SOLUCIÓN NUMÉRICA DEL MODELO NO LINEAL DE UN CIRCUITO EN CONDICIONES DC, HACIENDO USO DE LA META HEURÍSTICA UPSO EN ARQUITECTURAS HIBRIDAS.

WILLIAM JAVIER TRIGOS GUEVARA

ÁNGEL GABRIEL MEZA GARCÍA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2013

SOLUCIÓN NUMÉRICA DEL MODELO NO LINEAL DE UN CIRCUITO EN CONDICIONES DC, HACIENDO USO DE LA META HEURÍSTICA UPSO EN ARQUITECTURAS HIBRIDAS.

Trabajo de Grado para optar al título de
Ingeniero Electrónico

WILLIAM JAVIER TRIGOS GUEVARA
ÁNGEL GABRIEL MEZA GARCÍA

DIRECTOR
PhD. CARLOS RODRIGO CORREA CELY

CO - DIRECTOR
PhD. CARLOS JAIME BARRIOS HERNÁNDEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2013

AGRADECIMIENTOS

A DIOS todo poderoso por todos los favores recibidos día a día, salud, vida, espiritualidad y virtudes, junto a la bendición de tener la compañía de nuestros padres y seres queridos, cada día.

A Carlos J. Barrios, Rodrigo Correa, Mónica Hernández, Carlos Varela y tantos más, quienes con sus grandiosas e invaluables sugerencias hicieron posible el desarrollo de este proyecto

Ángel Gabriel Meza García

William Javier Trigos Guevara.

DEDICATORIA

Dedico especialmente este logro, a mi madre Gloria Liliana García Escandón y a mi padre Joel Meza Hernández, quienes por su gran esfuerzo y motivación, me han permitido culminar exitosamente esta profesión, y que a su vez ven reflejados en mí los frutos de labor.

A mi hermano Joel Meza García, por ser mi mejor amigo y ejemplo a seguir. A mis hermanas, carolina y Jessica Meza García, por ser mi principal motivación. Y a mí hermosa novia Karen D. Barbosa por brindarme su amor, cariño y comprensión. Además de ser mi gran inspiración.

Ángel Gabriel Meza García

A mi madre Edith Natalia Guevara, por ser el motor de mis triunfos, a mi padre Javier Trigos por incitarme a sorprenderlo y a superarme cada día, a mi hermanito Edwin, por ser quien recuerda mi inspiradora niñez.

A mi abuelo Juan Francisco Guevara, aquel hombre que me dio lo mejor de sí en cada palabra, haciendo de mí el hombre que soy.

A mi círculo cercado amigos y amigas, a quienes debo su valiosa compañía, en tantas anécdotas de alegría y tristezas.

A los profesores Pedro Jesús Rojas, Francisco García Russi, Jaime Barrero, Javier Mier, Daniel Sierra, Julio Rugeles Jones, Cesar Duarte,
Carlos Jaime Barrios y Gabriel Vargas, a quienes con su apoyo y conocimientos cultivaron el mí el don de la ciencia.

William Javier Trigos Guevara

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

TÍTULO: SOLUCIÓN NUMÉRICA DEL MODELO NO LINEAL DE UN CIRCUITO EN CONDICIONES DC, HACIENDO USO DE LA META HEURÍSTICA UPSO EN ARQUITECTURAS HIBRIDAS.*

AUTORES: WILLIAM JAVIER TRIGOS GUEVARA, ÁNGEL GABRIEL MEZA GARCÍA.**

PALABRAS CLAVE: optimización, circuitos no lineales, celdas solares, arquitecturas hibridas, OpenMP, CUDA.

RESUMEN

El presente trabajo plantea el modelado de un arreglo paralelo de celdas solares, como escenario para la evaluación del algoritmo UPSO considerado método alternativo para la solución de circuitos no lineales, tomando como estrategia la transformación de un problema de solución de un sistema de ecuaciones, en un problema de optimización numérica.

Por otra parte, basados en el paralelismo inherente tanto para el algoritmo propuesto como para el arreglo de celdas solares, el uso de una arquitectura hibrida brinda la posibilidad de acelerar la ejecución del algoritmo UPSO, a fin de solucionar el modelo matemático asociado al arreglo circuital.

En la primera parte se lleva a cabo una contextualización del proyecto a través de una breve introducción al algoritmo metaheurístico propuesto, a fin de conocer su estructura y diferencias con respecto a su equivalente PSO del cual deriva. Seguidamente la revisión de conceptos relacionados con las celdas solares y las arquitecturas hibridas y/o escalables.

La segunda parte describe el desarrollo del proyecto desde sus tres ejes, la definición del modelo matemático que describe la configuración circuital no lineal, la implementación del algoritmo UPSO en su versión serial y paralela por medio de lenguajes o directivas, y por último, el conjunto de herramientas de software utilizadas para la ejecución del algoritmo sobre las arquitecturas hibridas disponibles.

Finalmente se presenta el análisis de resultados para las distintas versiones del algoritmo, con los cuales se da como caso de éxito el uso de UPSO para la solución a una configuración de celdas solares paralelo. Los resultados son presentados en el conjunto de anexos B, C y D.

---

* Trabajo de Investigación.

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Director: PhD. Rodrigo Correa Cely. Co director: Carlos Jaime Barrios.

TÍTLE: NUMERICAL SOLUTION OF A NONLINEAR MODEL OF A DC CIRCUIT, USING METAHEURISTIC UPSO WITHIN A HYBRID ARCHITECTURE.*

AUTHORS: WILLIAM JAVIER TRIGOS GUEVARA, ÁNGEL GABRIEL MEZA GARCÍA.**

KEYWORDS: optimization, nonlinear circuits, solar cells, hybrid architectures, OpenMP, CUDA.

ABSTRACT

This work proposed the modeling of a parallel array of solar cells, as a context to evaluate UPSO algorithm as an alternative to solve non-lineal circuits, taking as strategy the transformation of a problem defined by a system of equations to one in which the problem is about numeric optimization.

In the other hand, based on the parallelism referred to both, the algorithm and the array of solar cells, the use of a hybrid architecture give us the possibility of accelerating the speed of executions of UPSO algorithms in order to solve the mathematic modeling faster.

In the first part of this document we put into context what this project is about, throughout a brief introduction of the metaheuristic algorithm proposed, in order to know the structure and differences between this and its similar PSO; and then we cited and described the several concepts related to the solar cells and hybrid or scalable architectures.

In the second part we developed the project from three axes, first defining the mathematic model that describes the non-lineal circuit. Additionally development aspects in the implementation of the serial and parallel UPSO algorithm through languages and commands and finally the group of software used to execute the algorithm over the hybrid architectures currently available.

In the last part we present the results of the analysis of the different algorithm versions, whose results show a successful case using UPSO algorithm to solve a configuration of parallel solar cells. These results are presented in the appendices B, C and D.

---

* Degree Work.

** Physical - Mechanical Engineering Faculty. Electrical, Electronics and Telecommunications. Engineering Departament. Director: PhD. Rodrigo Correa Cely. Codirector: Carlos Jaime Barrios.

GLOSARIO

CPU: Unidad de Procesamiento Central.

CUDA: Arquitectura Unificada de Dispositivos de Computo.

DC: Corriente directa

FITNESS: Mínimo valor que toma la función objetivo.

GPGPU: General-Purpose Computing on Graphics Processing Units: es un concepto dentro del campo de la informática asociado a la capacidad de cómputo numérico sobre una GPU.

GPU: Unidad de Procesamiento Gráfico.

KERNEL: Término asociado a la porción de código ejecutado por bloques de threads en la GPU y que puede ser invocado desde la CPU (HOST).

OpenMP: Interfaz de programación de aplicaciones, para la ejecución multihilo.

PSO: Optimización por Enjambre de Partículas.

THREAD: Hilo, Unidad básica de ejecución sobre la Arquitectura CUDA.

UPSO: Optimización por Enjambre de Partículas Unificado.

Función Benchmark: función utilizada comúnmente para determinar la precisión de un algoritmo de optimización.

INTRODUCCIÓN

En la actualidad a menudo se plantean modelos matemáticos más complejos, a fin de tratar de describir de mejor forma el comportamiento real de los sistemas, bien sean mecánicos, eléctricos, biológicos, entre otros. De la mano con la complejidad del modelo matemático que describe al sistema, se halla inherente la calidad del mismo, la cual se ve reflejada en mejores resultados numéricos, que permiten tomar decisiones respecto al diseño de un producto, su construcción o mantenimiento, según sea el ámbito en el que se busca modelar. Por mencionar algunos ejemplos, el modelado de fenómenos ambientales, sistemas circuitales hasta sistemas de seguridad para colisión automotriz (1), (2), (3), (4).

La complejidad de un modelo, puede verse como ese conjunto de funciones matemáticas que relacionan las variables que intervienen en el comportamiento del sistema, esto con el único fin de describir mejor su comportamiento bajo las condiciones dadas. Es común asociar la complejidad de un modelo a la naturaleza lineal o no lineal de su sistema de ecuaciones, así como a la cantidad de ecuaciones que conforman el modelo.

El considerar la existencia de no linealidad, permite muchas veces que el modelo se ajuste y describa de mejor forma el comportamiento del sistema dentro del entorno real; pero a su vez el asumir no linealidad implica un extenuante y complejo cálculo de soluciones para el sistema de ecuaciones.

Por ello a lo largo de los últimos años, se han desarrollado técnicas de optimización numérica que han demostrado ser una buena alternativa en la solución de problemas en ingeniería, que involucran no linealidad y a su vez permiten tener procesos de cálculo más veloces. Entre el conjunto de técnicas numéricas innovadoras se destacan los algoritmos genéticos y de optimización

metaheurísticos.

Dentro de los algoritmos metaheurísticos se encuentran las técnicas PSO[*], basadas en el uso de enjambres de partículas como espacio de muestra para la evaluación de una función objetivo, con el fin de hallar soluciones que satisfagan el modelo matemático que se desee solucionar.

Del algoritmo PSO planteado por Kennedy y Eberhart en 1995, han surgido gran cantidad de modificaciones entre ellas *UPSO*[†], cuya variación está enfocada al equilibrio entre la búsqueda de soluciones locales y globales, esto gracias a la introducción del parámetro de unificación, que permite definir una ponderación entre la búsqueda en el ámbito global y local.

Tomando como referencia lo antes mencionado, respecto a las técnicas metaheurísticas y la importancia del modelado de sistemas; nace la propuesta de evaluar la técnica *UPSO* como alternativa para la solución del sistema no lineal de ecuaciones, que describe el comportamiento de una configuración circuital en condiciones DC. Esto con el fin de brindar un nuevo enfoque en el área de modelado y simulación de circuitos, debido a que se considera a la solución del sistema de ecuaciones como un problema de optimización.

Dados los alcances planteados inicialmente para el desarrollo de la investigación, se considera importante el uso de la unidad GUANE perteneciente a la Universidad Industrial de Santander e interconectada a la red de supercomputación GRID-UIS2; considerando que está arquitectura cuenta con características de hardware que permiten el desarrollo de aplicaciones de alto rendimiento y cálculo científico.

[*] PSO (Particle Swarm Optimization) – Optimización por Enjambre de Partículas.
[†] UPSO (Unified Particle Swarm Optimization) – Optimización por Enjambre de Partículas Unificado.

Para ello, la Universidad a través de las escuelas de ingeniería de sistemas y la E3T*, ha establecido los medios y espacios de trabajo de tipo investigativo; como la disposición de la infraestructura GRID-UIS2, bajo el acompañamiento de los grupos de investigación CEMOS† y GRID‡ respectivamente.

* E3T - Escuela de Ingeniería Eléctrica Electrónica y telecomunicaciones.
† CEMOS, Grupo de Investigación asociado a la E3T, UIS y del cual forma parte los desarrolladores del proyecto junto al profesor Rodrigo Correa Cely.
‡ GRID, Grupo de Investigación asociado a EISI-UIS; y encargado del soporte y manejo de la Unidad GUANE, bajo la dirección del profesor Carlos Jaime Barrios.

1. DESCRIPCIÓN DE LA INVESTIGACIÓN

1.1. PLANTEAMIENTO Y DEFINICIÓN DEL PROBLEMA

La preocupación por el cuidado del medio ambiente, ha hecho más común el uso de fuentes de energía renovable y la creación de sistemas electrónicos de bajo consumo energético, a fin de suplir las necesidades de los seres humanos. En este ámbito es común hablar de los sistemas fotovoltaicos; sistemas de gran proliferación a nivel mundial, gracias a su flexible implementación y bajo costo con respecto a sus semejantes eólicos.

La creciente evolución de los sistemas fotovoltaicos, radica en los procesos de modelado y simulación realizados para el diseño de paneles solares, ya que gracias a ello se puede predecir el comportamiento eléctrico ante variaciones ambientales o de carga. Para que el diseño de paneles solares sea lo más acertado posible, es necesario contar con modelos precisos para describir las partes que lo componen; especialmente por la no linealidad presente en la característica I-V del diodo, elemento principal en los distintos modelos existentes (5).

La construcción de paneles solares es posible gracias a la interconexión de pequeñas celdas solares, que captan la radiación solar para transformarla en energía eléctrica. Hoy día, la interconexión modular es común en la construcción de sistemas eléctricos o electrónicos; ejemplo de ello se tienen los procesos de diseño y construcción de procesadores, memorias RAM, memorias USB, entre otros; los cuales están conformados por módulos que cuentan dentro de sí con miles de elementos circuitales que obedecen a leyes eléctricas de naturaleza lineal y no lineal.

Se puede comprender, que el proceso de diseño de los sistemas electrónicos

requiere de un equipo humano capacitado y dotado de herramientas veloces y precisas, que brinden al diseñador la posibilidad de ejercer análisis sobre el sistema, a fin de lograr un diseño robusto y de altas prestaciones, y que dan paso a los grandiosos dispositivos electrónicos con que se cuenta en la actualidad alrededor del mundo.

Hoy en día existe una buena cantidad de herramientas computacionales, destinadas al análisis de circuitos electrónicos, que se basan en métodos numéricos convencionales como Newton Rapson, Newton Rapson Multidimensional (NRM), Análisis Multipunto, entre otros. (6), (7). Estas herramientas permiten realizar estudios de régimen trascendente, operación DC, análisis en frecuencia entre otros más, a casi cualquier configuración circuital ingresada.

Sin embargo, el proceso de simulación en algunas de estas herramientas puede verse afectado por el número de componentes, y la complejidad del modelo eléctrico asociado a los mismos; teniendo en cuenta que cada elemento circuital por separado puede contar con su propio modelo de naturaleza no lineal, y que en la mayoría de casos es un factor que influye sobre el costo computacional del proceso de simulación.

Teniendo en cuenta lo antes mencionado, se propone evaluar el algoritmo *UPSO* como método alternativo para el modelado y solución de circuitos no lineales en condiciones DC. Por lo tanto el uso y desarrollo del algoritmo *UPSO* es considerado uno de los tres pilares de la investigación.

Como escenario de evaluación para el algoritmo, se plantea el modelado de celdas solares dada la naturaleza no lineal presente en estos elementos, y considerando que el estudio de los sistemas de energía limpia es un campo de investigación actual e importante a nivel científico.

Actualmente alrededor del mundo se han venido desarrollando sistemas que buscan sacar provecho de la energía proveniente de la naturaleza, como el movimiento de las mareas, la radiación solar y las corrientes de aire; esto para suplir las necesidades energéticas tanto en la industria como en los hogares.

En el caso específico de la energía solar, el auge de la celda fotovoltaica ha jugado un papel importante dentro de los sistemas de energía limpia, debido a la facilidad para crear arreglos de las mismas y conformar paneles solares; que suplen niveles de potencia del orden de 10W - 120W en condiciones de operación estándar[*], por unidad de panel solar.

La celda solar como unidad básica de la cual se componen los paneles solares, pasa a ser un segundo pilar en el desarrollo de la presente investigación, y por ello se propone modelar una agrupación de celdas solares en paralelo para conformar un sencillo prototipo de panel solar.

Finalmente, se ha considerado el uso de una arquitectura paralela como herramienta para el desarrollo del proyecto, teniendo en cuenta, el reporte bibliográfico acerca del paralelismo inherente en los algoritmos PSO y la posibilidad de implementación en grano fino del mismo para la mejora de su desempeño. El uso y programación de una arquitectura paralela es considerado el tercer pilar de la investigación (8).

[*] Especificaciones en condiciones de prueba estándar de: 1.000 W/m2, temperatura de la célula 25ºC y masa de aire de 1,5. (51)

1.2. JUSTIFICACIÓN

En busca de métodos alternativos para el modelado y simulación de circuitos eléctricos no lineales, se plantea evaluar la metaheurística *UPSO*, como alternativa para la solución numérica de una configuración en paralelo de celdas solares en condiciones DC.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Solucionar numéricamente el modelo no lineal para una configuración de celdas solares, haciendo uso de la metaheurística UPSO, como alternativa de análisis y solución de circuitos eléctricos.

1.3.2. OBJETIVOS ESPECÍFICOS

- ✓ Determinar el modelo matemático y la configuración del circuito no lineal que será utilizado para evaluar la técnica UPSO.
- ✓ Implementar el algoritmo UPSO en un lenguaje de programación, que permita dar solución numérica a la configuración circuital seleccionada.
- ✓ Evaluar el desempeño del método propuesto utilizando una arquitectura hibrida, tomando como criterios tiempo de ejecución y carga computacional del algoritmo bien sea usando GPUs y/o núcleos para su ejecución.

1.4. ALCANCES

El propósito de este proyecto de investigación es el de evaluar la técnica metaheurística *UPSO*, como alternativa para la solución numérica de circuitos eléctricos que contienen características no lineales.

El caso de estudio planteado en el presente proyecto de investigación, está enfocado a la solución de un modelo no lineal en condiciones DC, para una configuración de celdas solares. Por tanto se omiten las variaciones por efectos termodinámicos, magnéticos o mecánicos en los elementos de circuito.

Infraestructura: GRID-UIS 2 (red de Arquitecturas hibridas).
Tamaño: tamaño final al menos 25 celdas solares en arreglo serie o paralelo.

2. CONCEPTOS PRELIMINARES

La solución numérica de problemas de ingeniería por medio de técnicas de optimización e inteligencia artificial, ha tenido un amplio estudio durante la última década. Entre las técnicas de optimización puede destacarse el uso de la metaheurística PSO (Particle Swarm Optimization) y sus variantes; para la solución de sistemas numéricos multivariables, en gran variedad de aplicaciones científicas y de ingeniería. (9)

En los casos donde un sistema o fenómeno físico es modelado por medio de funciones multivariables y ecuaciones no lineales, la búsqueda de una solución puede no ser tarea fácil, y el uso de técnicas numéricas convencionales no suele ser muy flexible y efectiva. Por ello los métodos de optimización se ofrecen como una alternativa para resolver este tipo de modelos.

Basado en este tipo de escenarios multivariables y no lineales, han surgido una variedad de técnicas numéricas alternativas, como algoritmos genéticos, metaheurísticas, entre otros; que han mostrado tener capacidad para la solución de este tipo de problemas numéricos. En el caso específico de la metaheurística *PSO* y sus variaciones *UPSO*, *MPSO*, *SPSO*, *DPSO* entre otras; se destaca *UPSO* por su alta velocidad de convergencia y estabilidad. (10).

2.1. TÉCNICAS DE OPTIMIZACIÓN PSO

2.1.1. PSO (Particle Swarm Optimization)

PSO es un método de optimización propuesto por Kennedy y Eberhart (11), el cual se basa en la simulación del ambiente social en las bandadas de aves, por lo cual es catalogado como un algoritmo bio-inspirado. PSO en comparación con otras

técnicas numéricas y de optimización como algoritmos genéticos y evolutivos, puede considerarse como un algoritmo sencillo, ya que su núcleo de operaciones radica en la actualización de la velocidad y posición de cada partícula (9).

PSO es un algoritmo basado en población, debido a que emplea un enjambre de partículas, que son evaluadas dentro de un espacio de búsqueda por medio de una función objetivo asociada con el problema de optimización al que se quiere dar solución.

En forma resumida el algoritmo PSO describe la dinámica de las partículas por medio de las ecuaciones de posición y velocidad, presentadas a continuación.

$$\boldsymbol{X}_{\boldsymbol{i}}^{\boldsymbol{j+1}} = \boldsymbol{X}_{\boldsymbol{i}}^{\boldsymbol{j}} + \boldsymbol{V}_{\boldsymbol{i}}^{\boldsymbol{j+1}} \qquad \text{Ecuación 1}$$

$$\boldsymbol{V}_{\boldsymbol{i}}^{\boldsymbol{j+1}} = \boldsymbol{w}\boldsymbol{V}_{\boldsymbol{i}}^{\boldsymbol{j}} + \boldsymbol{c_1 rand_1}(\boldsymbol{p_i} - \boldsymbol{x_i^j}) + \boldsymbol{c_2 rand_2}(\boldsymbol{p_g} - \boldsymbol{x_i^j}) \qquad \text{Ecuación 2}$$

X_i^j, V_i^j Hacen referencia a la posición y velocidad de la partícula i en la iteración j, respectivamente

La actualización de X_i^j, V_i^j se realiza dentro de un proceso iterativo en el cual se tiene en cuenta la posición del mejor histórico de la partícula y el mejor global del enjambre.

En (11), (12), se puede encontrar una descripción detallada acerca de la concepción de enjambre de partículas, así también una breve definición de términos y parámetros que se manejan dentro del estudio de esta técnica de optimización.

2.1.2. UPSO (Unified Particle Swarm Optimization)

UPSO es una de las principales modificaciones de PSO, presentada por Parsopoulos y Vrahatis en el año 2004. Esta técnica es un escenario unificado, que implementa una combinación entre la capacidad de exploración y explotación del algoritmo PSO original. El escenario unificado se caracteriza por ejercer control sobre la velocidad de las partículas del enjambre y a su vez garantizar la rápida convergencia del mismo, si existe una solución óptima para la función objetivo f(x) dentro del espacio de búsqueda.

Para llevar a cabo la combinación del comportamiento local (vecindad) y global (enjambre), *UPSO* se rige por las ecuaciones $\mathcal{G}_{\mathrm{i}}^{(\mathrm{j}+1)}$ y $\mathcal{L}_{\mathrm{i}}^{(\mathrm{j}+1)}$, de las cuales se destaca la existencia de $\mathrm{p}_{\mathrm{gb}}^{\mathrm{j}}$ y $\mathrm{p}_{\mathrm{lb}}^{\mathrm{j}}$ para diferenciar la mejor posición global y local respectivamente; el parámetro χ conocido como coeficiente de constricción está enfocado a garantizar la convergencia del enjambre, acotando la velocidad de las partículas tanto en el ámbito local como global.

$$\boldsymbol{\mathcal{G}}_{\boldsymbol{i}}^{(\boldsymbol{j+1})} = \boldsymbol{\chi}\left[\boldsymbol{\vartheta}_{\boldsymbol{i}}^{\boldsymbol{j}} + \boldsymbol{\varphi_1}\left(\boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{j}} - \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{j}}\right) + \boldsymbol{\varphi_2}\left(\boldsymbol{p}_{\boldsymbol{gb}}^{\boldsymbol{j}} - \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{j}}\right)\right] \qquad \text{Ecuación 3}$$

$$\boldsymbol{\mathcal{L}}_{\boldsymbol{i}}^{(\boldsymbol{j+1})} = \boldsymbol{\chi}\left[\boldsymbol{\vartheta}_{\boldsymbol{i}}^{\boldsymbol{j}} + \boldsymbol{\varphi_1}\left(\boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{j}} - \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{j}}\right) + \boldsymbol{\varphi_2}\left(\boldsymbol{p}_{\boldsymbol{lb}}^{\boldsymbol{j}} - \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{j}}\right)\right] \qquad \text{Ecuación 4}$$

$\boldsymbol{p_i}, es\ la\ mejor\ posición\ local\ de\ la\ particula\ (autoaprendizaje)$

$\boldsymbol{p_{gb}},\ \boldsymbol{p_{lb}}, mejores\ posiciones\ global\ y\ local\ del\ enjambre\ (interacción\ social)$

$\boldsymbol{\varphi_1}, \boldsymbol{\varphi_2}\ son\ valores\ aleatorios\ (normalmente\ distribuidos)$

La integración del comportamiento local y global se da por medio de la Ecuación 1, en la cual se encuentra el parámetro u conocido como factor de unificación, el cual puede tomar valores entre 0 y 1.

$$\boldsymbol{U}_{\boldsymbol{i}}^{\boldsymbol{j+1}} = (\boldsymbol{1}-\boldsymbol{u}) * \boldsymbol{\mathcal{L}}_{\boldsymbol{i}}^{(\boldsymbol{j+1})} + \boldsymbol{u} * \boldsymbol{\mathcal{G}}_{\boldsymbol{i}}^{(\boldsymbol{j+1})} \quad ; \quad \boldsymbol{u} \in [\boldsymbol{0}, \boldsymbol{1}] \qquad \text{Ecuación 5}$$

Este parámetro permite al algoritmo tener ponderación entre exploración y explotación, para la actualización de la velocidad de las partículas, es decir cuando u toma valores cercanos a cero (0), se tiene que *UPSO* se comporta como una variante de *PSO* enfocada a la búsqueda local de soluciones para la función objetivo, o una variante global de *PSO* para valores de u cercanos o iguales a uno (1).

Por su parte la Ecuación 6, planteada para la actualización de la posición de las partículas en *UPSO*, permanece sin modificaciones con respecto a la Ecuación 1, presentada para *PSO*.

$$\boldsymbol{X}_{\boldsymbol{i}}^{\boldsymbol{j+1}} = \boldsymbol{X}_{\boldsymbol{i}}^{\boldsymbol{j}} + \boldsymbol{U}_{\boldsymbol{i}}^{\boldsymbol{j+1}} \qquad \text{Ecuación 6}$$

En el caso de las referencias (13), (14), se presenta una descripción detallada de la técnica *UPSO*.

Para el caso explícito de este proyecto, se cuenta con el apartado *3.1 DESARROLLO ALGORITMO UPSO*, en el que se presentan los detalles acerca del algoritmo utilizado tanto en la versión serial y paralela del algoritmo. En el apartado *3.2 MODELADO Y DEFINICIÓN DE LA FUNCIÓN OBJETIVO*, se expone el problema a solucionar y la respectiva definición de la función objetivo.

2.2. CELDAS SOLARES

Dados los efectos del cambio climático durante las últimas décadas sobre nuestro planeta, el desarrollo de las energías limpias y sistemas bio-sostenibles ha cobrado gran importancia. El desarrollo de sistemas eólicos, solares, geotérmicos, entre otros, ha tenido gran auge e importancia a nivel de investigación como industrial.

Una celda solar, es una unión p-n sobre la cual al incidir la radiación solar se produce la interacción fotón y electrón que da lugar a la generación de electrones libres, que a su vez generan la conocida corriente de isolación o fotocorriente (***Iph***).

El modelo eléctrico de una celda solar presentado en la Figura 1, fue tomado de la referencia (15).

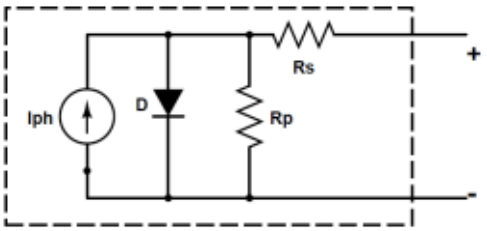


Figura 1 Modelo Eléctrico para una Celda Solar. [Tomada de (15)]

Donde ***Iph*** es la corriente producida debido a los electrones libres por efecto de la interacción fotón-electrón, ***Rs*** es la resistencia interna de la celda como consecuencia de las conexiones óhmicas, y el diodo que representa la unión p-n. (16)

Se decidió abordar el análisis de un arreglo de celdas solares en paralelo, ya que este arreglo es una de las posibilidades que permite implementar paneles solares. Adicionalmente y en virtud de que esta aplicación es una replicación de módulos que involucran modelos no lineales; se considera como un escenario apropiado para la evaluación del algoritmo *UPSO*.

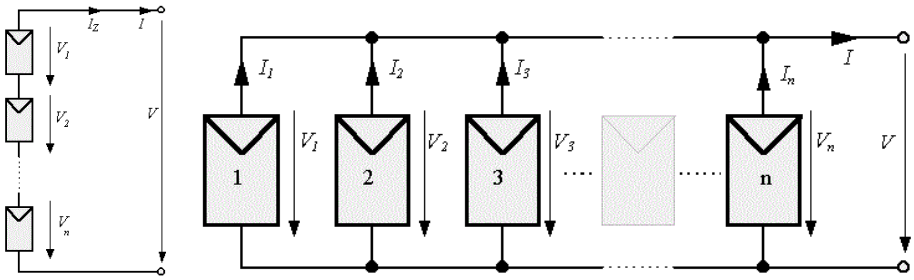


Figura 2 Arreglo de N-Celdas Solares Serie y/o Paralelo. [Autores]

Un panel solar está formado por un conjunto de celdas solares. Para formar un panel solar puede requerirse bien sea de configuraciones de N celdas conectadas en serie, en paralelo o arreglos paralelo serie, según la necesidad o el diseño ofertado por el fabricante. En la Figura 2 se pueden apreciar a modo de ejemplo los arreglos serie y paralelo de celdas solares.

2.3. ARQUITECTURAS ESCALABLES

Las arquitecturas escalables hacen referencia a los sistemas o redes de cómputo con capacidad de incrementar el número de tareas concurrentes ejecutadas, y comúnmente son asociadas con el termino (HPC)[*]. Ejemplo de este tipo de arquitecturas, son las redes de servidores web, supercomputadores y redes de cálculo científico.

En el caso específico de un supercomputador, se puede decir que es por sí solo un fiel ejemplo de una arquitectura escalable, donde se plasma paralelismo en la distribución de procesos, con el fin de reducir el tiempo de ejecución de tareas de software (17).

[*] High performance computing (HPC): nueva perspectiva de la arquitectura de computadores para la solución de problemas de distribución y carga de comunicación (Clústeres), simulación numérica y almacenamiento masivo (dedicated supercomputers) y el uso de internet como base de un modelo de computación de servicios (cloud computing).

Las características de hardware especiales en estos sistemas de cómputo, implican el cambio de la programación convencional por una programación paralela, y que permite la distribución de datos y tareas sobre el hardware disponible.

Esta nueva metodología, permite acelerar la ejecución de los procesos de simulación en aplicaciones de ingeniería; superando así, la limitación en tiempo de ejecución presente en las arquitecturas convencionales.

En cuanto a GUANE, se puede decir que es una máquina basada en *CPU*s y GPUs, que soporta programación en los lenguajes CUDA C, OpenMP, MPI, entre otros.

La información detallada acerca de la topología GRID-UIS2 y las especificaciones con que cuenta la unidad GUANE se puede hallar en (18)

Clúster GUANE							
Ref. Nodos	Arquitectura	RAM por Nodo	Tarjeta de Video	Cores por Nodo	GPUs por Nodo	Interfaz de Red	# de Nodos
Proliant SL390*G7	2.67 GHz Intel Xeon CPU E5640	104 GB	Tesla M2050	8	3564	10Gigabit/ Infiband/Ethernet 10/1000	5
Proliant SL390*G7	2.40 GHz Intel Xeon CPU E5645	104 GB	Tesla M2050	12	3564	10 Gigabit/ Infiband/Ethernet10/1000	3

Tabla 1Caracteristicas de los Nodos de Cálculo en GUANE [Tomada de (18)]

GUANE se compone de 8 unidades de cómputo denominadas *Nodos*[*] con las especificaciones presentadas en la Tabla 1. De los ocho nodos disponibles, solo el nodo Guane02 será el empleado para el desarrollo del proyecto, debido a que sus características son suficientes para ejecución del algoritmo *UPSO* (18).

[*] Nodo en este contexto, hace referencia a una unidad de cómputo con características de hardware específicas para realizar trabajo de alto desempeño.

Guane02 fue seleccionado indistintamente entre los 8 nodos disponibles. El nodo empleado cuenta con 8 cores Intel Xeon @2.67 GHz, 104GB RAM y una *GPU* de la familia NVIDIA Tesla M2050 de características expuestas en la Figura 3.

```
CUDA Device #6
Major revision number:         2
Minor revision number:         0
Name:                          Tesla M2050
Total global memory:           2817982464
Total shared memory per block: 49152
Total registers per block:     32768
Warp size:                     32
Maximum memory pitch:          2147483647
Maximum threads per block:     1024
Maximum dimension 0 of block:  1024
Maximum dimension 1 of block:  1024
Maximum dimension 2 of block:  64
Maximum dimension 0 of grid:   65535
Maximum dimension 1 of grid:   65535
Maximum dimension 2 of grid:   65535
Clock rate:                    1147000
Total constant memory:         65536
Texture alignment:             512
Concurrent copy and execution: Yes
Number of multiprocessors:     14
Kernel execution timeout:      No
```

Figura 3 Características de la *GPU* sobre Nodo Guane02. [Autores]

Sobre el nodo Guane02 se cuenta con la posibilidad de desplegar imágenes de entorno por medio de la herramienta *kadeploy*. La imagen de entorno utilizada para la ejecución del código serial y paralelo, corresponde a una imagen de sistema Linux con la toolkit CUDA 5.0.

```
root@guane02:/home/wtrigos# lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 6.0.4 (squeeze)
Release:        6.0.4
Codename:       squeeze
root@guane02:/home/wtrigos#
```

Figura 4 Características del ambiente squeeze-x64-cuda5 [Autores]

2.4. ANTECEDENTES Y SITUACIÓN ACTUAL (ESTADO DEL ARTE)

Si bien las técnicas de optimización ya cuentan con un amplio recorrido y un sin número de casos de estudio en el ámbito numérico, en el área de modelado de circuitos existe reporte escaso, para la solución de configuraciones circuitales no lineales; por el momento se pueden destacar trabajos previos en el modelado de circuitos desarrollados por Cruz (19), Kumar (20), Kuthadi (21), Samrat-Siba (22), Cooren (23).

Por su parte, los reportes existentes del uso de técnicas de optimización enfocadas al estudio de celdas solares y/o sistemas fotovoltaicos, solo buscan identificación de parámetros internos del panel solar (5), (24), (25).

3. ACERCA DEL DESARROLLO

El principal fin detrás de esta investigación, es evaluar el uso de técnicas de optimización enfocadas al modelado de circuitos no lineales en condiciones DC, por lo cual se realiza la transformación del ámbito de la solución de un sistema de ecuaciones de origen circuital, al de un problema de optimización apoyado en el teorema de raíces reales presentado en (26).

Dentro de este nuevo escenario de solución juega un rol importante la definición de una función objetivo, que represente de forma acertada el sistema de ecuaciones que describe el circuito.

El tamaño de la población de partículas, los coeficientes de aceleración, de constricción entre otros; son parámetros que están íntimamente relacionados con el algoritmo metaheurístico utilizado para la búsqueda de mínimos en la función objetivo, y que darían lugar a la solución del circuito planteado como problema en este proyecto de investigación.

Por otra parte, la definición de vecindarios y el paralelismo inherente en la estructura del algoritmo, hacen pensar en el uso de una metodología de programación paralela, para brindar una mayor agilidad y desempeño al algoritmo al momento de dar solución al problema planteado.

3.1. DESARROLLO ALGORITMO UPSO

3.1.1. TOPOLOGÍA IMPLEMENTADA

El concepto de vecindarios no es algo nuevo que se defina dentro de los algoritmos *PSO* y sus derivaciones, por el contrario, esta concepción se utiliza en

la mayoría de los algoritmos evolutivos numéricos. La definición de una topología de vecindad no tiene otro objetivo más que el representar de alguna forma, la influencia que tienen las partículas vecinas con posibles soluciones, y su respectivo aporte sobre la solución global de la función objetivo.

Según la opinión de Rui (27), el uso de modelos de población a través de vecindarios, da a los algoritmos evolutivos la posibilidad de no depender estrictamente de la función objetivo, debido a que cada partícula aporta información relevante a los demás miembros de la población, acerca de la existencia de un mejor mínimo con respecto al lugar donde actualmente se encuentran ubicadas.

Para el caso específico de este proyecto, se ha optado por definir una topología Ring para la implementación de vecindarios locales sobre el algoritmo *UPSO*. La razón de acoger este método, es la de facilitar la implementación paralela del algoritmo; ya que el uso de una topología de conectividad completa entre partículas vecinas, haría que la arquitectura paralela se viese más enfocada a la comunicación entre nodos o Cores de *GPU*, que a la ejecución del algoritmo; lo cual afectaría fuertemente el desempeño y la velocidad de cálculo del mismo.

En la Figura 5, se presentan 2 tipos de topologías de vecindarios entre partículas, la de conectividad completa (Wheels) y la de anillo (Ring), de uso común para vecindarios locales.

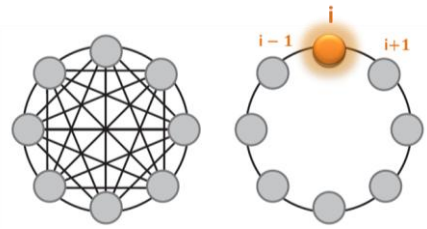


Figura 5 Topología (Conectividad)
(a) Full conectividad (Global) (b) Anillo (local) [Tomado de (28)]

3.2. MODELADO Y DEFINICIÓN DE LA FUNCIÓN OBJETIVO

Acorde a los objetivos planteados previamente para el desarrollo del presente proyecto, se optó por modelar un arreglo paralelo de celdas solares como un escenario viable, para la evaluación de *UPSO* como alternativa de modelado y solución en circuitos no lineales. Esto en virtud de la no linealidad en el modelo de la celda solar y la inherente escalabilidad de las celdas para conformar paneles solares; permitiendo ampliar la dimensión del sistema de ecuaciones y haciendo posible la evaluación del desempeño sobre arquitecturas escalables. Para una idea inicial acerca de la configuración circuital a estudiar se presenta la Figura 6.

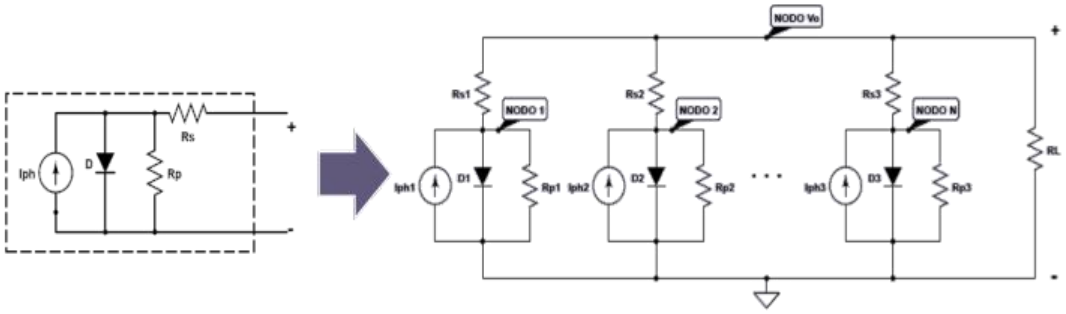


Figura 6 Contextualización del modelo. [Autores]

El modelo eléctrico de una celda solar está compuesto por una fuente de corriente, que entrega una corriente constante debido a que se asume una radiación solar constante. Teniendo en cuenta que en la investigación no se aborda el estudio ante variaciones de temperatura en la celda, ni variaciones en la radiación solar.

El comportamiento del diodo presente en el modelo de la celda solar obedece a la Ec.7, donde $\boldsymbol{V_D}$ es el voltaje diferencial en los terminales del diodo, $\boldsymbol{n}$ es el factor de no linealidad, $\boldsymbol{q}$ la constante de carga del electrón, $\boldsymbol{k}$ la constante de Boltzmann y $\boldsymbol{T}$ la temperatura termodinámica del diodo.

$$\boldsymbol{i_{DN}(V_D) = I_0\left(e^{\frac{q.V_D}{nKT}} - 1\right)} \qquad \text{Ecuación 7}$$

El modelo circuital básico de la celda solar asumido para la definición de la función objetivo, se presenta en la Figura 7. Donde ***Iph*** es la corriente producida por efecto de la interacción fotón-electrón, ***Rs*** es la resistencia interna de la celda como consecuencia de las conexiones óhmicas y el diodo que representa la unión p-n. (16)

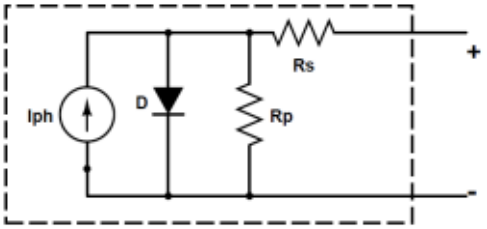


Figura 7 Modelo Eléctrico para una Celda Solar. [Tomada de (21)]

Las celdas solares pueden ser conectadas entre sí para formar un arreglo, el cual puede ser de tipo de serie, paralelo o una combinación de ellos. En general, en cuanto mayor sea la cantidad de celdas que formen el arreglo, mayor será la energía eléctrica producida. A medida que se agregan celdas en serie se incrementa el voltaje a la salida, o en caso de conexión paralelo aumenta la capacidad de corriente en la salida.

Se decidió abordar el análisis de un arreglo de celdas solares en paralelo, debido a que el arreglo permite implementar paneles solares entregando diferentes valores de corriente, a medida que se agregan celdas.

El prototipo de panel solar propuesto para la evaluación de *UPSO*, está formado por un conjunto N celdas conectadas en paralelo, como se ve en la Figura 8 .

3.2.1. ANÁLISIS MATEMÁTICO DEL MODELO CIRCUITAL

Dado que el circuito que modela cada celda se encuentra en paralelo dentro del esquema general, se puede realizar un análisis de nodo replicable en cada celda.

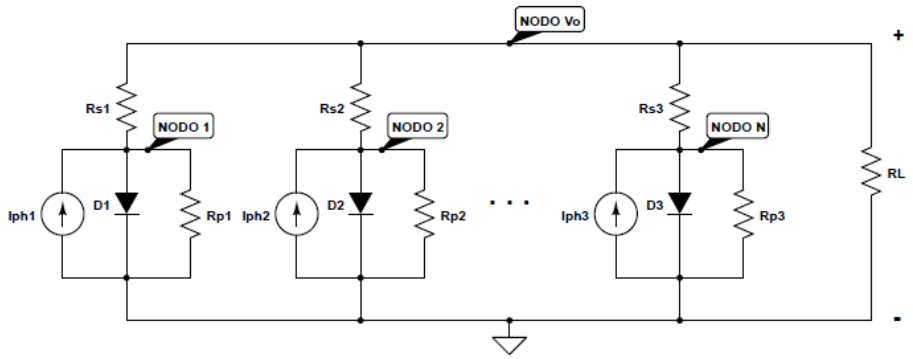


Figura 8 Arreglo de N-Celdas Solares en paralelo. [Autores]

El nodo ***Vo***, conecta la salida de cada celda con la resistencia de carga y obedece a la Ec.8. Donde ***Vo*** es el voltaje de salida del panel y ***Vi*** corresponde al voltaje presente en el ánodo de los diodos del modelo circuital de cada celda. Considerando que la corriente que recibe la carga es igual a la sumatoria de las corrientes que entregan las celdas.

(LIK) Nodo V_o:

$$\mathbf{f_0 = \sum_{i=1}^{N} \frac{V_i - V_0}{R_{Sn}} - \frac{V_0}{R_L} = 0} \qquad \text{Ecuación 8}$$

La ley de intensidades de Kirchhoff (LIK) sobre los nodos V_1, V_2 hasta V_n, se presenta como la diferencia entre la corriente ***If***, y suma de las corrientes en las resistencias ***Rp***, ***Rs*** y la corriente a través de cada diodo (***ID***); la cual depende del voltaje entre sus terminales.

El sistema de ecuaciones que se genera a partir del análisis de nodos, aumenta a medida que se agregan celdas al arreglo. Este sistema de ecuaciones es igualado a cero, a fin hacer posible la definición de una función objetivo para aplicar el método de optimización propuesto en este documento.

Nodo V_1:

$$\mathbf{f_1 = \frac{V_1}{R_{P1}} + i_{D1}(V_1) + \frac{V_1 - V_0}{R_{S1}} - I_{f1} = 0} \qquad \text{Ecuación 9}$$

Nodo V_2:

$$\mathbf{f_2 = \frac{V_2}{R_{P2}} + i_{D2}(V_2) + \frac{V_2 - V_0}{R_{S2}} - I_{f2} = 0} \qquad \text{Ecuación 10}$$

. .

. .

. .

Nodo V_N:

$$\mathbf{f_N = \frac{V_N}{R_{PN}} + i_{DN}(V_N) + \frac{V_N - V_0}{R_{SN}} - I_{fN} = 0} \qquad \text{Ecuación 11}$$

La definición de una función objetivo consistente para *UPSO*, se evidencia ante la existencia de un sistema cuadrado de N+1 variables con N+1 ecuaciones; teniendo en cuenta que la variable adicional corresponde a la asociada con el nodo Vo, y la cual es una incógnita para el problema planteado.

Finalmente la función objetivo, es definida como la sumatoria de los cuadrados de las funciones $\boldsymbol{f_i}$ obtenidas del análisis nodal del circuito; de forma que cuando se halle un mínimo de la $\boldsymbol{f_{obj}}$, los términos dentro de la raíz en la Ecu.12 no se anulen, si tienen la misma magnitud pero diferente signo. Para mayor detalle acerca del teorema empleado para la definición de la función objetivo, puede remitirse a (26), donde el profesor correa define el teorema de las raíces reales.

$$\mathbf{f_{Obj} = \sqrt{f_0^2 + f_1^2 + f_2^2 + \ ... + \ f_N^2} = \sqrt{f_0^2 + \sum_i^N f_n^2}} \qquad \text{Ecuación 12}$$

3.3. IMPLEMENTACIONES DEL ALGORITMO

3.3.1. IMPLEMENTACIÓN DEL ALGORITMO EN CÓDIGO SERIAL

Teniendo en cuenta la cantidad de información local y global disponible en las partículas a lo largo del número de iteraciones del algoritmo, se hace evidente que la implementación del código serial presenta una limitante computacional, en pro de la búsqueda de soluciones al modelo circuital por medio de la función objetivo.

La limitante computacional se fundamenta en tiempo de ejecución del algoritmo, donde una tanda de 10 experimentos con 12 celdas emplea 30 minutos sobre el PC Intel Corei5, teniendo un promedio de 3 minutos por experimento. Por tanto es tomado como referencia para la evaluación del desempeño relativo del algoritmo *UPSO*.

3.3.2. IMPLEMENTACIÓN DEL ALGORITMO EN CÓDIGO PARALELIZADO

La investigación moderna basada en modelos, hace uso de computación avanzada para dar mayor rendimiento a la experimentación, frente a los problemas que involucran alta escalabilidad, no linealidad y gran volumen de datos como se expone en (29).

El paralelismo inherente en el algoritmo *UPSO*, hace pensar en el uso de programación paralela como alternativa para lograr reducir los tiempos de ejecución del algoritmo, y adicionalmente hacer mejor uso del hardware disponible sobre los computadores multinúcleo, o que disponen de tarjetas gráficas con capacidad de cómputo. (*GPGPU*[*])

[*] **GPGPU**: General-Purpose Computing on Graphics Processing Units: es un concepto dentro del campo de la informática asociado a la capacidad de cómputo numérico sobre una GPU.

3.4. HERRAMIENTAS UTILIZADAS

En este apartado se exponen las herramientas utilizadas durante el desarrollo de la investigación, y que hicieron posible la evaluación de la técnica *UPSO* como herramienta de solución a circuitos no lineales en condiciones DC.

3.4.1. HERRAMIENTAS PARA LA SIMULACIÓN DE CIRCUITOS

Entre los objetivos adicionales de este trabajo de investigación, se busca contrastar el resultado obtenido mediante una herramienta de simulación al circuito propuesto, frente a la respuesta del algoritmo de optimización. Dado que se desea ejecutar el algoritmo sobre un sistema de cómputo basado en multinúcleo y/o *GPU*s, se requiere que la herramienta de simulación utilizada como referencia soporte este tipo de arquitecturas, para contar con condiciones equivalentes para la ejecución del simulador y el algoritmo.

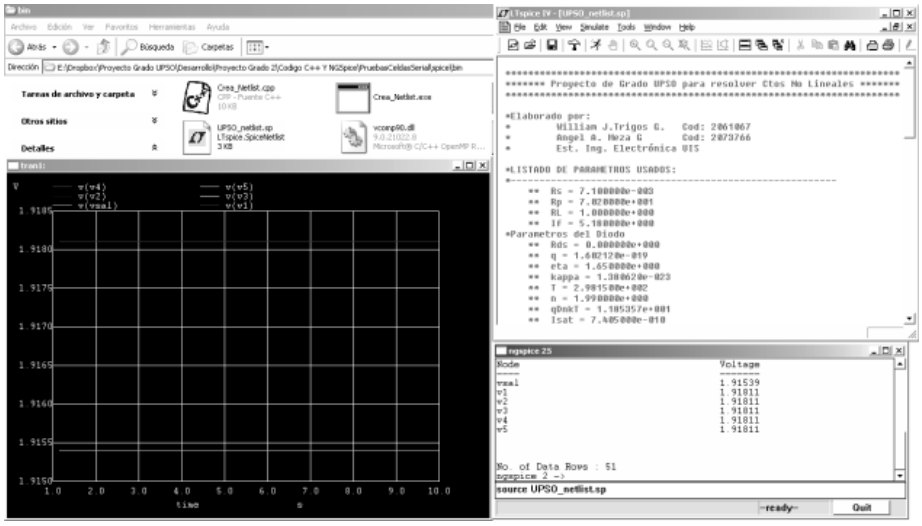


Figura 9 Detalle de una Ejecución de NGSpice. [Autores]

Desafortunadamente hasta el momento de reportar esta investigación, no se dispone de herramientas libres y/o asequibles que utilicen múltiples núcleos o

*GPU*s, para llevar a cabo procesos de simulación. Las herramientas existentes con estas características son de costo elevado, para ser consideradas viables en la validación de este logro adicional; que sería el evaluar la velocidad del algoritmo con respecto a un simulador de circuitos, que ejecute simulaciones de forma concurrente.

Se hace uso de NGSpice como software de simulación de circuitos, para el contraste de los resultados obtenidos mediante el algoritmo. Aunque NGSpice hace uso de directivas OpenMP para la aceleración del proceso de simulación, el uso de las mismas, se ejerce tan solo sobre modelos BSIM3, BSIM4 para algunos transistores (30).

Por tanto, los tiempos de ejecución de NGSpice no pueden servir de referencia para una medición de desempeño con respecto al algoritmo paralelo propuesto, ya que su funcionalidad en paralelismo no se ejerce sobre el modelo de diodo, el cual es el componente no lineal involucrado en el modelo circuital de la celda solar.

3.4.2. GENERADOR AUTOMÁTICO DE NETLIST

Figura 10 Concepción de la Herramienta Genera_Netlist.cpp [Autores]

Los ficheros Netlist son una alternativa para la simulación de circuitos eléctricos.

Básicamente los ficheros Netlist describen los atributos e interconexiones de los elementos circuitales entre sí, como en el caso de resistencias, diodos y Mosfets que componen el circuito a simular. A cada uno de estos elementos se le puede asignar valores de parámetros como IS, TT, RS, para definir el modelo PSICE asociado al elemento durante el proceso de simulación.

La creación de esta herramienta generadora de Netlist y su funcionalidad es descrita por medio de la Figura 10. Esta pequeña aplicación tiene como fin la creación del archivo con extensión .sp, que contiene el modelo de celdas paralelo basado en el número de celdas que se desee simular. En un principio puede pensarse que es algo inoficioso, sin embargo el pensar en desarrollar un Netlist para más de 10 o 15 celdas paralelo, suena algo extenuante y poco provechoso, ya que implica una buena cantidad de líneas de descripción.

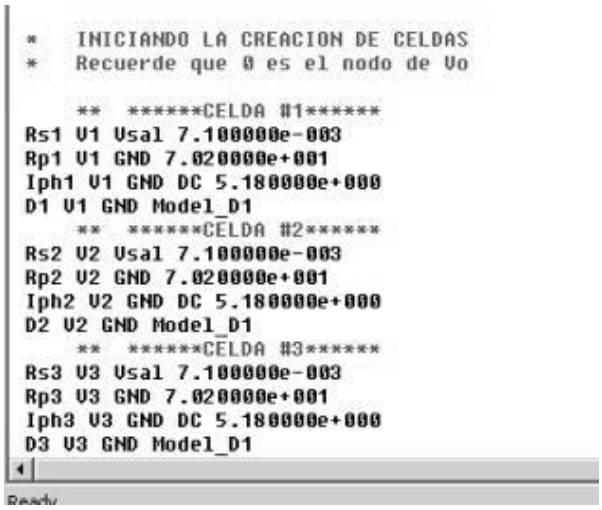


Figura 11 Detalle de un Netlist Generado desde C++. [Autores]

Por ello, se ha creado un subprograma en lenguaje C++ que permite exportar un fichero de simulación en extensión .sp, en el cual se encuentran desde la inserción de modelos para los elementos, la descripción Netlist del circuito y las directivas de control de NGSpice para la posterior simulación en análisis DC.

La Figura 11, presenta el detalle de un fichero de salida del generador de Netlist, donde se aprecia la creación de 3 celdas conectadas en paralelo entre los nodos Vo y GND respectivamente.

Si existe alguna inquietud o desea conocer más acerca de la utilidad y uso de los archivos Netlist, se remite al lector a la fuente (31) donde existe una documentación detallada acerca del tema.

3.4.3. DESPLIEGUE DE ENTORNOS OAR Y KADEPLOY

Este conjunto de aplicativos es utilizado para el despliegue de imágenes de entorno, sobre arquitecturas hibridas o escalables como en el caso de la plataforma GUANE, la cual ha sido usada para la ejecución del algoritmo *UPSO*, tanto para las pruebas preliminares con *funciones Benchmark*, como para el modelo de celdas solares.
Para ser más claros al respecto, una imagen de entorno es algo similar a un sistema operativo sobre el cual se corren intérpretes y compiladores de lenguaje serial o paralelo como gcc, g++, OpenMP, CUDA u OpenGL, según sea el caso. Para que esto sea posible, la imagen de entorno debe contar con las librerías necesarias tanto para la porción serial como paralela del respectivo código fuente.

Para mayor información acerca de la utilidad y uso de Kadeploy puede referirse a la fuente (32).

3.4.4. LENGUAJES Y DIRECTIVAS DE PROGRAMACIÓN

Esta breve contextualización se hace con el fin de enfocar conocimientos básicos de los lenguajes de programación, en especial del lenguaje c++, con el fin de facilitar la comprensión de algunos términos de la programación en paralelo que

serán mencionados más adelante.

La POO* hace parte de un paradigma de la programación, que usa a los objetos, sus características e interacciones para crear programas. Los lenguajes POO tienen a la clase como su unidad básica de programación, la cual permite crear instancias u objetos de una clase, que tienen la posibilidad de implementar los conceptos de herencia, polimorfismo, encapsulamiento entre otros conceptos propios de la POO. (33)

En lo que refiere al desarrollo del proyecto, los conceptos y enfoques aplican a C++ como lenguaje POO utilizado para el desarrollo del algoritmo *UPSO*. C++ en si es una extensión de lenguaje C nativo, que puede implementar funciones inline, declaración de tipos de datos, declaración de clases y/o estructuras abstractas, además de un compilador con la capacidad de interpretar e implementar las técnicas de la POO.

```
Inicializar () {
For cada partícula i do
        For cada dimensión j do
                Inicializar posición y velocidad con numero aleatorio (par_ij)
                Inicializar posiciones del mejor local, vecino y global(Pb_ij,
Gb_ij,Vb_ij)
        End for
End for

For cada partícula i do
        CalcularFitness(posición de la partícula)
        CalcularFitness(posición de la mejor partícula local)
End for
Inicializar el global fitness (gfit)
For cada dimensión j do
        Actualizar mejor posición global (pg_i)
```

Figura 12 Pseudocódigo de la inicialización del algoritmo UPSO Implementado. [Autores]

* POO – Programación Orientada a Objetos

El pseudocódigo del algoritmo implementado se presenta en la Figura 12, que describe la porción de inicialización del algoritmo, y la Figura 13 que presenta el cuerpo del algoritmo *UPSO* para la ejecución dentro de un proceso iterativo controlado.

Al algoritmo se ingresan los parámetros necesarios para la ejecución, como son las constantes propias del algoritmo *UPSO* como w, C1, C2, u, Dimensión del problema y demás, así como también la función objetivo del problema a evaluar.

```
USPO() {
For iteración k do
        For cada partícula i do
                CalcularFitness(posición de la partícula)
        End for
        For cada partícula i do
        If fitness de la partícula es menor que el mejor fitness vecino (Vbfit)
        Actualizar mejor fitness vecino (Vbfit)
                For cada dimensión j do
                        Actualizar mejor posición vecino
                End For
        End if
        If fitness de la partícula es menor que el menor fitness local (pbfit)
                Actualizar mejor fitness local (pbfit)
                For cada dimensión j do
                        Actualizar mejor posición local
                End For
                If si el mejor fitness local es menor que el mejor fitness global
                        Actualizar mejor fitness global (gfit)
                End if
        End if
        For cada particula i do
                For cada dimensión j do
                        Actualizar posición y velocidad de acuerdo a Ecs. ( 5 y 6 )
                End for
        End for
End for
}
```

Figura 13 Pseudocódigo del Proceso iterativo de UPSO. [Autores]

Posterior a ello se realiza el proceso de inicialización de velocidades y posiciones de las partículas, por medio de valores aleatorios dentro de los límites del espacio de búsqueda.

Una vez inicializado el enjambre de partículas, se realiza una evaluación inicial de la función objetivo, que permite inicializar los valores del mejor vecino de cada partícula, así como el mejor local y global del enjambre.

Una vez se cuenta con las mejores partículas inicializadas, empieza el proceso iterativo del algoritmo *UPSO*, el cual se apoya en las ecuaciones de velocidad, posición y la respectiva evaluación de la función objetivo, que actualiza la posición en cada partícula con respecto a las mejores partículas del enjambre.

Cabe aclarar que los resultados del algoritmo *UPSO* varían tras cada ejecución, dada la naturaleza aleatoria de la posición inicial en cada partícula dentro del espacio de búsqueda. Por ello es común la ejecución de varios experimentos, para identificar la precisión de una respuesta, la cual puede ser candidata directa a solución numérica del problema que modela la función objetivo.

3.4.5. API* OPENMP†

OpenMP es un estándar desarrollado por el grupo de los mejores distribuidores de hardware y software en su momento, como AMD, IBM, HP, Cray, NEC entre otros; los cuales buscaban un modelo de programación que ofreciera portabilidad y escalabilidad a los programadores. El estándar OpenMP está disponible en lenguajes de programación como Fortran, C y C++, por medio de directivas y clausulas (#pragma) que permiten al programador indicarle al hardware qué parte del código secuencial se ejecute en forma paralela.

* API - Application Program Interface
† Open Multi-Processing

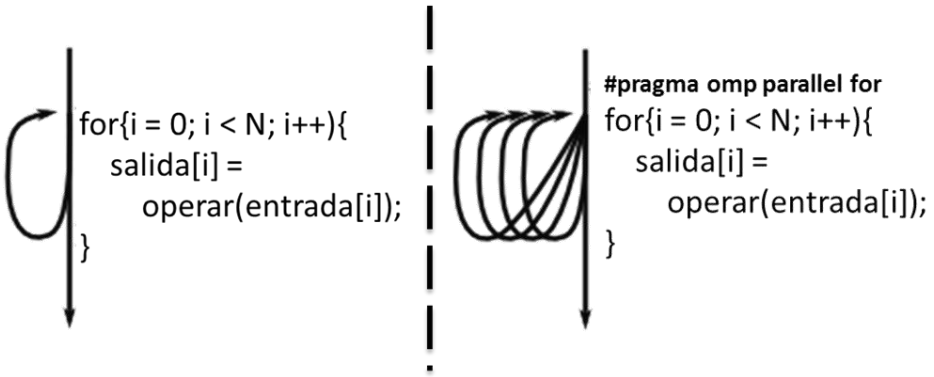


Figura 14 Paralelismo de un ciclo 'for' sin dependencia de datos. [Autores]

La distribución de trabajo en hilos es manejado directamente por el sistema operativo, guiado por directivas con el prefijo ***#pragma*** antepuesto. Estas directivas impuestas sobre el código fuente, permiten definir el ambiente para ejecución paralelo, hasta el comportamiento de las variables *(shared, private*, etc.) durante la ejecución.

Las directivas de OpenMP han sido utilizadas para la distribución de trabajo de algunos bucles iterativos de la versión serial, creando así una primera versión paralelizada de *UPSO* por medio de directivas. Es de aclarar, que la aplicación de estas directivas requirió de ciertas modificaciones en el manejo de las variables, para eliminar la dependencia de datos. Gracias a esto es posible que la versión OpenMP presente un mejor desempeño con respecto a la versión serial. Los resultados obtenidos de esta implementación mediante directivas OpenMP son presentados en la sección de resultados.

Para la ampliación de conceptos y directivas OpenMP se remite al lector a las referencias (34), (35), (36); las cuales sirvieron de soporte al desarrollo de la versión paralela de *UPSO*.

3.4.6. TECNOLOGÍA CUDA

CUDA es una plataforma de computación paralela y modelo de programación desarrollado por Nvidia, que permite aprovechar los recursos computacionales de la *GPU* (unidad de procesamiento gráfico), para desarrollar aplicaciones que involucren procesamiento masivo de datos de forma paralela.

CUDA está disponible desde el año 2006, momento desde el cual fue acogida con éxito por los desarrolladores de software, científicos e investigadores, quienes vieron la oportunidad de dar mayor provecho a las tarjetas gráficas disponibles en portátiles, estaciones de trabajo y supercomputadores. Las aplicaciones comúnmente desarrolladas con *CUDA* van desde astronomía, física, minería de datos, hasta la visualización de moléculas.

El uso de *CUDA* para el desarrollo de aplicaciones, se puede abordar de tres formas:

- Paralelizar automáticamente los bucles en Fortran o código C utilizando directivas OpenACC para aceleradores.
- Haciendo uso de librerías especializadas como MKL BLAS, IPP, FFTW, para que tanto *CPU* y *GPU* las utilicen en acelerar la ejecución de ciertas partes de la aplicación.
- Desarrollar algoritmos paralelos personalizados, utilizando un lenguaje de programación conocido como C, C + +, C #, Fortran, Java, Python, etc. Y que responden al modelo de programación y ejecución *CUDA*, tanto en manejo apropiado de jerarquía de memoria y como en la invocación de *kernels*[*] para su ejecución.

[*] Kernel - termino asociado a la porción de código ejecutado por bloques de threads en la GPU y que puede ser invocado desde la CPU (HOST).

Esta última opción se tomó como metodología para desarrollar la versión paralela de *UPSO* en *CUDA*, ya que permite administrar la jerarquía de memoria y el control de la comunicación entre la *GPU* y la *CPU*.

Para tener una noción del código desarrollado en lenguaje CUDA se presenta la Figura 15, en la cual se pueden apreciar las tareas ejecutadas en forma serial y paralela, por la CPU y GPU respectivamente.

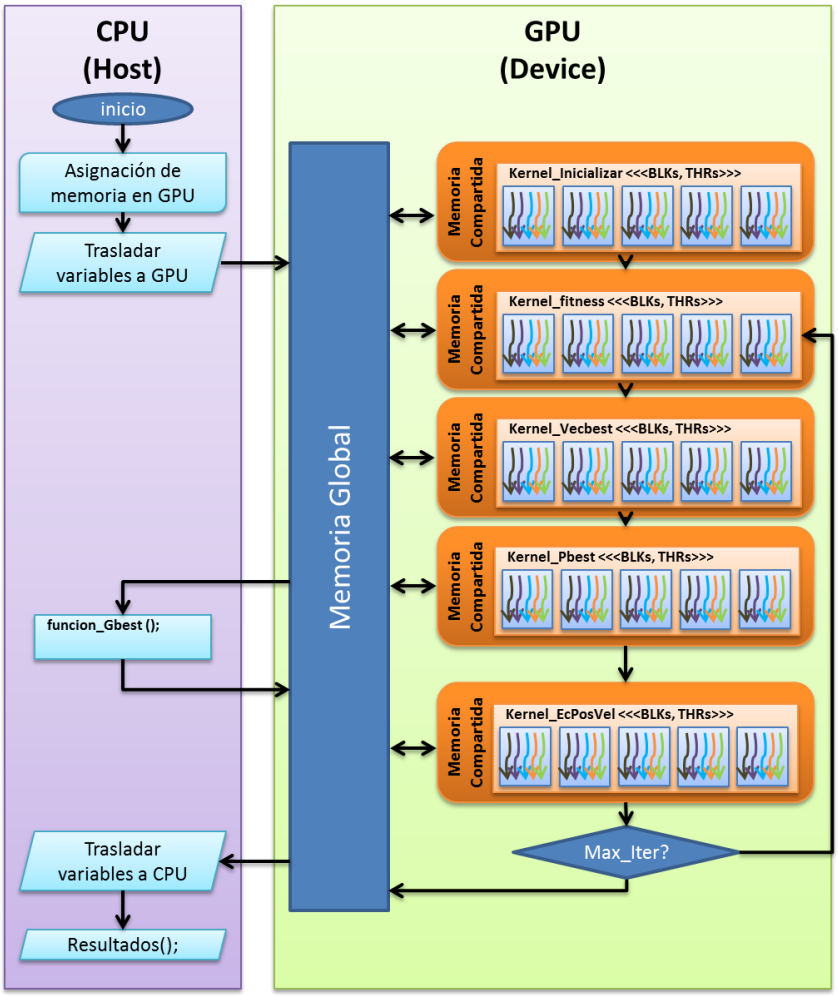


Figura 15 un detalle de la implementación en lenguaje CUDA [Autores]

Para profundizar sobre el modelo de programación que utiliza *CUDA* y sus directivas, refiérase a (37), (38), (39).

4. RESULTADOS Y ANÁLISIS

Para esta etapa del proyecto se planteó desarrollar un conjunto de pruebas preliminares, usando funciones Benchmark. Esto con el fin de validar el correcto funcionamiento de los algoritmos desarrollados.

Por su parte, para la evaluación del modelo de celdas solares se asume como valor teórico aceptable, el valor de las tensiones de nodo obtenidos por medio de la herramienta (NGSpice), con el fin de comparar la calidad de las respuestas obtenidas mediante el uso de la técnica metaheurística *UPSO*.

Las pruebas desarrolladas a lo largo de este capítulo fueron realizadas sobre una máquina de Intel corei5 y el nodo guane02, que corresponden a las siguientes especificaciones.

Maquina	**Referencia**	**procesador**	**# de cores**	**RAM**	**GPU**	**Capacidad de procesamiento**	**Núcleos CUDA**
PC[*]	Samsung®, NP- Q330-JS03CO	Intel® Core™ i5 M460 @2.53GHz	4	4 GB	GeForce 310M	1.1	16
Nodo[†] **Guane02**	Proliant SL390*G7	Intel Xeon E5640@2.67 GHz	8	104 GB	Tesla M2050	2.0	448

Tabla 2 Especificaciones de las maquinas usadas para la investigación. [Basada en (38)]

A continuación se presentan los resultados obtenidos en una serie de simulaciones realizadas con el algoritmo UPSO. Las pruebas se realizan evaluando funciones Benchmark (40).

[*] Samsung®, NP- Q330-JS03CO Intel® Core™ i5 M460 @2.53GHz, RAM 4 GB, Sistema Operativo Ubuntu 64-bit, bajo entorno de terminal y compilador GCC 4.5.2

[†] Proliant SL390*G7, 2.67 GHz Intel Xeon CPU E5640, RAM 104 GB, Sistema Operativo Debian 64-bit, bajo entorno de terminal y compilador GCC 4.5.2

Es importante resaltar, que la selección de parámetros para la ejecución del algoritmo se ha basado en el barrido del parámetro u entre [0,1), a lo largo de 10 experimentos, y a partir de los cuales se obtiene el valor de fitness promedio, a fin de seleccionar el valor de u que ofrece menor fitness promedio.

El uso del parámetro de fitness promedio para la selección correcta del parámetro u, no es un método empírico propuesto por los autores y que ha brindado frutos, para los problemas Benchmark y el problema de optimización para el modelo de celdas solares. Los parámetros C1, C2 y w, han sido tomados acorde a la referencia (1).

4.1. PRUEBAS PRELIMINARES CON FUNCIONES BENCHMARK

Las pruebas se realizan evaluando funciones Benchmark, las cuales son utilizadas comúnmente para determinar la precisión de un algoritmo de optimización para calcular los puntos máximos o mínimos de una función (40).

Las funciones Benchmark presentadas en la Tabla 3 (41), Están diseñadas como problemas de optimización en los cuales se requiere calcular el valor mínimo de la función objetivo. Estas y otras funciones plantean variedad de modelos matemáticos de diferente complejidad, permitiendo evaluar la precisión del algoritmo, considerando que se presentan varios máximos o mínimos relativos que podrían ocasionar que el algoritmo entregue respuestas erradas.

Se evalúan los problemas de optimización global Ackley, Griewank y Sphere, en los que se realizan 10 ejecuciones para cada función. En la Tabla 4, se muestra el promedio del óptimo global de cada función, y su desviación estándar (devStd), además de la solución calculada en cada dimensión.

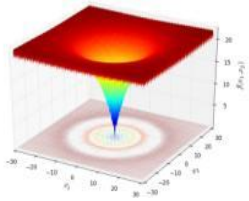
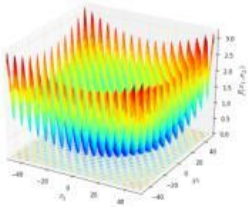
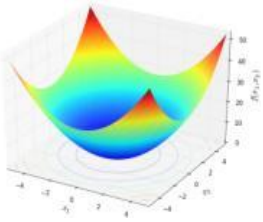
	Función Ackley	Función Griewank	Función Sphere
Gráfico			
Modelo matemático $f(x)$	$= -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}}$ $-e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$	$= \frac{1}{4000}\sum_{i=1}^{n} x_i^2$ $-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$= \sum_{i=1}^{n} x_i^2$
Límites	$x_i \in (-32, 32);$ $i = 1, \ldots, n.$	$x_i \in (-600, 600);$ $i = 1, \ldots, n.$	$x_i \in (-1, 1);$ $i = 1, \ldots, n.$
Óptimo global	$f(x_i) = 0;\ x_i = 0;$ $i = 1, \ldots, n.$	$f(x_i) = 0;\ x_i = 0;$ $i = 1, \ldots, n.$	$f(x_i) = 0;\ x_i = 0;$ $i = 1, \ldots, n.$

Tabla 3 Funciones Benchmark [Autores]

Dado que las funciones de prueba son multivariables se define un máximo de 12 dimensiones, considerando las limitantes en el tiempo de ejecución para la implementación serial. En esta etapa se aprecia que el algoritmo cuenta con una precisión del orden de 1e-6 para los valores de la función objetivo evaluada en los puntos críticos. La precisión en los valores de la solución para 3, 6, 12 dimensiones, son del orden de 1e-4 para la gran mayoría de las funciones Benchmark evaluadas.

Dim	Función Benchmark	Optimo global (referencia)	Optimo global (UPSO)	Fobj-devStd	Solución		
3	Ackley	0,00	0,00000536	3,97E-06	X_1=-2,95E-04	X_2=-2,41E-02	X_3=4,44E-07
	Griewank	0,00	0,00000705	5,09E-06	X_1=3,07E-05	X_2=-5,06E-04	X_3=4,79E-04
	Sphere	0,00	0,00000769	6,11E-06	X_1=9,59E-05	X_2=-6,93E-05	X_3=3,57E-05
6	Ackley	0,00	0,00001004	7,64E-06	X_1= -1,36E-11	X_3= -7,57E-09	X_5= -1,11E-09
					X_2= 1,06E-10	X_4= -1,20E-10	X_6= 1,43E-10
	Griewank	0,00	0,00000901	5,47E-06	X_1= 8,47E-04	X_3=-2,01E-03	X_5= 4,57E-05
					X_2=2,87E-05	X_4=1,91E-04	X_6=3,13E-03
	Sphere	0,00	0,00001320	5,91E-06	X_1= 6,93E-05	X_3=5,04E-04	X_5= -4,43E-04
					X_2=2,78E-05	X_4=2,32E-06	X_6=-1,10E-04
12	Ackley	0,00	0,00000888	4,34E-06	X_1= 1,00E-03	X_5= -3,09E-02	X_9= 1,33E-02
					X_2= -3,74E-03	X_6= -3,12E-03	X_{10}= -4,52E-02
					X_3= -6,56E-03	X_7= -1,07E-03	X_{11}= -5,38E-02
					X_4= 3,64E-02	X_8= -1,63E-02	X_{12}= 9,51E-07
	Griewank	0,00	0,00000935	5,23E-06	X_1= 1,48E-04	X_5=-9,83E-04	X_9= 2,59E-03
					X_2= -2,87E-04	X_6=-1,23E-03	X_{10}=2,08E-04
					X_3=1,69E-04	X_7=-2,29E-04	X_{11}=3,67E-03
					X_4=6,33E-04	X_8=-1,99E-03	X_{12}=1,49E-03
	Sphere	0,00	0,0000144	4,64E-06	X_1= -1,79E-04	X_5= -4,73E-04	X_9= -8,75E-05
					X_2=1,55E-04	X_6=-6,05E-04	X_{10}=-2,84E-04
					X_3=-6,82E-05	X_7=2,53E-04	X_{11}=-1,29E-06
					X_4=-1,25E-04	X_8=-3,11E-04	X_{12}=-4,36E-04

Tabla 4 Evaluación de problemas Benchmark con el algoritmo de optimización UPSO. [*Las pruebas se realizaron en el PC Intel Corei5*] [Autores]

Los parámetros empleados para el algoritmo en la evaluación de funciones Benchmark son presentados en la Tabla 5.

función	u	w	C1	C2
Ackley	0.5	0.7298	2.05	2.05
Griewank	0.9	0.7298	2.05	2.05
Sphere	0.5	0.7298	2.05	2.05

Tabla 5 Parámetros del algoritmo para pruebas Preliminares. [Autores]

4.2. PRUEBAS SOBRE EL MODELO DE CELDAS PARALELO.

En este apartado se presentan los resultados para las implementaciones serial y en paralelo del algoritmo *UPSO*, resolviendo la configuración de celdas solares en paralelo propuesta. Los resultados han sido contrastados con las respuestas obtenidas por medio del Simulador NGSpice, los cuales se presentan en la Tabla 6, y que se asumen como valor teórico de referencia.

Nodo	**DIMENSIÓN**					
	1	**3**	**6**	**12**	**15**	**25**
Vi	1,9373	1,95376	1,95887	1,96162	1,96218	1,9631
Vo	1,28298	1,66988	1,80541	1,88165	1,89766	1,92386
Vrs	0,65432	0,28388	0,15346	0,07997	0,06452	0,03924

Tabla 6 Voltajes de nodo obtenidos en NGSpice. [Autores]

Los parámetros adoptados para el modelo de celdas solares paralelo, se apoyan en los resultados del estudio presentado en (42), para el cual se identifican los parámetros de un panel solar por medio de la herramienta MATLAB Symulink, y que se presentan en la Tabla 7.

Parámetros	**Iph [A]**	**Io [A]**	**Rs [Ω]**	**Rp [Ω]**
Shell SP-70	4.7	$4.206\ 10^{-10}$	0.51	91

Tabla 7 Parámetros para el modelo de la celda Solar. [Basada en (42)]

El modelo para celdas solares de un diodo, ha sido considerado para el desarrollo de la investigación según las fuentes bibliográficas (25), (43). Aunque en estas referencias se plantea que el modelo con un diodo, presenta deficiencias ante variaciones de temperatura, no interfiere con el objetivo de estudio en condiciones DC. Adicionalmente se menciona que la presencia de la resistencia Rp contribuye en forma significa en la precisión del modelo, pero exige un esfuerzo computacional significativo (24), (42).

Con la intención de ser objetivos y concisos, se discuten los resultados para el modelo de tres (3) celdas. Por comodidad del lector se evita la extensión del documento con la discusión de los demás experimentos; a fin de que pueda evaluar su interés en la propuesta y tomar su punto de vista respecto a los resultados obtenidos.

Los parámetros empleados para el algoritmo serial en la evaluación del modelo de celdas solares de 3, 6 y 12 celdas se presentan en Tabla 8.

función	**u**	**w**	**C1**	**C2**
Celdas solares	0.9e-3	0.7298	2.05	2.05

Tabla 8 Parámetros del algoritmo serial para 3, 6 y 12 celdas paralelo. [Autores]

Toda la información relevante de estas pruebas se condensa en el apartado 5, donde se exponen las conclusiones del proyecto en general con base en los resultados presentados en los anexos B, C y D.

4.2.1. IMPLEMENTACIÓN DEL ALGORITMO EN CÓDIGO SERIAL

Posterior a las pruebas preliminares, con las cuales se comprueba el correcto funcionamiento del algoritmo. Se realiza un estudio de los parámetros w, C1, C2, número partículas, entre otros; con el objetivo de alcanzar la convergencia para la implementación serial del algoritmo.

Los parámetros empleados para las pruebas preliminares y el modelo de celdas solares en paralelo se presentan en las Tablas 5 y 8 respectivamente.

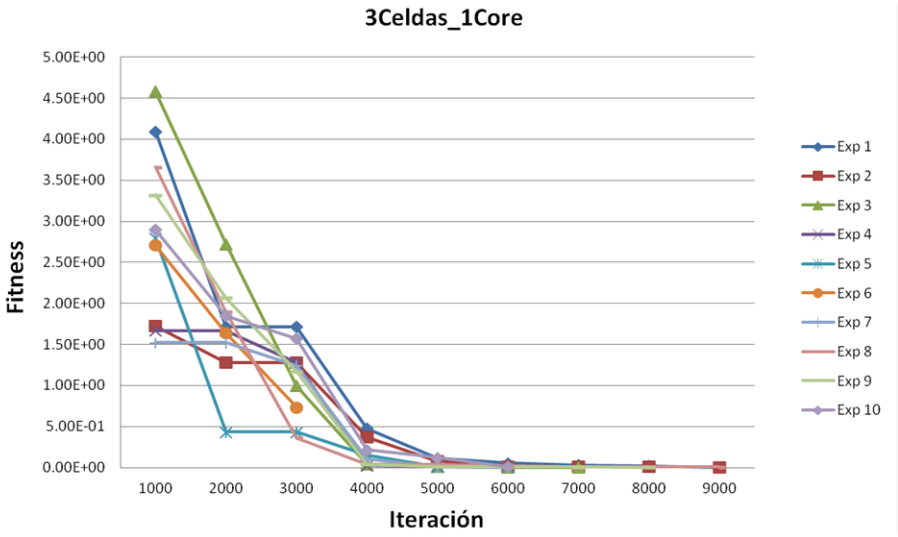


Figura 16 Detalle de la convergencia UPSO para 3 Celdas Solares [Autores]

Producto de todo lo anterior se aprecia en la Figura 16, que para la implementación serial de dimensión 4, alrededor de las 4000 iteraciones el valor de la función objetivo es inferior a la unidad. La Figura 16 hace referencia a los valores de la función objetivo para 10 experimentos efectuados a un modelo de 3 celdas paralelo, equivalente a un problema numérico de 4 dimensiones.

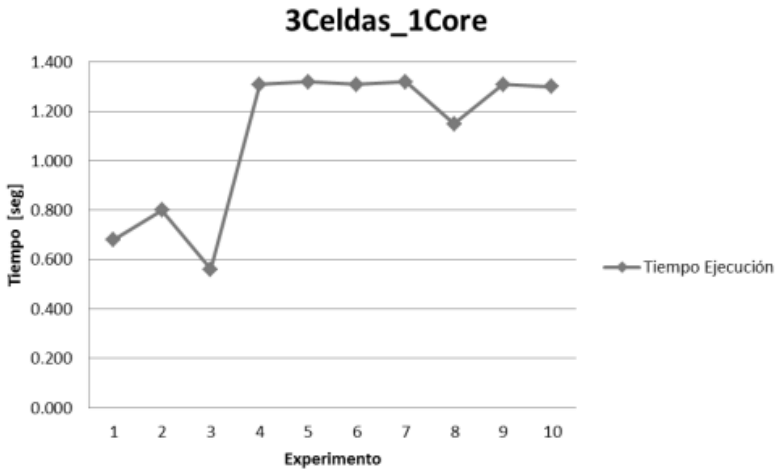


Figura 17 Detalle tiempo de ejecución, con un modelo de 3 Celdas. [Autores]

En cuanto al tiempo de ejecución empleado para cada uno de los experimentos realizados con el modelo de 3 celdas, se aprecia que empieza a mantenerse parcialmente constante a partir del cuarto experimento según la Figura 17.

Esta afirmación toma mayor fuerza para pruebas de mayor dimensión, como puede apreciarse en la Figura 18, donde el tiempo de ejecución es prácticamente constante para la ejecución con 6 celdas.

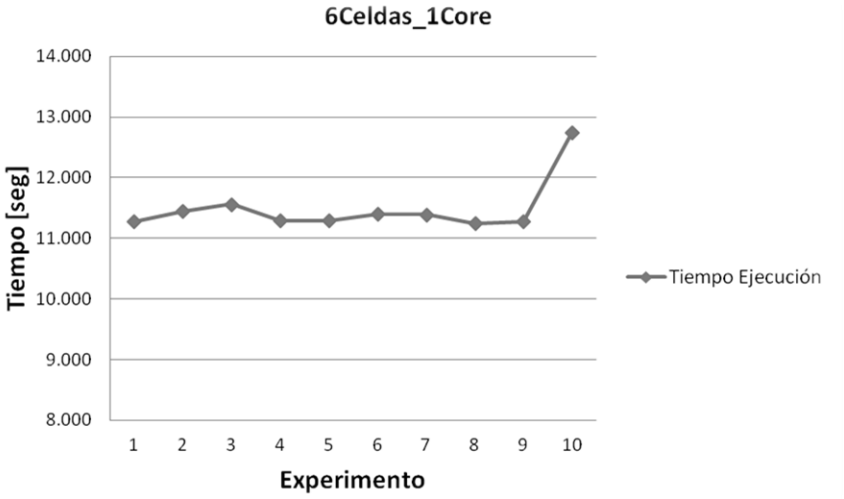


Figura 18 Detalle de la Figura 35 del anexo B. [Autores]

En cuanto a la implementación con 12 celdas, el tiempo de ejecución pasa al orden de los minutos; tanto así, que la ejecución de 10 experimentos sobre el PC Intel corei5 tarda cerca de 35 min, teniendo un tiempo promedio por experiencia de 3.5 minutos.

Es importante resaltar que el modelo asociado a 3, 6 y 12 celdas, es equivalente a un problema de 4, 7 y 13 dimensiones respectivamente; debido a que el nodo Vo es una variable adicional a los nodos Vi calculados. Por ello la Tabla 9, presenta 4 valores de tensión para la ejecución de 3 celdas.

3 celdas (4 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje
1	V[0]	1,66669	0,830	830	4,79E-03	1,66669	1,9E-03
	V[1]	1,95056				1,95054	1,6E-03
	V[2]	1,95054					
	V[3]	1,95051					
2	V[0]	1,66635	0,830	830	7,26E-03	1,66635	2,1E-03
	V[1]	1,95049				1,95056	1,6E-03
	V[2]	1,95063					
	V[3]	1,95057					
3	V[0]	1,66726	0,650	650	3,46E-03	1,66726	1,6E-03
	V[1]	1,95057				1,95061	1,6E-03
	V[2]	1,95066					
	V[3]	1,95060					
4	V[0]	1,66675	0,500	500	3,07E-03	1,66675	1,9E-03
	V[1]	1,95057				1,95056	1,6E-03
	V[2]	1,95055					
	V[3]	1,95057					
5	V[0]	1,66714	0,640	640	3,81E-03	1,66714	1,6E-03
	V[1]	1,95052				1,95057	1,6E-03
	V[2]	1,95060					
	V[3]	1,95059					
6	V[0]	1,66736	0,360	360	3,93E-03	1,66736	1,5E-03
	V[1]	1,95059				1,95063	1,6E-03
	V[2]	1,95065					
	V[3]	1,95066					
7	V[0]	1,66750	0,610	610	2,89E-03	1,66750	1,4E-03
	V[1]	1,95057				1,95060	1,6E-03
	V[2]	1,95062					
	V[3]	1,95061					
8	V[0]	1,66645	0,830	830	5,30E-03	1,66645	2,1E-03
	V[1]	1,95054				1,95054	1,6E-03
	V[2]	1,95054					
	V[3]	1,95054					
9	V[0]	1,66681	0,770	770	3,74E-03	1,66681	1,8E-03
	V[1]	1,95056				1,95056	1,6E-03
	V[2]	1,95059					
	V[3]	1,95053					
10	V[0]	1,66714	0,630	630	3,73E-03	1,66714	1,6E-03
	V[1]	1,95054				1,95059	1,6E-03
	V[2]	1,95064					
	V[3]	1,95059					

Tabla 9 Resultados de 10 Experimentos para un modelo con 3 celdas solares en paralelo. [Autores]

Respecto al análisis del circuito planteado en el apartado 3.2.1, es claro que el voltaje de salida debe ser menor al de cada nodo V[i]; esto para cumplir con la

convención de la ley de voltajes de Kirchhoff (LVK). De forma que la diferencia de potencial existente entre Vo y Vi, sea equivalente al voltaje VRs presente en cada celda solar.

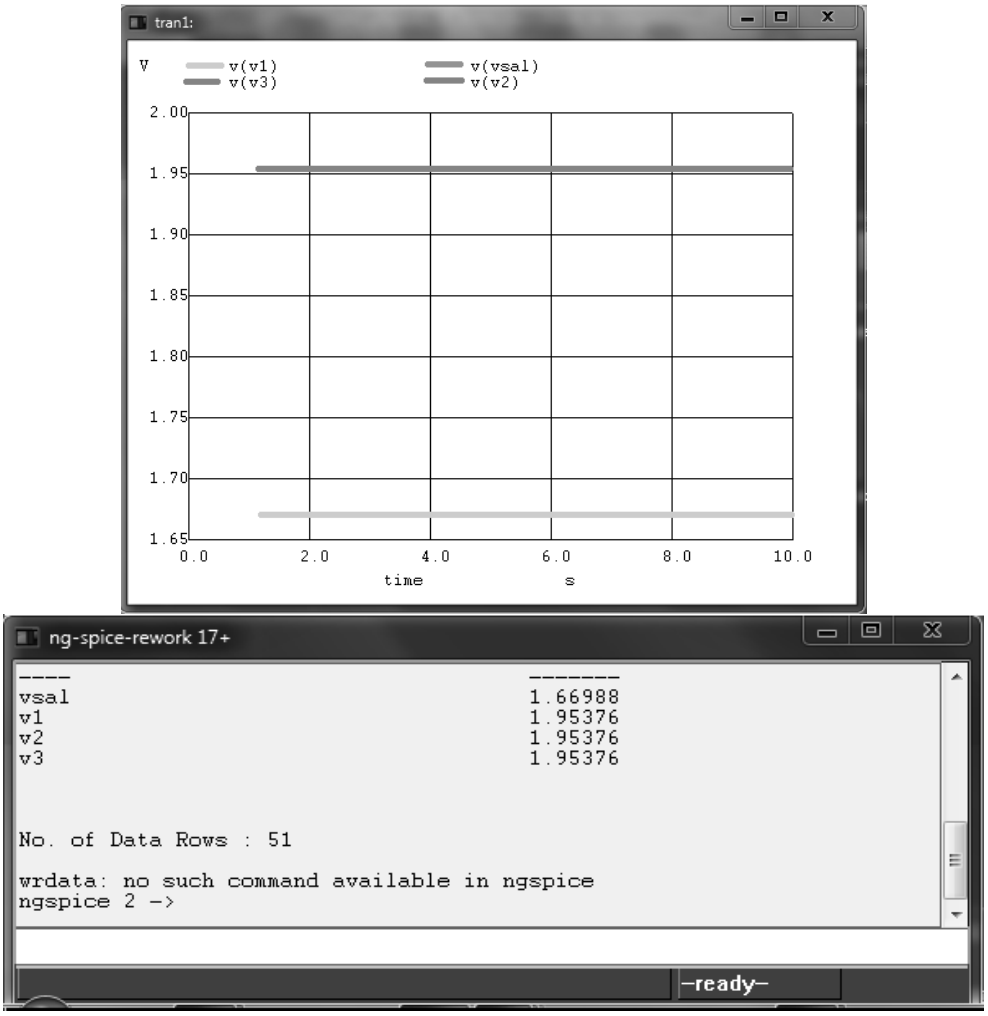


Figura 19 Detalle de la ejecución del Netlist de 3 Celdas Solares Paralelo. [Autores]

El valor del voltaje Vo es referenciado en la Tabla 9 como el nodo V[0], y en la Figura 19 se presenta la captura de la simulación obtenida con NGSpice, con los respectivos valores de voltaje en los nodos Vi y Vo tomados como referencia teórica para la evaluación del algoritmo *UPSO*, para el modelo con 3 celdas.

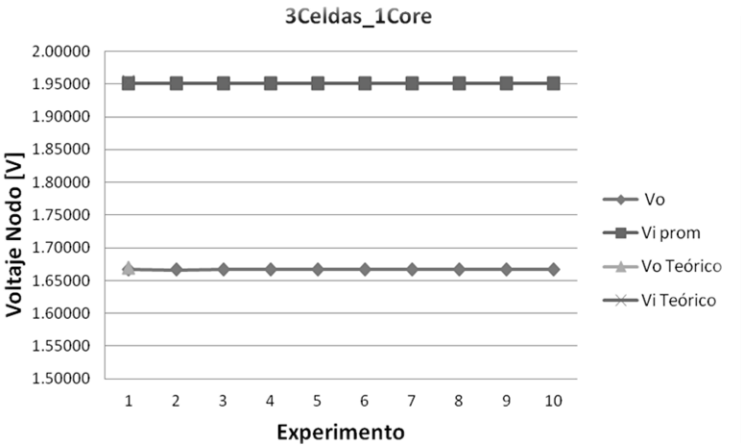


Figura 20 Resultados Experimentales para modelo de 3 celdas paralelo. [Autores]

De la simulación se puede ver que el voltaje sobre los nodos Vi es exactamente el mismo, y que la diferencia de potencial entre ellos y el nodo Vo es:

$$V_{Rs} = V_i - V_o = 1.95376 - 1.66988 = 283.88\, mV$$

Por su parte, los resultados experimentales *UPSO* referidos de la Tabla 9, y de los cuales se toman arbitrariamente 3 experiencias, equidistantes para evaluar las respuestas obtenidas.

Experiencia 3:

$$V_{Rs} = V_i - V_o = 1.95057 - 1.66726 = 283.31\, mV$$
$$V_{Rs} = V_i - V_o = 1.95066 - 1.66726 = 283.4\, mV$$
$$V_{Rs} = V_i - V_o = 1.95060 - 1.66726 = 283.34\, mV$$

Experiencia 6:

$$V_{Rs} = V_i - V_o = 1.95059 - 1.66736 = 283.23\, mV$$
$$V_{Rs} = V_i - V_o = 1.95065 - 1.66736 = 283.29\, mV$$
$$V_{Rs} = V_i - V_o = 1.95066 - 1.66736 = 283.3\, mV$$

Experiencia 9:

$$V_{Rs} = V_i - V_o = 1.95056 - 1.66681 = 283.75\, mV$$
$$V_{Rs} = V_i - V_o = 1.95059 - 1.66681 = 283.78\, mV$$
$$V_{Rs} = V_i - V_o = 1.95053 - 1.66681 = 283.72\, mV$$

Los resultados obtenidos mediante el algoritmo, son consistentes con respecto la diferencia de potencial VRs para el modelo de 3 celdas. A fin de tener idea de que tan cercanos al valor de referencia están los resultados, se introduce un indicador de error.

En el caso específico de la experiencia 3, los índices de error son:

$$E_{Vi} = \frac{V_{i\,teorico} - \overline{V_{\imath\,UPSO}}}{V_{i\,teorico}} = \frac{1.95376 - \frac{1.95057 + 1.95066 + 1.95060}{3}}{1.95376} = 1.6 * 10^{-3}$$

$$E_{Vo} = \frac{V_{o\,teorico} - V_{o\,UPSO}}{V_{o\,teorico}} = \frac{1.66988 - 1.66726}{1.66988} = 2.1 * 10^{-3}$$

En el caso específico de la experiencia 6:

$$E_{Vi} = \frac{V_{i\,teorico} - \overline{V_{\imath\,UPSO}}}{V_{i\,teorico}} = \frac{1.95376 - \frac{1.95059 + 1.95065 + 1.95066}{3}}{1.95376} = 1.6 * 10^{-3}$$

$$E_{Vo} = \frac{V_{o\,teorico} - V_{o\,UPSO}}{V_{o\,teorico}} = \frac{1.66988 - 1.66726}{1.66988} = 1.5 * 10^{-3}$$

En el caso específico de la experiencia 9:

$$E_{Vi} = \frac{V_{i\,teorico} - \overline{V_{\imath\,UPSO}}}{V_{i\,teorico}} = \frac{1.95376 - \frac{1.95056 + 1.95059 + 1.95053}{3}}{1.95376} = 1.6 * 10^{-3}$$

$$E_{Vo} = \frac{V_{o\,teorico} - V_{o\,UPSO}}{V_{o\,teorico}} = \frac{1.66988 - 1.66726}{1.66988} = 1.8 * 10^{-3}$$

Por efectos prácticos, se ha asumido el cálculo del error en los nodos Vi, con el promedio de los voltajes obtenidos en *UPSO*, a fin de comparar con el valor de referencia constante obtenido mediante NGSpice.

Los niveles de error del orden de 1.6e-3 y 1.8e-3, para la tensión de nodo Vi y Vo respectivamente. El comportamiento del error para los 10 experimentos es presentado en la Figura 21.

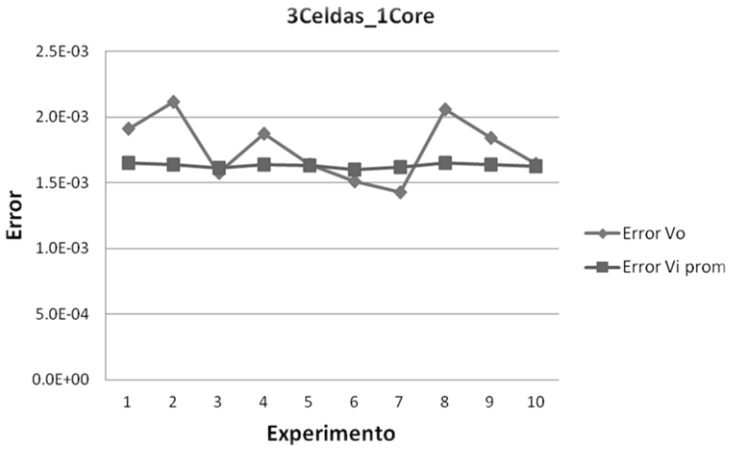


Figura 21 Error para 10 experimentos con el modelo de 3 celdas. [Autores]

Las respuestas obtenidas en los escenarios de 6 celdas (Figura 22) y 12 celdas (Figura 23), muestran que el margen de error se mantiene entre 1.5e-3 y 2e-3, para las tensiones Vo y Vi respectivamente.

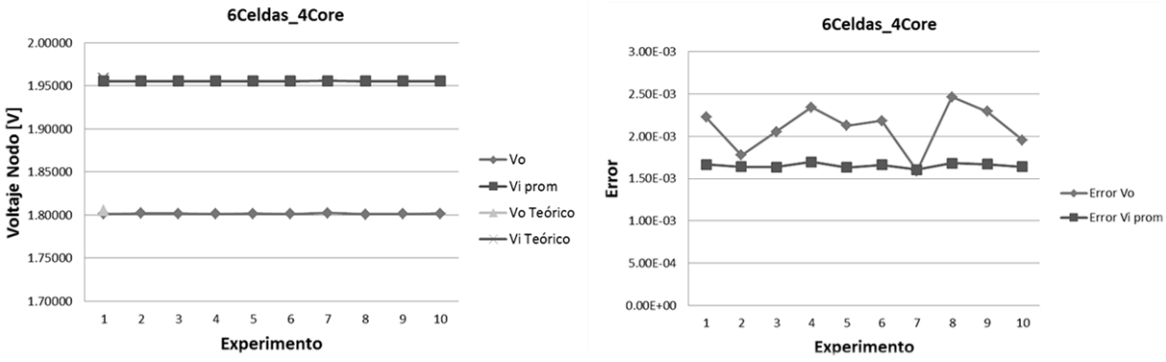


Figura 22 Detalle para ejecución con el modelo de 6 Celdas paralelas. [Autores]

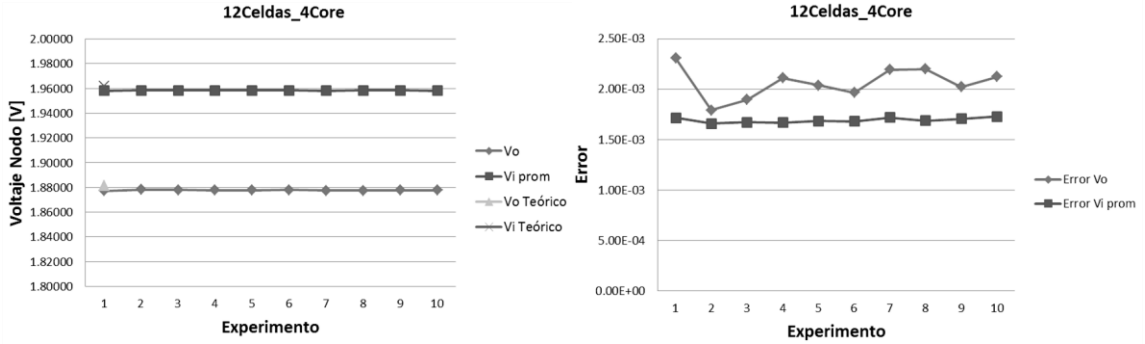


Figura 23 Detalle para ejecución con el modelo de 12 Celdas paralelas. [Autores]

Este último hecho habla a favor del algoritmo desarrollado, teniendo en cuenta que *UPSO* es un método de optimización experimental, basado en estadística o tendencias de una respuesta sobre un conjunto de experimentos.

4.2.2. IMPLEMENTACIÓN DEL ALGORITMO EN CÓDIGO PARALELIZADO

Los resultados presentados en el Anexo C, corresponden a la versión OpenMP del algoritmo, ejecutada sobre el equipo Core i5; son similares a los obtenidos sobre el nodo Guane02 presentados en el anexo D, correspondientes al algoritmo en lenguaje CUDA. Por lo cual nos limitaremos a hacer el análisis en el punto límite de ejecución serial (12 celdas) y que tarda cerca de 30 minutos en la ejecución de 10 experimentos *UPSO*. Teniendo un promedio de 3 min por ejecución.

4.2.3. DETERMINACIÓN DEL DESEMPEÑO.

Teóricamente se esperaría que a medida que se dobla el número de procesadores, el tiempo de ejecución debería reducirse a la mitad. Pero cabe aclarar, que todo algoritmo está compuesto por una o más porciones paralelas y otra porción completamente serial.

La estimación teórica del desempeño de un algoritmo paralelizado es posible a través del modelo matemático de la ley de Amdahl, presentada en la Ecuación 13.

$$\boldsymbol{S = \frac{1}{(1-P)+\frac{P}{N}}} \qquad \text{Ecuación 13}$$

Para tomar como referencia este índice de aceleración, se espera que tanto el algoritmo serial como paralelo se desarrollen sobre las misma arquitectura. Un ejemplo de ello es el algoritmo OpenMP, el cual se ejecuta sobre múltiples unidades de procesamiento, de las cuales solo una es usada para el algoritmo serial; caso contrario al del algoritmo CUDA, el cual se ejecuta sobre tarjetas gráficas con una arquitectura completamente diferente a los procesadores multinúcleo.

Por ello se establece el índice de aceleración teórico solo para la versión OpenMP, el cual se muestra en la Tabla 10, para cada una de las dimensiones.

Característica	DIMENSIÓN			
	1 Celda	3 Celdas	6 Celdas	12 Celdas
No. de hilos	4	4	4	4
Fracción P*	0.25	0.25	0.25	0.25
Aceleración (S)	1.23	1.23	1.23	1.23
* P es la fracción del algoritmo total que se puede paralelizar				

Tabla 10 Índice de aceleración teórico para el algoritmo con OpenMP.

4.2.3.1. DESEMPEÑO DEL ALGORITMO EN LA VERSIÓN PARALELA OPENMP

Para tener un contraste entre la implementación serial del algoritmo *UPSO* y la versión paralela con OpenMP*, se cuenta con dos equipos de cómputo denominados PC1 y PC2 con características diferentes especificadas en la Tabla 11.

El modelo paralelo OpenMP se desarrolló a partir del código serial, al cual se agregaron las directivas correspondientes al modelo de programación, sobre procesos iterativos del método *UPSO* que no posean dependencia de datos.

Convención	Clasificación	Equipo	Procesadores [Unidades]	Cantidad de threads (total)	Arquitectura
PC1	Computador personal	Samsung®, NP-Q330 JS03CO, Intel® Core™ i5 M460 @2.53GHz	2	4	64 - bit
PC2	HPC	Proliant SL390*G7, Intel Xeon CPU E5640, 2.67 GHz	4	8	64 - bit

Tabla 11 características de cada una de las arquitecturas. [Basada en (44), (45)] [Autores]

Para el análisis de desempeño del algoritmo UPSO, se plantea un índice de aceleración que relaciona el tiempo de ejecución de la versión serial frente a la versión paralela OpenMP sobre un mismo equipo.

En la Tabla 12, se muestra el tiempo de computo que requiere cada equipo al ejecutar el algoritmo UPSO en las dos versiones, para solucionar el modelo circuital con 1, 3, 6, 12 y 15 celdas respectivamente.

Según la Tabla 12, el PC1 presenta su máximo índice de aceleración en 3.16 al resolver el modelo circuital para 6 celdas (7 variables). Esto evidencia la reducción

* La versión paralela por procesadores desarrollada hace uso de la API *OpenMP* 3.1.

en el tiempo de ejecución en virtud de la distribución de trabajo que realizan las directivas OpenMP, sobre los 4 hilos disponibles en la arquitectura del PC1.

Equipo de Computo	Versión de Programa	Unidad	DIMENSIÓN				
			1 Celda	3 Celdas	6 Celdas	12 Celdas	15 celdas
NP- Q330-JS03CO Intel® Core™ i5 M460 @2.53GHz	Serial PC1	Tiempo [s]	0.013	0.665	11.492	167.23	511.7536
	OpenMP PC1	Tiempo [s]	0.016	0.252	3.633	58.29	184.22
	Indice PC1 Aceleracion S/P	[Adimensional]	0.81	2.64	3.16	2.87	2.78
Proliant SL390*G7, 2.67 GHz Intel Xeon CPU E5640 Nodo guane02	Serial PC2	Tiempo [s]	0.013	0.508	8.734	127.61	310.875
	OpenMP PC2	Tiempo [s]	0.024	3.197	5.712	49.08	118.88
	Indice PC2 Aceleracion S/P	[Adimensional]	0.54	0.16	1.53	2.60	2.62

Tabla 12 Estimación de la mejora en desempeño ofrecida por el modelo paralelo OpenMP en algoritmo UPSO, basado en el tiempo de ejecución, sobre cada arquitectura. [Autores]

La solución del modelo para una celda en el PC1, presenta un índice de aceleración menor a uno, que indica que la ejecución del algoritmo paralelo OpenMP emplea más tiempo que su equivalente serial; lo cual puede atribuirse a que el equipo emplea más tiempo en la comunicación entre hilos, que en el cálculo de las soluciones para un problema de pocas dimensiones.

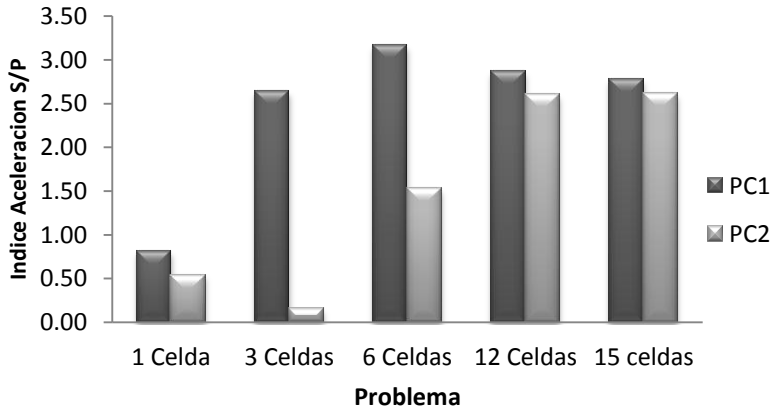


Figura 24 Aceleración en cada una de las arquitecturas. [Autores]

Por su parte, el desempeño sobre la arquitectura PC2 presenta un índice de aceleración menor a uno para el modelo con 1 y 3 celdas, y que igualmente se asocia al costo de comunicación entre hilos como se mencionó para el PC1.
En la Figura 24, se aprecia con claridad que para todas las dimensiones, el índice de aceleración es mayor en el PC1, debido a que el tiempo de cómputo es mucho mayor para la ejecución del algoritmo serial frente al PC2.

El número de *threads* disponibles en el PC2 es mayor, lo cual implica que el nivel de desempeño para la versión paralela OpenMP sea mayor para este equipo, cuando se ejecuta el algoritmo en múltiples dimensiones. La arquitectura del PC2, permite distribuir porciones de cálculo más pequeñas para una mayor cantidad de unidades de procesamiento, lo cual se evidencia en la diferencia de tiempos presentados en la Figura 25 .

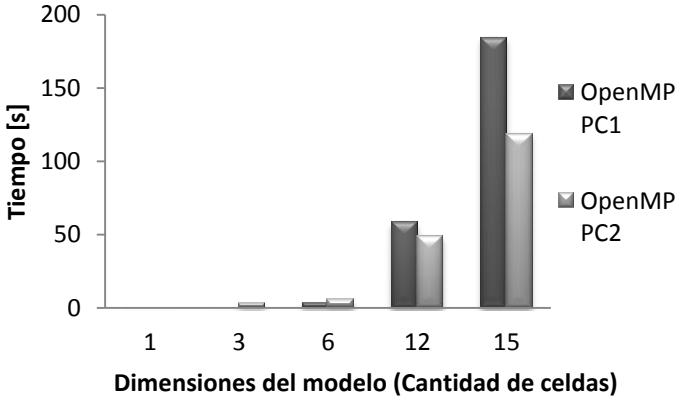


Figura 25 Tiempo de Cómputo versión OpenMP. [Autores]

Cabe resaltar que a medida que se agregan dimensiones al modelo, el índice de aceleración va en decremento para la arquitectura del PC1, como se aprecia en la ejecución del modelo con 12 celdas, con índice de 2.87. Por otra parte, el índice de aceleración sobre el PC2 aumenta después de 3 celdas; ya que dispone de

una mayor cantidad de *threads* para la ejecución de los procesos con mayor número de iteraciones.

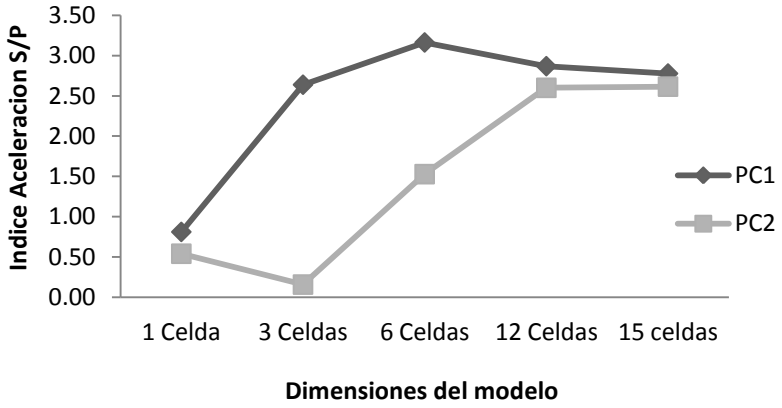


Figura 26 Crecimiento marcado en el desempeño de Xeon. [Autores.]

En la Figura 26, se observa la tendencia del índice de aceleración a un valor de 2.5 para los dos equipos. Lo que indica que el algoritmo reduce su tiempo de cómputo en un 60%, cuando se implementan las directivas de OpenMP para múltiples dimensiones.

Dado que la solución de un problema de múltiples dimensiones, involucra una mayor cantidad de información (partículas), el PC1 evidencia una carga computacional mayor comparado con el PC2. El PC1 debe distribuir la misma cantidad de cálculos sobre la mitad de unidades de procesamiento, lo cual implica colas de *threads* para la ejecución paralela de los datos.

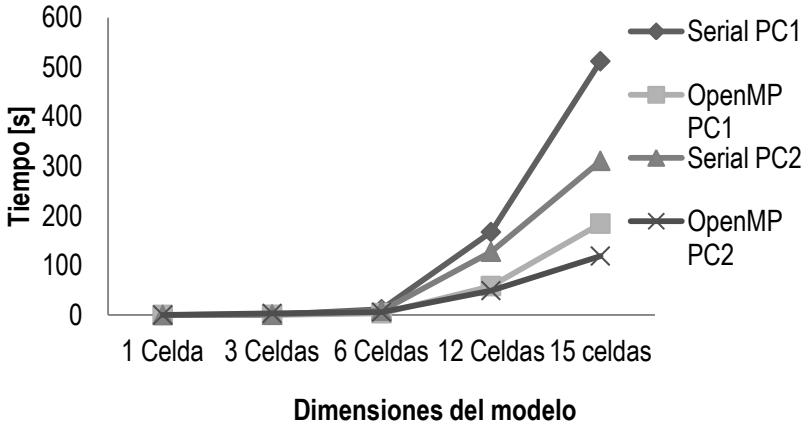


Figura 27 Tiempo de ejecución para las versiones serial y paralela con OpenMP. [Autores]

Por su parte la arquitectura del PC2, dispone de más hilos de procesamiento paralelo de datos, lo que hace visible el marcado crecimiento en el desempeño de la arquitectura Xeon sobre la corei5, para la solución del modelo a partir de 6 celdas, como se aprecia en la Figura 27.

Producto de lo anterior, es claro que el nodo guane02 es una arquitectura hibrida enfocada al manejo masivo de datos, mientras que Core i5 responde a las especificaciones de una máquina de propósito general, enfocada a la ejecución de múltiples tareas.

4.2.3.2. DESEMPEÑO DEL ALGORITMO EN LA VERSIÓN CUDA

En cuanto al desarrollo del algoritmo *UPSO* sobre una arquitectura hibrida en la que se usa conjuntamente *CPUs* y *GPUs*, por medio del modelo de programación paralelo CUDA, se hace necesario disponer de un equipo con tarjeta gráfica Nvidia y el toolkit, para hacer posible el desarrollo de aplicaciones *CUDA*.

El PC2 mencionado en las pruebas de OpenMP cumple con estas características. En la Tabla 13, se presentan las especificaciones de hardware sobre el nodo guane02.

Tarjeta grafica	Hillos por Bloque	Hilos por Grid	Grids disponibles
GPU Computing NVIDIA® Tesla™ M2050 , 1.15 GHz	1024	65535	3
	Memoria compartida por bloque [Bytes]	Memoria global (total) [Bytes]	
	49152	2817982464	

Tabla 13 Características de la tarjeta gráfica utilizada. [Autores]

Las pruebas realizadas para el algoritmo *CUDA,* se contrastan con los resultados obtenidos con la versión serial, a fin de conocer el desempeño que ofrece este modelo de programación.

En la Tabla 14, se presenta el resultado de los tiempos requeridos por el algoritmo serial y la versión *CUDA,* en la solución del modelo para 1, 3, 6 y 12 celdas respectivamente.

Equipo de Computo	Versión de Programa	Unidad	DIMENSIÓN					
			1 Celda	3 Celdas	6 Celdas	12 Celdas	24 Celdas	30 Celdas
Proliant SL390*G7, 2.67 GHz Intel Xeon CPU E5640 Nodo guane02	Serial	Tiempo [s]	0.01	0.51	8.73	127.61	-	-
NVIDIA® Tesla™ M2050 , 1.15 GHz	CUDA	Tiempo [s]	1.03	1.20	1.42	1.86	16.97	20.31
	Indice Aceleracion S/P	[Adimensional]	0.01	0.43	6.16	68.49	-	-

Tabla 14 Tiempo de ejecución requeridos por el algoritmo en serial y CUDA. [Autores]

Adicionalmente se muestran los tiempos para 24 y 30 celdas sólo para la versión paralela, visto como un logro adicional en el cual se haya el límite del correcto funcionamiento para el algoritmo *CUDA*. Por efectos prácticos no se realiza la comparación con el algoritmo serial, ya que su ejecución para estas dimensiones tardaría demasiado tiempo.

Para medir el desempeño del algoritmo desarrollado en *CUDA*, se plantea un índice de aceleración similar al usado con la versión paralela *OpenMP*, y que relaciona el tiempo de ejecución de las dos versiones.

El algoritmo *CUDA* presenta un índice de aceleración de 0.01 y 0.43, para el caso de 1 y 3 celdas respectivamente; lo que indica que el tiempo de cómputo es mayor para la versión paralela con respecto a la versión serial. Esto se debe a que la arquitectura *CUDA*, está pensada para desarrollar aplicaciones masivamente paralelas; por ello el algoritmo presenta poca eficiencia, al dar solución al modelo con pocas celdas. Como se aprecia en la Tabla 14.

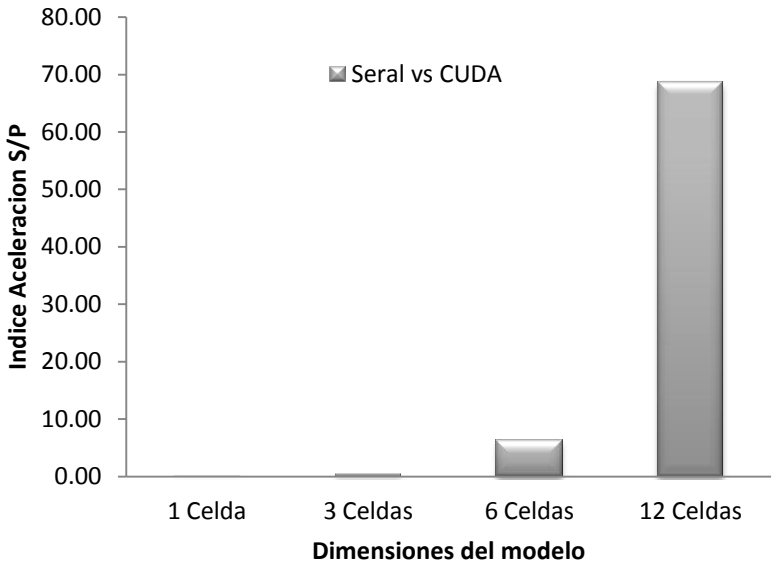


Figura 28 Índice de aceleración del modelo CUDA frente a la versión serial. [Autores]

El algoritmo *CUDA*, presenta un alto índice de aceleración para dimensiones arriba de 6 celdas. En el caso particular de 12 celdas, el índice de aceleración es cercano a 70 como se muestra en la Figura 28, que refleja la reducción en el tiempo de computo de 127.6 s a 1.86 s; equivalente al 1.5% del tiempo requerido por el algoritmo en la versión serial.

La reducción en el tiempo de ejecución se debe a que *CUDA*, permite hacer uso de las múltiples unidades aritméticas disponibles en la tarjeta gráfica; asumiendo la independencia en los datos entre hilos.

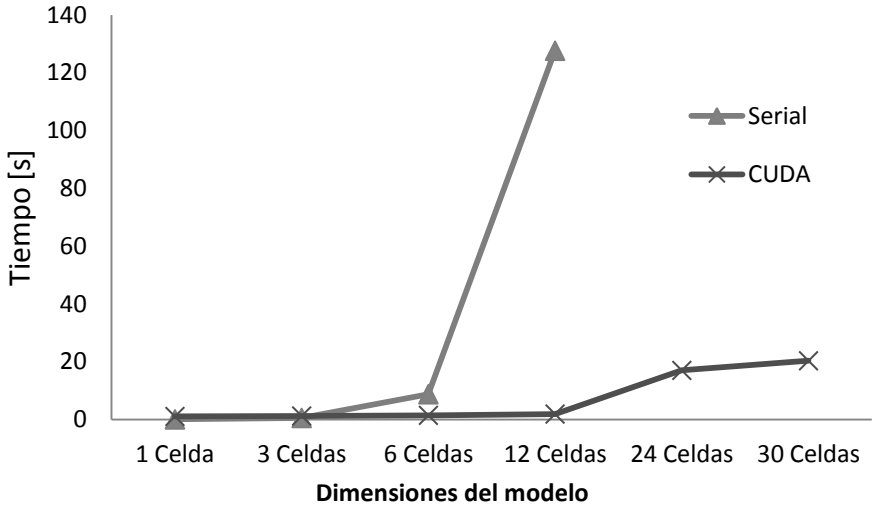


Figura 29 Tiempos de cómputo del algoritmo serial y la versión en CUDA

Sobre la Figura 29, se observa que el tiempo de cómputo en el algoritmo *CUDA*, posee una variación menos significativa frente a la versión serial a medida que se incrementa la complejidad del modelo, debido a que la metodología de programación *CUDA*, permite tener mayor granularidad frente al modelo *OpenMP*, gracias a que esta arquitectura dispone de una mayor cantidad de unidades de procesamiento; que se traduce en un mayor número de hilos, evitando así las colas de datos presentes en el modelo *OpenMP*.

5. CONCLUSIONES

Como producto del desarrollado en esta tesis, se tiene un estudio de la técnica metaheurística *UPSO*, que sirve como base para la simulación de circuitos no lineales, tanto en el modelo de programación serial como para su equivalente en ejecución paralela.

Se concluye que la solución numérica de circuitos no lineales en condiciones DC, mediante la técnica metaheurística *UPSO*; es viable tanto en arquitecturas convencionales como hibridas.

Teniendo en cuenta que el modelo circuital empleado para la evaluación de UPSO, presenta un nivel de precisión del orden de mV, al momento de hallar el voltaje sobre la resistencia Rs en el modelo del panel solar.

Las conclusiones derivadas del logro del proyecto, son las siguientes:

- Se aprecia que las soluciones obtenidas mediante el algoritmo, presentan un nivel de error constante con respecto al valor teórico de referencia; lo cual hace ver, cuán preciso es UPSO al dar solución a un modelo de celdas solares, teniendo en cuenta que se trata de un método de optimización estadístico, basado en resultados experimentales.

- Producto de la observación en el nivel de precisión de *UPSO*, se tiene como argumento que el factor de unificación utilizado para el algoritmo, es del orden de 1e-3, lo cual indica una mayor prioridad hacia la de búsqueda local que global, como se mencionó en la Ec. 5.

- Se puede considerar que el modelado de celdas solares paralelo, es un

problema de optimización enfocado a la búsqueda de soluciones en ámbito local. Por lo cual técnicas numéricas enfocadas a búsqueda local como *UPSO* son apropiadas, para la solución del circuito no lineal de celdas solares en condiciones DC.

- Se considera necesario el conocimiento del circuito y la técnica *UPSO* por parte del usuario, para hacer correcto uso de la técnica; a fin de asignar valores apropiados a los coeficientes y límites en el espacio de búsqueda. En el caso específico del modelado de celdas solares, es claro que el voltaje de salida en la resistencia RL, es un valor positivo con respecto al nodo de referencia; por ello el límite inferior del espacio de búsqueda es cero (0); y acorde con la interpretación eléctrica del problema.

- Cabe mencionar que el desarrollo de una aplicación CUDA, implica el uso de una metodología de programación hibrida; basada en hardware y software a la vez, que requiere de un conocimiento detallado tanto del modelo de programación como de la arquitectura a utilizar.

- Producto de lo anterior, se tiene que el 50% del tiempo empleado para el desarrollo de esta tesis, fue empleado en la creación del algoritmo en este lenguaje.

Adicionalmente se presenta un listado de recomendaciones para trabajos futuros tanto en el área de simulación no lineal mediante *UPSO* como sobre la ejecución en plataformas paralelas.

6. RECOMENDACIONES

- El fruto de este proyecto sirve como punto de partida a trabajos futuros en el área de modelado y simulación de circuitos electrónicos haciendo uso de técnicas de optimización numérica.

- Promover el uso de los sistemas de supercomputación para la solución de problemas de ingeniería, para ir más allá de las limitantes de procesamiento de los sistemas de cómputo convencionales.

- Como resultado del elevado tiempo de desarrollo empleado para la versión *CUDA* de *UPSO*, se considera importante la inclusión de asignaturas electivas a nivel de ingeniería, que permitan explorar este campo de aplicación, para quienes presentan interés por este tipo de arquitecturas y lenguajes de programación.

- Este trabajo de investigación puede ser la base para el desarrollo de una herramienta computacional, que aporte al estudio e investigación de la generación de energías limpias; ya sea para el diseño de sistemas fotovoltaicos o el cálculo de las condiciones de operación, para el máximo provecho de la energía proveniente de la radiación solar.

BIBLIOGRAFÍA

1. **Zomaya, Albert Y.** *PARALLEL COMPUTING FOR BIOINFORMATICS AND COMPUTATIONAL AND COMPUTATIONAL.* [ed.] INC JOHN WILEY & SONS. s.l. : The University of Sydney, Australia. ISBN-13 978-0-471-71848-2.

2. **Talukder, Satyobroto.** *Mathematical Modelling and Applications of Particle Swarm Optimization.* s.l. : Blekinge Institute of Technology, February 2011. pág. 65, Master Thesis .

3. *Time-based metamodeling technique for vehicle crashworthiness optimization.* **Hu Wang, G.Y. Li , Enying Li.** s.l. : Elsevier, 12 de April de 2010, Computer Methods in Applied Mechanics and Engineering, págs. 2497 - 2509. 0045-7825.

4. *Induction Generator Model Parameter Estimation using Improved Particle Swarm Optimization and On-Line Response to a Change in Frequency.* **P. Regulski, F. González-Longatt, E, P. Wall, and V. Terzija.** [ed.] IEEE. 2011, IEEE. 978-1-4577-1002-5.

5. *Evaluation the Accuracy of One-Diode and Two-Diode Models for a Solar Panel Based Open-Air Climate Measurements.* **Mohsen Taherbaneh, Gholamreza Farahani and Karim Rahmani.** [ed.] Leonid A. Kosyachenko. s.l. : www.intechopen.com, 02 de November de 2011, Solar Cells - Silicon Wafer-Based Technologies, págs. 201 - 228. 978-953-307-747-5.

6. **Paolo Nenzi, Holger Vogt,.** Ngspice Users Manual. [En línea] June de 2013. [Citado el: 20 de julio de 2013.]

7. **Nagel, Laurence W.** *SPICE: An Computer Program to Simulate Semiconductor Circuits.* [ed.] Electronic Research Laboratory. Berkeiey : University of Calofornia, 1975. págs. pp. 114 - 160. Vol. Vol. 2.

8. **Pupecki, Donald.** *A Fully Saturated OpenCL Particle Swarm Optimizer.* NY, USA : Computer Science Department, SUNYIT, 2011.

9. *Efficient Population Utilization Strategy for Particle Swarm Optimizer.* **Sheng-Ta Hsieh, Tsung-Ying Sun, Chan-Cheng Liu, and Shang-Jeng Tsai.** N° 2, APRIL de 2009, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART B: CYBERNETICS, Vol. 39, págs. pp. 444 - 456.

10. **Konstantinos E. Parsopoulos, Michael N. Vrahatis.** [ed.] Greece University of Patras. *Multi-Objective Particles Swarm Optimization Approaches.* University of Patras, Greece : s.n., 2008, Chapter II.

11. *A new optimizer using particle swarm theory.* **J. Kennedy, R. C. Eberhart .** Nagoya, Japan : s.n., 1995, Proc. 6th Int. Symp. Micro Mach. Human Sci, págs. pp. 39–43.

12. *The fully informed particle swarm: Simpler, maybe better.* **Mendes, R., Kennedy, J., Neves.** 2004, IEEE Trans. Evol. Comput., Vol. Vol. V.8, págs. pp. 204 – 210.
13. *UPSO: A unified particle swarm optimization scheme.* **Parsopoulos, K.E., Vrahatis, M.N.** Zeist, Netherlands : VSP International Science Publishers, ICCMSE 2004, Lecture Series on Computer and Computational Sciences, Vol. Vol. 1, págs. pp 868 – 873. Proc. Int. Conf. Comput. Meth. Sci. Engin.
14. *Defining a Standard for Particle Swarm Optimization.* **Daniel Bratton, James Kennedy.** 1-4244-0708-7, s.l. : Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007), 2007.
15. *A High Efficient Micro-controlled Buck Converter with Maximum Power Point Tracking for Photovoltaic Systems.* **P. C. M. Bernardo1, Z. M. A. Peixoto1 and L.V. B. Machado Neto2.** Valencia, España : s.n., 15th to 17th April, 2009.
16. *MODELAMIENTO Y SIMULACION DE UN CARGADOR DE BATERIA PARA UN SEGUIDOR DE MAXIMA TRANSFERENCIA DE POTENCIA.* **Antenor Aliaga Zegarra, Ing. Cesar Humberto Estrada Crisanto.** Puno, : Universidad Nacional de Piura, Departamento de Ingeniería Electrónica, Nov. de 2012, XIX Simposio Peruano de Energía Solar y del Ambiente (XIX- SPES), 2012, págs. 12 -17.
17. Club de Investigación Tecnologica. [En línea] [Citado el: 12 de 9 de 2013.] http://www.clubdeinvestigacion.com/usuarios/articles/sistemas-escalables.html.
18. **GRID-UIS, SC3UIS.** Supercomputación y Calculo Cientifico UIS. *SC3UIS.* [En línea] [Citado el: 25 de Octubre de 2012.] http://grid.uis.edu.co/index.php/Infraestructura#Arquitectura_Plataforma.2FInfraestructura_de_Computaci.C3.B3n_de_Alto_Rendimiento.
19. *"Solution of the Mathematical Model of a Nonlinear Direct Current Circuit Using Particle Swarm Optimization,".* **I. Amaya, J. Cruz, and R. Correa,.** no. 172 , s.l. : Revista Dyna, 2011., Vol. vol. 79, págs. pp. 77–84.
20. *Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems.* **Yamille del Valle, Ganesh Kumar Venayagamoorthy,Salman Mohagheghi, Jean-Carlos Hernandez, Ronald G. Harley.** NO. 2, s.l. : IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, , APRIL 2008, Vol. VOL. 12.
21. *A MODIFIED PARTICLE SWARM OPTIMIZATION TECHNIQUE FOR SOLVING IMPROVEMENT OF VOLTAGE STABILITY AND REDUCE POWER LOSSES USING UPFC.* **Kiran Kumar Kuthadi, M. Suresh Babu.** [ed.] IJERA. Issue 3, s.l. : www.ijera.com, May-Jun de 2012, International Journal of Engineering Research and Applications, Vol. Vol. 2, págs. pp. 1516-1521. ISSN: 2248-9622.

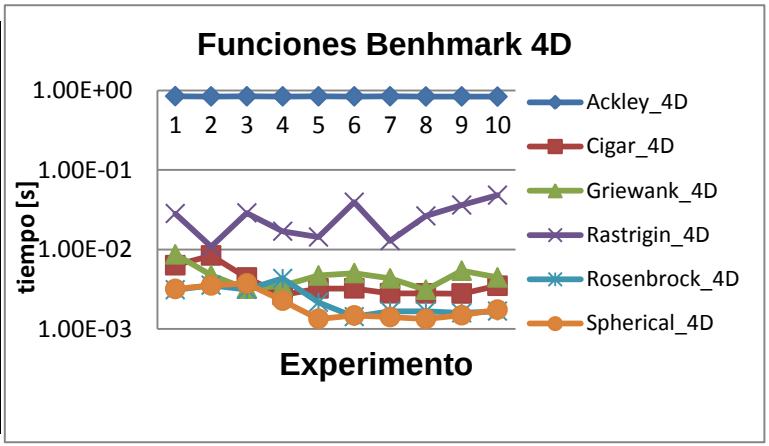
22. *Small signal parameter extraction of MESFET using quantum particle Small signal parameter extraction of MESFET using quantum particle.* **Samrat L. Sabat a, *, Siba K. Udgata b, K.P.N. Murthy a.** s.l. : Elsevier, (2010), Microelectronics Reliability, Vol. 50 , págs. 199–206.
23. *A Particle Swarm Optimization technique used for the improvement of analogue circuit performances.* **Mourad Fakhfakh, Yann Cooren, Mourad Loulou and Patrick Siarry.** [ed.] Aleksandar Lazinica. s.l. : www.intechopen.com, 2009, InTech Particle Swarm Optimization, págs. 170-192. ISBN 978-953-7619-48-0.
24. *Parameter extraction of solar photovoltaic modules using penalty-based.* **Kashif Ishaque a, b, Zainal Salam a, Saad Mekhilef c, Amir Shamsudin.** 99, s.l. : www.elsevier.com/locate/apenergy, 2012, Applied Energy, págs. 297 -308.
25. *Parameter extraction of solar cells using particle swarm optimization.* **Meiying Ye, Xiaodong Wang and Yousheng Xu.** 105, 4 de May de 2009, JOURNAL OF APPLIED PHYSICS . 0021-8979/2009/105.9.
26. *"Real Roots of Nonlinear Systems of Equations Through a Metaheuristic Algorithm".* **I. Amaya, J. Cruz, and R. Correa,.** s.l. : Revista Dyna, 2011, Revista Dyna, Vols. vol. 78,, págs. pp. 15–23.
27. **Rui, Mendes.** *Population Topologies and Their Infuence in Particle Swarm Performance.* 2004.
28. *Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem.* **Chen, Ruey-Maw.** 38, 2011, Expert Systems with Applications, págs. 7102–7111.
29. *High Performance and Scientific Computing.* **Carlos Jaime Barrios Hernandez, PhD.** Bucaramanga : http://sc3.uis.edu.co, 2013.
30. *On Performance Enhancement of Circuit Simulation Using Multithreaded Techniques.* **R.K. Perng, T.-H.Weng, and K.-C. Li:.** s.l. : IEEE International Conference on Computational Science and Engineering, 2009, págs. pp. 158-165 .
31. All About Circuits. [En línea] [Citado el: 2013 de Julio de 15.] http://www.allaboutcircuits.com/vol_5/chpt_7/3.html.
32. **GRID-.UIS.** [En línea] [Citado el: 15 de julio de 2013.] http://grid.uis.edu.co/index.php/Curso_oar-kadeploy_Talleres.
33. Wikipedia. *POO.* [En línea] [Citado el: 17 de 9 de 2013.] http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos.
34. www.openmp.org. *OpenMP Reference.* [En línea] [Citado el: 8 de 10 de 2013.] www.openmp.org/mp-documents/spec30.pdf.
35. OpenMP 3.1 API C/C++ Syntax Quick Reference Card. [En línea] [Citado el: 8 de 10 de 2013.] openmp.org/mp-documents/OpenMP3.1-CCard.pdf.

36. IBM public Lib. *OpenMPReference.* [En línea] [Citado el: 8 de 10 de 2013.] https://publib.boulder.ibm.com/infocenter/comphelp/v8v101/index.jsp?topic=/com.ibm.xlcpp8a.doc/compiler/ref/ruompplp.htm.
37. Docs Nvidia . [En línea] [Citado el: 16 de 10 de 2013.] http://docs.nvidia.com/cuda/index.html .
38. Developer Zone NVIDIA. [En línea] [Citado el: 17 de 9 de 2013.] https://developer.nvidia.com/category/zone/cuda-zone.
39. *CUDA C BEST PRACTICES GUIDE.* s.l. : www.nvidia.com, May 9, 2011. Vols. DG-05603-001_v4.0.
40. Funciones Benchmark Optimización. [En línea] http://infinity77.net/global_optimization/test_functions.html.
41. *A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems,.*
42. *Accurate MATLAB Simulink PV System Simulator Based on a Two-Diode Model.* **Kashif Ishaque, Zainal Salam, and Hamed Taheri.** Johor, Malaysia : Dept. of Energy Conversion, Faculty of Electrical Engineering, Universiti Teknologi Malaysia,.
43. *A single procedure for helping PV designers to select silicon PV module and evaluate the loss resistances.* **C. Carrero, J.Amador, and S. Arnaltes,.** No. 15, Dec. de 2007., Renewable Energy, Vol. Vol. 32, págs. 2579–2589.
44. Intel® Core™ i5-430M Processor (3M Cache, 2.26 GHz). [En línea] [Citado el: 9 de 10 de 2013.] http://ark.intel.com/products/43537.
45. Intel® Xeon® Processor E5640 (12M Cache, 2.66 GHz, 5.86 GT/s Intel® QPI). [En línea] [Citado el: 9 de 10 de 2013.] http://ark.intel.com/products/47923/.
46. *The particle swarm–explosion, stability, and convergence in a multidimensional complex space.* **Clerc, M., Kennedy, J.** 2002, IEEE Trans. Evol. Comput. 6(1) , págs. pp 58 – 73.
47. **Grid-UIS2.** Grid-UIS. [En línea] [Citado el: 29 de octubre de 2012.] http://grid.uis.edu.co/images/7/75/EspeC-2.pdf.
48. **William Javier Trigos Guevara, Angel Gabriel Meza Garcia.** Proyecto UPSO (Pág. Web). [En línea] 3 de 11 de 2012.
49. *Particle Swarm Optimization Methods, Taxonomy and Applications.* **Masehian, Davoud Sedighizadeh and Ellips.** No. 5, s.l. : International Journal of Computer Theory and Engineering, , December 2009, Vol. Vol. 1. 1793-8201.
50. **Kadeploy.** Kadeploy Official Web Site. [En línea] [Citado el: 15 de julio de 2013.] http://kadeploy3.gforge.inria.fr/.
51. CENTRAL SOLAR U.P.V. *Características del Módulo Fotovoltaico A-75 .* [En línea] [Citado el: 12 de 09 de 2013.] http://www.upv.es/ges/cen_sol/pan_sol/tecnicas.htm.

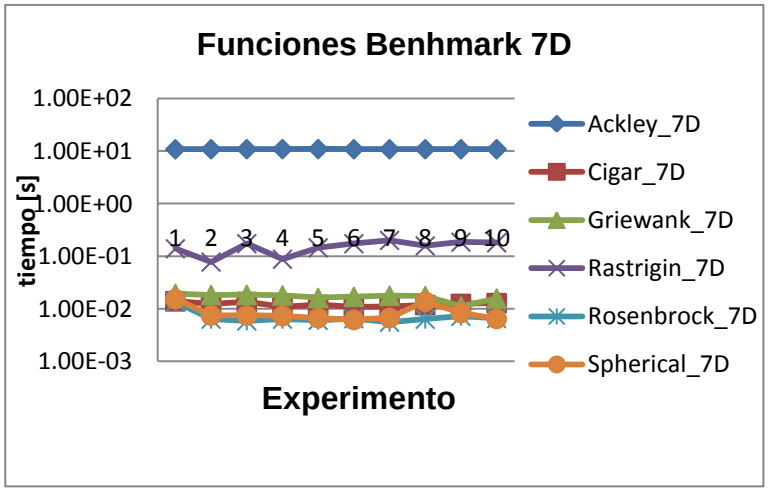
ANEXOS

ANEXO A RESULTADOS NUMÉRICOS DE LAS PRUEBAS PRELIMINARES CON FUNCIONES BENCHMARK

	Tiempo [seg]					
Experimento	**Ackley_4D**	**Cigar_4D**	**Griewank_4D**	**Rastrigin_4D**	**Rosenbrock_4D**	**Spherical_4D**
1	8,47E-01	6,32E-03	8,67E-03	2,83E-02	3,11E-03	3,18E-03
2	8,42E-01	8,39E-03	4,75E-03	1,08E-02	3,54E-03	3,52E-03
3	8,44E-01	4,43E-03	3,20E-03	2,88E-02	3,14E-03	3,73E-03
4	8,42E-01	2,67E-03	3,49E-03	1,69E-02	4,32E-03	2,27E-03
5	8,43E-01	3,23E-03	4,69E-03	1,43E-02	2,16E-03	1,34E-03
6	8,42E-01	3,23E-03	5,04E-03	3,93E-02	1,42E-03	1,48E-03
7	8,43E-01	2,80E-03	4,33E-03	1,29E-02	1,67E-03	1,41E-03
8	8,39E-01	2,80E-03	3,12E-03	2,63E-02	1,66E-03	1,34E-03
9	8,41E-01	2,79E-03	5,40E-03	3,61E-02	1,59E-03	1,49E-03
10	8,36E-01	3,49E-03	4,44E-03	4,81E-02	1,67E-03	1,75E-03
Tiempo prom.	8,42E-01	4,01E-03	4,71E-03	2,62E-02	2,43E-03	2,15E-03



	Tiempo [seg]					
Experimento	**Ackley_7D**	**Cigar_7D**	**Griewank_7D**	**Rastrigin_7D**	**Rosenbrock_7D**	**Spherical_7D**
1	1,09E+01	1,39E-02	1,93E-02	1,39E-01	1,35E-02	1,54E-02
2	1,09E+01	1,23E-02	1,83E-02	7,79E-02	6,35E-03	7,49E-03
3	1,10E+01	1,36E-02	1,85E-02	1,74E-01	5,90E-03	7,50E-03
4	1,10E+01	1,09E-02	1,80E-02	8,81E-02	6,35E-03	7,31E-03
5	1,10E+01	1,19E-02	1,62E-02	1,45E-01	6,05E-03	6,67E-03
6	1,09E+01	1,08E-02	1,70E-02	1,74E-01	6,50E-03	6,17E-03
7	1,09E+01	1,10E-02	1,78E-02	2,01E-01	5,57E-03	6,71E-03
8	1,09E+01	1,18E-02	1,75E-02	1,58E-01	6,33E-03	1,37E-02
9	1,09E+01	1,24E-02	1,11E-02	1,86E-01	7,21E-03	8,36E-03
10	1,09E+01	1,29E-02	1,57E-02	1,81E-01	6,52E-03	6,44E-03
Tiempo prom.	1,09E+01	1,21E-02	1,69E-02	1,52E-01	7,03E-03	8,57E-03



	Tiempo [seg]					
Experimento	**Ackley_13D**	**Cigar_13D**	**Griewank_13D**	**Rastrigin_13D**	**Rosenbrock_13D**	**Spherical_13D**
1	1,59E+02	5,63E-02	8,69E-02	9,15E-01	2,58E-02	4,18E-02
2	1,60E+02	5,09E-02	6,92E-02	3,42E-01	2,38E-02	4,49E-02
3	1,58E+02	5,05E-02	7,98E-02	3,84E-01	2,55E-02	3,81E-02
4	1,58E+02	4,87E-02	8,00E-02	7,60E-01	2,57E-02	3,13E-02
5	1,61E+02	5,24E-02	6,56E-02	8,02E-01	2,55E-02	3,07E-02
6	1,64E+02	5,01E-02	7,03E-02	9,50E-01	2,54E-02	3,08E-02
7	1,85E+02	5,21E-02	7,04E-02	7,90E-01	2,63E-02	2,90E-02
8	1,95E+02	5,08E-02	7,30E-02	1,09E+00	2,61E-02	3,02E-02
9	2,04E+02	4,68E-02	7,88E-02	5,58E-01	2,48E-02	2,53E-02
10	1,88E+02	4,87E-02	7,15E-02	1,04E+00	2,43E-02	2,83E-02
Tiempo prom.	1,73E+02	5,07E-02	7,45E-02	7,64E-01	2,53E-02	3,30E-02

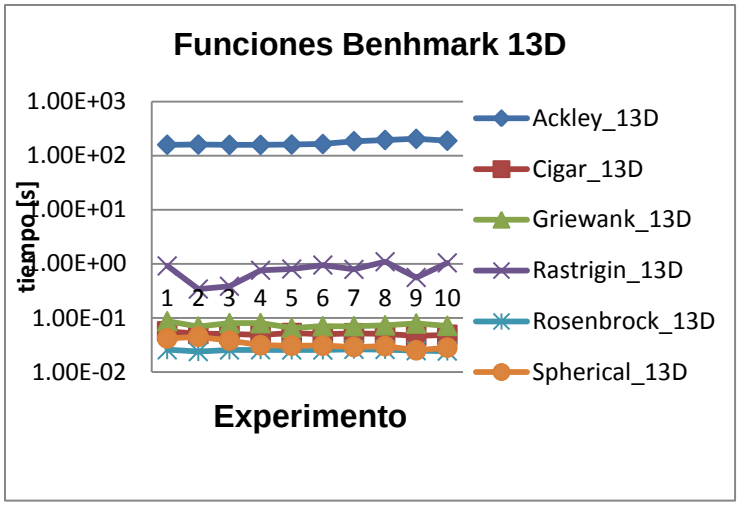


Tabla 15. Fitness en funciones Benchmark

Dimensión \ Función		Ackley	Cigar	Griewank	Rastrigin	Rosenbrock	Sphere
4D	Fitness Prom.	2.03E-03	1.67E-03	2.13E-03	2.52E-03	1.95E-03	1.63E-03
	Std_Desv	1.02E-03	8.29E-04	8.49E-04	1.18E-03	1.17E-03	1.17E-03
7D	Fitness Prom.	1.90E-03	1.59E-03	1.88E-03	2.99E-03	2.20E-03	2.63E-03
	Std_Desv	9.92E-04	8.90E-04	1.16E-03	7.44E-04	9.47E-04	8.63E-04
13D	Fitness Prom.	2.48E-03	2.93E-03	2.07E-03	3.19E-03	2.83E-03	2.69E-03
	Std_Desv	1.18E-03	8.87E-04	1.25E-03	4.09E-04	8.87E-04	1.04E-03

ANEXO B RESULTADOS PARA PRUEBAS CÓDIGO SERIAL

ANEXO B.1 EJECUCIÓN SERIAL 1 CELDA SOLAR

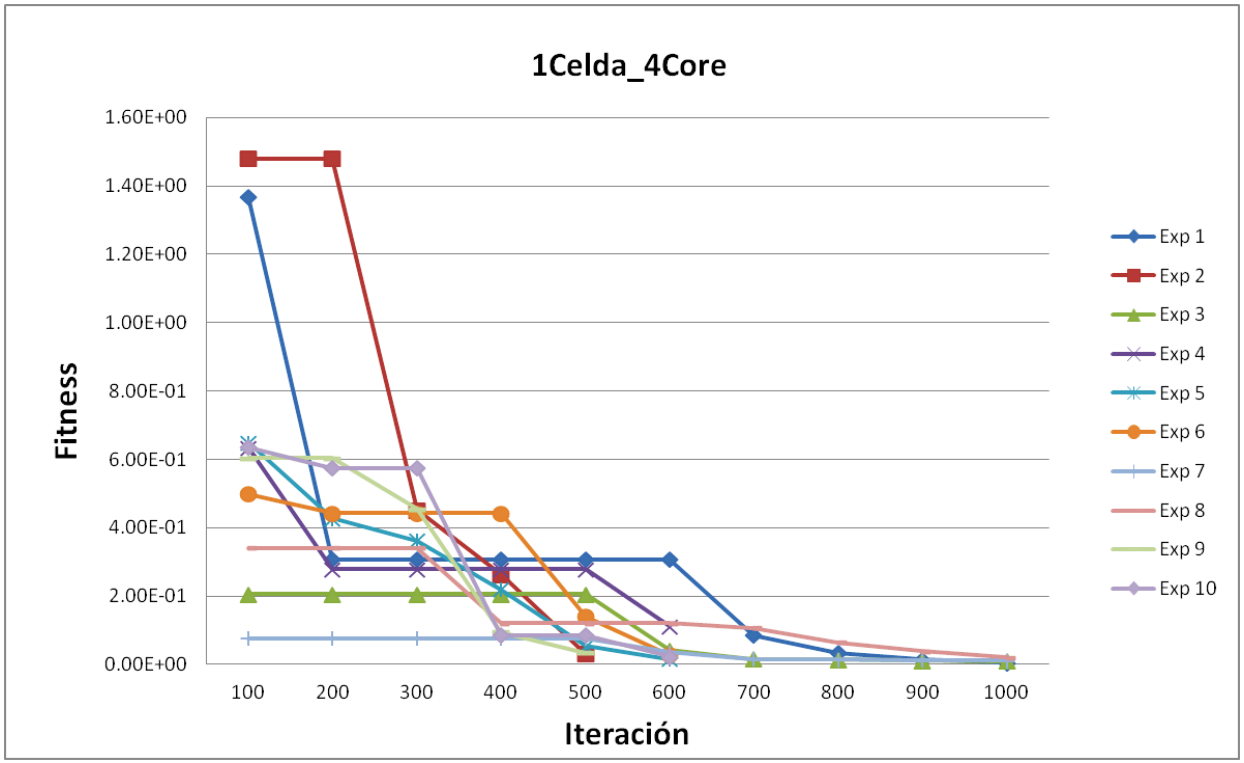


Figura 30 Convergencia del algoritmo para 1 Celda en la versión serial.

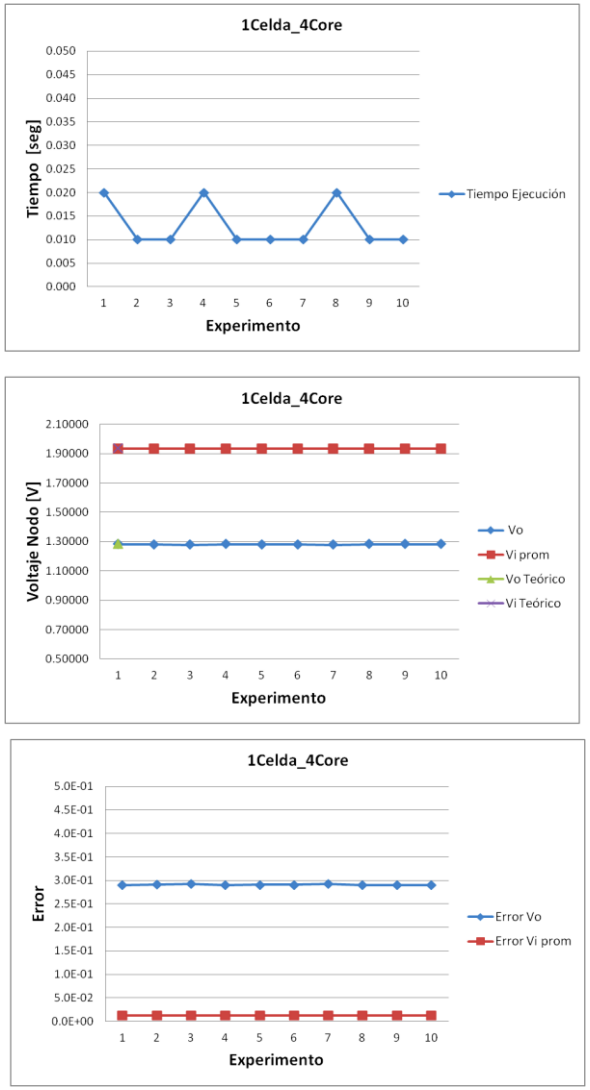


Figura 31 Tiempo de Ejecución, Voltajes y Error para ejecución con 1 Celdas en la versión serial.

Tabla 16. Tiempo de Ejecución, Voltajes, Fitness y Error para ejecución con 1 Celda en la versión serial.

1 celda (2 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje
1	V[0]	1.28E+00	0.020	20	3.28E-03	1.28195	2.9E-01
	V[1]	1.93E+00				1.93421	1.3E-02
2	V[0]	1.28E+00	0.010	10	3.91E-03	1.27966	2.9E-01
	V[1]	1.93E+00				1.93417	1.3E-02
3	V[0]	1.28E+00	0.010	10	8.83E-03	1.27785	2.9E-01
	V[1]	1.93E+00				1.93405	1.3E-02
4	V[0]	1.28E+00	0.020	20	1.03E-03	1.28128	2.9E-01
	V[1]	1.93E+00				1.93421	1.3E-02
5	V[0]	1.28E+00	0.010	10	3.39E-03	1.28050	2.9E-01
	V[1]	1.93E+00				1.93425	1.3E-02
6	V[0]	1.28E+00	0.010	10	2.60E-03	1.28066	2.9E-01
	V[1]	1.93E+00				1.93412	1.3E-02
7	V[0]	1.28E+00	0.010	10	9.79E-03	1.27752	2.9E-01
	V[1]	1.93E+00				1.93404	1.3E-02
8	V[0]	1.28E+00	0.020	20	4.83E-03	1.28251	2.9E-01
	V[1]	1.93E+00				1.93431	1.3E-02
9	V[0]	1.28E+00	0.010	10	3.99E-03	1.28207	2.9E-01
	V[1]	1.93E+00				1.93431	1.3E-02
10	V[0]	1.28E+00	0.010	10	2.66E-03	1.28165	2.9E-01
	V[1]	1.93E+00				1.93419	1.3E-02

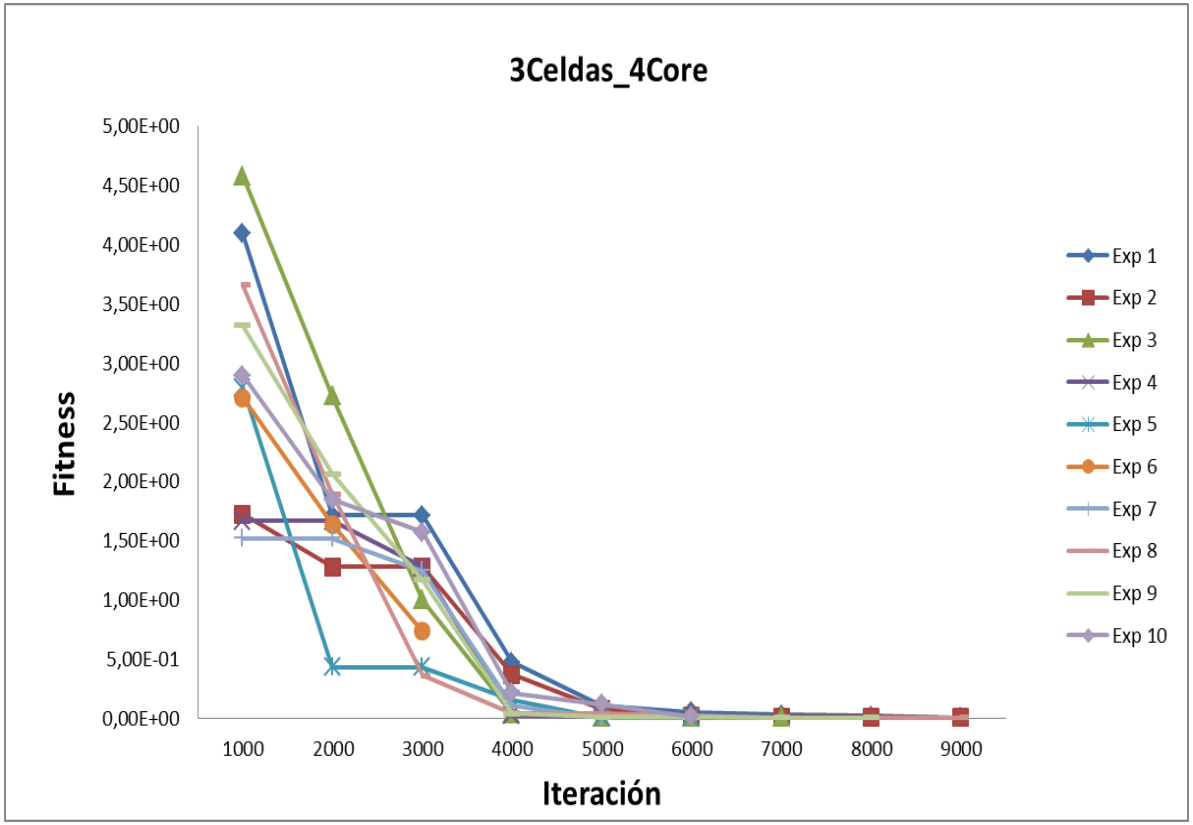


Figura 33 Convergencia del algoritmo para 3 Celdas en la versión serial

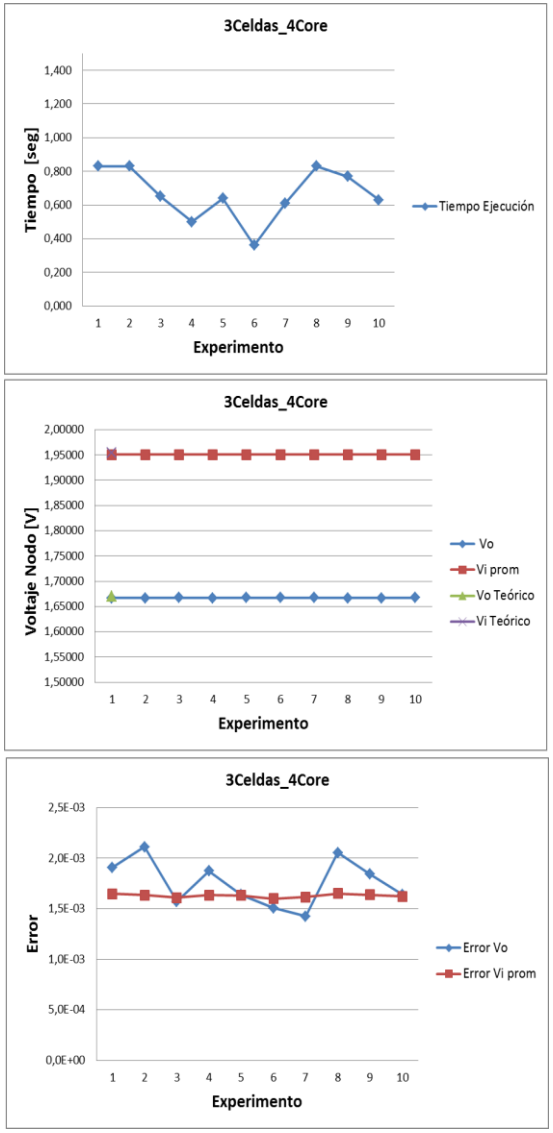


Figura 32 Tiempo de Ejecución, Voltajes y Error para ejecución con 3 Celdas en la versión serial.

Tabla 17. Tiempo de Ejecución, Voltajes y Error para ejecución con 3 Celdas en la versión serial.

3 celdas (4 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje
1	V[0]	1.67E+00	0.830	830	4.79E-03	1.66669	1.9E-03
	V[1]	1.95E+00				1.95054	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
2	V[0]	1.67E+00	0.830	830	7.26E-03	1.66635	2.1E-03
	V[1]	1.95E+00				1.95056	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
3	V[0]	1.67E+00	0.650	650	3.46E-03	1.66726	1.6E-03
	V[1]	1.95E+00				1.95061	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
4	V[0]	1.67E+00	0.500	500	3.07E-03	1.66675	1.9E-03
	V[1]	1.95E+00				1.95056	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
5	V[0]	1.67E+00	0.640	640	3.81E-03	1.66714	1.6E-03
	V[1]	1.95E+00				1.95057	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
6	V[0]	1.67E+00	0.360	360	3.93E-03	1.66736	1.5E-03
	V[1]	1.95E+00				1.95063	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
7	V[0]	1.67E+00	0.610	610	2.89E-03	1.66750	1.4E-03
	V[1]	1.95E+00				1.95060	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
8	V[0]	1.67E+00	0.830	830	5.30E-03	1.66645	2.1E-03
	V[1]	1.95E+00				1.95054	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
9	V[0]	1.67E+00	0.770	770	3.74E-03	1.66681	1.8E-03
	V[1]	1.95E+00				1.95056	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
10	V[0]	1.67E+00	0.630	630	3.73E-03	1.66714	1.6E-03
	V[1]	1.95E+00				1.95059	1.6E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					

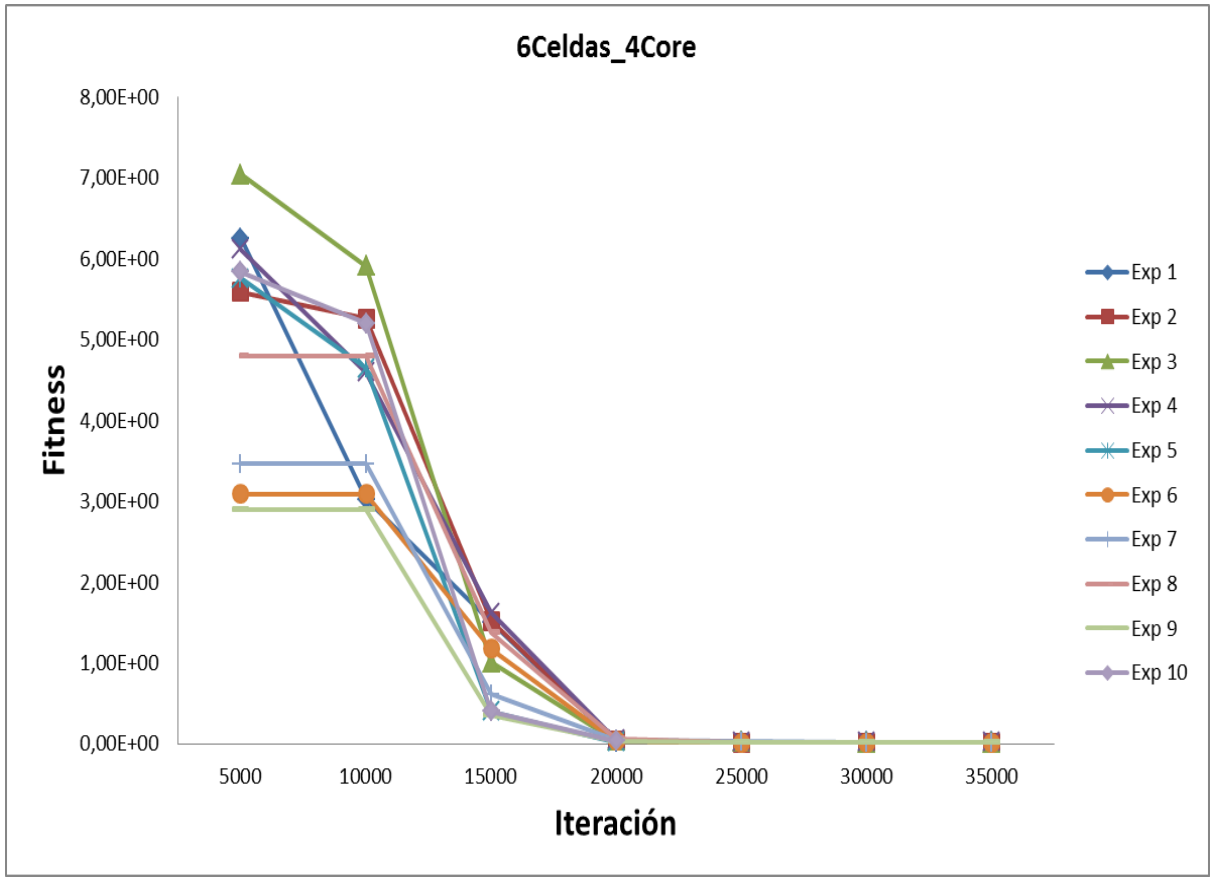


Figura 34 Convergencia del algoritmo para 6 Celdas en la versión serial.

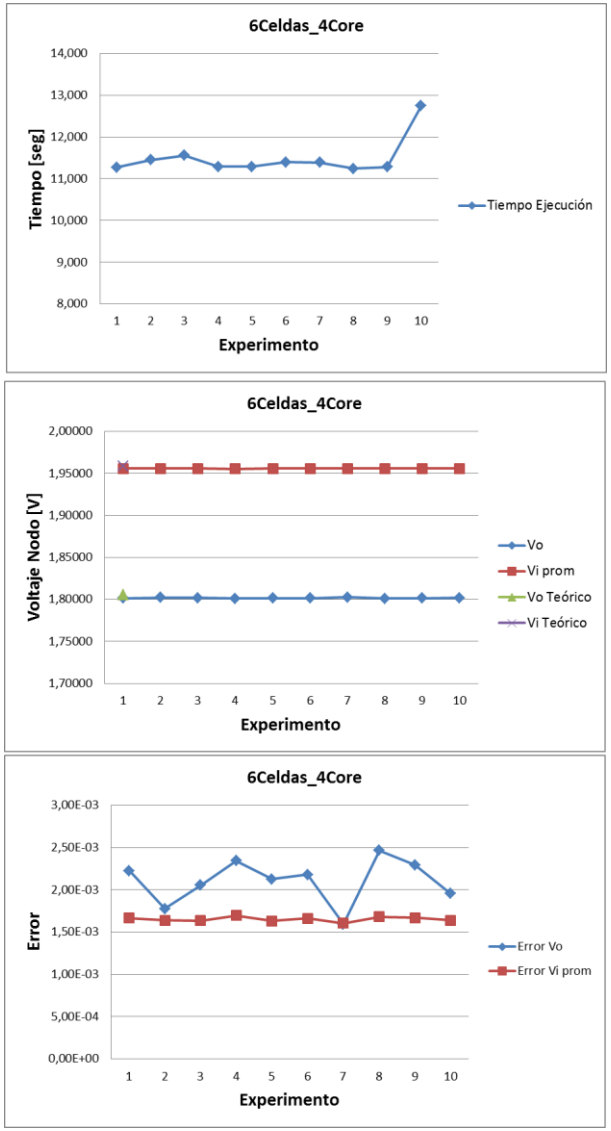


Figura 35 Tiempo de Ejecución, Voltajes y Error para ejecución con 6 Celdas en la versión serial.

Tabla 18 Tiempo de Ejecución, Voltajes y Error para ejecución con 6 Celdas en la versión serial.

6 celdas (7 dimensiones)																
Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje		Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje
1	V[0]	1.8014	11.270	11270	1.83E-02	1.80139	2.23E-03		6	V[0]	1.8015	11.400	11400	1.37E-02	1.80147	2.18E-03
	V[1]	1.9556				1.95561	1.67E-03			V[1]	1.9556				1.95562	1.66E-03
	V[2]	1.9557								V[2]	1.9556					
	V[3]	1.9554								V[3]	1.9556					
	V[4]	1.9556								V[4]	1.9557					
	V[5]	1.9557								V[5]	1.9556					
	V[6]	1.9557								V[6]	1.9556					
2	V[0]	1.8022	11.450	11450	8.39E-03	1.80221	1.77E-03		7	V[0]	1.8025	11.390	11390	9.75E-03	1.80254	1.59E-03
	V[1]	1.9557				1.95566	1.64E-03			V[1]	1.9556				1.95573	1.60E-03
	V[2]	1.9556								V[2]	1.9558					
	V[3]	1.9557								V[3]	1.9557					
	V[4]	1.9556								V[4]	1.9558					
	V[5]	1.9556								V[5]	1.9558					
	V[6]	1.9557								V[6]	1.9557					
3	V[0]	1.8017	11.560	11560	1.40E-02	1.80170	2.05E-03		8	V[0]	1.8010	11.240	11240	2.16E-02	1.80096	2.46E-03
	V[1]	1.9556				1.95567	1.64E-03			V[1]	1.9557				1.95558	1.68E-03
	V[2]	1.9556								V[2]	1.9556					
	V[3]	1.9557								V[3]	1.9556					
	V[4]	1.9558								V[4]	1.9555					
	V[5]	1.9555								V[5]	1.9556					
	V[6]	1.9557								V[6]	1.9555					
4	V[0]	1.8012	11.290	11290	2.13E-02	1.80118	2.34E-03		9	V[0]	1.8013	11.280	11280	1.93E-02	1.80127	2.29E-03
	V[1]	1.9555				1.95555	1.70E-03			V[1]	1.9554				1.95560	1.67E-03
	V[2]	1.9557								V[2]	1.9556					
	V[3]	1.9555								V[3]	1.9557					
	V[4]	1.9555								V[4]	1.9556					
	V[5]	1.9556								V[5]	1.9555					
	V[6]	1.9555								V[6]	1.9556					
5	V[0]	1.8016	11.290	11290	1.53E-02	1.80157	2.13E-03		10	V[0]	1.8019	12.750	12750	1.33E-02	1.80188	1.95E-03
	V[1]	1.9557				1.95567	1.63E-03			V[1]	1.9557				1.95566	1.64E-03
	V[2]	1.9556								V[2]	1.9555					
	V[3]	1.9557								V[3]	1.9558					
	V[4]	1.9556								V[4]	1.9557					
	V[5]	1.9558								V[5]	1.9557					
	V[6]	1.9557								V[6]	1.9556					

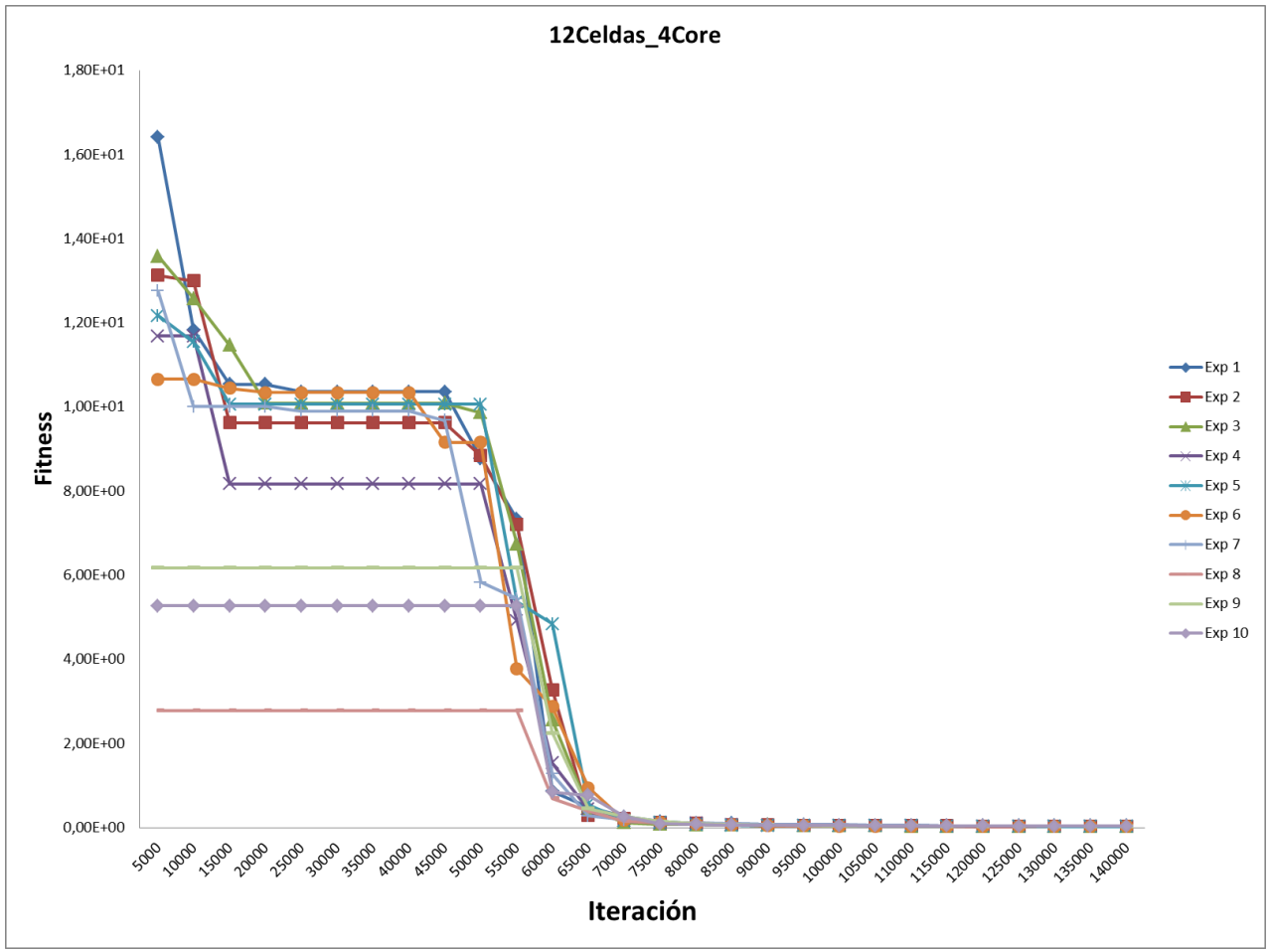


Figura 36 Convergencia del algoritmo para 12 Celdas en la versión serial.

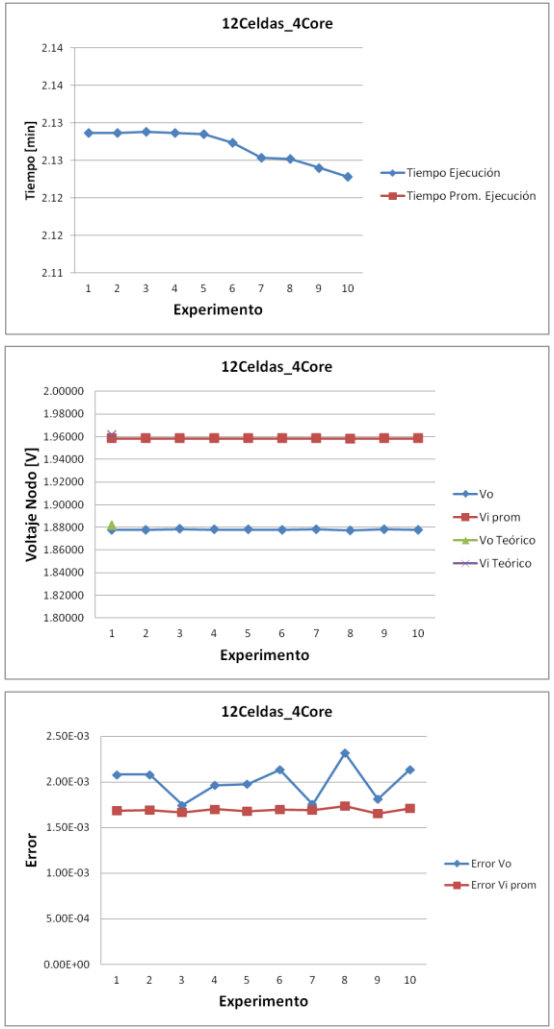


Figura 37 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión serial.

Tabla 19 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión serial.

12 celdas (13 dimensiones)															
Experimento	Nodo #	Voltaje	tiempo[min]	[ms]	Gfit	Promedios	Error voltaje	Experimento	Nodo #	Voltaje	tiempo[m]	[ms]	Gfit	Promedios	Error voltaje
1	V[0]	1.8773	2.76	165810	4.13E-02	1.87731	2.31E-03	6	V[0]	1.8780	2.78	166510	3.67E-02	1.87795	1.97E-03
	V[1]	1.9582				1.95825	1.72E-03		V[1]	1.9586				1.95832	1.68E-03
	V[2]	1.9583							V[2]	1.9582					
	V[3]	1.9584							V[3]	1.9582					
	V[4]	1.9582							V[4]	1.9584					
	V[5]	1.9582							V[5]	1.9584					
	V[6]	1.9582							V[6]	1.9584					
	V[7]	1.9582							V[7]	1.9584					
	V[8]	1.9582							V[8]	1.9584					
	V[9]	1.9584							V[9]	1.9582					
	V[10]	1.9582							V[10]	1.9582					
	V[11]	1.9581							V[11]	1.9580					
	V[12]	1.9583							V[12]	1.9585					
2	V[0]	1.8783	2.80	167960	2.67E-02	1.87828	1.79E-03	7	V[0]	1.8775	2.80	168200	4.62E-02	1.87752	2.19E-03
	V[1]	1.9584				1.95836	1.66E-03		V[1]	1.9584				1.95824	1.72E-03
	V[2]	1.9585							V[2]	1.9579					
	V[3]	1.9582							V[3]	1.9580					
	V[4]	1.9585							V[4]	1.9583					
	V[5]	1.9582							V[5]	1.9583					
	V[6]	1.9585							V[6]	1.9583					
	V[7]	1.9584							V[7]	1.9582					
	V[8]	1.9582							V[8]	1.9585					
	V[9]	1.9584							V[9]	1.9584					
	V[10]	1.9583							V[10]	1.9583					
	V[11]	1.9585							V[11]	1.9582					
	V[12]	1.9584							V[12]	1.9583					
3	V[0]	1.8781	2.77	166160	3.68E-02	1.87808	1.90E-03	8	V[0]	1.8775	2.80	168120	3.98E-02	1.87751	2.20E-03
	V[1]	1.9582				1.95834	1.67E-03		V[1]	1.9584				1.95831	1.69E-03
	V[2]	1.9584							V[2]	1.9582					
	V[3]	1.9583							V[3]	1.9583					
	V[4]	1.9585							V[4]	1.9584					
	V[5]	1.9581							V[5]	1.9582					
	V[6]	1.9585							V[6]	1.9583					
	V[7]	1.9581							V[7]	1.9585					
	V[8]	1.9584							V[8]	1.9583					
	V[9]	1.9583							V[9]	1.9584					
	V[10]	1.9587							V[10]	1.9584					
	V[11]	1.9582							V[11]	1.9579					
	V[12]	1.9583							V[12]	1.9583					
4	V[0]	1.8777	2.78	166830	3.51E-02	1.87767	2.11E-03	9	V[0]	1.8778	2.80	167910	3.81E-02	1.87784	2.02E-03
	V[1]	1.9583				1.95834	1.67E-03		V[1]	1.9583				1.95827	1.71E-03
	V[2]	1.9583							V[2]	1.9584					
	V[3]	1.9580							V[3]	1.9580					
	V[4]	1.9584							V[4]	1.9584					
	V[5]	1.9584							V[5]	1.9583					
	V[6]	1.9585							V[6]	1.9583					
	V[7]	1.9583							V[7]	1.9581					
	V[8]	1.9584							V[8]	1.9584					
	V[9]	1.9583							V[9]	1.9581					
	V[10]	1.9585							V[10]	1.9583					
	V[11]	1.9585							V[11]	1.9585					
	V[12]	1.9582							V[12]	1.9582					

ANEXO C RESULTADOS PARA PRUEBAS CÓDIGO PARALELO

ANEXO C.1 EJECUCIÓN OPENMP 1 CELDA SOLAR

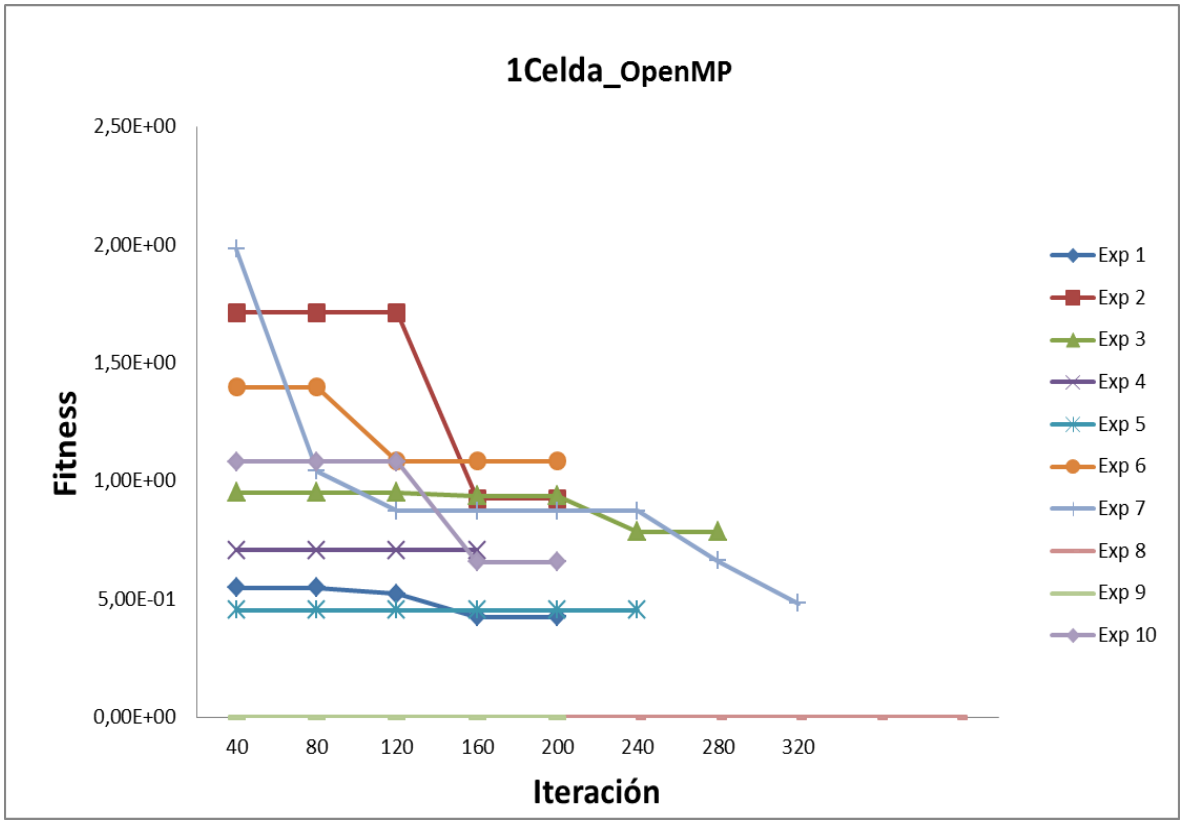


Figura 38 Convergencia del algoritmo para 1 Celda en la versión OpenMP.

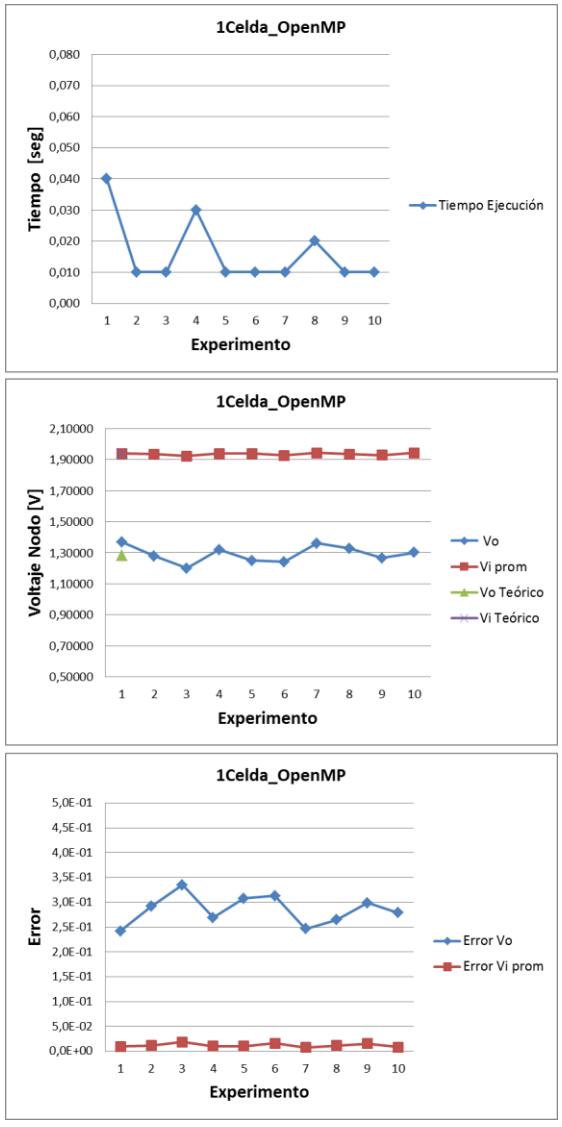


Figura 39 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión OpenMP.

Tabla 20 Tiempo de Ejecución, Voltajes, Fitness y Error para ejecución con 1 Celda en la versión OpenMP.

1 celda (2 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje
1	V[0]	1.37E+00	0.040	40	3.28E-03	1.36831	2.4E-01
	V[1]	1.94E+00				1.93995	9.7E-03
2	V[0]	1.28E+00	0.010	10	3.91E-03	1.27825	2.9E-01
	V[1]	1.94E+00				1.93594	1.2E-02
3	V[0]	1.20E+00	0.010	10	8.83E-03	1.20011	3.4E-01
	V[1]	1.92E+00				1.92241	1.9E-02
4	V[0]	1.32E+00	0.030	30	1.03E-03	1.31994	2.7E-01
	V[1]	1.94E+00				1.93906	1.0E-02
5	V[0]	1.25E+00	0.010	10	3.39E-03	1.24923	3.1E-01
	V[1]	1.94E+00				1.93947	9.9E-03
6	V[0]	1.24E+00	0.010	10	2.60E-03	1.24059	3.1E-01
	V[1]	1.93E+00				1.92700	1.6E-02
7	V[0]	1.36E+00	0.010	10	9.79E-03	1.36071	2.5E-01
	V[1]	1.94E+00				1.94433	7.4E-03
8	V[0]	1.33E+00	0.020	20	4.83E-03	1.32723	2.6E-01
	V[1]	1.94E+00				1.93640	1.1E-02
9	V[0]	1.27E+00	0.010	10	3.99E-03	1.26619	3.0E-01
	V[1]	1.93E+00				1.92902	1.5E-02
10	V[0]	1.30E+00	0.010	10	2.66E-03	1.30147	2.8E-01
	V[1]	1.94E+00				1.94288	8.2E-03

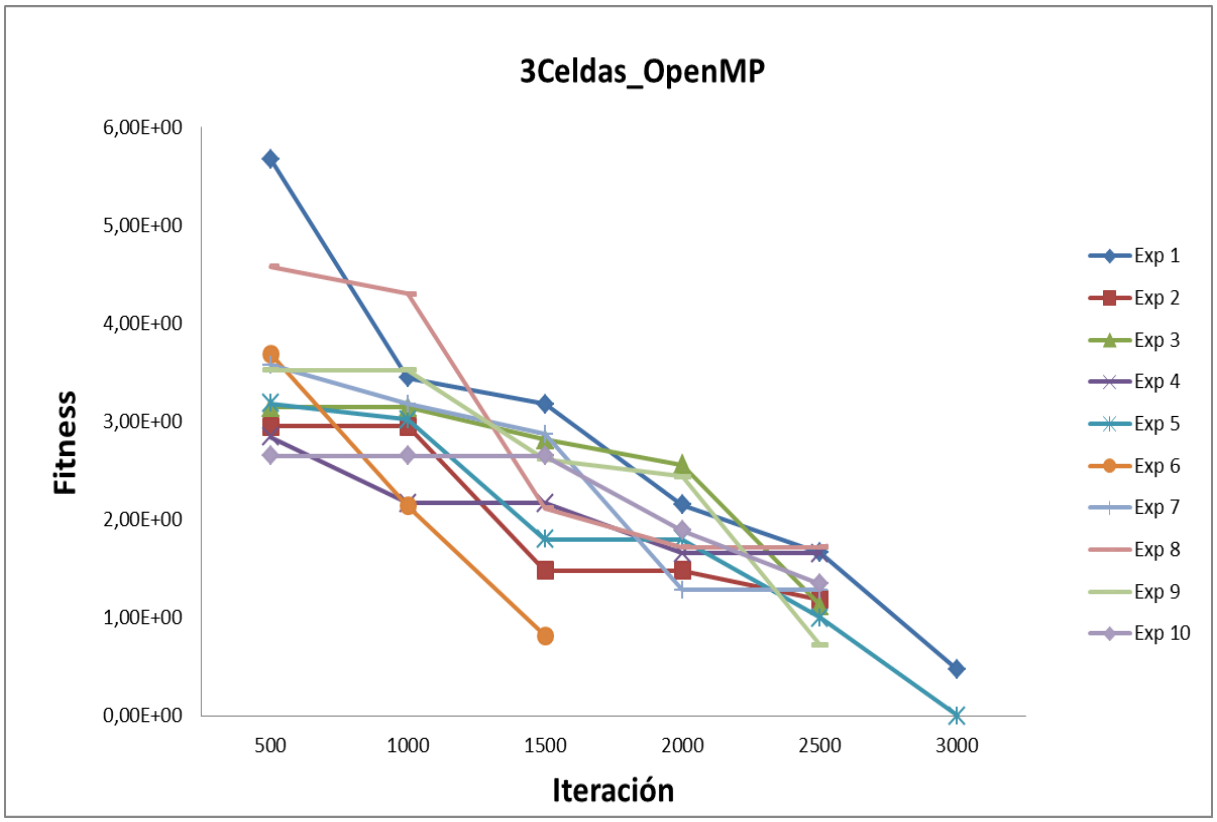


Figura 40 Convergencia del algoritmo para 3 Celdas en la versión OpenMP.

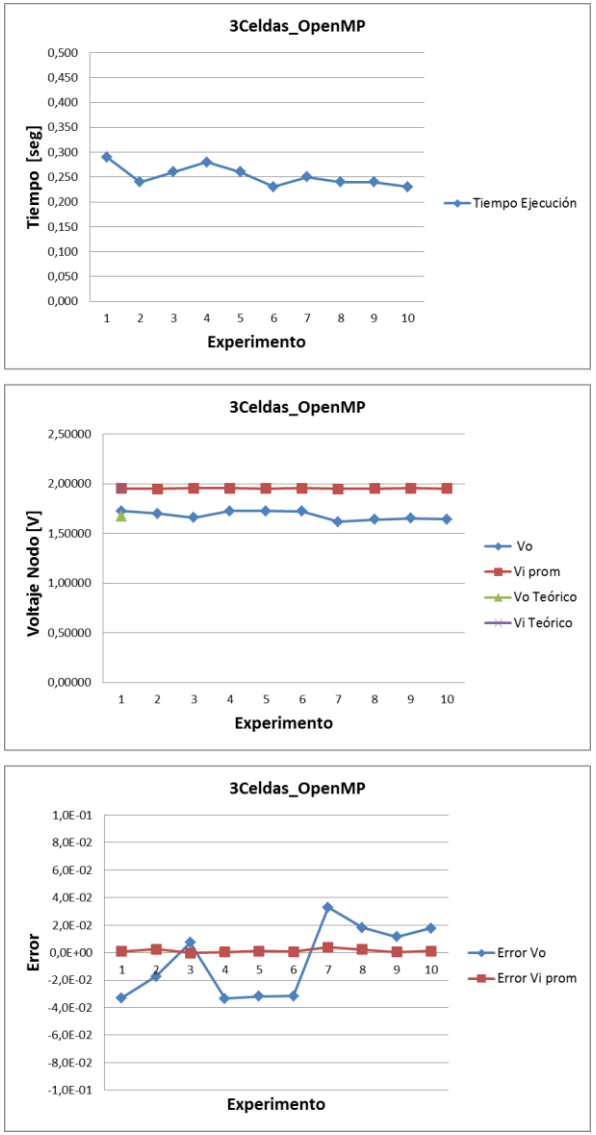


Figura 41 Tiempo de Ejecución, Voltajes y Error para ejecución con 3 Celdas en la versión OpenMP.

Tabla 21 Tiempo de Ejecución, Voltajes y Error para ejecución con 3 Celdas en la versión OpenMP.

3 celdas (4 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje
1	V[0]	1.72E+00	0.290	290	3.98E-01	1.72475	-3.3E-02
	V[1]	1.95E+00				1.95223	7.8E-04
	V[2]	1.95E+00					
	V[3]	1.95E+00					
2	V[0]	1.70E+00	0.240	240	3.86E-01	1.69875	-1.7E-02
	V[1]	1.95E+00				1.94892	2.5E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
3	V[0]	1.66E+00	0.260	260	3.72E-01	1.65771	7.3E-03
	V[1]	1.95E+00				1.95406	-1.5E-04
	V[2]	1.96E+00					
	V[3]	1.95E+00					
4	V[0]	1.73E+00	0.280	280	0.00E+00	1.72567	-3.3E-02
	V[1]	1.95E+00				1.95295	4.1E-04
	V[2]	1.95E+00					
	V[3]	1.95E+00					
5	V[0]	1.72E+00	0.260	260	3.98E-01	1.72306	-3.2E-02
	V[1]	1.95E+00				1.95154	1.1E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
6	V[0]	1.72E+00	0.230	230	3.74E-01	1.72262	-3.2E-02
	V[1]	1.95E+00				1.95258	6.0E-04
	V[2]	1.95E+00					
	V[3]	1.95E+00					
7	V[0]	1.62E+00	0.250	250	3.97E-01	1.61502	3.3E-02
	V[1]	1.95E+00				1.94625	3.8E-03
	V[2]	1.95E+00					
	V[3]	1.94E+00					
8	V[0]	1.64E+00	0.240	240	3.32E-01	1.63967	1.8E-02
	V[1]	1.95E+00				1.94926	2.3E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					
9	V[0]	1.65E+00	0.240	240	3.82E-01	1.65067	1.2E-02
	V[1]	1.95E+00				1.95281	4.9E-04
	V[2]	1.96E+00					
	V[3]	1.95E+00					
10	V[0]	1.64E+00	0.230	230	3.34E-01	1.64046	1.8E-02
	V[1]	1.95E+00				1.95143	1.2E-03
	V[2]	1.95E+00					
	V[3]	1.95E+00					

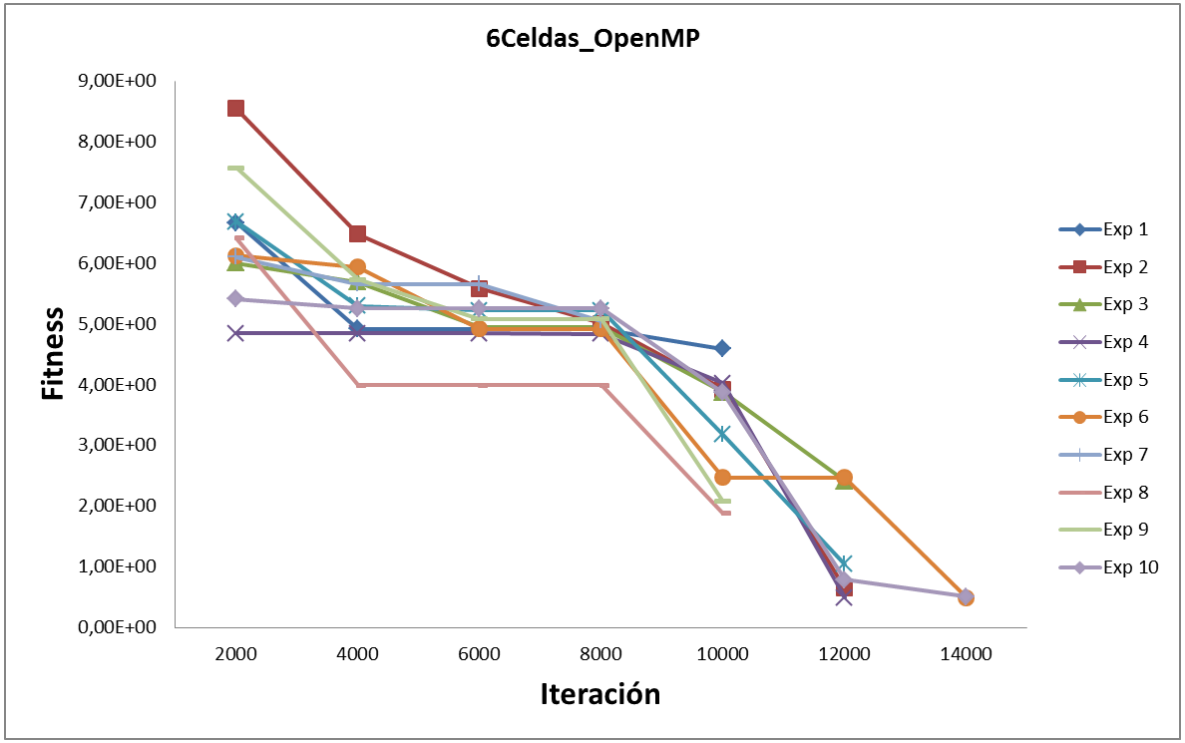


Figura 43 Convergencia del algoritmo para 6 Celdas en la versión OpenMP.

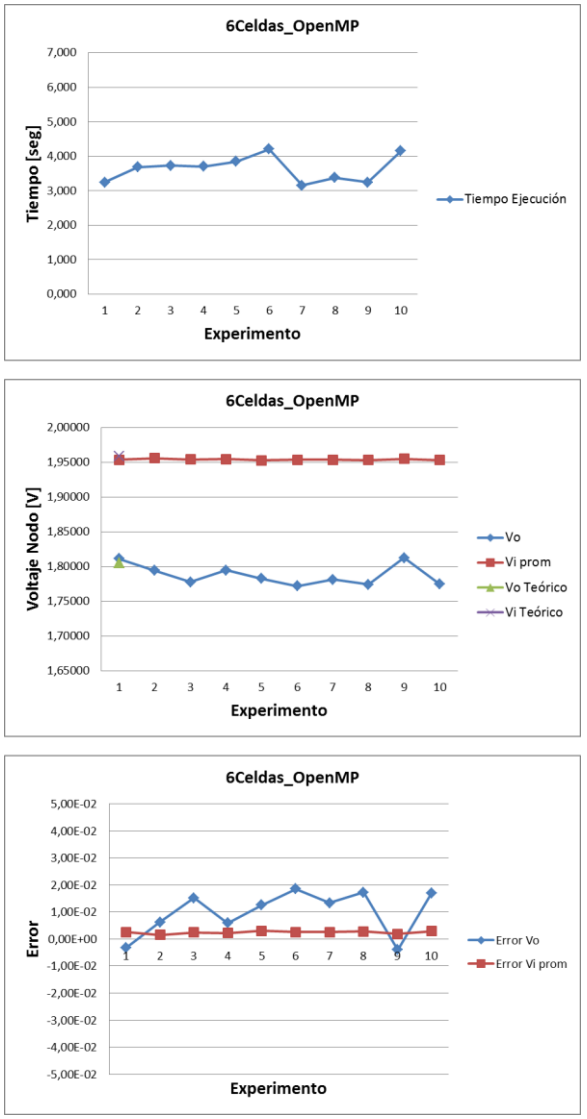


Figura 42 Tiempo de Ejecución, Voltajes y Error para ejecución con 6 Celdas en la versión OpenMP.

Tabla 22 Tiempo de Ejecución, Voltajes y Error para ejecución con 6 Celdas en la versión OpenMP.

6 celdas (7 dimensiones)																
Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje		Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje
1	V[0]	1.8113	3.240	3240	1.83E-02	1.81131	-3.27E-03		6	V[0]	1.7721	4.210	4210	1.37E-02	1.77206	1.85E-02
	V[1]	1.9557				1.95396	2.51E-03			V[1]	1.9544				1.95386	2.56E-03
	V[2]	1.9556								V[2]	1.9547					
	V[3]	1.9550								V[3]	1.9521					
	V[4]	1.9551								V[4]	1.9545					
	V[5]	1.9521								V[5]	1.9540					
	V[6]	1.9503								V[6]	1.9534					
2	V[0]	1.7942	3.680	3680	8.39E-03	1.79423	6.19E-03		7	V[0]	1.7814	3.150	3150	9.75E-03	1.78138	1.33E-02
	V[1]	1.9560				1.95586	1.54E-03			V[1]	1.9566				1.95387	2.55E-03
	V[2]	1.9548								V[2]	1.9552					
	V[3]	1.9557								V[3]	1.9540					
	V[4]	1.9605								V[4]	1.9553					
	V[5]	1.9511								V[5]	1.9502					
	V[6]	1.9571								V[6]	1.9519					
3	V[0]	1.7780	3.730	3730	1.40E-02	1.77799	1.52E-02		8	V[0]	1.7742	3.380	3380	2.16E-02	1.77422	1.73E-02
	V[1]	1.9544				1.95407	2.45E-03			V[1]	1.9533				1.95334	2.82E-03
	V[2]	1.9541								V[2]	1.9540					
	V[3]	1.9521								V[3]	1.9524					
	V[4]	1.9564								V[4]	1.9517					
	V[5]	1.9531								V[5]	1.9539					
	V[6]	1.9542								V[6]	1.9548					
4	V[0]	1.7947	3.700	3700	2.13E-02	1.79471	5.92E-03		9	V[0]	1.8126	3.240	3240	1.93E-02	1.81261	-3.99E-03
	V[1]	1.9572				1.95453	2.21E-03			V[1]	1.9551				1.95516	1.89E-03
	V[2]	1.9568								V[2]	1.9546					
	V[3]	1.9532								V[3]	1.9544					
	V[4]	1.9525								V[4]	1.9566					
	V[5]	1.9569								V[5]	1.9542					
	V[6]	1.9507								V[6]	1.9561					
5	V[0]	1.7827	3.850	3850	1.53E-02	1.78274	1.26E-02		10	V[0]	1.7749	4.150	4150	1.33E-02	1.77486	1.69E-02
	V[1]	1.9533				1.95298	3.01E-03			V[1]	1.9530				1.95315	2.92E-03
	V[2]	1.9525								V[2]	1.9541					
	V[3]	1.9559								V[3]	1.9537					
	V[4]	1.9532								V[4]	1.9517					
	V[5]	1.9520								V[5]	1.9521					
	V[6]	1.9510								V[6]	1.9543					

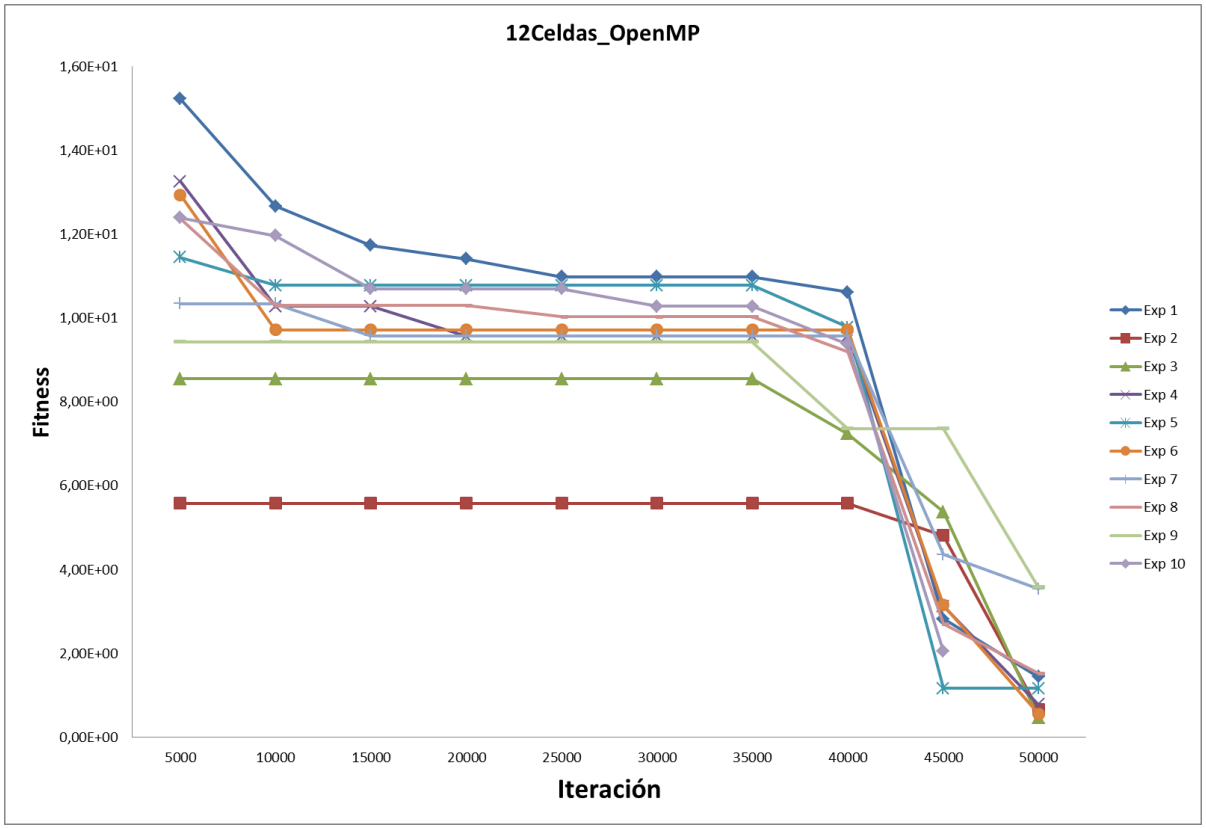


Figura 45 Convergencia del algoritmo para 12 Celdas en la versión OpenMP.

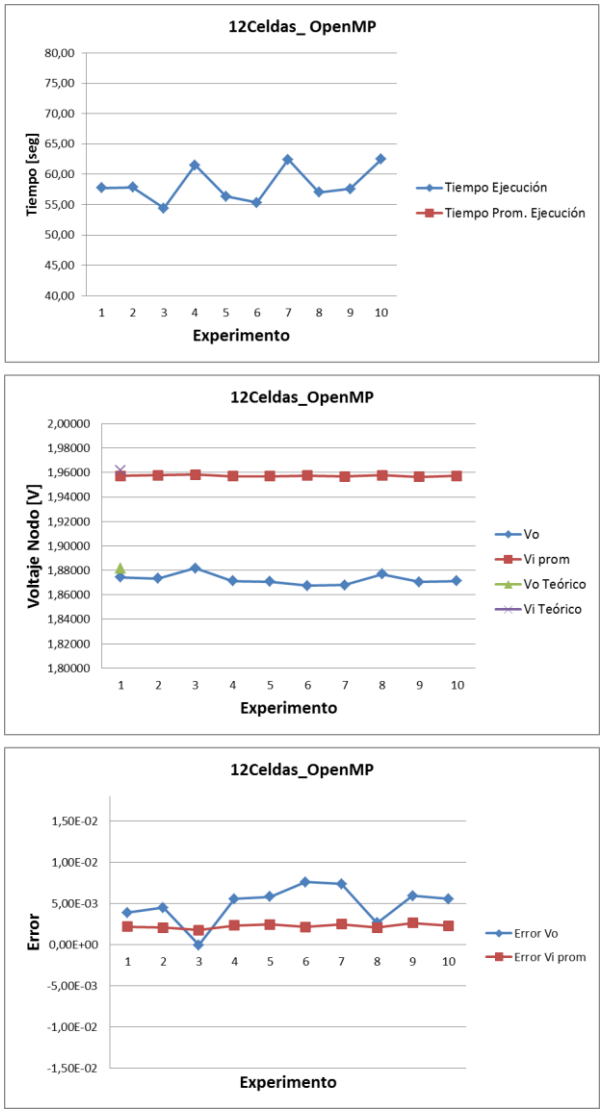


Figura 44 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión OpenMP.

Tabla 23 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión OpenMP.

12 celdas (13 dimensiones)																
Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje	Experimento	Nodo #	Voltaje	tiempo[s][ms]		Gfit	Promedios	Error voltaje
1	V[0]	1.8744	57.78	57780	3.75E-01	1.87442	3.84E-03	6	V[0]	1.8674	55.33	55330	3.89E-01	1.86738	7.58E-03
	V[1]	1.9581				1.95736	2.17E-03		V[1]	1.9574				1.95748	2.11E-03
	V[2]	1.9577							V[2]	1.9592					
	V[3]	1.9585							V[3]	1.9579					
	V[4]	1.9593							V[4]	1.9588					
	V[5]	1.9579							V[5]	1.9600					
	V[6]	1.9566							V[6]	1.9544					
	V[7]	1.9555							V[7]	1.9565					
	V[8]	1.9575							V[8]	1.9578					
	V[9]	1.9592							V[9]	1.9566					
	V[10]	1.9593							V[10]	1.9583					
	V[11]	1.9544							V[11]	1.9562					
	V[12]	1.9543							V[12]	1.9566					
2	V[0]	1.8733	57.88	57880	3.80E-01	1.87326	4.46E-03	7	V[0]	1.8678	62.46	62460	3.97E-01	1.86785	7.33E-03
	V[1]	1.9611				1.95764	2.03E-03		V[1]	1.9575				1.95677	2.47E-03
	V[2]	1.9592							V[2]	1.9572					
	V[3]	1.9568							V[3]	1.9572					
	V[4]	1.9543							V[4]	1.9546					
	V[5]	1.9597							V[5]	1.9568					
	V[6]	1.9560							V[6]	1.9584					
	V[7]	1.9571							V[7]	1.9578					
	V[8]	1.9594							V[8]	1.9563					
	V[9]	1.9576							V[9]	1.9556					
	V[10]	1.9567							V[10]	1.9548					
	V[11]	1.9576							V[11]	1.9570					
	V[12]	1.9561							V[12]	1.9581					
3	V[0]	1.8818	54.44	54440	3.39E-01	1.88177	-6.54E-05	8	V[0]	1.8768	57.02	57020	3.72E-01	1.87676	2.60E-03
	V[1]	1.9564				1.95824	1.72E-03		V[1]	1.9587				1.95765	2.02E-03
	V[2]	1.9582							V[2]	1.9601					
	V[3]	1.9603							V[3]	1.9575					
	V[4]	1.9560							V[4]	1.9568					
	V[5]	1.9600							V[5]	1.9567					
	V[6]	1.9588							V[6]	1.9581					
	V[7]	1.9559							V[7]	1.9572					
	V[8]	1.9583							V[8]	1.9589					
	V[9]	1.9596							V[9]	1.9526					
	V[10]	1.9576							V[10]	1.9597					
	V[11]	1.9568							V[11]	1.9585					
	V[12]	1.9611							V[12]	1.9570					
4	V[0]	1.8712	61.49	61490	3.88E-01	1.87124	5.53E-03	9	V[0]	1.8705	57.63	57630	3.94E-01	1.87051	5.92E-03
	V[1]	1.9590				1.95709	2.31E-03		V[1]	1.9573				1.95650	2.61E-03
	V[2]	1.9600							V[2]	1.9565					
	V[3]	1.9580							V[3]	1.9566					
	V[4]	1.9559							V[4]	1.9568					
	V[5]	1.9550							V[5]	1.9547					
	V[6]	1.9558							V[6]	1.9568					
	V[7]	1.9580							V[7]	1.9573					
	V[8]	1.9560							V[8]	1.9554					
	V[9]	1.9566							V[9]	1.9565					
	V[10]	1.9567							V[10]	1.9573					
	V[11]	1.9589							V[11]	1.9548					
	V[12]	1.9552							V[12]	1.9582					

ANEXO D RESULTADOS PARA PRUEBAS CÓDIGO PARALELO.

ANEXO D.1 EJECUCIÓN CUDA 1 CELDA SOLAR

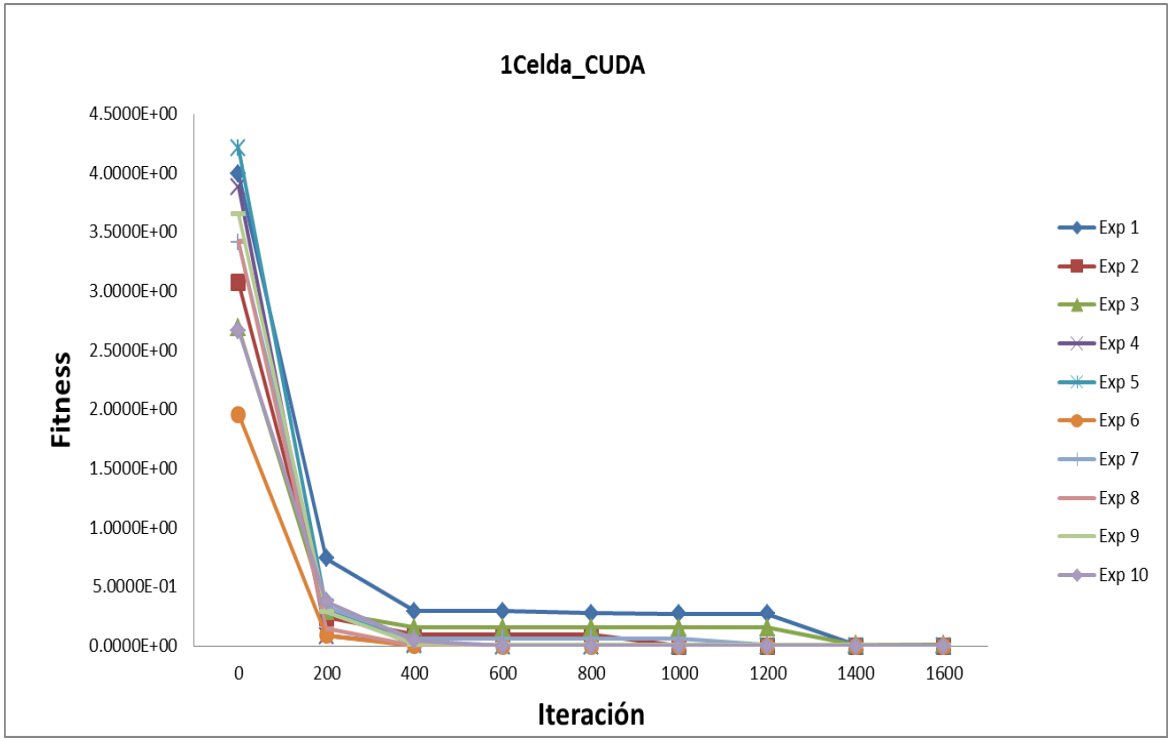


Figura 46 Convergencia del algoritmo para 1 Celda en la versión CUDA.

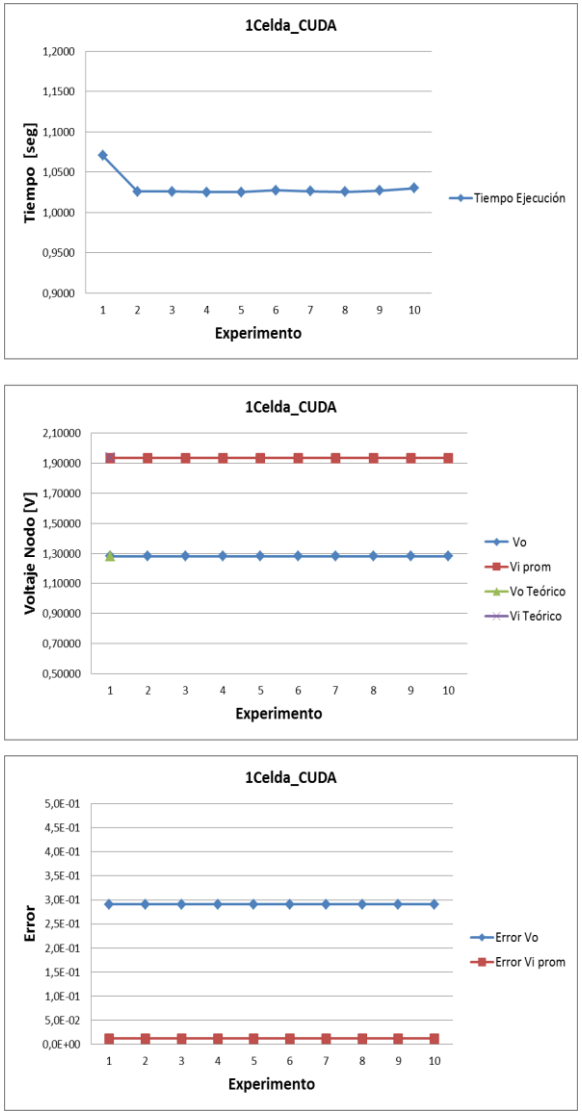


Figura 47 Tiempo de Ejecución, Voltajes y Error para ejecución con 1 Celda en la versión CUDA.

Tabla 24 Tiempo de Ejecución, Voltajes y Error para ejecución con 1 Celda en la versión CUDA.

1 celda (2 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s][us]		Gfit	Promedios	Error voltaje
1	V[0]	1,2809	1,0707	1070748	3,28E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
2	V[0]	1,2809	1,0260	1025978	3,91E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
3	V[0]	1,2809	1,0260	1025978	8,83E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
4	V[0]	1,2809	1,0254	1025355	1,03E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
5	V[0]	1,2809	1,0254	1025408	3,39E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
6	V[0]	1,2809	1,0277	1027736	2,60E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
7	V[0]	1,2809	1,0263	1026259	9,79E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
8	V[0]	1,2809	1,0255	1025479	4,83E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
9	V[0]	1,2809	1,0273	1027303	3,99E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02
10	V[0]	1,2809	1,0301	1030149	2,66E-03	1,28092	2,9E-01
	V[1]	1,9342				1,93420	1,3E-02

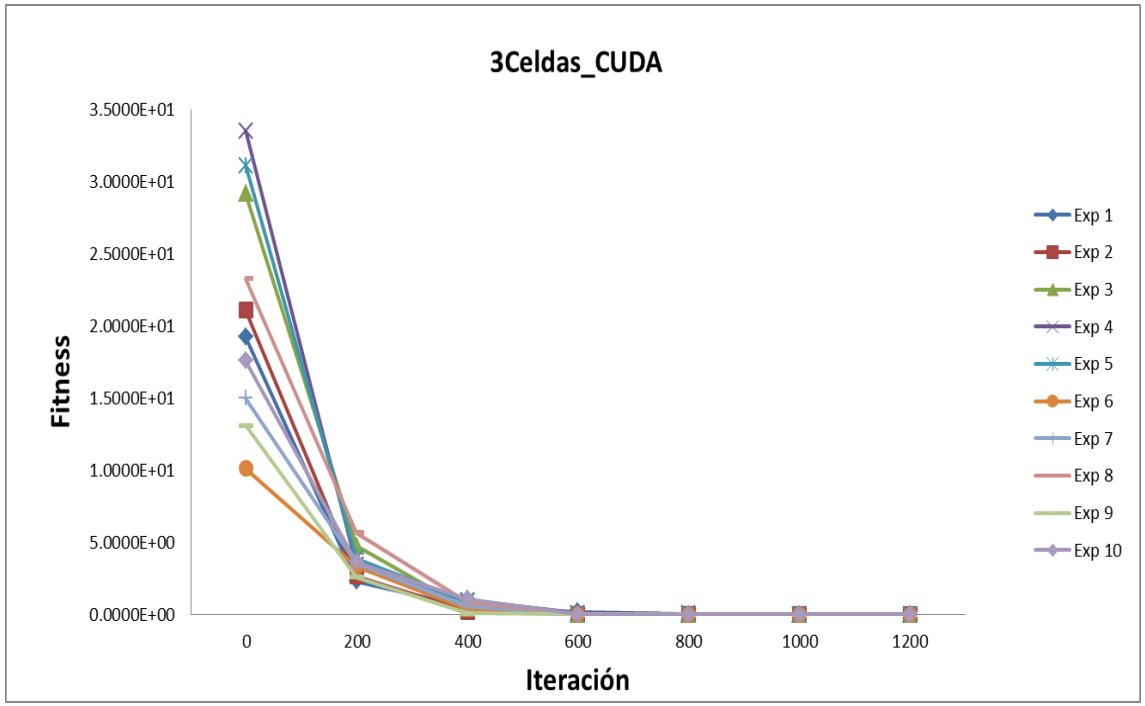


Figura 48 Convergencia del algoritmo para 3 Celdas en la versión CUDA.

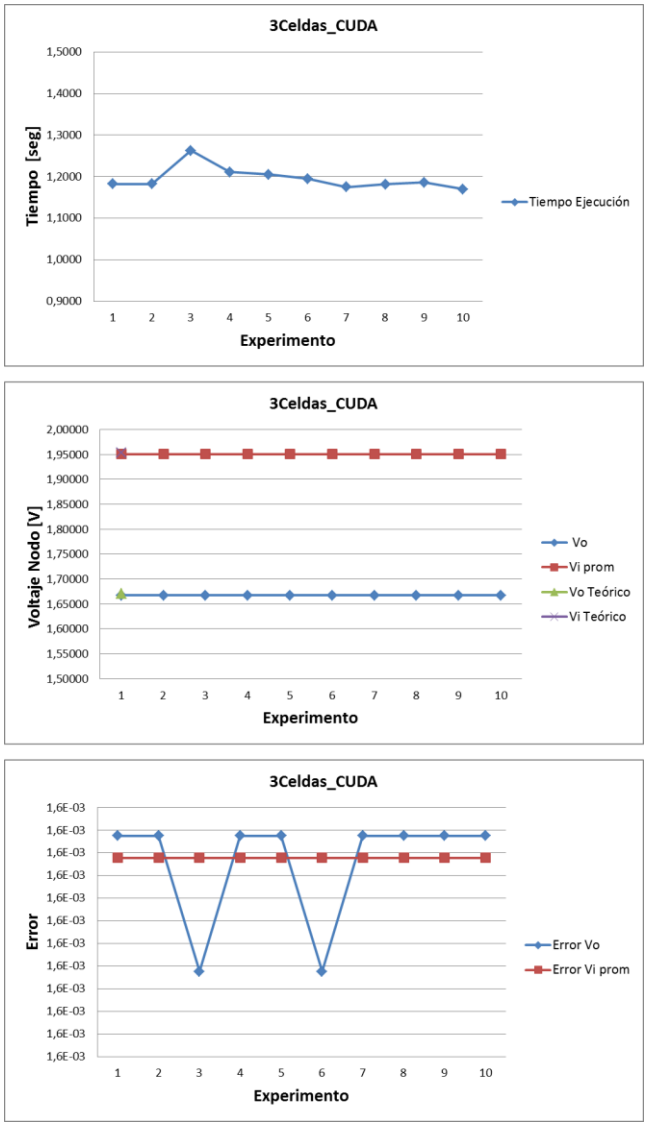


Figura 49 Tiempo de Ejecución, Voltajes y Error para ejecución con 3 Celdas en la versión CUDA.

Tabla 25 Tiempo de Ejecución, Voltajes y Error para ejecución con 3 Celdas en la versión CUDA.

3 celdas (4 dimensiones)							
Experimento	Nodo #	Voltaje	tiempo[s][us]		Gfit	Promedios	Error voltaje
1	V[0]	1,67E+00	1,1826	1182557	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
2	V[0]	1,67E+00	1,1824	1182364	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
3	V[0]	1,67E+00	1,2629	1262850	6,71E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
4	V[0]	1,67E+00	1,2113	1211330	1,12E-05	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
5	V[0]	1,67E+00	1,2055	1205451	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
6	V[0]	1,67E+00	1,1947	1194748	7,15E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
7	V[0]	1,67E+00	1,1746	1174617	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
8	V[0]	1,67E+00	1,1815	1181541	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
9	V[0]	1,67E+00	1,1862	1186191	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					
10	V[0]	1,67E+00	1,1699	1169895	5,19E-06	1,66717	1,6E-03
	V[1]	1,95E+00				1,95059	1,6E-03
	V[2]	1,95E+00					
	V[3]	1,95E+00					

ANEXO D.3 EJECUCIÓN CUDA 6 CELDAS SOLARES

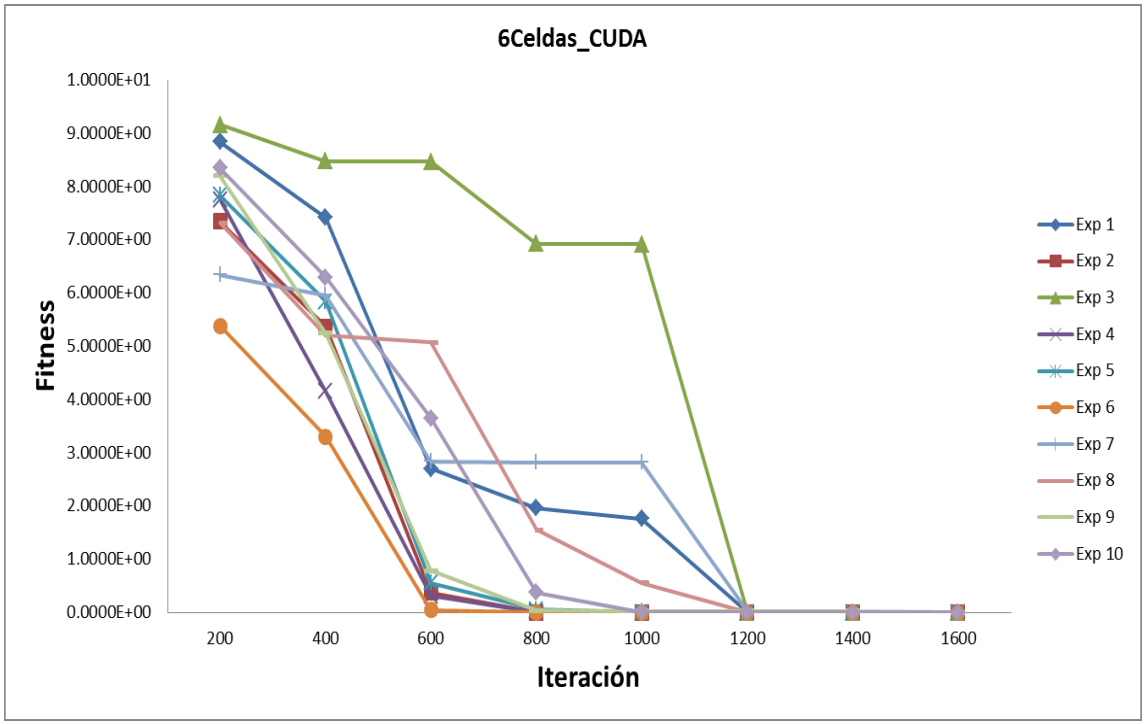
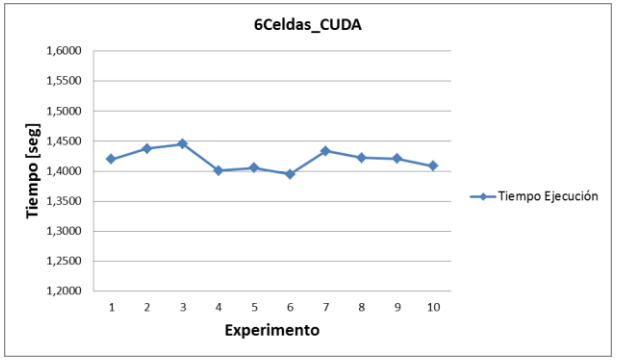
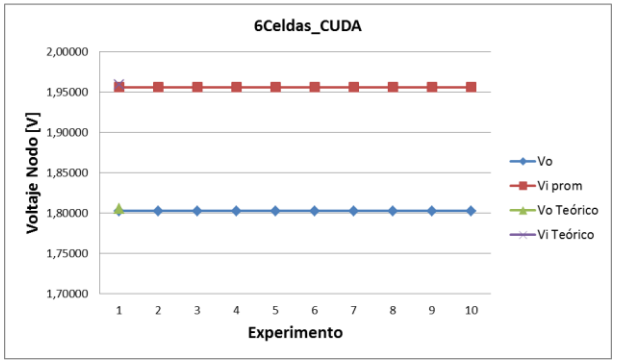


Figura 50 Convergencia del algoritmo para 6 Celdas en la versión CUDA.





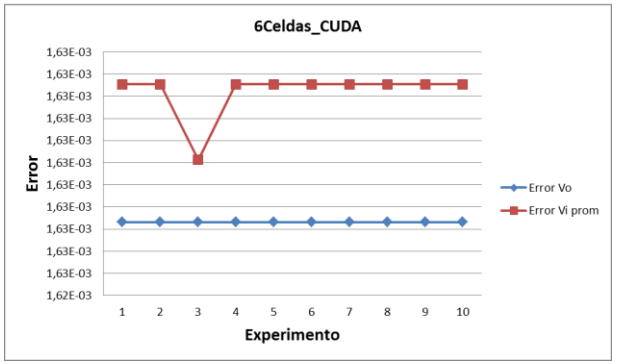


Figura 51 Tiempo de Ejecución, Voltajes y Error para ejecución con 6 Celdas en la versión CUDA.

Tabla 26 Tiempo de Ejecución, Voltajes y Error para ejecución con 6 Celdas en la versión CUDA.

6 celdas (7 dimensiones)

Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje		Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje
1	V[0]	1,8025	1,4194	1419443	1,83E-02	1,80248	1,63E-03		6	V[0]	1,8025	1,3948	1394803	1,37E-02	1,80248	1,63E-03
	V[1]	1,9557				1,95569	1,63E-03			V[1]	1,9557				1,95569	1,63E-03
	V[2]	1,9557								V[2]	1,9557					
	V[3]	1,9557								V[3]	1,9557					
	V[4]	1,9557								V[4]	1,9557					
	V[5]	1,9557								V[5]	1,9557					
	V[6]	1,9557								V[6]	1,9557					
2	V[0]	1,8025	1,4377	1437650	8,39E-03	1,80248	1,63E-03		7	V[0]	1,8025	1,4336	1433553	9,75E-03	1,80248	1,63E-03
	V[1]	1,9557				1,95569	1,63E-03			V[1]	1,9557				1,95569	1,63E-03
	V[2]	1,9557								V[2]	1,9557					
	V[3]	1,9557								V[3]	1,9557					
	V[4]	1,9557								V[4]	1,9557					
	V[5]	1,9557								V[5]	1,9557					
	V[6]	1,9557								V[6]	1,9557					
3	V[0]	1,8025	1,4451	1445074	1,40E-02	1,80248	1,63E-03		8	V[0]	1,8025	1,4221	1422093	2,16E-02	1,80248	1,63E-03
	V[1]	1,9557				1,95569	1,63E-03			V[1]	1,9557				1,95569	1,63E-03
	V[2]	1,9557								V[2]	1,9557					
	V[3]	1,9557								V[3]	1,9557					
	V[4]	1,9557								V[4]	1,9557					
	V[5]	1,9557								V[5]	1,9557					
	V[6]	1,9557								V[6]	1,9557					
4	V[0]	1,8025	1,4009	1400877	2,13E-02	1,80248	1,63E-03		9	V[0]	1,8025	1,4206	1420603	1,93E-02	1,80248	1,63E-03
	V[1]	1,9557				1,95569	1,63E-03			V[1]	1,9557				1,95569	1,63E-03
	V[2]	1,9557								V[2]	1,9557					
	V[3]	1,9557								V[3]	1,9557					
	V[4]	1,9557								V[4]	1,9557					
	V[5]	1,9557								V[5]	1,9557					
	V[6]	1,9557								V[6]	1,9557					
5	V[0]	1,8025	1,4054	1405401	1,53E-02	1,80248	1,63E-03		10	V[0]	1,8025	1,4086	1408609	1,33E-02	1,80248	1,63E-03
	V[1]	1,9557				1,95569	1,63E-03			V[1]	1,9557				1,95569	1,63E-03
	V[2]	1,9557								V[2]	1,9557					
	V[3]	1,9557								V[3]	1,9557					
	V[4]	1,9557								V[4]	1,9557					
	V[5]	1,9557								V[5]	1,9557					
	V[6]	1,9557								V[6]	1,9557					

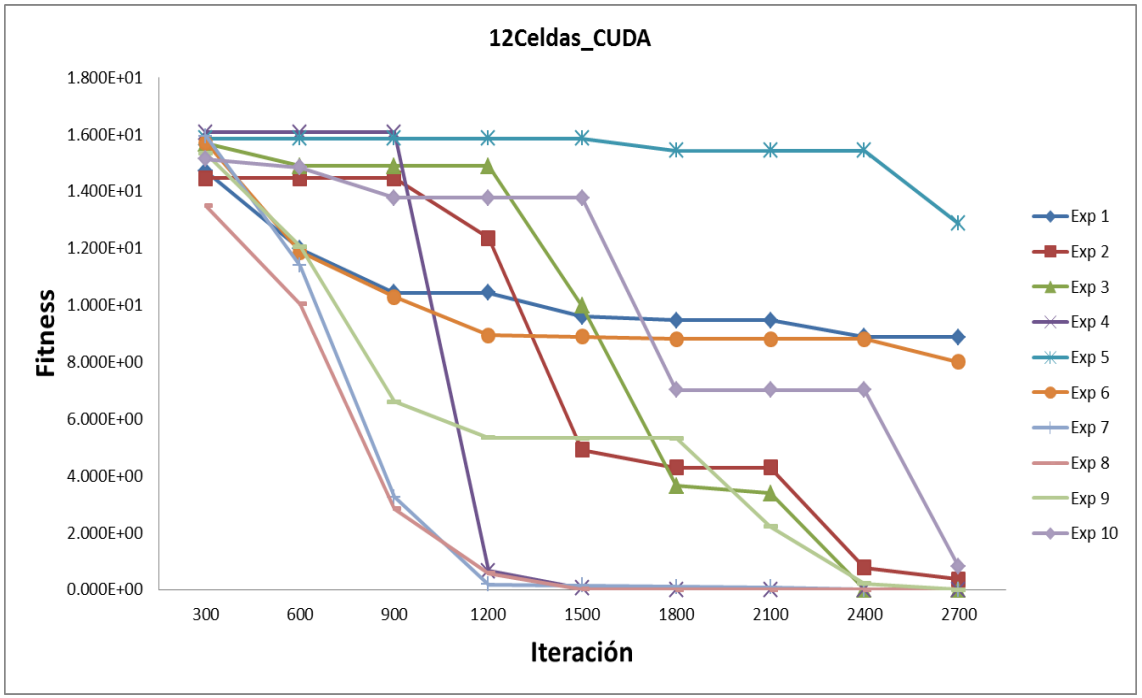


Figura 53 Convergencia del algoritmo para 12 Celdas en la versión CUDA.

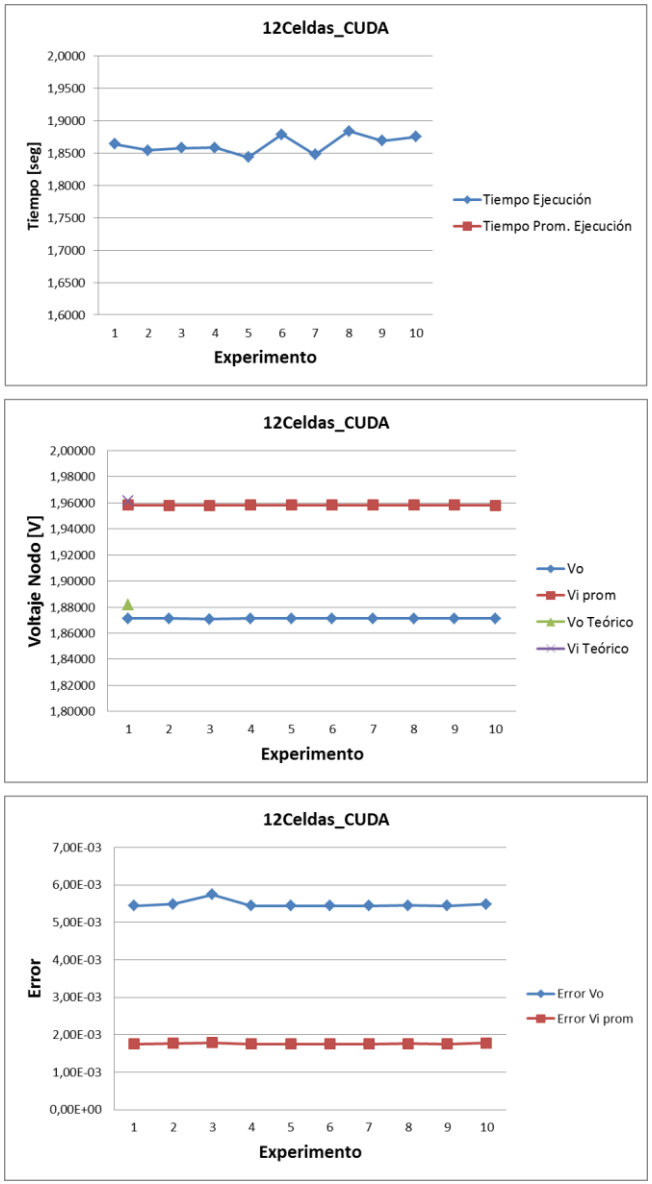


Figura 52 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión CUDA.

Tabla 27 Tiempo de Ejecución, Voltajes y Error para ejecución con 12 Celdas en la versión CUDA.

12 celdas (13 dimensiones)															
Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje	Experimento	Nodo #	Voltaje	tiempo[s]	[ms]	Gfit	Promedios	Error voltaje
1	V[0]	1,8714	1,8637	1863708	4,49E-05	1,87141	5,44E-03	6	V[0]	1,8714	1,8787	1878703	1,91E-05	1,87141	5,44E-03
	V[1]	1,9582				1,95817	1,76E-03		V[1]	1,9582				1,95817	1,76E-03
	V[2]	1,9582							V[2]	1,9582					
	V[3]	1,9582							V[3]	1,9582					
	V[4]	1,9582							V[4]	1,9582					
	V[5]	1,9582							V[5]	1,9582					
	V[6]	1,9582							V[6]	1,9582					
	V[7]	1,9582							V[7]	1,9582					
	V[8]	1,9582							V[8]	1,9582					
	V[9]	1,9582							V[9]	1,9582					
	V[10]	1,9582							V[10]	1,9582					
	V[11]	1,9582							V[11]	1,9582					
	V[12]	1,9582							V[12]	1,9582					
2	V[0]	1,8713	1,8541	1854104	1,42E-02	1,87132	5,49E-03	7	V[0]	1,8714	1,8476	1847578	4,84E-05	1,87141	5,44E-03
	V[1]	1,9582				1,95815	1,77E-03		V[1]	1,9582				1,95817	1,76E-03
	V[2]	1,9582							V[2]	1,9582					
	V[3]	1,9582							V[3]	1,9582					
	V[4]	1,9579							V[4]	1,9582					
	V[5]	1,9582							V[5]	1,9582					
	V[6]	1,9582							V[6]	1,9582					
	V[7]	1,9582							V[7]	1,9582					
	V[8]	1,9582							V[8]	1,9582					
	V[9]	1,9582							V[9]	1,9582					
	V[10]	1,9582							V[10]	1,9582					
	V[11]	1,9582							V[11]	1,9582					
	V[12]	1,9582							V[12]	1,9582					
3	V[0]	1,8708	1,8578	1857810	2,10E-02	1,87085	5,74E-03	8	V[0]	1,8714	1,8836	1883596	2,36E-04	1,87140	5,45E-03
	V[1]	1,9581				1,95811	1,79E-03		V[1]	1,9582				1,95817	1,76E-03
	V[2]	1,9580							V[2]	1,9582					
	V[3]	1,9581							V[3]	1,9582					
	V[4]	1,9582							V[4]	1,9582					
	V[5]	1,9579							V[5]	1,9582					
	V[6]	1,9581							V[6]	1,9582					
	V[7]	1,9582							V[7]	1,9582					
	V[8]	1,9581							V[8]	1,9582					
	V[9]	1,9581							V[9]	1,9582					
	V[10]	1,9580							V[10]	1,9582					
	V[11]	1,9582							V[11]	1,9582					
	V[12]	1,9582							V[12]	1,9582					
4	V[0]	1,8714	1,8580	1858020	1,53E-05	1,87141	5,44E-03	9	V[0]	1,8714	1,8688	1868809	2,28E-05	1,87141	5,44E-03
	V[1]	1,9582				1,95817	1,76E-03		V[1]	1,9582				1,95817	1,76E-03
	V[2]	1,9582							V[2]	1,9582					
	V[3]	1,9582							V[3]	1,9582					
	V[4]	1,9582							V[4]	1,9582					
	V[5]	1,9582							V[5]	1,9582					
	V[6]	1,9582							V[6]	1,9582					
	V[7]	1,9582							V[7]	1,9582					
	V[8]	1,9582							V[8]	1,9582					
	V[9]	1,9582							V[9]	1,9582					
	V[10]	1,9582							V[10]	1,9582					
	V[11]	1,9582							V[11]	1,9582					
	V[12]	1,9582							V[12]	1,9582					

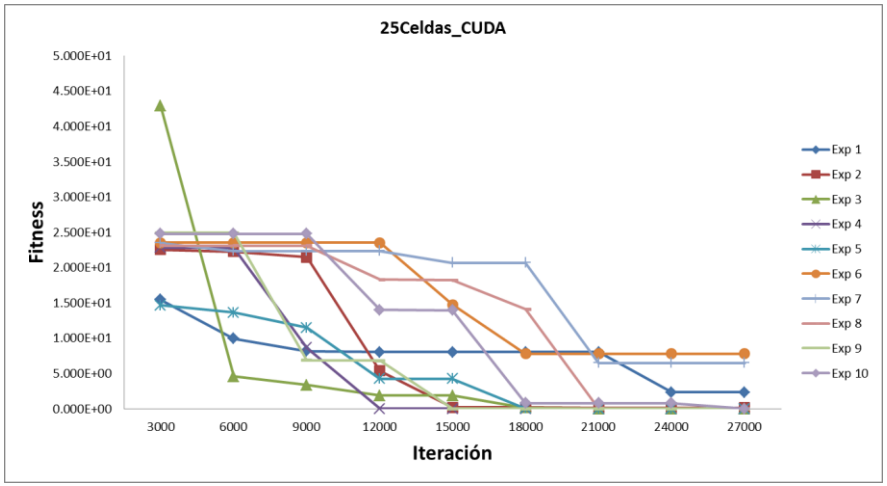


Figura 54 Convergencia del algoritmo para 25 Celdas

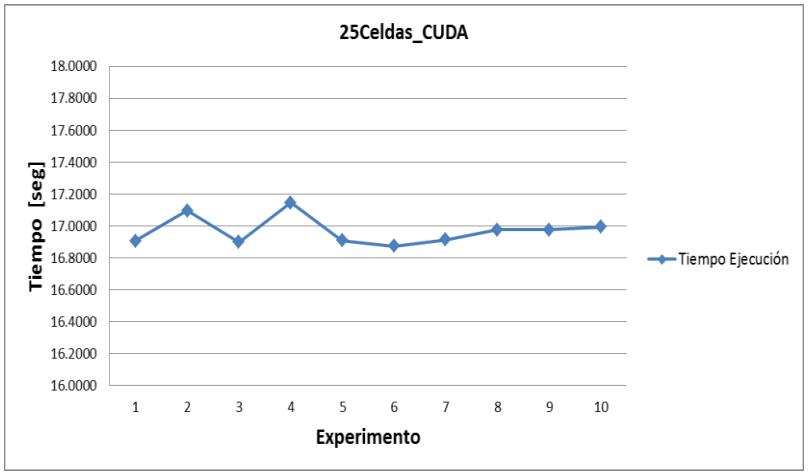


Figura 55 Tiempo de Ejecución para ejecución con 25 Celdas.

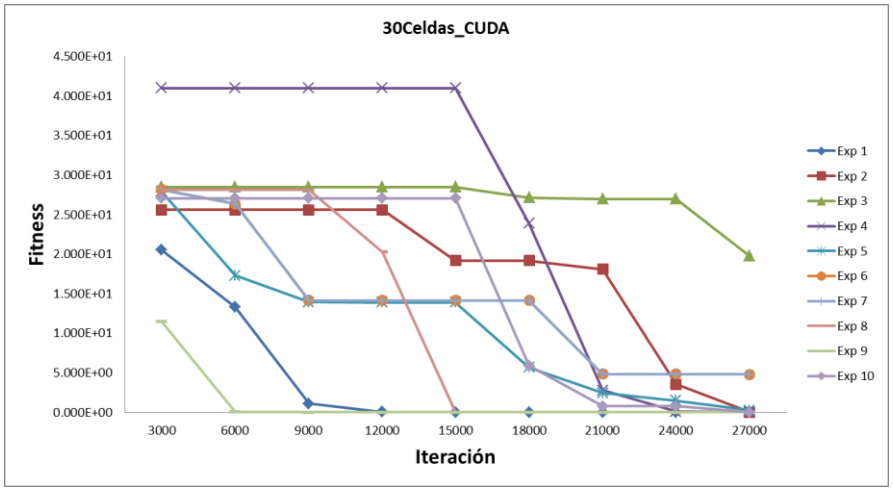


Figura 56 Convergencia del algoritmo para 30 Celdas

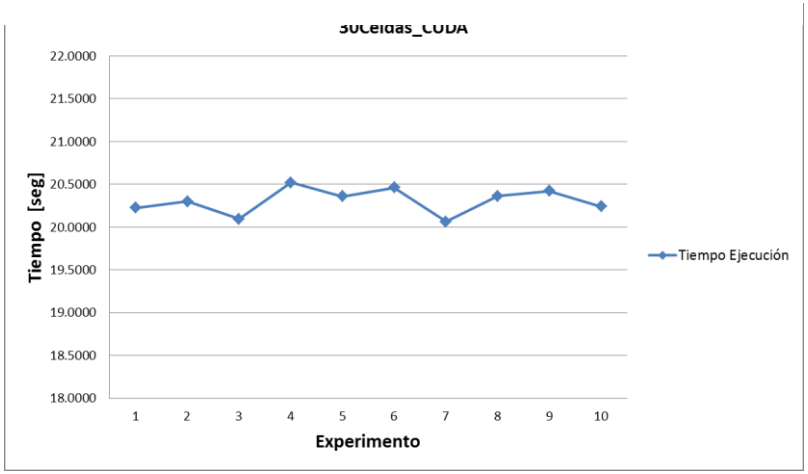


Figura 57 Tiempo de Ejecución para ejecución con 30 Celdas.