Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica
División de Estudios de Posgrado

Paralelización de Estimación de Estado usando Cómputo en Paralelo Basado en Unidades de Procesamiento Gráfico

T E S I S

que para obtener el grado de

Maestra en Ciencias en Ingeniería Eléctrica

presenta

Ing. Cindy Viridiana Zabala Oseguera

Director de Tesis:
Dr. Antonio Ramos Paz

Co-Director de Tesis:
Dr. Claudio Rubén Fuerte Esquivel

Agosto 2020

PARALELIZACIÓN DE ESTIMACIÓN DE ESTADO USANDO CÓMPUTO EN PARALELO BASADO EN UNIDADES DE PROCESAMIENTO GRÁFICO

Los Miembros del Jurado de Examen de Grado aprueban la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de ***Cindy Viridiana Zabala Oseguera.***

Dr. J Aurelio Medina Rios
Presidente del Jurado

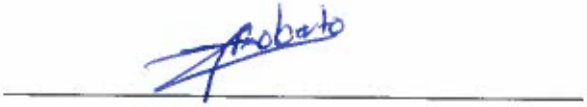
Dr. Antonio Ramos Paz
Director de Tesis

Dr. Claudio Rubén Fuerte Esquivel
Co-director

Dr. Roberto Tapia Sánchez
Vocal

Dr. Boris Adrián Alcaide Moreno
Revisor Externo (CENACE)

Dr. Roberto Tapia Sánchez
Jefe de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas)



UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Agosto 2020

Dedicatoria

A ti papá.

A ti Jorge, por estar siempre.

Agradecimientos

A mi mamá y hermanos de los cuales siempre he recibido apoyo incondicional en cada etapa de mi vida.

Al Dr. Antonio Ramos Paz, mi Director de Tesis, por su paciencia infinita y apoyo durante esta etapa.

A mi co-Director de Tesis, el Dr. Claudio R. Fuerte Esquivel, por su apoyo y observaciones.

A mis sinodales, por haber dedicado tiempo y esfuerzo en la revisión de este trabajo.

A mis compañeros de posgrado: Yeraldin, Juanma, Luisa, Carlitos, Iván, Marco, Víctor, Llamas y Garibo por haberme brindado su amistad y en muchas ocasiones su ayuda.

Al Departamento de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica por darme la oportunidad de cursar mis estudios de maestría.

A CONACyT por el apoyo financiero para cursar mis estudios de posgrado. También se agradece el apoyo recibido del Fondo de Sustentabilidad Energética SENER-CONACyT a través del proyecto 246949.

Resumen

El monitoreo continuo del sistema eléctrico tiene como objetivo garantizar el estado operativo y seguro del mismo. Las condiciones operativas de un sistema eléctrico se pueden conocer a través de su modelo de red y la magnitud y ángulo de voltaje en cada nodo del sistema. La estimación de estado es una herramienta matemática mediante la cual se obtiene la magnitud de voltaje y el ángulo en cada nodo de un sistema eléctrico a partir de mediciones existentes y la topología del sistema. Los resultados obtenidos de la estimación de estado se utilizan en otros estudios, de suma importancia para el monitoreo y operación del sistema eléctrico, tales como: flujos óptimos de potencia, análisis de contingencias, estudios de seguridad, entre otros.

Tradicionalmente, las mediciones en un sistema eléctrico, se obtenían únicamente a través del Sistema de Supervisión y Control de Adquisición de Datos (SCADA, por sus siglas en inglés); posteriormente se comenzaron a utilizar las Unidades de Medición Fasorial (PMUs, por sus siglas en inglés). El costo asociado al uso de las PMUs ha llevado a buscar nuevas formas para incluir las mediciones PMU en el proceso de estimación de estado. El estimador de estado utilizado en este trabajo consta de dos etapas: en una primera etapa, se utiliza la formulación convencional por mínimos cuadrados ponderados y en la segunda, se lleva a cabo nuevamente el proceso de estimación con las mediciones PMU y el vector de estado estimado resultado de la primera etapa. El tiempo de ejecución del algoritmo de estimación de estado de dos etapas es optimizado a través del uso de Cómputo en Paralelo Basado en Unidades de Procesamiento Gráfico (GPUs por sus siglas en inglés).

Palabras Clave: Sistemas eléctricos, Estimación de estado de dos etapas, Procesamiento en paralelo, Mínimos cuadrados ponderados, Unidades de medición fasorial

Abstract

The continuous monitoring of the electrical system aims to ensure its safe and operational state. The operating conditions of an electrical system can be known through its network model and the magnitude of voltage and angle at each node of the system. State estimation is a mathematical tool by which the voltage magnitude and angle at each node of an electrical system are obtained from existing measurements and the system topology. The results obtained from the state estimation are used in other studies, of great importance for the monitoring and operation of the electrical system, such as: optimal power flows, contingency analysis, safety studies, among others.

Traditionally, measurements in a power system were only obtained through the Data Acquisition Supervision and Control System (SCADA); Later, Phasor Measurement Units (PMUs) began to be used. The cost associated with the use of PMUs has led to the search for new ways to include PMU measurements in the state estimation process. The state estimator presented in this work consists of two stages: in a first stage, the conventional formulation by weighted least squares is used and finally in the second stage, the estimation process is carried out again with the PMU measurements and the vector of estimated state result of the first stage. The execution time of the two-stage state estimation algorithm is optimized through the use of Parallel Computing Based on Graphic Processing Units (GPUs).

Power Systems, Two-Stage State Estimation, Parallel Processing, Weighted Least Squares, Phasor Measurement Units

Contenido

Dedicatoria III

Agradecimientos IV

Resumen V

Abstract VI

Contenido VII

Lista de Figuras IX

Lista de Tablas X

Lista de Acrónimos y abreviaturas XI

1. Introducción 1
- 1.1. Antecedentes . . . 2
  - 1.1.1. Antecedentes de la estimación de estado en los sistemas eléctricos . . . 2
  - 1.1.2. Antecedentes del procesamiento en paralelo y su utilización en los sistemas eléctricos . . . 5
- 1.2. Objetivos . . . 8
  - 1.2.1. Objetivo general . . . 8
  - 1.2.2. Objetivos particulares . . . 8
- 1.3. Justificación . . . 8
- 1.4. Metodología . . . 10
- 1.5. Aportación de la tesis . . . 10
- 1.6. Contenido de la tesis . . . 10

2. Estimación de estado 12
- 2.1. Introducción . . . 12
- 2.2. Estimación convencional . . . 13
  - 2.2.1. Formulación matemática . . . 14
  - 2.2.2. Elementos del algoritmo de estimación de estado convencional . . . 17
- 2.3. Estimación de estado lineal . . . 20
  - 2.3.1. Formulación matemática . . . 22
  - 2.3.2. Formulación alternativa . . . 25
- 2.4. Estimación de estado de dos etapas . . . 26
  - 2.4.1. Formulación matemática . . . 26

2.4.2. Ejemplo de aplicación: sistema de 5 nodos 28
2.5. Conclusiones 33

3. Procesamiento en paralelo **34**
3.1. Introducción 34
3.2. Unidades de procesamiento Gráfico (GPU) 34
3.3. Modelo de programación en CUDA 35
3.4. Aceleración y eficiencia 38
3.5. Ley de Amhdal 38
3.6. Ley de Gustafson-Barsis 39
3.7. Parte secuencial de un programa paralelo 40
3.8. Conclusiones 42

4. Propuesta de paralelización del algoritmo de estimación de dos etapas **43**
4.1. Introducción 43
4.2. Esquema de paralelización del método de estimación de dos etapas 44
4.3. Conclusiones 47

5. Casos de Estudio **48**
5.1. Introducción 48
5.2. Casos de estudio considerando mediciones sin errores 49
5.2.1. Sistema IEEE 14 nodos 49
5.2.2. Sistema IEEE 30 nodos 50
5.2.3. Sistema IEEE 118 nodos 51
5.3. Tiempo de ejecución 52
5.4. Conclusiones 54

6. Conclusiones y trabajos futuros **55**
6.1. Conclusiones generales 55
6.2. Recomendaciones para trabajos futuros 56

Apéndice A. Archivos de entrada al estimador **57**

Apéndice B. Código ejecutado en la GPU **59**

Referencias **62**

Lista de Figuras

2.1. Funciones del estimador de estado 13
2.2. Algoritmo de estimación convencional 17
2.3. Estructura matriz Jacobiana H 19
2.4. Modelo PI de una línea de transmisión 21
2.5. Distribución de las PMUs en el sistema de 14 nodos 23
2.6. Distribución de las mediciones convencionales en el sistema de 5 nodos. . . . 29

3.1. Arquitectura GeForce GTX 295 35
3.2. Modelo de programación CUDA 36
3.3. Hilos, bloques y mallas en CUDA 37
3.4. Ley de Amdahl 39
3.5. Ley de Gustafson-Barsis 41

4.1. Algoritmo de estimación de estado de dos etapas 43
4.2. Diagrama de flujo para la paralelización del método de estimación de dos etapas 46

5.1. Equipo utilizado para las pruebas de procesamiento en paralelo 48
5.2. GeForce GTX 660 OEM 49
5.3. Voltajes y ángulos del sistema IEEE 30 nodos 51
5.4. Magnitudes de voltajes y ángulos del sistema IEEE 118 nodos 52
5.5. Aceleración para cada caso de estudio 54

Lista de Tablas

2.1. Y_{bus} del sistema IEEE 14 nodos 24
2.2. Admitancias en derivación del sistema IEEE 14 nodos 24
2.3. Mediciones consideradas en la primera etapa 29
2.4. Mediciones PMU utilizadas en la segunda etapa 30
2.5. Resultados de la primera y segunda etapa del estimador para el sistema de 5 nodos 32
2.6. MSE de cada etapa para el sistema de 5 nodos 33

5.1. Características de la GPU GeForce GTX 660 OEM 49
5.2. Comparación entre los resultados obtenidos de flujos de potencia y los resultados obtenidos en cada etapa del estimador para el sistema IEEE de 14 nodos 50
5.3. MSE de cada etapa para sistema IEEE de 14 nodos 50
5.4. MSE de cada etapa para sistema IEEE de 30 nodos 51
5.5. MSE de cada etapa para sistema de 118 nodos 52
5.6. Resumen casos de estudio primera etapa 53
5.7. Resumen casos de estudio segunda etapa 53
5.8. Aceleración obtenida en cada caso de estudio 53

A.1. Formato del archivo de mediciones primera etapa 57
A.2. Formato del archivo de datos del sistema 58
A.3. Formato del archivo susceptancia 58
A.4. Formato del archivo de mediciones PMU 58

Lista de Acrónimos y abreviaturas

IEEE	Instituto de Ingenieros en Electricidad y Electrónica.
SCADA	Sistema de supervisión, control y adquisición de datos.
PMU	Unidad de medición fasorial.
CCE	Centro de control de energía.
MATLAB	Laboratorio de matrices.
GPU	Unidad de procesamiento gráfico.
z	Vector de mediciones.
MSE	Error medio cuadrático.
x	Vector de estados del sistema.
ε	Tolerancia a la convergencia.
h	Vector de ecuaciones no-lineales.
G	Matriz de ganancia del sistema.
H	Matriz Jacobiana.
J(x)	Función cuadrática de mínimos cuadrados.
Y	Admitancia.
CUDA	Arquitectura Unificada de Dispositivos de Cómputo.
E	Eficiencia.
ψ	Aceleración.

Capítulo 1

Introducción

En este trabajo de investigación el estimador utilizado pertenece a la categoría de estimación de estado en estado estable, en específico, la estimación de estado estática. El algoritmo de estimación estático procesa un conjunto de mediciones y da como resultado un vector de estado estimado compuesto por magnitudes y ángulos de voltaje nodal. Por esa razón, en este capítulo se revisan los trabajos de investigación realizados en años anteriores en la categoría de estimación de estado estática y algunas aplicaciones que han tenido las GPUs en el área de los sistemas eléctricos de potencia.

En [Gulati *et al.*, 2013] se presenta una clasificación de las categorías de la estimación de estado. Esta clasificación está integrada por tres categorías:

- **Estática:** se utilizan mediciones obtenidas en el instante k para obtener el ángulo y la magnitud de voltaje de cada nodo del sistema para el mismo instante.

- **De seguimiento:** este tipo de estimador surgió por la necesidad de hacer más eficientes los estimadores estáticos. Actualiza el valor de los estados durante un periodo de muestreo a partir del valor más reciente disponible.

- **Dinámico:** estima los estados dinámicos del sistema como los valores del ángulo del rotor y la velocidad de los generadores, además de la magnitud y ángulo de voltaje en cada nodo del sistema [Farzanehrafat y Watson, 2010].

1.1. Antecedentes

1.1.1. Antecedentes de la estimación de estado en los sistemas eléctricos

En 1965 ocurrió un incidente que revolucionó el análisis de los sistemas eléctricos. Este incidente que dejó a una parte de Estados Unidos de América sin energía eléctrica representó grandes pérdidas económicas para muchas empresas. A partir de ese año, las empresas suministradoras de energía eléctrica comenzaron a considerar la seguridad y calidad en el servicio, lo que dio origen al Sistema de Supervisión y Control de Adquisición de Datos (SCADA, por sus siglas en inglés). Se pensó que el problema estaría resuelto manteniendo en constante actualización la base de datos del SCADA, pero no fue así [Periñán y Expósito, 1999].

En 1968, Fred Schweppe afirmó que la actualización constante del SCADA no era suficiente para asegurar el funcionamiento correcto del sistema eléctrico. Su afirmación se basaba en que las mediciones después de un periodo de tiempo se eliminaban y estas mediciones contenían errores [Periñán y Expósito, 1999]. Las principales causas de la aparición de errores en las mediciones se debían a la conversión analógica a digital, fases desbalanceadas y el ruido presente en los canales de comunicación.

Debido al ruido presente en las mediciones, Schweppe propuso y trabajó en los primeros estimadores de estado estáticos. Schweppe definió un estimador de estado como: “un algoritmo de procesamiento de datos para convertir lecturas de medidores redundantes y otra información disponible en una estimación del estado de un sistema de energía eléctrica” [Wu, 1990]. La principal ventaja que tiene el estimador de estado es que permite realizar una detección e identificación de mediciones que contienen errores para que posteriormente al eliminarlas se obtenga la mejor estimación del estado de un sistema.

La formulación propuesta por Schweppe en el algoritmo de estimación de estado se basaba en el método de mínimos cuadrados ponderados y desde que fue propuesto se han reportado diversos trabajos utilizando este método. A pesar de ofrecer buenos resultados una desventaja es que, en sistemas de gran escala, el proceso de estimación se vuelve lento [Karimipour y Dinavahi, 2013].

Con el fin de disminuir el tiempo de cómputo, en [Dobakhshari *et al.*, 2019] se presenta un método no iterativo para la estimación de estado utilizando mediciones SCADA. El método se probó en redes de transmisión de alto voltaje europeas de 1341 y 9241 nodos, mostrando una ventaja en el tiempo de ejecución en comparación con los métodos de estimación de estado convencionales.

El siguiente avance importante en la operación y análisis de los sistemas eléctricos de potencia son las unidades de medición fasorial. En la década de los 80s del siglo pasado, Arun Phadke comenzó a construir los primeros prototipos de las unidades de medición fasorial (PMU) y en 1991 comenzó su comercialización con *Virginia Tech* [Lozano *et al.*, 2012].

Las PMUs miden fasores de voltaje, frecuencia, tasa de cambio de frecuencia y fasores de corriente [Ortiz *et al.*, 2016]. A partir de la presencia de las PMUs en los sistemas eléctricos, se tiene una nueva formulación de estimación de estado: "la estimación de estado lineal", donde el proceso de estimación deja de ser iterativo y los cálculos se reducen significativamente en comparación con la estimación de estado convencional. Aunque lo ideal sería una red eléctrica con mediciones, únicamente de PMU, aun se visualiza difícil debido a lo costoso de los equipos y lo complejo que resulta su ubicación en redes de gran escala [Chatterjee *et al.*, 2015]. Actualmente, debido a estas razones, la mayoría de los sistemas eléctricos de gran escala no puede depender únicamente de las mediciones de PMU para realizar estimación del estado, sino que deben considerar las mediciones SCADA y PMU como un conjunto para incluir las mediciones PMU en el estudio de estimación de estado.

En la literatura se han reportado diversos trabajos de estimadores de estado donde se incluyen las mediciones PMU. Existen dos formas de incluir las mediciones PMU en el proceso de estimación: la primera es a través de un solo estimador de estado que incluye tanto mediciones proporcionadas por el sistema SCADA como de las PMUs realizando modificaciones a la formulación convencional, en este esquema el modelo utilizado es no lineal y, por lo tanto, la solución se obtiene mediante un proceso iterativo. A este primer esquema, se le llama Estimación de Estado Híbrida (HSE, por sus siglas en inglés) [Sodhi *et al.*, 2010] [Baltensperger *et al.*, 2010]. Un segundo esquema es la estimación de estado lineal de post procesamiento. En este esquema se utiliza un estimador de dos etapas en el que no se modifica

el método convencional y el proceso de estimación se mejora realizando una segunda etapa [Huang *et al.*, 2012].

Utilizando el primer esquema de inclusión de las mediciones PMU en la estimación de estado, en [Rabha *et al.*, 2015] se describe una metodología de estimación de estado híbrida en coordenadas rectangulares utilizando el sistema IEEE de 14 nodos. En ese mismo año, en [Rendon *et al.*, 2015] se presenta un estimador híbrido que es probado en los sistemas IEEE de 14 y 118 nodos.

En [Zhou *et al.*, 2006] [Nuqui y Phadke, 2007] se presenta una formulación utilizando dos etapas. En estos trabajos de investigación, las mediciones SCADA y PMU se procesan en etapas diferentes utilizando en la primera etapa la formulación convencional y en la segunda etapa se procesan las mediciones PMU y el vector de estado estimado obtenido en la primera etapa.

En [Soni *et al.*, 2012] se presenta un estimador de estado que utiliza mediciones SCADA y PMU considerando la detección de mediciones erróneas en las mediciones SCADA. El algoritmo comienza realizando el proceso de estimación utilizando la formulación convencional con mediciones SCADA, una vez que se cumple el criterio de convergencia establecido, se realiza la detección de mediciones erróneas y, por último, los resultados obtenidos de la estimación de estado convencional y las mediciones PMU, se utilizan para realizar nuevamente el proceso de estimación.

En el mismo año, en [Tarali y Abur, 2012] se presenta una metodología para la detección de mediciones erróneas en el algoritmo de estimación de estado estática en dos etapas. En este trabajo se propone la detección de datos erróneos al finalizar cada etapa además se aborda la detección de mediciones erróneas en mediciones críticas.

Posteriormente, en [James y Bindu, 2015] se presenta una comparación entre el estimador convencional y el estimador de dos etapas. Ambos métodos fueron probados con el sistema IEEE de 14 nodos, el estimador de estado de dos etapas arrojó menores errores que el estimador convencional.

1.1.2. Antecedentes del procesamiento en paralelo y su utilización en los sistemas eléctricos

Muchos de los avances más significativos de la humanidad en ramas como: la ciencia, la medicina y el entretenimiento, tienen su como principal impulsor los desarrollos tecnológicos de la computación. El aumento del poder de cálculo de las computadoras ha permitido resolver problemas más complejos en menor tiempo; por ejemplo, en problemas particulares asociados con la obtención de modelos más precisos para entender el cambio climático y el análisis de grandes volúmenes de datos. En sistemas eléctricos el aumento de la potencia de cómputo ha permitido programar modelos mucho más detallados de tecnologías como: las turbinas eólicas, celdas solares, baterías entre otros [Pacheco, 2011].

En un principio se pensaba que la única alternativa para aumentar el rendimiento de las computadoras era aumentar la velocidad de la Unidad de Procesamiento Central (CPU). Cuando aparecen las primeras computadoras en la década de los 80s, la velocidad de las CPUs era en promedio de 1 Megahercio (MHz). Actualmente la velocidad de la CPU es 1000 veces más rápida que en sus inicios [Rivera y Vargas-Lombardo, 2012]. Debido a que seguir aumentando la velocidad de la CPU no es la única alternativa para mejorar el rendimiento de las computadoras, se comienzan a construir computadoras con más de un elemento de proceso.

Con la aparición de las computadoras multiprocesador, se necesitaba una rama de la computación dedicada a explotar los beneficios de ejecutar programas computacionales en estos equipos. El cómputo en paralelo se basa en dividir los procesos que integran un programa computacional de tal forma que algunos de estos procesos se puedan realizar de forma simultánea en los distintos elementos de proceso con los que cuenta el equipo que soporta este tipo de programación, con el fin de reducir el tiempo de ejecución un programa.

Con el fin de unificar el proceso de paralelización de un programa, en [Foster, 1995] este proceso se divide en cuatro etapas: partición, comunicación, aglomeración y asignación. En la primera etapa lo que se busca es subdividir el problema en el mayor número de subproblemas posibles; en la segunda etapa se definen los canales y la información que será enviada y recibida; en la etapa de aglomeración las tareas se agrupan con el fin de reducir el costo de comunicación y finalmente en la cuarta etapa, las tareas son asignadas a los distintos

procesadores, el objetivo de esta etapa es disminuir el tiempo de ejecución del programa.

Las computadoras paralelas se definen como un conjunto de procesadores que pueden trabajar conjuntamente para resolver un problema computacional. Las computadoras donde se puede realizar este tipo de cómputo se dividen en dos grandes grupos: multiprocesador y multicomputadora. Las computadoras multiprocesador están compuestas por más de un elemento de proceso, la comunicación entre los procesadores se realiza por medio de una memoria global compartida. Las multicomputadoras están formadas a partir de un número de computadoras que se conectan a través de una red y la comunicación se realiza mediante paso de mensajes.

La métrica más comúnmente utilizada para cuantificar el beneficio obtenido al utilizar más de un elemento de proceso en la ejecución de un programa es la aceleración. La aceleración es una magnitud adimensional que se calcula dividiendo el tiempo de ejecución del programa utilizando un elemento de proceso entre el tiempo de ejecución utilizando más de un elemento de proceso y representa el número de veces que el algoritmo paralelo es más rápido que su contraparte secuencial.

Las plataformas para el procesamiento en paralelo para computadoras multiprocesadores son: OpenMP y CUDA (Compute Unified Device Architecture) esta última diseñada por NVIDIA [Barlas, 2014], mientras que para las multicomputadoras las plataformas basadas en el paso de mensajes son: MPI (*Message Passing Interface*) y PVM (*Parallel Virtual Machine*)[Cotronis, 2001].

Aunque las computadoras multiprocesador fueron una excelente aportación para agilizar la ejecución de los programas, la atención se volcó en un nuevo tipo de procesador, el cual tuvo su primer impulsor en los videojuegos. En el año 2000 NVIDIA lanza la primer GPU, la cual tenía como característica principal una gran capacidad para realizar operaciones de punto flotante. En el 2003 lanza una plataforma para realizar cómputo en la GPU, pero es hasta el 2006 cuando finalmente lanza la plataforma CUDA (*Compute Unified Device Architecture*)[Acosta *et al.*, 2012].

La GPU es un coprocesador ubicado en la tarjeta gráfica de la computadora. En la GPU se llevan a cabo tareas como las operaciones matriciales necesarias para la visualización gráfica en la ejecución de videojuegos. Frente a la arquitectura de la CPU, la principal ventaja que resalta la GPU, es que tiene cientos de núcleos mientras que la CPU solo está formada por unos cuantos. Además, económicamente es menos costoso un equipo con GPU que una computadora con múltiples procesadores. La desventaja que quizás limita las aplicaciones del procesamiento en paralelo utilizando GPUs, y las enfoca a problemas con un tamaño considerable, es que la velocidad de cada núcleo es aproximadamente la tercera parte de la velocidad de la CPU.

Entre las aplicaciones en las que se han utilizado las GPUs en los sistemas eléctricos se encuentra la estimación de estado. En [Karimipour y Dinavahi, 2013] se presenta una propuesta de paralelización basada en GPUs del algoritmo de estimación convencional utilizando la factorización LU para resolver el sistema de ecuaciones linealizado en cada iteración. El algoritmo se prueba en sistemas de 39, 78, 156, 312, 624, 1248, 2496 y 4992 nodos. La ventaja en la aceleración se presenta a partir del sistema de 78 nodos. Para el sistema de 4992 nodos se logra una aceleración de 38. En el año 2017, en [Xia *et al.*, 2017] se utiliza la plataforma híbrida CPU-GPU para la paralelización del algoritmo de estimación de estado estático utilizando el método desacoplado rápido. En esta propuesta, las operaciones son divididas para ser ejecutadas en la CPU o en la GPU. Esta propuesta de paralelización utiliza la plataforma CUDA.

Las GPUs también han sido utilizadas en la categoría de estimación de estado de la calidad de la energía. En el año 2018, en [Cisneros-Magaña *et al.*, 2018] se presenta una metodología de estimación de estado de la calidad de la energía en el dominio del tiempo (PQSE, por sus siglas en inglés) utilizando el filtro paralelo de Kalman. En esta propuesta, se emplea el cómputo en paralelo basado en GPUs utilizando la plataforma CUDA y la librería cuBLAS.

Además de la estimación de estado, las GPUs han sido utilizadas en otros estudios como flujos de potencia [Li *et al.*, 2017] [Garcia, 2010] [Singh y Aruni, 2010], solución en estado estable y estabilidad transitoria. En [Jalili-Marandi y Dinavahi, 2009] presenta un estudio de estabilidad transitoria donde se realiza una comparación entre la ejecución utilizando CPU

y CPU-GPU. Se prueban en sistemas de 39, 78, 156, 312, 624, 1248 nodos. Utilizando la plataforma CPU-GPU se logra una aceleración de 345 con el sistema de 1248 nodos.

En [Magaña-Lemus *et al.*, 2015] se presenta una propuesta para determinar el estado estacionario periódico de redes eléctricas utilizando cómputo en paralelo basado en GPUs. En este trabajo, el cómputo en paralelo es utilizado para reducir el esfuerzo computacional dentro del proceso de identificación de la matriz de transición en la técnica de Newton empleada.

1.2. Objetivos

1.2.1. Objetivo general

Aplicar procesamiento en paralelo utilizando la plataforma CPU-GPU a un algoritmo de estimación de estado estático de dos etapas, donde las operaciones más costosas computacionalmente se lleven a cabo en la GPU.

1.2.2. Objetivos particulares

- Utilizar el procesamiento en paralelo basado en GPU en un algoritmo de estimación de estado donde las mediciones PMU se incluyen de una forma no invasiva.
- Optimizar el tiempo de ejecución del algoritmo de estimación de dos etapas.
- Diseñar una herramienta computacional para realizar estimación de estado estático eficientando el tiempo de cómputo.
- Realizar comparaciones entre el algoritmo programado secuencialmente y el algoritmo propuesto utilizando métricas como la aceleración.

1.3. Justificación

El crecimiento acelerado de los sistemas eléctricos ha derivado en más interconexiones, inclusión de dispositivos que permiten su monitoreo y nuevas formas de generación de energía para satisfacer las necesidades cada vez más exigentes del usuario, por lo que la operación de los sistemas eléctricos ha aumentado su complejidad.

El objetivo de la operación de un sistema eléctrico es mantenerlo en un estado seguro. Para lograrlo se realizan ciertas acciones que tienen que ver con la recolección de variables del sistema, tales como: magnitudes de voltaje nodales, flujos de potencia en las líneas y transformadores, el intercambio de energía y potencia entre los sistemas interconectados e información de las unidades generadoras. Esa información proviene de los dispositivos de medición ubicados en el sistema y es enviada a los Centros de Control de Energía (CCE) donde el operador realizará acciones pertinentes a partir de esa información.

Una herramienta importante para el monitoreo del sistema es la estimación de estado. La información utilizada por esta herramienta proviene del sistema SCADA. En consecuencia, la capacidad de responder de los CCEs ante cambios repentinos está condicionado al tiempo empleado en la actualización del sistema SCADA y el tiempo que tarda el proceso de estimación. Ambos procesos se realizan en el orden de segundos.

El proceso de estimación involucra una serie de operaciones matriciales que representan una parte importante del tiempo de ejecución. Las operaciones requeridas en un algoritmo de estimación son: producto matriz-vector, producto matriz-matriz, resolver sistemas linealizados de la forma $Ax = b$ y suma de vectores. Las operaciones que involucran vectores y matrices son generalmente costosas computacionalmente y, por lo tanto, lentas. Debido a esa razón, una forma fácil de reducir el tiempo de ejecución es acelerar estas operaciones. En consecuencia, es en estas operaciones donde se encuentra una oportunidad de disminuir el tiempo de ejecución del estimador de estado utilizando herramientas como el procesamiento en paralelo basado GPU.

Una GPU está conformada por muchos núcleos que trabajan a una velocidad de reloj baja. La velocidad de reloj podría parecer una desventaja, pero cuando una operación involucra una gran cantidad de datos, el tiempo en el que se realiza esa operación puede disminuir. Los sistemas eléctricos actuales tienen una gran cantidad de mediciones y datos debido al tamaño del propio sistema. El procesamiento de los datos y los cálculos del proceso de estimación representan una parte importante del tiempo de ejecución de un programa que realiza este estudio, por lo que el procesamiento en paralelo basado en GPUs puede ser una opción viable para eficientar el proceso de estimación de estado.

1.4. Metodología

El desarrollo de esta tesis se basa en la realización de las siguientes actividades:

- Revisión bibliográfica de los métodos de estimación convencional, lineal y de dos etapas, así como los trabajos relacionados.
- Revisión bibliográfica de temas y conceptos relacionados con el procesamiento en paralelo basado en GPU, así como plataformas utilizadas para realizar procesamiento en paralelo.
- Revisión de los antecedentes de la programación en paralelo en los sistemas eléctricos, en específico en la estimación de estado.
- Implementación secuencial del algoritmo de estimación de estado de dos etapas en lenguaje C.
- Identificar operaciones que puedan ser calculadas en la GPU para realizar paralelización del algoritmo de estimación de estado de dos etapas.
- Planteamiento de casos de estudio para evaluar y comparar el desempeño del algoritmo programado secuencialmente y el algoritmo propuesto. Se realiza una comparación en base a métricas del procesamiento en paralelo como la aceleración.
- En la última etapa se realizan las conclusiones y recomendaciones que puedan mejorar lo realizado en este trabajo.

1.5. Aportación de la tesis

Se presenta el diseño de un algoritmo paralelo que optimiza el tiempo en el que se realiza el estudio de estimación de estado estático de dos etapas, en comparación con el algoritmo programado secuencialmente.

1.6. Contenido de la tesis

El contenido de esta tesis consta de 6 capítulos cuyo contenido se explica brevemente a continuación:

- **Capítulo 2.** En este capítulo se presenta la formulación de estimación de estado convencional, estimación de estado lineal y la formulación en dos etapas utilizada en este trabajo.

- **Capítulo 3.** En este capítulo se explican conceptos relacionados con el procesamiento en paralelo basado en GPUs y el modelo de programación utilizando la plataforma CUDA.

- **Capítulo 4.** Se describe el diseño del algoritmo paralelo de estimación de dos etapas. En esta etapa se buscan las operaciones del algoritmo que puedan llevarse a cabo utilizando la plataforma CUDA y que puedan disminuir el tiempo de ejecución.

- **Capítulo 5.** En este capítulo se describe la aplicación del algoritmo paralelo mediante 6 casos de estudio. Los sistemas utilizados son de: 14, 30, 118, 300, 500 y 1354 nodos. Se evalúa el desempeño y tiempo de ejecución del algoritmo propuesto.

- **Capítulo 6.** Se presentan las conclusiones y recomendaciones para trabajos de investigación futuros.

Capítulo 2

Estimación de estado

2.1. Introducción

La estimación de estado es una herramienta ampliamente utilizado por los Centros de Control de Energía y es la base de otras aplicaciones. De forma clásica, el estimador de estado cuenta con las siguientes funciones [Gomez-Exposito *et al.*, 2018]:

1. **Procesador topológico:** obtiene el modelo de la red mediante el estado de seccionadores e interruptores.

2. **Prefiltrado de mediciones:** se realiza una revisión de las mediciones para descartar mediciones con errores gruesos como magnitudes de voltaje negativas.

3. **Análisis de observabilidad:** esta función determina si la distribución de las mediciones permite que se pueda realizar la estimación sobre toda la red. En caso contrario, forma islas observables.

4. **Estimación de estado:** determina el estado del sistema de potencia calculando los voltajes complejos en cada nodo. Para ello utiliza el modelo proporcionado por el procesador topológico y las mediciones disponibles.

5. **Procesador de mediciones erróneas:** detecta e identifica posibles errores en el conjunto de mediciones.

En la Figura 2.1 se muestra la relación entre las funciones descritas anteriormente.

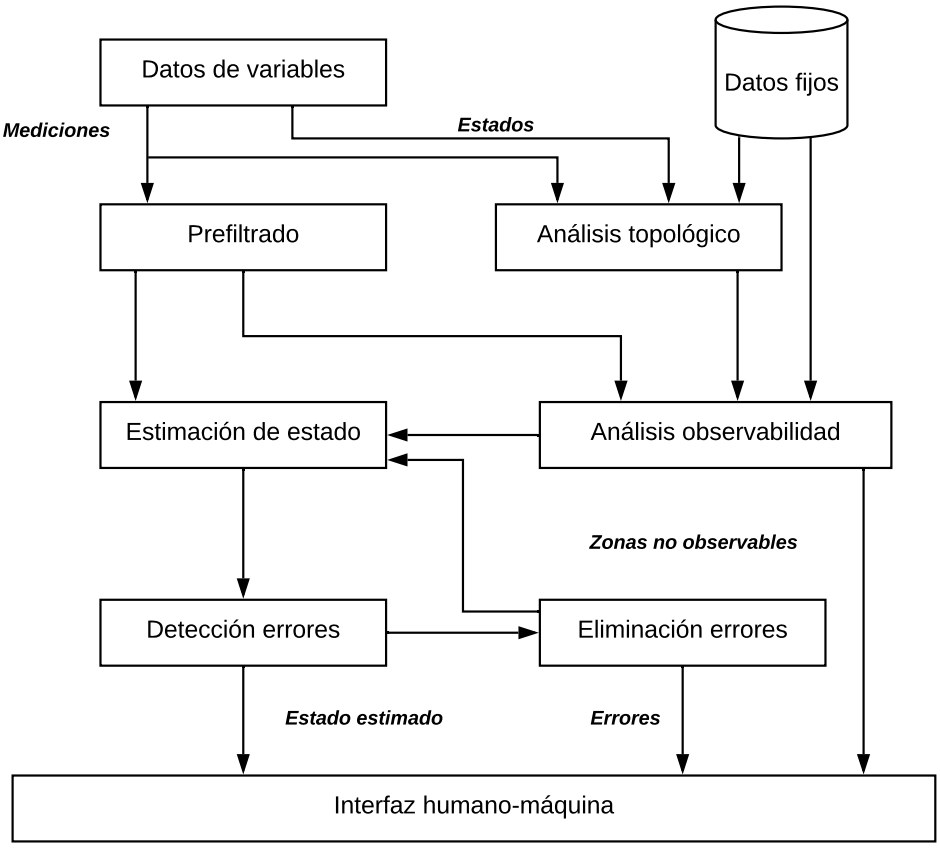


Figura 2.1: Funciones del estimador de estado

2.2. Estimación convencional

El primer método descrito en este capítulo es el método convencional por mínimos cuadrados ponderados. Este método iterativo consiste en calcular $2n$-1 variables de estado, donde n es el número de nodos del sistema. Se toma como referencia, un nodo donde el valor de su ángulo se mantiene en un valor especificado, generalmente cero, durante el proceso de estimación.

La principal ventaja de este método es su precisión. Entre las desventajas se encuentran que el resultado es afectado por la localización de las mediciones y el tipo de mediciones a utilizar, es decir, se necesita que las mediciones se distribuyan en el sistema de forma óptima

con el fin de obtener la mejor estimación posible [Sharma y Jain, 2018].

2.2.1. Formulación matemática

Considerando un vector z que contiene m número de mediciones, este vector se representa por la ecuación (2.1).

$$[z] = [h(x)] + [e] \tag{2.1}$$

donde

$h(x)$ es una función no lineal que relaciona las mediciones con los estados y está expresada en coordenadas cartesianas.

e es el error asociado a cada medición.

x es el vector de estados, expresado en términos de sus partes reales e imaginarias.

El proceso de estimación por medio del concepto de mínimos cuadrados ponderados consiste en minimizar la función objetivo que aparece en la ecuación (2.2).

$$J(x) = \sum_{i=1}^{m} \frac{(z_i - h_i(x))^2}{R_{ii}} \tag{2.2}$$

donde

m es el número de mediciones existentes.

z_i es el elemento i del vector de mediciones z.

$h_i(x)$ es el elemento i de la función de mediciones asociada a z_i.

R_{ii} es la covarianza asociada a z_i.

Los estimadores convencionales emplean conjuntos de mediciones que son funciones no lineales del vector de estados. Este conjunto de funciones se representa como la función de mediciones h.

En forma matricial la ecuación (2.2) se puede representar como:

$$J(x) = [z - h(x)]^T R^{-1} [z - h(x)] \tag{2.3}$$

R es una matriz diagonal de dimensiones $m \times m$, donde m es el número de mediciones dispo-

nibles. Los valores diagonales de esta matriz corresponden a la covarianza de cada medición. La matriz de covarianzas asigna un peso a cada tipo de medición de acuerdo al grado de precisión del aparato con que se realiza la telemetría.

$$[R] = \begin{bmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & \\ & & \sigma_3^2 & & \\ & & & \ddots & \\ & & & & \sigma_m^2 \end{bmatrix} \tag{2.4}$$

El mínimo de la ecuación (2.2) se tiene cuando $\frac{\partial J(x)}{\partial x} = 0$. Por lo que se debe cumplir que,

$$g(x) = \frac{\partial J\,(x)}{\partial x} = -H^T\,(x)\,R^{-1}\,[z - h\,(x)] = 0 \tag{2.5}$$

donde

$$[H] = \frac{\partial h(x)}{\partial x}$$

H es la matriz Jacobiana formada por la primera derivada de cada elemento de la función de mediciones h con respecto a cada variable de estado. Cada fila representa una medición y cada columna una variable de estado como se muestra en la ecuación (2.6).

$$[H] = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \frac{\partial h_m}{\partial x_2} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix} \tag{2.6}$$

La expansión de la función no lineal $g(x)$ en su serie de Taylor con respecto al vector de estado x^k es dada por:

$$g(x) = g(x^k) + G(x^k)(x - x^k) + ... = 0 \tag{2.7}$$

donde

$$G(x^k) = \frac{\partial g(x)}{\partial x} = [H]^T[R]^{-1}[H]$$

Despreciando los términos de orden superior, el vector de estados en la iteración $k + 1$ se

obtiene mediante el método de Gauss-Newton[Tarali y Abur, 2012]:

$$\begin{aligned} x^{k+1} &= x^k - [G(x^k)]^{-1} \cdot g(x) \\ &= x^k + [G(x^k)]^{-1}[H]^T[R]^{-1}[z - h(x)] \end{aligned} \tag{2.8}$$

donde

x^{k+1} es el vector de estados actualizado.

x^k es el vector de estados obtenidos en la iteración anterior.

$G(x)$ es la matriz de ganancia. Esta matriz es dispersa y definida positiva para garantizar que el sistema sea observable.

Al ser G una matriz dispersa, tradicionalmente no se calcula su inversa, sino que se descompone en dos matrices triangulares mediante la factorización LU o Cholesky. La ecuación (2.8) puede ser reescrita como:

$$[G(x^k)](x^{k+1} - x^k) = [H]^T[R]^{-1}[z - h(x)] \tag{2.9}$$

donde $x^{k+1} - x^k$ es la corrección Δx. Partiendo de la Ecuación (2.9), Δx se calcula resolviendo un sistema de ecuaciones linealizado utilizando la factorización LU, basada en sustituciones hacia delante y hacia atrás, respectivamente. Una vez que la matriz G se descompone como $G = LU$, se resuelven los siguientes sistemas de ecuaciones utilizando y como un vector auxiliar.

$$\begin{aligned} [L]y &= [H]^T[R]^{-1}[z - h(x)] \\ [U]\Delta x &= y \end{aligned} \tag{2.10}$$

En la Figura 2.2 se observa el algoritmo de estimación de estado convencional. El proceso de estimación de estado (en coordenadas cartesianas) parte de las siguientes condiciones iniciales: la parte real de los voltajes se inicializa con 1 y la parte imaginaria con 0. El vector de estados x se actualiza en cada iteración hasta que se cumple con la condición de que el máximo incremento de una de las variables de estado sea menor que la tolerancia establecida, es decir $max|\Delta x| < \varepsilon$.

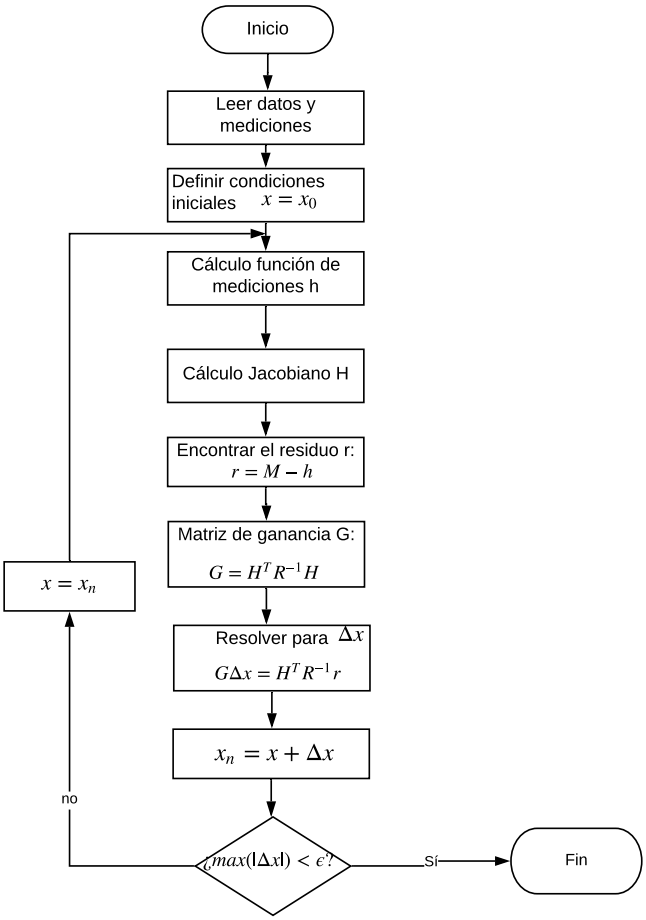


Figura 2.2: Algoritmo de estimación convencional

2.2.2. Elementos del algoritmo de estimación de estado convencional

Función de mediciones h

Las ecuaciones (2.11)-(2.14) representan cada una de las mediciones existentes y son utilizadas en el cálculo de la función de mediciones h. Donde h es de dimensiones mx1 siendo m el número de mediciones existentes. Las ecuaciones fueron obtenidas de [Tarali y Abur, 2012].

- Inyección de potencia activa y reactiva:

$$P_i = \sum_{j=1}^{N} (G_{ij}(e_i e_j + f_i f_j) + B_{ij}(f_i e_j + e_i f_j)) \tag{2.11}$$

$$Q_i = \sum_{j=1}^{N} (G_{ij}(f_i e_j - e_i f_j) - B_{ij}(e_i e_j + f_i f_j)) \tag{2.12}$$

- Flujos de potencia activa y reactiva:

$$P_{ij} = (e_i^2 + f_i^2)g_{ij} - g_{ij}(e_i e_j + f_i f_j) - b_{ij}(f_i e_j - e_i f_j) \tag{2.13}$$

$$Q_{ij} = -(e_i^2 + f_i^2)(b_{ij} + b_{si}) - g_{ij}(f_i e_j - e_i f_j) + b_{ij}(e_i e_j + f_i f_j) \tag{2.14}$$

donde

$G_{ij} + jB_{ij}$ es el elemento ij de la matriz admitancia.

$g_{ij} + jb_{ij}$ es el elemento ij de la matriz de admitancia serie.

$e_i + jf_i$ es el voltaje en el nodo i en forma rectangular.

N es el número de nodos.

Estructura y Formación de la matriz H

La matriz jacobiana H es de dimensiones $m \times n$ donde m es el número de mediciones totales y n el número de variables de estado. En la formulación rectangular el número de variables de estado es igual al doble del número de nodos. La matriz H se divide en 12 submatrices para su cálculo. En la Figura 2.3 se aprecia la estructura de H. En las ecuaciones (2.15)-(2.32) se muestran los elementos de la matriz jacobiana con respecto a cada tipo de medición.

- Elementos de la matriz jacobiana correspondientes a las mediciones de voltaje nodal:

$$\frac{\partial e_i}{\partial e_i} = 1 \quad \frac{\partial e_i}{\partial e_j} = 0 \quad \frac{\partial e_i}{\partial f_i} = 0 \quad \frac{\partial e_i}{\partial f_j} = 0 \tag{2.15}$$

$$\frac{\partial f_i}{\partial e_i} = 0 \quad \frac{\partial f_i}{\partial e_j} = 0 \quad \frac{\partial f_i}{\partial f_i} = 1 \quad \frac{\partial f_i}{\partial f_j} = 0 \tag{2.16}$$

- Elementos de la matriz jacobiana correspondientes a las mediciones de inyección de potencia real:

$$\frac{\partial P_i}{\partial f_i} = \sum_{j=1}^{N} (G_{ij} f_j + B_{ij} e_j) + G_{ii} f_i - B_{ii} e_i \tag{2.17}$$

$$H = \begin{array}{|c|c|}\hline \frac{\partial e_i}{\partial e} & \frac{\partial e_i}{\partial f} \\ \hline \frac{\partial f_i}{\partial e} & \frac{\partial f_i}{\partial f} \\ \hline \frac{\partial P_i}{\partial e} & \frac{\partial P_i}{\partial f} \\ \hline \frac{\partial Q_i}{\partial e} & \frac{\partial Q_i}{\partial f} \\ \hline \frac{\partial P_{ij}}{\partial e} & \frac{\partial P_{ij}}{\partial f} \\ \hline \frac{\partial Q_{ij}}{\partial e} & \frac{\partial Q_{ij}}{\partial f} \\ \hline \end{array}$$

Figura 2.3: Estructura matriz Jacobiana H

$$\frac{\partial P_i}{\partial f_j} = G_{ij} f_i - B_{ij} e_i \tag{2.18}$$

$$\frac{\partial P_i}{\partial e_i} = \sum_{j=1}^{N} (G_{ij} e_j - B_{ij} f_j) + G_{ii} e_i + B_{ii} f_i \tag{2.19}$$

$$\frac{\partial P_i}{\partial e_j} = G_{ij} e_i + B_{ij} f_i \tag{2.20}$$

- Elementos de la matriz jacobiana correspondientes a las mediciones de inyección de potencia reactiva:

$$\frac{\partial Q_i}{\partial f_i} = \sum_{j=1}^{N} (G_{ij} e_j - B_{ij} f_j) - G_{ii} e_i - B_{ii} f_i \tag{2.21}$$

$$\frac{\partial Q_i}{\partial f_j} = -G_{ij} e_i - B_{ij} f_i \tag{2.22}$$

$$\frac{\partial Q_i}{\partial e_i} = \sum_{j=1}^{N} (-G_{ij} f_j - B_{ij} e_j) + G_{ii} f_i - B_{ii} e_i \tag{2.23}$$

$$\frac{\partial Q_i}{\partial e_j} = G_{ij} f_i - B_{ij} e_i \tag{2.24}$$

- Elementos de la matriz jacobiana correspondientes a las mediciones de flujo de potencia real:

$$\frac{\partial P_{ij}}{\partial f_i} = 2g_{ij} f_i - g_{ij} f_j - b_{ij} e_j \tag{2.25}$$

$$\frac{\partial P_{ij}}{\partial f_j} = -g_{ij} f_i + b_{ij} e_i \tag{2.26}$$

$$\frac{\partial P_{ij}}{\partial e_i} = 2g_{ij} e_m - g_{ij} e_j + b_{ij} f_j \tag{2.27}$$

$$\frac{\partial P_{ij}}{\partial e_j} = -g_{ij} e_i - b_{ij} f_i \tag{2.28}$$

- Elementos de la matriz jacobiana correspondientes a las mediciones de flujo de potencia reactiva:

$$\frac{\partial Q_{ij}}{\partial f_i} = -2(b_{ij}+b_{si}) f_i - g_{ij} e_j + b_{ij} f_j \tag{2.29}$$

$$\frac{\partial Q_{ij}}{\partial f_j} = g_{ij} e_i + b_{ij} f_i \tag{2.30}$$

$$\frac{\partial Q_{ij}}{\partial e_i} = -2(g_{ij} + b_{sm}) e_i + g_{ij} f_j + b_{ij} e_j \tag{2.31}$$

$$\frac{\partial Q_{ij}}{\partial e_j} = -g_{ij} f_i + b_{ij} e_i \tag{2.32}$$

2.3. Estimación de estado lineal

Una forma simple de explicar el problema de estimación lineal es utilizando el modelo π de una línea de transmisión [Phadke y Thorp, 2008] mostrado en la Figura 2.3. Si se coloca un PMU en cada extremo de la línea de transmisión, se tendrá como mediciones fasores de voltaje en cada nodo y el flujo de corriente en la línea. Considerando todos los valores en coordenadas rectangulares el vector de estados x es:

$$[x] = \begin{bmatrix} V_i \\ V_j \end{bmatrix} \tag{2.33}$$

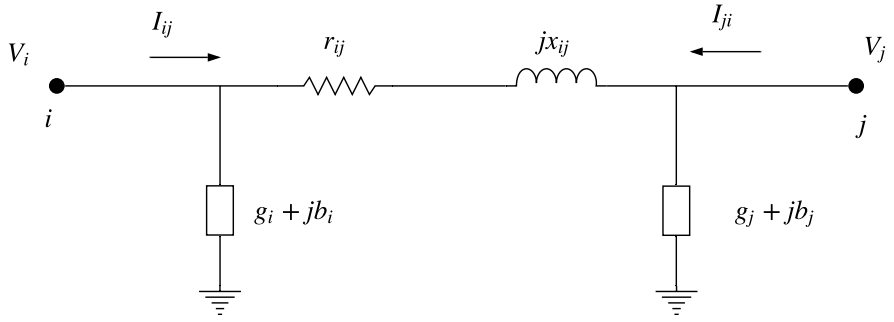


Figura 2.4: Modelo PI de una línea de transmisión

El conjunto de mediciones z se representa como:

$$[z] = \begin{bmatrix} V_i \\ V_j \\ I_{ij} \\ I_{ji} \end{bmatrix} \tag{2.34}$$

La admitancia serie y la susceptancia en derivación de la línea son las siguientes:

$$y_{ij} = (r_{ij} + jx_{ij})^{-1}$$
$$y_{i0} = g_i + jb_i$$
$$y_{j0} = g_j + jb_j$$

Utilizando las leyes de Kirchhoff, la relación entre el estado del sistema y los flujos de corriente en esta línea son:

$$\begin{bmatrix} I_{ij} \\ I_{ji} \end{bmatrix} = \begin{bmatrix} y_{ij} + y_{i0} & -y_{ij} \\ -y_{ij} & y_{ij} + y_{j0} \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix} \tag{2.35}$$

La ecuación de estado completa toma la siguiente forma:

$$\begin{bmatrix} V_i \\ V_j \\ I_{ij} \\ I_{ji} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ y_{ij} + y_{i0} & -y_{ij} \\ -y_{ij} & y_{ij} + y_{j0} \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix} \tag{2.36}$$

2.3.1. Formulación matemática

Si en un sistema se utilizan únicamente mediciones PMU los estados del sistema se están midiendo directamente. El conjunto de mediciones fasoriales se representan mediante la ecuación (2.37):

$$[z] = [H]x + e \tag{2.37}$$

donde

$$[H] = \begin{bmatrix} II \\ yA + ys \end{bmatrix} \tag{2.38}$$

En la Ecuación (2.38), II es la matriz de incidencia de voltaje, A es la matriz de incidencia de corriente, y es la matriz de admitancias serie y ys es la matriz de admitancias en derivación.

El conjunto de mediciones está formado por fasores de voltaje y mediciones de flujo de corriente:

$$[z] = \begin{bmatrix} V_{pmu} \\ I_{pmu} \end{bmatrix} \tag{2.39}$$

El vector de estados x se encuentra por minimos cuadrados ponderados utilizando la ecuación (2.40):

$$[x] = (H^T R^{-1} H)^{-1} H^T R^{-1} z \tag{2.40}$$

A continuación se describe como se forman las matrices II, A, y y ys [Jones, 2011].

Matriz de incidencia de voltajes II

La matriz II es de dimensiones $m \times n$ donde m son las mediciones de fasores de voltaje provenientes de las PMUs y n los nodos del sistema. Esta matriz muestra la ubicación que tienen las PMUs en el sistema. Cada fila representa una medición de voltaje y cada columna un nodo del sistema. Para su cálculo se coloca 1 en la columna asociada al nodo donde se ubica la PMU.

Matriz de incidencia de corriente A

La matriz de incidencia A es una matriz de dimensiones $m \times n$ donde m son las mediciones de fasores de corriente y n los nodos del sistema. Cada fila representa una medición de flujo de corriente y cada columna un nodo del sistema. Para su cálculo se sigue una regla:

se coloca un 1 en la columna asociada al nodo de envío y -1 en la columna que pertenece al nodo de recepción.

Matriz de admitancia serie y

La matriz de admitancia serie y es una matriz diagonal de dimensiones $m \times n$ donde m es el número de mediciones de fasores de corriente y n el número de nodos. Cada elemento de la diagonal es la admitancia de la rama de donde se toma la medición.

Matriz de admitancia en derivación ys

La matriz de admitancia en derivación ys es de dimensiones $m \times n$ donde m es el número de mediciones de fasores de corriente y n el número de nodos. En la columna asociada al nodo de envío de donde se toma la medición se coloca la admitancia en derivación de la línea medida.

Para ejemplificar el cálculo de las matrices II, A, ys y y se considera el sistema IEEE de 14 nodos en el cual se colocan 3 PMUs en los nodos 4, 7 y 14, respectivamente. Se consideran 3 mediciones de fasores de voltaje y 2 mediciones de fasores de corriente. En la Figura 2.4 se muestran la distribución de las PMUs y las mediciones de fasores de corriente.

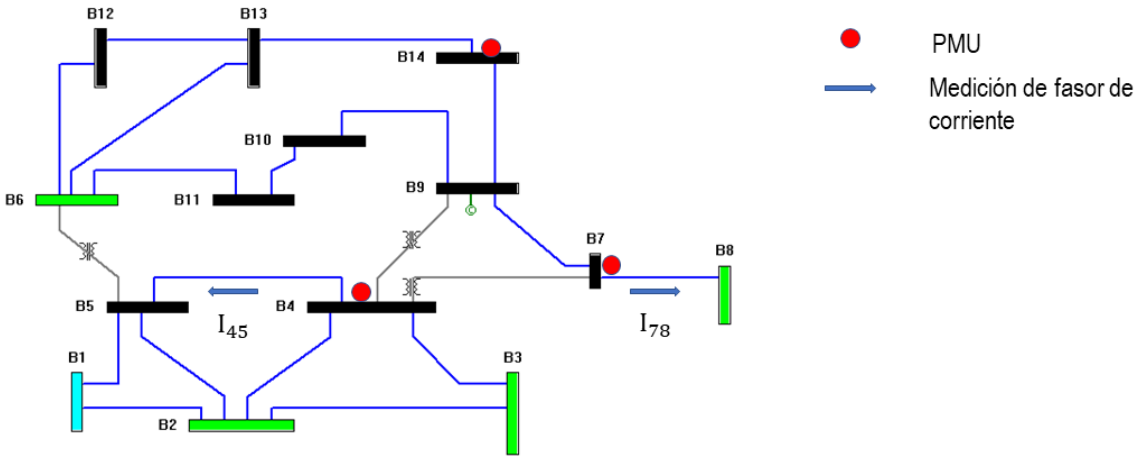


Figura 2.5: Distribución de las PMUs en el sistema de 14 nodos

La matriz de admitancia nodal para el sistema de 14 nodos es:

Tabla 2.1: Y_{bus} del sistema IEEE 14 nodos

NE,NR	Y	NE,NR	Y
(1,1)	6.0250 -19.4471i	(4,7)	0.0000 + 4.8895i
(2,1)	-4.9991 +15.2631i	(7,7)	0.0000 -19.5490i
(5,1)	-1.0259 + 4.2350i	(8,7)	0.0000 + 5.6770i
(1,2)	-4.9991 +15.2631i	(9,7)	0.0000 + 9.0901i
(2,2)	9.5213 -30.2707i	(7,8)	0.0000 + 5.6770i
(3,2)	-1.1350 + 4.7819i	(8,8)	0.0000 - 5.6770i
(4,2)	-1.6860 + 5.1158i	(4,9)	0.0000 + 1.8555i
(5,2)	-1.7011 + 5.1939i	(7,9)	0.0000 + 9.0901i
(2,3)	-1.1350 + 4.7819i	(9,9)	5.3261 -24.0925i
(3,3)	3.1210 - 9.8115i	(10,9)	-3.9020 +10.3654i
(4,3)	-1.9860 + 5.0688i	(14,9)	-1.4240 + 3.0291i
(2,4)	-1.6860 + 5.1158i	(9,10)	-3.9020 +10.3654i
(3,4)	-1.9860 + 5.0688i	(10,10)	5.7829 -14.7683i
(4,4)	10.5130 -38.6352i	(11,10)	-1.8809 + 4.4029i
(5,4)	-6.8410 +21.5786i	(6,11)	-1.9550 + 4.0941i
(7,4)	0.0000 + 4.8895i	(10,11)	-1.8809 + 4.4029i
(9,4)	0.0000 + 1.8555i	(11,11)	3.8359 - 8.4970i
(1,5)	-1.0259 + 4.2350i	(6,12)	-1.5260 + 3.1760i
(2,5)	-1.7011 + 5.1939i	(12,12)	4.0150 - 5.4279i
(4,5)	-6.8410 +21.5786i	(13,12)	-2.4890 + 2.2520i
(5,5)	9.5680 -35.5275i	(6,13)	-3.0989 + 6.1028i
(6,5)	0.0000 + 4.2574i	(12,13)	-2.4890 + 2.2520i
(5,6)	0.0000 + 4.2574i	(13,13)	6.7249 -10.6697i
(6,6)	6.5799 -17.3407i	(14,13)	-1.1370 + 2.3150i
(11,6)	-1.9550 + 4.0941i	(9,14)	-1.4240 + 3.0291i
(12,6)	-1.5260 + 3.1760i	(13,14)	-1.1370 + 2.3150i
(13,6)	-3.0989 + 6.1028i	(14,14)	2.5610 - 5.3440i

En la Tabla 2.2 se muestra las admitancias en derivación de las líneas del sistema IEEE de 14 nodos cuyo valor es distinto de cero.

Tabla 2.2: Admitancias en derivación del sistema IEEE 14 nodos

NE	NR	Ys
1	2	0.0528i
1	5	0.0492i
2	3	0.0438i
2	4	0.0340i
2	5	0.0346i
3	4	0.0128i

La matriz A de incidencia de corriente para el sistema de 14 nodos es:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La matriz de II de incidencia de voltaje es:

$$II = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz y es una matriz diagonal de 2×2 y cada elemento es la admitancia serie de cada línea de donde se toma la medición de corriente.

$$y = \begin{bmatrix} \text{6.8410-21.578i} & 0 \\ 0 & \text{-5.6770i} \end{bmatrix}$$

La matriz ys es de dimensiones 2×14. En la primera fila en la columna 4 se coloca la admitancia en derivación de la línea que va del nodo 4 al 5, la admitancia en derivación de la segunda línea medida es 0.

$$ys = \begin{bmatrix} 0 & 0 & 0 & \text{0.0128i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.3.2. Formulación alternativa

Aunque el manejo de números complejos de la formulación anterior no representa ningún problema en entornos como MATLAB®, en la vida real resulta más fácil el manejo de estos números separándolos en parte real e imaginaria. La ecuación (2.37) puede ser reescrita como

$$\begin{bmatrix} z_r \\ z_i \end{bmatrix} = \begin{bmatrix} II & 0 \\ 0 & II \\ gA + gs & -bA - bs \\ bA + bs & gA + gs \end{bmatrix} \begin{bmatrix} v_r \\ v_i \end{bmatrix} + [e] \tag{2.41}$$

donde

$gA + gs$ es la parte real de $yA + ys$

$bA + bs$ es la parte imaginaria de $yA + ys$

2.4. Estimación de estado de dos etapas

En esta sección se muestra la formulación para la segunda etapa del estimador. Se utilizan los subindices 1 y 2 para hacer referencia a las matrices y vectores que se calculan en la primera y segunda etapa del estimador, respectivamente.

2.4.1. Formulación matemática

Una vez terminado el proceso de estimación convencional (primera etapa), el vector de estados x_1 y las mediciones PMU son utilizados para realizar nuevamente el proceso de estimación. En la segunda etapa lo que se busca es mejorar el resultado obtenido de la estimación convencional a partir de la inclusión de mediciones PMU y la utilización de un cálculo lineal.

El vector de mediciones en la segunda etapa se relaciona con los estados como se muestra en la ecuación (2.42).

$$[z_2] = [H_2]x + e \tag{2.42}$$

donde

$$[H_2] = \begin{bmatrix} & I & \\ II & & 0 \\ 0 & & II \\ gA + gs & & -bA - bs \\ bA + bs & & gA + gs \end{bmatrix}$$

I es una matriz identidad de $2n \times 2n$ donde n es el número de nodos.

El vector z_2 está compuesto por la parte real e imaginaria del vector de estado estimado y de las mediciones procedentes de las PMUs. La variable $Xest_{real}$ y $Xest_{imag}$ contiene la parte real e imaginaria del vector de estados resultado de la primera etapa. En la ecuación

(2.43) se muestra la estructura del vector de mediciones.

$$[z_2] = \begin{bmatrix} Xest_{real} \\ Xest_{imag} \\ Vpmu_{real} \\ Vpmu_{imag} \\ Ipmu_{real} \\ Ipmu_{imag} \end{bmatrix} \tag{2.43}$$

Debido a que las mediciones de fasores de corriente y voltaje se encuentran en forma polar y los cálculos se realizan en forma rectangular es necesario transformar las mediciones PMU de polar a rectangular así como su matriz de covarianza. La matriz de covarianza de las mediciones PMU se transforma a rectangular por medio de una matriz de rotación T como se muestra en la ecuación (2.44).

$$[R_{pmu}] = [T][R_p][T^T] \tag{2.44}$$

donde

$$[T] = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix} \tag{2.45}$$

La matriz T es de dimensiones $2(m+n) \times 2(m+n)$ donde m es el número de mediciones de fasores de voltaje y n el número de mediciones de fasores de corriente. La matriz T_{11} se calcula como se muestra en la ecuación (2.46):

$$[\mathrm{T_{11}}] = \begin{bmatrix} \begin{bmatrix} cos(\theta_{v1}) & & & \\ & cos(\theta_{v2}) & & \\ & & \ddots & \\ & & & cos(\theta_{vm}) \end{bmatrix} & \begin{bmatrix} -V_1 sin(\theta_{v1}) & & & \\ & -V_2 sin(\theta_{v2}) & & \\ & & \ddots & \\ & & & -V_m sin(\theta_{vm}) \end{bmatrix} \\ \begin{bmatrix} sin(\theta_{v1}) & & & \\ & sin(\theta_{v2}) & & \\ & & \ddots & \\ & & & sin(\theta_{vm}) \end{bmatrix} & \begin{bmatrix} V_1 cos(\theta_{v1}) & & & \\ & V_2 cos(\theta_{v2}) & & \\ & & \ddots & \\ & & & V_m(\theta_{vm}) \end{bmatrix} \end{bmatrix} \tag{2.46}$$

donde

V_m es la medición de magnitud de voltaje m.

θ_{vm} es el ángulo asociada a la medición de fasor de voltaje m.

La matriz T_{22} está asociada a las mediciones de fasores de corriente y se calcula como se muestra en la ecuación (2.47):

$$[\mathrm{T}_{22}] = \begin{bmatrix} \begin{bmatrix} cos(\theta_{i1}) & & & \\ & cos(\theta_{i2}) & & \\ & & \ddots & \\ & & & cos(\theta_{in}) \end{bmatrix} & \begin{bmatrix} -I_1 sin(\theta_{i1}) & & & \\ & -I_2 sin(\theta_{i2}) & & \\ & & \ddots & \\ & & & -I_m sin(\theta_{in}) \end{bmatrix} \\ \begin{bmatrix} sin(\theta_{i1}) & & & \\ & sin(\theta_{i2}) & & \\ & & \ddots & \\ & & & sin(\theta_{in}) \end{bmatrix} & \begin{bmatrix} I_1 cos(\theta_{i1}) & & & \\ & I_2 cos(\theta_{i2}) & & \\ & & \ddots & \\ & & & I_m(\theta_{in}) \end{bmatrix} \end{bmatrix} \tag{2.47}$$

donde

I_m es magnitud del fasor de corriente m.

θ_{in} es el ángulo en radianes del fasor de corriente n.

La matriz R_2 de covarianza para la segunda etapa se forma a partir de la matriz de covarianza del vector estimado x_1 resultante de la primera etapa y la matriz de covarianza en coordenadas rectangulares de las mediciones PMU utilizadas en la segunda etapa como se muestra en la ecuación (2.48).

$$[R_2] = \begin{bmatrix} Cov & 0 \\ 0 & R_{pmu} \end{bmatrix} \tag{2.48}$$

donde

$$[Cov] = [H_1^T R_1^{-1} H_1]^{-1}$$

Finalmente, el vector de estados x_2 en coordenadas rectangulares se calcula utilizando la ecuación (2.49):

$$[x_2] = (H_2 R_2^{-1} H_2^T)^{-1} H_2 R_2^{-1} z_2 \tag{2.49}$$

2.4.2. Ejemplo de aplicación: sistema de 5 nodos

Con el fin de ejemplificar el método de estimación de estado de dos etapas, se considera el sistema de 5 nodos reportado en [Tarali y Abur, 2012]. Para el sistema de 5 nodos y 10 estados, para la primera etapa del estimador se utilizaron 12 mediciones. Las mediciones se muestran en la Tabla 2.3. La distribución de las mediciones en el sistema se muestra en la Figura 2.5.

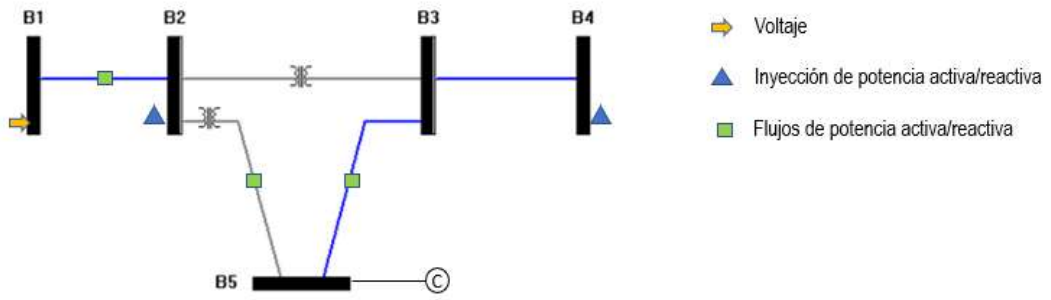


Figura 2.6: Distribución de las mediciones convencionales en el sistema de 5 nodos.

Tabla 2.3: Mediciones consideradas en la primera etapa

Tipo	**Magnitud**	**Covarianza**
V_1	1.0100	10^{-4}
θ_1	0	10^{-4}
P_2	-0.4780	10^{-4}
P_4	0	10^{-4}
Q_2	0.0390	10^{-4}
Q_4	0.21342	10^{-4}
P_{1-2}	0.8186	10^{-4}
P_{2-5}	0.10725	10^{-4}
P_{3-5}	0.18771	10^{-4}
Q_{1-2}	-0.18452	10^{-4}
Q_{2-5}	-0.04726	10^{-4}
Q_{3-5}	0.01280	10^{-4}

Como primer paso se lleva a cabo la estimación convencional. El jacobiano en la primera iteración es el siguiente:

$$H=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-1.9860 & 1.9860 & 0 & 0 & 0 & -5.0688 & 11.8138 & -4.8895 & 0 & -1.8555 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -5.6770 & 5.6770 & 0 \\
-5.0688 & 12.1179 & -4.8895 & 0 & -1.8555 & 1.9860 & -1.9860 & 0 & 0 & 0 \\
0 & 0 & -5.6770 & 5.6770 & 0 & 0 & 0 & 0 & 0 & 0 \\
1.9860 & -1.9860 & 0 & 0 & 0 & 5.0688 & -5.0688 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.8555 & 0 & 0 & -1.8555 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.0901 & 0 & -9.0901 \\
5.0342 & -5.0688 & 0 & 0 & 0 & -1.9860 & 1.9860 & 0 & 0 & 0 \\
0 & 1.9742 & 0 & 0 & -1.8555 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 9.0901 & 0 & -9.0901 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

La primera etapa del estimador converge después de 5 iteraciones. El valor de la función objetivo es $J(x)$ =0.00001565, dando como resultado un vector de 10 estados.

$$x_1=\begin{bmatrix}
1.0100 \\
0.9839 \\
1.0369 \\
1.0708 \\
1.0320 \\
0.000 \\
-0.1497 \\
-0.1968 \\
-0.2032 \\
-0.2152
\end{bmatrix}$$

Para la segunda etapa se consideran 2 mediciones provenientes del PMU ubicado en el nodo 3. Las mediciones consideradas se pueden observar en la Tabla 2.4.

Tabla 2.4: Mediciones PMU utilizadas en la segunda etapa

Nodo	**Magnitud**	**Covarianza**
$\|V_3\|$	1.055	10^{-6}
$\angle V_3$	-10.75°	10^{-6}
$\|I_{3-4}\|$	0.1962	10^{-6}
$\angle I_{3-4}$	79.26°	10^{-6}

El vector de estados x_2 para la segunda etapa se realiza de forma directa, formando el vector z_2, la matriz H_2 y la matriz R_2 de varianzas. El vector z_2 está conformado por la parte real e imaginaria de los estados resultado de la primera etapa y la parte real e imaginaria de las mediciones PMU.

$$z_2 = \begin{bmatrix} 1.0100 \\ 0.9839 \\ 1.0369 \\ 1.0708 \\ 1.0320 \\ 0.00 \\ -0.1497 \\ -0.1968 \\ -0.2032 \\ -0.2157 \\ 1.0365 \\ -0.1968 \\ 0.0366 \\ 0.1928 \end{bmatrix}$$

Y la matriz H_2 es:

$$H_2 = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.6770 & -5.6770 & 0 \\ 0 & 0 & -5.6770 & 5.6770 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Utilizando la ecuación (2.42), el vector de estados en la segunda etapa es:

$$x_2 = \begin{bmatrix} 1.0100 \\ 0.9835 \\ 1.0365 \\ 1.0704 \\ 1.0315 \\ 0.000 \\ \text{-}0.1497 \\ \text{-}0.1968 \\ \text{-}0.2032 \\ \text{-}0.2158 \end{bmatrix}$$

En la Tabla 2.5 se realiza una comparación de los resultados del estudio de flujos de potencia y de cada etapa del estimador. Debido a que las mediciones se consideraron sin errores y a la inclusión de las mediciones PMU, los resultados de la segunda etapa son más cercanos a los resultados obtenidos de flujos de potencia.

Tabla 2.5: Resultados de la primera y segunda etapa del estimador para el sistema de 5 nodos

Nodo	**Flujos de potencia V(p.u)**	**Etapa 1 V(p.u)**	**Etapa 2 V(p.u)**
1*	1.0100 $\angle$ 0.000°	1.0100 $\angle$ 0.000°	1.0100 $\angle$ 0.000°
2	0.9950 $\angle$ -8.650°	0.9953 $\angle$ -8.651°	0.9948 $\angle$ -8.655°
3	1.0550$\angle$ -10.750°	1.0555$\angle$ -10.744°	1.0550 $\angle$ -10.750°
4	1.0900 $\angle$ -10.750°	1.0899 $\angle$ -10.744°	1.0896 $\angle$ -10.750°
5	1.0540 $\angle$ -11.810°	1.0543 $\angle$ -11.808°	1.0539 $\angle$ -11.814°

* Nodo *slack*

Una forma de evaluar los resultados obtenidos anteriormente es, calculando el Error Cuadrático Medio (MSE, por sus siglas en inglés).

$$MSE = \frac{\sum_{i=1}^{N}(x_i - \hat{x_i})^2}{N} \tag{2.50}$$

donde x_i es el valor real del estado (en este caso, se tomo como valor real el resultado obtenido del estudio de flujos de potencia), $\hat{x}_i$ como el valor obtenido a través del estimador y N el número de nodos.

En la Tabla 2.6 se muestran el MSE para cada etapa. Se observa que en la segunda etapa el MSE para los ángulos es menor en comparación con la primera etapa.

Tabla 2.6: MSE de cada etapa para el sistema de 5 nodos

Etapa	MSE magnitud de voltaje	MSE ángulos
1	$1.54\text{x}10^{-5}$	$8.200\text{x}10^{-6}$
2	$8.800\text{x}10^{-8}$	$7.740\text{x}10^{-8}$

2.5. Conclusiones

En este capítulo se presentó la formulación de estimación de estado convencional, lineal y de dos etapas. Aunque los inicios de la estimación de estado y las PMUs solo se encuentran separados por una década, esta tecnología sigue siendo costosa económicamente. Debido a esta razón, se han reportado trabajos utilizando tanto mediciones SCADA como PMU. Los estimadores híbridos y de dos etapas han sido reportadas como alternativas para incluir las mediciones PMU en el proceso de estimación de estado en estado estable.

Tanto la estimación de estado híbrida como la estimación en dos etapas pretenden ser una solución transitoria para la integración total de las PMUs en el proceso de estimación de estado. La principal ventaja que ofrece el algoritmo de estimación en dos etapas sobre el método híbrido es obtener los mismos resultados, procesando las mediciones por separado, sin realizar cambios importantes al software existente [Zhou *et al.*, 2006]. Adicionalmente, mejora los resultados obtenidos en la primera etapa. En otros trabajos de investigación ha sido reportado que otra ventaja de este método es que contribuye a la capacidad de detectar mediciones erróneas [Baltensperger *et al.*, 2010], esta ventaja quedó fuera de los alcances de este trabajo.

Capítulo 3

Procesamiento en paralelo

3.1. Introducción

La pregunta al hablar de procesamiento en paralelo es ¿Por qué paralelizar algoritmos?, la respuesta se basa en dos principales razones. La primera tiene que ver con las limitaciones físicas de los equipos de cómputo, ya que existen topes en frecuencia de reloj de las computadoras, estos topes tienen que ver con que a mayor frecuencia existe un mayor consumo y temperatura. La segunda razón se debe a la complejidad de los problemas en campos como la ciencia y la ingeniería.

El procesamiento en paralelo permite la utilización de más de un procesador y la asignación explicita de las tareas a realizar con el fin de que el esfuerzo computacional y el tiempo de ejecución de un programa disminuyan. Las aplicaciones del procesamiento en paralelo están asociado a problemas de gran escala, donde las operaciones y el procesamiento de los datos es computacionalmente costoso. En el área de ingeniería eléctrica la utilización del procesamiento en paralelo ha sido influenciado por el constante crecimiento de los sistemas eléctricos. En este capítulo se aborda el modelo de programación basado en GPUs y la plataforma CUDA.

3.2. Unidades de procesamiento Gráfico (GPU)

La GPU es un procesador que un principio solo tenía como propósito aligerar la carga de trabajo de la CPU en funciones relacionadas con el procesamiento de los gráficos. Su

principal atributo es su arquitectura compuesta de cientos de núcleos que permiten ejecutar cientos de programas al mismo tiempo.

Para mostrar la arquitectura de una GPU, en la Figura 3.1 se muestra una GPU GeForce GTX 295 de 30 multiprocesadores (MPs) y 240 procesadores escalares (SPs). Una GPU está conformada por un número de MPs. Cada MP está formado a su vez por SPs [Romero *et al.*, 2012]. La memoria del dispositivo es la memoria de la tarjeta gráfica y es compartida por todos los multiprocesadores. El cómputo utilizando GPU tiene como principal ventaja que el código se vuelve compacto ya que una sola instrucción define N tareas a realizar.

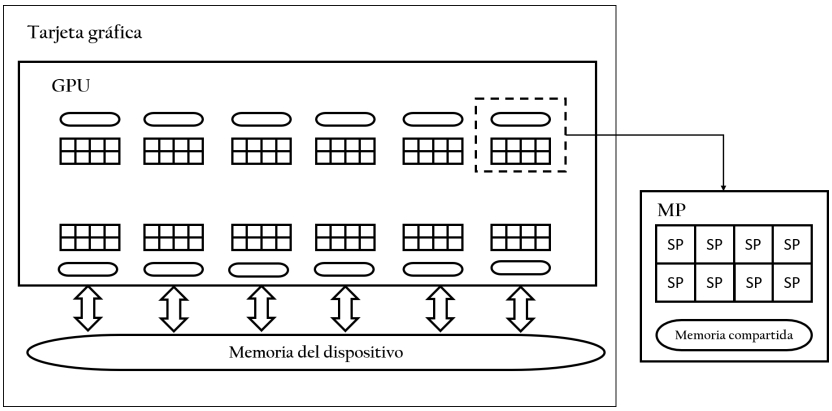


Figura 3.1: Arquitectura GeForce GTX 295

Debido a su arquitectura, la GPU necesita una gran cantidad de hilos para realizar las tareas asignadas, a diferencia de la CPU. Los hilos se pueden definir como una secuencia de instrucciones que se ejecuta de forma independiente dentro de un proceso. La utilización de hilos permite que se puedan obtener beneficios de este tipo de arquitecturas, ya que cada núcleo ejecuta una instrucción diferente dentro de un mismo proceso.

3.3. Modelo de programación en CUDA

CUDA es una plataforma para el cómputo en paralelo creada por NVIDIA para las GPU fabricadas por la misma compañía. El modelo de programación de CUDA se basa en

simple instrucción-múltiples datos (SIMD). CUDA utiliza el lenguaje C y un modelo de co-procesamiento, es decir, las tareas son divididas entre el CPU y la GPU. CUDA soporta lenguajes como: python, C/C++ y Fortran.

La CPU recibe el nombre de *host* y la GPU el nombre de *device*. Un programa en CUDA se divide en dos partes: la parte secuencial que se ejecuta en la CPU ① y la parte paralela que se ejecuta en la GPU ②, como se puede observar en la Figura 3.2. En un programa diseñado en CUDA se llevan a cabo los siguientes pasos [Laguna-Sánchez *et al.*, 2011]:

- La CPU o *host* ejecuta la parte principal del programa ①.
- Se reserva memoria en la GPU ①.
- Se copian los datos de la CPU a la GPU ①.
- La CPU llama al kernel ①.
- En la GPU se ejecuta el código paralelo ②.
- Los resultados se copian a la CPU y finalmente se libera la memoria en la GPU ②.
- La CPU termina el programa ①.

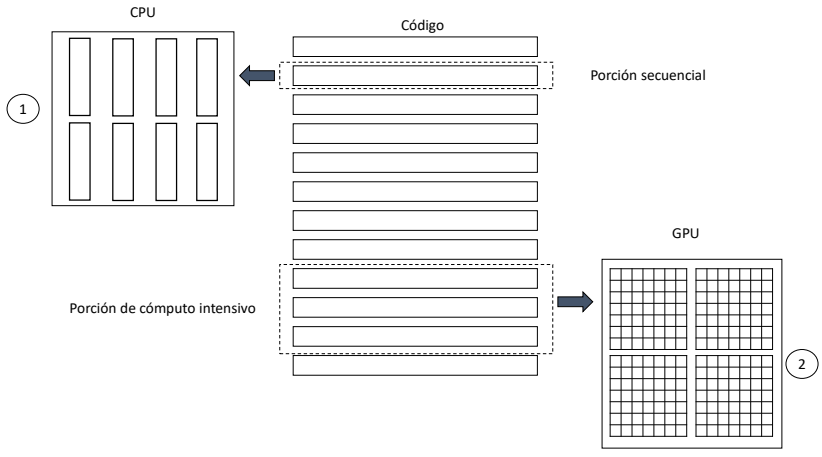


Figura 3.2: Modelo de programación CUDA

En un programa ejecutado en la plataforma CPU-GPU es necesaria una comunicación entre el *host* y el *device*. El *host* es el encargado de iniciar, dar por terminado el programa

y llamar al *device* para ejecutar las líneas que le son asignadas en el código. En las líneas ejecutadas en el *host* se realiza lo siguiente: reserva y liberación de memoria, llamado de *kernels*, intercambio de datos entre las memorias y el procesamiento de los datos obtenidos por la GPU. La parte del código ejecutado en el *device* se realiza a través de los *kernels* desde donde se tiene acceso a la memoria de la GPU. El kernel lanzará una malla que a su vez está compuesta por bloques [Torre, 2015].

Organización de hilos

La unidad básica de operación es el hilo. Un kernel de CUDA se ejecuta mediante un conjunto de hilos los cuales ejecutan el mismo código. Los hilos están organizados en bloques (*blocks*) y los bloques a su vez se organizan en mallas (*grids*). Una malla puede ejecutar un kernel.

Cada hilo tiene un identificador que se usa para direccionar la memoria y tomar decisiones. Este índice puede ser de 1, 2 o 3 dimensiones. Los hilos pueden cooperar entre si mediante el uso de la memoria compartida dentro del bloque y sincronizando su ejecución para coordinar los accesos a la memoria.

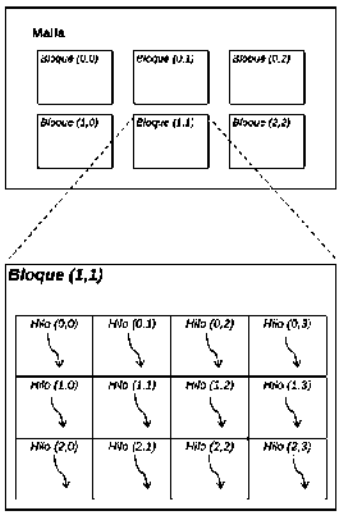


Figura 3.3: Hilos, bloques y mallas en CUDA

Las mallas pueden ser de 1, 2 o 3 dimensiones. Cada bloque es asignado a un multiprocesador y puede tener como máximo 512 hilos. Los hilos de un bloque se ejecutan en grupos de 32 hilos, cada grupo se denomina *warp* [Laguna-Sánchez *et al.*, 2011]. La organización de

una malla se determina durante el lanzamiento del kernel. El primer parámetro especifica las dimensiones de la malla y el segundo las dimensiones de un bloque en función de los hilos que contiene. Como se muestra a continuación:

$$kernel <<< dimGrid, dimBlock >>>$$

3.4. Aceleración y eficiencia

Para medir las ventajas de la utilización de más de un procesador en la ejecución de un programa se utilizan métricas como la eficiencia y la aceleración. La aceleración es la mejora en el tiempo de ejecución de un programa que utiliza más de un procesador en comparación con su contraparte secuencial. La aceleración ψ de un programa de tamaño k utilizando p procesadores se define como el cociente del tiempo de ejecución utilizando un procesador y el tiempo de ejecución utilizando p procesadores.

$$\psi(k,p) = \frac{T(k,1)}{T(k,p)} \tag{3.1}$$

donde $T(k,1)$ es el tiempo de ejecución del algoritmo secuencial y $T(k,p)$ es el tiempo correspondiente al algoritmo paralelo. Una aceleración mayor a 1 se considera favorable, ya que significa que el tiempo de ejecución fue optimizado con el uso de más de un procesador [Almeida *et al.*, 2008].

La eficiencia representa la porción de tiempo que los procesadores dedican a tareas útiles. Se calcula dividiendo la aceleración entre el número de procesadores que se utilizan en la ejecución del algoritmo.

$$E(k,p) = \frac{\psi(k,p)}{k} \tag{3.2}$$

3.5. Ley de Amhdal

La ley de Amhdal establece que la aceleración estará limitada por la parte secuencial del programa. Si e es la parte ejecutada secuencialmente, donde e toma valores entre 0 y 1, la máxima aceleración alcanzable utilizando p procesadores es [Aguilar *et al.*, 2004]:

$$\psi = \frac{1}{e + \frac{(1-e)}{p}} \tag{3.3}$$

Cuando $p \to \infty$, $\psi = 1/e$, por lo que entre mayor sea el valor de e la aceleración disminuirá proporcionalmente. Por ejemplo, si en un programa la parte secuencial representa el 25 %, la máxima aceleración alcanzable es 4.

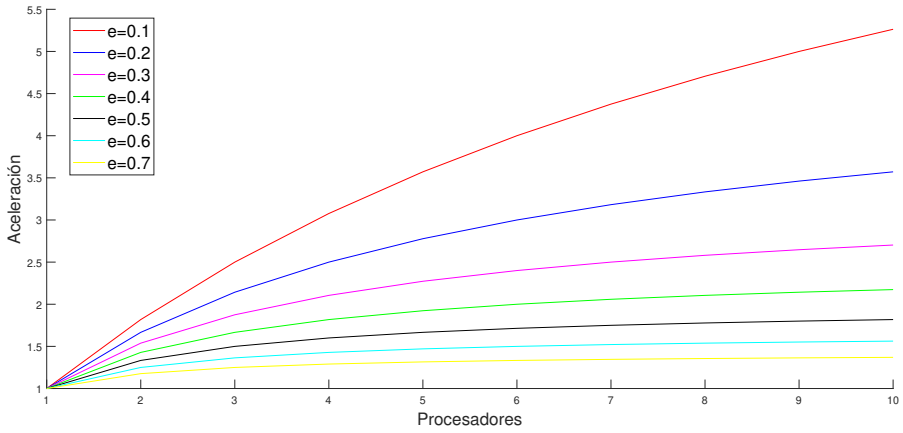


Figura 3.4: Ley de Amdahl

En un programa paralelo, existe un punto en el que la aceleración será la misma, aunque aumente el número de procesadores a utilizar, por lo que la aceleración dependerá de las características del algoritmo y no del número de procesadores. En la Figura 3.4 se muestra el comportamiento de la aceleración con respecto al número de procesadores y a diferentes valores de e. Conforme e aumenta se observa que la aceleración disminuye. El valor de aceleración más grande se obtiene con un valor de e =0.1 es decir, que solo el 10 % de las operaciones se llevan a cabo de forma secuencial.

3.6. Ley de Gustafson-Barsis

La ley de Amdahl demuestra que incrementar el número de procesadores reduce el tiempo de ejecución, pero esta ley considera un tamaño fijo del problema. En el enfoque de la ley de Gustafson-Barsis se establece que otro punto fuerte de la programación en paralelo, además de disminuir el tiempo de ejecución, es la precisión. A diferencia de la ley Amdahl establece que lo que debe quedar fijo es el tiempo de ejecución y que al aumentar la potencia

de cómputo se puede aumentar la precisión con la que se resuelve un problema. En aplicaciones donde la precisión es importante se desea que un problema grande sea resuelto en una máquina con una mayor potencia de cómputo, en el mismo tiempo que una máquina con características inferiores resolvería un problema pequeño.

El tiempo total de ejecución de un programa que utiliza más de un procesador se puede representar como la suma de la parte serial del programa a y la fracción que se ejecuta en paralelo b.

$$a(k) + b(k) = 1 \tag{3.4}$$

La suposición de la ley de Gustafson-Barsis es que el trabajo a realizar en la parte paralela varía linealmente con el número de procesadores utilizados. A partir de esta afirmación la ecuación (3.1) se puede representar como:

$$\psi = a(k) + pb(k) \tag{3.5}$$

Si $a = e$ y $b = 1 - e$, siendo e la parte serial del programa, la ecuación (3.5) se reescribe como

$$\psi = e + (1 - e)p \tag{3.6}$$

En la Figura 3.5 se observa el comportamiento de la aceleración con respecto al número de procesadores y la parte serial del algoritmo. A diferencia de la ley de Amdahl la aceleración sigue un comportamiento lineal.

3.7. Parte secuencial de un programa paralelo

En un programa paralelo siempre existirá una parte que solo puede ejecutarse secuencialmente; esta parte permanece constante independientemente del número de procesadores que se utilicen para la ejecución del programa. La métrica de Karp-Flatt permite calcular la parte secuencial de un programa paralelo [Quinn, 2003].

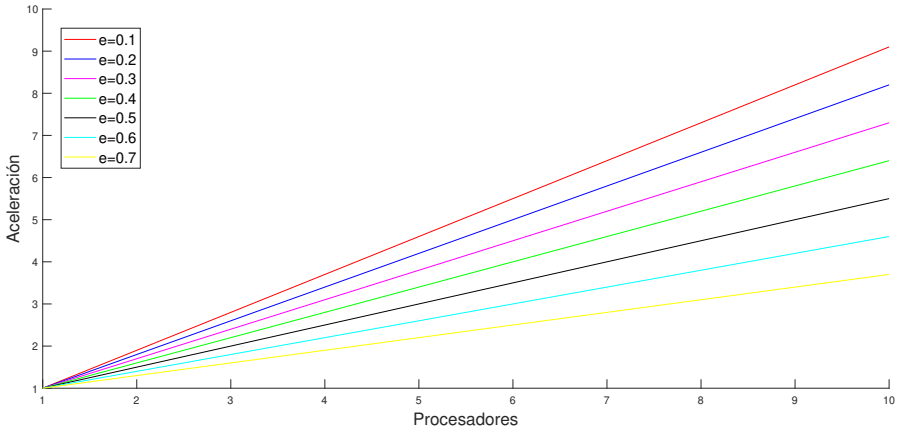


Figura 3.5: Ley de Gustafson-Barsis

El tiempo total de ejecución de un programa de tamaño k utilizando p procesadores se representa con la ecuación (3.7).

$$T(k,p) = T(k,1)e + \frac{T(k,1)(1-e)}{p} \tag{3.7}$$

donde

e es la parte secuencial del programa.

$1-e$ es la parte paralela del programa.

Sustituyendo (3.7) en (3.1), la aceleración de un problema de tamaño k con p procesadores es,

$$\psi(k,p) = \frac{T(k,1)}{T(k,1)e + \frac{T(k,1)(1-e)}{p}} \tag{3.8}$$

Simplificando la ecuación (3.8)

$$\psi(k,p) = \frac{1}{e + \frac{1-e}{p}} \tag{3.9}$$

Resolviendo la ecuación (3.9), la parte secuencial de un programa se puede calcular con la ecuación (3.10).

$$e = \frac{\frac{1}{\psi} - \frac{1}{p}}{1 - \frac{1}{p}} \tag{3.10}$$

3.8. Conclusiones

Las ventajas de aplicar la programación en paralelo están relacionadas con el tamaño del problema a resolver, es decir, si en un programa hay una gran cantidad de datos y operaciones que realizar, el uso de esta herramienta puede convertirse en una gran ventaja, porque permitirá agilizar su ejecución y reducir considerablemente el esfuerzo de cómputo, pero si el programa contiene operaciones que no representan ningún costo para una computadora, el utilizar el procesamiento en paralelo se convierte en una desventaja ya que puede aumentar el tiempo en la ejecución de un programa.

Capítulo 4

Propuesta de paralelización del algoritmo de estimación de dos etapas

4.1. Introducción

En la Figura 4.1 se muestra el algoritmo utilizado en el desarrollo de este trabajo. El algoritmo comienza con la lectura de los datos del sistema y las mediciones SCADA, una vez que se cumple el criterio de que $max|\Delta x| < \varepsilon$, donde ε es la tolerancia establecida, el vector de estados x_1 y las mediciones PMU pasan como entrada a la segunda etapa de estimación, donde el proceso se lleva a cabo nuevamente y da como resultado el vector de estados x_2.

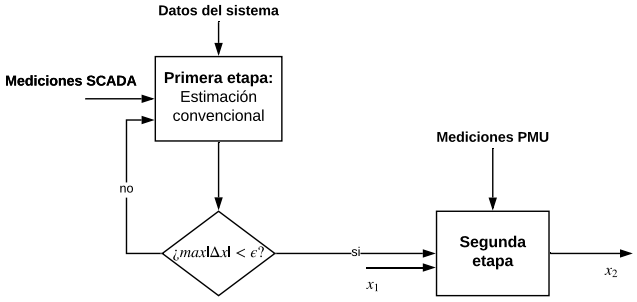


Figura 4.1: Algoritmo de estimación de estado de dos etapas

4.2. Esquema de paralelización del método de estimación de dos etapas

En la Figura 4.2 se muestran los bloques de operaciones en cada etapa de estimador y se indica en donde se calcula cada bloque de operaciones. A continuación, se describen las operaciones realizadas en cada etapa.

Primera etapa:

1. El algoritmo comienza con la lectura de las mediciones utilizadas en la primera etapa y el cálculo de la matriz de covarianza R.

2. Se leen los datos del sistema como: nodo envío, nodo recepción, resistencia, reactancia y tap, a partir de los cuales se forma la matriz de admitancia Y_{bus}.

3. Se leen las susceptancias de cada nodo y se añaden a la diagonal de la matriz Y_{bus}.

4. Se inicializa el vector de estados x y los vectores e y f que contienen a parte real e imaginaria de los voltajes complejos en cada nodo respectivamente.

5. Se calcula la función de mediciones h.

6. Se calcula el Jacobiano H.

7. Se calcula el vector de residuos r, el cual es la diferencia entre el vector de mediciones z y la función de mediciones h.

8. Obtener la transpuesta del Jacobiano H.

9. Calcular la matriz de Ganancia G y la parte izquierda del sistema de ecuaciones a resolver en cada iteración.

10. Δx se obtiene resolviendo el sistema de ecuaciones $G\Delta x = g$ donde G es la matriz de ganancia y g la parte izquierda del sistema de ecuaciones.

11. Una vez que se obtienen la matriz G y g el sistema de ecuaciones se resuelve mediante la factorización LU y la sustitución hacia delante y hacia atrás. Mediante el uso de cuSOLVER, este bloque de operaciones se realiza en la GPU.

12. Se obtiene el error. El error se calcula a partir del absoluto del máximo incremento de una de las variables de estado.

13. Comparar el error con la tolerancia de convergencia establecida. Si el valor del error es menor a la tolerancia el proceso termina en caso contrario se actualiza el vector x.

Segunda etapa:

14. La segunda etapa inicia con la lectura de las mediciones PMU.

15. Pasar de polar a rectangular las mediciones PMU.

16. Calcular la matriz T de rotación.

17. Pasar de polar a rectangular la matriz R_{pmu}.

18. Calcular la inversa de la matriz R_{pmu}.

19. Calcular la matriz H_2.

20. Calcular el vector z_2.

21. Obtener la matriz R_2.

22. Calcular las matrices G_2 y el vector g_2.

23. El vector de estados x_2 se calcula resolviendo el sistema de ecuaciones mediante la factorización LU y la sustitución hacia delante y hacia atrás. Este bloque de operaciones se realiza en la GPU.

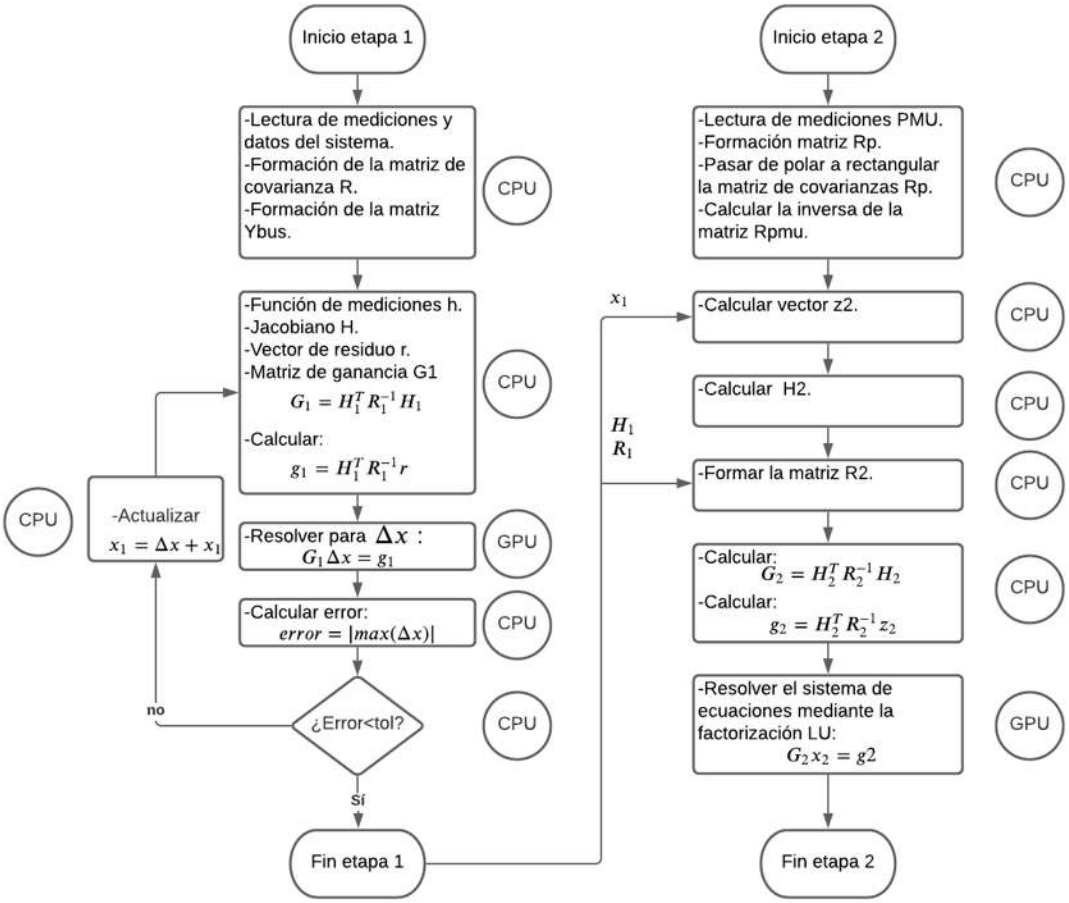


Figura 4.2: Diagrama de flujo para la paralelización del método de estimación de dos etapas

En esta propuesta se eligió la factorización LU debido a los buenos resultados obtenidos en otras propuestas enfocadas a la paralelización del algoritmo de estimación de estado convencional además de tener ventajas como un almacenamiento compacto ya que las matrices L y U se pueden guardar en una matriz y su uso se recomienda en matrices dispersas [Poole, 2011].

La factorización LU y la sustitución hacia delante y hacia atrás representan la mayor parte del tiempo de ejecución utilizando la formulación tradicional. En la primera etapa, la factorización LU representa aproximadamente el 44 % y resolver el sistema de ecuaciones linealizado utilizando la sustitución hacia delante y hacia atrás representa aproximadamente el 15 % [Karimipour y Dinavahi, 2013]. En la plataforma CUDA, la librería cuSOLVER tiene funciones para resolver sistemas $Ax = b$ utilizando factorización LU y sustitución hacia

delante y hacia atrás.

Tanto en la primera como en la segunda etapa se utilizan la factorización LU y la sustitución hacia delante y hacia atrás. En la primera etapa este bloque de operaciones se realiza para calcular Δx en cada iteración y en la segunda se realiza para encontrar los estados del vector x_2. En el Apéndice B, se muestra el código utilizado en el algoritmo propuesto y que se ejecuta en la GPU en cada etapa del estimador. Esta sección del código utiliza **getrf** para realizar la factorización LU y **getrs** para realizar la sustitución hacia delante y hacia atrás.

4.3. Conclusiones

En [Xia *et al.*, 2017] y [Karimipour y Dinavahi, 2013] se han reportado dos propuestas para reducir el tiempo de cómputo de los algoritmos de estimación de estado estáticos utilizando cómputo en paralelo basado en GPU. En estos trabajos de investigación solo se aborda la estimación utilizando mediciones SCADA, debido a lo cual, la propuesta no se apega a la situación real de la mayoría de los sistemas eléctricos actuales. Otra propuesta revisada en esta tesis y que está enfocada en reducir el tiempo de cómputo del algoritmo de estimación estático es la presentada en [Dobakhshari *et al.*, 2019], a pesar de ofrecer buenos resultados, nuevamente es una propuesta que no considera la inclusión de las mediciones PMU.

En este capítulo se presenta la propuesta de un algoritmo de estimación de estado estático en dos etapas donde la factorización LU y la sustitución hacia delante y hacia atrás se resuelven en la GPU utilizando la plataforma CUDA y la biblioteca cuSOLVER con el fin de reducir el tiempo de cómputo y el esfuerzo computacional. Ambas operaciones representan aproximadamente el 59 % del tiempo de cómputo de la primera etapa. De manera que, son las dos operaciones que más repercusión tienen en el tiempo de ejecución. Entre las ventajas de esta propuesta a nivel *hardware*, se encuentra que puede ser ejecutada en cualquier equipo de cómputo que tenga una tarjeta gráfica.

Capítulo 5

Casos de Estudio

5.1. Introducción

Se consideraron 5 casos de estudio de 14, 30, 118, 300, 500 y 1354 nodos. El formato de los archivos de entrada al estimador se muestra en el Apéndice A. El algoritmo fue escrito en lenguaje C y ejecutado en una estación de trabajo Alienware utilizando el sistema operativo Ubuntu/Linux. El procesador del equipo es Intel core i5-3450 con una velocidad de proceso de 3GHz. En la Figura 5.1 se observa la estación de trabajo utilizada para la ejecución de los casos de estudio de esta sección.

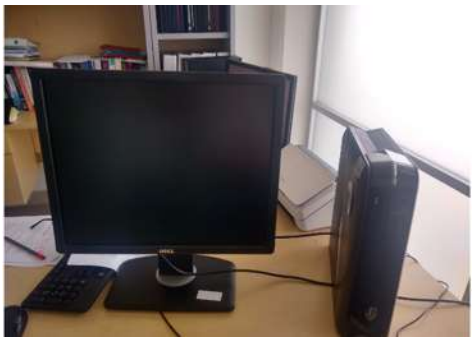


Figura 5.1: Equipo utilizado para las pruebas de procesamiento en paralelo

La GPU utilizada en este trabajo se puede observar en la Figura 5.2, es una tarjeta GeForce GTX 660 OEM de NVIDIA basada en el procesador de gráficos GK104. En la Tabla 5.1 se muestran sus principales características.

Figura 5.2: GeForce GTX 660 OEM

Tabla 5.1: Características de la GPU GeForce GTX 660 OEM

Núcleos CUDA	1152
Ancho de banda de memoria	134 GB/s
Memoria	2GB
Rendimiento pico	2.05 Tflops

5.2. Casos de estudio considerando mediciones sin errores

En la primera sección de este capítulo, las mediciones SCADA y PMU se consideraron sin errores. En cada uno de los casos el sistema a resolver es sobre determinado ya que el número de mediciones es mayor al número de estados. Para todos los casos de estudio considerados, la covarianza utilizada en la primera etapa es $1x10^{-4}$ y para la segunda es $1x10^{-6}$. La covarianza de las mediciones utilizadas en la primera etapa fue asignada por el programa *Power Education Toolbox* de donde se obtuvieron las mediciones. La covarianza utilizada en la segunda etapa se asigna con un valor mayor debido a que las mediciones PMU se consideran más precisas. Se consideraron los sistemas de 14, 30, 118, 500 y 1354 para demostrar la funcionalidad del algoritmo.

5.2.1. Sistema IEEE 14 nodos

Para este caso de estudio, se utilizaron 32 mediciones para la primera etapa y 10 mediciones en la segunda etapa del estimador. El total de PMUs ubicadas en el sistema son 3. Las PMUs se encuentran en los nodos 4, 7 y 14. La primera etapa del estimador se resuelve en 7 iteraciones. En la Tabla 5.2 se realiza la comparación entre los resultados obtenidos en

cada etapa del estimador y los resultados obtenidos en el estudio de flujos de potencia.

Tabla 5.2: Comparación entre los resultados obtenidos de flujos de potencia y los resultados obtenidos en cada etapa del estimador para el sistema IEEE de 14 nodos

Nodo	**Flujos de potencia V(p.u)**	**Etapa 1 V(p.u)**	**Etapa 2 V(p.u)**
1*	1.0600 ∠ 0.000°	1.0600 ∠ 0.000°	1.0600 ∠ 0.000°
2	1.0450 ∠ -4.980°	1.0450∠ -4.981°	1.0450 ∠ -4.977°
3	1.0100 ∠ -12.720°	1.0100 ∠ -12.718°	1.0100 ∠ -12.714°
4	1.0186 ∠ -10.320°	1.0186 ∠ -10.324°	1.0186 ∠ -10.321°
5	1.0203 ∠ -8.780°	1.0203 ∠ -8.783°	1.0203 ∠ -8.780°
6	1.0700 ∠ -14.220°	1.0698 ∠ -14.223°	1.0700 ∠ -14.225°
7	1.0620∠ -13.370°	1.0619 ∠ -13.368°	1.0620∠-13.369°
8	1.0900 ∠ -13.370°	1.0900 ∠ -13.368°	1.0900 ∠ -13.369°
9	1.0563 ∠ -14.950°	1.0561 ∠ -14.947°	1.0563 ∠-14.946°
10	1.0513 ∠ -15.100°	1.0511∠ -15.105°	1.0513∠ -15.104°
11	1.0571 ∠ -14.790°	1.0568 ∠ -14.796°	1.0571 ∠-14.796°
12	1.0552 ∠ -15.080°	1.0550 ∠ -15.078°	1.0552 ∠-15.079°
13	1.0505 ∠ -15.160°	1.0502 ∠-15.160°	1.0504 ∠-15.161°
14	1.0358 ∠ -16.040°	1.0356 ∠ -16.040°	1.0358 ∠-16.040°

* Nodo *slack*

El resultado en la segunda etapa en algunos nodos se mejora, coincide o está más cercanos que lo obtenido en la primera etapa en comparación con el estudio de flujos de potencia. Tal como se observa en los resultados a partir del nodo 6. Lo mencionado anteriormente, se puede volver a confirmar con lo que se muestra en la Tabla 5.3. Las magnitudes de voltaje obtenidas en la segunda etapa presentan un error menor.

Tabla 5.3: MSE de cada etapa para sistema IEEE de 14 nodos

Etapa	**MSE magnitud de voltaje**	**MSE ángulos**
1	$2.7445\text{x}10^{-8}$	$7.9834\text{x}10^{-6}$
2	$7.4948\text{x}10^{-10}$	$1.1426\text{x}10^{-5}$

5.2.2. Sistema IEEE 30 nodos

Para el sistema IEEE de 30 nodos y 60 estados, se utilizaron 78 mediciones para la primera etapa y 10 mediciones en la segunda etapa del estimador. El total de PMUs en el sistema son 6. Las PMUs se encuentran ubicados en los nodos 8, 9, 12, 24, 25 y 26. En la Figura

5.3 se comparan los resultados del estudio de flujos de potencia y los resultados obtenidos por el estimador de estado de dos etapas. Debido a que las mediciones no contienen errores, los resultados obtenidos del estimador coinciden con los resultados del estudio de flujos de potencia.

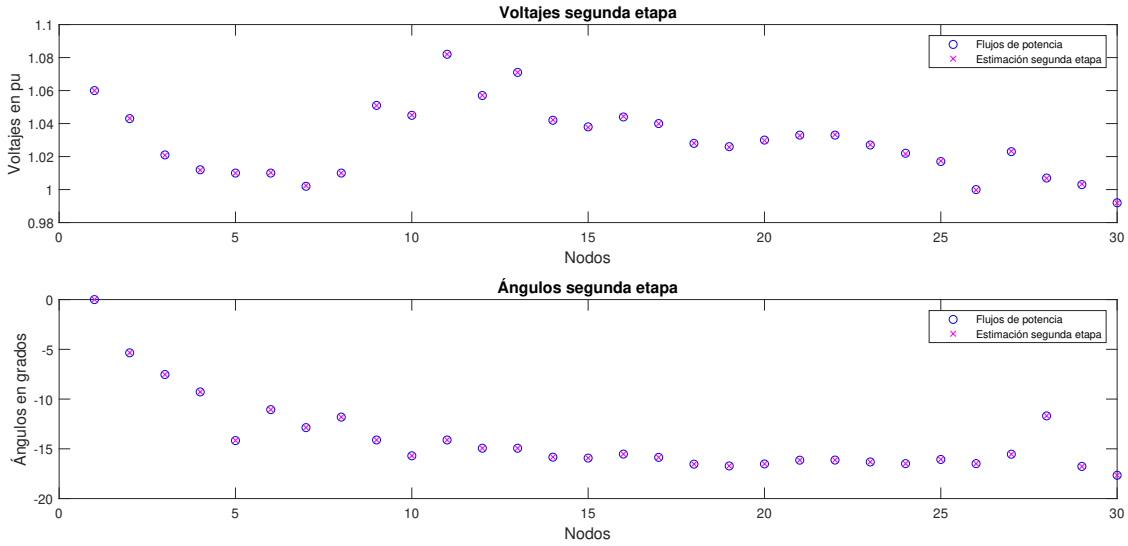


Figura 5.3: Voltajes y ángulos del sistema IEEE 30 nodos

Tabla 5.4: MSE de cada etapa para sistema IEEE de 30 nodos

Etapa	MSE magnitud de voltaje	MSE ángulos
1	$5.8626\text{x}10^{-8}$	$1.1089\text{x}10^{-5}$
2	$5.5541\text{x}10^{-8}$	$2.2805\text{x}10^{-5}$

En la Tabla 5.4 se muestra el MSE para cada etapa. Se muestra que, en los resultados de la segunda etapa, la magnitud de voltaje presenta un error menor en comparación a la primera etapa.

5.2.3. Sistema IEEE 118 nodos

Para el sistema IEEE de 118 nodos y 236 estados, se utilizaron 304 mediciones para la primera etapa del estimador. El total de PMUs ubicados en el sistema es 19 de los cuales se obtuvieron 19 mediciones de fasores de voltaje y 15 mediciones de fasores de corriente. La primera etapa del estimador converge en 7 iteraciones. Los resultados obtenidos de la segunda

etapa se muestran de forma comparativa en la Figura 5.4.

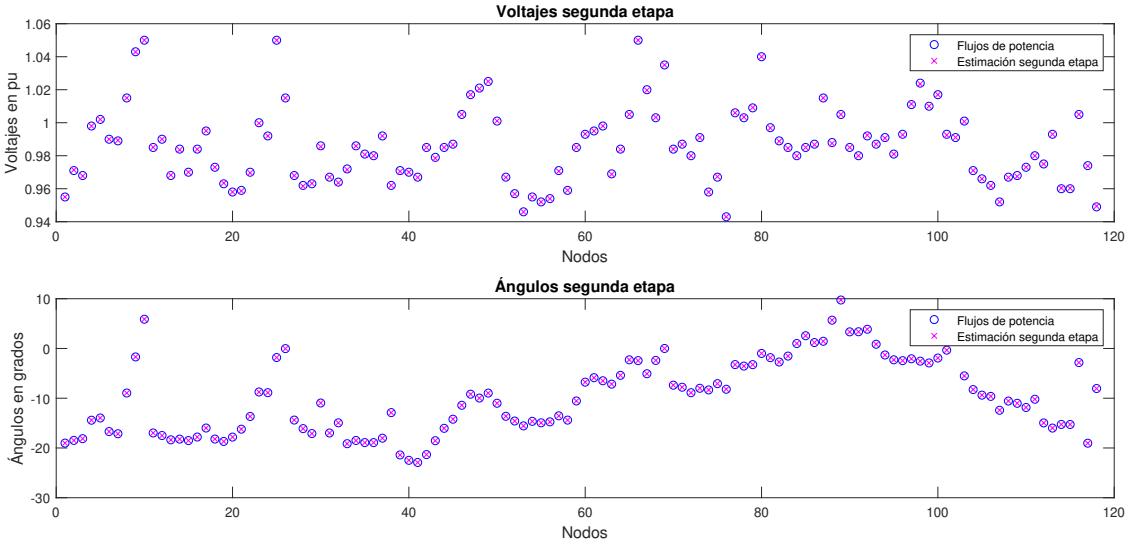


Figura 5.4: Magnitudes de voltajes y ángulos del sistema IEEE 118 nodos

Tabla 5.5: MSE de cada etapa para sistema de 118 nodos

Etapa	MSE magnitud de voltaje	MSE ángulos
1	$1.6907\text{x}10^{-9}$	$1.4064\text{x}10^{-5}$
2	$4.4386\text{x}10^{-10}$	$1.0843\text{x}10^{-5}$

En la Tabla 5.5 se muestra el MSE obtenido en cada etapa para el sistema IEEE de 118 nodos. Para este caso de estudio, los resultados en la segunda etapa tanto en magnitudes de voltaje y ángulos tienen un error menor en comparación con los resultados de la primera etapa.

5.3. Tiempo de ejecución

En esta segunda sección se calculan las aceleraciones para cada caso de estudio y se muestra un resumen de los casos de estudio considerados en el capítulo. Para los sistemas de 300, 500 y 1354 se utilizaron los datos y resultados de flujos de potencia utilizando MATPOWER. En las Tablas 5.6 y 5.7 se presenta un resumen del tamaño de las matrices y número

de mediciones utilizadas en cada etapa, respectivamente.

Tabla 5.6: Resumen casos de estudio primera etapa

Caso	**Nodos**	**No. mediciones**	**Estados**	**Jacobiano** H	**Matriz de Ganancia** G
1	14	32	28	32x28	28x28
2	30	78	60	78x60	60x60
3	118	304	236	304x236	236x236
4	300	1638	600	1638x600	600x600
5	500	2338	1000	2338x1000	1000x1000
6	1354	6842	2708	6842x2708	2708x2708

Tabla 5.7: Resumen casos de estudio segunda etapa

Caso	**Nodos**	**No. mediciones**	G_2	H_2	R_2
1	14	10	28x28	38x28	38x38
2	30	22	60x60	82x60	82x82
3	118	68	236x236	304x236	304x304
4	300	302	600x600	902x600	902x902
5	500	692	1000x1000	1692x1000	1692x1692
6	1354	1354	2708x2708	4062x2708	4062x4062

En la Tabla 5.8 se muestran las aceleraciones en cada caso de estudio. La comparación se realizó utilizando el algoritmo programado secuencialmente y el algoritmo propuesto utilizando cuSOLVER en el bloque de operaciones que involucra la factorización LU y la sustitución hacia delante y hacia atrás. A partir de los resultados obtenidos se observa que la aceleración tiende a crecer conforme el tamaño del sistema aumenta.

Tabla 5.8: Aceleración obtenida en cada caso de estudio

Caso	**Nodos**	**Aceleración**
1	14	0.0087
2	30	0.0358
3	118	0.9055
4	300	1.0738
5	500	1.9900
6	1354	5.7400

En la Figura 5.5 se observa que para los primeros cuatro casos el programa paralelo es más lento que su contraparte secuencial, lo cual también se observa en la aceleración de cada caso. Es a partir del caso de estudio del sistema IEEE de 300 nodos donde el algoritmo paralelo obtiene su mejor aceleración.

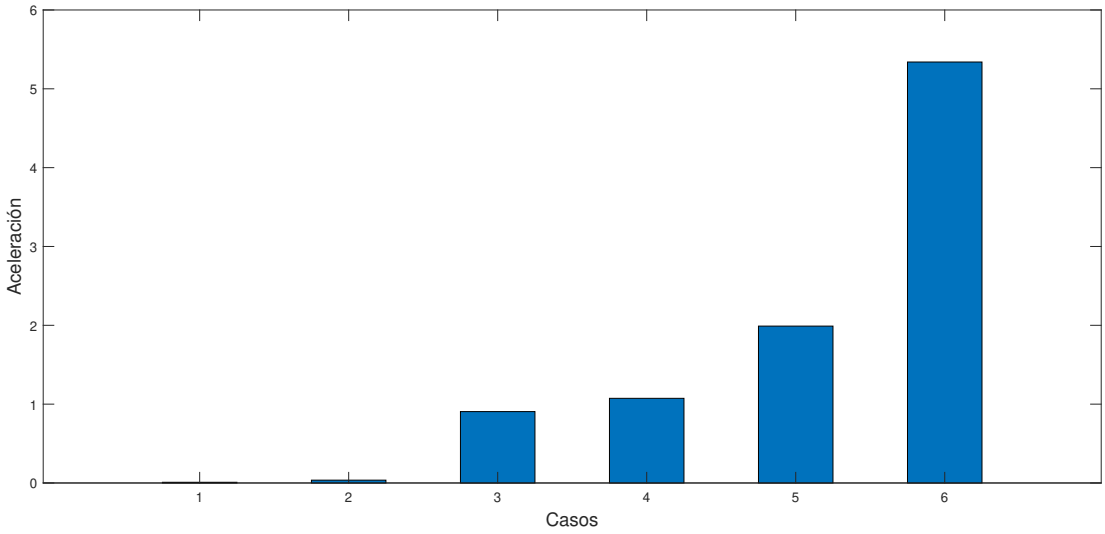


Figura 5.5: Aceleración para cada caso de estudio

5.4. Conclusiones

El objetivo de este capítulo es la comparación del algoritmo propuesto y el algoritmo programado de forma secuencial mediante el cálculo de la aceleración, para cuantificar la mejora en el tiempo de ejecución. A partir de las aceleraciones obtenidas, se mostró que el algoritmo propuesto es aproximadamente 6 veces más rápido que el algoritmo programado de forma secuencial. Debido a que sólo se utilizaron mediciones sin errores, la precisión en los resultados obtenidos en la segunda etapa se puede observar a partir del tercer y cuarto dígito.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones generales

El algoritmo paralelo de estimación de dos etapas presentado en esta tesis, utilizó mediciones SCADA y PMU en dos etapas distintas. El conjunto de mediciones SCADA, considerado en este trabajo, incluyen mediciones de inyecciones nodales de potencia activa y reactiva, magnitudes de voltaje y flujos de potencia a través de líneas de transmisión y transformadores del sistema. El conjunto de mediciones PMU considera mediciones fasoriales de voltaje y corriente en forma polar. Las mediciones consideradas tanto SCADA como PMU fueron obtenidas de un programa de flujos de potencia.

Se mostró la funcionalidad del algoritmo a través de los sistemas de prueba IEEE de 14, 30, 118, 500 y 1354 nodos. Para los primeros 3 casos de estudio presentados en esta tesis, se observó que el MSE obtenido en la segunda etapa es menor que en el estimador convencional, ya que las mediciones PMU aumentan la precisión con la que se obtienen los estados de un sistema.

Se utilizó la librería cuSOLVER para eficientar el proceso de estimación de estado. En el algoritmo propuesto, el bloque que representa la mayor parte del tiempo de ejecución es la factorización LU y la sustitución hacia delante y hacia atrás. Por lo que, las operaciones de este bloque se llevan a cabo en la GPU.

Se realizaron comparaciones entre el algoritmo propuesto y su contraparte secuencial

a través de métricas como la aceleración para evaluar los beneficios de la utilización del procesamiento en paralelo. Es a partir del sistema de 300 nodos donde el algoritmo propuesto comenzó a obtener aceleraciones favorables, obteniéndose un factor de aceleración de aproximadamente 6 en el caso de 1354, lo cual confirma la ventaja de la paralelización.

Se mostró una alternativa de bajo costo para la paralelización del algoritmo de estimación de estado de dos etapas.

6.2. Recomendaciones para trabajos futuros

Para dar continuidad al trabajo de investigación reportado en esta tesis, se recomienda proceder en las siguientes vertientes de investigación:

- Implementar en el algoritmo la inclusión de otros dispositivos presentes en los sistemas eléctricos como los dispositivos FACTS.
- Incluir en el algoritmo la detección e identificación de mediciones erróneas tanto en las mediciones SCADA como en las mediciones PMU.
- Implementar una interfaz gráfica para la interacción con el usuario.

Apéndice A

Archivos de entrada al estimador

Archivo de mediciones primera etapa

En la primera línea del archivo se indica el número de mediciones que se utilizan en la primera etapa.

Tabla A.1: Formato del archivo de mediciones primera etapa

Columna	Valor
1	Tipo de medición 0-Magnitud de voltaje 1-Ángulo medición de voltaje 2-Inyección de potencia real 3-Inyección de potencia reactiva 4-Flujo de potencia real 5-Flujo de potencia reactiva
2	Nodo envío
3	Nodo recepción
4	Magnitud
5	Covarianza

Archivo de datos del sistema

La primera línea del archivo tiene el siguiente formato:

5 5 1

donde:

5 es el número de nodos.

5 es el número de ramas.

1 es el nodo de referencia.

Tabla A.2: Formato del archivo de datos del sistema

Columna	Valor
1	Nodo envío
2	Nodo recepción
3	Resistencia (p.u)
4	Reactancia (p.u)
5	B/2 (p.u)
6	Tap

Susceptancia añadida a cada nodo

Tabla A.3: Formato del archivo susceptancia

Columna	Valor
1	Susceptancia en p.u

Archivo de mediciones PMU

La primera línea del archivo tiene el siguiente formato:

197 101 96

donde:

197 número total de mediciones provenientes de las PMUs.
101 número de mediciones de voltaje.
96 número de mediciones de flujo de corriente.

Tabla A.4: Formato del archivo de mediciones PMU

Columna	Valor
1	Tipo de medición 1-Medición de magnitud de voltaje 2-Medición de angulo de voltaje 3-Medición de magnitud de voltaje 4-Medición de angulo de voltaje
2	Nodo envío
3	Nodo recepción
4	Magnitud
5	Covarianza

Apéndice B

Código ejecutado en la GPU

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <cuda_runtime.h>
#include <cusolverDn.h>

//Funcion que imprime las matrices
void printMatrix(int m, int n, const double*A, int lda, const char* name)
{
 for(int row = 0 ; row < m ; row++){
 for(int col = 0 ; col < n ; col++){
 double Areg = A[row + col*lda];
 printf("%s(%d,%d) = %f\n", name, row+1, col+1, Areg);

 }
 }
}
//Programa principal
int main(int argc, char*argv[])
{
 cusolverDnHandle_t cusolverH = NULL;
 cudaStream_t stream = NULL;
 cusolverStatus_t status = CUSOLVER_STATUS_SUCCESS;
 cudaError_t cudaStat1 = cudaSuccess;
 cudaError_t cudaStat2 = cudaSuccess;
 cudaError_t cudaStat3 = cudaSuccess;
 cudaError_t cudaStat4 = cudaSuccess;
 const int m = 3;
 const int lda = m;
 const int ldb = m;
 \\Declaracion y formacion de A,B,LU y X.
 double A[lda*m] = {};
 double B[m] = { };
 double X[m]; /* X = A\B */
 double LU[lda*m]; /* L y U */
 int Ipiv[m];
 int info = 0;
 double *d_A = NULL; /*Copia de A en el device
 double *d_B = NULL; /* Copia de B en el device */
 int *d_Ipiv = NULL;
 int *d_info = NULL;
 int lwork = 0;
```

```
 double *d_work = NULL;
 const int pivot_on = 0;

 status = cusolverDnCreate(&cusolverH);
 assert(CUSOLVER_STATUS_SUCCESS == status);
 cudaStat1 = cudaStreamCreateWithFlags(&stream, cudaStreamNonBlocking);
 assert(cudaSuccess == cudaStat1);
 status = cusolverDnSetStream(cusolverH, stream);
 assert(CUSOLVER_STATUS_SUCCESS == status);
/*Copiar A al device*/
 cudaStat1 = cudaMalloc ((void**)&d_A, sizeof(double) * lda * m);
 cudaStat2 = cudaMalloc ((void**)&d_B, sizeof(double) * m);
 cudaStat2 = cudaMalloc ((void**)&d_Ipiv, sizeof(int) * m);
 cudaStat4 = cudaMalloc ((void**)&d_info, sizeof(int));
 assert(cudaSuccess == cudaStat1);
 assert(cudaSuccess == cudaStat2);
 assert(cudaSuccess == cudaStat3);
 assert(cudaSuccess == cudaStat4);
 cudaStat1 = cudaMemcpy(d_A, A, sizeof(double)*lda*m,
 cudaMemcpyHostToDevice);
 cudaStat2 = cudaMemcpy(d_B, B, sizeof(double)*m, cudaMemcpyHostToDevice);
 assert(cudaSuccess == cudaStat1);
 assert(cudaSuccess == cudaStat2);
 /*Consulta el espacio de trabajo de getrf */
 status = cusolverDnDgetrf_bufferSize(
 cusolverH,
 m,
 m,
 d_A,
 lda,
 &lwork);
 assert(CUSOLVER_STATUS_SUCCESS == status);
 cudaStat1 = cudaMalloc((void**)&d_work, sizeof(double)*lwork);
 assert(cudaSuccess == cudaStat1);
/*Se realiza la factorizacion LU */
 if (pivot_on){
 status = cusolverDnDgetrf(
 cusolverH,
 m,
 m,
 d_A,
 lda,
 d_work,
 d_Ipiv,
 d_info);
 }else{
 status = cusolverDnDgetrf(
 cusolverH,
 m,
 m,
 d_A,
 lda,
 d_work,
 NULL,
 d_info);
 }
 cudaStat1 = cudaDeviceSynchronize();
 assert(CUSOLVER_STATUS_SUCCESS == status);
 assert(cudaSuccess == cudaStat1);
```

```
 if (pivot_on){
 cudaStat1 = cudaMemcpy(Ipiv , d_Ipiv, sizeof(int)*m,
 cudaMemcpyDeviceToHost);
 }
 cudaStat2 = cudaMemcpy(LU , d_A , sizeof(double)*lda*m,
 cudaMemcpyDeviceToHost);
 cudaStat3 = cudaMemcpy(&info, d_info, sizeof(int), cudaMemcpyDeviceToHost);
 assert(cudaSuccess == cudaStat1);
 assert(cudaSuccess == cudaStat2);
 assert(cudaSuccess == cudaStat3);

 /*Resuelve A*x = B
 if (pivot_on){
 status = cusolverDnDgetrs(
 cusolverH,
 CUBLAS_OP_N,
 m,
 1, /* nrhs */
 d_A,
 lda,
 d_Ipiv,
 d_B,
 ldb,
 d_info);
 }else{
 status = cusolverDnDgetrs(
 cusolverH,
 CUBLAS_OP_N,
 m,
 1,
 d_A,
 lda,
 NULL,
 d_B,
 ldb,
 d_info);
 }
 cudaStat1 = cudaDeviceSynchronize();
 assert(CUSOLVER_STATUS_SUCCESS == status);
 assert(cudaSuccess == cudaStat1);
 cudaStat1 = cudaMemcpy(X , d_B, sizeof(double)*m, cudaMemcpyDeviceToHost);
 assert(cudaSuccess == cudaStat1);
/*Libera memoria */
 if (d_A ) cudaFree(d_A);
 if (d_B ) cudaFree(d_B);
 if (d_Ipiv ) cudaFree(d_Ipiv);
 if (d_info ) cudaFree(d_info);
 if (d_work ) cudaFree(d_work);
 if (cusolverH ) cusolverDnDestroy(cusolverH);
 if (stream ) cudaStreamDestroy(stream);
 cudaDeviceReset();
 return 0;
}
```

Referencias

[Acosta *et al.*, 2012] Acosta, F. A., Segura, O. M., y Ospina, A. E. (2012). Guía y fundamentos de la programación en paralelo. *Revista en telecomunicaciones e informática*, 2(4):81–97. 6

[Aguilar *et al.*, 2004] Aguilar, J and Leiss, E and others (2004). Introducción a la computación paralela. *Editorial Venezolana, Universidad de Los Andes, Mérida*. 38

[Almeida *et al.*, 2008] Almeida, F., Giménez, D., Mantas, J. M., y Vidal, A. M. (2008). *Introducción a la programación paralela*. Thompson Paraninfo. 38

[Baltensperger *et al.*, 2010] Baltensperger, R., Loosli, A., Sauvain, H., Zima, M., Andersson, G., y Nuqui, R. (2010). An implementation of two-stage hybrid state estimation with limited number of pmu. 3, 33

[Barlas, 2014] Barlas, G. (2014). *Multicore and GPU Programming: An integrated approach*. Elsevier. 6

[Chatterjee *et al.*, 2015] Chatterjee, P., Pal, A., Thorp, J. S., y De La Ree, J. (2015). Partitioned linear state estimation. En *2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5. IEEE. 3

[Cisneros-Magaña *et al.*, 2018] Cisneros-Magaña, R., Medina, A., Dinavahi, V., y Ramos-Paz, A. (2018). Time-domain power quality state estimation based on kalman filter using parallel computing on graphics processing units. *IEEE Access*, 6:21152–21163. 7

[Cotronis, 2001] Cotronis, Y. (2001). *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 8th European PVM/MPI Users' Group Meeting, Santorini/Thera, Greece, September 23-26, 2001. Proceedings*, volumen 2131. Springer Science & Business Media. 6

[Dobakhshari *et al.*, 2019] Dobakhshari, A. S., Azizi, S., Paolone, M., y Terzija, V. (2019). Ultra fast linear state estimation utilizing scada measurements. *IEEE Transactions on Power Systems*. 3, 47

[Farzanehrafat y Watson, 2010] Farzanehrafat, A. y Watson, N. R. (2010). Review of power quality state estimation. En *2010 20th Australasian Universities Power Engineering Conference*, pp. 1–5. IEEE. 1

[Foster, 1995] Foster, I. (1995). *Designing and building parallel programs: concepts and tools for parallel software engineering*. Addison-Wesley Longman Publishing Co., Inc. 5

[Garcia, 2010] Garcia, N. (2010). Parallel power flow solutions using a biconjugate gradient algorithm and a newton method: A gpu-based approach. En *IEEE PES General Meeting*, pp. 1–4. 7

[Gomez-Exposito *et al.*, 2018] Gomez-Exposito, A., Conejo, A. J., y Cañizares, C. (2018). *Electric energy systems: analysis and operation*. CRC press. 12

[Gulati *et al.*, 2013] Gulati, A., Gulati, N., y Solanki, S. K. (2013). Overview of state estimation technique for power system control. *IOSR Journal of Electrical and Electronics Engineering*, 8:51–55. 1

[Huang *et al.*, 2012] Huang, Y.-F., Werner, S., Huang, J., Kashyap, N., y Gupta, V. (2012). State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid. *IEEE Signal Processing Magazine*, 29(5):33–43. 4

[Jalili-Marandi y Dinavahi, 2009] Jalili-Marandi, V. y Dinavahi, V. (2009). Large-scale transient stability simulation on graphics processing units. En *2009 IEEE Power & Energy Society General Meeting*, pp. 1–6. IEEE. 7

[James y Bindu, 2015] James, J. y Bindu, S. (2015). Hybrid state estimation including pmu measurements. En *2015 International Conference on Control Communication & Computing India (ICCC)*, pp. 309–313. IEEE. 4

[Jones, 2011] Jones, K. D. (2011). *Three-phase linear state estimation with phasor measurements*. Tesis doctoral, Virginia Tech. 22

[Karimipour y Dinavahi, 2013] Karimipour, H. y Dinavahi, V. (2013). Accelerated parallel wls state estimation for large-scale power systems on gpu. En *2013 North American Power Symposium (NAPS)*, pp. 1–6. IEEE. 2, 7, 46, 47

[Laguna-Sánchez *et al.*, 2011] Laguna-Sánchez, G. A., Olgun-Carbajal, M., y Barrón-Fernández, R. (2011). Introducción a la programación de códigos paralelos con cuda y su ejecución en un gpu multi-hilos. *ContactoS*, 80:65–69. 36, 37

[Li *et al.*, 2017] Li, X., Li, F., Yuan, H., Cui, H., y Hu, Q. (2017). Gpu-based fast decoupled power flow with preconditioned iterative solver and inexact newton method. En *2017 IEEE Power Energy Society General Meeting*, pp. 1–1. 7

[Lozano *et al.*, 2012] Lozano, C. A., Castro, F., y Ramírez, S. L. (2012). Unidades de medición fasorial (pmu). *EL Hombre y la Máquina*, (38):66–74. 3

[Magaña-Lemus *et al.*, 2015] Magaña-Lemus, E., Medina, A., y Ramos-Paz, A. (2015). Periodic steady state solution of power systems by selective transition matrix identification, lu decomposition and graphic processing units. En *2015 IEEE Power & Energy Society General Meeting*, pp. 1–5. IEEE. 8

[Nuqui y Phadke, 2007] Nuqui, R. y Phadke, A. G. (2007). Hybrid linear state estimation utilizing synchronized phasor measurements. En *2007 IEEE Lausanne Power Tech*, pp. 1665–1669. IEEE. 4

[Ortiz *et al.*, 2016] Ortiz, G. A., Colomé, D. G., y Quispe Puma, J. J. (2016). State estimation of power system based on scada and pmu measurements. En *2016 IEEE ANDESCON*, pp. 1–4. 3

[Pacheco, 2011] Pacheco, P. (2011). *An introduction to parallel programming*. Elsevier. 5

[Periñán y Expósito, 1999] Periñán, P. J. Z. y Expósito, A. G. (1999). *Estimación de estado y de parámetros en redes eléctricas*. Número 11. Universidad de Sevilla. 2

[Phadke y Thorp, 2008] Phadke, A. G. y Thorp, J. S. (2008). *Synchronized phasor measurements and their applications*, volumen 1. Springer. 20

[Poole, 2011] Poole, D. (2011). *Álgebra lineal: una introducción moderna*. Cengage Learning Editores. 46

[Quinn, 2003] Quinn, M. J. (2003). Parallel programming. *TMH CSE*, 526. 40

[Rabha *et al.*, 2015] Rabha, P., Shyam, C. C., y Sinha, A. K. (2015). Hybrid state estimation of power system using conventional and phasor measurements. En *2015 International Conference on Energy, Power and Environment: Towards Sustainable Growth (ICEPE)*, pp. 1–6. IEEE. 4

[Rendon *et al.*, 2015] Rendon, A., Fuerte, C., y Calderon, J. (2015). State estimation of electrical power grids incorporating scada and pmu measurements. *IEEE Latin America Transactions*, 13(7):2245–51. 4

[Rivera y Vargas-Lombardo, 2012] Rivera, I. O. y Vargas-Lombardo, M. (2012). Principios y campos de aplicación en cuda programación paralela y sus potencialidades. *Nexo Revista Científica*, 25(2):39–46. 5

[Romero *et al.*, 2012] Romero, Laorden, D., Martínez Graullera, O., Martín Arguedas, C., Ibanez, A., y Ullate, L. (2012). Paralelización de los procesos de conformación de haz para imagen ultrasónica con técnicas gpgpu. *Revista Iberoamericana de Automática e Informática Industrial*, 9(2):144–151. 35

[Sharma y Jain, 2018] Sharma, A. y Jain, S. K. (2018). A review and performance comparison of power system state estimation techniques. En *2018 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, pp. 770–775. IEEE. 14

[Singh y Aruni, 2010] Singh, J. y Aruni, I. (2010). Accelerating power flow studies on graphics processing unit. En *2010 Annual IEEE India Conference (INDICON)*, pp. 1–5. IEEE. 7

[Sodhi *et al.*, 2010] Sodhi, R., Srivastava, S., y Singh, S. (2010). Phasor-assisted hybrid state estimator. *Electric Power Components and Systems*, 38(5):533–544. 3

[Soni *et al.*, 2012] Soni, S., Bhil, S., Mehta, D., y Wagh, S. (2012). Linear state estimation model using phasor measurement unit (pmu) technology. En *2012 9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–6. IEEE. 4

[Tarali y Abur, 2012] Tarali, A. y Abur, A. (2012). Bad data detection in two-stage state estimation using phasor measurements. En *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, pp. 1–8. IEEE. 4, 16, 17, 28

[Torre, 2015] Torre, A. R. (2015). Implementación de una red neuronal tipo perceptrón en gpu. 37

[Wu, 1990] Wu, F. F. (1990). Power system state estimation: a survey. *International Journal of Electrical Power & Energy Systems*, 12(2):80–87. 2

[Xia *et al.*, 2017] Xia, Y., Chen, Y., Ren, Z., Huang, S., Wang, M., y Lin, M. (2017). State estimation for large-scale power systems based on hybrid cpu-gpu platform. En *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1–6. IEEE. 7, 47

[Zhou *et al.*, 2006] Zhou, M., Centeno, V. A., Thorp, J. S., y Phadke, A. G. (2006). An alternative for including phasor measurements in state estimators. *IEEE transactions on power systems*, 21(4):1930–1937. 4, 33