UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO

FACULTAD DE INGENIERIA ELECTRICA

APLICACIÓN DE TÉCNICAS DE PROCESAMIENTO EN PARALELO BASADAS EN MULTITHREADING A LA SOLUCIÓN DEL PROBLEMA DE FLUJOS DE POTENCIA MONOFÁSICOS USANDO EL MÉTODO DE NEWTON-RAPHSON

TÉSIS

Que para obtener el título de

INGENIERO ELECTRICISTA

Presenta

ISMAEL BAZAN ACOSTA

Asesor de Tesis

DR. ANTONIO RAMOS PAZ

Diciembre de 2008

Agradecimientos

A Dios por brindarme el privilegio de estar vivo.

A mis padres que nunca han dudado en apoyarme en mis proyectos.

A mis hermanos que siempre han estado conmigo y me han dado ánimos para seguir luchando.

A mis amigos y maestros que han contribuido a mi formación social y profesional.

A ti que llegaste al final, pero no por eso ha dejado de ser importante tu participación para la culminación de este trabajo.

Dedicatoria

Con todas mis fuerzas se la dedico a toda mi familia.

Resumen

Uno de los estudios más importantes en el análisis de los Sistemas Eléctricos de Potencia es el estudio de los flujos de potencia monofásicos, ya que este permite conocer las magnitudes y ángulos de voltaje en todos los nodos de la red y, a partir de ello, poder calcular las corrientes que circulan por todas las líneas que conectan a las diversas centrales generadoras con los centros de consumo. Además, el estudio de flujos de potencia monofásico es la base para otros estudios de gran importancia en los sistemas eléctricos tales como flujos de potencia trifásicos, estabilidad, análisis armónicos, estudios de corto circuito, etc.

Este trabajo presenta la aplicación de técnicas de procesamiento en paralelo a la solución del problema de flujos de potencia monofásicos aplicando el método de Newton-Raphson. Todo esto se realiza con la intención de ahorrar tiempo de procesamiento en la solución del problema de flujos de potencia.

Se aplica la técnica propuesta a la solución de los sistemas de prueba IEEE de 14, 30, 57 y 118 nodos.

Los resultados obtenidos se presentaran en forma de eficiencia relativa para diferentes equipos de cómputo con los diferentes sistemas de prueba.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos y Abreviaturas

K	kilo
M	mega
G	giga
m	metros
W	watts
Hz	hertz
€	euro
W-h	watt-hora
g	aceleración de la gravedad
t	tiempo
d/dt	derivada con respecto del tiempo
KD	ganancia derivativa
KI	ganancia integral
KP	ganancia proporcional
Ω	ohms
δ	ángulo de potencia
X	reactancia
R	resistencia
P	potencia real
$\frac{dy}{dx}$	derivada parcia

Capítulo 1

Introducción

Dentro de los estudios que se realizan dentro de Ingeniería Eléctrica y sus áreas afines existen problemas de mucha importancia debido a su complejidad y a su efecto dentro del sistema a estudiar. La solución del problema de flujos de potencia en un sistema eléctrico de potencia (SEP) es uno de ellos, ya que, en base a los resultados obtenidos de este, se realizan estudios tales como: flujos óptimos, despacho económico, estabilidad, flujos de potencia trifásicos, análisis armónicos, etc.

Al realizar un estudio de flujos de potencia esperamos obtener voltajes, ángulos de fase, potencia real y potencia activa de todos y cada uno de los nodos que forman el sistema a estudiar.

Como ya es bien sabido, el estudio de los flujos de potencia se puede realizar a un SEP tanto de pequeñas dimensiones, como a uno que incluya el total de la red de un país (como el que hace Comisión Federal de Electricidad (CFE) a la red de nuestro país). Cuando estamos trabajando en una red pequeña no existe ninguna dificultad, ya que estaríamos trabajando con matrices de bajas dimensiones.

Adicional al problema de las dimensiones de la matriz, recordemos que con los elementos que existen en la matriz debemos realizar operaciones matemáticas; con lo que particularidades como el espacio de memoria y tiempo de ejecución dentro de la computadora se convierten en variables de gran importancia. Es por eso que se han desarrollado métodos computacionales tales como la Programación Orientada a Objetos (POO), Técnicas de Dispersidad y Procesamiento en Paralelo para el estudio de diferentes problemas matemáticos y, en particular, para el estudio de los flujos de potencia en un sistema eléctrico.

1.1 Antecedentes

El estudio del problema de los flujos de potencia no es algo nuevo y existen una infinidad de trabajos, publicaciones e investigaciones acerca de este tema; algunos de estos estudios se referencian a continuación:

- [Dommel Herman 1968] utiliza las soluciones optimas para resolver el problema de flujos de potencia.
- [Tinney William 1967] muestra dentro de sus trabajos la solución de los flujos de potencia mediante el método de Newton.
- [Keyhani A. 1989] explica las diferentes técnicas de la evaluación de los flujos de potencia en computadoras personales (PC's). (Apartado 1.1.2)
- [Dixit Shishir 2006] utiliza Lógica Difusa para resolver el problema de flujos de potencia.

Esto lo podemos observar al investigar en cualquier literatura donde se hagan estudios de SEP's y encontremos que en todos ellos hay un estudio minucioso de este tipo de problemas. La innovación en este tipo de estudios, y a la cual se enfoca esta Tésis, consiste en la optimización de los métodos computacionales utilizados en la solución numérica para resolver el problema de los flujos de potencia.

En particular en este trabajo de Tesis, se va a trabajar con el método de procesamiento en paralelo que junto con otras técnicas computacionales actuales favorecen al momento de la solución de métodos numéricos de grandes dimensiones los tiempos de solución y reducen el problema de la memoria computacional que las técnicas computacionales convencionales presentan. Entre estas otras técnicas "modernas" de programación son la POO y las técnicas de dispersidad, las cuales junto con el Procesamiento en Paralelo van de la mano y nos reducen en gran manera el esfuerzo computacional en el momento de programar un computador.

1.1.1 Procesamiento en Paralelo

El procesamiento en paralelo consiste en resolver un gran problema en base a su división en una serie de problemas pequeños que pueden ser resueltos de manera simultánea a través de algún sistema de cómputo que permita llevar a cabo esta acción. Lo cual se resuelve a través de una cooperación de un gran número de procesadores interconectados [Jin 1994].
El incremento en la demanda para resolver grandes problemas en varias aplicaciones y los avances en las tecnologías computacionales (especialmente la muy grande escala de integración), han contribuido al desarrollo del procesamiento en paralelo.
Se cree que, para un sólo procesador, el incremento de la frecuencia de reloj es limitada por la velocidad de la luz, y la reducción del tamaño físico es limitado por el diámetro de las moléculas. Es por eso que, los procesadores tradicionales, son incapaces de proveer toda la potencia computacional necesitada. Debido a esto, el procesamiento en paralelo es la única forma de sobrepasar las limitaciones de las arquitecturas tradicionales.

1.1.2 Resultados obtenidos por Keyhani

Como ya se comentó en el apartado 1.1 en 1989 Keyhani A. realizó estudios acerca de la evaluación de los flujos de potencia, utilizando diferentes métodos de solución, ayudado de una computadora personal, la cual era de las siguientes características: marca IBM/AT, 30 MB de disco duro, 1.2 MB de disco floppy y 1 MB de memoria.

Para esto, nos dice que los requerimientos de memoria en Bytes para la solución de flujos de potencia utilizando el método de Newton-Raphson es igual a $240n + 240L$; donde n es igual al número de nodos y L al número de líneas de la matriz Jacobiana. Así, nos entrega los resultados que se muestran en la Tabla 1.1 para los diferentes IEEE Sistemas de prueba.

Tabla 1.1 Resultados obtenidos por Keyhani A. para diferentes Sistemas de prueba

Sistema de Prueba	Tiempo (s)
IEEE 5 NODOS	0.83
IEEE 14 NODOS	14.33
IEEE 30 NODOS	102.21
IEEE 57 NODOS	1019.25
IEEE 118 NODOS	NO DISPONIBLE

1.2 Objetivo

Los objetivos principales de esta Tésis son:

- Aplicar técnicas de procesamiento en paralelo al problema de flujos de potencia monofásicos.
- Utilizar el método de Newton-Raphson en su forma polar para la solución del problema de flujos de potencia.
- Desarrollar una base para resolver otros tipos de problemas de los sistemas eléctricos de potencia.

Es asi que, de acuerdo a los resultados mostrados en el apartado 1.1.2, el objetivo es realizar estas mismas pruebas (a excepción de el sistema de 5 nodos) con diferentes equipos y utilizando diferentes herramientas de programación, para eficientar en cuanto a los tiempos de ejecución la solución del problema de flujos de potencia monofásicos. Los resultados finales se muestran en el Capítulo 5 de esta Tesis.

1.3 Justificación

El estudio del problema de los flujos de potencia no radica solamente, como ya se mencionó, en resolverlos para el estudio específico de este tema, sino que es una herramienta de gran importancia que nos facilita en gran manera hacer aplicaciones en estudios tales como el de corto circuito (monofásico y trifásico), estabilidad en los Sistemas

de Potencia, condiciones de estado estable, continuidad en las líneas de transmisión y los Sistemas de Potencia, etc.
Se podría pensar que los equipos de computación crecen a gran escala y que no es necesario utilizar otras plataformas, pero debemos tomar en cuenta que el límite para el crecimiento de los equipos consiste en su tamaño físico y en su velocidad. El tamaño está limitado por el tamaño de la molécula, y la velocidad del procesador está limitada por la velocidad de la luz.

De acuerdo a esto, este trabajo pretende utilizar otras alternativas, tal como la programación en paralelo, para poder dar solución a grandes problemas utilizando varios procesadores (no importa que no sean muy grandes), con equipos comunes.

Al hacer este tipo de trabajos, se busca encontrar una plataforma de procesamiento en paralelo que cumpla con los requisitos de ser eficiente y de fácil acceso para cualquier usuario; asi como de que exista la posibilidad de realizar el análisis de cualquier sistema de potencia sin tener la necesidad de algún tipo de cómputo de acceso limitado. Al mismo tiempo, se pretende que este trabajo sirva como plataforma para estudios posteriores relacionados con el tema y/o con fines didácticos en el estudio de los flujos de potencia.

1.4 Metodología

De acuerdo a la forma tradicional de programación tenemos que la programación de una computadora se realiza programando en bloque lo que necesitamos para formar un programa completo. En la programación orientada a objetos, ya se hace por bloques individuales que pueden ser reutilizados en futuros programas de las mismas aplicaciones, pero con el mismo problema de que el procesador debe de terminar de leer todas las instrucciones del código para poder correr el programa.

La metodología que se plantea para los tipos de estudios especificados en este trabajo de tesis es la de paralelizar el código a nivel de instrucciones del programa; es decir paralelizar dentro del código de programa lo que en forma tradicional tiene que seguir un orden; tal y

como en los ciclos. Al paralelizar los ciclos, dividiríamos el tiempo y el espacio de memoria que utiliza generalmente un sólo procesador, en varios procesadores.

La forma de llevar a cabo estos objetivos será ayudándonos de una herramienta de paralelización que contiene el lenguaje de programación C y que se denomina Multithreading. Es decir, todo nuestro esfuerzo estará basado en la filosofía de un refrán popular que se enuncia como "divide y vencerás".

1.5 Contenido de la Tesis

En el Capítulo 1 se da una explicación a grandes rasgos de lo que se pretende llevar a cabo con la realización de este trabajo. Se da una breve reseña de lo que consiste el problema de los flujos de potencia y del procesamiento en paralelo; así como sus principales utilizaciones y antecedentes históricos. También se plantea la forma en que se trabajará, dando la justificación y la metodología a desarrollar.

En el Capítulo 2 se lleva a cabo un estudio más profundo de los flujos de potencia monofásicos y con sus respectivas ecuaciones y principales métodos de solución.

En el Capítulo 3 se presenta una recopilación acerca del procesamiento en paralelo aplicando multithreading (multithreads), sus principales características, las razones para usar multithreading y de las aplicaciones más recientes de este modo de programación.

En el Capítulo 4 se da a conocer el esquema que se esta proponiendo para la realización de este trabajo y que consiste en programar en C, y utilizando multithreading, el método de Newton-Raphson para la solución del problema de flujos de potencia monofásicos.

En el Capítulo 5 se presentan los casos de estudio para llevar a cabo los objetivos planteados; y que consisten en resolver los sistemas de prueba IEEE de 14, 30, 57, y 118 nodos con el programa descrito en el Capítulo 4; además de realizarlas con diferentes

equipos de cómputo existentes en las instalaciones de la División de Posgrado de la Facultad de Ingeniería Eléctrica.

En el Capítulo 6 se muestran las conclusiones obtenidas en este trabajo.

Es por esto, que se pretende inicialmente dar una idea de lo que se va a tratar a lo largo del trabajo de Tesis, que se hará en un total de 6 capítulos, dentro de los cuales se harán los estudios y definiciones de la terminología utilizada referente a flujos de potencia, procesamiento en paralelo y mutithreading. Además de que en el Capítulo 5 se llevarán a cabo las pruebas finales del sistema utilizado para poder corroborar si se cumplieron o no los objetivos propuestos.

Capítulo2

Flujos de Potencia Monofásicos

2.1 Introducción

Cuando existe un sistema eléctrico de potencia, sean cuales sean sus dimensiones, existen parámetros que debido a sus características, son de vital importancia saber su comportamiento. Dichos parámetros son magnitud y ángulo de voltaje y la potencia en cada nodo. Los estudios de los flujos de potencia son necesarios para planificar y controlar un sistema existente, así como planear una futura expansión del sistema. El problema consiste en determinar las magnitudes y ángulos de fases de los voltajes en cada bus y el flujo de potencia activa y reactiva en cada una de las líneas de transmisión.

En la solución de un problema de flujo de potencia, se asume que el sistema es operado en condiciones balanceadas y se utiliza el modelo de una sola fase. En cada bus existen cuatro cantidades asociadas, que son la magnitud de voltaje (V), el ángulo de fase (δ), potencia real (P) y potencia reactiva (Q). En todos los sistemas, los nodos son clasificados dentro de 3 tipos:

1. Nodo slack o de referencia: en toda la red, sólo existe un nodo como este y es tomado como referencia donde la magnitud y ángulo de fase son especificados (por lo general $1.0 \angle 0^{o}$ por unidad). En este bus se nota la diferencia entre la potencia generada y la consumida en las cargas, debido a las perdidas en el sistema y el problema del flujo de potencia es calcular P y Q en este nodo.
2. Nodos de carga: en estos nodos, la potencia activa y reactiva son conocidas y la magnitud y el ángulo de fase del voltaje son los parámetros a calcular. La mayoría de los nodos dentro de un sistema son de este tipo.
3. Nodos de Voltaje Controlado: estos son nodos generadores, tales como los conectados a generadores, capacitores en derivación o sistemas compensadores estáticos de VARs. En estos nodos, la potencia real y la magnitud del voltaje son especificadas y los parámetros a conocer son los ángulos de fase de los voltajes y la potencia reactiva.

El objetivo del cálculo de los flujos de potencia es determinar las características de operación en estado estable de generación/transmisión en un sistema de potencia para un grupo de nodos de carga dado. Las cargas son normalmente especificadas por su constante potencia activa y reactiva requerida, asumiendo que no se verán afectadas por pequeñas variaciones de voltaje y frecuencia ocurridas durante la operación de estado estable.

Normalmente, los flujos de potencia son representados en su forma básica analítica, con un sistema lineal, bilateral y con sus parámetros balanceados; sin embargo, la operación del voltaje y la potencia hacen el problema no lineal y la solución numérica debe ser iterativa.

Existen propiedades requeridas en la solución de un método de flujo de potencia.

- Alta rapidez computacional. Es importante cuando enfrentamos a sistemas muy grandes, aplicaciones en tiempo real, múltiples casos de flujos de potencia y en aplicaciones interactivas.
- Bajo almacenamiento computacional. Es importante en grandes sistemas y en el uso de computadoras con poca capacidad de memoria.
- Confianza en la solución. Es necesario que una solución sea obtenida para problemas mal condicionados, en estudios futuros y aplicaciones de tiempo real.
- Versatilidad. Es una habilidad en los flujos de potencia para la manipulación convencional y reexaltaciones especiales, y su acomodo para la incorporación en procesos mas complicados.
- Simplicidad. La facilidad de codificar dentro de un programa computacional del algoritmo de flujo de potencia.

El tipo de solución requerida para un flujo de potencia, es lo que va a determinar el método usado.

Exacto	o	Aproximado
Ajustado	o	No ajustado
En línea	o	Fuera de línea
De un sólo caso	o	De múltiples casos

En la primera columna son requerimientos necesarios tomando en cuenta condiciones óptimas de flujo de potencia y estudios de estabilidad, y la segunda columna son condiciones necesarias para contribuir en la seguridad de un sistema de potencia.

El primer método de solución digital práctico para los flujos de potencia son los métodos iterativos de matriz de admitancias (Ymatriz). Es conveniente debido a los bajos requerimientos de capacidad de memoria, pero tiene la desventaja de converger muy poco o nada. Los métodos de matriz de impedancias (Zmatriz) fueron desarrollados para vencer el problema de la formalidad, pero la rapidez y el almacenamiento son sacrificados en sistemas muy grandes.

Los métodos de Newton-Raphson fueron desarrollados para obtener una convergencia de muy alta satisfacción. Sin embargo, no son competitivos bajo programas de dispersidad y en donde se introduce la eliminación Gaussiana óptimamente ordenada, lo cual reduce almacenamiento y tiempos de solución.

Programación no lineal y métodos híbridos también han sido desarrollados, pero ha creado sólo intereses académicos y no ha sido aceptado para usos industriales de los flujos de potencia. El método de Newton-Raphson y técnicas derivadas de su algoritmo satisfacen los requerimientos de los tipos de solución y de propiedades de programación de una manera mejor que las utilizadas anteriormente y las están reemplazando gradualmente.

2.2 Definición convencional del problema de flujos de Potencia

Uno de los criterios para establecer el concepto de estado estacionario, es que el sistema está operando a frecuencia nominal y constante, de modo que esto implica que existe un balance global de potencia, donde la generación debe ser igual a la carga en el sistema más las pérdidas y que matemáticamente puede definirse como:

$$\sum_{i=1}^{n} P_{Gi} = \sum_{i=1}^{n} P_{Di} - \sum_{m=1}^{n_l} P_{Lm} \tag{2.1}$$

Donde G indica generación, D carga, L pérdidas, n número de nodos y n_l el número de elementos de transmisión de potencia en el sistema eléctrico.

Básicamente, el problema de flujos de potencia convencional puede definirse como el cálculo de voltajes nodales y, posteriormente, el de flujos de potencia a través de cada elemento de la red de transmisión, para valores conocidos de generación y carga nodales en MW y MVAr, en un instante de tiempo específico.

La solución del problema puede o no estar sujeta a restricciones de red, tales como límites de generación de potencia activa y reactiva, magnitud de voltajes complejos nodales, así como flujos en elementos, entre otras.

2.3 Formulación del problema

Matemáticamente, el problema consiste en resolver un conjunto de ecuaciones algebraicas no lineales y diferenciables, cuyó orden depende de la formulación utilizada.

En términos de potencia, el balance en cualquier nodo i del sistema eléctrico es:

$$S = P + jQ \tag{2.1}$$

$$S = V^T + I^* \tag{2.1.a}$$

$$S = V^T + YV^* \tag{2.1.b}$$

Donde:

$S =$ vector de inyeccion de potencia compleja

$P =$ vector de inyeccion de potencia real

$Q =$ vector de inyeccion de potencia reactiva

$I =$ vector de inyeccion de corriente

$V =$ vector de voltajes nodales

$Y = G + jB$ es el sistema de matriz de admitancias

En sistema de coordenadas polares, el voltaje complejo y las potencias real y reactiva pueden escribirse como:

$$Vi = Vi(Cos\theta_i + jSen\theta_i) \tag{2.2}$$

$$P_i^{cal} = Vi \sum_{j-1}^{n} V_j \, (G_{ij} \, Cos\theta_{ij} + B_{ij} \, Sen\theta_{ij}) \tag{2.3}$$

$$Q_i^{cal} = Vi \sum_{j\ 1}^{n} V_j \, (G_{ij} \, Sen\theta_{ij} - B_{ij} \, Cos\theta_{ij}) \tag{2.4}$$

Donde $\theta = \theta i - \theta j$.

En sistemas de coordenadas cartesianas las ecuaciones anteriores pueden reescribirse como:

$$V_i = e_i + jf_i \tag{2.5}$$

$$P_i^{cal} = e_i \sum_{j=1}^{n} (G_{ij} e_j - B_{ij} f_j) + f_i \sum_{j=1}^{n} (G_{ij} f_j + B_{ij} e_j) \tag{2.6}$$

$$Q_i^{cal} = f_i \sum_{j-1}^{n} (G_{ij} e_j - B_{ij} f_j) - f_i \sum_{j-1}^{n} (G_{ij} f_j + B_{ij} e_j) \tag{2.7}$$

Donde n es el número de nodos en el sistema.

2.4 Cálculo de flujos de potencia

Después de obtener una solución para los voltajes, se podrá calcular los flujos de potencia en todos los elementos de la red eléctrica, donde cada uno de ellos se representa por su modelo π equivalente de secuencia positiva, tal como se muestra en la Figura 2.1.

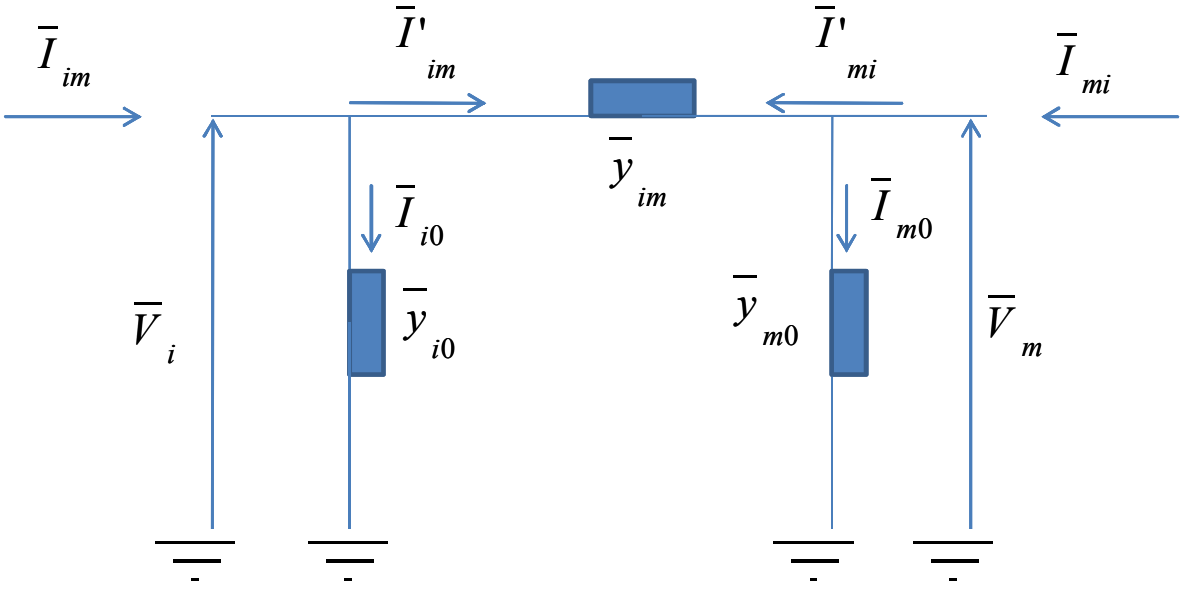

Figura 2.1 Modelo π de un elemento de transmisión para calcular los flujos de potencia a través de él.

Debe recordarse que las admitancias del circuito π son parámetros físicos del elemento, y en el que existen pérdidas, de modo que, términos generales, se cumple que $P_{im} \neq P_{mi}$ y que $Q_{im} \neq Q_{mi}$.

Por otra parte, se tiene las relaciones de corriente siguientes:

$$\bar{I}_{im} = \bar{I}_{i0} + \bar{I}'_{im} \tag{2.8}$$

y la potencia que se envía del nodo *i* al nodo *m*:

$$\bar{S}_{im} = \bar{V}_i \bar{I}^*_{im} \tag{2.9}$$

Además, se tiene lo siguiente:

$$\bar{I}_{i0} = \bar{y}_{i0} \bar{V}_i \tag{2.10}$$

$$\bar{I}'_{im} = \bar{y}_{im}(\bar{V}_i - \bar{V}_m) \tag{2.10.a}$$

Determinando los conjugados de las corrientes, y substituyendo en Ec. (2.9):

$$\bar{S}_{im} = \bar{V}_i \left[\bar{y}^*_{im} \bar{V}^*_i + \bar{y}^*_{im} \left(\bar{V}^*_i - \bar{V}^*_m \right) \right] \tag{2.11}$$

obteniéndose:

$$\bar{S}_{im} = V_i^2 \left(\bar{y}^*_{i0} + \bar{y}^*_{im} \right) - \bar{V}_i \bar{V}^*_m \bar{y}^*_{im} \tag{2.12}$$

De igual manera, el flujo de potencia compleja del nodo *m* al nodo *i* será:

$$\bar{S}_{mi} = V_m^2 \left(\bar{y}^*_{m0} + \bar{y}^*_{im} \right) - \bar{V}_m \bar{V}^*_i \bar{y}^*_{im} \tag{2.13}$$

2.5 Método de Newton-Raphson

Las ecuaciones de flujos de potencia dadas por las Ecs. (2.3) y (2.4) se pueden expandir en series de Taylor obteniendose las siguientes aproximaciones de primer orden:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}^{(k)} = \begin{bmatrix} \frac{\partial P}{\partial \theta} & \frac{\partial P}{\partial V} \\ \frac{\partial Q}{\partial \theta} & \frac{\partial Q}{\partial V} \end{bmatrix}^{(k)} \begin{bmatrix} \Delta \theta \\ \Delta V \end{bmatrix}^{(k)} \tag{2.14}$$

Desarrollando las derivadas parciales de F_P y F_Q se notará que pueden expresarse en función de las derivadas parciales de las potencias netas inyectadas:

$$\frac{\partial P_i}{\partial \theta_i} = -Q_i - B_{ii} V_i^2 \qquad (2.15)$$

$$\frac{\partial P_i}{\partial \theta_m} = V_i V_m Y_{im} Sen(\theta_i - \theta_m - \gamma_{im}) \qquad i \neq m \qquad (2.15.a)$$

$$\frac{\partial P_i}{\partial V_i} V_i = P_i + G_{ii} V_i^2 \qquad (2.15.b)$$

$$\frac{\partial P_i}{\partial V_m} V_m = V_i V_m Y_{im} Cos(\theta_i - \theta_m - \gamma_{im}) \qquad i \neq m \qquad (2.15.c)$$

$$\frac{\partial Q_i}{\partial \theta_i} = P_i - G_{ii} V_i^2 \qquad (2.15.d)$$

$$\frac{\partial Q_i}{\partial \theta_m} = -V_i V_m Y_{im} Cos(\theta_i - \theta_m - \gamma_{im}) \qquad i \neq m \qquad (2.15.e)$$

$$\frac{\partial Q_i}{\partial V_i} V_i = Q_i - B_{ii} V_i^2 \qquad (2.15f)$$

$$\frac{\partial Q_i}{\partial V_m} V_m = V_i V_m Y_{im} Sen(\theta_i - \theta_m - \gamma_{im}) \qquad i \neq m \qquad (2.15g)$$

Al observar estas ecuaciones, puede notarse lo siguiente:

- Hay elementos de la matriz de admitancias nodal en todas las ecuaciones, de modo que el Jacobiano tendrá las mismas características de dispersidad.

- Debido a que $(\theta_i - \theta_m) \neq (\theta_m - \theta_i)$, el Jacobiano no resulta simétrico, desde un punto de vista numérico, aunque es simétrico en estructura, es decir, por cada elemento (i,m) distinto de cero, existirá otro que también será distinto de cero en la posición (m,i), pero de valor diferente.

Lo anterior permite, en forma relativamente sencilla, la aplicación de técnicas de dispersidad para resolver Ec. (2.14), evitando tanto el uso excesivo de memoria de computadora como de la ejecución de operaciones aritméticas innecesarias.

El vector independiente de Ec. (2.14) debe calcularse de la siguiente manera:

$$\left.\begin{aligned} \Delta P_i^{(k)} &= P_{Gi} - P_{Di} - P_i^{(k)} \\ \Delta Q_i^{(k)} &= Q_{Gi} - Q_{Di} - Q_i^{(k)} \end{aligned}\right\} \qquad i = 1, \ldots, n \qquad (2.16)$$

donde:

$$\left.\begin{aligned} P_i^{(k)} &= V_i^{(k)} \sum_{m \in i} V_m^{(k)} Y_{im} Cos\left(\theta_i^{(k)} - \theta_m^{(k)} - \gamma_{im}\right) \\ Q_i^{(k)} &= V_i^{(k)} \sum_{m \in i} V_m^{(k)} Y_{im} Sen\left(\theta_i^{(k)} - \theta_m^{(k)} - \gamma_{im}\right) \end{aligned}\right\} \qquad i = 1, \ldots, n \qquad (2.17)$$

Una vez que se resuelve (2.17), será necesario multiplicar cada incremento de voltaje por su magnitud, esto es,

$$\Delta V_i^{(k)} = \Delta V_i^{(k)} V_i^{(k)} \qquad i = 1, \ldots, n \qquad (2.18)$$

y estar en condiciones de actualizar ángulos de fase y magnitudes de voltaje mediante las expresiones:

$$V_i^{(k+1)} = V_i^{(k)} + \Delta V_i^{(k)} \qquad i = 1, \ldots, n \qquad (2.19)$$

$$\theta_i^{(k+1)} = \theta_i^{(k)} + \Delta\theta_i^{(k)}$$

Nótese que Ec. (2.14) son las ecuaciones de balance de potencia nodal evaluadas en cada iteración. El error entre los valores que se calculan para V y θ en cada paso del proceso iterativo y los de solución causa, en consecuencia, la desviación en el balance de potencia nodal. Entonces, es de esperar que, conforme se tiende hacia una solución, las desviaciones de potencia nodal, ΔP y ΔQ, tiendan a cero. Esto permite establecer un criterio de convergencia para el método de Newton en coordenadas polares, expresado en la siguiente forma:

$$Max\left|\Delta P_i^{(k)}\right| \leq tol_P \; i = 1, \ldots, n \qquad (2.20)$$

$$Max\left|\Delta Q_i^{(k)}\right| \leq tol_Q$$

Cuando se cumpla con este criterio, se ha logrado la convergencia hacia una solución, terminando así el proceso iterativo. En caso contrario, se continúa con el proceso, revisándose la convergencia en las iteraciones subsecuentes.

Para iniciar el proceso iterativo, será necesario especificar valores de magnitud y ángulo de fase para cada voltaje nodal del sistema eléctrico. En la práctica, se ha comprobado que el *perfil plano de voltaje* es el más adecuado, si no se conoce otra condición inicial mejor (un estudio de flujos de potencia previo). Esta condición inicial consiste en especificar todas las magnitudes de voltaje en los nodos PQ iguales a 1.0 *p.u.*, mientras que los ángulos de fase en nodos PQ y PV serán iguales a 0.0 grados.

En resumen, el método de Newton en coordenadas polares consta de los siguientes pasos:

1. **Condiciones iniciales**. En este paso, se proporciona o inicializa la información necesaria para arrancar el proceso iterativo, lo cual incluye:

   (a) Parámetros de líneas, transformadores, compensación fija en derivación y cargas.

(b) Para nodos PV, se especifica los límites máximo y mínimo de potencia reactiva de generación.
(c) Número máximo de iteraciones, potencia base y algunos otros parámetros que se consideren necesarios para controlar el proceso iterativo.
(d) Cálculo de la matriz de admitancias nodal, potencias especificadas, especificación de magnitudes y ángulos de fase iniciales.

2. **Proceso iterativo**. Con los valores actuales de V y θ, calcular las potencias netas inyectadas real y reactiva usando (2.16), determinar las desviaciones de potencia nodal, ΔP y ΔQ, con las expresiones (2.15), así como sus máximos y aplicar el criterio de convergencia dado por (2.20). En caso de cumplir con las tolerancias especificadas, ir al paso 4; de otra manera, hacer el siguiente.

3. **Solución del conjunto de ecuaciones lineales y actualización de variables**. Con los valores actuales de V y θ, determinar los elementos del Jacobiano mediante las expresiones (2.15)-(2.15.g), substituir en (2.14) y resolver para $\Delta\theta$ y $\Delta V/V$. Corregir el vector de incrementos de voltaje con (2.19) y actualizar V y θ mediante (2.20). Por último, regresar al paso 2.

4. **Cálculos finales**. Cuando ya se tiene conocido el estado del sistema (voltajes complejos nodales), puede calcularse, entre otras cosas, lo siguiente:

(a) Potencias de generación en el nodo compensador.
(b) Flujos de potencia en elementos de red.
(c) Pérdidas por transmisión.

Existen otros métodos de solución al problema de los flujos de Potencia, como es el caso del método de Gauss-Seidel, pero no es considerado en este proyecto debido a que su solución normalmente representa la aplicación de tolerancias de convergencia demasiado estrechas (del orden de 10^{-6}), que, para problemas de tamaño relativamente grandes, puede causar problemas de convergencia si el programa de computadora desarrollado maneja

variables reales de precisión sencilla. Por otro lado, esta medida no garantiza que el balance de potencia nodal se cumpla con aproximaciones aceptables.

Adicionalmente, aun cuando el método es económico en trabajo computacional por iteración, presenta el inconveniente de que su convergencia es dependiente del número de ecuaciones, de modo que su proceso iterativo tiende a ser demasiado lento para problemas involucrando varios centenares de ecuaciones.

Por las razones anteriores, la aplicación de otros métodos de solución no fue aplicada, a fin de obtener el algoritmo más eficiente.

La Figura 2.2 muestra el diagrama de flujo del método de Newton en coordenadas polares. Los cuatro pasos anteriores representan un método de Newton *sin ajustes* o modelos de dispositivos especiales. Al incluir algún tipo de estos, se debe agregar los pasos necesarios y coordinarlos adecuadamente con los anteriores. Un ajuste imprescindible de agregar es el manejo de límites de potencia reactiva de generación, cuya aplicación se describe en secciones posteriores.

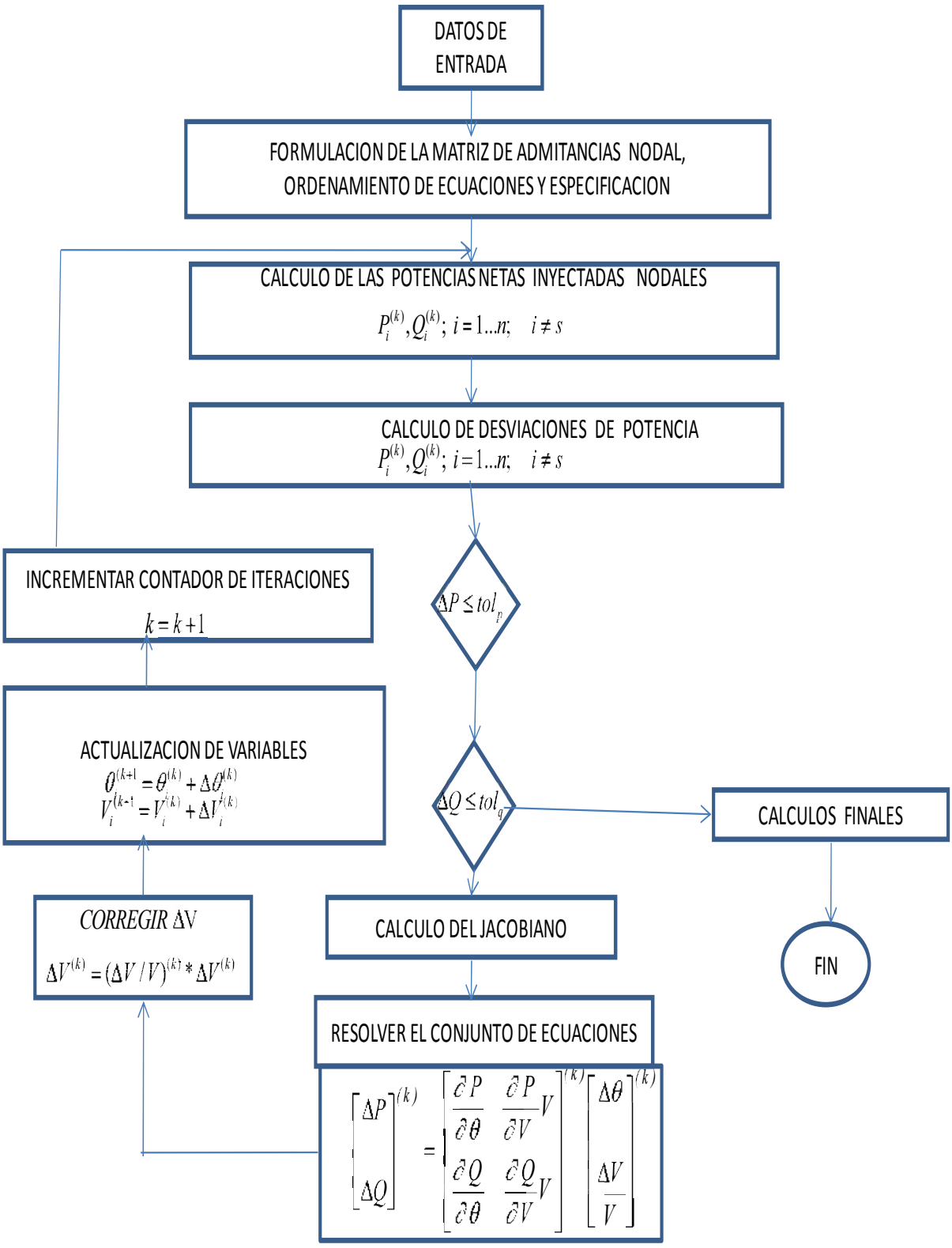

Figura 2.2 Diagrama de flujo del método de Newton en coordenadas polares.

Como se puede apreciar en la explicación del capítulo, existen diversos métodos para la solución del problema de flujos de potencia; pero por sus características ya mencionadas y por el objetivo que persigue este trabajo de Tesis se utilizara el método de Newton-Raphson en su forma polar para programar la solución del problema de flujos de potencia monofásicos.

2.5 Flujos de Potencia y Pérdidas

Después de la solución iterativa de los voltajes nodales, el próximo paso es encontrar los flujos de potencia y las perdidas en las líneas de transmisión. Considerando la línea que conecta los buses i y j en la Fig. 2.3

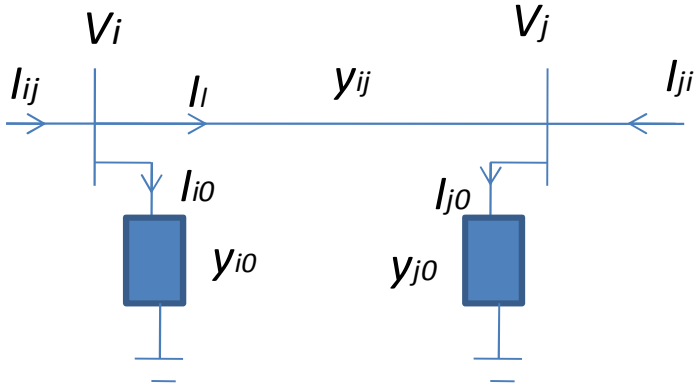

Figura 2.3 Modelo de una línea de transmisión para calcular flujos

La corriente de línea I_{ij}, medida en el nodo *i* y positiva en la dirección i-j esta dada por:

$$I_{ij} = I_l + I_{i0} = y_{ij}(V_i - V_j) + y_{i0}V_i \tag{2.21}$$

De igual manera, la corriente de línea I_{ji}, medida en el nodo j y positiva en la dirección j-i es dada por:

$$I_{ji} = -I_l + I_{j0} = y_{ij}(V_j - V_i) + y_{j0}V_j \tag{2.22}$$

La potencia compleja $\underline{S_{ij}}$, del bus *i* al *j* y S_{ji}, del nodo *j* al *i* son:

$$S_{ij} = V_i I_{ij}^* \tag{2.23}$$

$$S_{ji} = V_j I^*_{ji} \tag{2.24}$$

Las pérdidas de potencia en la línea *i-j* es la suma algebraica de los flujos de potencia determinados por:

$$S_{Lij} = S_{ij} + S_{ji} \tag{2.25}$$

Capítulo 3

Procesamiento en Paralelo basado en Multithreading

3.1 Introducción

Comparado con un proceso, un hilo es una carga ligera en la creación, mantenimiento y manejo de un sistema operativo, debido a que muy poca información es asociada con un hilo. El contexto de un proceso usa muchos recursos del sistema para guardar toda su información. Un hilo también tiene un contexto. Cuando un hilo es definido, un contexto elige entre los hilos que deben ocurrir. Las principales características de los hilos son las siguientes:

- No tiene un espacio de dirección.
- El que debería de ser espacio de dirección es contenido en el espacio de dirección del proceso.
- El contexto de un hilo consiste de sólo una pila, un set de registros y una precedencia.
- El set de registros contiene el programa y/o puntos de instrucción y el punto de apilamiento.
- El texto de un hilo existe en el segmento de texto de su propio proceso.
- Adicionalmente, la demás información, tal y como listas y cuentas son definidas por el proceso.

Un hilo puede leer y escribir en las locaciones de memoria de su proceso, y el proceso tiene acceso a sus datos. Los Hilos pueden crear otros hilos en su proceso. Todos los hilos en un proceso simple son llamados pares. Todos los hilos comparten los recursos y memoria del proceso con los demás hilos. Algunos recursos creados por algún hilo son compartidos

entre sus pares, al mismo tiempo, los hilos pueden suspender, reanudar y terminar a otros hilos fuera de su proceso.

Un hilo es el encargado del proceso. Todo lo que un hilo necesita para funcionar es entregado y definido por el proceso.

Todos los hilos tienen una identificación de Hilo, un set de registros que define el estado del hilo, su precedencia y su pila, lo cual da a cada hilo una identidad separada.

En la Figura 3.1 se muestra las capas lógicas de un proceso comparadas con las capas lógicas de un hilo.

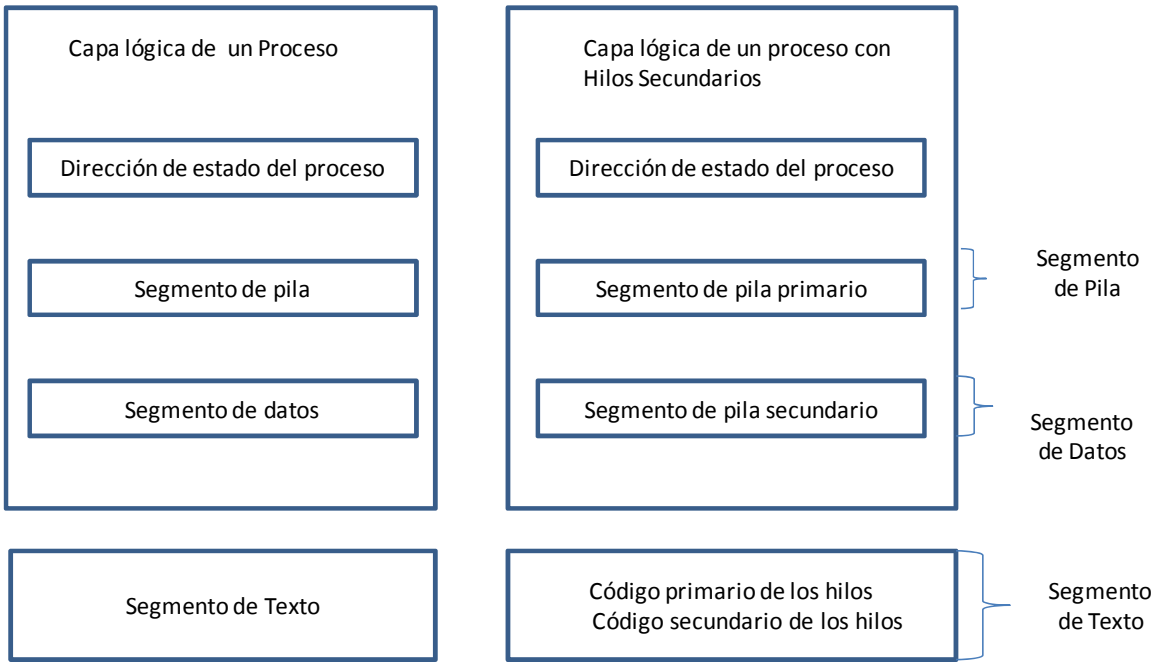

Figura 3.1 Capas lógicas de un proceso comparadas con las capas lógicas de un hilo.

Un proceso de Multithreading contiene más de un hilo, el cual ejecuta el código del programa de su propio proceso. La secuencia de instrucciones es llamada hilo de ejecución; es decir, si un proceso contiene un sólo hilo, llamado principal o primario, el proceso tiene sólo un flujo de control. Las instrucciones dentro de un proceso pueden ser divididas en subtareas que pueden ser ejecutadas asíncronamente. La secuencia de estas instrucciones puede ser representada como un hilo. El hilo puede ejecutar una sola función o un grupo de funciones. Cada contador de programas de un hilo puede esperar a la próxima instrucción para ser ejecutada. Cada hilo permite a las funciones de los procesos ejecutarse

independientemente sin considerar el flujo de control del proceso principal. Las aplicaciones de multithreading son aplicaciones que pueden beneficiarse con la ejecución asíncrona de subtareas.

3.2 Similitudes entre Hilos y Procesos

Tanto los hilos como los procesos se identifican con las siguientes características:
Tienen una identidad, un set de registros, un estado y una precedencia.
Tienen un bloque de información.
Comparten recursos con procesos padres.
Son entidades independientes después de ser creados.
Crean ejercicios de control sobre ellos.
Pueden cambiar sus atributos después de ser creados, y crear nuevos recursos.
No pueden acceder directamente a los recursos de otros procesos o hilos no relacionados.

3.3 Diferencias entre Hilos y Procesos

- Los hilos y procesos, al igual que similitudes, también presentan características que marcan claramente sus diferencias; las cuales son:
- Los procesos tienen una dirección de espacio; los hilos no.
- Procesos padres e hijos deben usar mecanismos de comunicación en interprocesos; los hilos de mismos procesos se comunican leyendo y escribiendo datos en las variables de los procesos.
- Procesos hijos no ejercen control de ninguna especie con otros procesos hijos. Los hilos de un proceso son considerados pares y ejercen control con otros hilos del mismo proceso.
- Los procesos hijo no pueden ejercer control sobre procesos padre. Algunos hilo de un proceso pueden ejercer control sobre el hilo principal e, incluso, con los procesos principales.

3.4 Ventajas de usar Multithreading

En la Tabla 3.1 se muestran las razones por las cuales es benéfico utilizar Multithreading.

Tabla 3.1 Ventajas de usar Multithreading

Beneficio	Razón
Mejorar el grado de acción de las aplicaciones	Cualquier programa en el cual muchas actividades no son dependientes, unas con respecto a las otras puede ser rediseñado de tal forma que cada actividad se defina como un hilo.
Utilizar los multiprocesadores más eficientes	Típicamente, las aplicaciones que expresan requerimientos de concurrencia con hilos no toman en cuenta el número de procesadores disponibles. El funcionamiento de la aplicación mejora con procesadores adicionales. Los algoritmos numéricos y las aplicaciones con un alto grado de paralelismo, tales como las multiplicaciones de matrices, pueden ejecutarse más rápidamente cuando son implementadas con hilos sobre un multi-procesador.
Mejorar la estructura del programa	Muchos programas son más eficientes estructurados como unidades de ejecución múltiples independientes o semi-independientes, en lugar de un simple hilo. Los programas Multithreading pueden ser más adaptativos a variaciones en las demandas del usuario que los programas con hilos simples.
Utilizar menos recursos del sistema	Los programas que usan dos o más procesadores que acceden datos comunes a través de memoria compartida están aplicando más de un hilo de control. Sin embargo, cada proceso tiene un espacio completo de direcciones y un estado del Sistema Operativo. El costo de crear y mantener esta gran cantidad de información de estado hace a cada proceso más caro que un hilo tanto en tiempo como en espacio. Adicionalmente, la separación inherente entre procesos puede requerir un esfuerzo mayor para el programador para comunicar los hilos entre procesos diferentes, o sincronizar sus acciones.

3.5 Desventajas de usar Multithreading

Los hilos necesitan sincronizar accesos concurrentes a la memoria; es por eso que un hilo C puede leer los datos depositados en la memoria por un hilo A antes de que un hilo B lo sobrescriba. Consecuentemente, la sincronización es necesaria para prevenir que ningún hilo sobrescriba los valores antes de que el dato sea usado, es decir, cuando un proceso de escritura toma lugar, uno de lectura debería ocurrir también.

Si una tares es dividida en hilos, un sólo hilo puede generar datos erróneos que afecten a los demás hilos. Los datos erróneos pueden afectar todos los datos del proceso si los hilos usan los datos que se encuentran en proceso e incluso, los mismos hilos pueden crear error de datos que afecten todo el espacio de memoria de los hilos. Comparando, los procesos

pueden proteger sus recursos de accesos indiscriminados por otros procesos, mientras que los hilos comparten recursos con todos los otros hilos dentro del proceso.

3.6 Tipos de Hilos

Como lo manifiesta Steve Kleiman [Kleiman 1996] pueden haber diferentes tipos de hilos, además define como los hilos pueden aproximar el trabajo que están realizando. El tipo de thread a utilizar nos permite controlar el flujo del programa y acceder a datos compartidos en hilos de ejecución concurrente.
Los principales tipos de hilos son descritos a continuación:

- Mutex locks: Permite sólo a un hilo en un tiempo ejecutar una sección de código o acceder a un dato específico.
- Variables de condición: Bloque a los hilos hasta que una condición en particular es verdadera.
- Semáforos: Son contadores que típicamente coordinan el acceso a los recursos. El contador es el límite de cuantos hilos pueden tener acceso a un recurso controlado por un semáforo. Cuando el contador se alcanza, el semáforo se bloquea.

3.7 Información importante acerca de Hilos

La información referente a un hilo es almacenada en estructuras de datos y retornada por funciones suministradas por el sistema operativo. Alguna información acerca de los hilos es contenida en una estructura llamada *bloque de información de un hilo,* la cual es creada al mismo tiempo que es creado el hilo. La estructura puede contener información tal como los puntos de cabeza de la manipulación de excepciones de los hilos, la base y el tamaño de la pila de hilos, la identidad del hilo y la precedencia de llamada del hilo.

Información de un hilo y Procesadores Múltiples

Los atributos de objeto de un hilo pueden incluir información del procesador. En un sistema que consta de más de un procesador, un hilo puede ser asignado a un procesador específico o a cierto rango de procesadores. En sistemas multiprocesadores simétricos o asimétricos, el sistema suministra una función que regresa la afinidad del procesador para un hilo específico.

3.8 Creación de un Hilo

El hilo principal de un proceso es creado automáticamente por el sistema operativo, mientras que los hilos secundarios pueden ser creados haciendo una llamada a la función "creación de un hilo". Cuando un nuevo hilo es creado, un nuevo flujo de control es creado a través del código de programa. El desarrollo del programa puede suministrar 2 formas de crear un hilo: una llamada a la función provista por la librería hilo o llamar a una función provista por el sistema operativo.

Cuando un hilo es creado, las direcciones de la función ejecuta el hilo si es necesario. Cuando un hilo es creado el sistema regresa la identidad del hilo y almacena en el parámetro creado para este propósito.

Cada hilo mantiene una pila que es automáticamente almacenada en la memoria del proceso. El tamaño de la pila debe ser lo suficientemente larga para todas las llamadas a funciones que el hilo puede realizar. Cuando un nuevo hilo es creado, el contexto para ese hilo es creado también. Cuando un hilo es creado por primera vez, el estado del hilo es dado por una bandera de creación, la cual puede sostener al hilo para ser creado en un estado suspendido.

La precedencia permite al despachador saber cual hilo está listo para la próxima ejecución, y cual hilo debería ser detenido porque un hilo de mayor precedencia se encuentra disponible. Es así que, la precedencia consiste en una clase de precedencia y un nivel de precedencia.

3.9 Terminación de un Hilo

Un hilo es terminado después de que la ejecución se ha completado. También puede ser forzado a terminar por otro hilo fuera del proceso. Forzar a un hilo a terminar significa que el hilo debería terminar antes de que la ejecución haya sido completada. Si un hilo no responde o no se ejecuta propiamente, puede ser necesario detener el hilo.

3.10 Pila de Hilo

Cada hilo tiene su propia pila. El tamaño de la pila es fijado en el momento de ser creado el hilo. La pila del hilo es alojada en el segmento de pila de la dirección de espacio del proceso. Si el creador del hilo no especifica el tamaño de la pila del hilo, el sistema asigna un tamaño por default; dependiendo estos de el número máximo de hilos, el tamaño permitido de la dirección de espacio del proceso y el espacio usado por los recursos del sistema. La pila debe ser lo suficientemente grande para algunas funciones llamadas por el hilo, para códigos externos al proceso y para el almacenamiento de variables locales.

Si un hilo intenta acceder espacio más allá del tamaño de su pila, el sistema tiene que idear varias formas de prevenir a un hilo de corromper la memoria del proceso. Una área de peligro, página o zona es designada fuera de la memoria de la pila cundo se notifica la sistema que ha sido violado.

3.11 Control de un Hilo

Aunque los hilos pueden crear otros hilos, no pueden tener una relación padre-hijo en el proceso. Los hilos tienen igual acceso a los recursos padres, lo cual permite comunicarse unos a otros fácilmente sin crear mecanismos de comunicación especiales.

Así como los hilos pueden crear nuevos hilos, también pueden eliminarlos. Esto puede hacerse debido a que pertenecen al mismo proceso y el hilo que está siendo ejecutado tiene la identidad del hilo en proceso.

Existe un término que se conoce como sección crítica del hilo, la cual es otro tipo de mecanismo de control. La forma de que un hilo entre a esta sección critica es accediendo a la memoria compartida con todos los hilos que se están desarrollando dentro del mismo proceso.

3.12 Precedencias de los Hilos

La manera de dar precedencia a los hilos es almacenando las tareas que requieren inmediata ejecución o respuesta del sistema. El esquema de precedencia es usado para determinar cual hilo usa el procesador y por cuánto tiempo; ya que el tiempo del CPU no es dividido equitativamente entre los hilos. Cada hilo tiene una precedencia y el hilo con la mayor precedencia es ejecutado antes de los de menor precedencia.

3.12.1 Precedencia de Clases y de Niveles

La precedencia de un hilo consiste en una precedencia de clase y una de nivel. La precedencia de clase es determinada por la precedencia de clase del proceso dentro del cual esta; mientras que la precedencia de nivel de un nuevo hilo es heredada del hilo que lo creo. Cada precedencia de clases tiene un rango de niveles. En la plataforma Win32 la mas alta precedencia de clases tiene niveles del 16 al 31 y las demás clases del o al 15; mientras que en OS/2 existen 32 niveles. Así, la precedencia del hilo es llamada precedencia base. El cambio el nivel de precedencia

3.12.2 Normas de Precedencia

Se debe tener cuidado cuando se asignan precedencias y hilos. El sistema se asegura de que en un proceso específico los hilos se ejecuten de acuerdo al de mayor precedencia. Los hilos que controlan la interface pueden ser afectados drásticamente causando que el teclado, el mouse y la pantalla permanezcan inactivos. La precedencia puede ser más alta que la del sistema, previniéndose para responder a cambios críticos en el sistema.

3.13 Estados de los Hilos

Un hilo está enlistado para ser ejecutado en el procesador. Cuando un hilo está ejecutando otro hilo de mayor precedencia, se convierte en un hilo que está listo para ser ejecutado. Si el hilo actualmente activo se encuentra en otro proceso, entonces el contexto del proceso elige entre los procesos que el hilo puede ser ejecutado. Cada hilo se ejecuta independientemente y compite por el tiempo del procesador. En la Figura 3.2 se muestra el diagrama de estado para un hilo y sus transiciones.

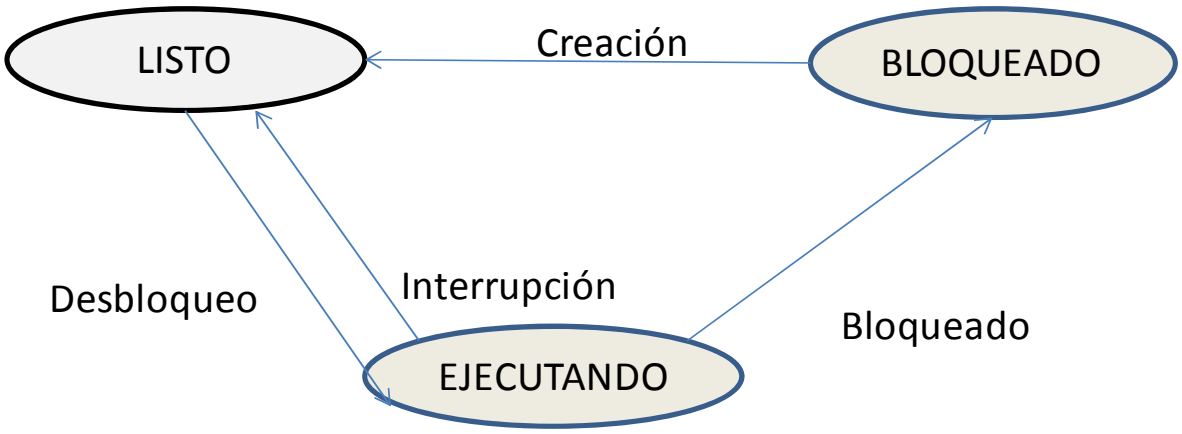

Figura 3.2 Diagrama de estado para un hilo y sus transiciones.

3.14 Aplicaciones actuales de MT

En el 2006 Intel lanzó su primer procesador de doble núcleo y cuatro núcleos de la nueva micro arquitectura Core en menos de seis meses. Esto fue un logro impresionante para la historia de los semiconductores. En la Figura 3.3 se muestra tal procesador

Figura 3.3 Procesador Intel Core2 Duo

La mayoría de servidores y sistemas operativos ya están diseñados para tomar ventaja de los procesadores multi-núcleo ("multi-core") porque están programados para separar diferentes procesos por una metodología que se llama "multithreading". Una tarea es básicamente cualquier programa realizado en una computadora, pero una tarea con hilos de ejecución de multithreading podrá realizar dos o más programas a la misma vez. ¿Por qué es esto tan importante? Al separar tareas entre diferentes hilos de ejecución, la computadora puede hacer más a la misma vez.

Esto aumenta la productividad y permite conseguir que más trabajo sea terminado en menos tiempo, mientras que consume menos energía eléctrica. Ahora esto no es nada fácil de hacer porque creando una aplicación con hilos de ejecución es bastante difícil. Requiere crear diferente secciones de códigos que corren a diferente tiempo y controlar como el procesador procesa datos e instrucciones de la aplicación.

Hay una abundancia de servidores y sistemas operativos que ya están tomando ventaja de multithreading con procesadores de multi-núcleo. El desafió para Intel y otras compañías semiconductores es conseguir más aplicaciones que aprovechen de la tecnología multi-core y que programen aplicaciones multithreaded. Si más software está programado para tomar ventaja de esta nueva tecnología el usuario verá los beneficios que provee Intel con su nueva tecnología.

La técnica de procesamiento en paralelo basada en Multithreading es una herramienta poderosa en el momento de la programación de cualquier sistema mediante computadora. Para nuestro caso se utiliza esta herramienta que proporciona el lenguaje de programación C++ y se explotan todas las características antes mencionadas de dicha técnica.

Capítulo 4

Esquema paralelo propuesto

4.1 Introducción

Para poder paralelizar el programa de la solución de los flujos de potencia, debemos tomar en cuenta que el método a desarrollar es el método de Newton-Raphson, así como que sólo una parte del método pude ser paralelizable. Por eso es que el esquema paralelo propuesto consiste en calcular de manera simultánea las filas de la matriz Jacobiana; esto puede realizarse debido a que existe independencia en el cálculo de los elementos de esta matriz.

Es también de advertirse, que en el momento de paralelizar un código dentro de un programa de computadora, lo único que se puede paralelizar a nivel de instrucciones son los ciclos que existen dentro del programa; pero teniendo el cuidado de que sean ciclos que no tengan dependencia directa de otra instrucción. Es decir, el siguiente código es paralelizable, ya que las instrucciones dentro de el dependen solamente de un parámetro, y el ciclo se realiza de manera secuencial:

```
for (i=0; i<10; i++)
        {
        ……..
        }
```

En la Figura 4.1 se puede apreciar fácilmente la manera en la cual va a trabajar el programa paralelizado de los flujos de potencia monofásicos.

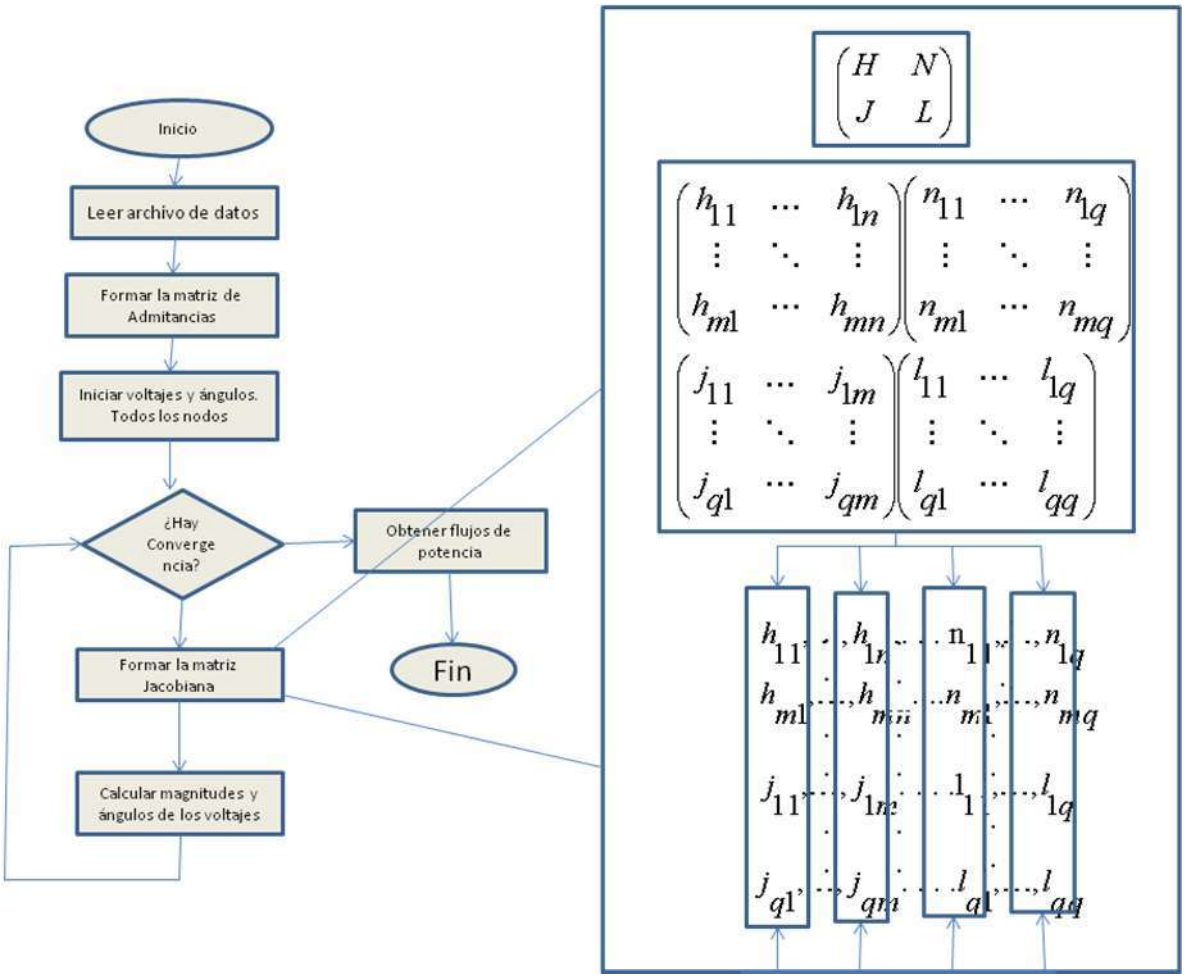

Figura 4.1 Diagrama de flujo del programa paralelizado

En el siguiente ejemplo, es lo contario ya que las instrucciones dependen directamente de la instrucción anterior, lo que lo hace un ciclo no paralelizable.

```
for (i=0; i <i+1; i++)
{
 …..
}
```

Como ya se dijo, el esquema a utilizarse consiste en paralelizar el problema que resuelve el Jacobiano, ya que de lo contrario, el procesador no puede hacer alguna instrucción hasta que no haya terminado la instrucción que esta ejecutando.

4.2 Modo de ejecución del programa de computadora (nivel usuario)

La ejecución del programa en si se basa en 2 procesos:

- La parte del programa que extrae los datos del sistema a resolver y que lo hace leyendo un archivo de texto.
- La otra parte que hace todos los cálculos numéricos.

Por ejemplo, para el sistema de prueba IEEE de 14 nodos tenemos que el código del archivo de texto donde se encuentran los datos se muestran en la Tabla 4.2; y tiene la siguiente estructura:

La primera fila nos indica los números de nodos, líneas de transmisión, transformadores y compensadores que existen en la red. Las siguientes filas (5 para este caso) indican el nodo de envío, nodo de recepción, resistencia, reactancia y el valor de tap para el transformador. La siguiente parte (15 filas), indican el nodo de envío, nodo de recepción, resistencia, reactancia, conductancia y susceptancia de las líneas de transmisión existentes en la red. La siguiente línea a su vez, indica el número de nodo, conductancia y susceptancia de donde se encuentra el (o los, dependiendo del caso) compensador de corriente. En el resto del archivo de datos contiene el número de nodo, tipo de nodo, magnitud de voltaje en el nodo, ángulo, potencia real generada, potencia reactiva generada, potencia real consumida y potencia reactiva consumida de todos los nodos existentes en la red. La clasificación que utiliza, de acuerdo al tipo de nodo, se muestra en la Tabla 4.1:

Tabla 4.1 Número de nodos y su tipo

Número de Nodo	Tipo de Nodo
1	Slack
2	PV
3	PQ

Tabla 4.2. Datos para el sistema de prueba de 14 nodos

14	15	5	1				
4	7	0.00000	0.20912	0.978			
4	9	0.00000	0.55618	0.969			
5	6	0.00000	0.25202	0.932			
7	8	0.00000	0.17615	1.0			
7	9	0.00000	0.11001	1.0			
1	2	0.01938	0.05917	0.0	0.0528		
1	5	0.05403	0.22304	0.0	0.0492		
2	3	0.04699	0.19797	0.0	0.0438		
2	4	0.05811	0.17632	0.0	0.0374		
2	5	0.05695	0.17388	0.0	0.0340		
3	4	0.06701	0.17103	0.0	0.0346		
4	5	0.01335	0.04211	0.0	0.0128		
6	11	0.09498	0.19890	0.0	0.0000		
6	12	0.12291	0.25581	0.0	0.0000		
6	13	0.06615	0.13027	0.0	0.0000		
9	10	0.03181	0.08450	0.0	0.0000		
9	14	0.12711	0.27038	0.0	0.0000		
10	11	0.08205	0.19207	0.0	0.0000		
12	13	0.22092	0.19988	0.0	0.0000		
13	14	0.17093	0.34802	0.0	0.0000		
9	0	0.19					
1	1	1.06	0.0	0.0	0.0	0.0	0.0
2	2	1.45	0.0	40.0	0.0	21.7	12.7
3	2	1.010	0.0	0.0	0.0	94.2	19.3
4	3	1.0	0.0	0.0	0.0	47.8	-3,9
5	3	1.0	0.0	0.0	0.0	4.6	1.6
6	2	1.070	0.0	0.0	0.0	11.2	7.5
7	3	1.0	0.0	0.0	0.0	0.0	0.0
8	2	1.090	0.0	0.0	0.0	0.0	0.0
9	3	1.0	0.0	0.0	0.0	29.5	16.6
10	3	1.1	0.0	0.0	0.0	9.0	5.8
11	3	1.2	0.0	0.0	0.0	3.5	1.8
12	3	1.3	0.0	0.0	0.0	6.1	1.6
13	3	1.4	0.0	0.0	0.0	13.5	5.8
14	3	1.5	0.0	0.0	0.0	14.9	5.0

4.3 Modo de ejecución del programa de computadora (nivel de instrucciones)

En la Figura 4.2, se muestra como es que normalmente trabaja un programa no paralelizado al momento de resolver el Jacobiano.

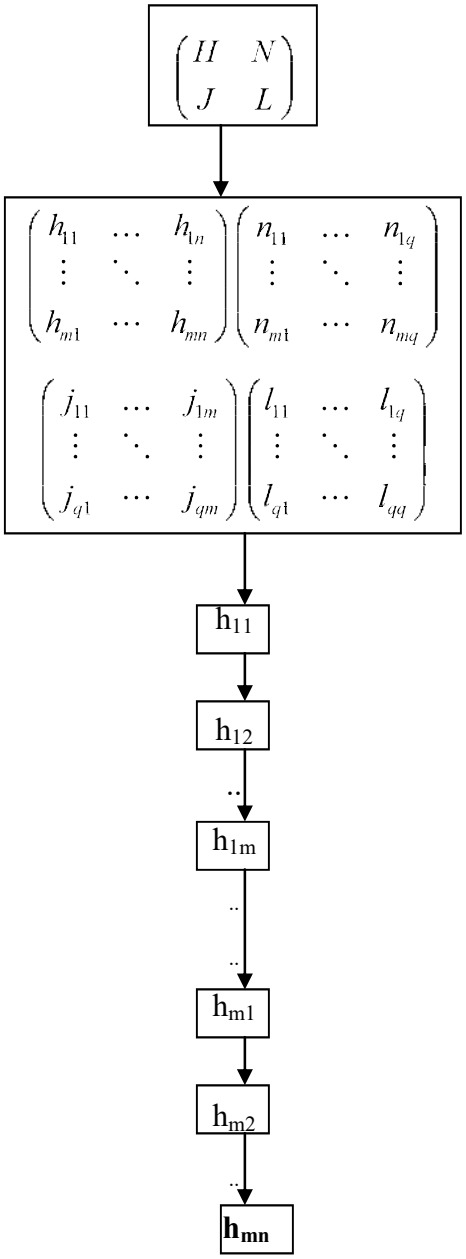

Figura 4.2 Esquema de solución de el Jacobiano utilizando programación secuencial

Es decir, tiene que hacer todo el proceso del cálculo del Jacobiano secuencialmente y celda por celda. Sin embargo, a continuación se muestra como se lleva a cabo este mismo proceso utilizando un esquema de paralelización.

La Figura 4.1 nos muestra también el esquema de cálculo simultáneo de los elementos de la matriz Jacobiana propuesto. De la figura puede apreciarse que el número de filas de la matriz Jacobiana es igual a 2PQ + PV, se reparte entre los elementos de proceso; dado que esta división no es necesariamente exacta, a cada elemento se le asigna el entero de la división entre el número de filas y el número de elementos de proceso; el remanente se reparte por de uno en uno entre los primeros elementos de proceso.

El elemento de proceso principal, en base al número de elementos de procesos disponibles, determina el número de filas que serán calculadas por cada elemento de proceso. El elemento de proceso principal envía a cada uno de los hilos la fila inicial y final. En base a esto se comienza a calcular los elementos de la matriz Jacobiana, los cuales se almacenan mediante un conjunto de listas simplemente enlazadas. Dado que un recorrido sobre la lista enlazada que representa el primer renglón es independiente al recorrido sobre la lista enlazada que almacena el segundo renglón y así sucesivamente, las listas se van llenando de manera simultánea, tal y como se muestra en la Figura 4.1.

Una vez que han sido calculados los elementos de la matriz Jacobiana se procede a llevar a cabo el proceso de solución, el cual se basa en un proceso de ordenamiento, bifactorizacion y sustitución. Una vez que finaliza este proceso se procede a actualizar las magnitudes y ángulos de los voltajes en los nodos PQ y PV; además de la actualización de la magnitud de los voltajes en los nodos PQ. Finalmente, se procede a verificar si se ha llegado a la convergencia; si es este el caso se procede a determinar los flujos de potencia de todas las líneas que conforman el sistema; de lo contrario, se repite el proceso hasta que se llegue a la convergencia o, en su defecto, hasta que se rebase de un número máximo de iteraciones especificadas.

Cabe hacer mención que el proceso de ordenamiento se realiza en la primera iteración del algoritmo. El ordenamiento puede volverse a realizar solamente si alguno de los nodos PV viola sus límites de generación y se convierte en un nodo de PQ, lo cual provoca que la topología de la matriz Jacobiana cambie.

Se propone trabajar la paralelizacion mediante la programación del método de Newton-Raphson en coordenadas polares y que se basará en leer un archivo de datos que contenga la información acerca del sistema que se pretende solucionar, y otra que corresponde a la solución del método propuesto; dentro de la cual existe la paralelización del programa consistente en paralelizar solamente la solución del Jacobiano.

Capítulo 5

Casos de estudio

5.1 Introducción

Para finalizar con este trabajo, es necesario poner en práctica todo lo asentado en los capítulos anteriores para comprobar que se cumplieron los objetivos planteados al inicio del trabajo de investigación, y que fueron:

- Aplicar técnicas de procesamiento en paralelo al problema de flujos de potencia monofásicos.
- Realizar estudios de flujos de potencia monofásicos a cualquier sistema de potencia, sea cual sea su número de nodos.
- Desarrollar una plataforma para poder resolver otros tipos de problemas de los sistemas de potencia.

5.2 Métodos de prueba

Es por eso que, para cumplir con los objetivos, se eligieron los sistema de prueba IEEE de 14, 30, 57 y 118 nodos con el objetivo claro de mostrar la aplicación del problema paralelo propuesto. En todos los casos de estudio presentados en este capítulo, se utilizó un criterio de convergencia de 1×10^{10} en *p.u.* para los desajustes en la potencia activa.

5.3 Equipos de prueba

Las características de los equipos utilizados al momento de realizar las pruebas son mostrados en la siguiente Tabla 5.1:

Tabla 5.1 Equipos utilizados para las pruebas

No. EQUIPO	PROCESADOR	CAPACIDAD PROC.	CARACTERISTICAS DE MULTITHREADING	CAPACIDAD DE MEMORIA	MARCA
1	Intel Xeon	3,06 GHz	2 procesadores con 2 núcleos	0,512 MB	
2	Intel Xeon	2,0 GHz	1 procesador con 4 núcleos	6,14 MB	
3	Intel Core2 Duo	2,4 GHz	1 procesador con 2 núcleos	1 MB	Macintosh

Que son equipos existentes en la Biblioteca de la División de Posgrado de la Facultad de Ingeniería Eléctrica de UMSNH. La Figura 5.1 muestra físicamente el primer equipo utilizado para las pruebas, la Figura 5.2 y la 5.3 el tercer equipo.

Figura 5.1 Equipo 1 utilizado en las pruebas

Figura 5.2 Equipo 2 utilizado en las pruebas

Figura 5.3 Equipo 3 utilizado en las pruebas

5.4 Especificaciones necesarias para hacer las pruebas

Algunas consideraciones que se tomaron en cuenta al momento de hacer las pruebas fueron para darle forma a los requerimientos planteados y para poder cumplir cabalmente con los objetivos planteados. Entre ellas se encuentran:

- No tomar mucho en cuenta los resultados en cuanto a los parámetros eléctricos, sino más bien a los tiempos de ejecución.
- Hacer el mismo número de pruebas en ambos equipos (10 pruebas consecutivas).
- Seleccionar el valor mínimo de tiempo que dio cada prueba y obtener la eficiencia del programa. La eficiencia se obtiene como a continuación se describe:

η = tiempo obtenido con n núcleos (s) / tiempo obtenido con un núcleo (s) (5.1)

5.5 Pruebas realizadas

Para el sistema de prueba de 14 nodos, como se puede ver en la Tabla 5.2, no se obtienen grandes cambios en los tiempos de ejecución; por lo que obtenemos para este caso una eficiencia relativa de 1. La Tabla 5.2 muestra los tiempos obtenidos con cada equipo utilizado.

Tabla 5.2 Datos de salida para el sistema de 14 nodos en los diferentes equipos

PRUEBA IEEE 14 NODOS

	NÚMERO DE HILOS				
	1	2	3	4	
No. PRUEBA	t(s)	t(s)	t(s)	t(s)	No. Equipo
1	0,007	0,007	0,007	0,007	1
2	0,007	0,007	0,007	0,007	1
3	0,007	0,007	0,007	0,009	1
7	0,007	0,007	0,007	0,007	1
3	0,008	0,005	0,005	0,005	2
4	0,005	0,005	0,005	0,005	2
8	0,005	0,005	0,005	0,005	2
9	0,009	0,005	0,005	0,009	2
1	0,005	0,004	0,004	0,004	3
2	0,005	0,004	0,004	0,004	3
9	0,004	0,004	0,004	0,004	3
10	0,004	0,004	0,004	0,005	3

En el de 30 nodos sólo un equipo mostro variación en la eficiencia (Equipo 2), mientras que los demás no muestran gran variación en los tiempos de ejecución, tal y como lo muestra la Tabla 5.3.

Tabla 5.3 Datos de salida para el sistema de 30 nodos en los diferentes equipos

PRUEBA IEEE 30 NODOS

	NÚMERO DE HILOS				
	1	2	3	4	
No. PRUEBA	t(s)	t(s)	t(s)	t(s)	No. Equipo
1	0,055	0,056	0,081	0,055	1
2	0,066	0,077	0,055	0,056	1
7	0,063	0,056	0,064	0,057	1
8	0,055	0,06	0,056	0,067	1
3	0,025	0,025	0,025	0,025	2
4	0,029	0,025	0,025	0,025	2
9	0,031	0,025	0,025	0,025	2
10	0,025	0,025	0,032	0,03	2
2	0,018	0,018	0,019	0,018	3
3	0,018	0,018	0,018	0,02	3
9	0,018	0,018	0,017	0,018	3
10	0,018	0,018	0,018	0,019	3

Ya para los sistemas de 57 nodos es más perceptible el resultado en cuanto a la eficiencia del programa que resuelve el problema de los flujos de potencia usando multithreading. Así que, en las Tablas 5.4, 5.5 y 5.6, respectivamente se muestran los valores de salida que se registraron en cuanto al tiempo en el momento de efectuar las pruebas al sistema de 57 nodos en los equipos 1, 2 y 3 utilizados.

Tabla 5.4 Datos de salida para el sistema de 57 nodos en el equipo 1

PRUEBA IEEE 57 NODOS

	NÚMERO DE HILOS			
	1	2	3	4
No. PRUEBA	t(s)	T(s)	t(s)	t(s)
1	0,229	0,213	0,247	0,18
2	0,197	0,21	0,2	0,205
3	0,215	0,179	0,21	0,177
4	0,224	0,179	0,181	0,199
5	0,204	0,205	0,213	0,179
6	0,193	0,256	0,179	0,202
7	0,198	0,176	0,178	0,225
8	0,229	0,199	0,184	0,225
9	0,219	0,196	0,228	0,179
10	0,206	0,176	0,178	0,174
Maximo	0,229	0,256	0,247	0,225
Minimo	0,193	0,176	0,178	0,174
Promedio	0,2114	0,1989	0,1998	0,1945
Desv. Std.	0,01357449	0,02465067	0,02423863	0,01962
Eficiencia	1	1,09659091	1,08426966	1,1091954

Tabla 5.5 Datos de salida para el sistema de 57 nodos en el equipo 2

PRUEBA IEEE 57 NODOS

	NÚMERO DE HILOS			
	1	2	3	4
No. PRUEBA	t(s)	t(s)	t(s)	t(s)
1	0,081	0,082	0,079	0,082
2	0,091	0,082	0,081	0,079
3	0,082	0,08	0,086	0,082
4	0,082	0,077	0,082	0,079
5	0,082	0,086	0,082	0,078
6	0,081	0,081	0,079	0,082
7	0,082	0,088	0,083	0,077
8	0,082	0,083	0,075	0,082
9	0,083	0,081	0,082	0,077
10	0,081	0,084	0,078	0,082
Maximo	0,091	0,088	0,086	0,082
Minimo	0,081	0,077	0,075	0,077
Promedio	0,0827	0,0824	0,0807	0,08
Desv. Std.	0,00298329	0,00309839	0,00305687	0,00221108
Eficiencia	1	1,05194805	1,08	1,05194805

Tabla 5.6 Datos de salida para el sistema de 57 nodos en el equipo 3

PRUEBA IEEE 57 NODOS

	NÚMERO DE HILOS			
	1	2	3	4
No. PRUEBA	t(s)	t(s)	t(s)	t(s)
1	0,182	0,18	0,17	0,18
2	0,175	0,18	0,174	0,177
3	0,179	0,178	0,171	0,177
4	0,178	0,177	0,173	0,168
5	0,172	0,171	0,172	0,171
6	0,178	0,173	0,171	0,174
7	0,177	0,172	0,171	0,181
8	0,175	0,175	0,159	0,175
9	0,175	0,178	0,161	0,174
10	0,177	0,178	0,17	0,171
Maximo	0,182	0,18	0,174	0,181
Minimo	0,172	0,171	0,161	0,168
Promedio	0,1768	0,1762	0,1692	0,1748
Desv. Std.	0,00274064	0,00325918	0,00502881	0,0041042
Eficiencia	1	1,00584795	1,06832298	1,02380952

En el sistema de 118 nodos es aún más perceptible el beneficio de usar Multithreading en la solución del problema de flujos de potencia. Así que en las Tablas 5.7, 5.8 y 5.9 se muestran los resultados de salida con los equipos 1, 2 y 3, respectivamente.

Tabla 5.7 Datos de salida para el sistema de 118 nodos en el equipo 1

PRUEBA IEEE 118 NODOS

	NÚMERO DE HILOS			
	1	2	3	4
No. PRUEBA	t(s)	T(s)	t(s)	t(s)
1	0,529	0,445	0,469	0,447
2	0,539	0,511	0,473	0,487
3	0,533	0,448	0,493	0,481
4	0,537	0,449	0,483	0,488
5	0,538	0,498	0,529	0,463
6	0,538	0,491	0,463	0,464
7	0,562	0,478	0,472	0,485
8	0,528	0,489	0,467	0,485
9	0,543	0,481	0,489	0,473
10	0,537	0,475	0,506	0,488
Maximo	0,562	0,511	0,529	0,488
Minimo	0,528	0,445	0,463	0,447
Promedio	0,5384	0,4765	0,4844	0,4761
Desv. Std.	0,009477459	0,022618822	0,020640844	0,013979747
Eficiencia	1	1,186516854	1,140388769	1,181208054

Tabla 5.8 Datos de salida para el sistema de 118 nodos en el equipo 2

PRUEBA IEEE 118 NODOS

	NÚMERO DE HILOS			
	1	2	3	4
No. PRUEBA	t(s)	t(s)	t(s)	t(s)
1	0,249	0,216	0,217	0,215
2	0,249	0,22	0,216	0,217
3	0,255	0,224	0,216	0,216
4	0,249	0,218	0,219	0,217
5	0,259	0,222	0,232	0,222
6	0,249	0,211	0,216	0,222
7	0,249	0,229	0,224	0,238
8	0,252	0,224	0,218	0,213
9	0,249	0,212	0,218	0,215
10	0,253	0,223	0,227	0,216
Maximo	0,259	0,229	0,232	0,238
Minimo	0,249	0,211	0,216	0,213
Promedio	0,2513	0,2199	0,2203	0,2191
Desv. Std.	0,0034657	0,00568526	0,00551866	0,00724875
Eficiencia	1	1,18009479	1,15277778	1,16901408

Tabla 5.9 Datos de salida para el sistema de 118 nodos en el equipo 3

PRUEBA IEEE 118 NODOS

	NÚMERO DE HILOS			
	1	2	3	4
No. PRUEBA	t(s)	t(s)	t(s)	t(s)
1	0,929	0,918	0,861	0,94
2	0,922	0,92	0,851	0,933
3	0,941	0,918	0,855	0,935
4	0,925	0,914	0,859	0,929
5	0,952	0,916	0,844	0,922
6	0,934	0,911	0,942	0,839
7	0,945	0,922	0,927	0,942
8	0,946	0,937	0,93	0,929
9	0,923	0,891	0,938	0,839
10	0,944	0,855	0,932	0,832
Maximo	0,952	0,937	0,942	0,942
Minimo	0,922	0,855	0,844	0,832
Promedio	0,9361	0,9102	0,8939	0,904
Desv. Std.	0,010877602	0,022458851	0,042495621	0,046844898
Eficiencia	1	1,078362573	1,092417062	1,108173077

Finalmente, se presenta como resumen la Tabla 5.10 donde se señalan las eficiencias relativas finales de acuerdo al sistema y al número de equipo utilizados.

Tabla 5.10 Eficiencias finales por sistema de prueba y número de equipo

	NÚMERO DE HILOS			
	1	2	3	4
No. Equipo	H	H	η	η
PRUEBA IEEE 14 NODOS				
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
PRUEBA IEEE 30 NODOS				
1	1	1	1	1,058
2	1	1	1	1
3	1	1	1	1
PRUEBA IEEE 57 NODOS				
1	1	1,10	1,08	1,11
2	1	1,05	1,08	1,05
3	1	1,01	1,07	1,02
PRUEBA IEEE 118 NODOS				
1	1	1,19	1,14	1,18
2	1	1,18	1,15	1,17
3	1	1,08	1,09	1,11

Con el fin de mostrar los sistemas de prueba utilizados y su solución, en Fig. 5.4 se muestra el esquema completo del sistema de 14 nodos, señalando en las Tablas 5.10, 5.11 y 5.12 los voltajes en cada nodo, los flujos de potencia en cada nodo y las pérdidas de potencia en líneas de transmisión, respectivamente.

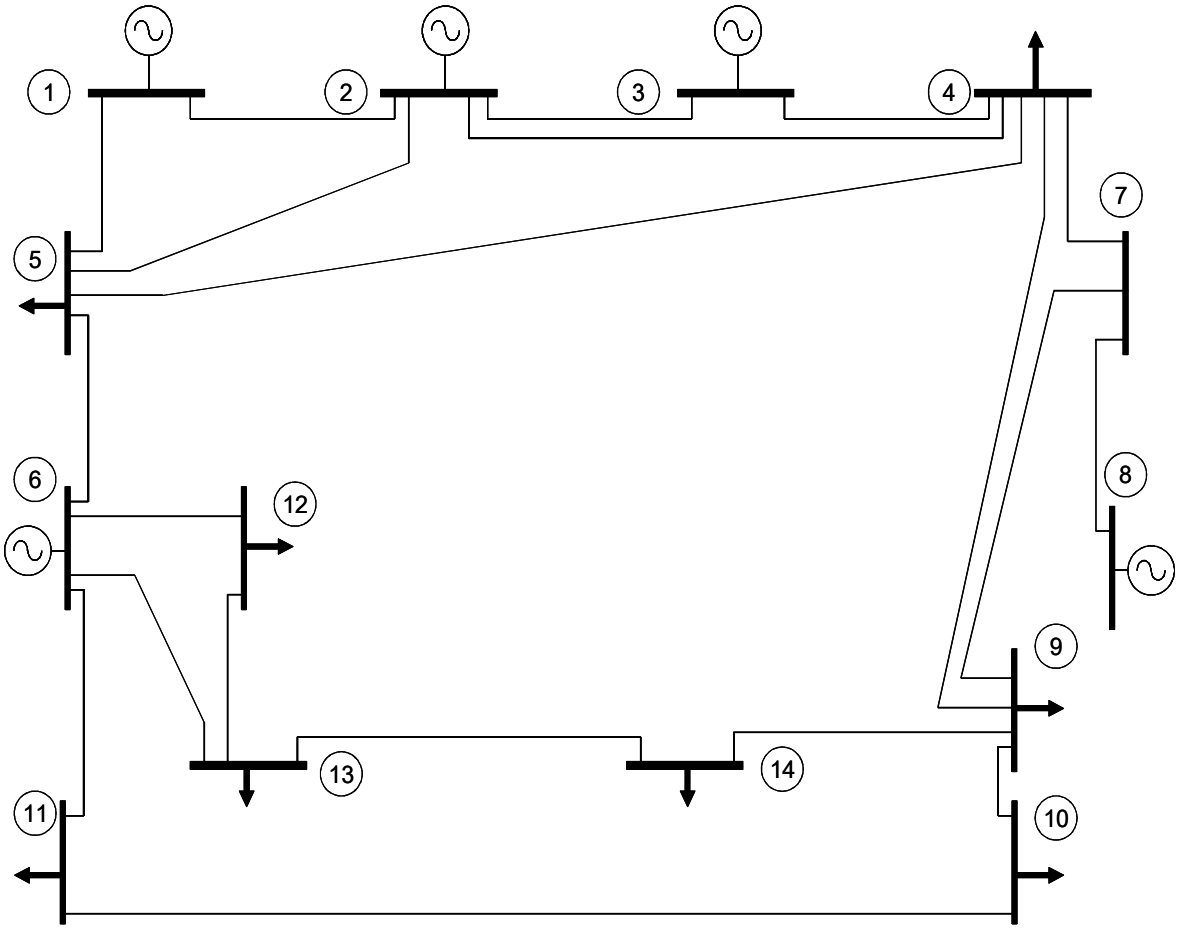

Figura 5.4 Diagrama del sistema de prueba IEEE 14 nodos

Tabla 5.11 Voltajes de salida para el sistema de 14 nodos

No DE NODO	TIPO DE NODO	VOLTAJE
1	SLACK	1.060000 0.000000
2	PV	1.045000 -4.980938
3	PV	1.010000 -12.717940
4	PQ	1.018623 -10.324199
5	PQ	1.020264 -8.782556
6	PV	1.070000 -14.222621
7	PQ	1.061951 -13.368218
8	PV	1.090000 -13.368218
9	PQ	1.056346 -14.946567
10	PQ	1.051328 -15.104290
11	PQ	1.057082 -14.795231
12	PQ	1.055220 -15.077390
13	PQ	1.050443 -15.158905
14	PQ	1.035795 -16.038898

Tabla 5.12 Flujos para el sistema de 14 nodos

Del Nodo_ al Nodo	P Q	Del Nodo_ al Nodo	P Q
Nodo 1 a Nodo 2	156.833294 -20.392725	Nodo 2 a Nodo 1	-152.538447 27.656279
Nodo 1 a Nodo 5	75.552618 3.503795	Nodo 5 a Nodo 1	-72.788853 2.580476
Nodo 2 a Nodo 3	73.188103 3.565048	Nodo 3 a Nodo 2	-70.867926 1.584360
Nodo 2 a Nodo 4	56.138129 -2.287607	Nodo 4 a Nodo 2	-54.461092 3.393761
Nodo 2 a Nodo 5	41.512214 0.762713	Nodo 5 a Nodo 2	-40.609940 -1.633923
Nodo 3 a Nodo 4	-23.332071 2.809235	Nodo 4 a Nodo 3	23.703419 -5.421251
Nodo 4 a Nodo 5	-61.219162 15.669502	Nodo 5 a Nodo 4	61.735689 -15.370477
Nodo 6 a Nodo 11	7.341203 3.472221	Nodo 11 a Nodo 6	-7.286491 -3.357649
Nodo 6 a Nodo 12	7.781910 2.492228	Nodo 12 a Nodo 6	-7.71023 -2.343042
Nodo 6 a Nodo 13	17.739993 7.170901	Nodo 13 a Nodo 6	-17.528451 -6.754309
Nodo 9 a Nodo 10	5.238928 4.305991	Nodo 10 a Nodo 9	-5.225819 -4.271166
Nodo 9 a Nodo 14	9.437908 3.665771	Nodo 14 a Nodo 9	-9.321136 -3.417379
Nodo 10 a Nodo 11	-3.774182 -1.528834	Nodo 11 a Nodo 10	3.786491 1.557649
Nodo 12 a Nodo 13	1.610230 0.743042	Nodo 13 a Nodo 12	-1.60399 -0.737397
Nodo 13 a Nodo 14	5.632441 1.691705	Nodo 14 a Nodo 13	-5.578864 -1.582621

Tabla 5.13 Perdidas de Potencia en las líneas de transmisión

PERDIDAS EN LINEAS DEL NODO_ AL NODO_	P	Q
1_2	4.294847	7.263554
1_5	2.763765	6.084270
2_3	2.320177	5.149408
2_4	1.677036	1.106154
2_5	0.902274	-0.871211
3_4	0.371348	-2.612016
4_5	0.516527	0.299025
6_11	0.054711	0.114572
6_12	0.07168	0.149186
6_13	0.211542	0.416592
9_10	0.013110	0.034825
9_14	0.116773	0.248392
10_11	0.012309	0.028815
12_13	0.00624	0.005645
12_14	0.053577	0.109085

A continuación se muestran las graficas que se obtuvieron utilizando los 3 equipos diferentes, probándolos con los sistemas de 14, 30, 57 y 118 nodos, en base a la Tabla 5.9.

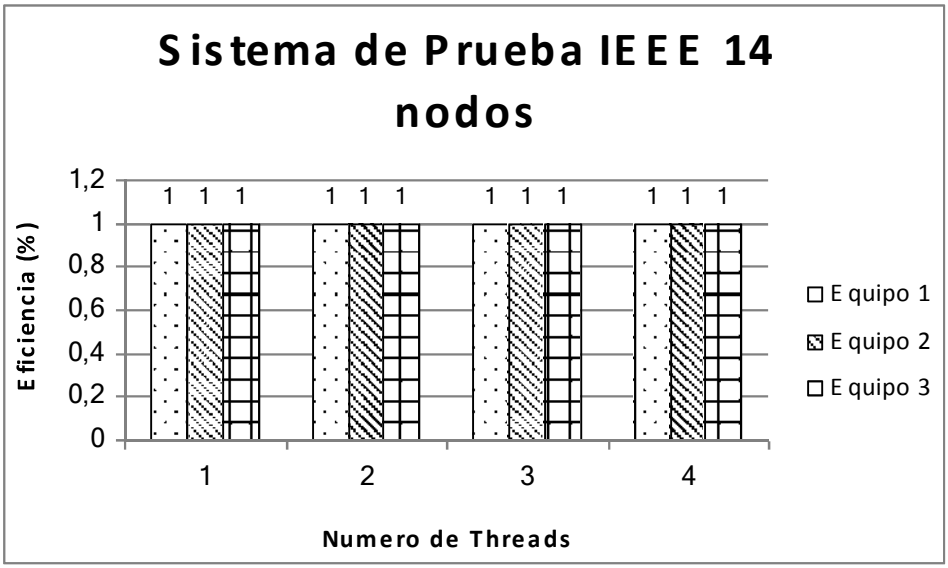

Figura 5.5 Eficiencia del programa que calcula el sistema IEEE 14 nodos

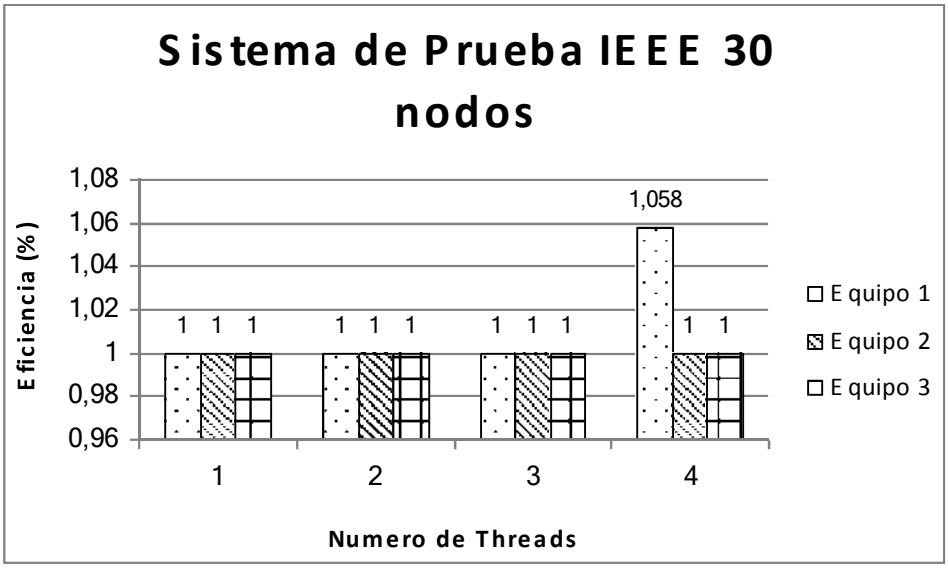

Figura 5.6 Eficiencia del programa que calcula el sistema IEEE 30 nodos

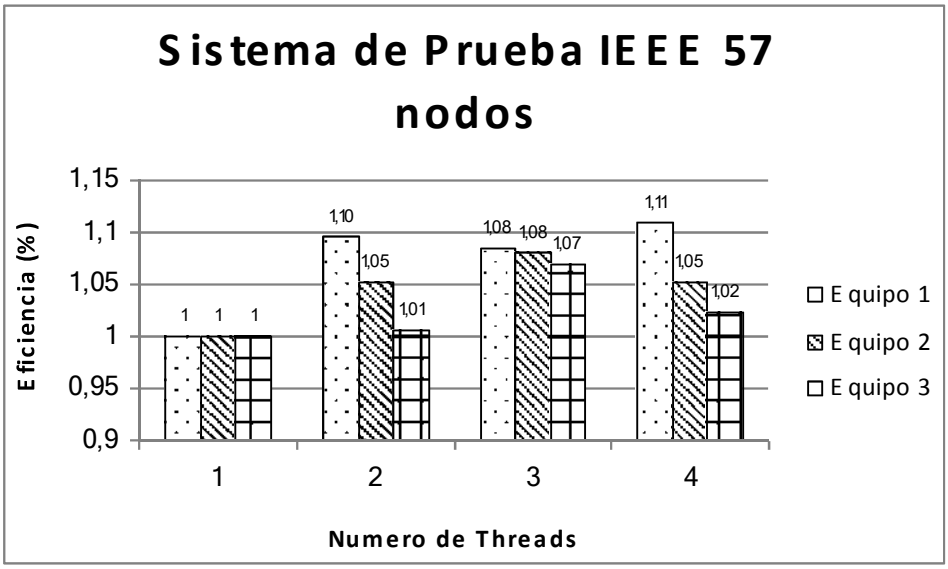

Figura 5.7 Eficiencia del programa que calcula el sistema IEEE 57 nodos

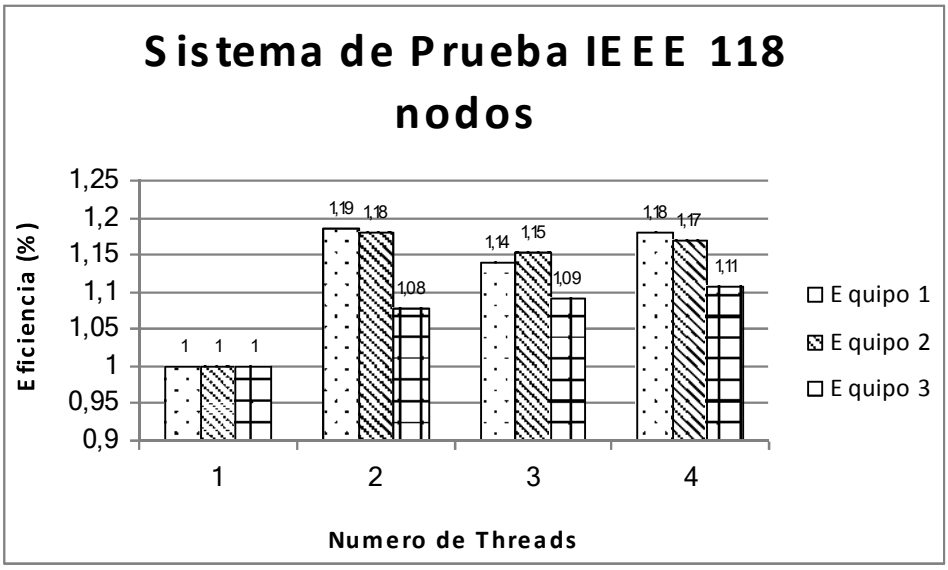

Figura 5.8 Graficas de eficiencia para el sistema de 118 nodos

5.6 Conclusiones de los resultados obtenidos

Como se puede observar en las pruebas realizadas, en los sistemas de 14 y 30 nodos no alcanza a ser perceptible el beneficio de usar procesamiento en paralelo debido a que los tiempos de ejecución son muy pequeños (ya que la solución del método para estos casos no implica uso excesivo de la memoria). Ya en los sistemas de 57 y 118 nodos se alcanza eficiencias relativas más altas; alcanzando una eficiencia de 1.19 % en el sistema de 118 nodos con el equipo 1. Se observa también, que el equipo que muestra mayor eficiencia es el equipo número 1 y el que muestra menos es el equipo número 3; pero todo esto se debe a las características de los equipos mostradas en el apartado 5.3.

Capítulo 6

Conclusiones y Recomendaciones

6.1 Conclusiones

A lo largo de todo este trabajo de tesis se plantearon 2 problemas: el estudio de los flujos de potencia y la programación de métodos numéricos utilizando procesamiento en paralelo. Se plantearon los objetivos generales y se dio cumplimiento a ello, ya que:

Se aplicaron técnicas de procesamiento en paralelo al estudio de los flujos de potencia monofásicos, que consistieron en usar multithreading al momento de resolver el Jacobiano para el método de Newton-Raphson en coordenadas polares.

La técnica propuesta se aplicó en los sistemas de prueba IEEE de 14, 30, 57 y 118 nodos.

Se ha desarrollado una plataforma para realizar estudios de flujos de potencia que incorporan técnicas de procesamiento en paralelo, la cual mejora la eficiencia en tiempo de los estudios.

6.2 Trabajos futuros

Se deja el antecedente de este trabajo como una plataforma para quien desee hacer estudios de los flujos de potencia como métodos de solución, o ya sea también hacer investigaciones acerca de los Sistemas de Potencia monofásicos de cualquier tipo.
Dentro de los estudios que se pudieran hacer a partir de este Trabajo son:

- Aplicar técnicas de procesamiento en paralelo al estudio de flujos de potencia trifásicos.
- Incorporar otros tipos de dispositivos en los análisis de los flujos de potencia.

Referencias

[Tinney W. 1967]

Tinney W., “Power Flow solution by Newton`s Method”, *IEEE Transactions on Power Systems*, págs. 1449-1460, Noviembre 1967.

[Dommel W. 1968]

Dommel W., “Optimal power flow solutions”, *IEEE Transactions on Power apparatus and Systems*, págs. 1866-1876, Octubre 1968.

[Keyhani A. 1989]

Keyhani A., “Evaluation of Power Flow Techniques for Personal Computers”, *IEEE Transactions on Power Systems*, págs. 817-826, Mayo 1989.

[Lan Jin 1994]

Lan Jin, “exploring the architectures and algorithms close relationship”, *IEEE Parallel procesing*, págs. 17-20 , Diciembre 1994.

[Dixit S. 2006]

Dixit S., “Power Flow analysis using fuzzy logic”, *IEEE, Junio* 1967.

[Chapra 2003]

Steven C. Chapra, Metodos Numericos para Ingenieros, Mexico: Mc Graw_Hill, 2003.

[Kleiman 1996]

Steve Kleiman, Programming with Threads, USA: Prentice Hall, 1996.

[Arrillaga 1990]

J. Arrillaga, Computer Análisis of Power Systems, England: Wiley, 1990.

[Hughes 1997]

Cameron Hughes, Objected_Oriented Multithreading Using C++, USA: John Wiley & Sons, 1997.

[IEEE 2008]

IEEE. Página principal. Estados Unidos. 17 de mayo de 2008. http://www.ieee.org

Apéndice A.

CODIGO FUENTE DEL PROGRAMA PARALELIZADO

```
void calcula_jacobiano()
{
	int i,j,x,y;
	int cont_x, cont_y;
	double teta;
	complex double aux1;

	cont_re = 0;
	cont_itag = 0;
	cont_lnxt = 0;
	ref,cont;
	lf=0;



	/*H*/
	cont_y = 1;
	cont_x = 1;
	for (i=0; i<(pv+pq); i++)
	{
		x = nodo[i+1].numero;		// Correccion del nodo slack
		cont_y = 1;
		for (j=0; j<(pv+pq); j++)
		{
			y = nodo[j+1].numero;
			teta = argu(nodo[i+1].voltaje)-argu(nodo[j+1].voltaje);
			aux1 = busca(x,y);
			if (x==y)
				aux = -nodo[i+1].q-imag(aux1)*pow(abso(nodo[i+1].voltaje),2.0);
			else
				aux = abso(nodo[i+1].voltaje)*abso(nodo[j+1].voltaje)*(real(aux1)*sin(teta)-
imag(aux1)*cos(teta));
			if (aux != 0)
			{
				noze[cont_x] = noze[cont_x] + 1;
				if (cont_x != cont_y)
				{
					re[cont_re +1] = aux;
					cont_re = cont_re + 1;
					itag[cont_itag+1] = cont_y;
					cont_itag = cont_itag + 1;
					cont_lnxt = cont_lnxt + 1;
					lnxt[cont_lnxt] = cont_lnxt + 1;
				}
			}
			if (cont_x==cont_y)
				de[cont_x] = aux;
			cont_y = cont_y + 1;
		}
		for (j=0; j<pq; j++)
		{
			y = nodo[j+1+pv].numero;
			teta = argu(nodo[i+1].voltaje)-argu(nodo[j+1+pv].voltaje);
			aux1 = busca(x,y);
			if (x==y)
				aux = nodo[i+1].p+real(aux1)*pow(abso(nodo[i+1].voltaje),2.0);
			else
				aux =
abso(nodo[i+1].voltaje)*abso(nodo[j+1+pv].voltaje)*(real(aux1)*cos(teta)+imag(aux1)*sin(teta));
			if (aux != 0)
			{
				noze[cont_x] = noze[cont_x] + 1;
				if (cont_x != cont_y)
```

```
                                        {
                                                re[cont_re +1] = aux;
                                                cont_re = cont_re + 1;
                                                itag[cont_itag+1] = cont_y;
                                                cont_itag = cont_itag + 1;
                                                cont_lnxt = cont_lnxt + 1;
                                                lnxt[cont_lnxt] = cont_lnxt + 1;
                                        }
                                }
                                if (cont_x==cont_y)
                                        de[cont_x] = aux;
                                cont_y = cont_y + 1;

                        }
                        lnxt[cont_lnxt] = 0;
                        nseq[i+1] = i+1;
                        cont_x = cont_x + 1;



                }
                /*J*/

                for (i=0; i<pq; i++)
                {
                        cont_y = 1;
                        x = nodo[i+1+pv].numero;
                        for (j=0; j<(pq+pv); j++)
                        {
                                y = nodo[j+1].numero;
                                teta = argu(nodo[i+1+pv].voltaje)-argu(nodo[j+1].voltaje);
                                aux1 = busca(x,y);
                                if (x==y)
                                        aux = nodo[i+1+pv].p-real(aux1)*pow(abso(nodo[i+1+pv].voltaje),2.0);
                                else
                                        aux = -
abso(nodo[i+1+pv].voltaje)*abso(nodo[j+1].voltaje)*(real(aux1)*cos(teta)+imag(aux1)*sin(teta));
                                if (aux != 0)
                                {
                                        noze[cont_x] = noze[cont_x] + 1;
                                        if (cont_x != cont_y)
                                        {
                                                re[cont_re +1] = aux;
                                                cont_re = cont_re + 1;
                                                itag[cont_itag+1] = cont_y;
                                                cont_itag = cont_itag + 1;
                                                cont_lnxt = cont_lnxt + 1;
                                                lnxt[cont_lnxt] = cont_lnxt + 1;
                                        }
                                }
                                if (cont_x==cont_y)
                                        de[cont_x] = aux;
                                cont_y = cont_y + 1;
                        }


                        for (j=0; j<pq; j++)
                        {
                                y = nodo[j+1+pv].numero;
                                teta = argu(nodo[i+1+pv].voltaje)-argu(nodo[j+1+pv].voltaje);
                                aux1 = busca(x,y);
                                if (x==y)
                                        aux = nodo[i+1+pv].q-imag(aux1)*pow(abso(nodo[i+1+pv].voltaje),2.0);
                                else
                                        aux = abso(nodo[i+1+pv].voltaje)*abso(nodo[j+1+pv].voltaje)*(real(aux1)*sin(teta)-
imag(aux1)*cos(teta));
                                if (aux != 0)
                                {
                                        noze[cont_x] = noze[cont_x] + 1;
                                        if (cont_x != cont_y)
```

```
                                        {
                                                re[cont_re +1] = aux;
                                                cont_re = cont_re + 1;
                                                itag[cont_itag+1] = cont_y;
                                                cont_itag = cont_itag + 1;
                                                cont_lnxt = cont_lnxt + 1;
                                                lnxt[cont_lnxt] = cont_lnxt + 1;
                                        }
                                }
                                if (cont_x==cont_y)
                                        de[cont_x] = aux;
                                cont_y = cont_y + 1;
                        }
                        lnxt[cont_lnxt] = 0;
                        cont_x = cont_x + 1;
                }
                lcol[1] = 1;
                for (i=1; i<=(2*pq+pv); i++)
                        lcol[i+1] = lcol[i]+noze[i]-1;
                cont = 0;
                for (i=0; i<(2*pq+pv)+1;i++)
                {
                        ref = 0;
                        do
                        {
                                if (itag[ref] == i)
                                {
                                        ce[cont] = re[ref];
                                        cont ++;
                                }
                                ref++;
                        }while(itag[ref] != 0);
                }
                lf = lcol[2*pq+pv]+noze[2*pq+pv];
                for (i=1; i<=(2*pq+pv); i++)
                        nseq[i] = i;
}
```