Aplicación de Técnicas de Procesamiento en Paralelo al Análisis en Estado Estable de Redes Eléctricas

PROYECTO DE TESIS

Que para obtener el Título de
INGENIERO ELECTRICISTA

Presenta
Rubén Darío Cervantes Gómez

Asesor de Tesis
Dr. Antonio Ramos Paz

Universidad Michoacana de San Nicolás de Hidalgo

Febrero de 2009

Agradecimientos

A mi familia por el apoyo incondicional que me ha brindado durante toda la vida.

Dedicatoria

A Darío, Miriam e Iván.

Resumen

El tiempo requerido en una simulación digital de un sistema eléctrico para alcanzar el estado estable aumenta en gran manera si dicho sistema presenta un pobre o nulo amortiguamiento. Para llevar a cabo la simulación digital es necesario resolver el sistema de ecuaciones diferenciales que describen al sistema eléctrico. Es resolviendo el sistema de ecuaciones diferenciales donde transcurre la mayor parte del tiempo de la simulación.

Debido a la necesidad de disminuir el tiempo de simulación se han utilizado distintos métodos numéricos para llegar rápidamente al estado estable. Uno de estos métodos es el de Diferenciación Numérica de Newton. Este método consiste en tomar el vector de condiciones iniciales de n elementos, sumarle al primer elemento una distorsión (un valor muy pequeño) y con este nuevo vector de condiciones iniciales resolver el sistema. El nuevo vector se guarda en la primera columna de una nueva matriz de tamaño nn. Esta matriz se llena sumándole una pequeña distorsión a cada elemento del vector de condiciones iniciales original.

Una alternativa utilizada para reducir aún más el tiempo de simulación es aprovechar el procesamiento en paralelo. Esto se logra dándole un enfoque en paralelo al método de diferenciación numérica en la parte en la que se obtiene la matriz de distorsiones. Dicho enfoque es diferente para cada una de las plataformas de procesamiento en paralelo. El enfoque para PVM (Máquina Virtual Paralela) consiste en dividir la tarea y asignar una parte a cada una de las computadoras de la máquina paralela, mientras que el enfoque para Multi-threading consiste en dividir la tarea y asignar una parte a cada uno de los núcleos que posee la computadora. En esta tesis en particular se hace una comparativa entre estos dos enfoques en cuanto al tiempo requerido para llegar al estado estable así como en la eficiencia obtenida.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos y Abreviaturas

mseg	milisegundos
EDO	Ecuación diferencial ordinaria
PVM	Máquina paralela virtual
t	Tiempo
d/dt	derivada con respecto del tiempo
V	Voltaje
L	Inductancia
R	Resistencia
C	Capacitancia
EEP	Estado Estable Periódico
DN	Diferenciación Numérica
AD	Aproximación Directa
FB	Fuerza Bruta
VLSI	*Very Large Scale Integration*
IEEE	*Institute of Electrical and Electronics Engineers*
LCK	Ley de corrientes de Kirchoff
LVK	Ley de voltajes de Kirchoff
J(x)	Jacobiano
CD	Corriente Directa
RK	Runge-Kutta
T	Periodo
E	Eficiencia

Capítulo 1

Introducción

1.1 Antecedentes

1.1.1 Simulación digital

En el diseño y planeación de los sistemas eléctricos de potencia, la simulación se utiliza principalmente para determinar su comportamiento bajo diversas condiciones de operación que resultan de efectuar algún cambio en los parámetros del sistema. Por su gran importancia en el estudio de los sistemas eléctricos, la simulación ha sido estudiada ampliamente y de manera general se presentan dos tipos de técnicas de análisis: técnicas aproximadas y técnicas exactas [Martínez-Velasco 1997]. Las técnicas aproximadas consisten en analizar el sistema eléctrico por medio de métodos numéricos, mientras que las técnicas exactas consisten en analizar el sistema eléctrico por medio de ecuaciones diferenciales.

En los años 1960s, la teoría de redes y la teoría de sistemas comenzaron a divergir después de haberse desarrollado juntas por casi 20 años. Por un lado, la teoría de redes se encaminó más en el análisis de grandes redes eléctricas que se interconectaban por medio de componentes relativamente sencillos (resistivos-inductivos-capacitivos). Por otro lado, la teoría de sistemas analizó configuraciones de redes muy simples con componentes muy complejos [De Carlo y Saeks 1981]. Pero no fue sino hasta los años 1980s que los dos campos comenzaron a unir esfuerzos motivados por el énfasis que se dio en el análisis de grandes redes, en donde se trató de explorar el desacoplamiento inherente en los sistemas eléctricos [De Carlo y Saeks 1981].

A principios de los años 1960s se dio un enfoque en el análisis de los circuitos eléctricos basado en el concepto de ondas viajeras. Este primer enfoque se aplicó en la

solución de pequeñas redes eléctricas con parámetros tanto lineales como no lineales y con elementos con parámetros distribuidos. Este análisis se extendió a las redes multinodos por Dommel [Dommel 1969]. Dommel implementó un algoritmo, basado en la regla trapezoidal, capaz de resolver transitorios en redes eléctricas monofásicas y polifásicas con parámetros concentrados y distribuidos.

La regla trapezoidal se utiliza para convertir las ecuaciones diferenciales asociadas a los componentes de la red en ecuaciones algebraicas que involucran voltajes, corrientes y valores pasados. La matriz de admitancias resultantes es simétrica y constante si se mantiene el mismo paso de integración. La solución del transitorio electromagnético se obtiene utilizando la factorización triangular. Cabe señalar que el esquema que propuso Dommel es utilizado para resolver redes lineales, por lo tanto se han venido proponiendo modificaciones de su esquema para poder incorporar elementos no lineales y variantes en el tiempo [Zamora *et al.* 2005].

Los siguientes pasos nos llevan, de manera general, a la solución de las ecuaciones asociadas al modelo de los circuitos eléctricos:

1. Encontrar el punto inicial de operación.
2. Determinar el primer paso de integración
   2.1 Discretizar las derivadas, formando modelos aproximados.
      2.1.1 Linealizar los elementos no lineales formando modelos aproximados.
         2.1.1.1 Formar las ecuaciones lineales asociadas con el circuito aproximado.
         2.1.1.2 Resolver las ecuaciones lineales.
      2.1.2 Iterar hasta obtener una solución o reducir el paso de integración e intentar nuevamente
   2.2 Analizar la convergencia.
3. Si la convergencia es adecuada seguir adelante en el tiempo, de lo contrario reducir el paso de integración e intentar nuevamente a partir de este paso.

1.1.2 Métodos de análisis

Cuando se tiene el sistema de ecuaciones que describen el comportamiento dinámico de una red eléctrica, existen tres tipos de métodos de análisis: métodos en el dominio del tiempo, métodos en el dominio de la frecuencia y métodos híbridos.

En los métodos en el dominio del tiempo, el comportamiento periódico de la red se determina mediante la integración del conjunto de Ecuaciones Diferenciales de Estado que describen su dinámica. Los elementos no lineales, variantes en el tiempo o cargas también se representan por medio de Ecuaciones Diferenciales de Estado. Ya que se cuenta con el conjunto de ecuaciones, éstas se integran con algún método de integración. Ya sea explícito, implícito o predictor-corrector [Chua y Pen-Min 1975]. Con este método se obtienen implícitamente, en la solución, los compontes armónicos asociados a las formas de onda de los diversos elementos del sistema. La desventaja es que el número de periodos completos de integración, para llegar al estado periódico estacionario, se puede incrementar de forma excesiva cuando hay dispositivos con poco o nulo amortiguamiento en la red.

El método de análisis en el domino de la frecuencia es uno de los más utilizados ya que, a diferencia del método en el dominio del tiempo, se facilita en gran manera la construcción de las ecuaciones que describen la dinámica de la red eléctrica a analizar. Los elementos no lineales y variantes en el tiempo se pueden incorporar en este análisis por medio de sus equivalentes Norton [Arrillaga *et al.* 1995]. Las desventajas de este método son: el gran esfuerzo computacional requerido para llegar a la solución y la pérdida de precisión del método a cambio de un menor número de requerimientos computacionales.

Las ventajas de los métodos anteriores son utilizadas en el método de análisis híbrido. En [Usaola 1990] se presenta un método de análisis híbrido que toma la parte lineal de la red y la analiza en el dominio de la frecuencia, después toma la parte no lineal de la red y la analiza en el dominio del tiempo.

1.1.3 Herramientas aplicadas en el análisis de sistemas eléctricos de potencia

1.1.3.1 Técnicas de acercamiento rápido al estado estable periódico

Al analizar redes eléctricas, uno de los mayores problemas para determinar de una manera rápida el Estado Estable Periódico (EEP) se encuentra cuando dicha red tiene componentes no lineales y variantes en el tiempo. Para resolver este problema se han propuesto diversos métodos, los cuales se denominan técnicas de acercamientos rápido al Estado Estable Periódico.

Los métodos de acercamiento rápido al EEP son clasificados en métodos de: Fuerza Bruta, Perturbación, Balance Armónico y Disparo.

Los métodos de Fuerza Bruta realizan la integración del conjunto de Ecuaciones Diferenciales de Estado que modelan el comportamiento del sistema hasta que llegan al Estado Estacionario Periódico. El inconveniente de este método es que se necesitan demasiados ciclos completos de integración para llegar al EEP, los cuales se pueden incrementar en sistemas con elementos con poco o nulo amortiguamiento.

En los métodos de perturbación se realiza un proceso iterativo partiendo de una solución inicial, la cual se obtiene linealizando la ecuación que describe el comportamiento del sistema.

Los métodos de balance armónico representan cada variable de estado con una serie de Fourier que satisfaga el requisito de periodicidad [Wylie 1951]. Por medio de un algoritmo de optimización se ajustan los coeficientes de las series de Fourier para que las ecuaciones del sistema se satisfagan con el mínimo error [Sundaram 1996]. La ventaja de estos métodos es que evitan los procesos de integración numérica de EDO's. La desventaja es el gran número de variables a ser optimizadas, ya que esto requiere de un gran esfuerzo computacional y del uso de recursos computacionales.

En los métodos de disparo se integra un sistema de ecuaciones $x = f(x,t)$ en un periodo completo de tiempo *T*, el objetivo es encontrar una condición inicial del vector $x(0)$ tal que satisfaga $x(T) = x(0)$. Uno de los primeros trabajos que buscaba encontrar el vector $x(0)$ fue presentado en [Aprile y Trick 1972], en donde se recuelve este problema usando un método Newton-Raphson [Hombeck 1975]. Posteriormente en [Nakhla y Branin 1977] se propone una técnica basada en un gradiente para encontrar la aproximación de $x(0)$. En [Skelboe 1980] se proponen algoritmos de extrapolación para llegar al EEP de redes eléctricas no lineales. En [Semlyen y Medina 1995] se hace la propuesta de tres métodos Newton de extrapolación del Ciclo Límite: Diferenciación Numérica (DN), Aproximación Directa (AD) y Matriz Exponencial (ME), los cuales están basados en el concepto del Plano Poincaré. Estas técnicas en particular han sido utilizadas en el análisis del EEP de redes eléctricas con componentes no lineales y variantes en el tiempo.

1.1.3.2 Procesamiento en paralelo

El procesamiento en paralelo es una forma de procesamiento en la cual se pueden llevar a cabo muchas instrucciones de forma simultánea. El procesamiento en paralelo se basa en el principio de poder dividir casi cualquier problema por muy grande que sea en problemas más pequeños, los cuales pueden ser procesados en paralelo. Actualmente el procesamiento en paralelo se ha vuelto el paradigma dominante en la arquitectura de computadoras, sobre todo en la forma de procesadores multi-núcleo. Esto se debe a dos grandes factores, el primero es la posibilidad de resolver problemas de mayor envergadura y el segundo tiene que ver con los avances en las tecnologías VLSI (*Very Large Scaled of Integration*).

El procesamiento en paralelo es la solución que hay para las limitaciones que presentan las tecnologías basadas en uni-procesadores. Estas limitaciones son: las frecuencias de reloj que no pueden superar la velocidad de la luz y la reducción del tamaño físico de los procesadores está en función del tamaño de las moléculas [Jin 1994].

Con la reducción del tiempo de cómputo se espera tener la posibilidad de realizar estudios en tiempo real, incrementar el tamaño de los problemas y el nivel de detalle de los modelos que representa a los sistemas eléctricos de potencia [Lemaitre y Thomas 1996].

1.2 Objetivo

Aplicar técnicas de procesamiento en paralelo basadas en PVM y Multithreading a la solución en Estado Estable de redes eléctricas de gran escala con componentes variantes en el tiempo, tanto lineales como no lineales.

1.3 Justificación

En la planeación de Sistemas Eléctricos de Potencia es de gran utilidad contar con simulaciones precisas que permitan conocer el comportamiento de una red eléctrica sujeta a una diversidad de condiciones de operación, tales como: fallas, diferentes topologías, cambios en los parámetros de la misma, etc. Por lo que contar con una herramienta que permita hacer estas simulaciones en forma rápida y eficiente es de gran utilidad.

1.4 Metodología

La metodología a seguir durante la elaboración de esta tesis será:

1) Se revisará bibliografía referente a la simulación digital de sistemas eléctricos, las técnicas de acercamiento rápido y las técnicas de procesamiento en paralelo.
2) Se implementará una propuesta de paralelización del método de acercamiento rápido al estado periódico estacionario en dos plataformas distintas de procesamiento en paralelo: PVM y Multithreading.
3) Se validarán los resultados obtenidos usando el simulador ATP-DRAW.
4) Se evaluarán distintos casos de estudio.
5) Se hará la comparación entre un esquema secuencial y los esquemas paralelos del método de diferenciación numérica.
6) Conclusiones.

1.5 Contenido de la Tesis

En el Capítulo 1 se da una introducción de este trabajo así como una breve reseña de la simulación digital de sistemas eléctricos. También se habla acerca de los métodos de análisis de Sistemas Eléctricos de Potencia y de las herramientas que se aplican para dicho análisis.

En el Capítulo 2 se detallan las distintas técnicas de acercamiento rápido al ciclo límite, en especial la de Diferenciación Numérica.

En el Capítulo 3 se abordan dos plataformas de procesamiento en paralelo evaluando las ventajas y desventajas de las mismas.

En el Capítulo 4 se hace una propuesta de paralelización del método de Diferenciación Numérica para las dos plataformas de procesamiento en paralelo.

En el Capítulo 5 se analizarán distintos casos de estudio para comprobar cuál de los esquemas de procesamiento en paralelo es el más eficiente.

En el Capítulo 6 se darán las conclusiones y sugerencias de trabajos de investigación futuras a partir de este trabajo.

Capítulo 2

Técnicas de Acercamiento Rápido al Ciclo Límite

2.1 Introducción

La simulación computacional precisa ha ayudado a reducir el costo y el tiempo requerido para simular el comportamiento de: sistemas de generación, transmisión y utilización de energía eléctrica. Esto se debe a que el diseño se puede corregir y su funcionamiento puede ser reajustado de manera rápida y más económica usando sistemas computacionales. La principal dificultad en la simulación de redes eléctricas radica en calcular los valores en el Estado Estable (EE), en particular en aquéllas donde se tiene un pobre o nulo amortiguamiento.

Los programas de simulación de circuitos más comúnmente utilizados, como SPICE [nagel75] y ASTAP [weeks73], se basan en algoritmos de integración numérica que son ideales para el análisis transitorio de circuitos no-lineales, pero estos algoritmos a menudo no pueden calcular de manera eficiente y precisa los valores en el EE que son de interés para un diseñador de circuitos analógicos, tampoco pueden simular eficientemente circuitos que contengan dispositivos distribuidos. Los programas de simulación basados en el análisis fasorial, tales como Touchstone®, están disponibles de manera comercial y pueden calcular fácilmente valores en el EE e incluyen dispositivos distribuidos. Sin embargo, el análisis fasorial requiere que todo el circuito sea lineal y por lo tanto no se aplica a circuitos analógicos inherentemente no-lineales como multiplicadores o mezcladores y, desde luego, el enfoque no puede usarse para calcular la distorsión armónica. Proveer a los diseñadores de circuitos analógicos y microondas con programas de simulación que puedan calcular

eficientemente y de manera precisa la respuesta en EE de circuitos no-lineales, que muy seguido contienen dispositivos distribuidos, es un desafiante e importante problema.

2.2 El Método Estándar de Simulación de Circuitos

El comportamiento de un circuito eléctrico construido con resistores, capacitores, y fuentes de corriente puede describirse con un sistema de ecuaciones que involucra voltajes nodales, corrientes resistivas y cargas capacitivas. El sistema de ecuaciones puede ser construido de una descripción del circuito utilizando el análisis nodal [desoer69], que involucra el aplicar la ley de corrientes de Kirchoff (LCK) a cada uno de los nodos del circuito, y aplicando las ecuaciones constitutivas a cada elemento del circuito. El sistema de ecuaciones generado tiene la siguiente forma,

$$\frac{d}{dt}q(v(t)) = -i(v(t)) - u(t) \tag{2.1}$$

Donde *N* es el número de nodos que hay en el circuito excluyendo el nodo de referencia, $q(v(t)) \in IR^N$ es el vector de la suma de las cargas capacitivas en cada nodo, $i(v(t)) \in IR^N$ es el vector de la suma de las corrientes resistivas en cada nodo, $u(t) \in IR^N$ es el vector de corrientes de entrada y $v(t) \in IR^N$ es el vector de voltajes nodales.

La técnica del análisis nodal puede ser extendida con facilidad para que incluya circuitos con inductores y fuentes de voltaje usando el análisis nodal modificado [ho75], manteniendo la forma de (2.1). Las variables entonces se convierten en una mezcla de voltajes nodales y dispositivos de corriente, y la parte izquierda de (2.1) se convierte en la derivada en el tiempo del vector de cargas capacitivas y flujos inductivos.

Encontrar la solución de (2.1) sobre un intervalo de tiempo específico partiendo de una condición inicial específica es un *problema de valor inicial* y calcular su solución es llamado con frecuencia *análisis transitorio*. El enfoque para el análisis transitorio usado en programas como SPICE [nagel75] y ASTAP [weeks73] consiste en aplicar un algoritmo de

integración numérica como la regla trapezoidal para aproximar (2.1) a una secuencia de ecuaciones algebraicas implícitas. Proveyendo un paso de integración h, el algoritmo de integración trapezoidal aplicado a (2.1) nos da:

$$q(v(t+h)) - q(v(t)) = -½h[i(v(t+h)) + u(t+h) + i(v(t)) + u(t)] \tag{2.2}$$

donde $v(t)$ se conoce y la ecuación sebe ser resuelta para calcular $v(t+h)$. Nótese que los pasos de integración se seleccionan de tal manera que (2.2) dé una aproximación razonable a $q(v(t+h))$ dado $q(v(t))$.

El algoritmo iterativo de Newton-Raphson usualmente se utiliza para resolver el sistema algebraico implícito no-lineal dado por (2.2). La ecuación Newton-Rapshon de iteración para resolver sistema general no-lineal de la forma $F(x) = 0$ es

$$J_F\left(x^{(j)}\right)\left(x^{(j+1)} - x^{(j)}\right) = F(x^{(j)}) \tag{2.3}$$

donde $F(x^{(j)})$ se refiere al residuo Newton-Rapshon, J_F es la matriz Jacobiana de F con respecto a x y j es el índice de iteración.

Si el algoritmo de Newton-Rapshon se utiliza para resolver (2.2), para $v(t+h)$ el residuo $F(v^{(j)}(t+h))$ sería:

$$\begin{aligned} F\left(v^{(j)}(t+h)\right) & \\ &= q\left(v^{(j)}(t+h)\right) - q(v(t)) \\ &+ ½h\left[i\left(v^{(j)}(t+h)\right) + u(t+h) + i(v(t)) + u(t)\right] \end{aligned}$$

y el Jacobiano de $F(v^{(j)}(t+h))$, $J_F\left(v^{(j)}(t+h)\right)$ sería:

$$J_F\left(v^{(j)}(t+h)\right) = \frac{\partial q(v^{(j)}(t+h))}{\partial v} + ½h\frac{\partial i(v^{(j)}(t+h))}{\partial v}$$

Entonces $v^{(j+1)}(t+h)$ se deriva de $v^{(j)}(t+h)$ resolviendo el sistema lineal de ecuaciones

$$J^F\left(v^{(j)}(t+h)\right)\left[v^{(j+1)}(t+h) - v^{(j)}(t+h)\right] = -F(v^{(j)}(t+h)) \tag{2.4}$$

utilizando una de las formas de eliminación Gaussiana. La iteración continua hasta que se logre llegar a una convergencia, la cual es $\left|v^{(j+1)}(t+h) - v^{(j)}(t+h)\right| < \varepsilon$ y que $F(v^{(j)}(t+h))$ esté cercano a cero.

2.3 Estado Estable

Aunque la mayoría de los programas de simulación de circuitos se enfocan al análisis transitorio, el comportamiento en el estado estable de los circuitos analógicos y microondas es típicamente de vital interés para un diseñador. Esto se debe a que ciertos aspectos del desempeño del sistema son fáciles de caracterizar y verificar en estado estable. Ejemplos de valores que se miden mejor cuando un circuito se encuentra en estado estable incluyen la distorsión, potencia, frecuencia, ruido y características de transferencia tales como la ganancia y la impedancia.

En términos generales una solución en estado estable de una ecuación diferencial es aquella que se aproxima asintóticamente mientras el efecto de la condición inicial desaparece. Una ecuación diferencial puede que no tenga una solución en el estado estable o que tenga cualquier cantidad de soluciones en estado estable. Si hay múltiples soluciones en estado estable, el estado estable que se aproxima asintóticamente dependerá de la condición inicial. En particular, para cada solución en estado estable debe corresponder una región de atracción para la cual si la condición inicial está contenida en la región asociada, entonces la solución se aproxima a la solución en estado estable dada. Por ejemplo, considere el sistema de ecuaciones diferenciales que describe al circuito de un biestable. Hay más de una solución en estrado estable, ya sea un inversor o el otro produce un uno lógico al cual se hará la aproximación dependiendo del estado inicial. Además, para el caso

de circuitos inversores típicos, la región de atracción para una solución en estado estable incluye todas las condiciones iniciales dentro de una distancia no cercana a cero de la solución en estado estable. Esto implica que si la solución es ligeramente perturbada por una solución en estado estable dada, regresará a la misma solución en estado estable. Tal solución en estado estable es nombrada asintóticamente estable. Nótese que esta definición excluye a los integradores no-lineales sin pérdidas a los osciladores armónicos de tener solución en estado estable. La solución de esos circuitos no es asintóticamente estable.

Hay varios tipos diferentes de comportamiento en estado estable que son de interés. El primero es el estado estable en CD. Aquí la solución es un punto de equilibrio del circuito y no varía con el tiempo. Circuitos lineales asintóticamente estables alimentados por fuentes sinusoidales eventualmente presentan una solución sinusoidal en estado estable, la cual se caracteriza por ser puramente sinusoidal excepto tal vez por algún *offset* o desplazamiento de CD. Si la respuesta en estado estable de un circuito consiste solamente de una combinación lineal de un desplazamiento de CD y de un posible número infinito de sinusoides armónicamente relacionadas, se dice que el circuito se encuentra en un *estado estable periódico*. El estado estable periódico resulta de auto oscilaciones o como una respuesta a la variación periódica de las entradas. El periodo de solución es usualmente igual al que hay en la entrada, si existe, aunque en ocasiones ambos periodos serán múltiplos de algún periodo común. Si un circuito no-lineal es alimentado con varias fuentes periódicas a frecuencias desasociadas, el circuito tendrá típicamente una respuesta en estado estable que será *quasiperiódica.* Una respuesta *quasiperiódica* consiste en una combinación lineal de sinusoides a la frecuencia resultante de un conjunto finito de frecuencias fundamentales y sus armónicos. Las frecuencias fundamentales regularmente igualan la frecuencia de las señales de entrada, sin embargo en ocasiones son múltiplos pares.

Hay repuestas en estado estable que no encajan en ninguna de las clasificaciones anteriores. Esto ocurre cuando las fuentes de alimentación no son quasiperiódicas o cuando el circuito es extraño. Ejemplos incluyen circuitos caóticos y circuitos con ruido como estímulo. Sólo se abordará el cálculo de soluciones periódicas y quasiperiódicas. Además,

la mayoría de los métodos descritos en esta tesis no distinguen entre soluciones que son asintóticamente estables y aquellas que no lo son.

2.3.1 Calculando el Estado Estable

La mayoría de los circuitos analógicos prácticos, al menos aquéllos que están razonablemente bien diseñados alcanzarán un estado estable periódico o quasiperiódico. Esto presume que un circuito razonable es lo "suficientemente estable" y excluye ejemplos como integradores sin pérdidas y a los resonadores. Es posible calcular la respuesta en estado estable de estos circuitos "suficientemente estables" por medio de simuladores transitorios como SPICE o ASTAP mediante la integración numérica de las ecuaciones diferenciales que describen al circuito hasta que el comportamiento transitorio se extinga a partir de alguna condición inicial arbitraria. Sin embargo, este enfoque bastante general se puede volver impráctico cuando las frecuencias en la respuesta en estado estable son mucho más grandes que el ritmo con el que el circuito se aproxima al estado estable o cuando la proporción entre las frecuencias más alta y más baja presentes en la solución del estado estable es grande. En estos casos, el número de pasos de tiempo utilizado por el algoritmo de integración numérica sería enorme, debido a que al intervalo de tiempo sobre el cual las ecuaciones diferenciales deben ser integradas está dado por la frecuencia más baja o por el tiempo que le toma al circuito alcanzar el estado estable, pero el tamaño de los pasos de tiempo está limitado por la más alta frecuencia de respuesta.

Hay una amplia variedad de métodos que calculan directamente la solución en el estado estable de manera más eficiente que realizando integración numérica en las ecuaciones diferenciales que describen al circuito a partir de alguna condición inicial arbitraria.

2.4 Métodos en el Dominio del Tiempo

Si existe una solución de una ecuación diferencial, ella representa un conjunto de puntos en el plano cartesiano. Se utiliza una sucesión de puntos distintos cuyas coordenadas (Figura 2.1) se aproximen a las coordenadas de los puntos de la curva real de solución.

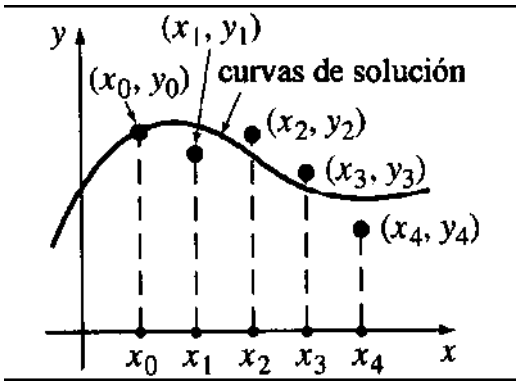

Figura 2.1 Curva de solución de una Ecuación Diferencial Ordinaria

Estos métodos resuelven ecuaciones diferenciales de primer orden tal como:

$$\frac{dy}{dx} = f(x,y), y(x_0) = y_0 \tag{2.5}$$

2.4.1 Métodos explícitos

Los métodos explícitos son métodos que se utilizan para resolver ecuaciones diferenciales ordinarias, en los cuales el paso en la aproximación de la curva de solución se hace en función del paso anterior, es decir aproxima un punto y en base a este punto se genera el siguiente punto de la aproximación. Uno de los métodos explícitos más utilizados y más sencillos para aproximar soluciones del problema de valor inicial es el método de Euler,

$$y' = f(x,y), \qquad y(x_0) = y_0$$

Se llama método de Euler o método de las tangentes**.** Aplica el hecho que la derivada de una función $y(x)$**,** evaluada en un punto x_0, es la pendiente de la tangente a la gráfica de $y(x)$ en este punto. Como el problema de valor inicial establece el valor de la derivada de la solución en (x_0, y_0), la pendiente de la tangente a la curva de solución en este punto es $f(x_0, y_0)$. Si recorremos una distancia corta por la línea tangente obtenemos una aproximación a un punto cercano de la curva de solución. A continuación se repite el proceso en el punto nuevo. Para formalizar este procedimiento se emplea la linealización.

$$L(x) = y'(x_0)(x - x_0) + y_0 \tag{2.6}$$

de y(x) en x = x_0. La gráfica de esta linealización es una recta tangente a la gráfica de y = y(x) en el punto (x_0, y_0). Ahora se define *h* como un incremento positivo sobre el eje x (Figura 2). Reemplazamos x con $x_1 = x_0 + h$ en (2.6) y llegamos a

$$L(x_1) = y'(x_0 + h - x_0) + y_0 = y_0 + hy_0' \tag{2.7}$$

o sea

$$y_1 = y_0 + hf(x_0, y_0)$$

en donde $y_0' = y'(x_0) = f(x_0, y_0)$ y $y_1 = L_1(x)$. El punto (x_1, y_1) sobre la tangente es una aproximación al punto $(x_1, y(x_1))$ en la curva de solución; esto es, $y_1 \approx y(x_1)$ es una aproximación lineal local de $y(x)$ en x_1. La exactitud de la aproximación depende del tamaño h del incremento. Si a continuación repetimos el proceso, identificando al nuevo punto de partida (x_1, y_1) como (x_0, y_0) de la descripción anterior, obtenemos la aproximación:

$$y(x_2) = y(x_0 + 2h) = y(x_1 + h) \approx y_2 = y_1 + hf(x_1, y_1)$$

La consecuencia general es que

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{2.8}$$

en donde $x_n = x_0 + nh$

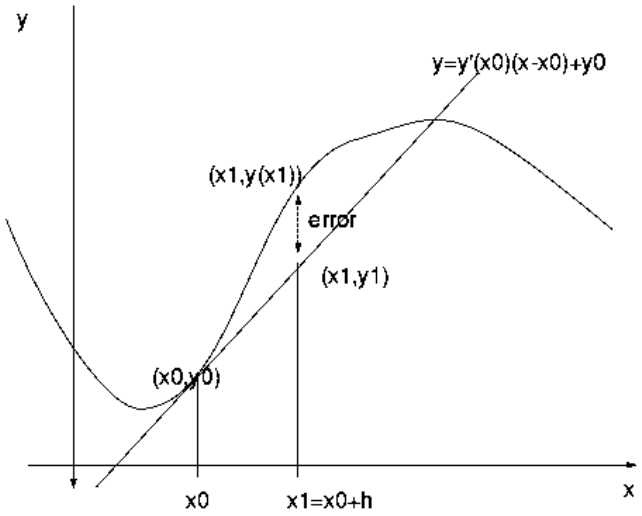

Figura 2.2 Método de Euler

Dentro de los métodos explícitos también se tienen los métodos de Runge-Kutta. La expresión del método de Runge-Kutta de cuarto orden es,

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.9}$$

donde:

$$k_1 = hf(t_n, x_n)$$

$$k_2 = hf\left(t_n + \frac{1}{2}h, x_n + \frac{1}{2}k_1\right)$$

$$k_3 = hf\left(t_n + \frac{1}{2}h, x_n + \frac{1}{2}k_2\right)$$

$$k_4 = hf(t_n + h, x_n + k_3)$$

Para resolver un conjunto de Ecuaciones Diferenciales Ordinarias se resume el método de Runge-Kutta de cuarto orden en el diagrama de flujo de la Figura 2.3

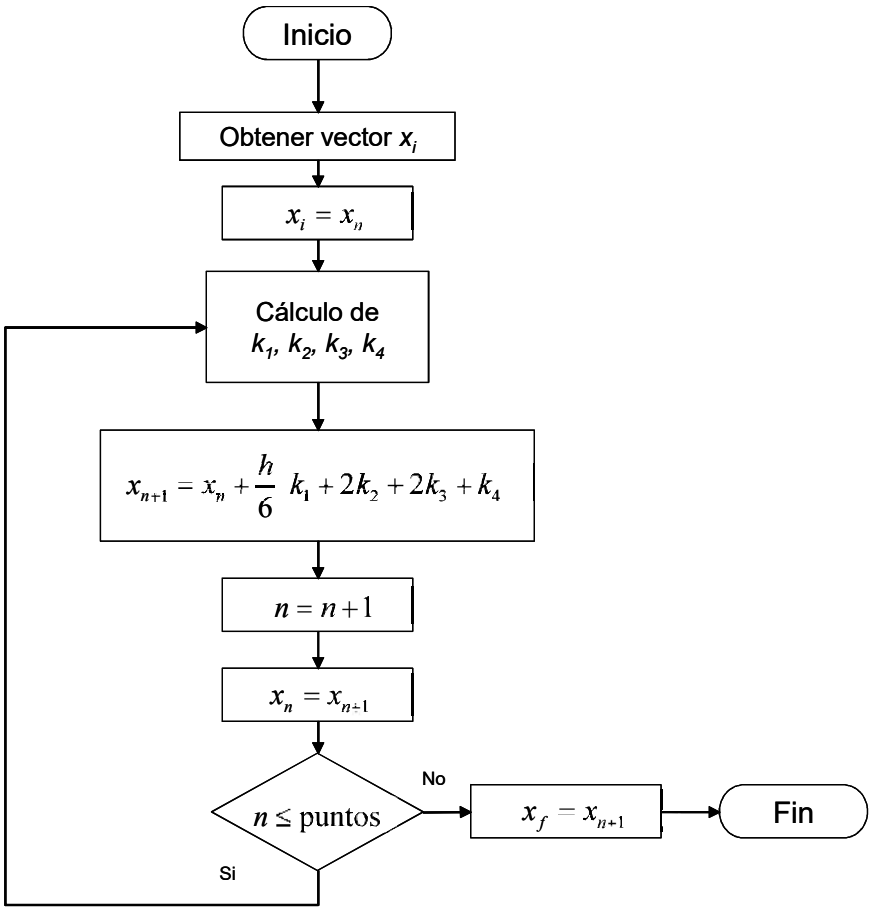

Figura 2.3 Diagrama de flujo del método de Runge-Kutta de cuarto orden

La ventaja que presentan los métodos explícitos es la rapidez con la que llegan a una solución, esta velocidad sólo se ve limitada por el tamaño del paso elegido a la hora de obtener una respuesta.

Las desventajas son muchas, empezando que este tipo de método es el más inexacto ya que sólo traza líneas tangentes a la curva de la solución en base a los valores calculados con error y debido a que no cuenta con método corrector el resultado tiende a alejarse cada vez más de la solución real. Esto se compensa eligiendo un tamaño de paso muy pequeño, pero al hacer esto se pierde la ventaja de la velocidad de obtener una solución pero a cambio de una mejora en la solución.

2.4.2 Métodos implícitos

Un método implícito es en el que en cada paso se debe calcular y_{n+1} pero este valor está definido implícitamente (en función de si mismo). Como ejemplo podemos ver la regla trapezoidal la cual se basa en la integración la ecuación diferencial,

$$y' = f(x, y), x\epsilon[x_n, x_{n+1}]$$

Integrando, se obtiene

$$\int_{x_n}^{x_{n+1}} y'(x)dx = \int_{x_n}^{x_{n+1}} f\big(x, y(x)\big)dx$$

Si se aproxima el valor de la segunda integral al área del rectángulo con una altura de valor f en (x_n, y_n), se obtiene $y_{n+1} - y_n = hf(x_n, y_n)$, que corresponde a la fórmula del método de Euler.

Aproximando la integral al área del trapecio es posible obtener una mejor estimación, que corresponde a la fórmula del método del trapecio:

$$y_{n+1} - y_n = \frac{h}{2}\big(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\big)$$

El método del trapecio es un método implícito**.** En cada paso se debe calcular y_{n+1} pero este valor está definido implícitamente (en función de si mismo) por la ecuación (4). Si f es una función no lineal, se debe resolver una ecuación (o un sistema) no lineal en cada paso.

La resolución implicará entonces el uso de un método iterativo.

Una posible iteración de punto fijo para determinar y_{n+1} es utilizando el Método de Newton-Rapshon

$$y_{n+1} = y_n - \frac{f(x_n, y_n)}{f'(x_n, y_n)}$$

En donde es necesario obtener la derivada de la función *f* y determinar un criterio de convergencia.

Para comenzar la iteración con el método Newton-Raphson se necesita un valor inicial, que se puede obtener de los datos conocidos por ejemplo calculando un paso de un método explícito. Puede utilizarse el valor de un paso del método de Euler como valor inicial,

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n)$$

Se puede apreciar que el método de la regla trapezoidal tiende a aproximarse más a la solución real. Lo anterior se puede observar en la siguiente gráfica, en donde se hace una comparación entre el método de Euler y el del trapecio.

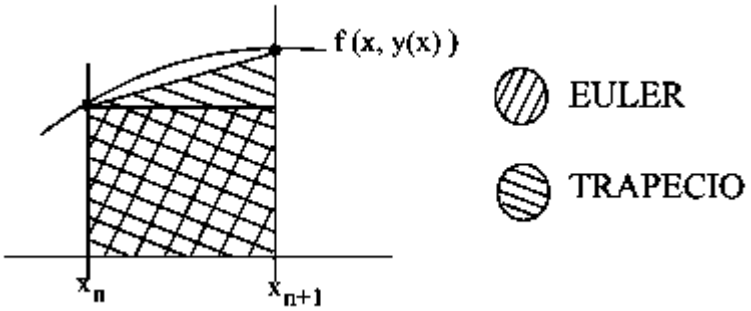

Figura 2.4 Comparación entre aproximaciones por el método de Euler y el método trapezoidal

El proceso para resolver un conjunto de Ecuaciones Diferenciales Ordinarias por medio del método de la regla trapezoidal se muestra en el diagrama de flujo de la Figura 2.5.

La desventaja de los métodos implícitos es que para obtener un resultado se requiere de mayor tiempo, debido a las iteraciones con el método de Newton-Raphson, y de mayor esfuerzo computacional.

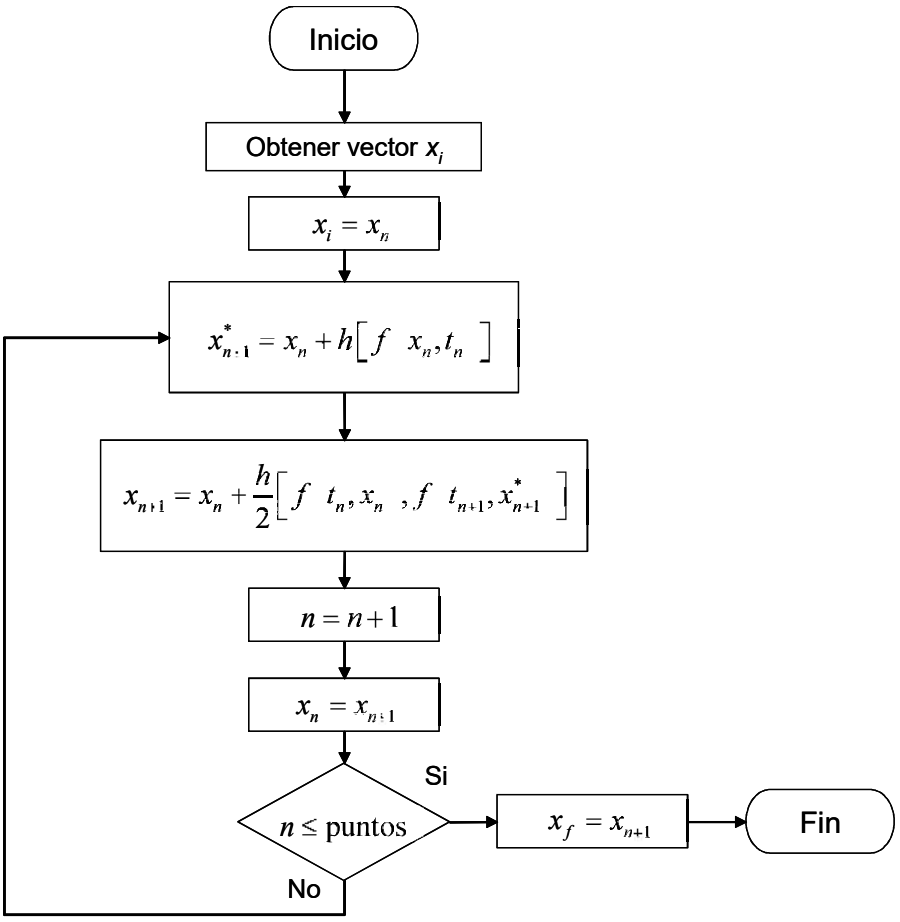

Figura 2.5 Diagrama de flujo del método de la Regla Trapezoidal

Las ventajas son que se aproxima a la solución real y no tiende a desviarse mucho de dicha solución, a diferencia del método de Euler que no tiene predictor, esto se ve a continuación.

2.4.3 Métodos predictor-corrector

En un método implícito se pudo observar que es necesario contar con 2 ecuaciones para encontrar una aproximación de la solución. La primera ecuación que se utiliza es la fórmula necesaria para determinar el valor inicial para la iteración, ésta se denomina predictor; la segunda ecuación es la fórmula de iteración, a la cual se le denomina corrector. El procedimiento general se llama método predictor - corrector.

El criterio de parada de la iteración puede involucrar el control de la diferencia $y_{n+1}^{(k+1)} = y_{n+1}^{(k)}$ respecto de una tolerancia establecida, o fijando el número de iteraciones. La última idea es la más común para los métodos más usuales. Es deseable utilizar un predictor apropiado de forma que baste con una sola iteración (del corrector) para lograr la precisión deseada. Son muy usadas las fórmulas de Adams - Bashforth como predictor, y la fórmula de Adams Moulton como corrector.

Este procedimiento llamado predictor-corrector tiene grandes ventajas, ya que los resultados que nos entrega se mantienen muy aproximados al valor real de la solución. Esto es porque "predice" y "corrige" el valor siguiente, y en base a esa predicción es como calcula el valor de corregido. Las desventajas son el tiempo que lleva en realizarse la aproximación, debido al número de iteraciones que realiza el método corrector.

Se puede observar que los métodos implícitos utilizan en gran manera el procedimiento predictor-corrector.

2.5 Definición técnica de acercamiento rápido

Un gran problema en el diseño de circuitos eléctricos en los cuales se requiere tener poco amortiguamiento se encuentra en calcular la respuesta en el estado estable cuando el circuito es impulsado por señales de entrada de "p" componentes de frecuencia distintas $\{\omega_1, \omega_2, \cdots, \omega_p\}$, donde $p \geq 2$. Asumimos que el circuito o sistema está descrito por un sistema implícito de ecuaciones diferenciales de la forma

$$f_j(\dot{x}, x, y;\ \omega_1 t, \omega_2 t, \cdots\ \omega_p t) = 0, \qquad j = 1, 2, \cdots, m+n \tag{2.10}$$

donde $\boldsymbol{x}$ es un vector de tamaño n que representa las variables de estado, $\boldsymbol{y}$ es un vector de tamaño m que representa las demás variables que no son de estado y $f_j(\cdot)$ contiene p señales de entrada periódicas de frecuencias $\omega_1, \omega_2, \cdots, \omega_p$ respectivamente.

Dado un valor inicial x_0, (2.10) posee una única solución periódica casi asintótica, es decir:

$$x(t) = x_{tr}(t) + x_{ss}(t) \tag{2.11}$$

donde

$$x_{tr}(t) \to 0 \; cuando \; t \to \infty \tag{2.12}$$

se conoce como la componente transitoria, y

$$x_{ss}(t) = a_0 + \sum_{k=1}^{M} \{a_{2k-1} \cos v_k t + a_{2k} \operatorname{sen} v_k t\} \tag{2.13}$$

se conoce como la respuesta en el estado estable, en donde la sumatoria se toma de todas las posibles frecuencias

$$v_k \triangleq m_{1k}\omega_1 + m_{2k}\omega_2 + \cdots + m_{pk}\omega_p \tag{2.14}$$

generadas por la frecuencia base $\omega_1, \omega_2, \cdots, \omega_p$ y los enteros $\{m_{ik}\}$

Nótese que (2.13) no es una serie de Fourier ordinaria, ya que su espectro de frecuencias $\{v_1, v_2, \cdots, v_M\}$ no está relacionado armónicamente. De hecho $x_{ss}(t)$ ni siquiera sería periódica si la frecuencia base $\{\omega_1, \omega_2, \cdots, \omega_p\}$ fuera inconmensurable. (2.13 se conoce como una función quasi periódica.

El objetivo de la tesis es utilizar un método eficiente para calcular la respuesta en estado estable de $x_{ss}(t)$.

Los métodos existentes actuales para calcular $x_{ss}(t)$ pueden ser clasificados en cuatro categorías:

2.5.1 Método de fuerza bruta.

Esta técnica resuelve (2.10) con integración numérica (partiendo de un valor inicial arbitrario x_0 hasta alcanzar el estado estable. La Figura 2.6 muestra el diagrama del flujo del método FB.

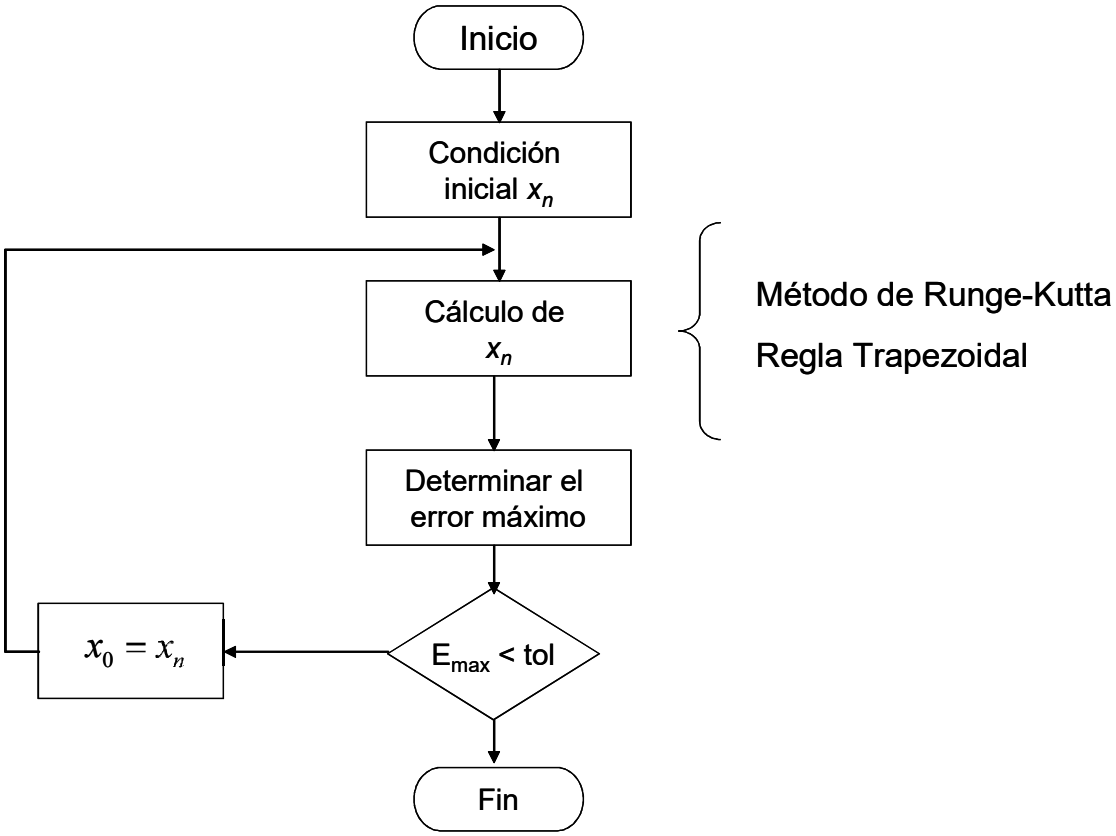

Figura 2.6 Diagrama de flujo del método de Fuerza Bruta

Aunque este método es considerablemente general es extremadamente costoso para circuitos con poco amortiguamiento en los cuales se requiere una gran cantidad de tiempo para que la componente transitoria se extinga.

Además, si la frecuencia base es inconmensurable $x_{ss}(t)$ no es periódico y es difícil determinar cuándo se ha llegado al estado estable.

2.5.2 Método de perturbación

Esta técnica resuelve (2.10) por medio de iteraciones partiendo de una solución inicial generalmente obtenida de la solución de una ecuación linealizada.

Desafortunadamente este método funciona sólo con circuitos casi lineales en donde la no-linealidad es a menudo extremadamente débil. Para los circuitos que dependen de una esencial de la no-linealidad este método se hace altamente impreciso, dejando al un lado el hecho de que la iteración muy a menudo no llega a converger.

2.5.3 Método de balance armónico

Esta técnica resuelve (2.10) aproximando la solución a una serie trigonométrica finita, posteriormente hace un balance de todos los términos que tienen componentes de frecuencia idénticas.

Aunque en teoría es muy interesante, este método es con frecuencia extremadamente lento debido a que las diversas componentes de frecuencia son estimadas mediante un análisis de Fourier multidimensional.

2.5.4 Método de disparo

Esta técnica resuelve (2.10) encontrando primeramente un valor inicial para x_0 (a menudo utilizando el método Newton-Raphson) tal que la solución iniciando de x_0 sea periódica, es decir, que no tenga ninguna componente transitoria.

Hay dos grandes problemas asociados con este método.

(a) No puede ser utilizado cuando la solución no es periódica.

(b) Incluso si la solución fuera periódica, el periodo T es a menudo muchos órdenes de magnitud más grande que los periodos de las componentes de frecuencia individuales v_k; por esta razón, al realizar la integración numérica sobre este largo periodo T sería costosísimo.

2.6 Técnica de diferenciación numérica de Newton

Esta técnica está clasificada dentro de los métodos de disparo y el objetivo es encontrar una condición inicial del vector $x(0)$ tal que al integrar el sistema de ecuaciones $x = f(x,t)$, sobre un periodo de tiempo completo T, a partir de la condición inicial $x(0)$, se obtenga $x(T) = x(0)$.

La descripción general del conjunto de EDOs que describen la dinámica de un sistema está en términos de la ecuación diferencial

$$\dot{x} = f(x,t) \tag{2.15}$$

donde x es el vector de estado de m elementos x_k. La fuerza impulsora es periódica, así que $f(\cdot,t)$ será un vector T-periódico y por lo tanto la solución en estado estable $x(t)$ también lo será. Esto se puede representar como el ciclo límite para x_k en términos de otro elemento periódico de x o en términos de una función T-periódica, por ejemplo $sen(wt)$. Tal como se muestra en la Figura 2.7.

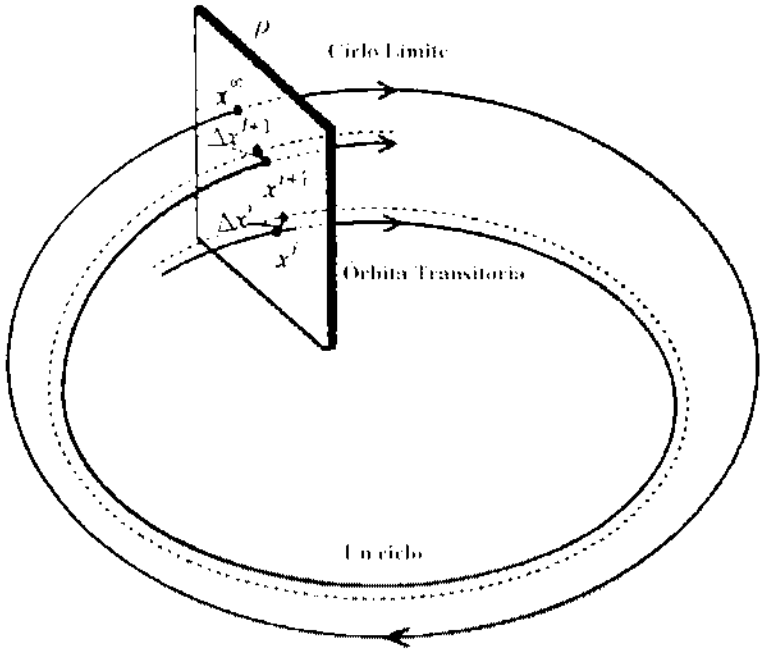

Figura 2.7 Orbita de un Vector de Estado *X*

Antes de alcanzar el Ciclo Límite, los ciclos de la órbita transitoria están cercanos a éste. Su posición está convenientemente descrita por su trazado en el plano de Poncaré ρ. Un solo ciclo mapea su punto de inicio x^i hasta su punto final x^{i+1} y mapea un segmento de perturbación $\mathbf{\Delta x}^{\mathrm{i}}$ (desde este Ciclo Base [Semlyen y Medina 1995]) a $\mathbf{\Delta x}^{\mathrm{i+1}}$, esto se puede ver en la Figura 2.7. Todos los mapeos cerca de un Ciclo Límite son Quasi-lineales así que el método de Newton o su aproximación pueden ser utilizados para obtener el punto x^{∞} para el Ciclo Límite. Esto es posible sin tomar en cuenta su estabilidad [Semlyen y Medina1995].

A fin de sacar ventaja de la linealidad que se tiene en las cercanías de un Ciclo Base, se puede linealizar (2.15) alrededor de una solución de $x(t)$ desde t_i hasta t_{i+T}. Esto resulta en el problema variacional:

$$\dot{\mathbf{\Delta x}} = \Delta f(x(t), t) = D_x f(x_t, t)\mathbf{\Delta x} = \mathbf{J}(t)\mathbf{\Delta x} \tag{2.16}$$

donde *J(t)* es la matriz Jacobiana (*T*-Periódica). La condición inicial es:

$$\mathbf{\Delta x}(t_i) = \mathbf{\Delta x}^{\mathrm{i}} \tag{2.17}$$

La Ecuación (2.17) es una EDO, lineal variante en el tiempo, con una solución de la forma cerrada:

$$\Delta x(t) = \exp\left(\int_{t_i}^{t} J(t)dt\right)\Delta x^{\mathrm{i}} \tag{2.18}$$

Esta ecuación claramente satisface (2.15). Para $t = t_i + T$, (2.17) se tiene:

$$\boldsymbol{\Delta x}^{\mathbf{i+1}} = \Phi\mathbf{\Delta x}^{\mathrm{i}} \tag{2.19}$$

Con

$$\Phi = \exp\left(\int_{t_i}^{t_1+T} J(t)dt\right)$$

(2.20)

Se puede ver que Φ resulta casi la misma para cualquier t_i tal que el mapeo cerca del ciclo límite está cerca de la linealidad.

La Ecuación (2.19) muestra que los segmentos de entrada son mapeados para corresponder a los segmentos de salida por medio de la matriz Φ. Por el conjunto de relaciones en el plano de Poincaré ilustrados por la Figura 2.7, se pretende identificar también la matriz C definida por medio de [Semlyen y Medina 1995] de la manera siguiente: de la Figura 2.7 se tiene que $\mathbf{\Delta x}^i = x^\infty - x^i$, con lo que $\mathbf{\Delta x}^{i+1} = x^\infty - x^{i+1}$. La sustitución de la expresión anterior en (2.19) y la solución para x^∞ resulta en,

$$x^\infty = x^i + C\left(x^{i+1} - x^i\right)$$

(2.21)

La cual es una estimación de la localización del Ciclo Límite con

$$C = (I - \Phi)^{-1}$$

(2.22)

La Ecuación (2.21) conduce a un proceso Newton si Φ y C son actualizadas en cada iteración usando (2.20) y (2.22). Esto se convierte en un proceso linealmente convergente si C es mantenida constante o se actualiza en alguna etapa del proceso iterativo después de su primera evaluación empleando el método de la regla trapezoidal.

El principal problema para encontrar de forma eficiente el Ciclo Límite es la identificación de la matriz Φ. Para esto es necesario tener inicialmente calculado el Ciclo Base $x(t)$ sobre un periodo *T*, comenzando desde x^i *(*Figura 2.7).

Considérese que el vector inicial Δx^i es una columna de la matriz identidad $\mathbf{I}$. Entonces, si todas las columnas son consideradas, (2.15) da

$$\Delta X^{i+1} = \Phi \tag{2.23}$$

Así, al integrar

$$\dot{\Delta x} = J(t)\Delta x \tag{2.24}$$

con vectores iniciales Δx^i siendo de forma secuencial las columnas de la matriz identidad $\mathbf{I}$, se producen directamente las columnas de Φ.

Esto requiere el conocimiento de la matriz Jacobiana $J(t)$. A menudo $J(t)$ puede calcularse analíticamente pero no siempre es el caso. En particular, en el caso de dispositivos conmutados es más fácil utilizar en (2.16) el incremento Δf en lugar de $J(t)\Delta x$. Esto implica que además de integrar (2.15) con condiciones iniciales x^i para obtener el Ciclo Base $x(t)$, también se debe integrar con el valor de perturbación inicial $x^i + \varepsilon e^i$, donde e^i es columna de *i* de la matriz identidad $\mathbf{I}$, y ε es un número pequeño, por ejemplo de 1.0^{-6} *p.u.* Al calcular las diferencias de los dos valores de *x* al final del ciclo, se obtienen las columnas de ΔX^{i+1} de (2.23), por ejemplo de $\varepsilon\Phi$. En el caso de entrada escalada hacia abajo, el cálculo de Φ requiere *m* cálculos, justo como en el caso de la matriz Jacobiana.

La Figura 2.8 muestra el diagrama de flujo del método DN, en donde se puede apreciar la forma en la que se realiza el cálculo de las columnas de la matriz de identificación Φ.

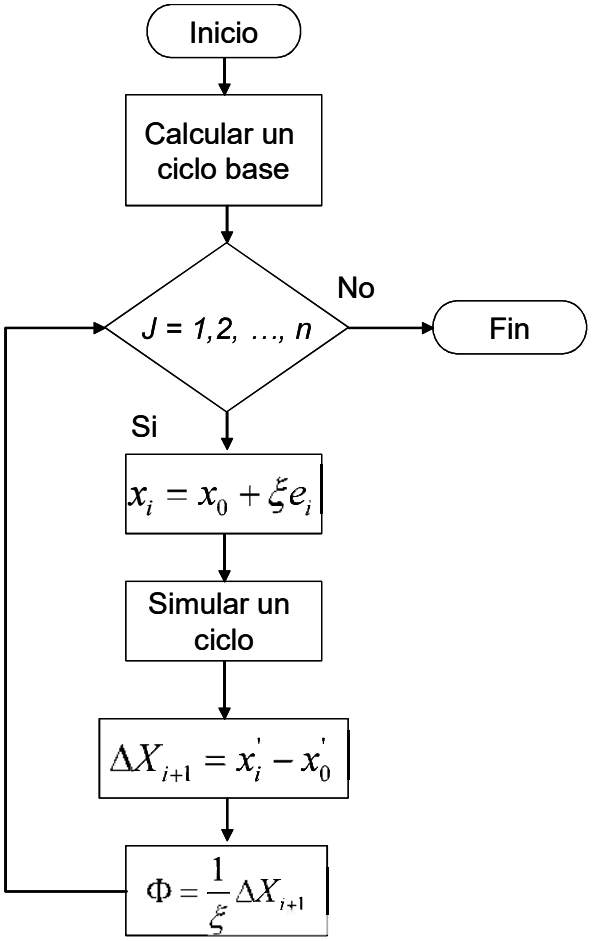

Figura 2.8 Diagrama de flujo del método Diferenciación Numérica

Capítulo 3

Procesamiento en Paralelo

3.1 Introducción

La demanda por computadoras más rápidas y más grandes se incrementa constantemente. Para satisfacer esta demanda los arquitectos computacionales han seguido dos enfoques generales. El primero emplea tecnología exótica en una arquitectura computacional serial bastante convencional. Este enfoque se ve aquejado por problemas en la manufactura, problemas de mantenimiento y altos costos. El segundo enfoque saca provecho del paralelismo inherente en muchos problemas. El enfoque en paralelo parece ofrecer la mejor estrategia a largo plazo porque mientras los problemas aumentan más y más oportunidades se originan para explotar el paralelismo inherente en los mismos datos. Varios sistemas altamente concurrentes o altamente paralelos están ahora disponibles de manera comercial.

3.2 Definición de procesamiento en paralelo

El procesamiento en paralelo se puede definir como muchos solucionadores de problemas operando simultáneamente o concurridamente para alcanzar una solución. Un sistema computacional de procesamiento en paralelo típicamente descompone un gran problema en muchos subproblemas. Luego lo soluciona con la íntima colaboración de una gran cantidad de procesadores interconectados.

El rápido desarrollo del procesamiento en paralelo ha sido motivado por dos fuerzas: 1) la cada vez mayor demanda para solucionar grandes problemas en diversos campos de aplicación y 2) los recientes grandes avances de la tecnología computacional, especialmente la tecnología de Integración a gran escala (VLSI). Además de contar con

muchas motivaciones, tales como: 1) Desempeño: se obtiene un incremente el poder mientras más procesadores se añadan. 2) Elegante crecimiento: permite al proveedor ofrecer un sistema con un rango de poder y al usuario le permite expandir su sistema. 3) Tolerancia al error: al contar con sistemas multiprocesador capaces de buscar errores y operar con configuraciones degradadas cuando hay fallas presentes.

La mayoría de las computadoras de propósito general en uso son diseñadas a partir de uniprocesadores con arquitectura tradicional. Esta arquitectura sufre de una serie de limitaciones físicas, tecnológicas y económicas. Se cree que, para un uniprocesador, el incremento en la frecuencia del reloj está limitado a la larga por la velocidad de la luz, y la reducción del tamaño físico está limitada al diámetro de las moléculas. Por lo tanto los uniprocesadores tradicionales son incapaces de proveernos toda la potencia computacional necesaria.

El procesamiento en paralelo es la única manera de superar las limitaciones de la arquitectura tradicional. Siguiendo esta dirección muchas arquitecturas computacionales nuevas han sido propuestas. Han abierto la posibilidad de construir sistemas computacionales en paralelo masivos que proveen ultra-alto rendimiento a costos razonables.

3.2.1 Eficiencia Relativa

La manera en la que se cuantifica la eficiencia de un algoritmo paralelo o secuencial es calculando la efectividad con la que utiliza los recursos computacionales de la computadora paralela. La eficiencia relativa se define como,

$$E_{relativa} = \frac{t_1}{t_p} \tag{3.1}$$

Donde t_1 es el tiempo que tarda el programa en ejecutarse en un solo elemento de proceso y t_p es el tiempo que tarda el programa en ejecutarse en p elementos de proceso.

3.2.2 Tiempo relativo

Así como se cuantifica la eficiencia de un algoritmo paralelo o secuencial también se cuantifica el tiempo de ejecución del programa con respecto al tiempo que tarda el programa en ejecutarse en un solo elemento de proceso. El tiempo relativo se define como,

$$t_{relativo} = \frac{t_p}{t_1} \tag{3.2}$$

3.3 Fundamentos de la arquitectura en paralelo

Una característica importante del cálculo en paralelo es la relación tan cercana entre los algoritmos en paralelo y las arquitecturas en paralelo. Dado el mismo problema los algoritmos en paralelo pueden desarrollarse basándose en diferentes modelos de cálculo los cuales requieren la arquitectura en paralelo para proveer diferentes modos de ejecución.

Muchos sistemas computacionales en paralelo se desarrollaron simultáneamente a sus arquitecturas, correspondiendo así de forma más eficiente a un respectivo tipo de problemas. Las otras se inclinaron a tener reconfigurables sus arquitecturas para encajar en diversas aplicaciones. Adicionalmente, en la actualidad, los sistemas en paralelo obtienen menos soporte de su software que de su hardware.

Las arquitecturas en paralelo se pueden clasificar en distintos tipos de acuerdo a sus modos de ejecución. Según la clasificación de Flynn, de una manera realista, tenemos tres tipos básicos de sistemas computacionales: flujo de Instrucciones Sencillas flujo de Datos Sencillos (SISD); flujo de Instrucciones Sencillas flujo de Datos Múltiples (SIMD) y flujo de Instrucciones Múltiples flujo de Datos Múltiples (MIMD). El paralelismo puede ser explotado en cada una de las categorías basándose en dos principios: operaciones superpuestas en el tiempo (llamado paralelismo temporal) y/o replicando recursos en el espacio (llamado paralelismo espacial).

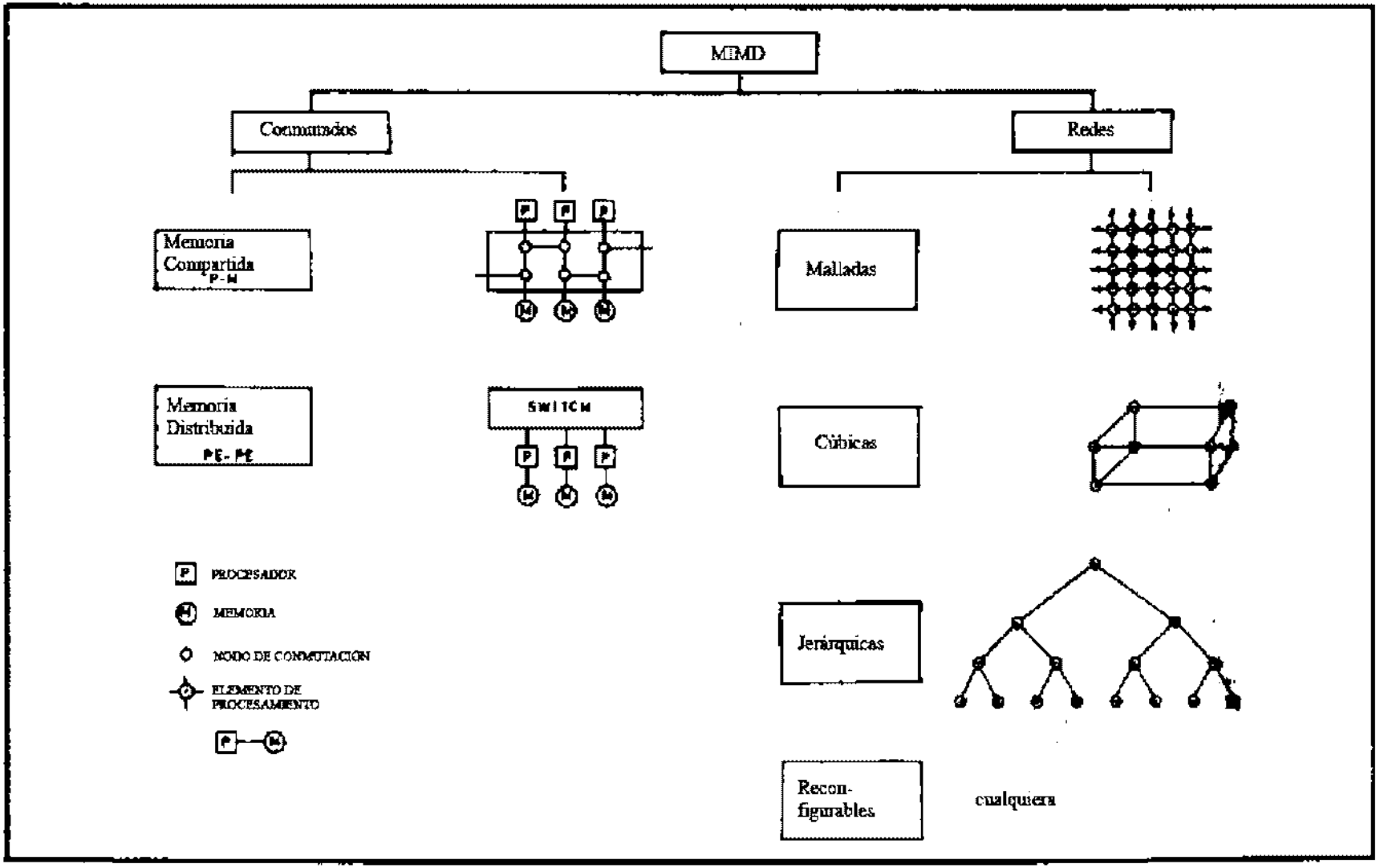

Figura 3.1 Arquitecturas de computadoras en paralelo

Los sistemas computacionales en paralelo incrementan la velocidad de cálculo efectiva, esto se logra permitiendo que varias instrucciones sean ejecutadas al mismo tiempo. Para estos sistemas el asunto más importante es el cómo son interconectadas las diversas unidades de procesamiento y de memoria. En el más alto nivel las computadoras en paralelo se subdividen en sistemas conmutados y sistemas en red (Figura 3.1).

3.3.1 Sistemas conmutados

Estos sistemas por lo regular son complejos. Pueden contar con varias etapas. El conmutador conecta en conjunto un número de módulos de proceso (p) y de memoria (m). Los sistemas conmutados adicionalmente son subdivididos de acuerdo al tipo de interconexiones en sistemas de bus, multi-etapa o crossbar. Dentro de los sistemas conmutados hay sistemas de memoria compartida y de memoria distribuida. En un sistema de memoria compartida una cantidad de procesadores son conectados por medio de la unidad de conmutación a otra cantidad de módulos de memoria independiente formando una memoria global común compartida. En un sistema de memoria distribuida cada procesador tiene su propia memoria local y da forma a un Elemento de Procesamiento (EP). El papel de la unidad de conmutación es el de interconectar los EPs.

3.3.2 Sistemas de red

Es estos sistemas una cantidad de EPs se conectan en conjunto a una red con una topología reconocible. Las topologías son malladas, cúbicas, jerárquicas y reconfigurables; y pueden ser utilizadas para seguir subdividiendo este tipo de sistemas. En las topologías malladas las redes que existen son malladas unidimensionales (ej. anillos) y malladas multidimensionales (ej. cuadrada, hexagonal y otras formas geométricas). En las redes cúbicas las variaciones oscilan entre redes hipercúbicas y redes de cubos-círculo-conectados en las que cada nodo del hipercubo es remplazado por un anillo (o círculo) de EPs. Las redes jerárquicas se definen de manera recursiva tal como las redes de árboles, las jerarquías de pirámide y los clústeres de clústeres. Adicionalmente existen las llamadas

redes reconfigurables las cuales incluyen a todos los sistemas en donde el patrón de interconexión entre los EPs pueda ser cambiado.

3.3.3 Sistemas de procesamiento de transacción

Estos sistemas son la clase de computadoras más importante comercialmente hablando. Algunos tipos de sistemas de procesamiento de transacción en paralelo son:

1) Sistema Unix en paralelo (ej. SECUENT BALANCE): Este sistema utiliza la estructura en paralelo de Unix.

2) Sistema de base de datos en paralelo (ej. TERADATA DBC/1012): Tales sistemas soportan grandes bases de datos relacionales.

3) Sistemas tolerantes de fallas (ej. TANDEM TXP): En este caso el potencial en paralelo ha sido tradicionalmente limitado a jugar sólo el papel “repuesto en espera”.

4) Sistema de comunicación (ej. BBN BUTTERFLY): Este sistema es capaz de encaminar simultáneamente una cierta cantidad de mensajes.

Por ejemplo, los sistemas Unix en paralelo trabajan en el concepto de un “banco de procesadores” con todos los códigos de instrucciones y datos residiendo en la memoria global. Cuando un procesador queda inactivo éste es asignado al siguiente proceso en la lista de procesos del banco. Mientras el procesador lleva a cabo el proceso, el código y los datos son enviados por el bus del sistema al caché local reduciendo de esta manera los costos generales de comunicación.

3.3.4 Supercomputadoras numéricas

La arquitectura para computadoras en paralelo a gran escala, comúnmente manejada por supercomputadoras, es la de la topología hipercúbica. La arquitectura es llamada hipercúbica debido a que la organización se puede visualizar como la de un cubo. El cubo está comprendido de 2^n nodos de computadoras dispuestos como vértices de un cubo de n dimensiones. La dimensión denota el número de otros nodos más cercanos a los cuales cada nodo es directamente conectado así como el máximo número de saltos que un mensaje necesita dar para llegar de un nodo a otro.

El modelo de programación de un sistema hipercúbico se basa típicamente en el proceso concurrente que se comunica pasando mensajes con un solo nodo soportando cierta cantidad de procesos. Cada proceso tiene una identificación única de destino, identificación del remitente, tipo de mensaje y tamaño del mensaje. Los mensajes son puestos en fila de espera en tránsito y el orden de los mensajes se mantiene entre cualquier par de procesos. Los programas son escritos en lenguajes secuenciales convencionales.

3.3.5 Arquitecturas de VLSI

Por definición, las estructuras en paralelo implican un elemento computacional básico repetido una gran cantidad de veces. Esta propiedad estructural reduce la complejidad del diseño y hace a las estructuras en paralelo adecuadas para la *integración en gran escala* (VLSI). La VLSI es generalmente aplicada a un chip que contiene más de 100,000 dispositivos. Los problemas con las técnicas de VLSI son:

1. La complejidad del diseño es crítica: La colocación de dispositivos y la simulación de circuitos se vuelven más difíciles mientras se minimiza el área del chip ocupada por los cables de interconexión.
2. Los cables ocupan la mayoría del espacio en un circuito.
3. La comunicación no local reduce el rendimiento. Tiempo valioso se pierda cuando los módulos que están alejados se deben comunicar.

Al diseñar arquitecturas de VLSI dos enfoques son importantes:

1. Arreglo sistólico: Son un conjunto de celdas interconectadas, cada una capaz de realizar alguna simple operación. El intercambio de información entre celdas se da al estilo de segregación (pipeline) y la comunicación con el mundo exterior ocurre sólo en las "celdas fronterizas". Todas las operaciones están sincronizadas por un reloj global.
2. Computadora con Conjunto de Instrucciones Reducido (RISC): RISC depende de varios principios de arquitectura claves

    a) Todas las instrucciones son ejecutadas en un solo ciclo de reloj.
    b) Todas las instrucciones están definidas de manera predeterminada en el diseño.
    c) Pocas y simples instrucciones.
    d) Modos simples de direccionamiento.

Consideremos un ejemplo del enfoque RISC el cual es el más importante de los dos. El procesador en paralelo INMOS está basado en la tecnología RISC. Consiste en un procesador RISC de 16 bit ó 32 bit, 2 kilobytes de RAM estática, una interfaz de memoria multiplexada de 32 bit y cuatro conectores serie bidireccionales. El procesador ha incorporado soporte para multiprocesamiento y paralelismo. La comunicación entre procesadores en paralelos es operada por los conectores. Un procesador en paralelo puede ser utilizado de manera independiente o como un componente en una red en paralelo de procesadores en paralelo.

3.4 Máquina Virtual Paralela (PVM)

La máquina virtual paralela o PVM es una aplicación que permite ver una red de computadoras heterogéneas UNIX, por medio de un programa del usuario, como una sola computadora en paralelo.

El sistema PVM se ha ido convirtiendo en una tecnología viable para el procesamiento distribuido y paralelo en distintas disciplinas. PVM soporta un modelo sencillo pero completamente funcional de message-passing (paso de mensajes).

PVM está diseñado para enlazar los recursos computacionales y para proveer a los usuarios de una plataforma paralela en la cual correr sus aplicaciones independientemente de la cantidad de computadoras que se utilicen y de dónde estén localizadas. Cuando PVM está instalado correctamente es capaz de aprovechar los recursos combinados de las plataformas computacionales típicamente heterogéneas conectadas en red para entregar altos niveles de desempeño y funcionalidad.

El software PVM ha sido distribuido de forma gratuita y es utilizado en aplicaciones computacionales alrededor del mundo.

El procesamiento en paralelo, método por el cual se resuelve un gran problema mediante muchas tareas pequeñas, ha emergido como una tecnología aplicada crucial en la computación moderna. Los años que han pasado han sido testigos de una cada vez mayor aceptación y adopción del procesamiento en paralelo.

3.5 Multithreading

La idea central del multithreading se basa en el "cómo escribir programas computacionales que hagan muchas cosas diferentes, todas al mismo tiempo."

Actualmente un gran porcentaje de software escrito C/C++ aún utiliza un solo flujo de instrucciones (thread) para realizar su procesamiento.

3.5.1 Programación de un solo Thread (hilo)

Un programa de computadora tradicional normalmente incluye una serie de instrucciones, las cuales son ejecutadas una tras otra – El flujo de control es básicamente de arriba abajo con los ciclos *while* o *for* y las ramas *if-then-else* dándole la estructura. La

secuencia de operaciones que ejecuta la computadora cuando se corre un programa constituye un simple "*thread*" o hilo de ejecución. En C/C++ este thread está definido dentro de la función *main()* y en cualquier subrutina que pudiera invocar la función main(). En el momento en que el usuario corre el programa, el sistema operativo manda llamar la función *main()* y hace pasar cualquiera que sea el parámetro en la línea de comandos provisto por el usuario. El único thread del programa es activado y se mantiene vivo en todo el programa proveyendo todo el procesamiento requerido. Eventualmente la función *main()* cede el control al sistema operativo regresando un código de salida el cual termina el thread.

Si un programa de un solo thread quisiera llevar a cabo varias tareas diferentes al mismo tiempo entonces debería implementar la concurrencia deseada para sí mismo. El mecanismo esencial se ilustra en la Figura 3.2.

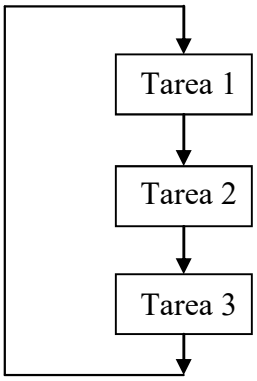

Figura 3.2 Ejecución de un solo hilo

El thread ejecuta las tareas secuencialmente pero realmente todavía no termina una tarea completamente cuando procede a la siguiente. El thread completa parcialmente un tarea particular y luego guarda la suficiente información acerca del estado actual de la tarea para luego resumirla en la siguiente iteración del ciclo. El cambio de tarea debe llevarse a cabo de manera regular para que parezca que todas las tareas corren al mismo tiempo. En particular, una tarea no debe bloquearse mientras espera que un evento ocurra, ya que esto evitaría que el resto de las tareas se siguieran ejecutando.

3.5.2 Programación Multi-Thread

La técnica para la ejecución simultánea de varias tareas ha sido aplicada tradicionalmente en sistemas operativos multi-tareas tales como UNIX. Cada tarea esta asociada a un "proceso" de ejecución independiente y el mismo sistema operativo es responsable de programar los cambios de una tarea a otra. En el caso de que varios procesos necesiten interactuar, éstos pueden hacer uso de varios servicios que ofrece el sistema operativo (tales como señales y la memoria compartida) que les permiten comunicación de inter-procesamiento (IPC).

La programación Multi-threads ofrece una alternativa a la programación multi-procesos, la cual demanda menos recursos del sistema. En la programación multi-threading el conjunto de tareas que estás interactuando son implementadas como múltiples threads (hilos) dentro de un solo proceso tal como se aprecia en la Figura 3.3.

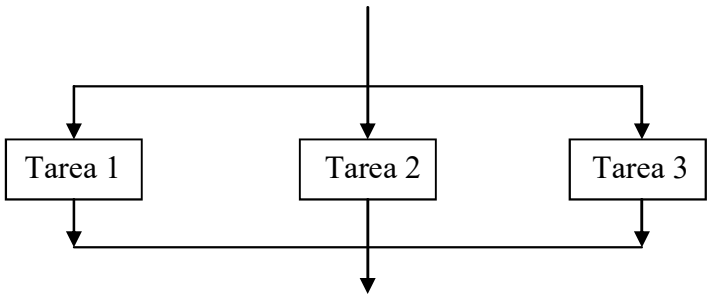

Figura 3.3 Ejecución de múltiples threads en un proceso

En el proceso uno de los threads es llamado "thread primario" y corresponde al único thread que se ejecuta en un programa de un solo thread. Se activa cuando el sistema operativo invoca la función principal *main()*. El thread primario es responsable de crear threads "secundarios" adicionales cuando estos sean requeridos por el programa. Cada uno de los threads secundarios ejecuta su propia función, llamada "función thread", la cual es análoga a la función *main()* del thread primario. Todos los threads dentro del proceso comparten el espacio de memoria destinado a ese proceso de esta manera todos los threads pueden comunicarse directamente a través de variables del programa ordinarias.

3.5.3 Sincronización de Threads

El conjunto de threads pertenecientes a un proceso comparten en su totalidad los mismos recursos. Por ejemplo, varios threads pueden invocar a la misma función para procesar el mismo conjunto de datos. Para evitar conflictos entre múltiples threads en tales situaciones es necesario que exista algún mecanismo de sincronización, Hay dos elementos de sincronización básicos: Mutex y Evento.

El "mutex" es un sistema de sincronización que provee recíprocamente acceso exclusivo a los recursos compartidos. Al recurso se le asigna un mutex para protegerlo y antes de que a un thread se le permita manipular el recurso éste debe de hacerse del mutex. Ningún otro thread puede tocar el recurso mientras éste tenga en su poder al mutex. Eventualmente el thread que está utilizando el recurso se deshará de él y el mutex asociado al recurso será liberado para así permitirle a los otros threads competir por él.

Un "evento" le permite a un thread enviar una señal a otro thread para avisarle que algo ha ocurrido. En tanto que un mutex es un mediador entre muchos threads que se encuentran en competencia, un evento permite que varios threads coordinen sus actividades. Por ejemplo, si un thread actúa como "productor" de un bien (i.e. datos en un búfer) mientras que otro es el "consumidor" de ese bien, entonces los dos threads pueden utilizar un evento para sincronizar su procedimiento. El thread consumidor aguarda a que el evento tenga lugar mientras que el thread productor manda una señal cuando el evento se encuentra listo. Es importante hacer notar que un evento contiene una especie de memoria integrada para el caso en que un thread mande una señal antes de que el otro haya comenzado a esperar, así la señal no se pierde.

Un evento suministra una señal binaria (encendido o apagado) de un thread a otro. Un "semáforo" es una útil generalización de un evento que incorpora un contador de enteros, el cual puede ser aumentado o disminuido por múltiples threads interrelacionados. El semáforo provee internamente la sincronización necesaria para asegurar que el contador

sea actualizado de forma segura sin corrupción, incluso si varios threads intentan modificarlo simultáneamente.

Tradicionalmente la mayoría de las aplicaciones en C/C++ usaron un solo thread para ejecutar todos los procesos requeridos por el programa. Si se requería de varias tareas ejecutadas simultáneamente la solución era utilizar múltiples procesos cada uno ejecutando un solo thread. Con el advenimiento de la programación multi-threading se dio cabida a los beneficios de la multi-tarea dentro de un solo proceso, lo cual es mucho menos exigente en cuanto a recursos del sistema que su equivalente multi-proceso.

Capítulo 4

Aplicación del Procesamiento en Paralelo para el Análisis en Estado Estable de Redes Eléctricas

4.1 Introducción

Como se ha dicho anteriormente para poder simular el comportamiento de un sistema eléctrico es necesario, en primer lugar, conocer las EDOs que lo describen. Una vez que se tiene la representación de la red eléctrica en un sistema de ecuaciones diferenciales, se hace uso de alguno de los distintos métodos numéricos para encontrar la solución y la respuesta en el estado estable periódico.

4.2 Obtención de las Ecuaciones de Estado que describen el comportamiento de un Sistema Eléctrico

Para simular un sistema eléctrico es necesario representarlo en EDOs. Como ejemplo se toma el sistema de 3 nodos de la IEEE.

4.2.1 Construcción del Sistema de Ecuaciones de Estado que describen el comportamiento del Sistema Eléctrico de 3 nodos de la IEEE

A continuación se describe la forma en que se obtienen las ecuaciones de estado del sistema de la Figura 4.1

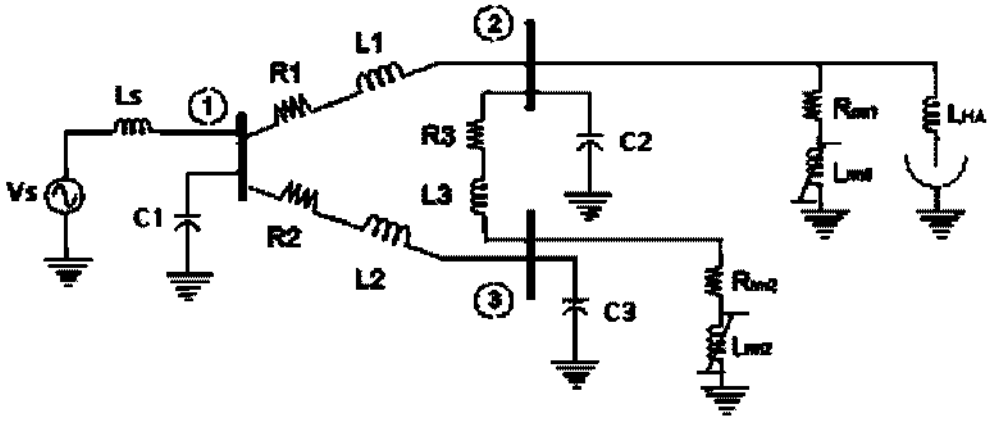

Figura 4.1 Sistema de 3 nodos de la IEEE con un horno de arco eléctrico conectado al nodo 2

Primeramente, aplicando LVK en la trayectoria que va del C1 al C2 se tiene:

$$V_1 = V_{R_1} + V_{L_1} + V_{C_2}$$
$$V_1 = i_1 R_1 + L_1 \frac{di_1}{dt} + V_{C_2}$$
$$\frac{di_1}{dt} = -\frac{i_1 R_1}{L_1} - \frac{V_{C_2}}{L_1} + \frac{V_1}{L_1} \tag{4.1}$$

Aplicando LVK en la trayectoria que va del C1 al C3 se tiene:

$$V_1 = V_{R_2} + V_{L_2} + V_{C_3}$$
$$V_1 = i_2 R_2 + L_2 \frac{di_2}{dt} + V_{C_3}$$
$$\frac{di_2}{dt} = -\frac{i_2 R_2}{L_2} - \frac{V_{C_3}}{L_2} + \frac{V_1}{L_2} \tag{4.2}$$

Aplicando LVK en la trayectoria que va del C2 al C3 se tiene:

$$V_{C_2} = V_{R_3} + V_{L_3} + V_{C_3}$$
$$V_{C_2} = i_3 R_3 + L_3 \frac{di_3}{dt} + V_{C_3}$$

$$\frac{di_3}{dt} = -\frac{i_3 R_3}{L_3} - \frac{V_{C_3}}{L_3} + \frac{V_{C_2}}{L_3} \tag{4.3}$$

Para la Rama magnetizante 1 se tiene:

$$i_{rm} = \psi^n$$
$$V_{C_2} = V_{rm_1} + V_{L_{rm_1}}$$
$$V_{C_2} = \psi_1^n R_{rm_1} + \frac{d\psi_1}{dt}$$
$$\frac{d\psi_1}{dt} = -\psi_1^n R_{rm_1} + V_{C_2} \tag{4.4}$$

Para la Rama magnetizante 2 se tiene:

$$V_{C_3} = V_{rm_2} + V_{L_{rm_2}}$$
$$V_{C_3} = \psi_2^n R_{rm_2} + \frac{d\psi_2}{dt}$$
$$\frac{d\psi_2}{dt} = -\psi_2^n R_{rm_2} + V_{C_3} \tag{4.5}$$

Para el generador se tiene

$$V_s = V_1 + V_{L_s}$$
$$V_s = L_s \frac{di_s}{dt} + V_{C_1}$$
$$\frac{di_s}{dt} = -\frac{V_{C_1}}{L_1} + \frac{V_s}{L_1} \tag{4.6}$$

Ahora, aplicando LCK en el nodo 1

$$i_s = i_{C_1} + i_1 + i_2$$
$$i_{C_1} = i_s - i_1 - i_2$$

$$\frac{dv_{C_1}}{dt} = \frac{i_{C_1}}{C_1}$$
$$\frac{dv_{C_1}}{dt} = \frac{i_s - i_1 - i_2}{C_1} \tag{4.7}$$

Aplicando LCK en el nodo 2

$$i_1 = i_{C_2} + i_{rm_1} + i_3 + i_{HA}$$
$$i_{C_2} = i_1 - i_{rm_1} - i_3 - i_{HA}$$
$$\frac{dv_{C_2}}{dt} = \frac{i_{C_2}}{C_2}$$
$$\frac{dv_{C_2}}{dt} = \frac{i_1 - \psi_1^n - i_3 - i_{HA}}{C_2} \tag{4.8}$$

Aplicando LCK en el nodo 3

$$i_2 + i_3 = i_{C_3} + i_{rm_2}$$
$$i_{C_3} = i_2 + i_3 - i_{rm_2}$$
$$\frac{dv_{C_3}}{dt} = \frac{i_{C_3}}{C_3}$$
$$\frac{dv_{C_3}}{dt} = \frac{i_2 + i_3 - \psi_2^n}{C_3} \tag{4.9}$$

Para el horno de arco eléctrico se tiene,

$$V_{C_2} = V_{L_{HA}} + V_{R_{HA}}$$
$$V_{R_{HA}} = k_3 r^{-(m+2)} i_{HA}$$
$$V_{L_{HA}} = L_{HA} \frac{di_{HA}}{dt}$$
$$V_{C_2} = L_{HA} \frac{di_{HA}}{dt} + k_3 r^{-(m+2)} i_{HA}$$
$$\frac{di_{HA}}{dt} = \frac{-k_3 r^{-(m+2)} i_{HA} + V_{C_2}}{L_{HA}}$$

(4.10)

Y finalmente, para r se tiene,

$$\frac{dr}{dx} = \frac{k_3}{k_2} r^{-(m+3)} i_{HA}^2 - \frac{k_1}{k_2} r^{n-1}$$

(4.11)

Con estas EDO's se obtiene la representación en variables de estado $\dot{x} = Ax + Bu$ que describe el comportamiento del sistema eléctrico.

4.3 Aplicación del método de Diferenciación Numérica de Newton para encontrar la solución en el Estado Estable

Después de obtener el sistema de ecuaciones que describe al sistema eléctrico se aplica el método de DN de Newton para encontrar la solución en estado estable del mismo. El método de DN a su vez requiere de un método de solución de ecuaciones diferenciales ordinarias como lo es el método de Runge-Kutta de cuarto orden.

4.3.1 Cálculo de la matriz Φ

Al utilizar el método de DN de Newton se requiere encontrar una matriz denominada Φ. La obtención de esta matriz se describe por medio del diagrama de flujo de la Figura 2.8.

Se toma el vector de condiciones iniciales x_0 del sistema de ecuaciones diferenciales que describen al sistema eléctrico. Se suma una pequeña perturbación ξ al vector de condiciones iniciales,

$$x_i = x_0 + \xi e_i$$

(4.12)

Con estas condiciones iniciales se simula un ciclo utilizando el método de Runge-Kutta. Así se obtiene el valor en el ciclo límite de x_i' y se calcula la diferencia obteniendo las columnas de la matriz ΔX^{i+1}

$$\Delta X^{i+1} = x_i' - x_0' \tag{4.13}$$

Donde x_0' es el vector de condiciones iniciales en el ciclo base. Con esta matriz se obtiene la columna $\Phi_{:i}$ de la matriz de identificación. De 2.23 se tiene,

$$\Phi_{:i} = \frac{\Delta X^{i+1}}{\xi} \tag{4.14}$$

El cálculo de las columnas de la matriz de identificación se puede realizar de forma simultánea ya que cada cálculo es independiente de los demás. Esta característica hace ideal al método de Diferenciación Numérico para utilizarse en plataformas de procesamiento en paralelo.

4.4 Propuesta de paralelización

En [García et al. 2001] se mostró la posibilidad de realizar el cálculo de la matriz de identificación Φ por columnas de manera simultánea, usando en particular la plataforma de procesamiento en paralelo Multithreading, en tanto que en [Ramos-Paz 2002] se utilizó PVM. Finalmente, en [Ramos-Paz 2007] se aplica tanto PVM como Multithreading.

En [Ramoz-Paz 2007] se hace uso de una red heterogénea de computadoras. En esta tesis se utiliza PVM en una red homogénea de computadoras con el objetivo de mostrar una comparación entre una red heterogénea de computadoras utilizando PVM y una computadora multinúcleo utilizando Multithreading en donde los núcleos son heterogéneos.

4.4.1 Paralelización del Método de Diferenciación Numérica con Multithreading

Como se mencionó anteriormente, el método de Diferenciación Numérica es ideal para ser paralelizado puesto que el cálculo de las columnas de Φ es independiente del cálculo de las demás columnas. El algoritmo de paralelización con Multithreading se realiza en dos partes, la primera por medio del proceso maestro y la segunda parte por medio de los *threads* o hilos. Es importante conocer el número de variables de estado que describen al sistema que se va a simular, de esta manera se puede determinar el número de columnas de Φ que serán calculadas por cada uno de los hilos.

El proceso maestro se explica a continuación.

1. En esta parte se define el número de hilos que habrán de emplearse durante la ejecución del programa.
2. Se cuantifica el número de columnas de la matriz de identificación Φ que serán calculadas por cada hilo utilizando el siguiente criterio.
    a. $número\ de\ columnas/hilo = \frac{número\ de\ columnas}{número\ de\ hilos}$
    b. El residuo se reparte por unidad entre los primeros hilos.
3. Se ejecutan todos los hilos.
    a. Durante la ejecución existe un protocolo de procesamiento para acceder a la matriz Φ, en el cual ésta se bloquea y se desbloquea continuamente para evitar que dos o más hilos accedan al mismo tiempo a modificarla. Por lo tanto un hilo sólo tendrá acceso a la matriz Φ cuando éste haya terminado de calcular los elementos de ésta y así poder modificarla.
4. Conforme van terminándose de ejecutar, los hilos esperan a que el último de ellos termine su tarea.
5. Al terminar todos los hilos de ejecutarse se habrá formado la Matriz de identificación Φ.

El proceso realizado por cada uno de los hilos se muestra en el diagrama de flujo de la Figura 4.2

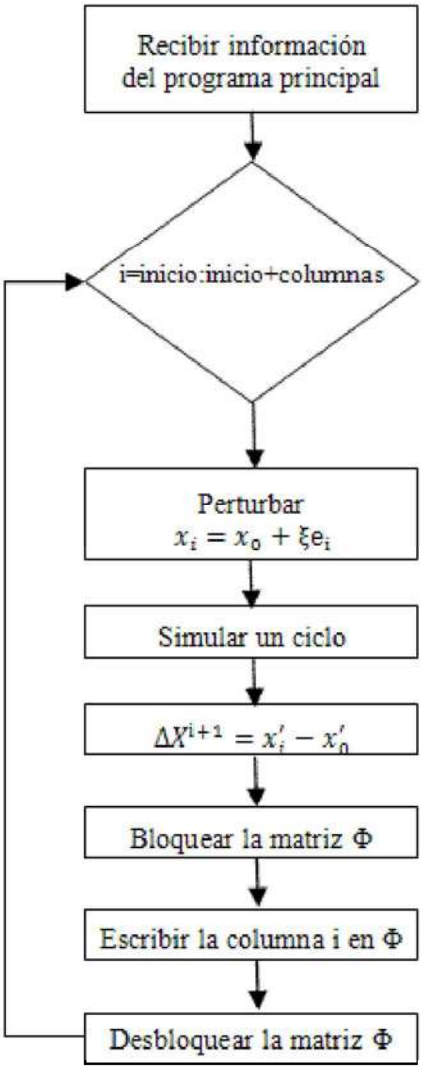

Figura 4.2 Programa Hilo Diferenciación Numérica

4.4.2 Paralelización del Método de Diferenciación Numérica con PVM

El algoritmo de paralelización con PVM se realiza en dos partes, la primera por medio del proceso o programa maestro y la segunda parte por medio de los programas esclavos. El número de variables de estado que describen al sistema que se va a simular es el mismo que el número de columnas de Φ que serán calculadas por cada uno de los esclavos.

El proceso esclavo se describe a continuación.

1. Se define el número de programas esclavos que se utilizarán durante la simulación., éste depende del número de computadoras disponibles y del número de computadoras que se desean utilizar.
2. Se define el número de columnas de la matriz Φ que serán calculadas por cada uno de los programas esclavos utilizando el siguiente criterio.
    a. Si el número de programas esclavos es menor al número de variables de estado cada programa esclavo calculará al menos una columna de la matriz Φ y el residuo se reparte entre los primeros programas esclavos.
    b. Si el número de programas esclavos es igual al número de variables de estado cada programa esclavo calculará una columna de la matriz de identificación Φ.
    c. Si el número de programas esclavos es mayor al número de variables de estado los primeros programas esclavos calcularán las columnas necesarias de la matriz Φ.
3. Se empaqueta la información que será enviada a cada uno de los programas esclavos. Esta información es necesaria para construir el conjunto de EDOs que modelan al sistema. También se empaqueta el vector de variables de estado en el ciclo base y el rango de columnas para ser calculadas. El empaquetamiento se realiza haciendo uso de rutinas de empaquetado y desempaquetado de la librería de PVM.
4. Se asegura que los programas esclavos estén disponibles para recibir la información que se les enviará.
5. Se envía la información a los programas esclavos.
6. Se ejecutan los programas esclavos.
7. Se recibe la información proveniente de los programas esclavos y se actualiza la matriz Φ.

El proceso realizado por cada uno de los programas esclavos se puede apreciar en el diagrama de flujo de la Figura 4.3

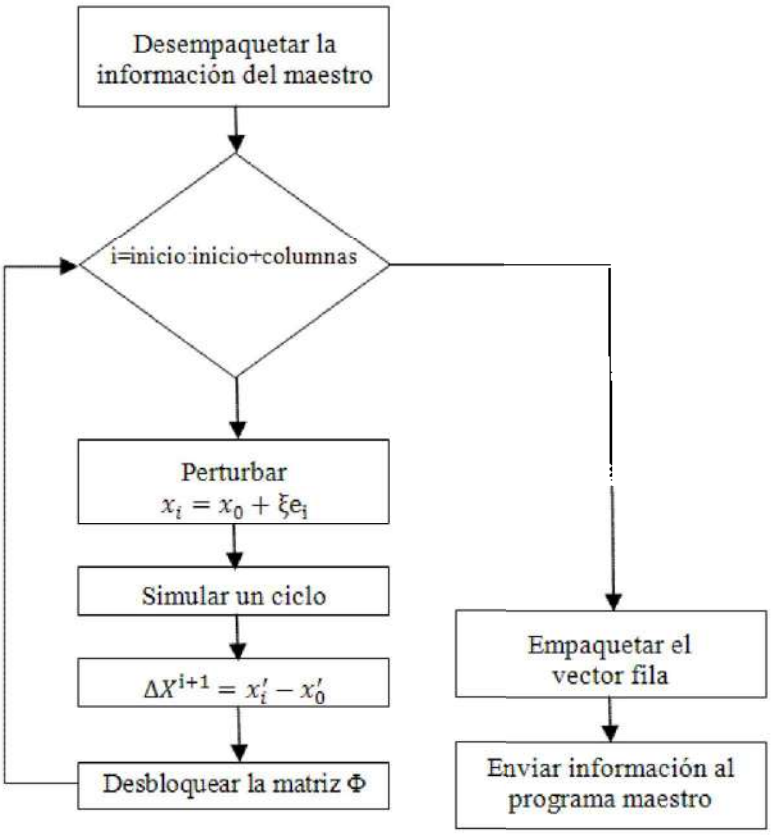

Figura 4.3 Programa Esclavo Diferenciación Numérica

Capítulo 5

Casos de Estudio

5.1 Introducción

En este capítulo se hace la comparación entre las plataformas de procesamiento en paralelo PVM y Multi-threading en cuanto al tiempo requerido por cada plataforma para llegar al estado estable así como la eficiencia relativa para cada plataforma. En este capítulo se presentan 5 casos de estudio del análisis en Estado Estable Periódico para redes eléctricas monofásicas que la IEEE provee como redes de muestra para distinto número de nodos. El conjunto de ecuaciones diferenciales que describen la dinámica de las redes a ser analizadas fue construido mediante la técnica de generación automática de ecuaciones diferenciales desarrollada en [Ramos-Paz 2007].

5.2 Casos de estudio utilizando PVM

5.2.1 Sistema de 3 nodos

El primer circuito a analizar es el sistema de 3 nodos de la IEEE modificado que se muestra en la Figura 4.1. Este sistema está conformado por un generador, tres líneas de transmisión, tres bancos de capacitores, dos ramas magnetizantes y un horno de arco eléctrico. La modificación está en añadir el horno de arco eléctrico. El comportamiento de este circuito se puede representar por medio de 11 ecuaciones diferenciales ordinarias.

En la Tabla 5.1 se muestra el proceso de convergencia al Estado Estable Periódico de la red de prueba de 3 nodos (Figura 4.1). Se puede ver que el método de Fuerza Bruta requiere de 114 periodos completos de integración mientras que el método de

Diferenciación Numérica requiere de sólo 56 periodos. Se observa que el método DN realiza el 49.12% del número de periodos que se requieren en el método de FB.

Tabla 5.1 Número de ciclos para converger de DN y FB para el sistema de 3 nodos

NC	FB	DN
1	8.519476E+00	8.519476E+00
2	1.279059E+00	1.279059E+00
3	5.751477E-01	5.751477E-01
4	2.575977E-01	2.575977E-01
5	1.025878E-01	1.025878E-01
6	4.383459E-02	4.383459E-02
7	2.568821E-02	2.568821E-02
8	2.025382E-02	2.025382E-02
⋮	⋮	⋮
20	2.089136E-03	3.731868E-03
32	2.388814E-04	6.711146E-05
44	2.736605E-05	1.130740E-10
56	3.135477E-06	5.151435E-14
⋮	⋮	
114	8.884027E-11	

Adicionalmente se llevaron a cabo las pruebas de eficiencia y velocidad utilizando el método DN en el sistema de 3 nodos. Sin embargo para este sistema no se aprecia la reducción en el tiempo de cómputo debido a que el tiempo que se puede ahorrar en el cómputo es invertido en la transmisión interna de datos dentro del sistema PVM llegando a ser ineficiente la paralelización del método DN para este caso en particular.

5.2.2 Sistema de 14 nodos

El sistema eléctrico de análisis es el sistema de 14 nodos de la IEEE. Este sistema está conformado por 15 líneas de transmisión, 5 transformadores, 15 generadores y 15 bancos de capacitores. El comportamiento de este circuito se puede representar por medio de 56 ecuaciones diferenciales ordinarias.

Tabla 5.2 Número de ciclos para converger de DN y FB para el sistema de 14 nodos

NC	FB	DN
1	2.537909E+00	2.537909+00
2	3.649385E-01	3.649385E-01
3	5.820431E-02	5.820431E-02
⋮	⋮	⋮
8	1.210356E-03	1.210356E-03
64	3.817456E-05	3.887882E-05
120	3.508878E-05	4.259000E-13
⋮	⋮	
5053	9.991160E-11	

En la Tabla 5.2 se muestra el proceso de convergencia al Estado Estable Periódico de la red de prueba de 14 nodos. Se aprecia que el método de Fuerza Bruta requiere de 5053 periodos completos de integración mientras que el método de Diferenciación Numérica requiere de sólo 120 periodos. Se observa que el método DN realiza el 2.37% del número de periodos que utiliza el método de FB.

Tabla 5.3 Tiempo requerido por el método DN al aumentar el número de procesadores para solucionar el sistema de 14 nodos

Tiempo (mseg)	Número de Procesadores			
	1	2	3	4
1	271	176	141	133
2	270	174	143	132
3	269	272	145	133
4	268	173	143	178
5	270	175	214	133
Promedio	**269.67**	**175**	**143.67**	**133**

En la Tabla 5.3 se tienen los distintos tiempos que se requirieron para solucionar el sistema de 14 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.1 (a) y la Figura 5.1 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores .En este caso al aumentar el número de procesadores a 4, se reduce el tiempo de cómputo casi al 50% lo que nos da una eficiencia cercana a 2. En este caso se nota la gran ventaja de paralelizar el método DN.

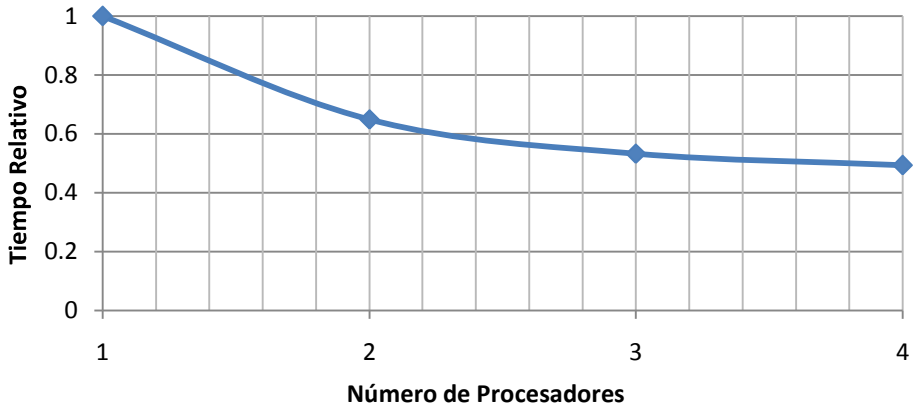

Figura 5.1 (a) Reducción del tiempo al aumentar el número de procesadores para el sistema de 14 nodos

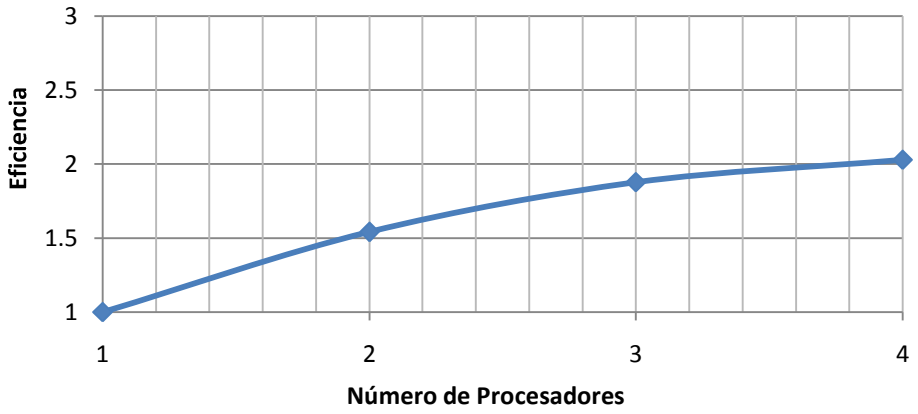

Figura 5.1 (b) Eficiencia al aumentar el número de procesadores para el sistema de 14 nodos

5.2.3 Sistema de 30 Nodos

El sistema eléctrico de análisis es el sistema de 30 nodos de la IEEE. Este sistema está conformado por 34 líneas de transmisión, 7 transformadores, 6 generadores y 34 bancos de capacitores. El comportamiento de este circuito se puede representar por medio de 97 ecuaciones diferenciales ordinarias.

Tabla 5.4 Número de ciclos para converger de DN y FB para el sistema de 30 nodos

NC	FB	DN
1	1.117788E+00	1.117788E+00
2	2.162617E-01	2.162617E-01
3	6.681112E-02	6.681112E-02
⋮	⋮	⋮
8	2.545522E-03	2.545522E-03
106	1.674104E-05	4.044592E-08
204	2.667905E-07	2.200000E-15
⋮	⋮	
14898	9.999510E-11	

En la Tabla 5.4 se muestra el proceso de convergencia al Estado Estable Periódico de la red de prueba de 30 nodos. Se aprecia que el método de Fuerza Bruta requiere de 14898 periodos completos de integración mientras que el método de Diferenciación Numérica requiere de sólo 204 periodos. Se observa que el método DN realiza el 1.36% del número de periodos que utiliza el método de FB.

Tabla 5.5 Tiempo requerido por el método DN al aumentar el número de procesadores para solucionar el sistema de 30 nodos

Tiempo (mseg)	Número de Procesadores			
	1	2	3	4
1	1949	1070	785	628
2	1972	1073	785	629
3	1947	1064	785	631
4	1963	1087	784	628
5	1948	1066	786	633
Promedio	**1953.33**	**1069.67**	**785**	**629.33**

En la Tabla 5.3 se tienen los distintos tiempos que se requirieron para solucionar el sistema de 30 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.2 (a) y la Figura 5.2 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores .En este caso al aumentar el número de procesadores a 4, se requiere de sólo el 32.2% de tiempo que nos tomaría si se realizara la simulación con un solo procesador, esto nos da una eficiencia de 3.1 En este caso también se nota la gran ventaja de paralelizar el método DN.

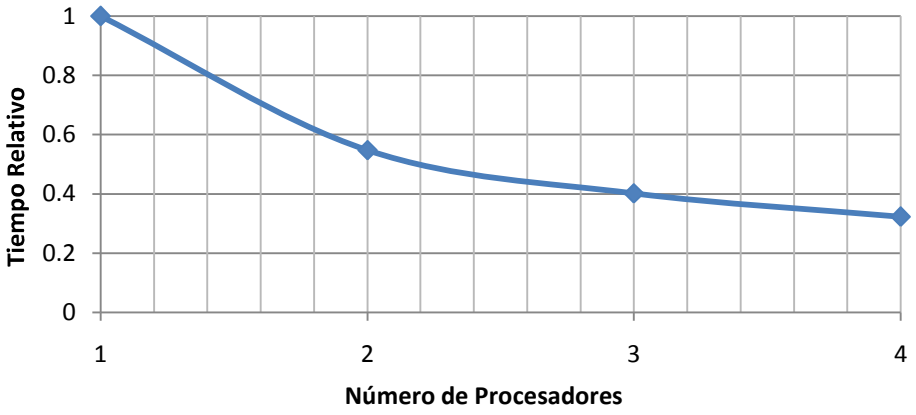

Figura 5.2 (a) Reducción del tiempo al aumentar el número de procesadores para el sistema de 30 nodos

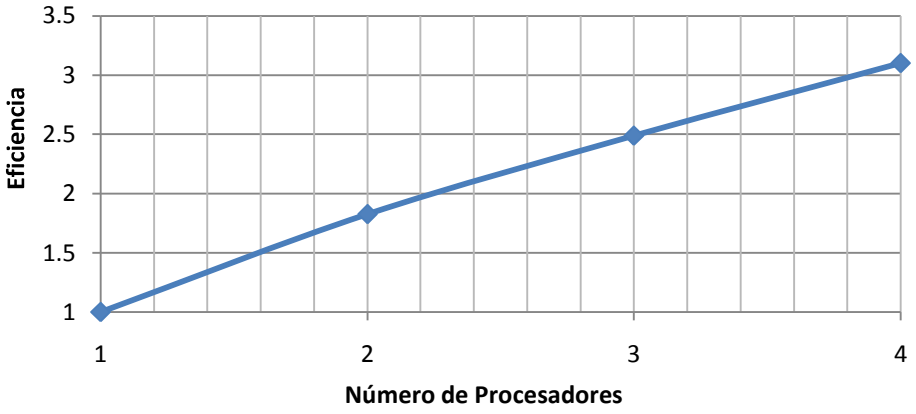

Figura 5.2 (b) Eficiencia al aumentar el número de procesadores para el sistema de 30 nodos

5.2.4 Sistema de 57 Nodos

El sistema eléctrico de análisis es el sistema de 57 nodos de la IEEE. Este sistema está conformado por 63 líneas de transmisión, 34 transformadores, 7 generadores y 63 bancos de capacitores. El comportamiento de este circuito se puede representar por medio de 186 ecuaciones diferenciales ordinarias.

Tabla 5.6 Número de ciclos para converger de DN y FB para el sistema de 57 nodos

NC	FB	DN
1	1.488684E+00	1.488684E+00
2	5.580060E-01	5.580060E-01
3	3.950136E-01	3.950136E-01
⋮	⋮	⋮
8	1.127601E-02	1.127601E-02
195	4.888185E-05	1.383870E-05
382	2.272502E-05	1.246100E-12
⋮	⋮	
2507	9.964190E-11	

En la Tabla 5.6 se muestra el proceso de convergencia al Estado Estable Periódico de la red de prueba de 57 nodos. Se aprecia que el método de Fuerza Bruta requiere de 2507 periodos completos de integración mientras que el método de Diferenciación Numérica requiere de sólo 382 periodos. Se observa que el método DN realiza el 15.23% del número de periodos que utiliza el método de FB.

Tabla 5.7 Tiempo requerido por el método DN al aumentar el número de procesadores para solucionar el sistema de 57 nodos

Tiempo (mseg)	Número de Procesadores			
	1	2	3	4
1	7410	3942	2797	2284
2	7444	3947	2775	2185
3	7391	3947	2830	2292
4	7436	3946	2780	2188
5	7428	3947	2779	2189
Promedio	**7424.67**	**3946.67**	**2785.33**	**2220.33**

En la Tabla 5.7 se tienen los distintos tiempos que se requirieron para solucionar el sistema de 57 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.3 (a) y la Figura 5.3 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores .En este caso al aumentar el número de procesadores a 4, se requiere de sólo el 30% de tiempo que nos tomaría si se realizara la simulación con un solo procesador, esto nos da una eficiencia de 3.34 En este caso también se nota la gran ventaja de paralelizar el método DN.

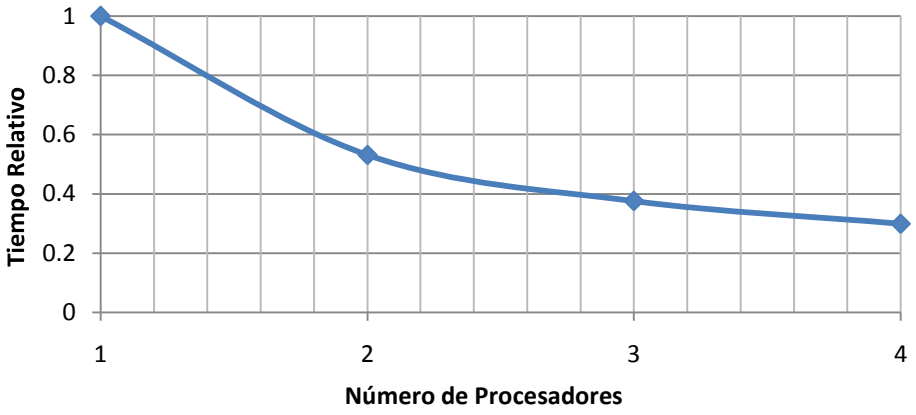

Figura 5.3 (a) Reducción del tiempo al aumentar el número de procesadores para el sistema de 57 nodos

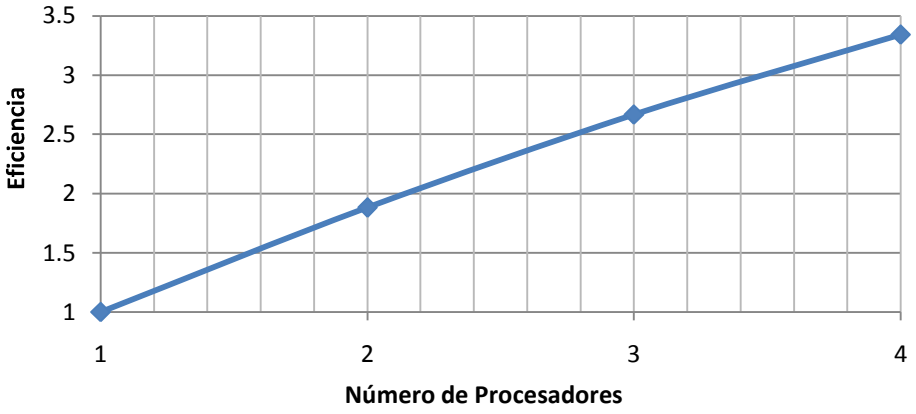

Figura 5.3 (b) Eficiencia al aumentar el número de procesadores para el sistema de 57 nodos

5.2.5 Sistema de 118 Nodos

El sistema eléctrico de análisis es el sistema de 118 nodos de la IEEE. Este sistema está conformado por 177 líneas de transmisión, 9 transformadores, 7 generadores y 177 bancos de capacitores. El comportamiento de este circuito se puede representar por medio de 390 ecuaciones diferenciales ordinarias.

Tabla 5.8 Número de ciclos para converger de DN y FB para el sistema de 118 nodos

NC	FB	DN
1	1.419945E+00	1.419945E+00
2	1.046282E+00	1.046282E+00
3	8.238483E-01	8.238483E-01
⋮	⋮	⋮
8	2.518677E-01	2.518677E-01
399	3.835443E-05	1.042335E-02
790	2.917796E-05	7.218852E-05
1181	2.217269E-05	2.126601E-10
1572	1.683363E-05	2.700000E-15
⋮	⋮	
11776	9.989880E-11	

En la Tabla 5.8 se muestra el proceso de convergencia al Estado Estable Periódico de la red de prueba de 118 nodos. Se aprecia que el método de Fuerza Bruta requiere de 11776 periodos completos de integración mientras que el método de Diferenciación Numérica requiere de sólo 1572 periodos. Se observa que el método DN realiza el 13.34% del número de periodos que utiliza el método de FB.

Tabla 5.9 Tiempo requerido por el método DN al aumentar el número de procesadores para solucionar el sistema de 118 nodos

Tiempo (mseg)	Número de Procesadores			
	1	2	3	4
1	73033	37883	26336	20627
2	72750	38106	27040	20411
3	73484	38092	26931	20395
4	72782	37814	27280	20392
5	72288	37772	26213	20399
Promedio	**72855**	**37929.67**	**26769**	**20401.67**

En la Tabla 5.9 se tienen los distintos tiempos que se requirieron para solucionar el sistema de 118 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.4 (a) y la Figura 5.4 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores .En este caso al aumentar el número de procesadores a 4, se requiere de sólo el 28% de tiempo que nos tomaría si se

realizara la simulación con un solo procesador, esto nos da una eficiencia de 3.57 En este caso también se nota la gran ventaja de paralelizar el método DN.

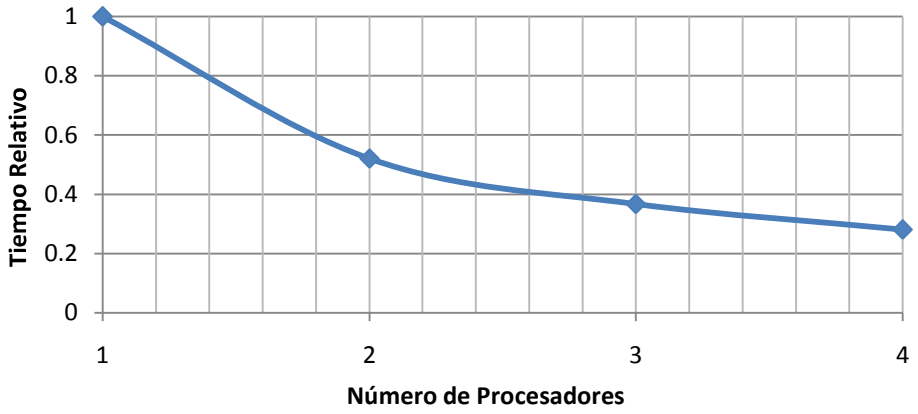

Figura 5.4 (a) Reducción del tiempo al aumentar el número de procesadores para el sistema de 118 nodos

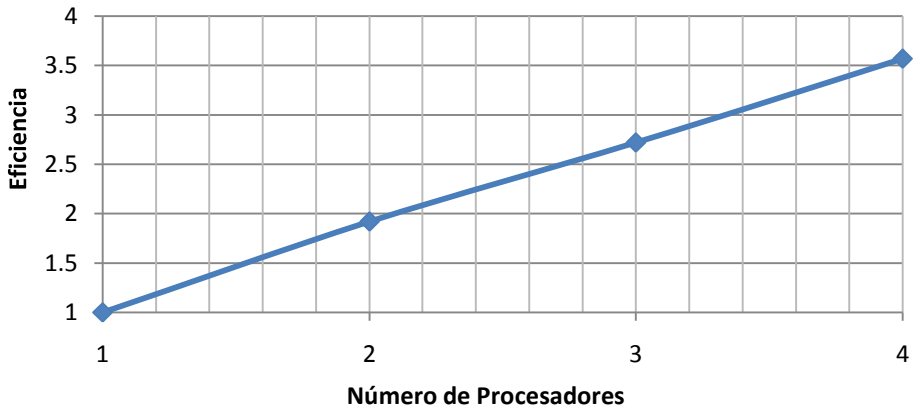

Figura 5.4 (b) Eficiencia al aumentar el número de procesadores para el sistema de 118 nodos

5.3 Casos de estudio utilizando Multi-threading

5.3.1 Sistema de 14 nodos

5.3.1.1 Pruebas con 8 threads

Tabla 5.10 (a) Tiempo requerido por el método DN para solucionar el sistema de 14 nodos en la computadora de 8 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	2	3	4	5	6	7	8	9	10	prom
1	278	291	278	282	278	278	279	278	278	278	**278.625**
2	174	188	178	184	184	180	186	186	179	194	**183.125**
3	173	160	151	161	146	145	151	140	143	157	**151.75**
4	146	153	162	144	147	142	158	147	132	148	**148.125**
5	145	134	140	153	131	166	166	141	140	136	**144.375**
6	143	144	131	147	176	146	151	148	139	165	**147.875**
7	156	138	146	146	133	136	145	129	136	150	**141.25**
8	128	147	134	149	136	136	136	149	145	126	**138.875**

Tabla 5.10 (b) Tiempo relativo y eficiencia en la computadora de 8 núcleos para el sistema de 14 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	278.625	1	1
2	183.125	0.6572454	1.52150171
3	151.75	0.54463885	1.83607908
4	148.125	0.53162853	1.88101266
5	144.375	0.51816958	1.92987013
6	147.875	0.53073127	1.88419273
7	141.25	0.50695379	1.97256637
8	138.875	0.49842979	2.00630063

En la Tabla 5.10 (a) y Tabla 5.10 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 14 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.5 (a) y la Figura 5.5 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de threads. En este caso las pruebas se realizaron en la computadora de 8 núcleos, se puede apreciar que se

reduce el tiempo de cómputo al 50% lo que nos da una eficiencia de 2. En este caso se nota la gran ventaja de paralelizar el método DN.

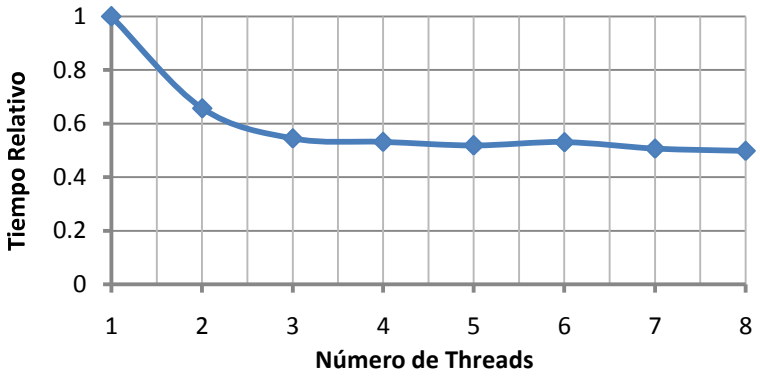

Figura 5.5 (a) Reducción del tiempo en la computadora de 8 núcleos para el sistema de 14 nodos

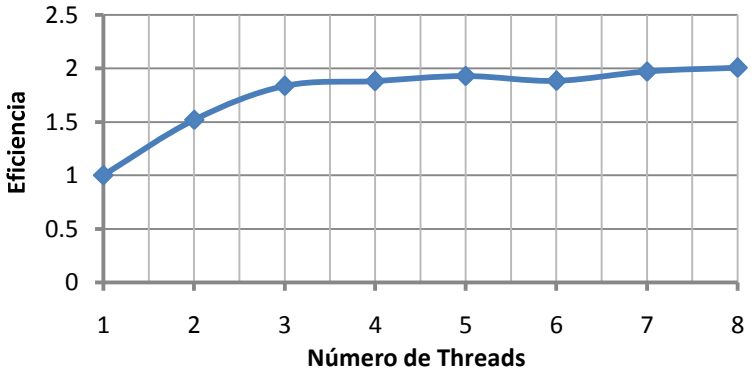

Figura 5.5 (b) Eficiencia en la computadora de 8 núcleos para el sistema de 14 nodos

5.3.1.2 Pruebas con 4 threads

Tabla 5.11 (a) Tiempo requerido por el método DN para solucionar el sistema de 14 nodos en la computadora de 4 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	345	375	373	346	373	373	346	362	344	345	**357.875**
2	284	275	263	293	246	278	302	295	315	300	**286.25**
3	286	248	260	228	268	249	292	318	238	286	**265.875**
4	245	272	277	262	298	286	277	255	256	269	**269.25**

Tabla 5.11 (b) Tiempo relativo y eficiencia en la computadora de 4 núcleos para el sistema de 14 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	357.875	1	1
2	286.25	0.79986029	1.25021834
3	265.875	0.742927	1.34602727
4	269.25	0.75235767	1.32915506

En la Tabla 5.11 (a) y 5.11 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 14 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.6 (a) y la Figura 5.6 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de threads. En este caso las pruebas se realizaron en la computadora de 4 núcleos, se puede apreciar que se reduce el tiempo de cómputo cerca del 25% lo que nos da una eficiencia de 1.3291.

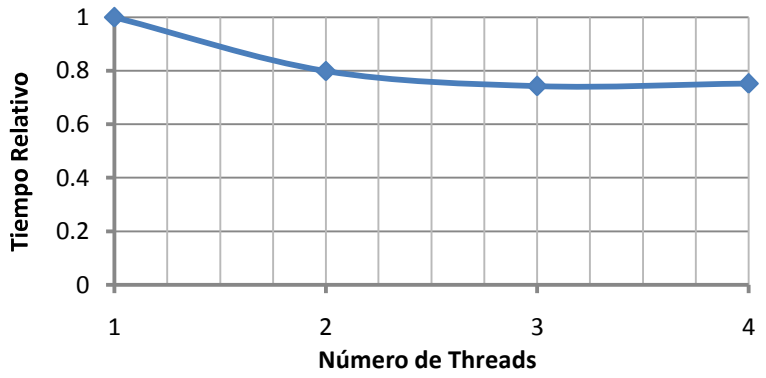

Figura 5.6 (a) Reducción del tiempo en la computadora de 4 núcleos para el sistema de 14 nodos

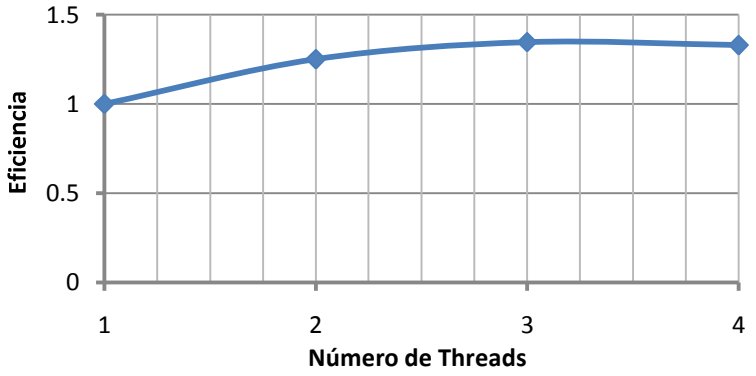

Figura 5.6 (b) Eficiencia en la computadora de 4 núcleos para el sistema de 14 nodos

5.3.1.3 Pruebas con 2 threads

Tabla 5.12 (a) Tiempo requerido por el método DN para solucionar el sistema de 14 nodos en la computadora de 2 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	228	265	222	228	250	248	225	230	240	247	**237**
2	167	161	149	183	170	167	157	163	151	156	**161.5**

Tabla 5.12 (b) Tiempo relativo y eficiencia en la computadora de 2 núcleos para el sistema de 14 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	237	1	1
2	161.5	0.6814346	1.46749226

En la Tabla 5.12 (a) y Tabla 5.12 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 14 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.7 (a) y la Figura 5.7 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de threads. En este caso las pruebas se realizaron en la computadora de 2 núcleos, se puede apreciar que se reduce el tiempo de cómputo cerca del 32% lo que nos da una eficiencia de 1.4674. Se puede apreciar que con sólo utilizar un núcleo más se redujo en buena parte el tiempo de ejecución.

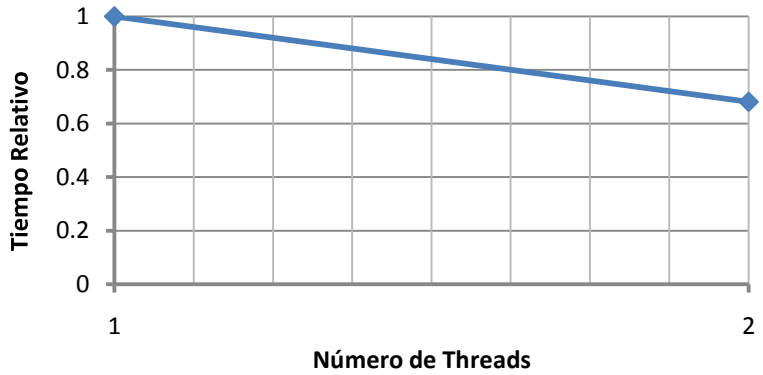

Figura 5.7 (a) Reducción del tiempo en la computadora de 2 núcleos para el sistema de 14 nodos

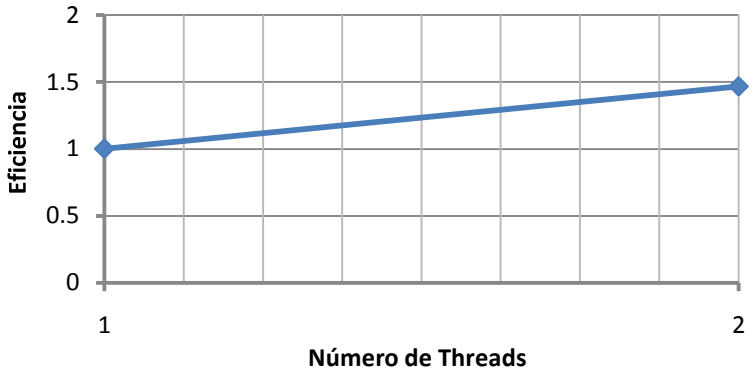

Figura 5.7 (b) Eficiencia en la computadora de 2 núcleos para el sistema de 14 nodos

5.3.2 Sistema de 30 nodos

5.3.2.1 Pruebas con 8 threads

Tabla 5.13 (a) Tiempo requerido por el método DN para solucionar el sistema de 30 nodos en la computadora de 8 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	522	522	523	523	522	522	521	524	522	522	**522.25**
2	309	310	312	296	307	305	306	300	309	309	**306.875**
3	241	246	241	236	255	242	247	248	238	247	**243.75**
4	202	251	211	227	234	223	197	222	229	237	**223.125**
5	198	222	239	212	216	222	216	210	215	206	**214.875**
6	201	208	212	224	216	213	208	213	229	197	**211.875**
7	203	201	202	208	228	198	210	230	200	189	**206.25**
8	218	224	190	197	197	210	203	196	207	226	**206.5**

Tabla 5.13 (b) Tiempo relativo y eficiencia en la computadora de 8 núcleos para el sistema de 30 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	522.25	1	1
2	306.875	0.58760172	1.70183299
3	243.75	0.46673049	2.1425641
4	223.125	0.42723791	2.34061625
5	214.875	0.41144088	2.43048284
6	211.875	0.40569651	2.46489676
7	206.25	0.3949258	2.53212121
8	206.5	0.3954045	2.52905569

En la Tabla 5.13 (a) y Tabla 5.13 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 30 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.8 (a) y la Figura 5.8 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 8 núcleos, se puede observar que se requiere de sólo el 39.49% de tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 2.529. En este caso también se nota la gran ventaja de paralelizar el método DN.

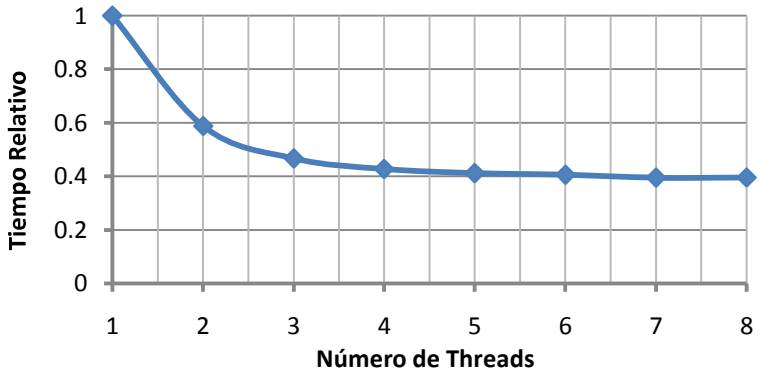

Figura 5.8 (a) Reducción del tiempo en la computadora de 8 núcleos para el sistema de 30 nodos

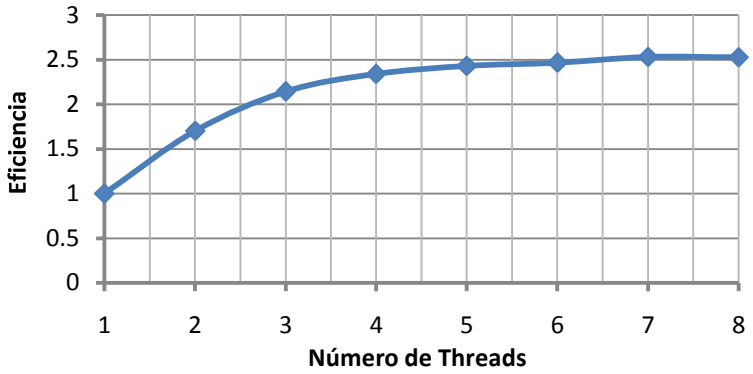

Figura 5.8 (b) Eficiencia en la computadora de 8 núcleos para el sistema de 30 nodos

5.3.2.2 Pruebas con 4 threads

Tabla 5.14 (a) Tiempo requerido por el método DN para solucionar el sistema de 30 nodos en la computadora de 4 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	737	726	737	726	733	737	727	738	726	737	**732.5**
2	627	442	748	636	600	708	507	453	635	582	**593.5**
3	487	531	544	530	543	531	515	527	515	531	**527.875**
4	543	547	529	487	537	514	518	534	542	454	**525.5**

Tabla 5.14 (b) Tiempo relativo y eficiencia en la computadora de 4 núcleos para el sistema de 30 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	732.5	1	1
2	593.5	0.81023891	1.23420388
3	527.875	0.72064846	1.38763912
4	525.5	0.71740614	1.39391056

En la Tabla 5.14 (a) y Tabla 5.14 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 30 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.9 (a) y la Figura 5.9 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 4 núcleos, se

puede observar que se reduce el tiempo de cómputo cerca del 30% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 1.3939.

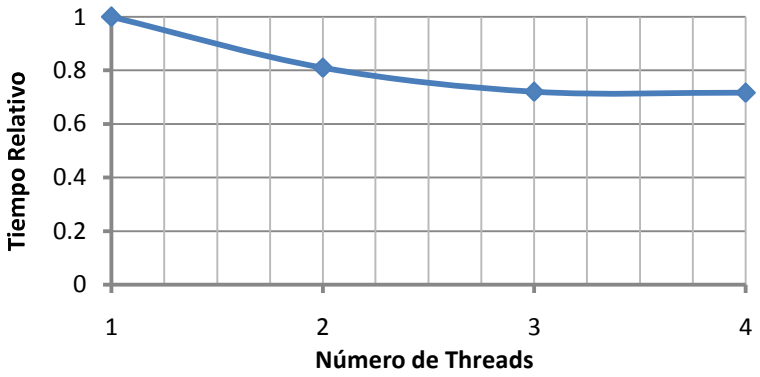

Figura 5.9 (a) Reducción del tiempo en la computadora de 4 núcleos para el sistema de 30 nodos

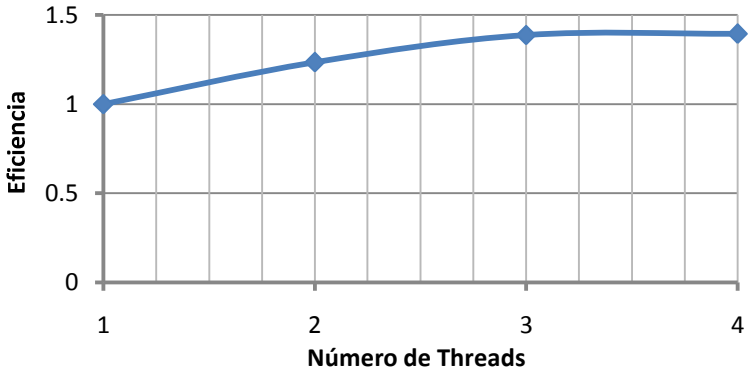

Figura 5.9 (b) Eficiencia en la computadora de 4 núcleos para el sistema de 30 nodos

5.3.2.3 Pruebas con 2 threads

Tabla 5.15 (a) Tiempo requerido por el método DN para solucionar el sistema de 30 nodos en la computadora de 2 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	453	431	457	443	444	444	451	448	487	483	**452.875**
2	312	346	296	284	325	315	318	335	353	279	**316.375**

Tabla 5.15 (b) Tiempo relativo y eficiencia en la computadora de 2 núcleos para el sistema de 30 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	452.875	1	1
2	316.375	0.69859233	1.43145002

En la Tabla 5.15 (a) y Tabla 5.15 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 30 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.10 (a) y la Figura 5.10 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 2 núcleos, se puede observar que se ahorra cerca del 30% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 1.43. En este caso también se nota la gran ventaja de paralelizar el método DN.

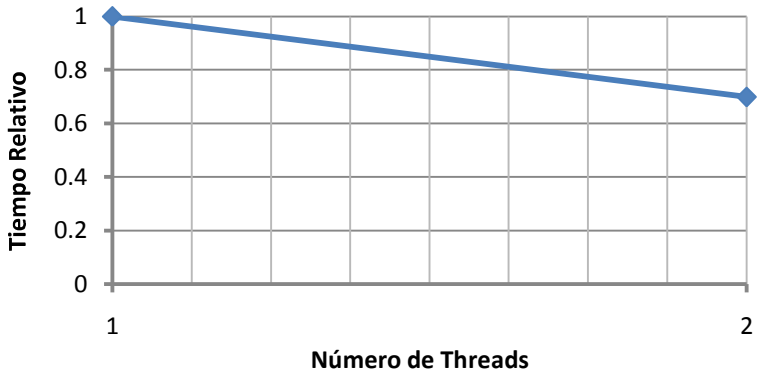

Figura 5.10 (a) Reducción del tiempo en la computadora de 2 núcleos para el sistema de 30 nodos

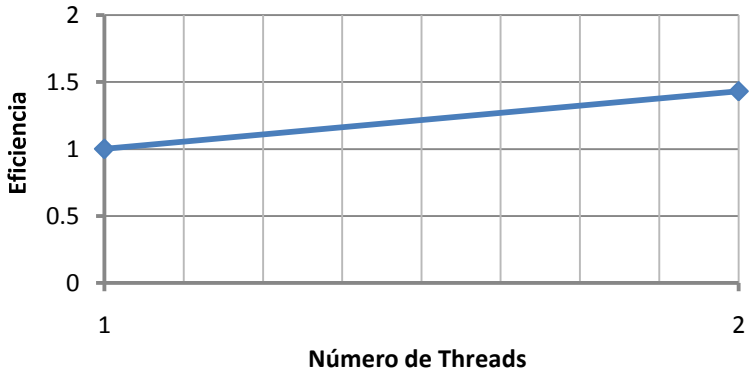

Figura 5.10 (b) Eficiencia en la computadora de 2 núcleos para el sistema de 30 nodos

5.3.3 Sistema de 57 nodos

5.3.3.1 Pruebas con 8 threads

Tabla 5.16 (a) Tiempo requerido por el método DN para solucionar el sistema de 57 nodos en la computadora de 8 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	2638	2684	2693	2685	2684	2685	2683	2696	2682	2684	**2685**
2	1477	1488	1459	1459	1487	1461	1470	1470	1470	1468	**1470.25**
3	1087	1088	1087	1083	1115	1080	1087	1082	1108	1081	**1087.875**
4	903	873	908	903	904	901	889	893	936	926	**903.375**
5	795	843	779	810	807	805	814	792	803	793	**802.375**
6	807	775	730	772	807	766	702	764	710	722	**755.75**
7	729	713	717	702	693	730	731	706	723	694	**714.25**
8	746	674	744	730	753	784	755	710	744	765	**743.375**

Tabla 5.16 (b) Tiempo relativo y eficiencia en la computadora de 8 núcleos para el sistema de 57 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	2685	1	1
2	1470.25	0.54757914	1.82622003
3	1087.875	0.4051676	2.46811444
4	903.375	0.33645251	2.97218763
5	802.375	0.29883613	3.34631563
6	755.75	0.28147114	3.55276216
7	714.25	0.2660149	3.75918796
8	743.375	0.2768622	3.61190516

En la Tabla 5.16 (a) y Tabla 5.16 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 57 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.11 (a) y la Figura 5.11 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 8 núcleos, se

puede observar que se reduce el tiempo de ejecución al 27% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 3.61. A pesar de que se requiere menor tiempo en la ejecución con 7 threads que con 8, aquí se aprecia la gran ventaja de utilizar el procesamiento en paralelo. Esto se debe a que se ya no se puede reducir más el tiempo en la paralelización puesto que se ha llegado al tiempo que requiere la parte secuencial del programa para ejecutarse.

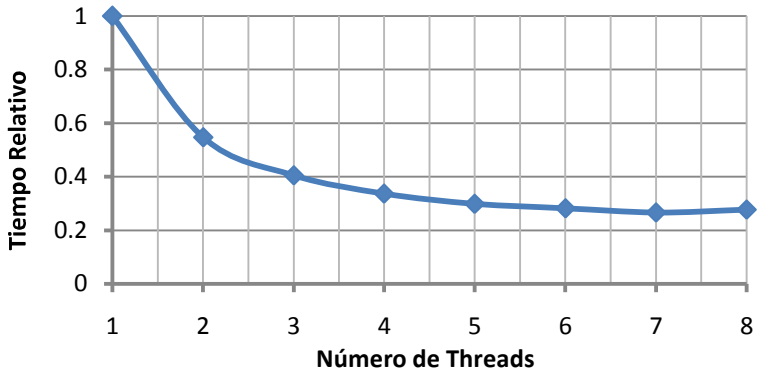

Figura 5.11 (a) Reducción del tiempo en la computadora de 8 núcleos para el sistema de 57 nodos

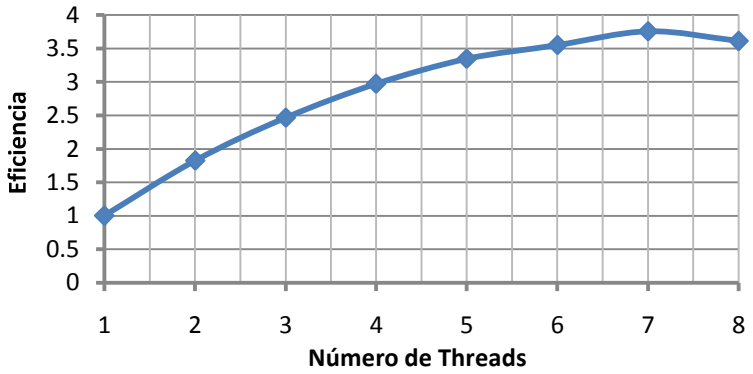

Figura 5.11 (b) Eficiencia en la computadora de 8 núcleos para el sistema de 57 nodos

5.3.3.2 Pruebas con 4 threads

Tabla 5.17 (a) Tiempo requerido por el método DN para solucionar el sistema de 57 nodos en la computadora de 4 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	2	3	4	5	6	7	8	9	10	prom
1	3636	3657	3638	3659	3684	3662	3661	3648	3649	3649	**3652.875**
2	2417	2300	2033	2680	2053	2122	2253	2211	2299	3162	**2291.875**
3	2171	2164	2170	2223	2169	2202	2163	2100	2186	2152	**2172.125**
4	2188	2147	2170	2259	2184	2093	2202	2149	2150	2216	**2175.75**

Tabla 5.17 (b) Tiempo relativo y eficiencia en la computadora de 4 núcleos para el sistema de 57 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	3652.875	1	1
2	2291.875	0.62741676	1.59383692
3	2172.125	0.59463436	1.6817057
4	2175.75	0.59562673	1.67890383

En la Tabla 5.17 (a) y Tabla 5.17 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 57 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.12 (a) y la Figura 5.12 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 4 núcleos, se puede observar que se reduce el tiempo de ejecución cerca del 60% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 1.67.

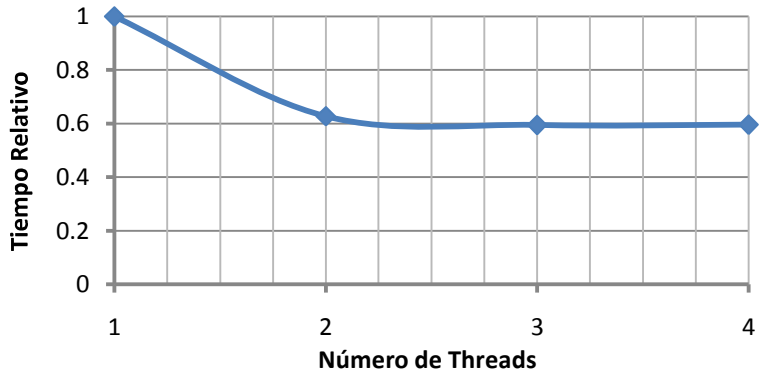

Figura 5.12 (a) Reducción del tiempo en la computadora de 4 núcleos para el sistema de 57 nodos

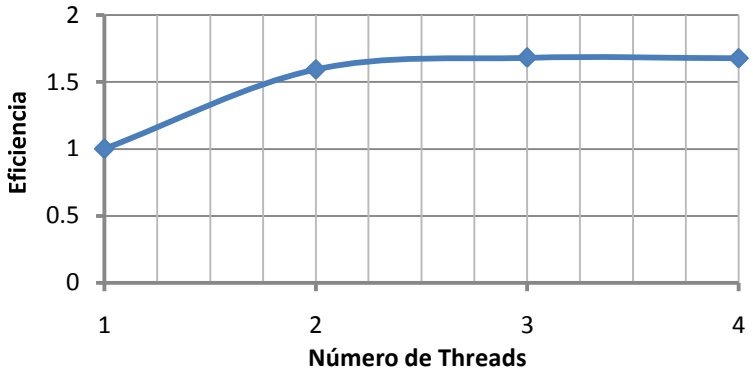

Figura 5.12 (b) Eficiencia en la computadora de 4 núcleos para el sistema de 57 nodos

5.3.3.3 Pruebas con 2 threads

Tabla 5.18 (a) Tiempo requerido por el método DN para solucionar el sistema de 57 nodos en la computadora de 2 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	2769	2658	2740	2725	2762	2764	2723	2750	2734	2764	**2745.25**
2	1746	1732	1760	1713	1732	1745	1762	1817	1755	1744	**1747**

Tabla 5.18 (b) Tiempo relativo y eficiencia en la computadora de 2 núcleos para el sistema de 57 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	2745.25	1	1
2	1747	0.63637192	1.57140813

En la Tabla 5.18 (a) y Tabla 5.18 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 57 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.13 (a) y la Figura 5.13 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 2 núcleos, se puede observar que se reduce el tiempo de ejecución a 63.63% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 1.57. Se puede apreciar que la computadora de 2 núcleos es muy rápida puesto que tiene una buena eficiencia para sólo contar con 2 núcleos.

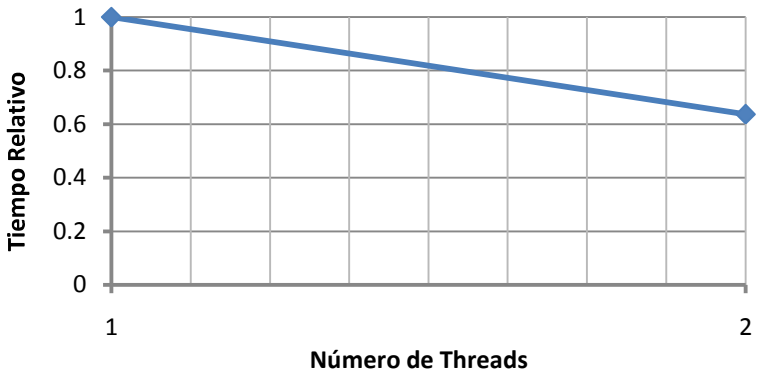

Figura 5.13 (a) Reducción del tiempo en la computadora de 2 núcleos para el sistema de 57 nodos

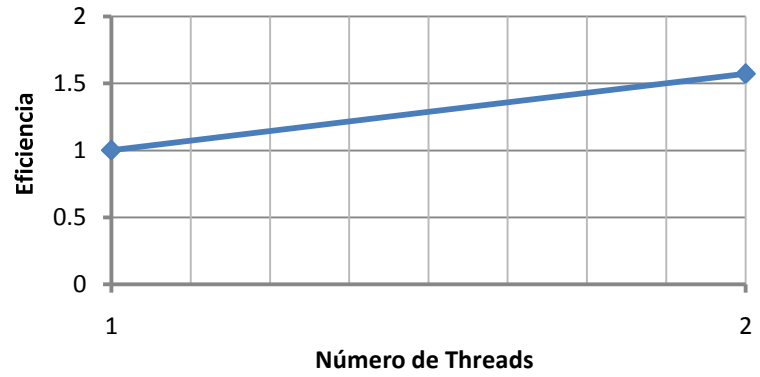

Figura 5.13 (b) Eficiencia en la computadora de 2 núcleos para el sistema de 57 nodos

5.3.4 Sistema de 118 nodos

5.3.4.1 Pruebas con 8 threads

Tabla 5.19 (a) Tiempo requerido por el método DN para solucionar el sistema de 118 nodos en la computadora de 8 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	12501	12502	12505	12576	12502	12501	12503	12581	12502	12501	**12511.5**
2	7027	7062	7017	7013	7022	7042	7026	7044	7009	7043	**7029.25**
3	5193	5193	5219	5213	5195	5204	5197	5198	5203	5236	**5202.75**
4	4290	4297	4300	4308	4291	4304	4318	4287	4298	4326	**4300.75**
5	3764	3748	3735	3752	3755	3764	3772	3756	3731	3734	**3751**
6	3440	3398	3422	3441	3373	3379	3414	3420	3433	3436	**3417.75**
7	3186	3183	3136	3168	3159	3141	3162	3174	3153	3159	**3162.375**
8	3034	3017	3019	2995	3012	3016	3024	3005	2992	2979	**3010**

Tabla 5.19 (b) Tiempo relativo y eficiencia en la computadora de 8 núcleos para el sistema de 118 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	12511.5	1	1
2	7029.25	0.56182312	1.77991962
3	5202.75	0.41583743	2.40478593
4	4300.75	0.34374376	2.90914375
5	3751	0.29980418	3.33551053
6	3417.75	0.27316868	3.66074172
7	3162.375	0.25275746	3.95636191
8	3010	0.24057867	4.15664452

En la Tabla 5.19 (a) y Tabla 5.19 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 118 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.14 (a) y la Figura 5.14 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 8 núcleos, se puede observar que se reduce el tiempo de ejecución a 24.05% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 4.1566. Aquí se observa que el método de DN es poco más de 4 veces más rápido al ser paralelizado.

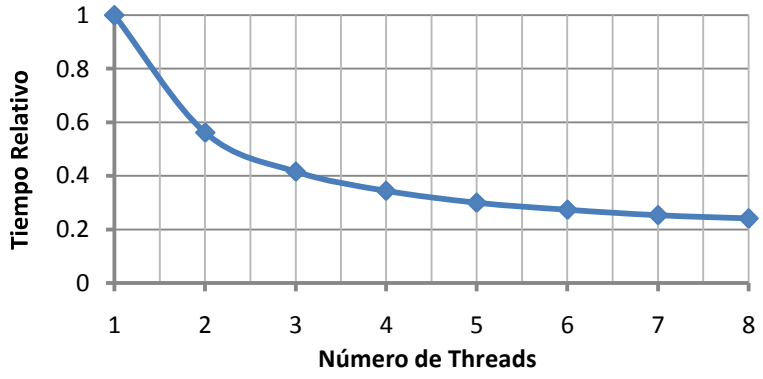

Figura 5.14 (a) Reducción del tiempo en la computadora de 8 núcleos para el sistema de 118 nodos

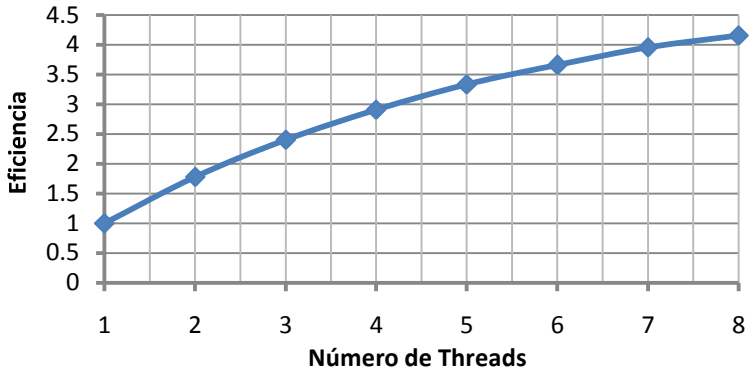

Figura 5.14 (b) Eficiencia en la computadora de 8 núcleos para el sistema de 118 nodos

5.3.4.2 Pruebas con 4 threads

Tabla 5.20 (a) Tiempo requerido por el método DN para solucionar el sistema de 118 nodos en la computadora de 4 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	2	3	4	5	6	7	8	9	10	prom
1	15126	12137	15117	15120	15121	15115	15124	15122	15106	15139	15118.875
2	8827	8839	8692	8815	9470	9136	8771	8702	8779	8923	8849
3	8827	8813	8841	8801	8842	8793	8793	8785	8876	8799	8813.625
4	9135	9031	9055	9098	9094	9046	9087	9223	9006	9049	9074.375

Tabla 5.20 (b) Tiempo relativo y eficiencia en la computadora de 4 núcleos para el sistema de 118 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	15118.875	1	1
2	8849	0.58529487	1.70854051
3	8813.625	0.58295508	1.71539803
4	9074.375	0.60020173	1.66610648

En la Tabla 5.20 (a) y 5.20 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 118 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.15 (a) y la Figura 5.15 (b) que nos muestran respectivamente la

velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 4 núcleos, se puede observar que se reduce el tiempo de ejecución a cerca del 60% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia de 1.66.

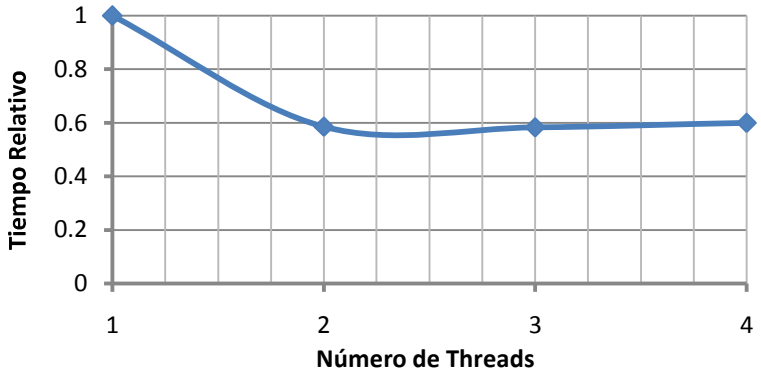

Figura 5.15 (a) Reducción del tiempo en la computadora de 4 núcleos para el sistema de 118 nodos

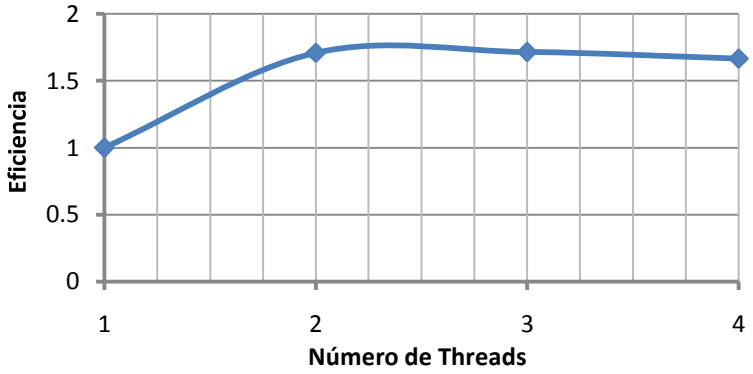

Figura 5.15 (b) Eficiencia en la computadora de 4 núcleos para el sistema de 118 nodos

5.3.4.3 Pruebas con 2 threads

Tabla 5.21 (a) Tiempo requerido por el método DN para solucionar el sistema de 118 nodos en la computadora de 2 núcleos

No Threads	Tiempo de ejecución (milisegundos)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**prom**
1	11317	10925	11418	11312	11371	10940	11415	10820	11332	11218	**11228.75**
2	6205	6215	6208	6208	6210	6079	6263	6212	6172	6303	**6211.625**

Tabla 5.21 (b) Tiempo relativo y eficiencia en la computadora de 2 núcleos para el sistema de 118 nodos

No Threads	Tiempo promedio (mseg)	Tiempo Relativo	Eficiencia
1	11228.75	1	1
2	6211.625	0.55318936	1.80769927

En la Tabla 5.21 (a) y 5.21 (b) se tienen los distintos tiempos que se requirieron para solucionar el sistema de 118 nodos con el método DN. De estos datos se obtienen las gráficas de la Figura 5.16 (a) y la Figura 5.16 (b) que nos muestran respectivamente la velocidad y la eficiencia obtenida al aumentar el número de procesadores. En este caso las pruebas se realizaron en la computadora de 2 núcleos, se puede observar que se reduce el tiempo de ejecución a 50.31% del tiempo que nos tomaría si se realizara la simulación con un solo núcleo, esto nos da una eficiencia cercana a 1.8. De nuevo se tiene una gran eficiencia para una computadora que sólo cuenta con 2 núcleos.

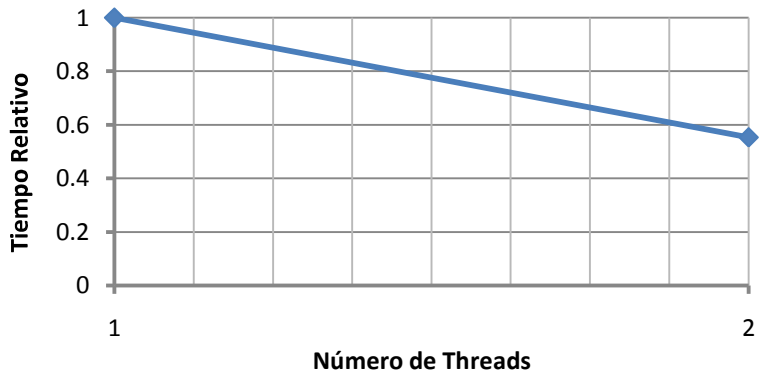

Figura 5.16 (a) Reducción del tiempo en la computadora de 2 núcleos para el sistema de 118 nodos

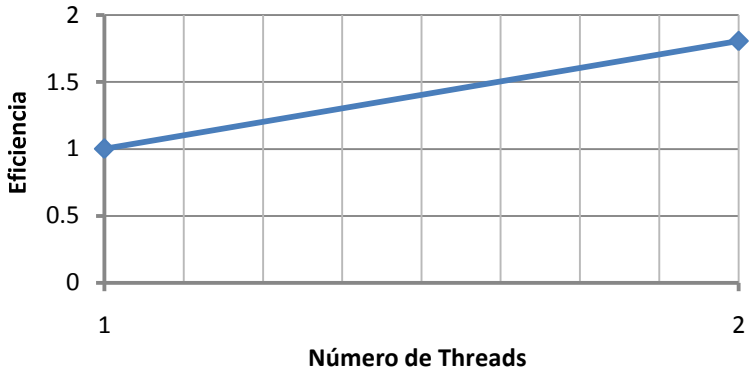

Figura 5.16 (b) Eficiencia en la computadora de 2 núcleos para el sistema de 118 nodos

5.4 Comparación de eficiencia entre PVM y Multithreading

Para poder hacer una comparación directa entre los resultados obtenidos en las plataformas de procesamiento en paralelo PVM y Multithreading a continuación se muestran los resultados obtenidos para cada uno de los sistemas eléctricos simulados.

Tabla 5.22 Comparación de eficiencias obtenidas en PVM con 4 procesadores y en Multithreading con 4 hilos.

Hilo / Proceso	Sistema 14 nodos		Sistema 30 nodos		Sistema 57 nodos		Sistema 118 nodos	
	PVM	MT	PVM	MT	PVM	MT	PVM	MT
1	1	1	1	1	1	1	1	1
2	1.54	1.52	1.82	1.70	1.88	1.82	1.92	1.77
3	1.87	1.83	2.48	2.14	2.66	2.46	2.72	2.40
4	2.02	1.88	3.10	2.34	3.34	2.97	3.57	2.90

De la tabla 5.22 se aprecia que la plataforma PVM fue más eficiente en las simulaciones realizadas para los sistemas eléctricos que fueron caso de estudio en esta tesis.

5.5 Comparación de los resultados obtenidos con PVM y Multithreading contra los obtenidos en ATP-DRAW

Como parte final del análisis de los casos de estudio se hace una validación de los resultados con los obtenidos de un simulador comercial como lo es ATP-DRAW.

En la Figura 5.17 se muestran sobrepuestas las formas de onda de algunas variables de estado que describen al circuito de 3 nodos modificado mostrado en la Figura 4.1 y su espectro armónico. Los datos utilizados para graficar fueron obtenidos con PVM y con ATP. El hecho de que ambas gráficas seas idénticas valida que los datos obtenidos en las simulaciones son datos correctos que se pueden obtener de igual manera con el simulador ATP.

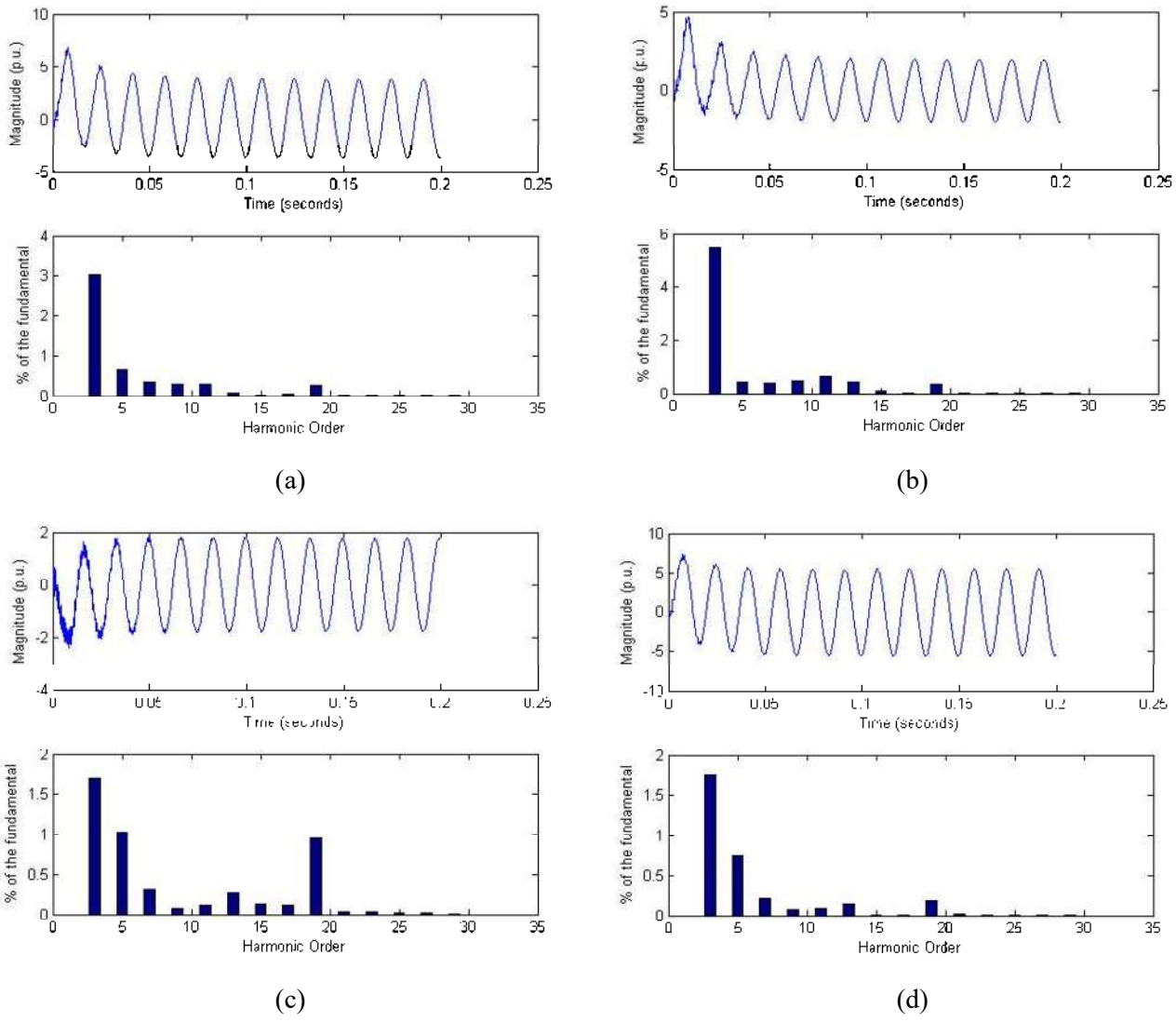

Figura 5.17 *Formas de onda de algunas variables de estado y su espectro armónico del circuito de muestra de 3 nodos de la IEEE modificado. (a) Corriente del nodo 1 al nodo 2 (b) Corriente del nodo 1 al nodo 3 (c) corriente del nodo 2 al nodo 3 (d) corriente en el horno de arco eléctrico*

Capítulo 6

Conclusiones

6.1 Conclusiones generales

En el desarrollo de esta tesis se puede observar que el procesamiento en paralelo es una muy buena opción para simular sistemas eléctricos. Al buscar más poder de cómputo el procesamiento en paralelo es una gran alternativa a los sistemas de un solo procesador, sobre todo porque el procesamiento en paralelo reduce el tiempo de ejecución obteniendo así una mayor eficiencia.

Particularmente se puede apreciar, para el sistema de procesamiento en paralelo PVM, que el método de Diferenciación Numérica de Newton al ser ejecutado en 4 procesadores aumenta su eficiencia a cerca de 3.

En los casos de estudio se puede ver el gran poder que tiene la programación en paralelo ya que se reduce el tiempo de ejecución de un programa casi al 30% de lo que se llevaría en una computadora con un solo procesador. A esto se le suma la ventaja de que se puede construir un sistema PVM con computadoras que prácticamente se pueden considerar obsoletas. De esta manera se aprovechan los todos los recursos que estén disponibles.

6.2 Conclusiones particulares

En lo particular pude observar la gran ventaja que se tiene al aprovechar la programación en paralelo. Ésta herramienta es una gran alternativa para simular sistemas eléctricos actualmente ya que las nuevas computadoras se ven limitadas en cuando a velocidad de procesamiento y a escala.

6.3 Trabajos Futuros

En trabajos futuros se recomienda abordar más plataformas en paralelo para la resolución de sistemas eléctricos así como ver las ventajas que se tienen utilizando el método de Diferenciación Numérica con más procesadores. Esta tesis se enfocó más a ver la ventaja en cuanto a velocidad de procesamiento se refiere, pero esto no quiere decir que sean las únicas pruebas que se puedan realizar.

Referencias

[Aprille y Trick 1972]

Aprille T.J. y Trick T. N. “A Computer Algorithm to Determine the Steady State Response of Nonlinear Oscillators”, IEEE Transactions on Circuit Theory, Vol. CT-19, No. 4, Julio 1972, págs. 354-360.

[Arrillaga *et al.* 1995]

Arrillaga J., Medina A., Lisboa M., Cavia M., Sanchez P., “The Harmonic Domain. A Frame of Reference for Power System Harmonic Analysis”. IEEE Transactions on Power Systems, Vol. 10, No. 1, Febrero 1995, págs. 433-440.

[Chapra y Canale 2003]

S. C. Chapra y R. P. Canale, Métodos Numéricos para Ingenieros, México: McGraw-Hill, 2003

[Chua y Pen-Min 1975]

Chua L.O. y Pen-Min L. “Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques”, Prentice-Hall, EUA, 1975.

[De Carlo y Saeks 1981]

De Carlo R.A. y Saeks R. “Interconnected Dynamical Systems”, Marcel Dekker, Inc, EUA, 1981.

[Dommel 1969]

Dommel, H. W., “Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks”, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-88, No. 4, Abril 1969, págs.388-399.

[García *et al.* 2001]

Garcia N., Acha E. y Medina A., "Swift Time Domain Solutions of Electric Systems Using Parallel Processing", Proceedings of the Sixth Internacional Conference IASTED, Rodas, Grecia, Julio 2001, Págs. 172-177.

[Geist 1994]

A. Geist, PVM Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing, Massachusetts: The MIT Press, 1994

[Hornbeck 1975]

Hornbeck R. "Numerical Methods". Quantum Publishers, Inc. 1975. EUA.

[Jin 1994]

Jin L., "Parallel Processing: Exploring the Archictectures and Algorithms Close Relationship", IEEE POTENTIALS, Diciembre 94-Enero 95, págs. 17-20.

[Lemaitre y Thomas 1996]

Lemaitre C. y Thomas B., "Two Applications of Parallel Processing in Power System Computation", IEEE Transactions on Power Systems, Vol. 11, No. 1, Febrero 1996, págs. 246-253.

[Martínez-Velasco 1997]

Martínez-Velasco J. A. "Computer Analysis of Electric Power System Transients: Selected Readings", IEEE PRESS.

[Medina, Ramos-Paz y Fuerte-Esquivel 2003a]

Medina, A.; Ramos-Paz, A.; Fuerte-Esquivel, C.R., "Periodic Steady State Solution of Electric Systems with Nonlinear Components Using Parallel Processing". IEEE Transactions on Power Systems, Vol. 18, No. 2, Mayo 2003, págs. 963 – 965.

[Nakhla y Branin 1977]

Nakhla M. S. y Branin F. H., “Determining the Periodic Response of Nonlinear Systems by a Gradient Method”, Circuit Theory Appl. Vol. 5, 1977, págs. 255 – 273.

[Patel y Trivedi 1990]

R. J. Patel y H. P. Trivedi, "Parallel Processing: An Overview", *IEEE Potenctials*, págs. 40-42, Octubre 1990.

[Ramos-Paz 2002]

Ramos-Paz A., “Aplicación de Técnicas de Procesamiento en Paralelo a la Solución en Estado Periódico Estacionario de Redes Eléctricas con Componentes no Lineales y Variantes en el Tiempo”, Tesis de Maestría, Facultad de Ingeniería Eléctrica, U.M.S.N.H. 2002.

[Ramos-Paz 2007]

Ramos-Paz A., “Técnica para la Generación Automática de Ecuaciones Diferenciales No Autónomas para Representar el Comportamiento Dinámico de Sistemas Eléctricos No-Lineales Incorporando Herramientas Avanzadas de Computo”, Tesis de Doctorado, Facultad de Ingeniería Eléctrica, U.M.S.N.H. 2007.

[Semlyen y Medina 1995]

Semlyen A., Medina A., “Computation of the Periodic Steady State in Systems with Nolinear Components Using a Hybrid Time and Frequency Domain Methodology”, IEEE Transactions on Power Systems, Vol. 10, No. 3, Agosto 1995, págs. 1498-1504.

[Skelboe 1980]

Skelboe S., “Computation of the Periodic Steady-State Response of Nonlinear Networks by Extrapolation Methods”, IEEE Transactions on Circuits and Systems, Vol. CAS-27, Marzo 1980, págs. 161-175.

[Sundaram 1996]

Sundaram R.K. "A First Course in Optimization Theory" Cambridge University Press., Estados Unidos de América 1996.

[Usaola 1990]

Usaola-García J. , "Régimen Permanente de Sistemas Eléctricos de Potencia con Elementos No lineales Mediante un Procedimiento Híbrido de Análisis en los Dominios del Tiempo y de la Frecuencia", Tesis de Doctorado, Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Industriales, 1990.

[Walmsley 2001]

M. Walmsley, Multi-threaded programming in C++, Londres: Springer-Verlag London, 2001.

[Wylie 1951]

Wylie C.R., "Advanced Engineering Mathematics", Mc. Graw Hill. Japón 1951.

[Zamora *et al.* 2005]

Zamora M.A., et al., "Simulación de Sistemas Eléctricos", Prentice Hall. España 2005.