ASÍNCRONA DE PSO.



TOMADA DE [7].

FIGURA 3. DIAGRAMA DE FLUJO VERSIÓN SÍNCRONA DE PSO.



TOMADA DE [7].

2.2 PROCESADOR DE SEÑALES DIGITALES

El procesamiento de señales digitales se refiere al procesamiento electrónico de señales haciendo uso de estrategias matemáticas para realizar transformaciones o análisis de información a través de un software característico y exclusivo, el cual puede hacer uso de lenguajes de alto o bajo nivel. Este procesamiento ha estado en constante evolución de la mano con la tecnología. Comenzó alrededor de la década de los años 60 cuando los computadores y dispositivos digitales se volvieron lo suficientemente rápidos para procesar grandes cantidades de datos con eficiencia. La principal ventaja del procesamiento digital es que puede ser modificado, corregido o actualizado a través de la reprogramación de un microprocesador o microcontrolador, contrario al procesamiento analógico[11].

Cuando el término “Digital” es usado, usualmente se refiere a un conjunto de valores finitos. Es el contraste de lo “Analógico” que se refiere a un rango continuo de valores. En el procesamiento de señales digitales, nosotros nos concentramos en procesar señales que son discretas en tiempo (muestreadas) y en muchos casos, discretas también en amplitud (cuantificadas), es decir, primordialmente analizamos secuencias de datos[3].

Un Procesador de Señales Digitales, DSP (de sus siglas en inglés para *Digital Signal Processor*) es un tipo de microprocesador o procesador de propósito especial, cuyas características lo hacen eficaz y eficiente; esto porque el DSP es capaz de procesar señales en tiempo real. La característica de procesamiento en tiempo real lo hace ideal para aplicaciones que no permiten retardos considerables. Por ejemplo, en los dispositivos de telefonía móvil es indispensable un DSP para poder administrar las conversaciones de los usuarios, y que estas ni se interrumpan ni se interfieran[12].

En la práctica, los DSP y las características que poseen para procesar señales, los hacen buenos candidatos para otros propósitos tales como graficación de alta calidad, modelado y simulación en ingeniería, entre otras; pero su fortaleza está en las aplicaciones que requieran procesamiento en tiempo real.

Entre los aspectos básicos por medio de los cuales podemos caracterizar un DSP, se encuentran por ejemplo: El formato aritmético, la rapidez de procesamiento, la disposición de memoria y la arquitectura o diseño interno. Actualmente en su mayoría podemos encontrar DSP de las empresas *Texas Intruments Incorporated*, *Freescale Semiconductor Incorporated* y *Analog Devices Incorporated*.

Los DSP hacen uso de arquitecturas especiales que permiten acelerar operaciones matemáticas intensas, como las que se encuentran en muchos sistemas en que se procesen señales en tiempo real. Un concepto relacionado estrechamente con los DSP es la operación MAC (multiplicar y acumular), para lo cual estos dispositivos cuentan con componentes electrónicos que posibilitan ejecutar dicha operación de manera rápida. También poseen la capacidad de dar acceso múltiple a la memoria y de cargar varios valores simultáneamente para ser utilizados en alguna operación.

Entre las arquitecturas más comúnmente utilizadas se encuentran La Von Neumann y la Harvard, cuyos esquemas se muestran en las Figuras 4 y 5:

FIGURA 4. ARQUITECTURA VON NEUMANN

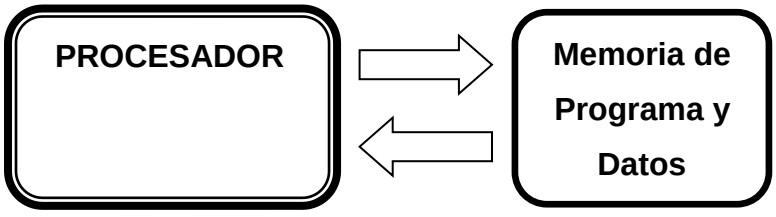


FIGURA 5. ARQUITECTURA HARVARD

3 ESTADO DEL ARTE

3.1 OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULA (PSO):

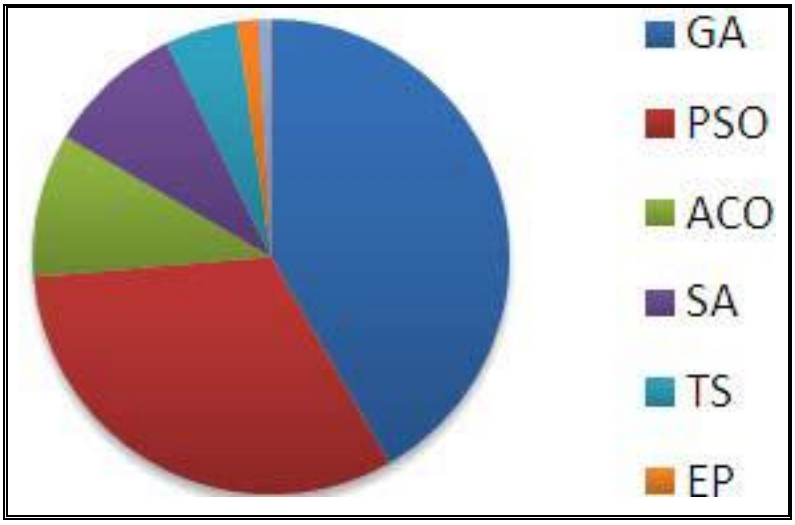
En la última década la utilización de los diferentes algoritmos de optimización se ha ido expandiendo a variados campos de la ciencia, sobre todo como herramientas para tomas de decisiones. Actualmente, se buscan algoritmos eficientes que sean capaces de solucionar problemas complejos de modelos matemáticos, así mismo, la tendencia de dicha búsqueda es por algoritmos con gran sencillez y que en lo posible no involucren operaciones diferenciales, ya que estas terminan presentando inconvenientes como incrementos en la complejidad computacional, o trampas en mínimos locales en las que los agentes de búsqueda caen sin retorno, y tal vez uno de los más relevantes, no pueden ser aplicados en ciertas funciones objetivo[13][1].

Eslami et al. hicieron una revisión exhaustiva sobre las técnicas de optimización más utilizadas y su evolución tanto en modificaciones sobre sus propios códigos, como también en los conocidos híbridos[13]. Según ellos, entre los algoritmos de optimización global más populares se encuentran:

- Algoritmos genéticos (GA) (Holland, 1992).
- Optimización por enjambre de partícula (PSO) (Kennedy y Eberhart, 1995).
- Evolución diferencial (DE) (Price y Storn, 1995).
- Programación evolutiva (EP) (Maxfield y Fogel, 1965).
- Optimización por colonia de hormigas (ACO) (Dorigo et al., 1991).

La gran presencia de artículos científicos sobre la técnica de optimización de PSO y según esta misma referencia bibliográfica, puede decirse que en estos días sigue siendo una de las más utilizadas junto con sus modificaciones e híbridos, ya que cualidades como su simplicidad y capacidad de resolución de problemas multidimensionales hacen que se resalte entre las demás técnicas.

FIGURA 6. DISTRIBUCIÓN DE PUBLICACIONES REALIZADAS SOBRE TÉCNICAS DE OPTIMIZACIÓN. TOMADA DE [13].



Entre las modificaciones más importantes hechas al algoritmo original de PSO, están las de la inclusión de un factor de inercia por parte de Shi y Eberhart (1999)[14], que lograba una mejora en cuanto a la exploración del campo de búsqueda. Otra de las modificaciones fue hecha por Clerc y Kennedy (2002)[9], los cuales agregaron un factor de constricción que permite mayor convergencia de los agentes, logrando así una reducción del tiempo de procesamiento y de la cantidad de iteraciones necesarias para hallar un óptimo preciso.

Según Eslami et al. se encuentran más de veinte modificaciones del algoritmo de PSO, las cuales se centran en enfatizar en alguna de las características propias de PSO. Entre las modificaciones más recientes se destacan:

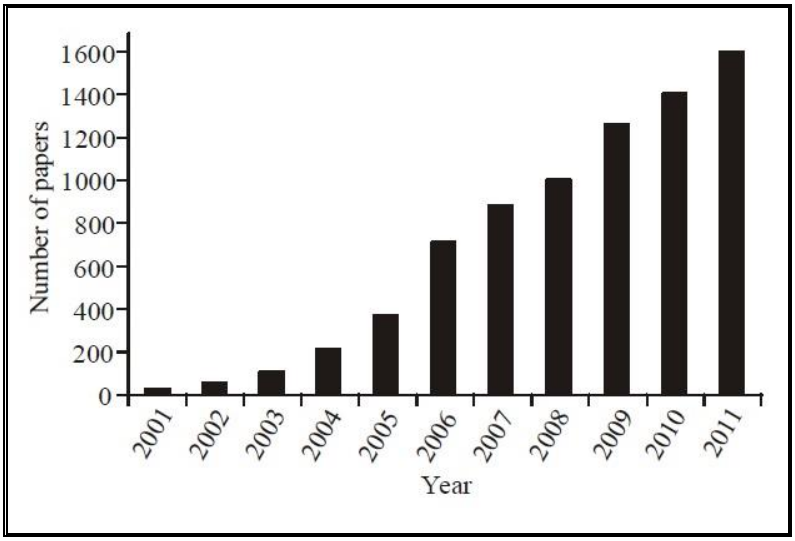
- MSSE-PSO: (Jiang et al., 2010) Desarrollaron un algoritmo de enjambre evolutivo tipo maestro-esclavo aleatorio, con el cual propusieron un sistema que muestrea la población de forma aleatoria y crea sub-enjambres (Maestro y esclavos) que ejecutan PSO independientemente y que tienen comunicación entre sí.
- HPSO: (Engelbrecht, 2010) En este caso las partículas pueden seguir mostrar diferentes comportamientos de búsqueda, obtenidos de un depósito de comportamientos predeterminados (En el documento de origen llaman a estos depósitos, piscinas.)
- C-Catfish PSO: (Chuang et al., 2011) La modificación pretende reemplazar las partículas con los peores óptimos por otras (Catfish) que se posicionan en los puntos extremos del espacio de búsqueda donde no se ha mejorado el óptimo global por cierto número de iteraciones. La utilización de los mapas de caos, aumentaron la capacidad de convergencia del Catfish-PSO.

Entre los híbridos más recientes se destacan:

- HGAPSOA: (Kuo y Lin, 2010) Se presenta un hibrido entre un GA y PSO con agrupación de datos, que permite reducir el tiempo de preparación o de inicio en la tecnología SMT.
- PSO-DE: (Liu et al., 2010) Este utiliza DE (Evolución Diferencial) para actualizar la mejor posición previa de las partículas de PSO y forzarlas a escapar si se presenta algún “estancamiento” en un mínimo local.

- PSO-DE: (Khamsawang et al. 2010) Estos autores propusieron otra modificación más para PSO-DE incluyendo un generador de restricciones. De igual forma, hicieron que el operador de mutaciones mejorara la capacidad de exploración y activara, si fuera necesario, excepciones de las condiciones limites o de la utilización de velocidades cercanas a cero.

FIGURA 7. NUMERO DE PUBLICACIONES SOBRE PSO POR AÑO, ENTRE LOS AÑOS 2001 – 2011. TOMADA DE [13].



En conclusión, se puede afirmar que las modificaciones y los híbridos de PSO seguirán buscando un espacio entre la multitud de investigadores, ya que como se puede ver en las gráficas utilizadas de Eslami et al. la aparición de publicaciones sobre este poderoso optimizador posee un tendencia creciente. Desde su simple planteamiento ha logrado demostrar que es una de las herramientas computacionales, basada en un paradigma natural, más útiles en nuestros días y en muchos campos del conocimiento.

3.2 PROCESADOR DE SEÑALES DIGITALES (DSP):

En la última década el aprovechamiento de los recursos para el procesamiento digital de señales se ha ido consolidando, desplazando casi completamente, al procesamiento analógico. Uno de los dispositivos más ampliamente utilizado en dicho propósito son los microprocesadores de propósito especial como lo es el DSP[15].

Obviamente, lo que puede hacer un DSP tiene su equivalente en otro (u otros) dispositivos, pero lo que lo hace realmente especial, es que el uso de sus técnicas de procesamiento digital, producen mucha más eficiencia y eficacia[15].

Desde su creación, el DSP ha sufrido cambios sustanciales, en arquitectura, dimensiones, formatos, en fin, en muchas de sus características. Inicialmente, sus primeras modificaciones se basaban en entregarle al usuario un dispositivo de gran rapidez de procesamiento matemático de bajo consumo de potencia; así la prioridad para los diseñadores fue reducir esta última variable. Luego de esto, se centraron mucho más en mejorar la relación de operaciones por segundo o ciclo. En algún momento también el enfoque de desarrollo estuvo dispuesto para hacer de este, un procesador mucho más dedicado a aplicaciones específicas, por lo cual agregaron a las tarjetas de desarrollo, sistemas para propósitos especiales tales como las telecomunicaciones, el procesamiento exclusivo de audio o video y periféricos para el control de motores[16][11][15].

El siguiente paso en la evolución de los DSP fue la implementación de una arquitectura multiproceso. Fue razonable el paso que dieron en la consecución de dicha característica, ya que modificaron la forma de programar una simple función.

Es evidente que con la tecnología actualmente desarrollada los fabricantes buscan aumentar cada vez más el poder de cálculo, la rapidez y la integración de los DSP, tanto así que las frecuencias de operación han pasado de unos cuantos megahertz hasta varios gigahertz.

Usualmente los aspectos que permiten identificar y seleccionar un DSP para una tarea específica son:

- **Tipo de aritmética – Ancho de palabra:** Determinar si un DSP es de punto fijo o de punto flotante es de vital importancia dado que los resultados de las operaciones realizadas por este (En especial, la operación de multiplicación) dependen directamente de esta característica. A partir de la aplicación se debe utilizar uno u otro, pero debe tenerse en cuenta que debe escogerse el DSP con el menor ancho de palabra que dicha aplicación pueda soportar. Cabe incluir la observación que en determinadas condiciones pueden tener el mismo desempeño, el de punto flotante posee un mayor costo y un mayor consumo de energía.
- **Rapidez:** Las diferentes arquitecturas que poseen los DSP, claramente proveen ventajas en ciertas áreas, una de esas es la rapidez de ejecución o procesamiento. Aunque los fabricantes suministran la información de las mediciones de MIPS (Millones de Instrucciones por Segundo) y de los MFLOPS (Millones de Instrucciones de Punto Flotante por Segundo), se recomienda hacer pruebas diagnósticas para finalmente seleccionar a partir de este aspecto, sobre todo cuando los valores de las dos mediciones son equivalentes o cercanas.
- **Tamaño de memoria:** Disponer de mayor cantidad de memoria interna es aconsejable ya que se puede realizar a mayor velocidad el acceso a esta. Así mismo, poseer una memoria flash interna reduce la complejidad del sistema.

- **Periféricos integrados:** Nuevamente este puede ser unos de los aspectos más importantes en la selección, pero solo dependiendo de la aplicación en especial. Ya que la utilización de periféricos que interactúen con el procesador marca un impacto directo sobre la calidad del procesamiento mismo. Los DSP de punto fijo son los que más poseen una variedad de dispositivos periféricos o interfaces de comunicación externa, son fabricados para aplicaciones determinadas, por ejemplo, para control de motores, pueden tener ADC (Conversor Analógico-Digital), DAC (Conversor Digital-Analógico), PWM (Modulación por Ancho de Pulso), en fin.
- **Consumo:** En aplicaciones en las cuales el consumo de energía es vital, debe seleccionarse un DSP dentro de una gama de bajo consumo, actualmente existen varias clasificaciones con una tensión nominal por debajo de los 3.3 (V) en donde los diseñadores están controlando el ahorro de energía a través del software o del hardware.
- **Costo:** En las aplicaciones de mayor consumo de energía, el costo del DSP es decisivo. En las de bajo consumo puede ser que solo sea un gasto, lo importante es que existen tal cantidad de diversos precios como fabricantes.
- **Rango dinámico:** La mayor precisión se puede obtener sin dudas con los DSP de punto flotante. El rango dinámico se refiere al máximo y al mínimo valor (diferente de cero) que puede ser representado, se expresa como una relación y es una consecuencia directa del tipo de aritmética y del ancho de palabra.
- **Software de desarrollo y soporte para el usuario:** Tener en cuenta que tipo de software de programación se utiliza para el dispositivo que se adquiera es esencial ya que dependerá de la experiencia del usuario en el conocimiento de lenguajes de programación la selección del mismo. Ya se mencionara más delante aspectos relevantes sobre este tema. De igual forma, que el fabricante ponga a disposición del diseñador todos los

recursos necesarios para el desarrollo de aplicaciones es un aspecto particular en la escogencia.

4 ANÁLISIS Y RESULTADOS

Ahora se mostrara el procedimiento realizado paso a paso para completar los objetivos planteados.

4.1 SELECCIONAR:

SE SELECCIONÓ UN DISPOSITIVO ADECUADO QUE PERMITIÓ REALIZAR EL PROCESAMIENTO DEL ALGORITMO CON ALTA CONFIABILIDAD Y RAPIDEZ.

La selección del procesador de señales digitales se hizo teniendo en cuenta la disponibilidad de los siguientes dispositivos en CEMOS (Grupo de Investigación en Control, Electrónica, Modelado y Simulación).

Se recopilo la información más relevante de los dos dispositivos que estaban disponibles en la Tabla 1. Cabe aclarar que la información mostrada fue extraída directamente de las hojas de datos de los dos dispositivos[17][18].

Observando la información, se seleccionó el DSP de *Texas Instruments Incorporated* de referencia TMS320C6711, ya que proveía una mayor frecuencia de procesamiento (150 MHz), aun cuando era de formato numérico Punto Flotante (*Float Point*), este era de 32 bits, lo cual permitía una mayor precisión y un rango dinámico de datos más amplio que el de Punto Fijo (*Fixed Point*) a 16 bits. Otra de las características decisivas para la selección fue la cantidad de millones de operaciones de punto flotante por segundo (MFLOPS) que podían realizar cada uno de ellos, sin lugar a dudas, la ventaja del DSP de *Texas Instruments Incorporated* era evidente.

Tabla 1. Comparativo entre los DSP de referencia TMS320C6711 de Texas Instruments Incorporated y MC56F8357 Freescale Semiconductor Incorporated.

Procesador de señales digitales Opción A	**Características**	**Procesador de señales digitales Opción B**
Texas Instruments Incorporated Punto flotante (*Floating point*)	**Fabricante**	*Freescale Semiconductor Incorporated* Punto fijo (*Fixed point*)
TMS320C6711	**Referencia**	MC56F8357
150 (MHz)	**Frecuencia (*Clock Rate*)**	60 (MHz)
32 (Bits)	**Formato numérico**	16 (Bits)
900 (MFLOPS)	**Instrucciones por segundo (MFLOPS - MIPS)**	60 (MIPS)
64 (Kilobytes)	**RAM interna (*Internal RAM*)**	16 (Kilobytes)
16 (Megabytes)	**RAM externa (*External RAM*)**	32 (Megabytes)
128 (Kilobytes)	**FLASH ROM**	256 (Kilobytes)
Harvard	**Arquitectura (*Architecture*)**	Harvard
VLIW	**Características especiales**	-

Finalmente y no por ello menos importante, la característica de tener arquitectura con tecnología VLIW, era un punto más a favor del DSP seleccionado ya que permitiría en el proceso de programación un mejor aprovechamiento de la segmentación y en consecuencia de la rapidez del mismo. Al poseer arquitectura VLIW este DSP resalta, ya que es una característica propia de los dispositivos de altas prestaciones y como en este caso tendría que soportar grandes volúmenes de datos numéricos y se buscaba que, en lo posible, el procesamiento se realizara de forma paralela con el fin de que en cada instrucción se ahorrara tiempo, apareció como el candidato ideal. Además esto permitiría reducir la carga operacional del procesador, ya que en los DSP con tecnología VLIW la

planificación de las instrucciones para mejorar el paralelismo, la realiza el compilador, más no el procesador[11].

Las principales propiedades del software que administra al DSP seleccionado serán mencionadas más adelante, que aunque son tantas como diversas, se presentaran solo las generalidades.

4.2 EFECTUAR:

SE EFECTUARON PRUEBAS DE DESEMPEÑO DEL PROCESO A TRAVÉS DE CINCO FUNCIONES ESTÁNDAR QUE SE EMPLEAN PARA ESTE PROPÓSITO Y, A PARTIR DE LOS RESULTADOS OBTENIDOS, SE ANALIZÓ SU EFICIENCIA.

El algoritmo que se utilizó en todas las pruebas fue el propuesto por Maurice Clerc [9] que posee un factor de constricción que asegura la convergencia de las partículas. La velocidad, la posición y la fórmula para el cálculo del factor de constricción son mostradas en las ecuaciones (3), (4) y (5), respectivamente:

$$v_{in}(k+1) = \chi\,[v_{in}(k) + c_1.r_1(k).(P_{in}(k) - x_{in}(k)) + c_2.r_2(k).(G_{in}(k) - x_{in}(k))] \quad (3)$$

$$x_{in}(k+1) = x_{in}(k) + v_{in}(k+1) \quad (4)$$

$$\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad (5)$$

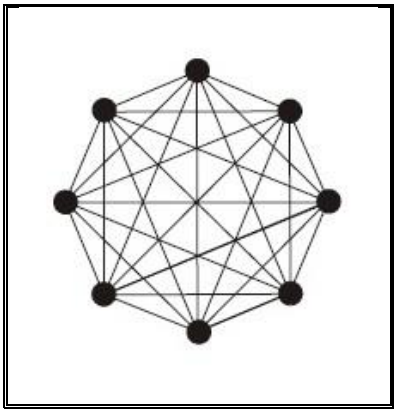
Donde $\varphi=c_1+c_2$ y $\varphi>4$.

Para efecto de este trabajo se tomó el valor de $\varphi=4.1$, siendo $c_1=c_2=2.05$ (Esto, para lograr el esquema completo del PSO, para que no tienda a resultados ni locales, ni globales y poder explorar el espacio de búsqueda uniformemente), lo cual resulta un factor de constricción de $\chi=0.729$. Los valores seleccionados son los comúnmente utilizados en las pruebas desempeño para el algoritmo en estudio[23][9][7][12][10].

Para mejorar el intercambio de información entre agentes, normalmente se establece una topología por medio de la cual las partículas crean una “vecindad” dada por el usuario programador. Cada una de las topologías posee una propiedad que la hace especial para una aplicación, entre las principales se encuentran: La global, la local, tipo abanico, de extensión y tipo estadística.

En el presente desarrollo se ha escogido la topología global, la cual se caracteriza porque las partículas o agentes se interrelacionan entre sí, sin una regla definida, teniendo así, de inmediato, los descubrimientos de las demás, como se muestra en la siguiente figura:

FIGURA 8. TOPOLOGÍA GLOBAL PARA EL PSO.



El algoritmo de optimización PSO fue aplicado utilizando las funciones estándares que se ilustraran en breve. La estrategia de implementación se realizó inicialmente en un computador personal con las siguientes características:

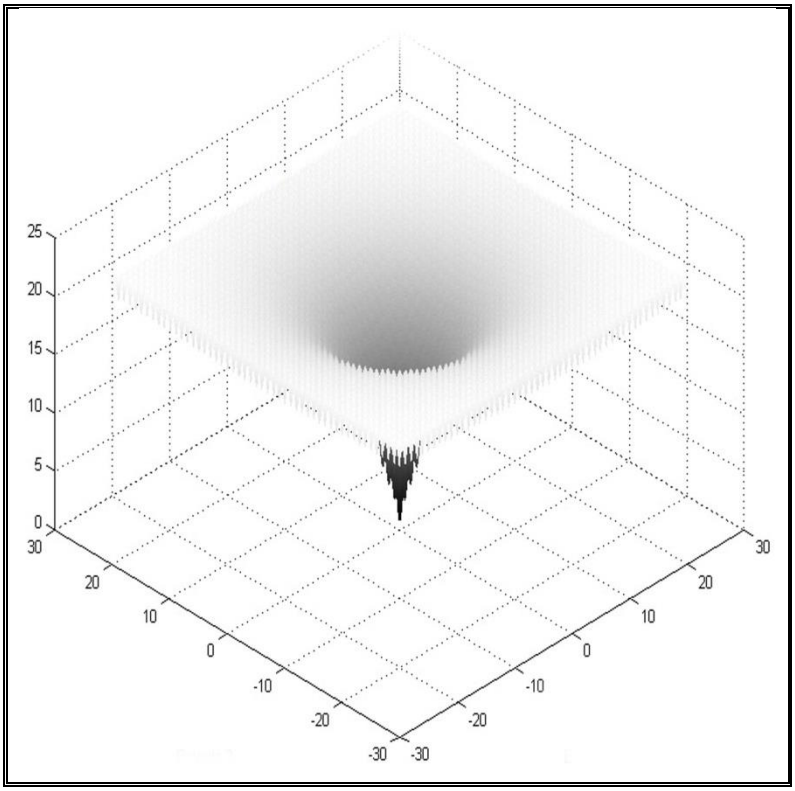
Tabla 2. Características PC.

Fabricante	IBM®
Modelo	8191LSS (KCGCOM9)
Procesador	Intel® Pentium® IV 2.8 GHz
RAM	2048 MB
Sistema operativo	Microsoft® Windows™ XP Professional 32-bit

Las funciones estándares de prueba (*Benchmark functions*) son útiles para evaluar ciertas características propias de los algoritmos de optimización, características tales como los tiempos de convergencia, la robustez, precisión, su comportamiento en general, es decir, su eficacia y su eficiencia. Se seleccionaron las siguientes funciones de prueba estándares, en su mayoría multidimensionales (Exceptuando la función Schaffer F6, que solo posee dos) [5][24][25][23][14]:

4.2.1 Función Ackley:

FIGURA 9. FUNCIÓN ACKLEY.



$$\boldsymbol{f(x) = 20 + exp(1) - 20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right]}$$

El área de prueba para esta función usualmente está restringida al hipercubo comprendido en:

$$-32.768 \leq x_i \leq 32.768$$

$$i = 1,2,3,\ldots,n$$

La función Ackley tiene un mínimo global:

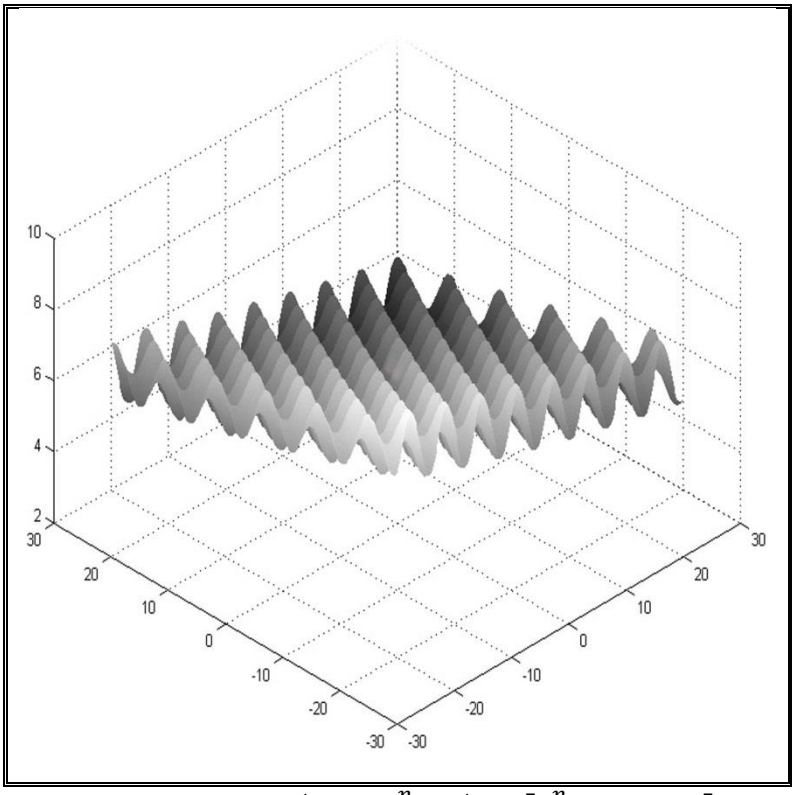
$$f(x_i) = 0$$

Para

$$x_i = 0$$

4.2.2 Función Griewank:

FIGURA 10. FUNCIÓN GRIEWANK.



$$f(x) = 1 + \left(\frac{1}{4000}\sum_{i=1}^{n} x_i^2\right) - \left[\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)\right]$$

El área de prueba para esta función usualmente está restringida al hipercubo comprendido en:

$$-600 \leq x_i \leq 600$$

$$i = 1,2,3,\ldots,n$$

La función Griewank tiene un mínimo global:

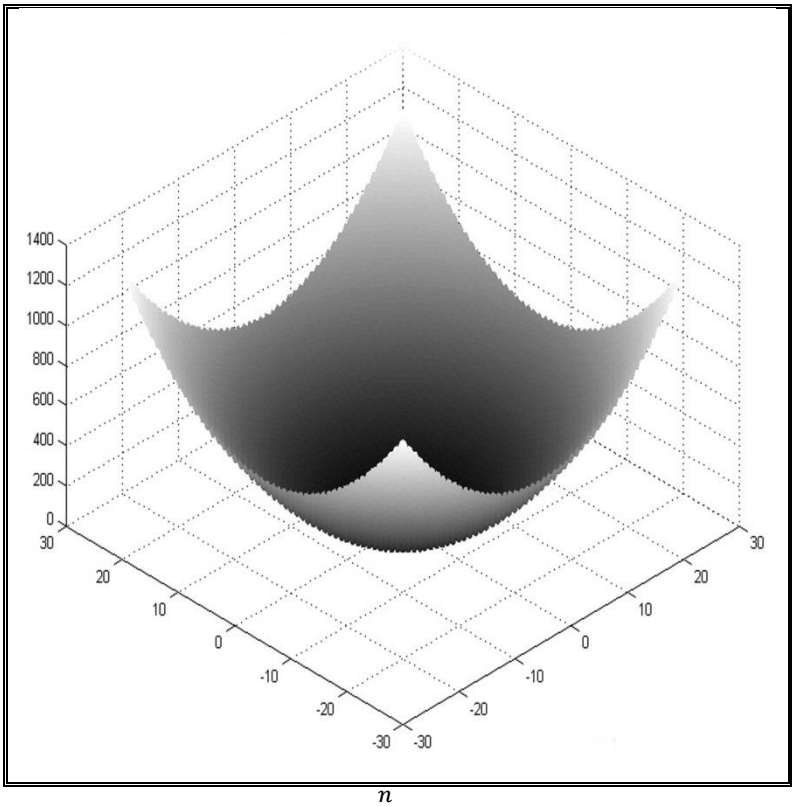
$$f(x_i) = 0$$

Para

$$x_i = 0$$

4.2.3 Función Rastrigin:

FIGURA 11. FUNCIÓN RASTRIGIN.



$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$$

El área de prueba para esta función usualmente está restringida al hipercubo comprendido en:

$$-5.12 \leq x_i \leq 5.12$$

$$i = 1,2,3,\ldots,n$$

La función Rastrigin tiene un mínimo global:

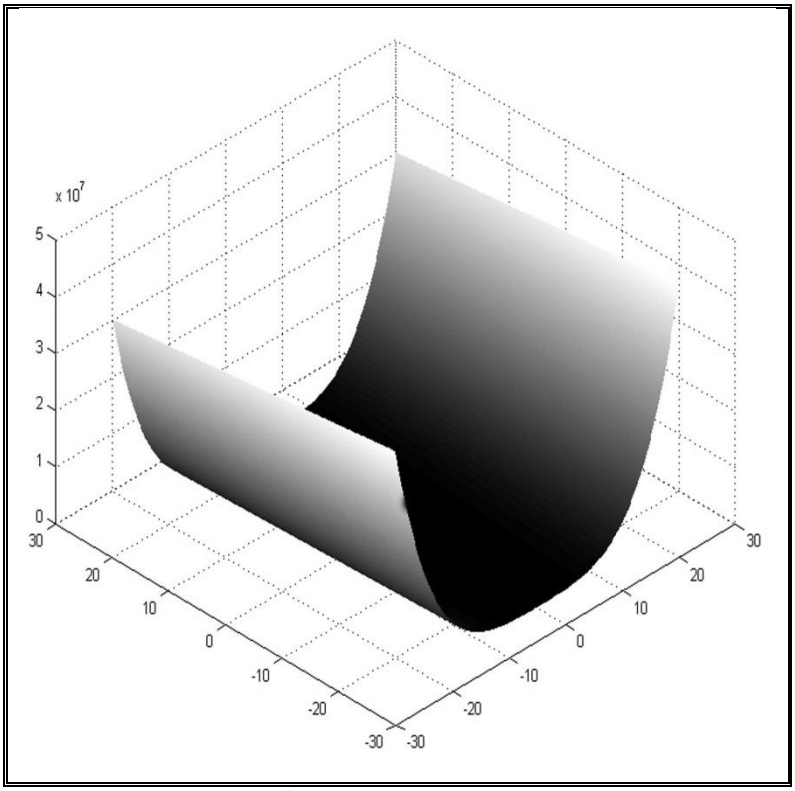
$$f(x_i) = 0$$

Para

$$x_i = 0$$

4.2.4 Función Rosenbrock:

FIGURA 12. FUNCIÓN ROSENBROCK.



$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

El área de prueba para esta función usualmente está restringida al hipercubo comprendido en:

$$-5.12 \leq x_i \leq 5.12$$

$$i = 1,2,3, \dots, n$$

La función Rosenbrock tiene un mínimo global:

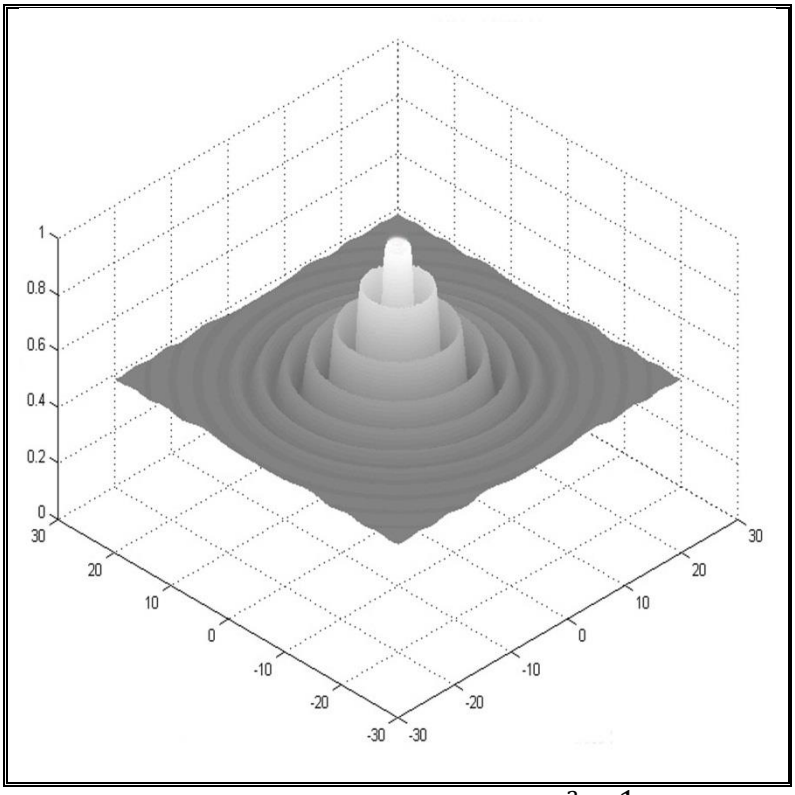
$$f(x_i) = 0$$

Para

$$x_i = 0$$

4.2.5 Función Schaffer F6:

FIGURA 13. FUNCIÓN SCHAFFER F6.



$$f(x) = \frac{1}{2} + \frac{\sin\left(\sqrt{x^2 + y^2}\right)^2 - \frac{1}{2}}{[1 + 0.01(x^2 + y^2)]^2}$$

El área de prueba para esta función usualmente está restringida o comprendida en:

$$-100 \leq x, y \leq 100$$

La función Schaffer F6 tiene un mínimo global:

$$f(x, y) = 0$$

Para

$$x = 0$$

$$y = 0$$

Como se pudo observar, además de la representación gráfica y su fórmula se dieron a conocer los mínimos que poseen cada una ellas, tanto su posición como también su valor, incluyendo así mismo los intervalos de prueba usualmente usados para este algoritmo de optimización, incluso los valores de los errores aceptables en los resultados, tal y como es resumido en la siguiente tabla:

Tabla 3. Características de las funciones estándares seleccionadas.

	Función	Dimensiones	Intervalos	Mínimos		Error (Aceptable)
1	**Ackley**	n	[-32.768,32.768]	$f(x_i)=0$	i=1,2,3,…,n para $x_i=0$	< 0.00001
2	**Griewank**	n	[-600,600]	$f(x_i)=0$	i=1,2,3,…,n para $x_i=0$	< 0.1
3	**Rastrigin**	n	[-5.12,5.12]	$f(x_i)=0$	i=1,2,3,…,n para $x_i=0$	< 100
4	**Rosenbrock**	n	[-5.12,512]	$f(x_i)=0$	i=1,2,3,…,n para $x_i=0$	< 0.01
5	**Schaffer F6**	2	[-100,100]	$f(x_i)=0$	i=1,2 para $x_i=0$	< 0.001

Las pruebas se realizaron con valores específicos de rapidez y cantidad de partículas. Dichos valores fueron determinados según la literatura sobre PSO [26] y las funciones de prueba, ya que en varios informes mencionan que 25 partículas son suficientes para explorar el espacio de búsqueda, así mismo la rapidez igual a 5 no está situada en ninguno de los extremos en los cuales podría presentar problemas el algoritmo. Aunque, cabe resaltar que la selección de dichos valores no afecta visiblemente las observaciones y conclusiones que se harán más

adelante, ya que la comparación a realizar es uno a uno y se necesita simplemente de una igualdad de condiciones.

Las pruebas que se mostraran en breve se llevaron a cabo variando la cantidad de dimensiones para cada una de las funciones entre 2, 5, 10, 25, 50 y 100 (Las dimensiones se variaron ya que los tiempos de procesamiento normalmente aumentan directamente con la cantidad de las mismas), los resultados tabulados corresponden al promedio de los valores obtenidos de repetir el proceso doscientas cincuenta (250) veces. Es importante aclarar que se descartaron las primeras cincuenta (50) para lograr baja variabilidad y mayor estabilidad. Se presentaran las iteraciones que requirió el algoritmo de entre mil (1000) posibles de ellas para encontrar el mínimo propio de cada función y el tiempo que necesito para alcanzar dicho objetivo.

Con el banco de funciones de prueba se pretende someter al algoritmo a funciones de distintos tipos y grados de dificultad y así probar la eficiencia de este en situaciones distintas también, para finalmente compararlo objetivamente y de cierta manera medible.

Los resultados obtenidos para cada una de las funciones son los que se encuentran en la Tabla 4:

Tabla 4. Resultados pruebas con el PC.

(Donde la letra D representa las dimensiones)

Función	D	Iteraciones requeridas		Mínimo encontrado		Tiempo de procesamiento (s)	
		$\overline{x}$	Σ	$\overline{x}$	σ	$\overline{x}$	σ
Ackley	2	311	3.19	8.88e-16	2.54e-16	58.78	2.41
	5	376	21.9	1.06e-16	2.95e-17	57.88	2.31
	10	437	10.9	2.09e-15	6.02e-16	61.31	0.38
	25	771	20.4	9.61e-06	2.87e-06	106.0	1.75
	50	952	14.3	1.99e-05	5.52e-06	144.3	1.26
	100	921	6.13	1.01e-02	2.90e-03	113.0	0.90
Griewank	2	227	7.96	0.00000	0.00000	29.23	2.54
	5	354	14.6	9.04e-09	2.70e-09	50.56	0.17
	10	327	7.64	1.93e-09	5.72e-10	41.97	0.53
	25	999	29.1	1.07e-07	3.13e-08	118.0	2.30
	50	978	21.8	1.60e-07	4.51e-08	118.1	2.42
	100	990	25.5	1.00e-05	3.08e-06	131.4	0.41
Rastrigin	2	226	7.84	0.00000	0.00000	28.95	2.65
	5	328	8.13	4.44e-11	1.33e-11	49.08	2.59
	10	333	9.52	0.99e-09	2.98e-10	41.45	0.42
	25	759	17.3	3.00e-12	8.37e-13	86.05	1.72
	50	914	4.14	3.00e-05	8.43e-06	114.7	1.35
	100	833	8.88	2.69e-02	7.50e-03	95.33	1.55
Rosenbrock	2	705	1.41	0.00000	0.00000	86.54	1.98
	5	1000	0.28	8.07e-11	2.40e-11	123.1	0.89
	10	1000	0.30	5.02e-08	1.45e-08	119.7	2.79
	25	775	21.6	3.74e-06	1.11e-06	94.15	1.18
	50	1000	0.29	1.56e-02	4.60e-03	118.8	2.46
	100	775	21.5	7.90e-04	2.25e-04	117.2	2.08
Schaffer F6	2	220	5.69	0.00000	0.00000	30.02	5.80e-3

4.3 DESARROLLAR:

SE DESARROLLÓ LA PROGRAMACIÓN DEL ALGORITMO DE OPTIMIZACIÓN APROVECHANDO LAS CARACTERÍSTICAS DEL DSP.

La programación de PSO en el DSP se hizo inicialmente por medio del programa exclusivo de *Texas Instruments Incorporated* conocido como *Code Composer Studio*^{TM} (CCS). Se utilizó la versión 2.10 que contenía el DSK de la tarjeta de desarrollo del mismo DSP. Este programa brinda un ambiente de desarrollo integrado (IDE) que posee en sí mismo el compilador C, el ensamblador, el enlazador y el depurador de algoritmos. El sistema operativo usado para ejecutar el mencionado software fue *Microsoft*® *Windows*^{TM} XP ya que no soportaba versiones posteriores y además se debía hacer conexión con *MatLab*® 6.5 de *MathWorks* a través de su herramienta de S*imulink*. Para el desarrollo del proyecto se presentaron tres opciones diferentes para poder programar el DSP [6]:

- Se podía codificar el algoritmo directamente en lenguaje *Assembler* lo cual proveía de un buen desempeño del algoritmo, aun así implementarlo por este método requería de un gran cantidad de tiempo de desarrollo.
- Se podía codificar el algoritmo en lenguaje C y luego de esto utilizar el compilador, el enlazador, el ensamblador y el depurador de CCS, lo que permitiría que aunque el código inicial no fuera eficiente, al utilizar las propiedades del ambiente de desarrollo, este optimizaría los recursos del mismo procesador.
- Se podía utilizar *MatLab*® y su entorno de desarrollo *Simulink* ya que posee herramientas (*Real Time Workshop*) para administrar el DSK, igualmente podía codificarse en lenguaje C y ser enviado al DSP.

Con respecto a los lenguajes de programación, se puede decir que *Assembly* fue una vez el más utilizado en los DSP, exigía al programador, administrar de una u otra manera los registros y los eventos del núcleo. En este lenguaje se tenía la ventaja de poder optimizar completamente un programa, aunque ello conllevaba a un gasto de tiempo mayor.

El lenguaje más comúnmente utilizado es el C, que para poder ser usado con el DSP necesita de un compilador que transforme el código C en código *Assembly*, dadas unas instrucciones. Los algoritmos programados por medio del lenguaje C pueden ser implementados en diversas plataformas, lo cual es una ventaja. El lenguaje del cual se hace uso en el CCS es una combinación entre C y *Assembly* conocida como Código Ensamblador Lineal (*Linear Assembly Code*). Entre los principales beneficios están que permite trabajar con nombres simbólicos, se logra una mayor eficiencia del DSP y otro ya mencionado, libra al programador de trabajar directamente con los registros del núcleo.

La utilización del DSP se basó completamente en los recursos ofrecidos por el mismo fabricante (*Texas Instruments Incorporated*) y que se pueden encontrar en línea, además de los documentos informativos que el mismo kit de desarrollo trae consigo. La guía del programador (*TMS320C6000 Programmer's Guide*)[19], el tutorial del *Code Composer Studio*^{TM} (*TMS320C6000 Code Composer Studio Tutorial*)[20], una guía de iniciación (*How to Begin Development with the TMS320C6711 DSP*)[21] y las respectivas notas de aplicación para la herramienta de *Simulink* mencionada (*MATLAB Link for Code Composer Studio Development Tools Release Notes*)[22].

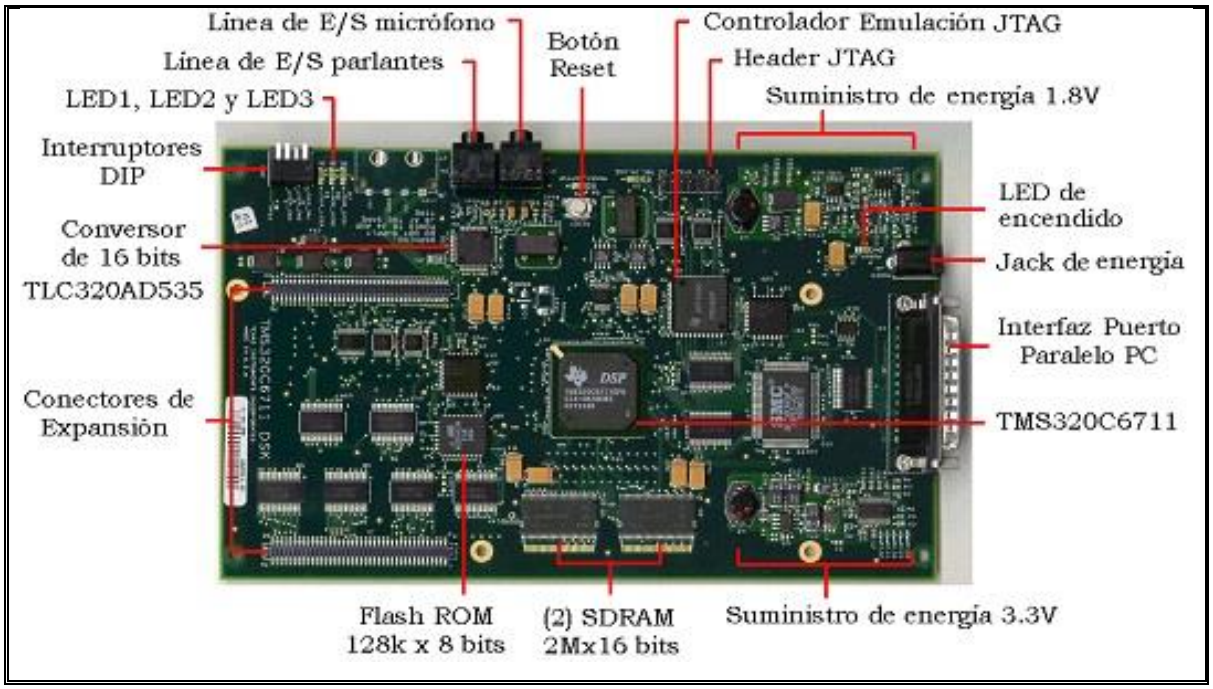
4.4 VISUALIZAR:

SE VISUALIZÓ Y TABULÓ EL PROCESO DE OPTIMIZACIÓN QUE REALIZÓ PSO DESARROLLADO EN EL DSP, ANALIZANDO SUS CARACTERÍSTICAS ESPECÍFICAS.

Se realizó nuevamente la recolección de datos hecha en el paso 2, solo que esta vez se implementó el algoritmo de optimización PSO en el DSP seleccionado.

Se hizo uso del kit de inicio (DSK) TMDS3206711 de *Texas Instruments Incorporated*. Este kit contaba con los siguientes elementos:

FIGURA 14. TARJETA DE DESARROLLO TMDS320C6711. TOMADA DE [6].



- Fuente de poder universal (UPS).

- CD-ROM de *Code Composer Studio*^{TM}.
- Tarjeta del kit de inicio del DSP (DSK).
- Cable de puerto paralelo (DB25).
- Cable AC.

Para aprovechar completamente las ventajas que muestra el DSP el procesamiento de cada una de las funciones, en cada una de las dimensiones se realizó sin utilizar las características gráficas y demás de visualización que posee el ambiente de desarrollo integrado (IDE) del *Code Composer Studio*^{TM} para eliminar las fuentes de consumo de recursos en tiempo real. Luego de realizadas las 250 repeticiones se sustraían los datos procesados y estos a su vez se administraban luego con el software *MatLab*® 6.5 de *MathWorks*, para poder tratarlos gráfica y estadísticamente.

Es importante mencionar ahora, que anteriormente se había hecho exactamente lo mismo con el computador personal, ya que en lo posible se cerraron o eliminaron por completo procesos activos al iniciar el PC, así mismo se disminuyó al mínimo la personalización gráfica y demás prestaciones innecesarias para el procesamiento.

Los resultados obtenidos para cada una de las funciones son los siguientes (Tabla 5):

Tabla 5. Resultados pruebas con el DSP.

(Donde la letra D representa las dimensiones)

Funciones	D	Iteraciones requeridas		Mínimo encontrado		Tiempo de procesamiento (s)	
		$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ
Ackley	2	326	7.08	8.88e-16	2.68e-16	0.668	2.09e-02
	5	353	15.3	3.52e-16	9.31e-17	0.600	3.00e-04
	10	345	13.0	1.57e-15	4.72e-16	0.574	2.15e-02
	25	793	25.7	0.99e-05	2.85e-06	0.932	8.90e-03
	50	741	11.9	0.99e-05	2.88e-06	1.428	1.24e-01
	100	916	4.54	2.58e-02	8.10e-03	1.698	2.00e-01
Griewank	2	239	11.5	7.40e-16	2.26e-16	0.542	1.22e-02
	5	248	14.5	5.43e-09	1.59e-09	0.508	2.20e-03
	10	311	2.99	3.24e-09	9.62e-10	0.594	2.66e-02
	25	750	13.8	1.30e-08	3.66e-09	0.918	5.30e-03
	50	767	20.2	3.90e-07	1.11e-07	1.389	1.11e-01
	100	804	1.17	5.94e-03	1.70e-03	1.688	1.91e-01
Rastrigin	2	237	10.7	0.00000	0.00000	0.504	1.10e-03
	5	336	10.3	1.04e-09	2.68e-10	0.599	2.99e-02
	10	373	20.7	1.99e-09	5.71e-10	0.609	2.40e-03
	25	927	8.26	1.93e-09	5.39e-10	1.080	2.33e-02
	50	1000	0.29	1.00e-04	2.71e-05	1.283	8.15e-02
	100	1000	0.29	1.15e-02	3.30e-03	1.736	2.05e-01
Rosenbrock	2	726	7.19	0.00000	0.00000	0.708	2.20e-03
	5	1000	0.28	7.02e-11	1.95e-11	0.835	9.70e-03
	10	979	23.2	5.55e-08	1.61e-08	0.879	2.26e-02
	25	953	15.0	2.63e-06	7.62e-09	1.069	1.99e-02
	50	976	19.9	1.68e-02	4.80e-03	1.329	9.21e-02
	100	1000	0.29	2.49e-02	7.30e-3	1.798	2.26e-01
Schaffer F6	2	212	3.50	0.00000	0.00000	0.507	2.00e-03

4.5 VERIFICAR:

SE VERIFICÓ EL DESEMPEÑO DEL MÉTODO DE OPTIMIZACIÓN PSO EJECUTADO POR MEDIO DE UN DSP, OBTENIENDO GRÁFICAS Y TABLAS DE SU COMPORTAMIENTO, ENTRE OTRAS.

A continuación se analizaran los datos de las pruebas efectuadas del PSO tanto en un PC como en un DSP.

En la Tabla 6. Se muestra la diferencia de error de los procesos de las funciones ejecutadas, es decir, se hace la diferencia entre el valor del mínimo encontrado por el PC y el encontrado por el DSP. Lo anterior con el objetivo de detectar la diferencia en la precisión con la que los procesos consiguen el valor del mínimo encontrado.

Si observamos la tabla 6 podemos suponer que existe una tendencia a disminuir la precisión del proceso ejecutado en el DSP. La mayor diferencia se dio en la función Griewank con dos dimensiones, en la cual se obtuvo un -7.40e-16, lo que quiere decir que el PSO obtuvo un mejor valor para el mínimo de la función en el origen del espacio, que cuando este fue ejecutado en el PC.

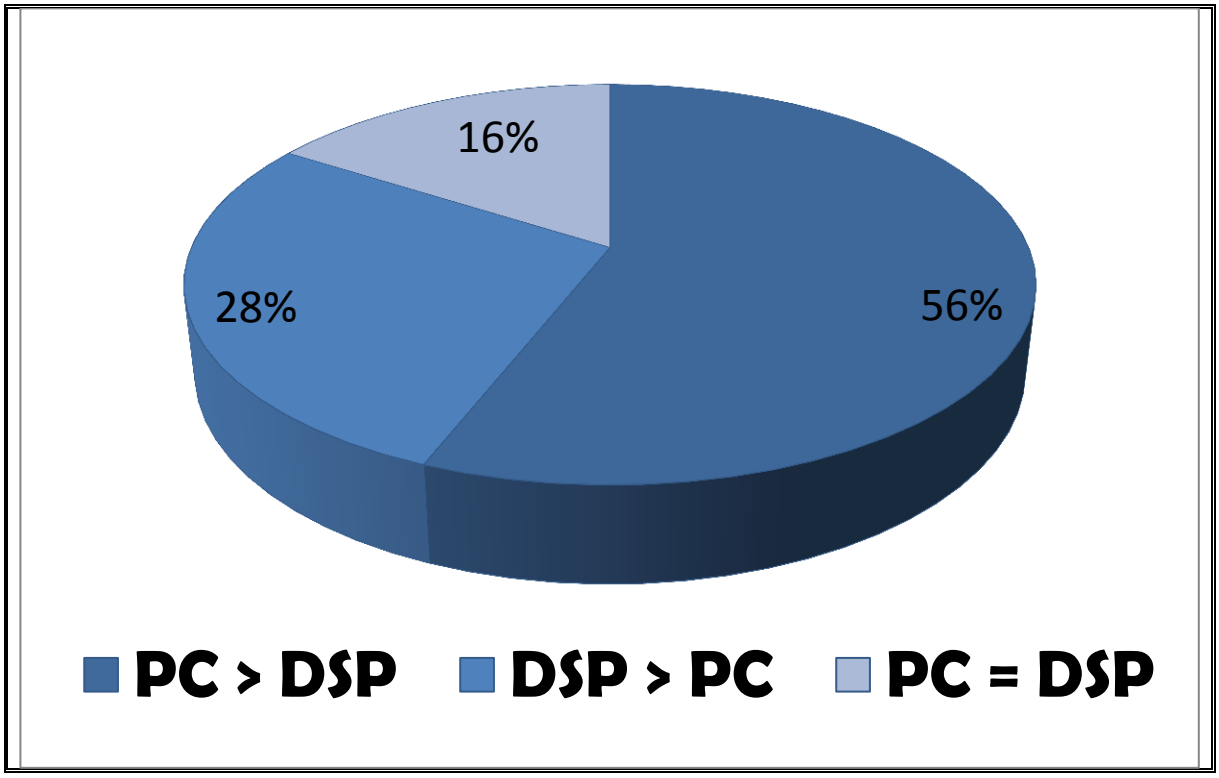
Un efecto particular se revela al aumentar las dimensiones en cada una de las funciones, y es que la diferencia disminuye entre los dos procesos de implementación (al mismo tiempo que decae la precisión del valor mínimo encontrado), generando así la menor diferencia (diferente de cero), cuyo valor es de 1.54e-2 para la función Rastrigin en la dimensión 100. El proceso alcanza la igualdad en sus resultados en las funciones Ackley, Rastrigin, Rosenbrock y

Schaffer F6 para dos dimensiones. Los promedios obtenidos para los valores de estos mínimos fueron 0.0.

Tabla 6. Diferencia de error entre los valores mínimos encontrados por el PC y el DSP.

Función	Dimensiones	Diferencia de error PC - DSP
Ackley	2	0.00000
	5	-2.46e-16
	10	5.12e-16
	25	-2.66e-07
	50	9.96e-06
	100	-1.57e-02
Griewank	2	-7.40e-16
	5	3.60e-09
	10	-1.31e-09
	25	9.40e-08
	50	-2.30e-07
	100	-5.93e-03
Rastrigin	2	0.00000
	5	-9.92e-10
	10	-9.96e-10
	25	-1.93e-09
	50	-6.98e-05
	100	1.54e-02
Rosenbrock	2	0.000000
	5	1.05e-11
	10	-5.28e-09
	25	1.10e-06
	50	-1.21e-03
	100	-2.41e-02
Schaffer F6	2	0.00000

FIGURA 15. COMPARATIVO DE LA PRECISIÓN DE LOS SISTEMAS PSO-PC Y PSO-DSP CON RESPECTO A LA DIFERENCIA DE ERROR EN LOS VALORES DE LA FUNCIÓN OBJETIVO



La Figura 15 representa aquellos casos (de los 25 de la Tabla 6) en los que la precisión del PC fue mayor que la del DSP (56%), los casos en los que el DSP obtuvo mayor precisión (28%) y aquellos en los que la precisión fue igual (16%).

De igual forma, las siguientes Figuras (16-17-18-19-20) realizan la interpretación gráfica de la tabla anterior para cada una de las funciones.

FIGURA 16. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN ACKLEY.

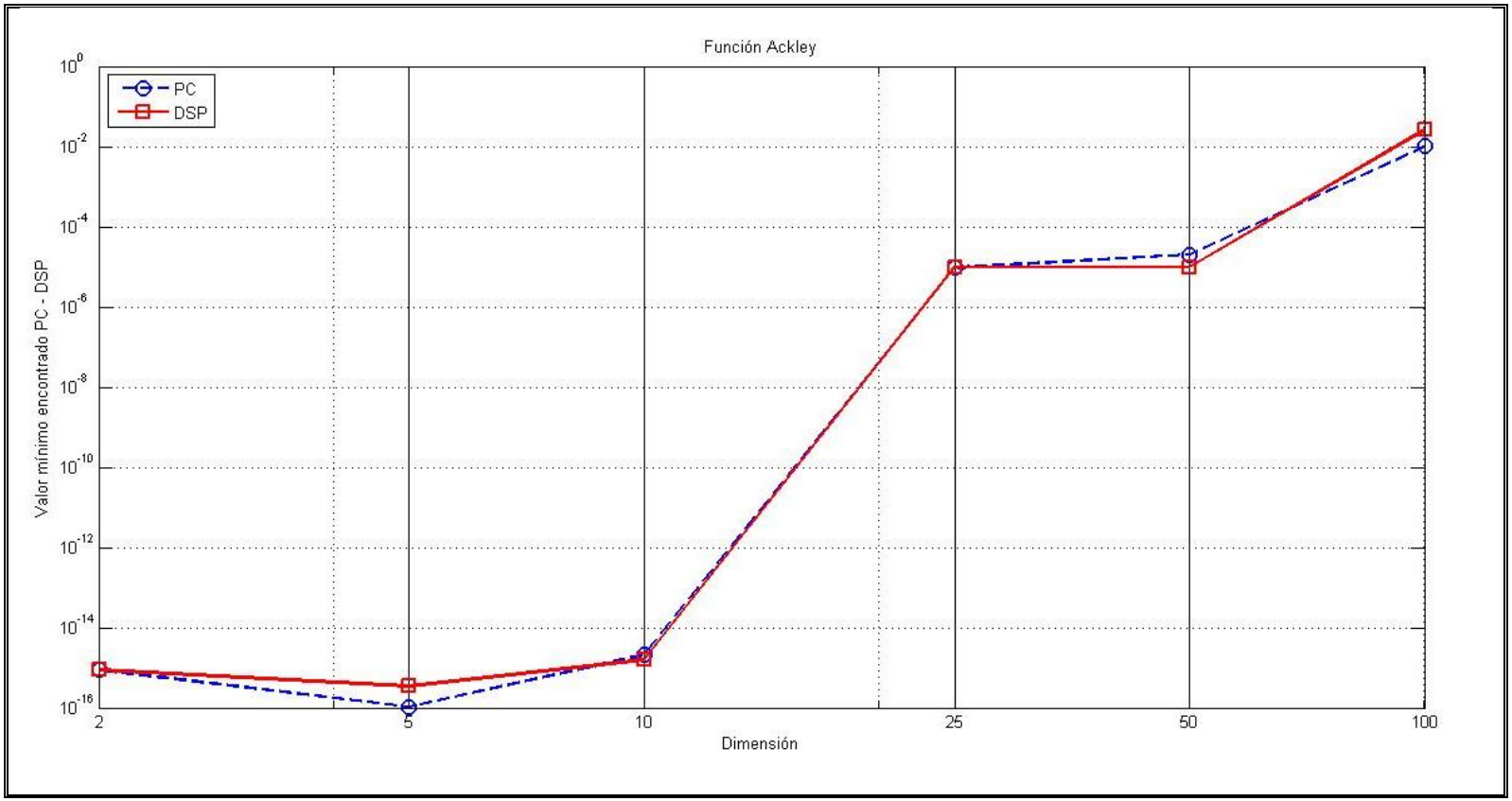


FIGURA 17. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN GRIEWANK.

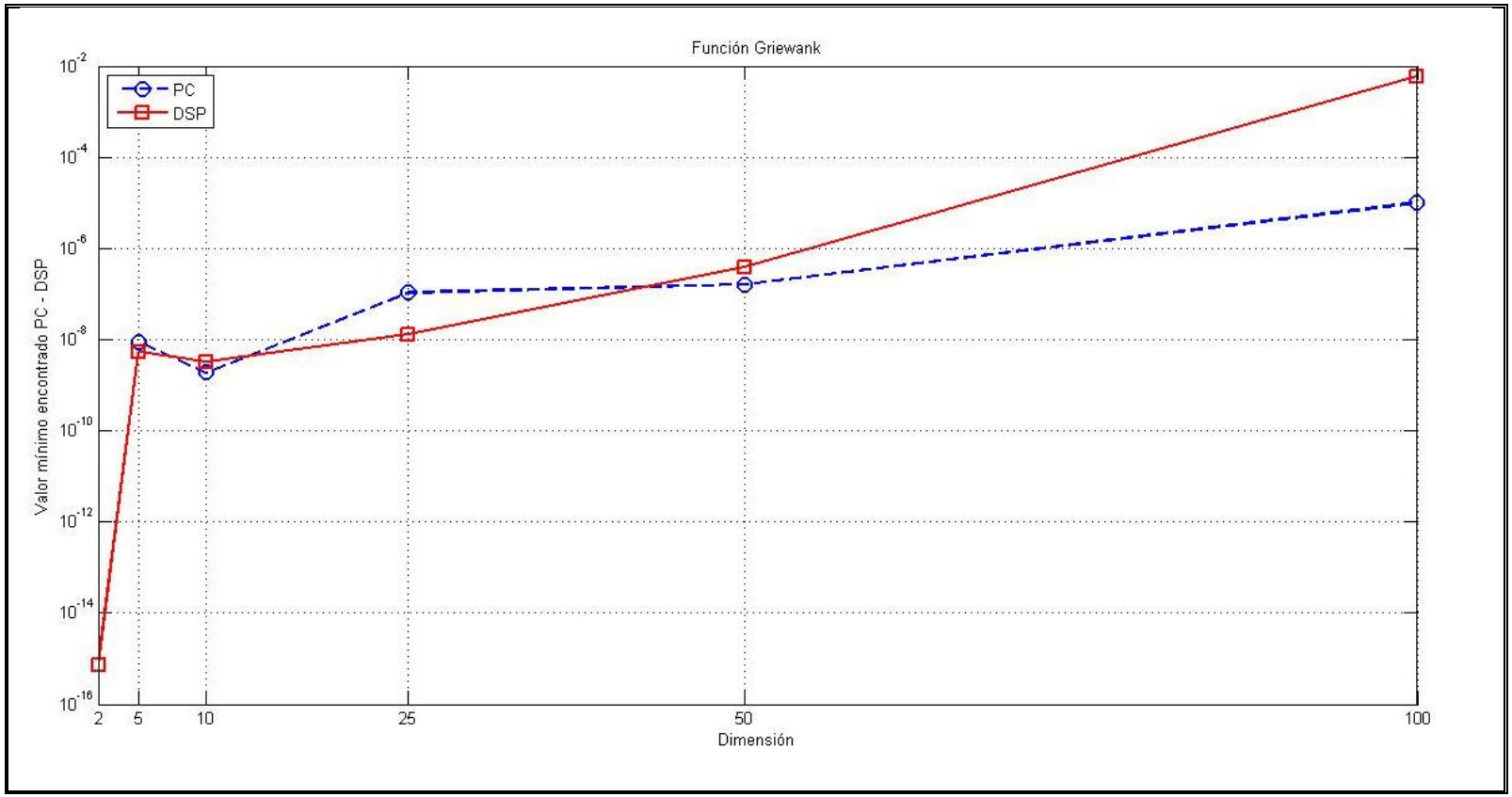


FIGURA 18. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN RASTRIGIN.

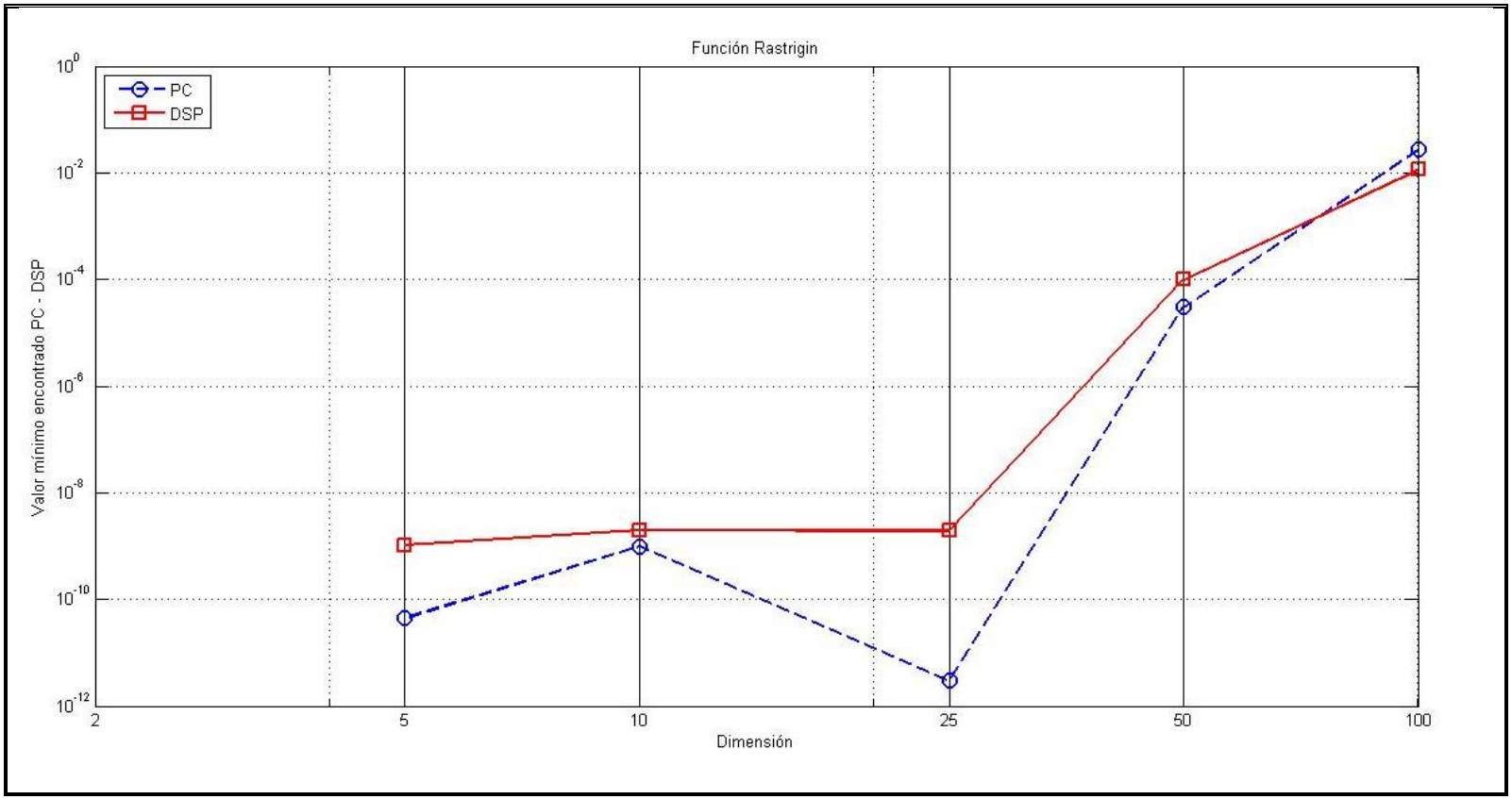


FIGURA 19. COMPARATIVO DE LOS VALORES MÍNIMOS ENCONTRADOS POR EL PC Y EL DSP PARA LA FUNCIÓN ROSENBROCK.

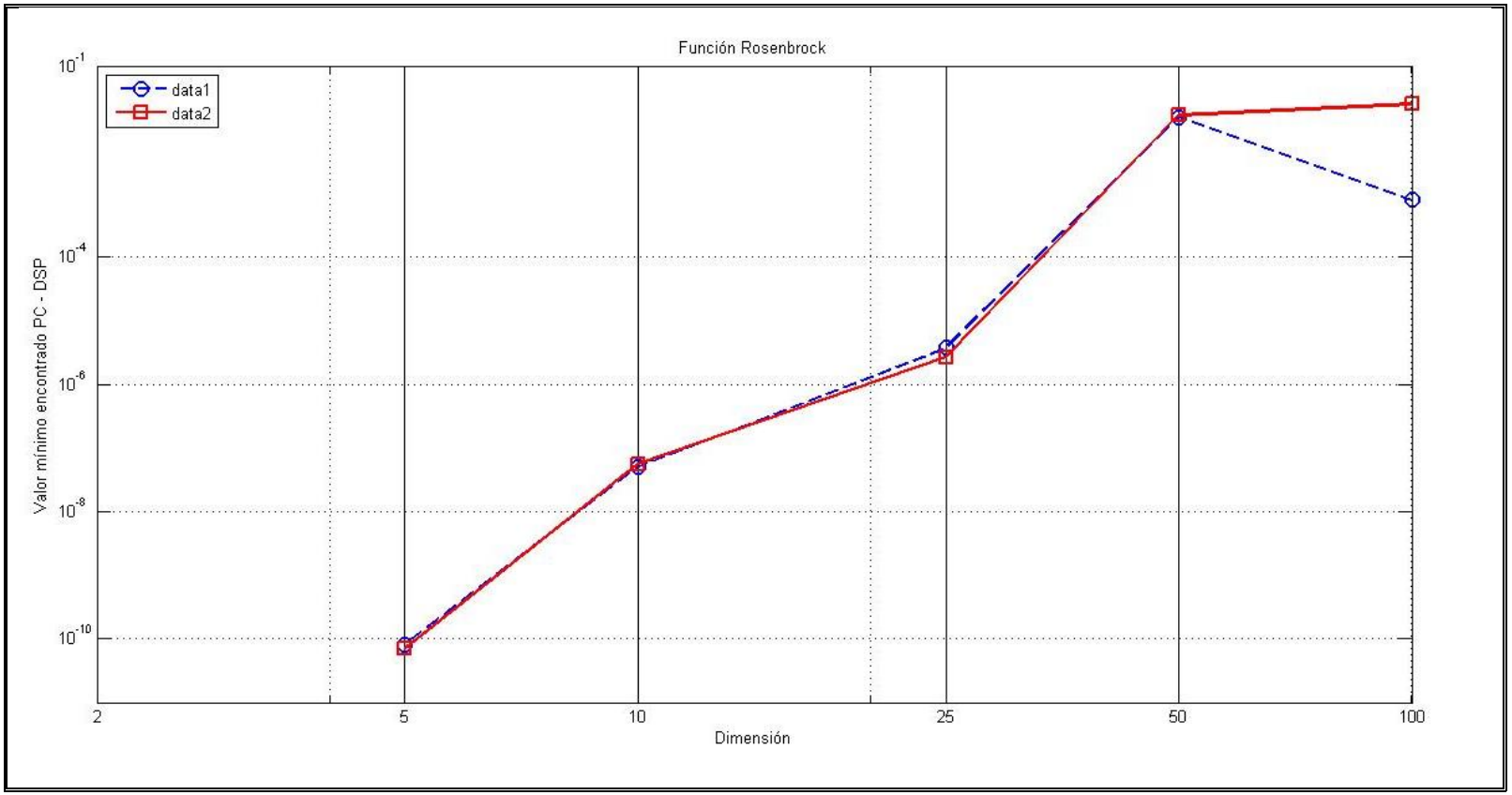
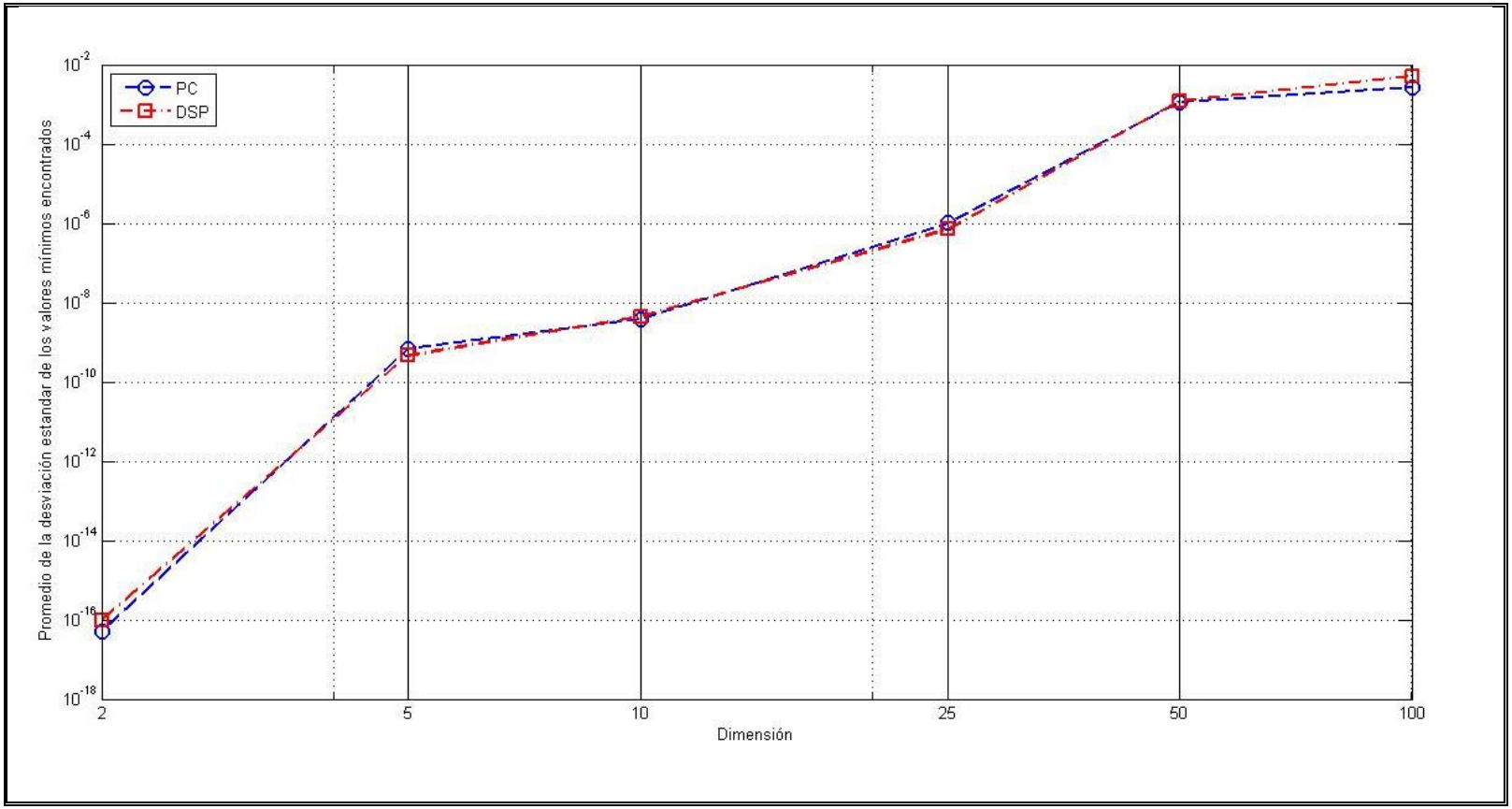


FIGURA 20. PROMEDIO DE LA DESVIACIÓN ESTÁNDAR DE LOS VALORES MÍNIMOS ENCONTRADOS.



En la siguiente tabla se presenta la relación existente entre el tiempo requerido por el PC para encontrar el valor del mínimo obtenido sobre el requerido por el DSP.

Tabla 7. Relación de tiempos de procesamiento entre el PC y el DSP.

Función	Dimensiones	Relación Tiempo PC / Tiempo DSP
Ackley	2	87.99
	5	96.42
	10	106.8
	25	113.7
	50	101.0
	100	66.54
Griewank	2	53.92
	5	99.42
	10	70.64
	25	128.6
	50	85.08
	100	77.86
Rastrigin	2	57.37
	5	81.87
	10	68.09
	25	79.68
	50	89.42
	100	54.92
Rosenbrock	2	122.1
	5	147.4
	10	136.2
	25	88.04
	50	89.42
	100	65.19
Schaffer F6	2	59.16

La relación anteriormente mencionada y mostrada, significa calcular el cociente entre el tiempo de procesamiento que invirtió el PC en ejecutar PSO en cada una de las funciones y en cada una de las dimensiones con el que consumió el DSP haciendo las mismas actividades, lo cual nos da un aproximado de cuantas veces es una estrategia de implementación más rápida o lenta que la otra.

En la totalidad de las funciones se puede afirmar, que la implementación PSO-DSP es al menos 50 veces más rápida que la del PSO-PC. En algunos casos particulares, como en la función Rosenbrock, dicha implementación es casi 150 veces más rápida.

En forma gráfica los datos contenidos en la Tabla 7 se presentarían en la Figura 21, así mismo en la Figura 22 se encuentra el promedio de la desviación estándar de los tiempos de procesamiento para los dos sistemas, ya que los valores usados de la Tabla 7, se refieren a promedios obtenidos anteriormente.

FIGURA 21. COMPARATIVO DE LA RELACIÓN DE TIEMPOS DE PROCESAMIENTO PARA LAS FUNCIONES DE PRUEBA (TIEMPO PC / TIEMPO DSP).

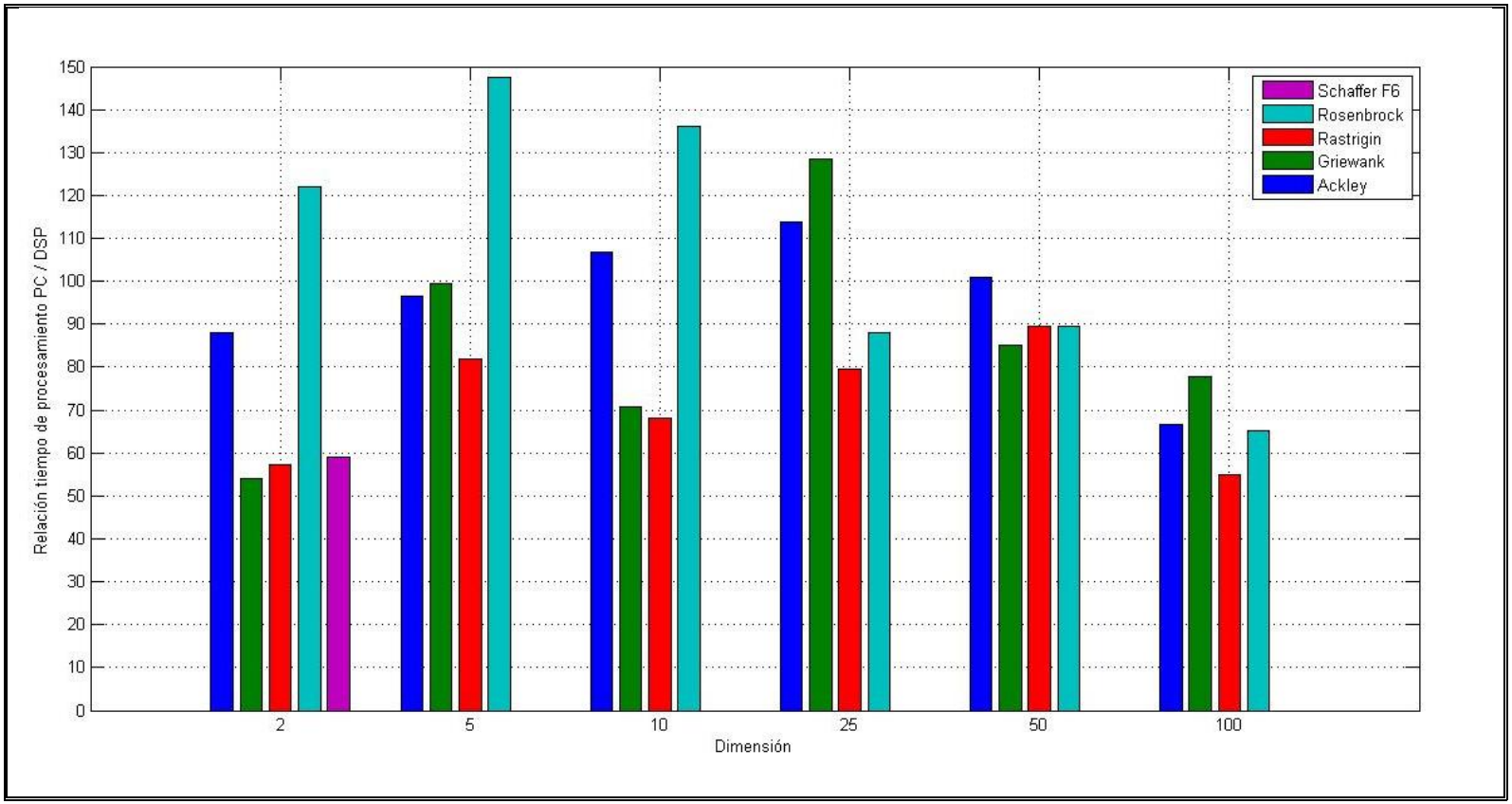
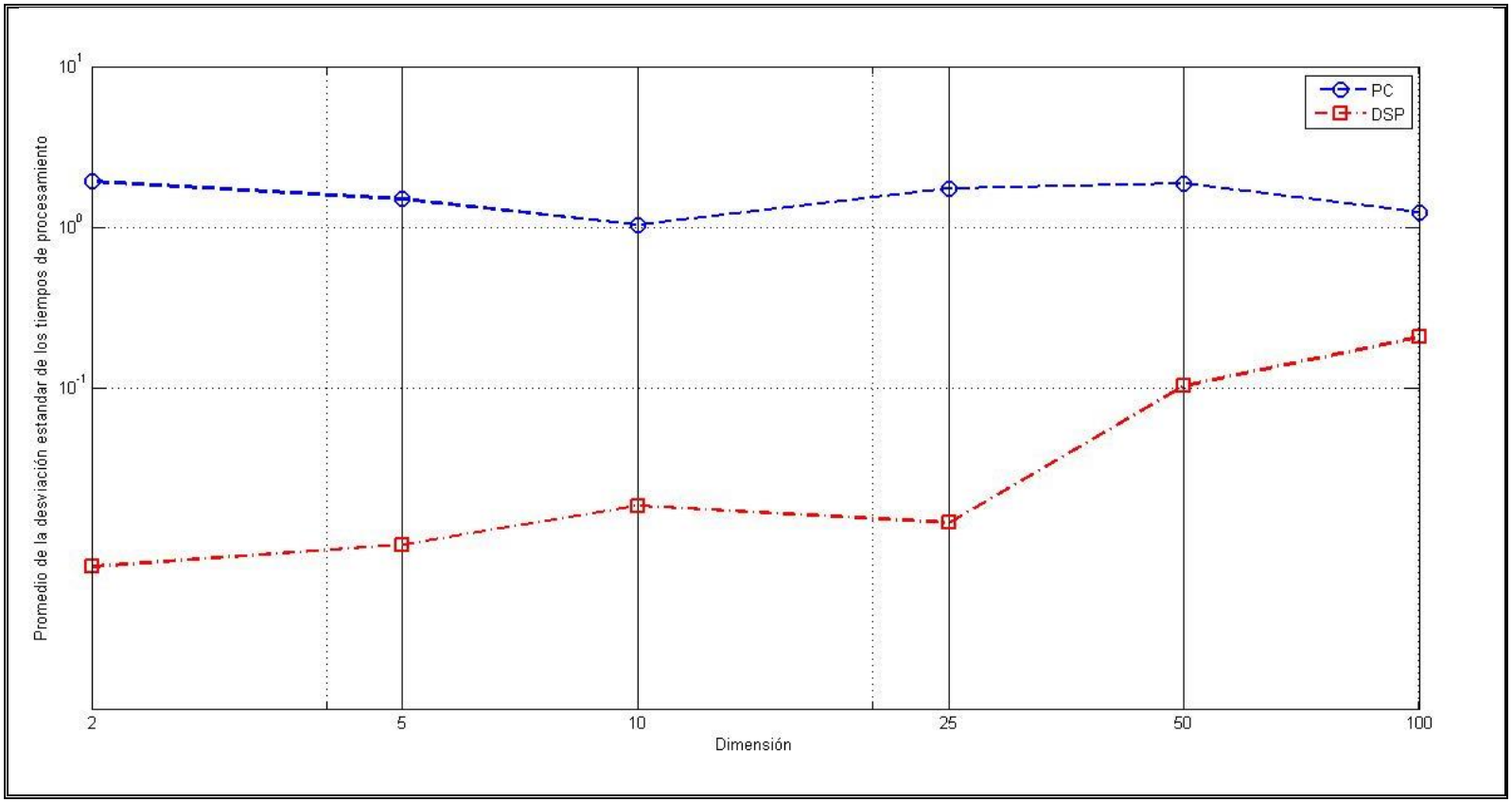


FIGURA 22. PROMEDIO DE LA DESVIACIÓN ESTÁNDAR DE LOS TIEMPOS DE PROCESAMIENTO.



La relación entre las iteraciones requeridas tanto por el PC como por el DSP para hallar el valor del mínimo encontrado en cada una de las funciones y cada una de las dimensiones se expresa de la siguiente manera (Figuras 23-24-25-26-27):

FIGURA 23. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN ACKLEY.

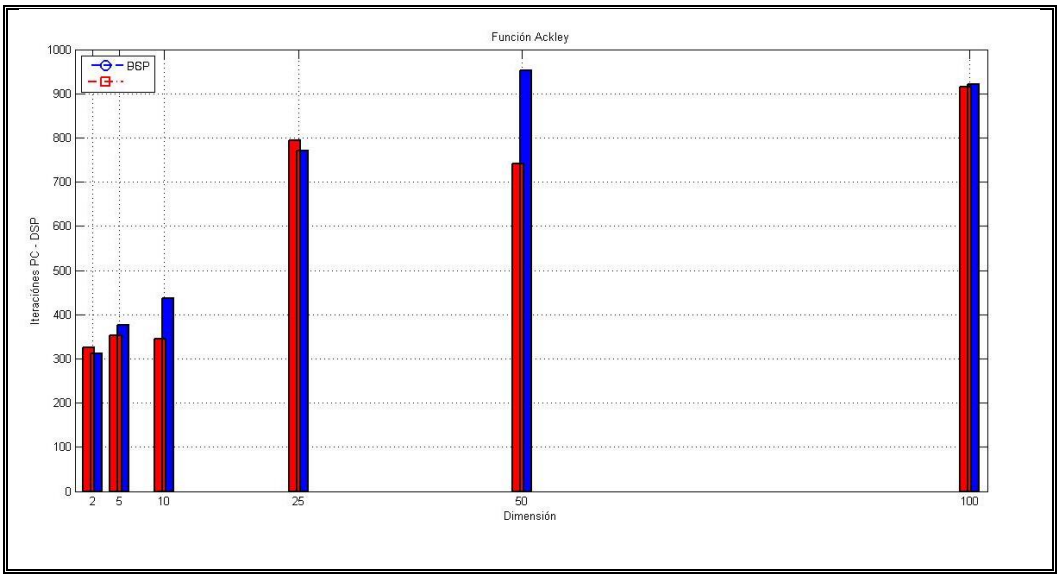
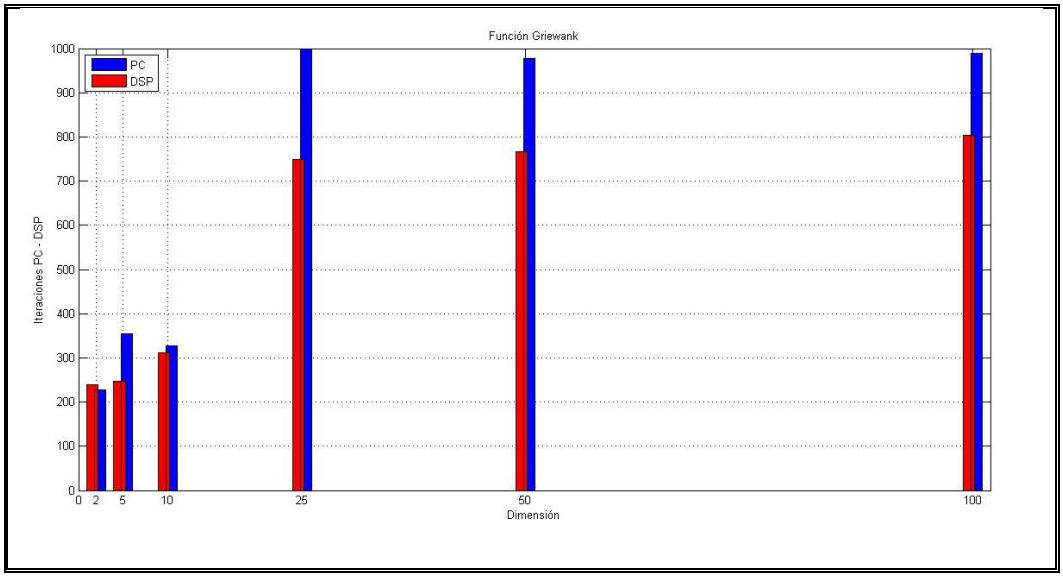


FIGURA 24. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN GRIEWANK.



Las diferencias de estas Figuras van a ser examinadas más adelante teniendo en cuenta algunas otras variables, que pueden entregar mayor y mejor información al respecto:

FIGURA 25. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN RASTRIGIN.

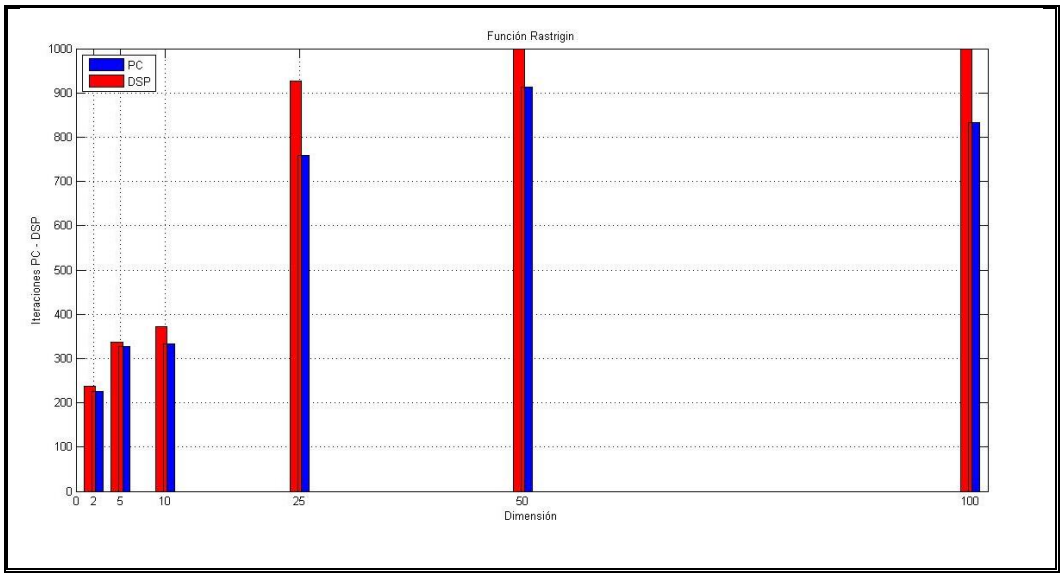


FIGURA 26. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN ROSENBROCK.

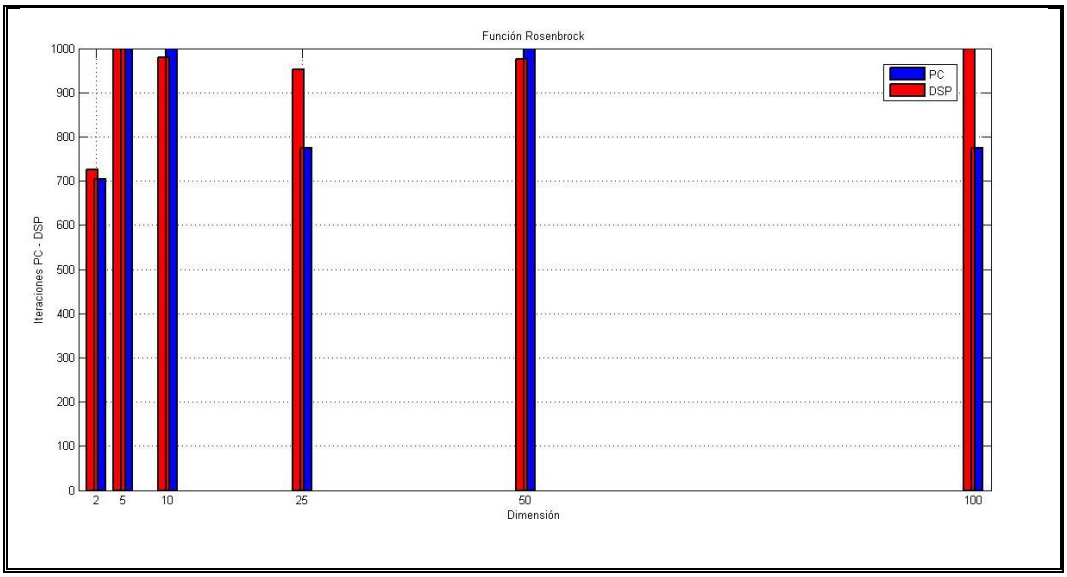


FIGURA 27. COMPARATIVO DE LAS ITERACIONES REALIZADAS POR EL PC Y EL DSP PARA LA FUNCIÓN SCHAFFER F6.

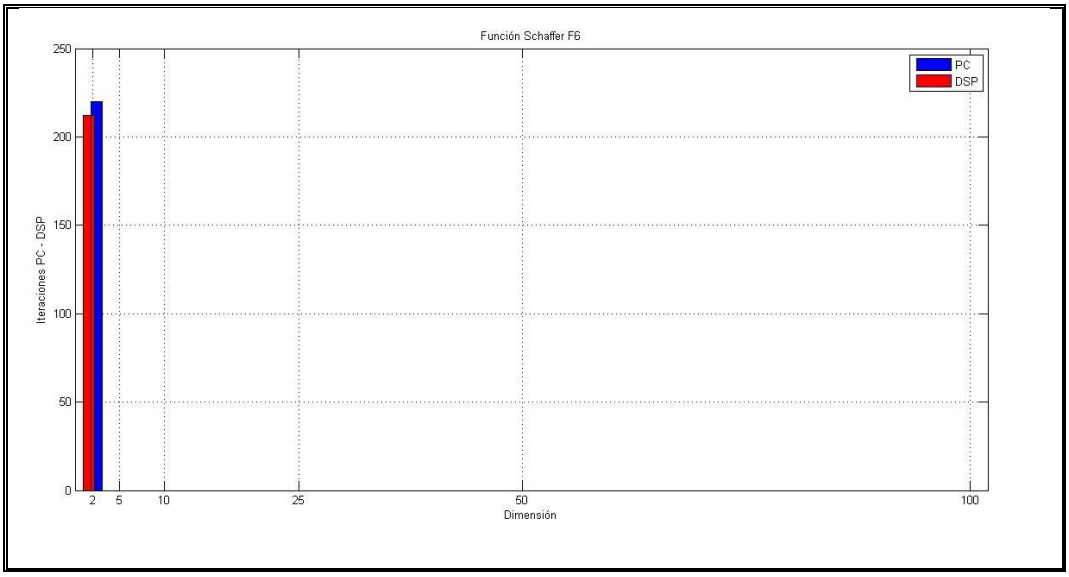
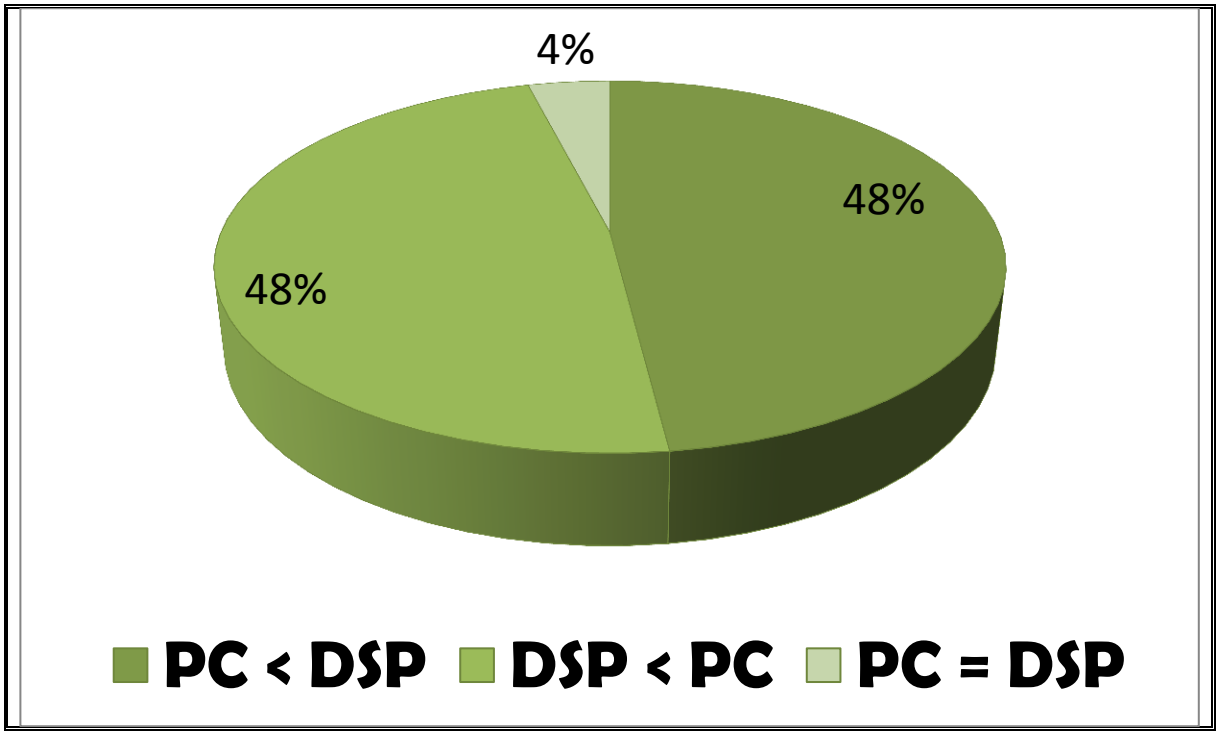


FIGURA 28. DIFERENCIA DE ITERACIONES REALIZADAS POR LOS SISTEMAS PSO-PC Y PSO-DSP.



En la Figura 28 se presenta los datos obtenidos para las iteraciones requeridas por los sistemas, teniendo en cuenta cuando fueron menores para el PC (48%), cuando fueron menores para el DSP (48%) y cuando utilizaron la misma cantidad de estas.

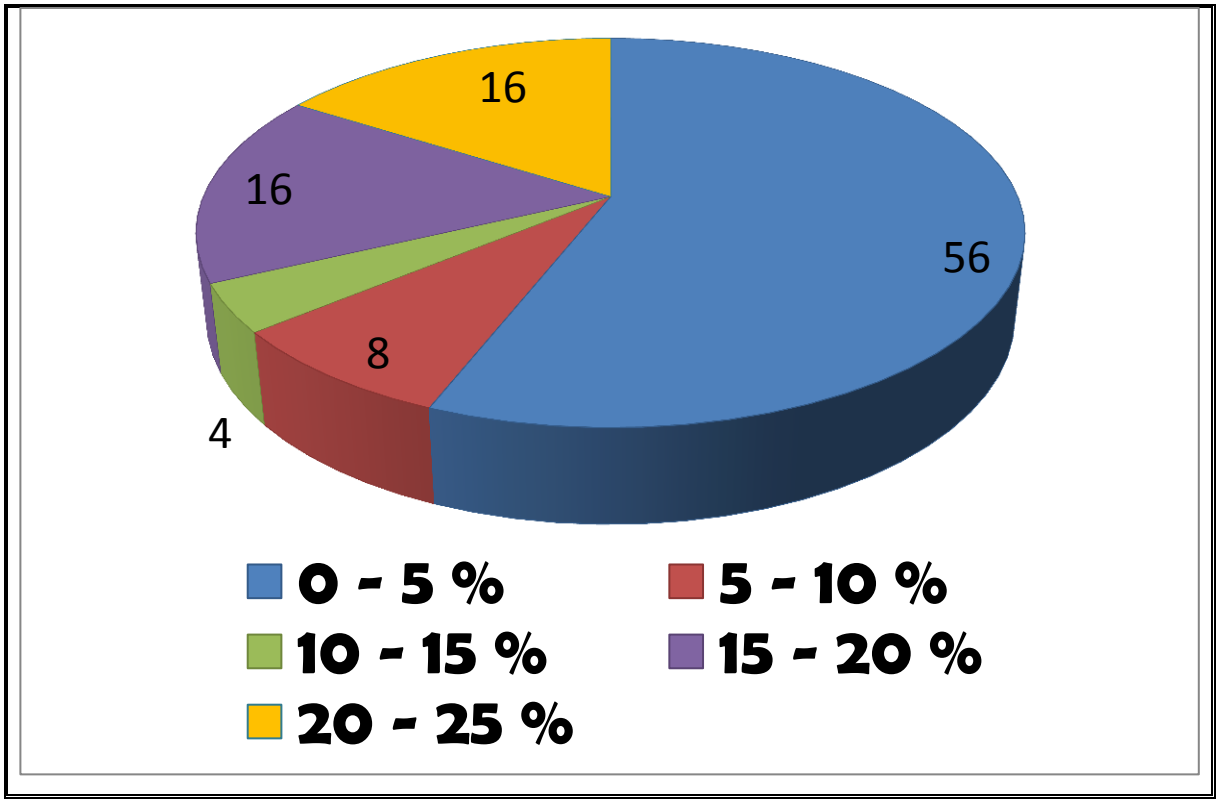
Tabla 8. Porcentaje de diferencia de error entre las iteraciones requeridas por el PC y el DSP con respecto a las máximas definidas (1000).

Función	Dimensiones	Porcentaje de diferencia de error PC - DSP
Ackley	2	1.50
	5	2.30
	10	9.20
	25	2.20
	50	21.1
	100	0.50
Griewank	2	1.20
	5	10.6
	10	1.60
	25	24.9
	50	21.1
	100	18.6
Rastrigin	2	1.10
	5	0.80
	10	4.00
	25	16.8
	50	8.60
	100	16.7
Rosenbrock	2	2.10
	5	0.00
	10	2.10
	25	17.8
	50	2.40
	100	22.5
Schaffer F6	2	0.80

La Tabla 8, se explica de la siguiente forma: Esta contiene la información de lo relativamente considerable o no que fueron las diferencias de iteraciones entre los

dos sistemas (PSO-PC y PSO-DSP), es decir, muestra el porcentaje de los casos, entre 25 en total, que utilizaron respectivamente, entre 0-5%, 10-15%, 15-20% y 20-25% de las iteraciones totales (1000):

FIGURA 29. PORCENTAJE DE LA DIFERENCIA DE ITERACIONES REALIZADAS POR LOS PROCESOS EN CADA UNA DE LAS FUNCIONES CON RESPECTO AL TOTAL DE ITERACIONES REALIZADAS (1000).



En cuanto a las desviaciones estándares que se obtuvieron de los procesos implementados, las cuales nos suministran un conocimiento aproximado de la dispersión de los datos, se analizaron las siguientes comparaciones:

- Para las iteraciones realizadas por cada uno de los procesos:

FIGURA 30. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN ACKLEY.

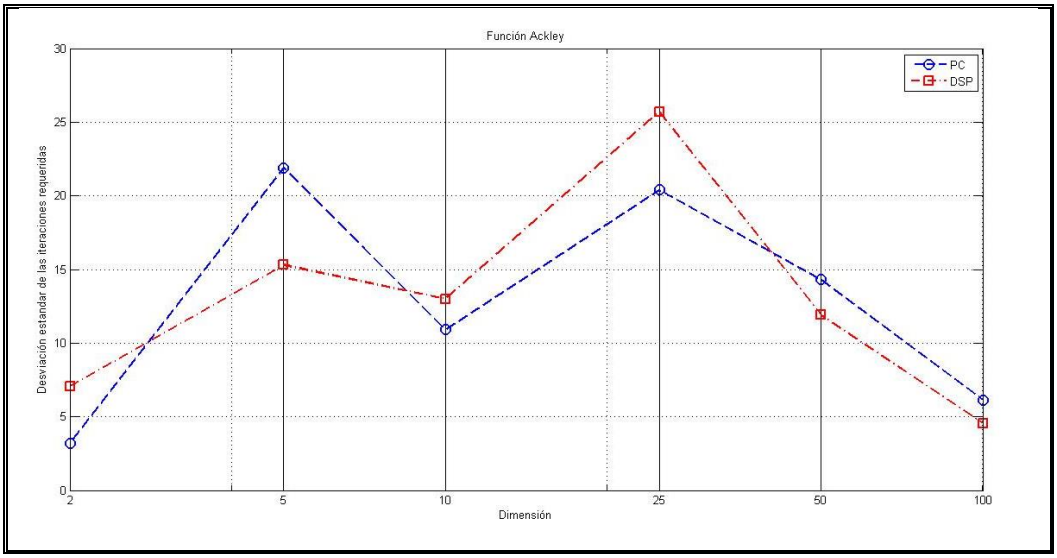


FIGURA 31. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN GRIEWANK.

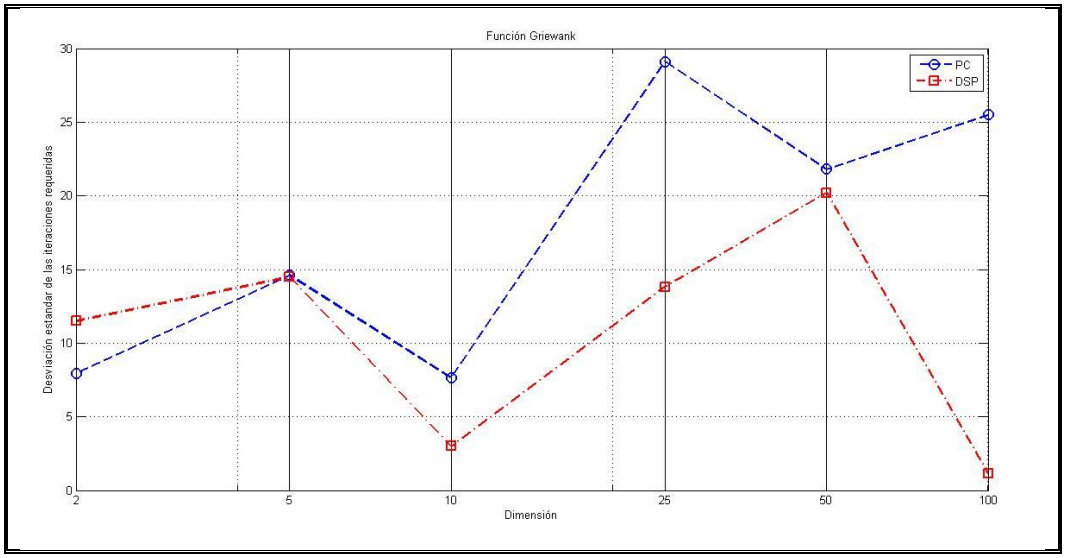


FIGURA 32. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN RASTRIGIN.

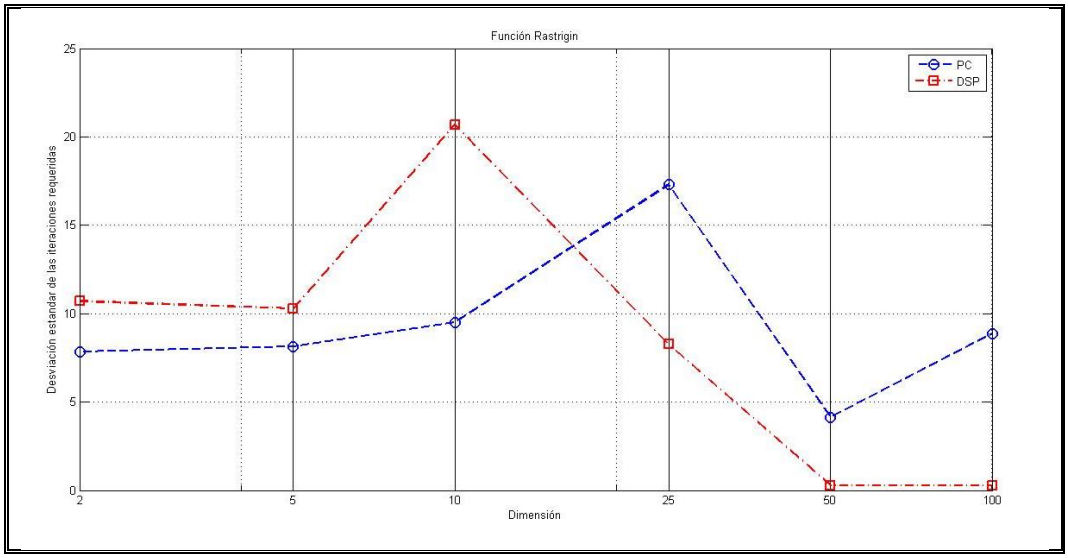
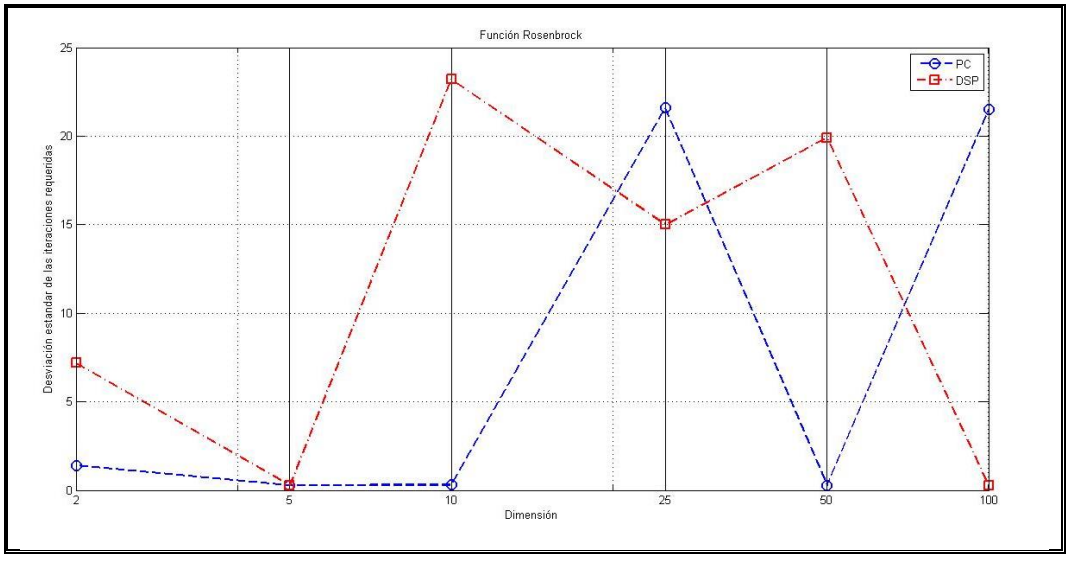


FIGURA 33. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LAS ITERACIONES ENTRE PC-DSP PARA LA FUNCIÓN ROSENBROCK.



- Para los valores mínimos de la encontrados por cada uno de los procesos:

FIGURA 34. COMPARATIVO ENTRE LAS DESVIACIONES DE LOS MÍNIMOS ENCONTRADOS ESTÁNDARES ENTRE PC-DSP PARA LA FUNCIÓN ACKLEY.

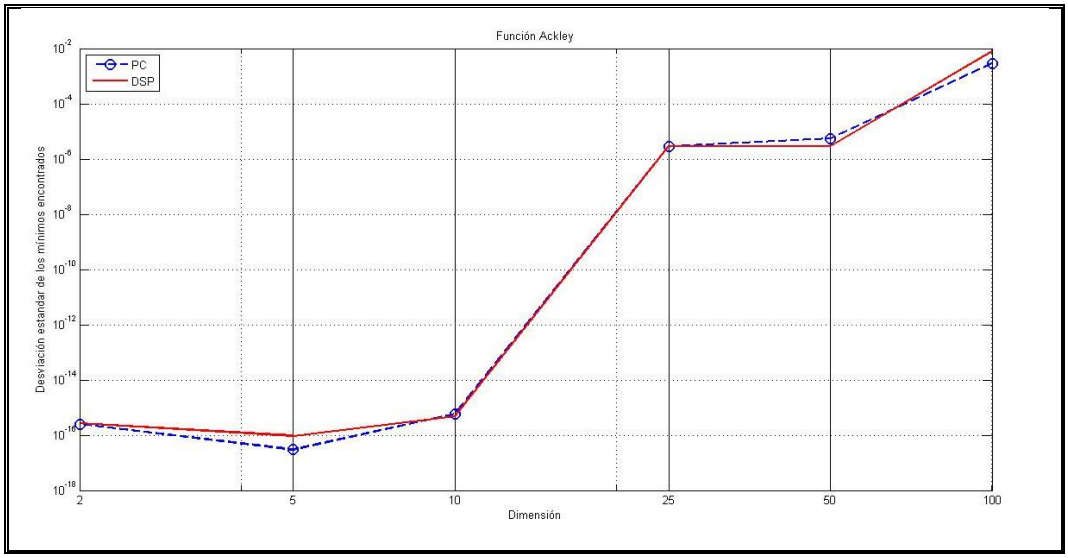


FIGURA 35. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS MÍNIMOS ENCONTRADOS ENTRE PC-DSP PARA LA FUNCIÓN GRIEWANK.

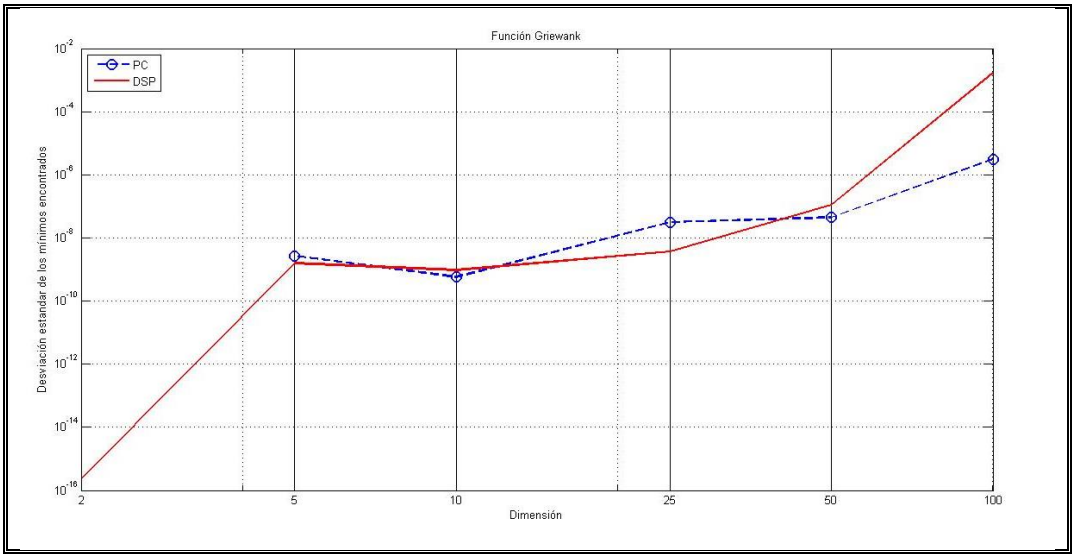


FIGURA 36. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS MÍNIMOS ENCONTRADOS ENTRE PC-DSP PARA LA FUNCIÓN RASTRIGIN.

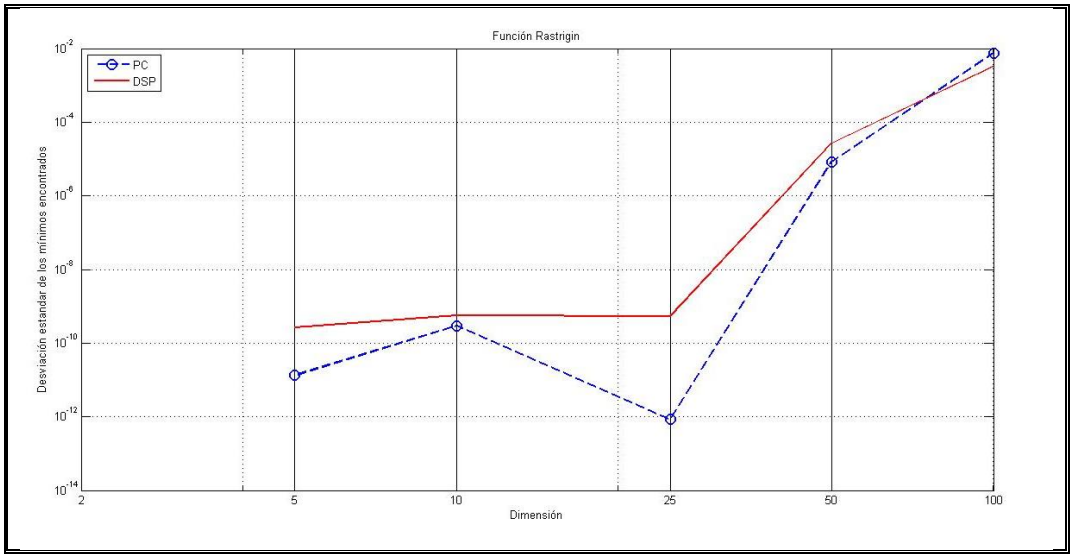
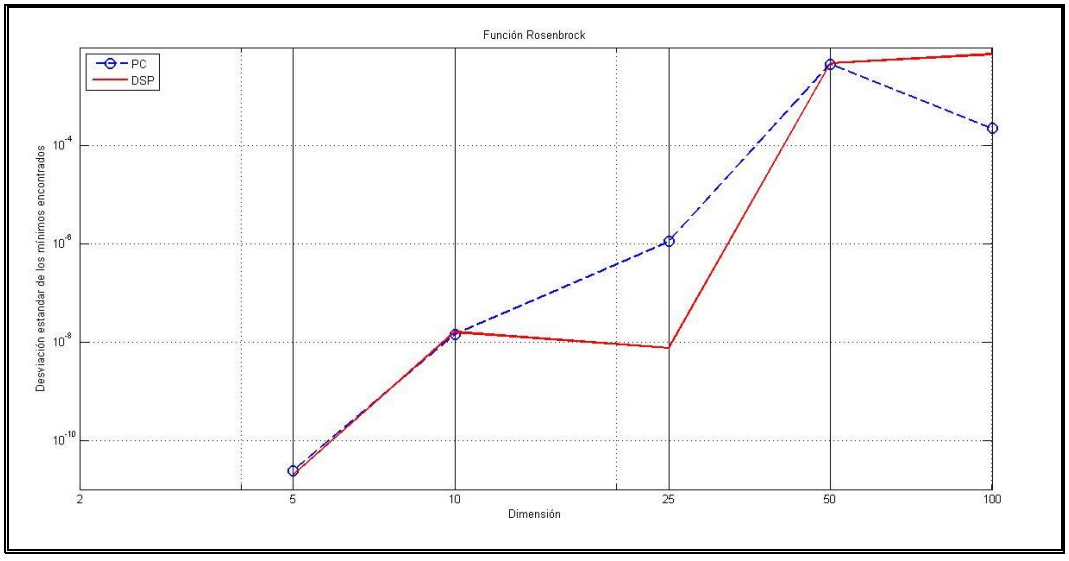


FIGURA 37. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS MÍNIMOS ENCONTRADOS ENTRE PC-DSP PARA LA FUNCIÓN ROSENBROCK.



- Para los tiempos de procesamiento alcanzados por cada uno de los procesos:

De entre las tres comparaciones para la desviación estándar, las del tiempo de procesamiento fueron aquellas que suministraron la información más relevante, ya que es evidentemente notoria la mínima dispersión de los datos en estos procesos en la totalidad de las funciones.

FIGURA 38. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN ACKLEY.

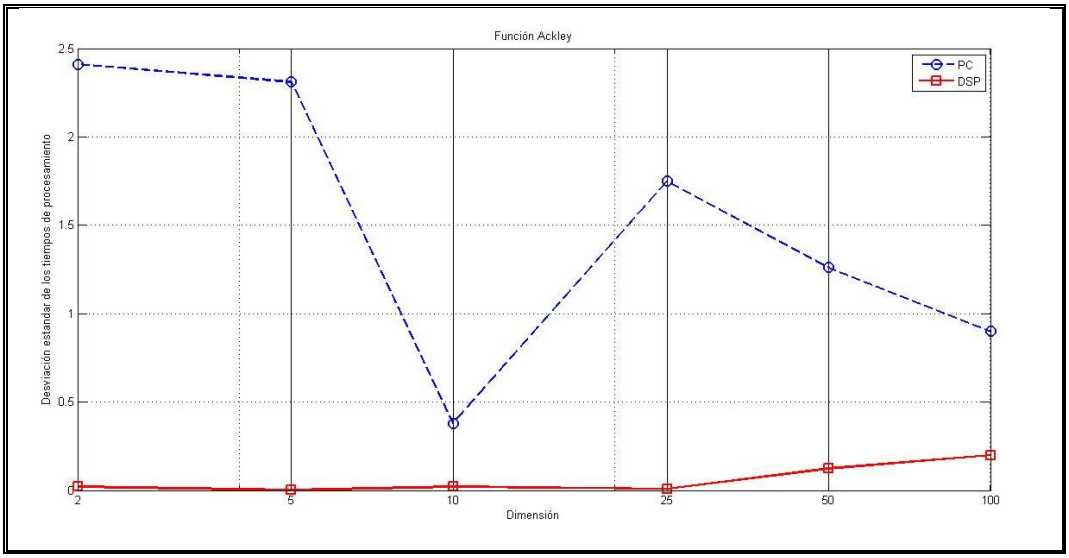


FIGURA 39. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN GRIEWANK.

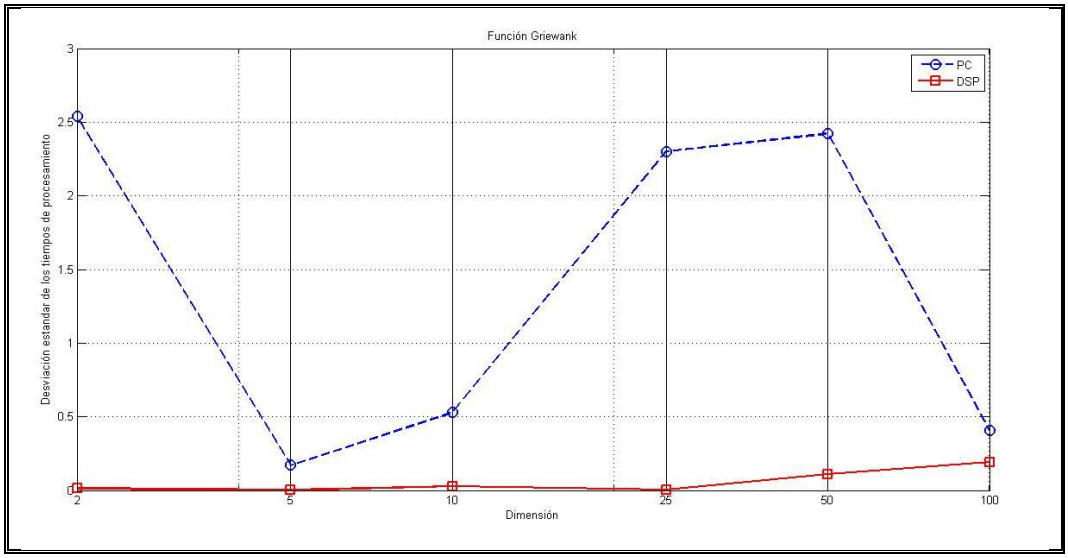


FIGURA 40. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN RASTRIGIN.

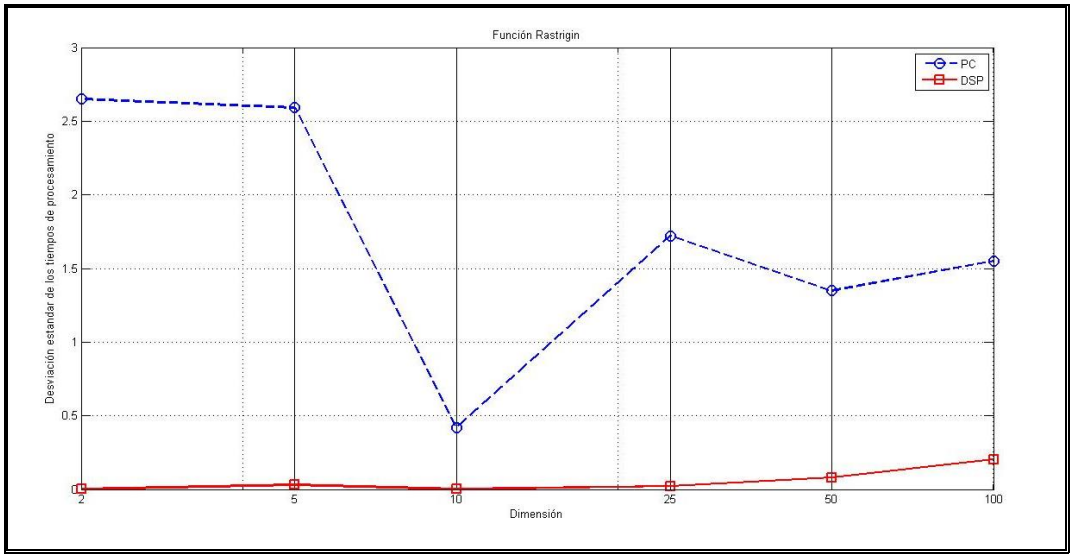


FIGURA 41. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN ROSENBROCK.

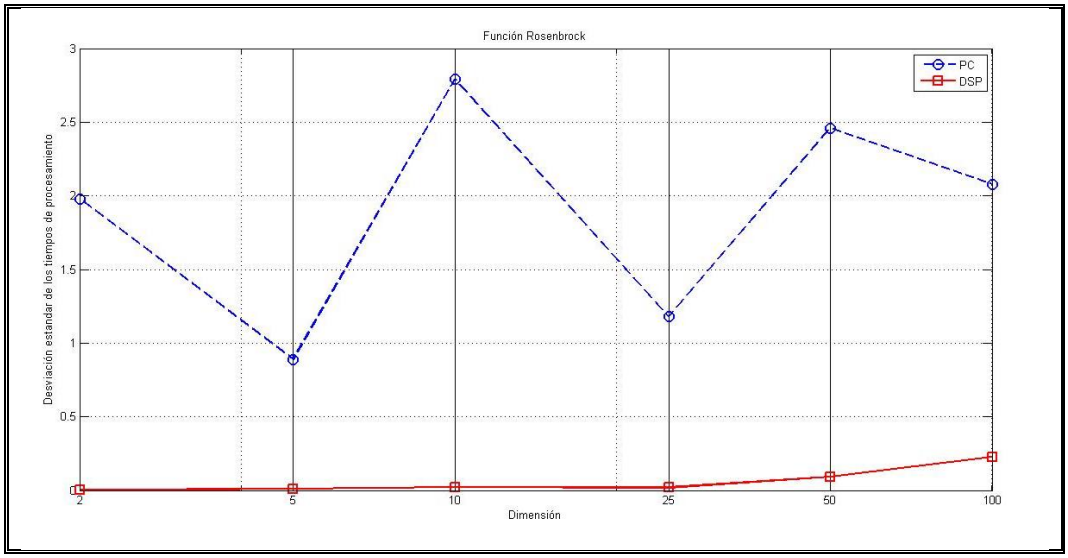


FIGURA 42. COMPARATIVO ENTRE LAS DESVIACIONES ESTÁNDARES DE LOS TIEMPOS DE PROCESAMIENTO ENTRE PC-DSP PARA LA FUNCIÓN SCHAFFER F6.

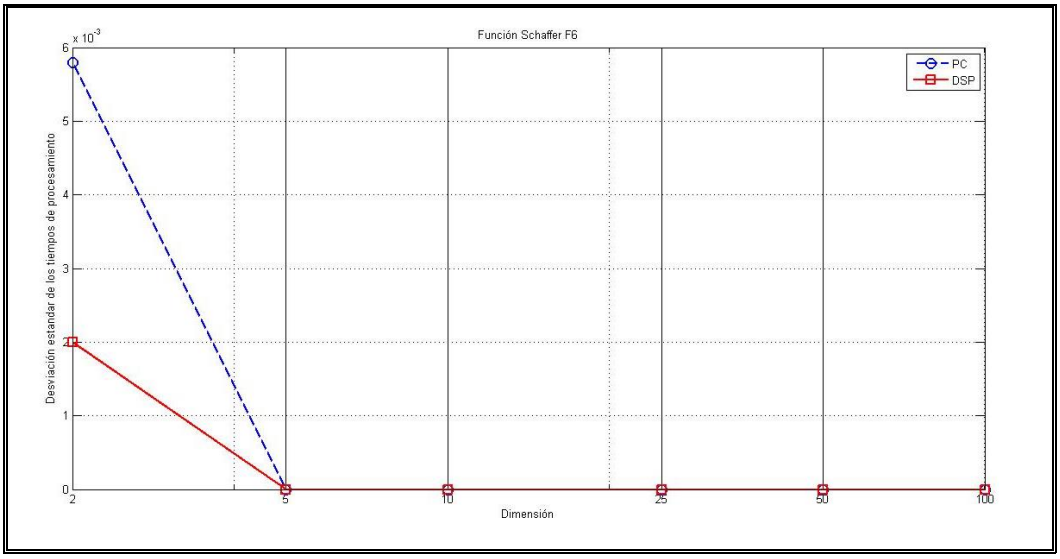
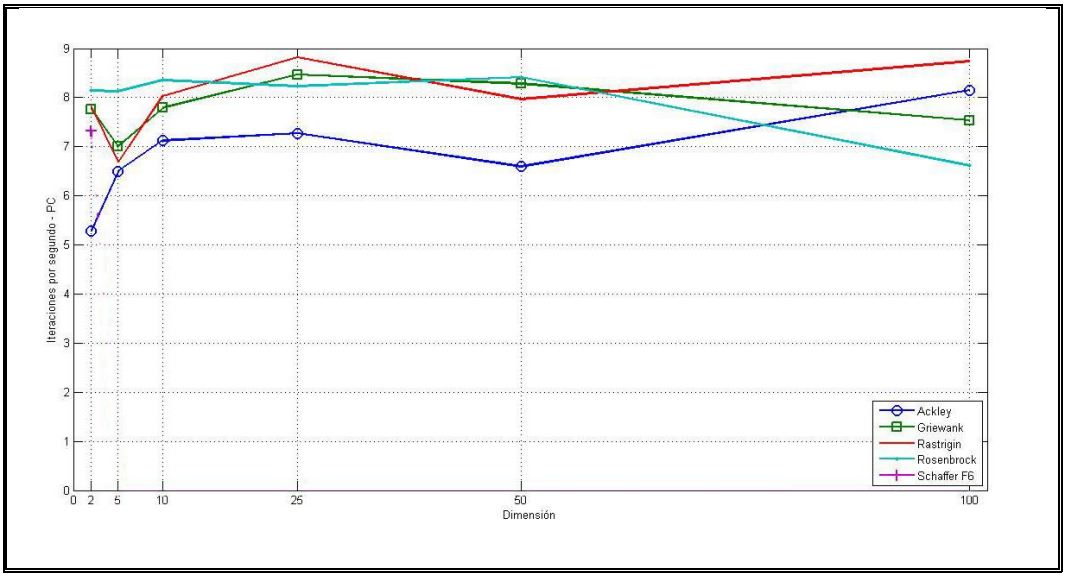
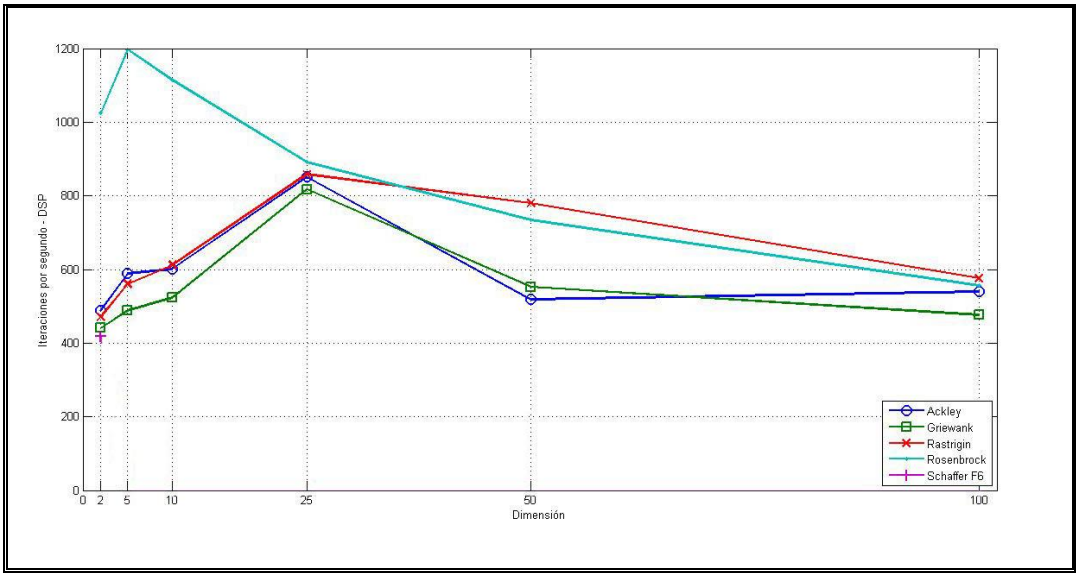


FIGURA 43. ITERACIONES POR SEGUNDO DEL PROCESO PSO-PC PARA TODAS LAS FUNCIONES.



Para las Figuras 43 y 44 se determinan las tendencias de las funciones aplicadas en cuanto a las iteraciones por segundo que realizaron tanto el PC como el DSP.

FIGURA 44. ITERACIONES POR SEGUNDO DEL PROCESO PSO-DSP PARA TODAS LAS FUNCIONES.



5 OBSERVACIONES Y CONCLUSIONES

1. Se produce una notable reducción del tiempo de procesamiento del algoritmo de optimización al ser implementado en el DSP (Al menos 50 veces más rápido que el PC utilizado). Es decir, el sistema conformado por el PSO cuando es ejecutado por medio del DSP, provee de mejores tiempos, lo cual convierte en eficaz esta estrategia.
2. La eficacia y eficiencia de la estrategia de implementación PSO-DSP es evidente, ya que el objetivo al que se dedica, se logra al menos cincuenta veces más rápido que cuando se usa un PC para su ejecución. Aun así, la precisión del valor de la función objetivo disminuye en algunos casos y en otros se supera levemente, lo cual es algo a tener en cuenta para futuras implementaciones. A pesar de lo anteriormente dicho, los resultados son considerablemente aceptables y no se puede descartar que para ciertas aplicaciones el desempeño del sistema en prueba (PSO-DSP) pueda ser utilizado y puedan obtenerse así excelentes resultados.
3. El DSP se presenta como una alternativa viable y efectiva cuando se requiere disminuir tiempos de cómputo en problemas de optimización.
4. Se hizo presente en las pruebas realizadas el conocido "*trade off*" o compromiso entre variables del sistema porque inicialmente se esperaba reducir los tiempos de procesamiento (Aspecto que definitivamente se logró) a la vez que se mejoraba la precisión de los valores de los mínimos encontrados (Lo que definitivamente no es concluyente).
5. El promedio de la relación entre el tiempo requerido por el sistema PSO-PC con respecto al del sistema PSO-DSP es de 89.07 con una desviación estándar de 25.77, lo cual indica una gran mejoría en este aspecto, como ya se ha analizado antes.
6. El promedio de diferencia de error entre los valores encontrados por el PSO-PC y el PSO-DSP es de 1.30e-3 con una desviación estándar de 6.60e-3.

7. No hay una tendencia definida para el número de iteraciones requeridas por cada una de las estrategias de implementación, en algunos casos el PSO-DSP consumió más iteraciones pero eso se debió a que el valor del mínimo encontrado mejoró con respecto a la otra alternativa (sólo en pocos casos). Y aunque el sistema PSO-DSP consumiera más iteraciones lo hacía en un tiempo mínimo.
8. En realidad se establecieron las diferencias entre los cálculos matemáticos o numéricos que realizan las dos plataformas (PC y DSP), dependerán entonces estos resultados de las librerías o de las reglas de cálculo de cada una de ellas más no del algoritmo en análisis. A pesar que las plataformas puedan brindar más precisión, el código que se realice del PSO será finalmente el que determine la precisión general.
9. El promedio de iteraciones por segundo del PSO-DSP es de 667.1 con una desviación estándar de 217.9, lo cual reconfirma la hipótesis inicial de que el DSP presentaría mejor desempeño en el aspecto temporal.
10. Aunque de los 25 casos analizados en los datos obtenidos tanto del PC como del DSP, solo 4 de dichos casos mostraron una diferencia de las iteraciones que esta entre el 20% y el 25% de las iteraciones totales, las mayoría de las diferencias se encuentran por debajo del 20%, lo cual indica que para las dos plataformas no es relevante la mencionada diferencia, concluyendo que los sistemas poseen un desempeño similar.
11. El promedio de iteraciones por segundo del PSO-PC es de 7.64 con una desviación estándar de 8.42, en contraste con el promedio de iteraciones por segundo del PSO-PC que equivale a 667.1 con una desviación estándar de 217.9.
12. Se presentó también una gran desviación estándar en cuanto a las iteraciones requeridas por los dos sistemas, debido principalmente a las reglas de generación de números aleatorios que poseen ambas plataformas.

6 RECOMENDACIONES

- Para el caso de utilizar la estrategia de implementación del PSO-DSP se recomienda hacer uso de la actualización síncrona, aprovechando así la programación paralela y la distribución interna de las unidades aritmético-lógicas del DSP.
- Para mejorar la precisión del sistema PSO-DSP se podría definir una precisión distinta en la bandera de error permitido (Que representa la percepción de los cambios entre los valores óptimos) del programa, junto a la bandera de saturación (La cual consiste en que pasadas un número determinado de iteraciones sin mejorar el valor del optimo, suspenda el procesamiento), de la misma forma aumentar el número de iteraciones máximas a realizar.
- Realizar las mismas pruebas en un DSP de última generación, ya que estos poseen muchas más y mejores características en cuanto a memoria, frecuencia de procesamiento y número de núcleos se refiere, solo por mencionar algunas.
- Queda a disposición del usuario la implementación de cualquiera de las versiones de PSO que intensifique alguna característica en particular, por ejemplo, la de la precisión.
- Realizar un estudio con mayor proyección sobre la escalabilidad de las variables inmersas en el procesamiento del PSO, completando así la caracterización del sistema PSO-DSP.

7 REFERENCIAS BIBLIOGRAFÍCAS

[1] F. Glover and G. A. Kochenberger, *HANDBOOK OF METAHEURISTICS*. 2003, pp. 1–570.

[2] M. A. Muñoz, J. A. López, and E. F. Caicedo, “Inteligencia de enjambres: Sociedades para la solución de problemas (Una revisión) - Swarm intelligence: Problem-solving societies (A review),” *Ing. e Investig.*, vol. 28, no. 2, pp. 119–130, 2008.

[3] E. S. Gopi, *Algorithms Collections for Digital Signal Processing Applications using MatLab*. 2007, pp. 1–199.

[4] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.

[5] D. Bratton and D. Bratton, “Defining a Standard for Particle Swarm Optimization,” *Symp. A Q. J. Mod. Foreign Lit.*, no. Sis, pp. 120–127, 2007.

[6] J. Aching and D. Rojas, “Reconocimiento Biométrico de Huellas Dactilares y su implementación en DSP,” 2005.

[7] J. Ramón and P. López, “CONTRIBUCIÓN A LOS MÉTODOS DE OPTIMIZACIÓN BASADOS EN PROCESOS NATURALES Y SU APLICACIÓN A LA MEDIDA DE ANTENAS EN CAMPO PRÓXIMO,” 2005.

[8] I. C. Trelea, “The particle swarm optimization algorithm: Convergence analysis and parameter selection,” *Comput. Sci. Web - Inf. Process. Lett.*, vol. 85, pp. 317–325, 2003.

[9] M. Clerc, “The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization,” *Proc. 1999 Congr. Evol. Comput. CEC 1999*, vol. 3, pp. 1951–1957, 1999.

[10] J. García, “Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos,” 2006.

[11] J. Salazar, “Procesadores digitales de señal (DSP) - Arquitecturas y criterios de selección.” pp. 46–57, 2000.

[12] E. Lai, *Practical Digital Signal Processing for Engineers and Technicians*. 2003, pp. 1–299.

[13] M. Eslami, H. Shareef, M. Khajehzadeh, and A. Mohamed, “A Survey of the State of the Art in Particle Swarm Optimization,” *Res. J. Appl. Sci. Eng. an Technol.*, no. January, pp. 1180–1197, 2012.

[14] Y. Shi and R. C. Eberhart, “Empirical Study of Particle Swarm Optimization,” *Proc. 1999 Congr. Evol. Comput.*, pp. 1945–1950, 1999.

[15] R. Repas, “DIGITAL SIGNAL PROCESSORS ‘ THINK ’ ANALOG BUT WORK DIGITALLY,” *Mach. Des.*, vol. 77, no. 13, p. 92, 2005.

[16] G. Frantz and L. Brantingham, “Signal Core - A short history of the digital signal processor,” *IEEE Solid-State Circuits Mag.*, vol. 4, no. June, pp. 16–20, 2012.

[17] freescale Semiconductor, “56F8357/56F8157.” pp. 1–177, 2007.

[18] Texas Instruments, “TMS320C6711, TMS320C6711B, TMS320C6711C FLOATING-POINT DIGITAL SIGNAL PROCESSORS,” no. February 1999. pp. 1–125, 2003.

[19] Texas Instruments, “TMS320C6000 Programmer ’s Guide,” no. July. pp. 1–441, 2011.

[20] Texas Instruments, “TMS320C6000 Code Composer Studio Tutorial,” no. February. pp. 1–126, 2000.

[21] D. Bell, “How to Begin Development with the TMS320C6711 DSP,” 1999.

[22] MathWorks, “MATLAB Link for Code Composer Studio Development Tools Release Notes,” 2000.

[23] R. C. Eberhart and Y. Shi, “Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization,” *Congr. Evol. Comput. 2000*, no. 7, pp. 84–88, 2000.

[24] A. J. Carlisle, “APPLYING THE PARTICLE SWARM OPTIMIZER TO NON-STATIONARY ENVIROMENTS,” 2002.

[25] M. Molga and C. Smutnicki, “Test functions for optimization needs.” pp. 1–43, 2005.

[26] D. Bratton and J. Kennedy, “Defining a Standard for Particle Swarm Optimization,” no. Sis, pp. 120–127, 2007.

BIBLIOGRAFÍA

Aching, J., & Rojas, D. (2005). *Reconocimiento Biométrico de Huellas Dactilares y su implementación en DSP*.

Bell, D. (1999). *How to Begin Development with the TMS320C6711 DSP*. *Application Report SPRA522* (pp. 1–11).

Bratton, D., & Bratton, D. (2007). Defining a Standard for Particle Swarm Optimization. *Symposium A Quarterly Journal In Modern Foreign Literatures*, (Sis), 120–127.

Bratton, D., & Kennedy, J. (2007). Defining a Standard for Particle Swarm Optimization, (Sis), 120–127.

Carlisle, A. J. (2002). *APPLYING THE PARTICLE SWARM OPTIMIZER TO NON-STATIONARY ENVIROMENTS*.

Clerc, M. (1999). The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, *3*, 1951–1957. doi:10.1109/CEC.1999.785513

Eberhart, R. C., & Shi, Y. (2000). Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. *The Congress on Evolutionary Computation 2000*, (7), 84–88. doi:10.1109/CEC.2000.870279

Eslami, M., Shareef, H., Khajehzadeh, M., & Mohamed, A. (2012). A Survey of the State of the Art in Particle Swarm Optimization. *Research Journal of Applied Sciences, Engineering an Technology*, (January), 1180–1197.

Frantz, G., & Brantingham, L. (2012). Signal Core - A short history of the digital signal processor. *IEEE Solid-State Circuits Magazine*, *4*(June), 16–20. doi:10.1109/MSSC.2012.2193074

freescale Semiconductor. (2007). 56F8357/56F8157.

García, J. (2006). *Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos*.

Glover, F., & Kochenberger, G. A. (2003). *HANDBOOK OF METAHEURISTICS* (pp. 1–570).

Gopi, E. S. (2007). *Algorithms Collections for Digital Signal Processing Applications using MatLab* (pp. 1–199).

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, *4*, 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968

Lai, E. (2003). *Practical Digital Signal Processing for Engineers and Technicians* (pp. 1–299).

MathWorks. (2000). *MATLAB Link for Code Composer Studio Development Tools Release Notes* (pp. 1–36).

Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs.

Muñoz, M. A., López, J. A., & Caicedo, E. F. (2008). Inteligencia de enjambres: Sociedades para la solución de problemas (Una revisión) - Swarm intelligence: Problem-solving societies (A review). *Ingeniería E Investigación*, *28*(2), 119–130. Retrieved from http://www.doaj.org/doaj?func=abstract&id=1115302

Ramón, J., & López, P. (2005). CONTRIBUCIÓN A LOS MÉTODOS DE OPTIMIZACIÓN BASADOS EN PROCESOS NATURALES Y SU APLICACIÓN A LA MEDIDA DE ANTENAS EN CAMPO PRÓXIMO.

Repas, R. (2005). DIGITAL SIGNAL PROCESSORS “ THINK ” ANALOG BUT WORK DIGITALLY. *Machine Design*, *77*(13), 92.

Salazar, J. (2000). Procesadores digitales de señal (DSP) - Arquitecturas y criterios de selección.

Shi, Y., & Eberhart, R. C. (1999). Empirical Study of Particle Swarm Optimization. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, 1945–1950. doi:10.1109/CEC.1999.785511

Texas Instruments. (2000). TMS320C6000 Code Composer Studio Tutorial.

Texas Instruments. (2003). TMS320C6711, TMS320C6711B, TMS320C6711C FLOATING-POINT DIGITAL SIGNAL PROCESSORS.

Texas Instruments. (2011). TMS320C6000 Programmer ’s Guide.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Computer Science Web - Information Processing Letters*, *85*, 317–325.

ANEXOS

Anexo A: Procesamiento Función Ackley.

A continuación se presentan algunas de las figuras en las cuales se puede observar el procesamiento de PSO en el DSP para cada una de las funciones en cada una de las dimensiones analizadas:

FIGURA 45. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (2 DIMENSIONES)

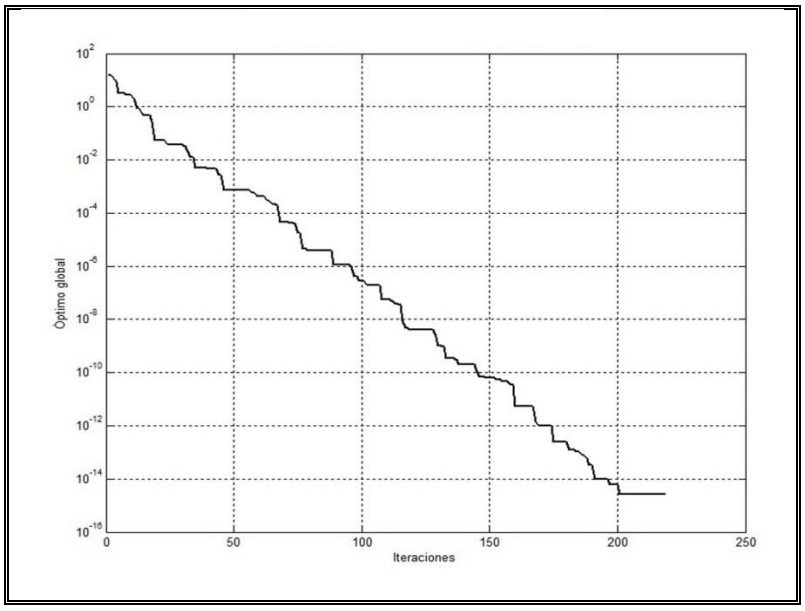


FIGURA 46. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (5 DIMENSIONES)

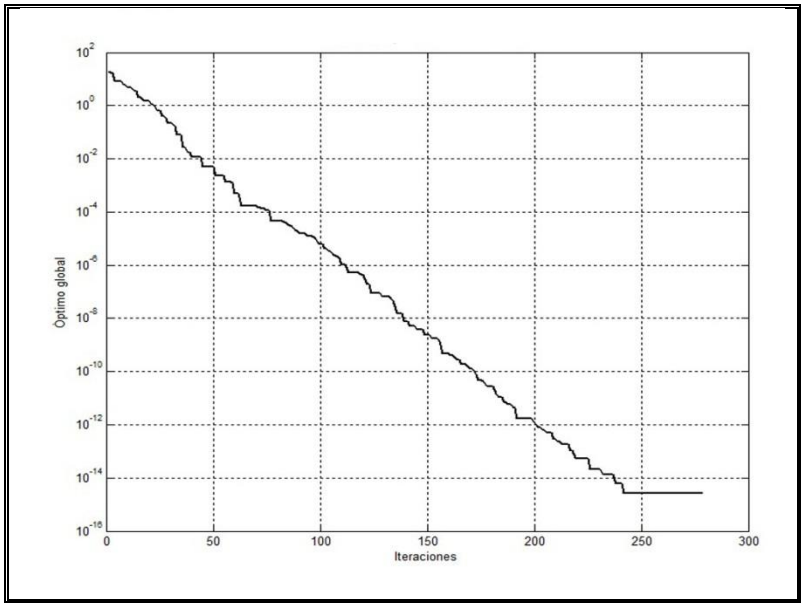


FIGURA 47. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (10 DIMENSIONES)

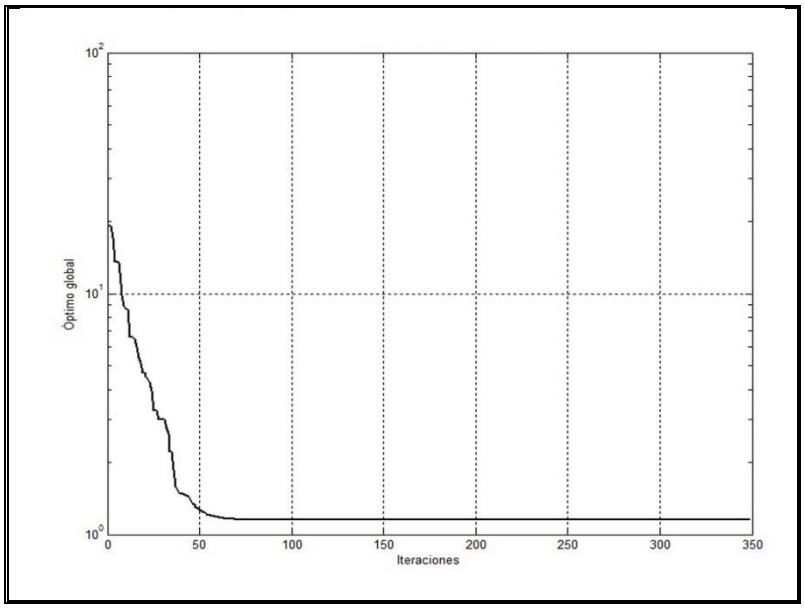


FIGURA 48. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (25 DIMENSIONES)

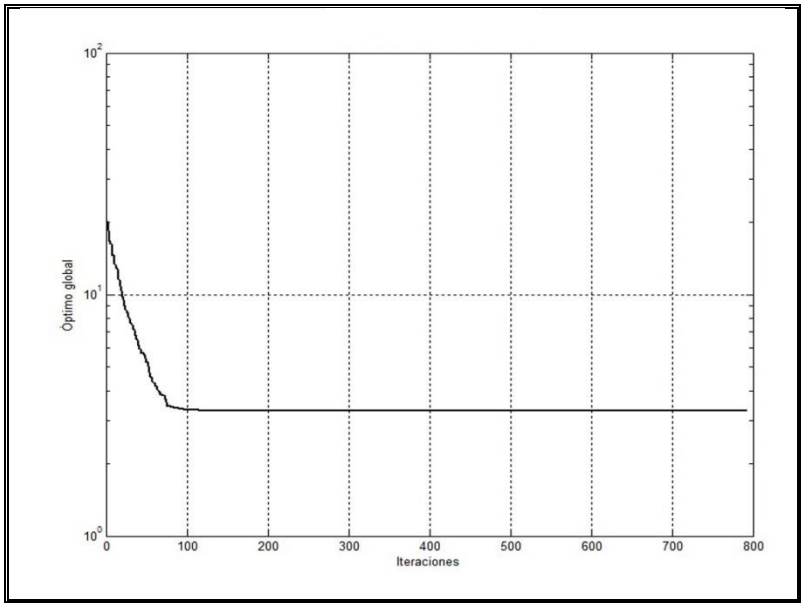


FIGURA 49. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (50 DIMENSIONES)

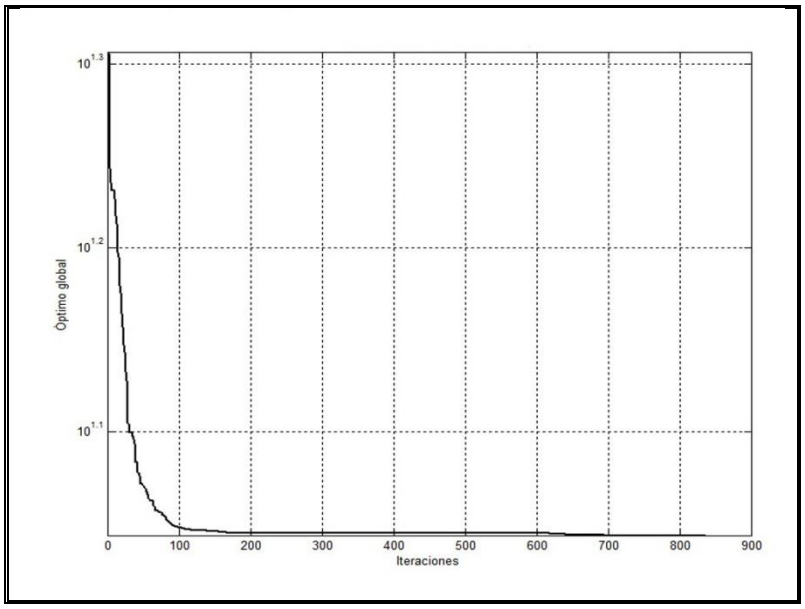
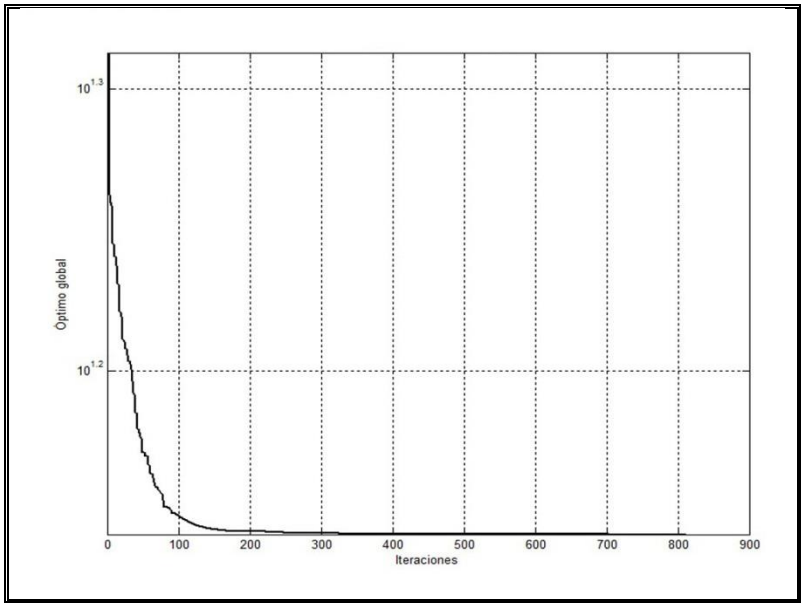


FIGURA 50. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ACKLEY. (100 DIMENSIONES)



Anexo B: Procesamiento Función Griewank.

FIGURA 51. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (2 DIMENSIONES)

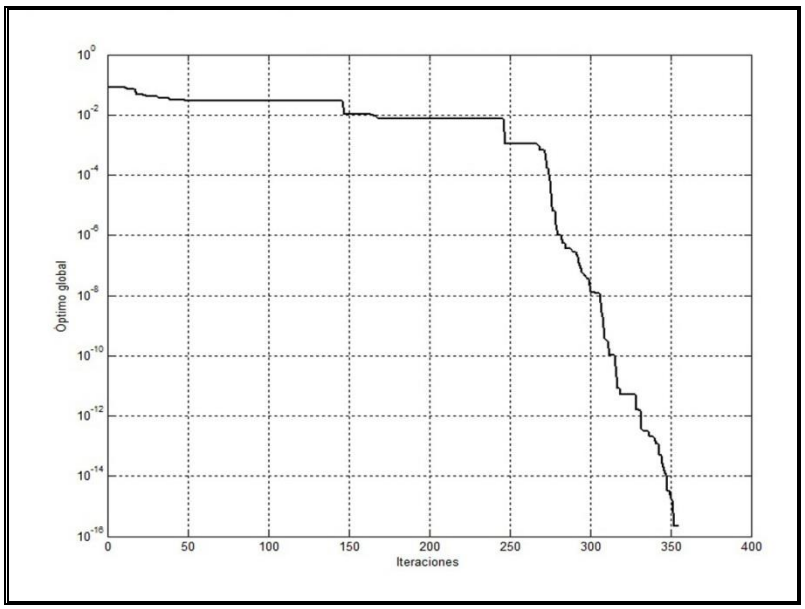


FIGURA 52. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (5 DIMENSIONES)

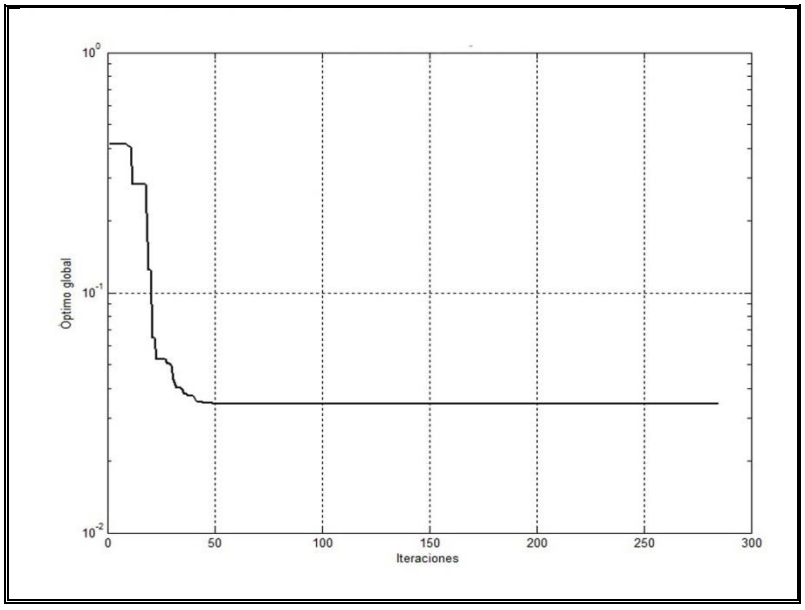


FIGURA 53. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (10 DIMENSIONES)

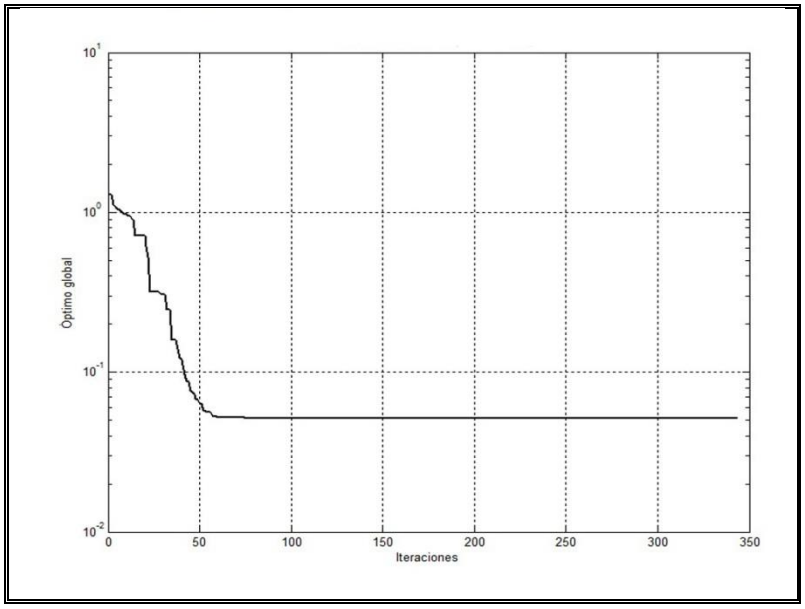


FIGURA 54. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (25 DIMENSIONES)

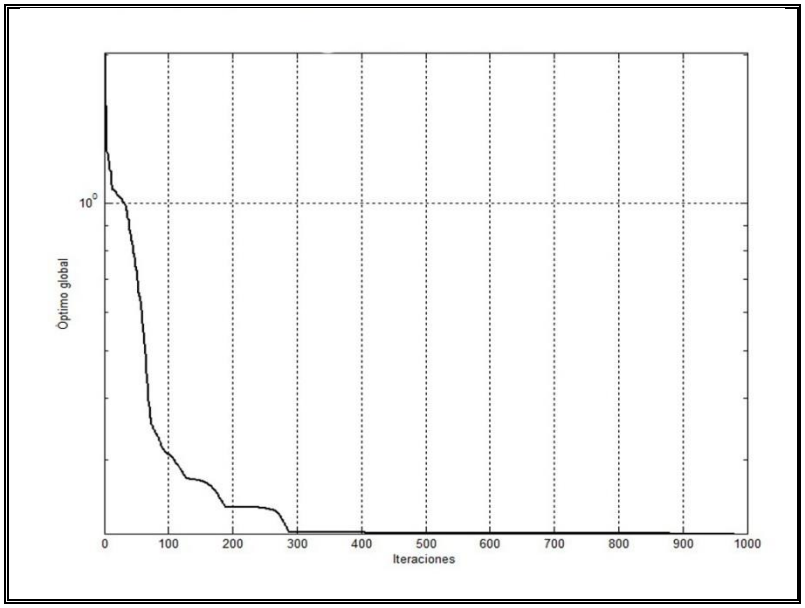


FIGURA 55. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (50 DIMENSIONES)

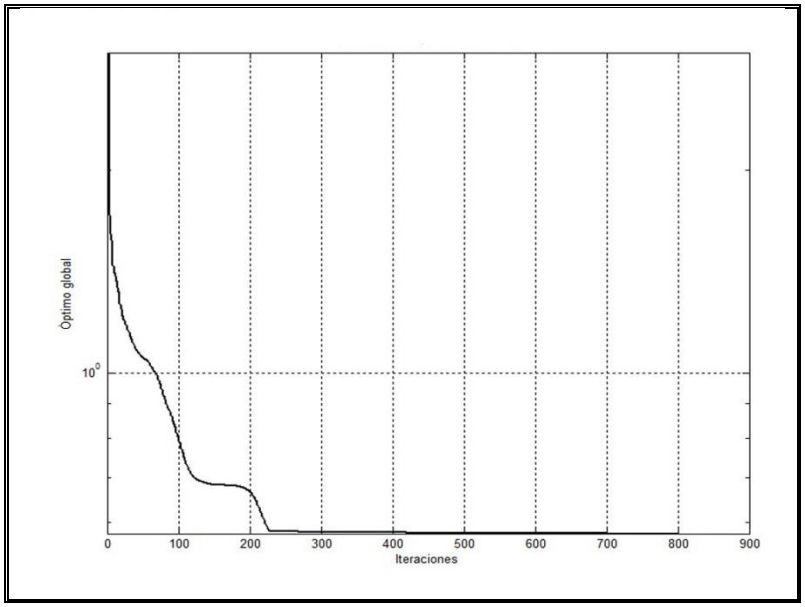
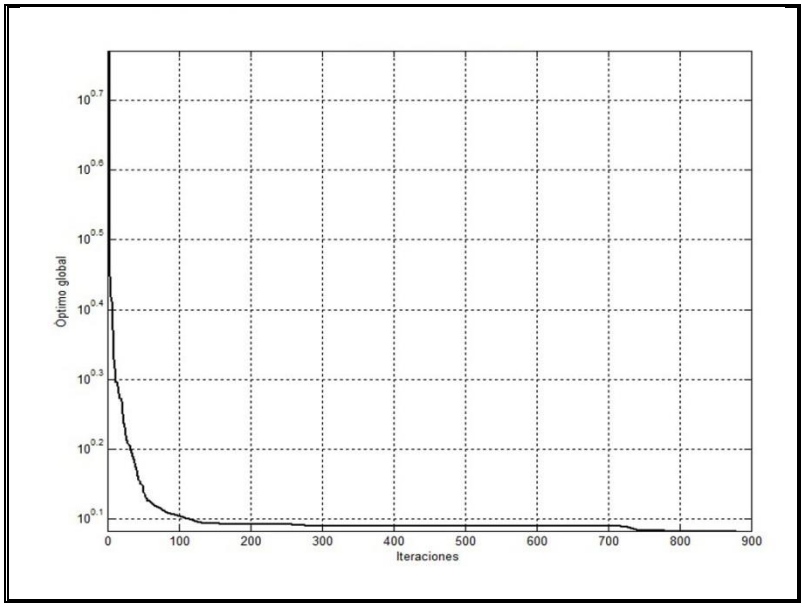


FIGURA 56. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN GRIEWANK. (100 DIMENSIONES)



Anexo C: Procesamiento Función Rastrigin.

FIGURA 57. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (2 DIMENSIONES)

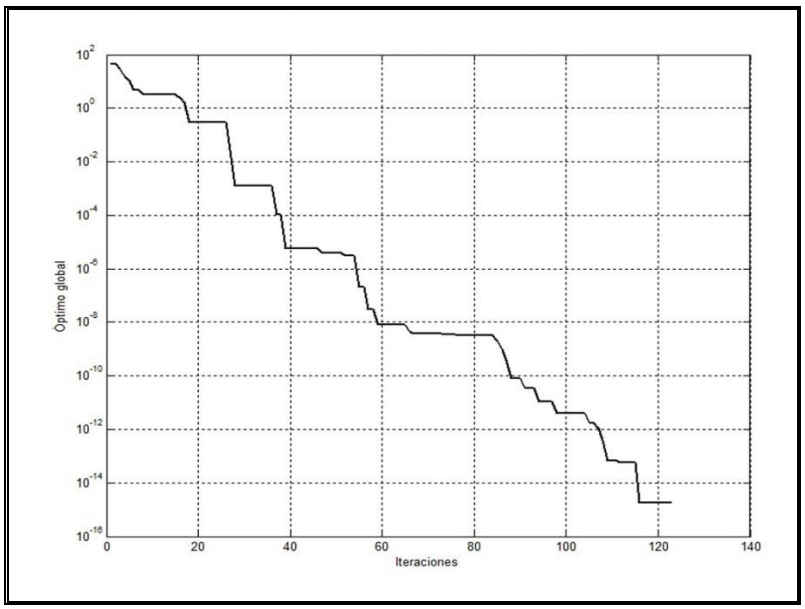


FIGURA 58. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (5 DIMENSIONES)

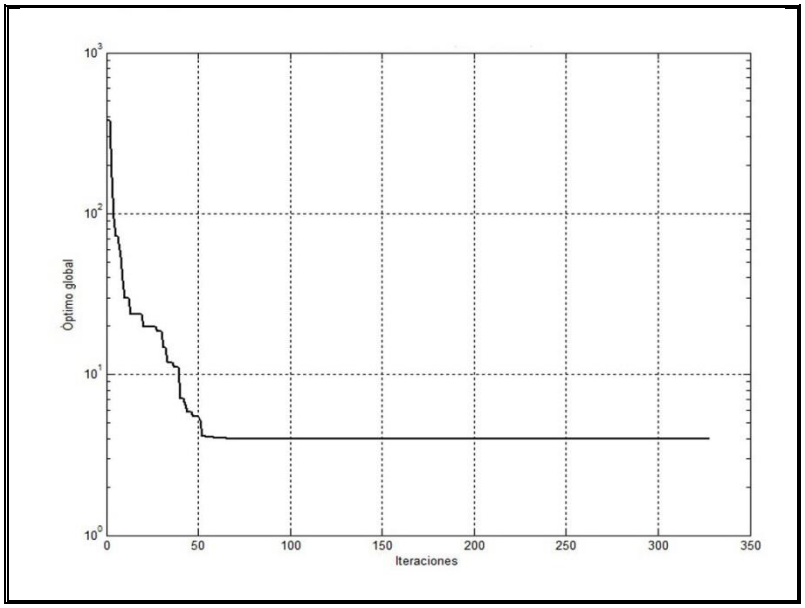


FIGURA 59. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (10 DIMENSIONES)

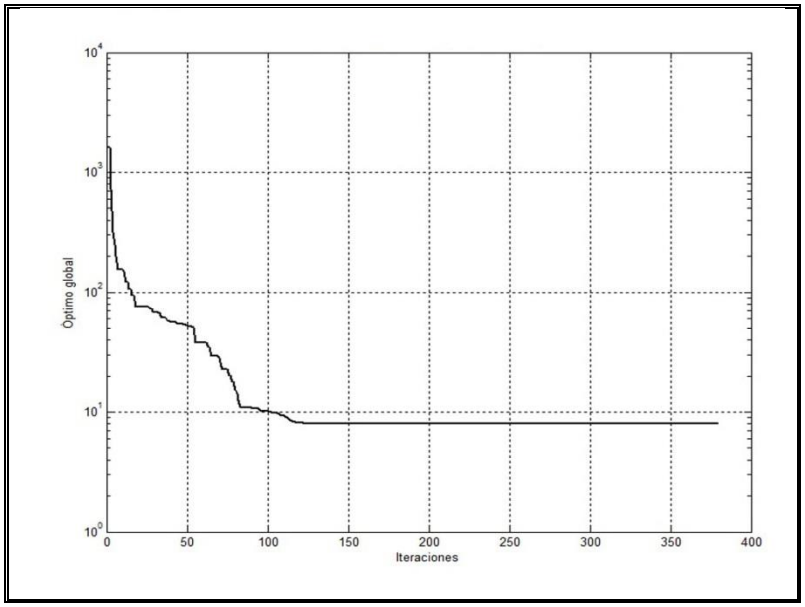


FIGURA 60. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (25 DIMENSIONES)

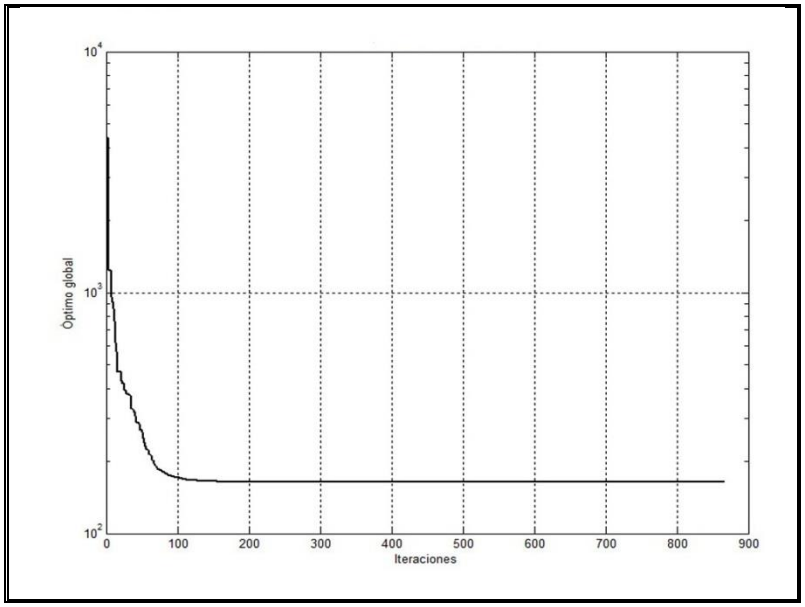


FIGURA 61. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (50 DIMENSIONES)

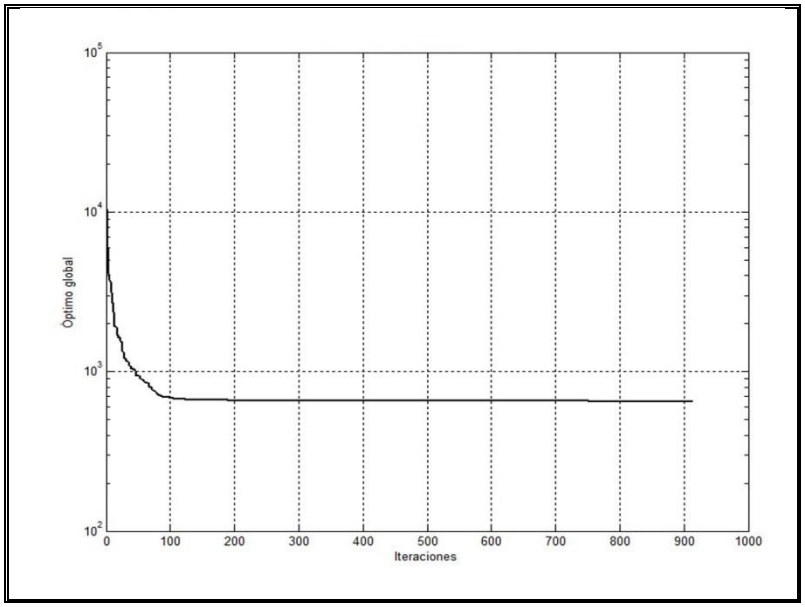
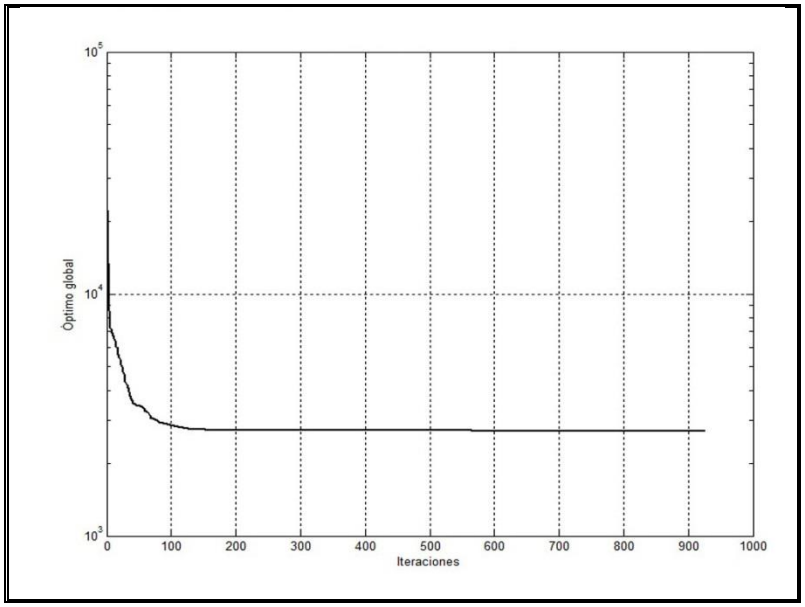


FIGURA 62. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN RASTRIGIN. (100 DIMENSIONES)



Anexo D: Procesamiento Función Rosenbrock.

FIGURA 63. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (2 DIMENSIONES)

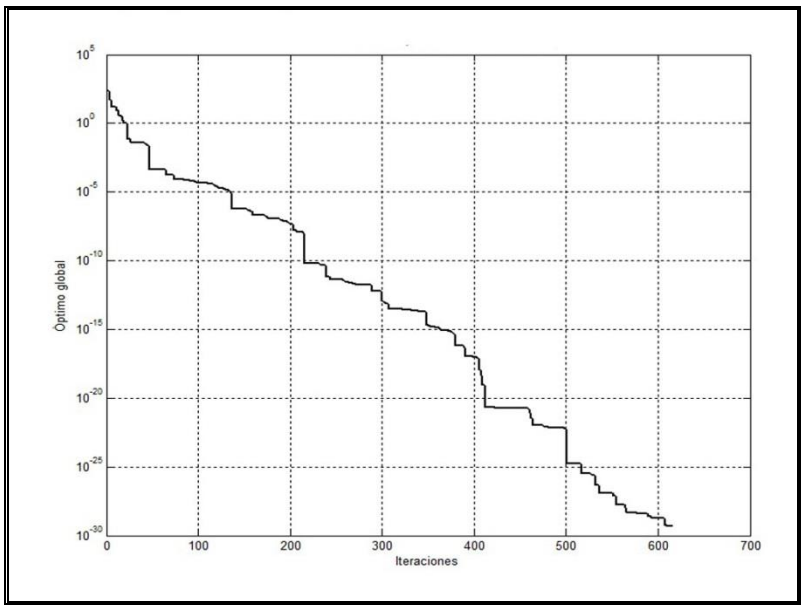


FIGURA 64. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (5 DIMENSIONES)

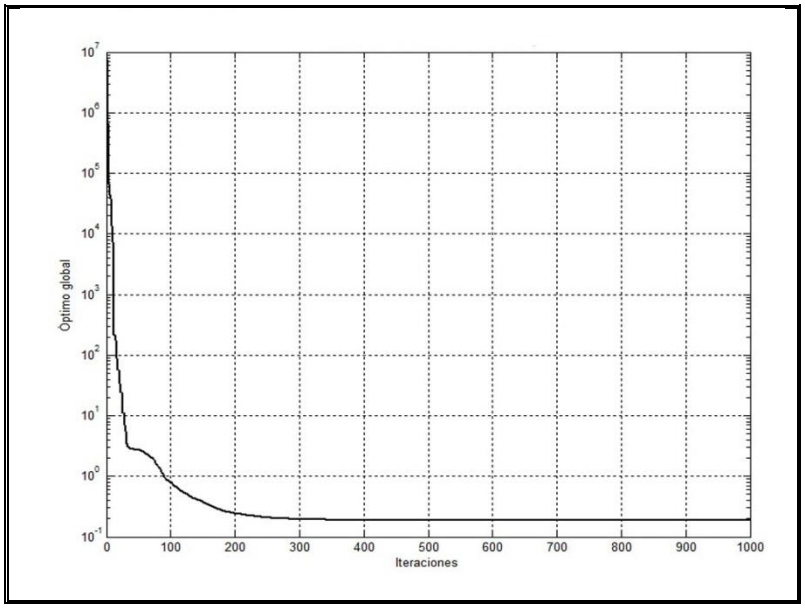


FIGURA 65. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (10 DIMENSIONES)

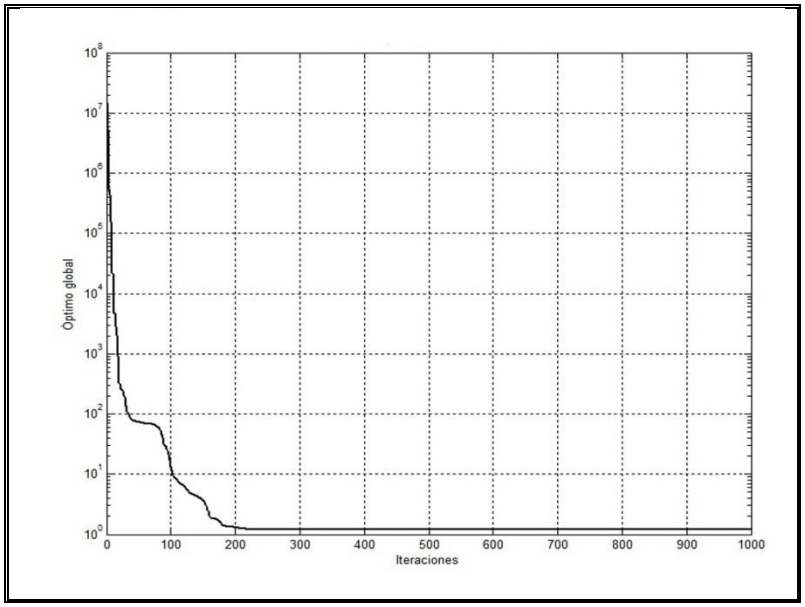


FIGURA 66. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (25 DIMENSIONES)

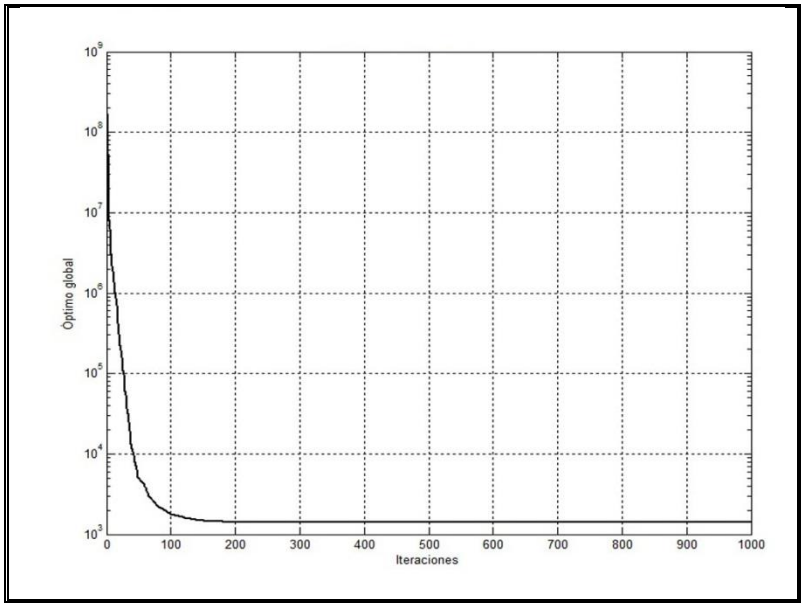


FIGURA 67. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (50 DIMENSIONES)

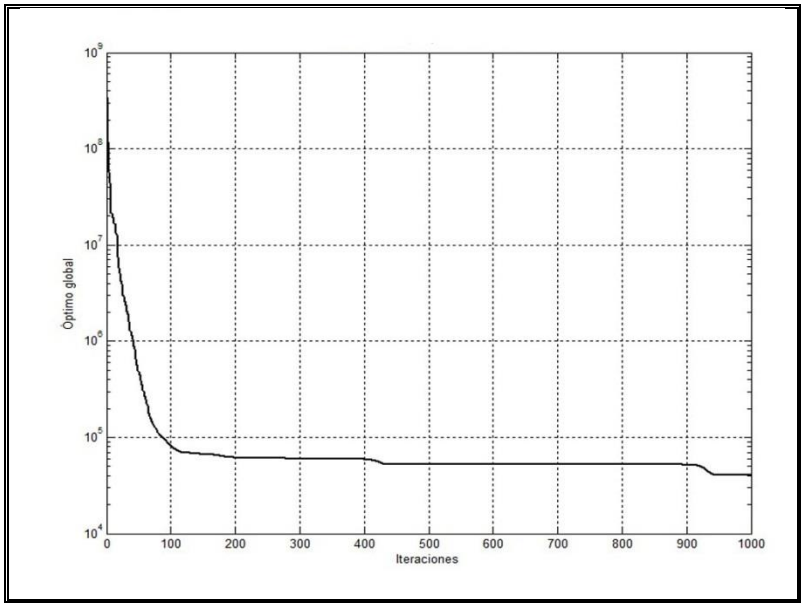
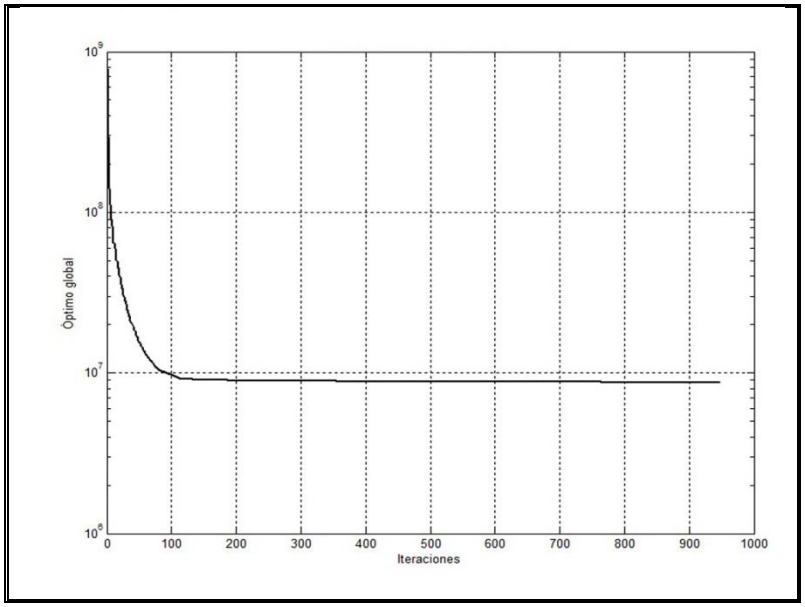


FIGURA 68. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN ROSENBROCK. (100 DIMENSIONES)



Anexo E: Procesamiento Función Schaffer F6.

Figura 69. PROCESAMIENTO DEL PSO EN EL DSP DE LA FUNCIÓN SCHAFFER F6. (2 DIMENSIONES)