MODELADO Y SIMULACIÓN DE LA MÁQUINA SÍNCRONA CONECTADA A UNA BARRA INFINITA EMPLEANDO PROCESAMIENTO PARALELO

GERSON DAVID JAIMES SANTOS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2014

MODELADO Y SIMULACIÓN DE LA MÁQUINA SÍNCRONA CONECTADA A UNA BARRA INFINITA EMPLEANDO PROCESAMIENTO PARALELO

GERSON DAVID JAIMES SANTOS

Trabajo de grado para optar al título de Ingeniero Electricista

Director

HERMANN RAÚL VARGAS TORRES
Doctor Ingeniero Electricista

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2014

A mi madre, Ana Rosa Santos y a mi Padre Guillermo Jaimes.
Lo mas valioso que tengo en mi vida.

Agradecimientos

Le doy gracias a Dios por permitirme lograr esta gran meta.

A mis familiares, en especial a mis padres. Por el apoyo dado a lo largo de este tiempo, ya que ayudaron a hacer de mi sueño, una realidad.

A mis amigos, por brindarme su amistad y apoyo.

Al grupo de universitarios de Visión mundial, por todas la enseñanzas y experiencias que dejaron en mí.

A Visión Mundial por apoyarme durante todo este tiempo y haberme permitido participar en sus diferentes grupos, ya que de alguna u otra forma contribuyeron con mi formación personal y profesional.

Al profesor Hermann Raúl Vargas, por sus grandes aportes en la realización de este proyecto.

A los profesores de la escuela de ingeniería de eléctrica y electrónica, por sus enseñanzas a lo largo de este camino.

A todos los que de alguna manera estuvieron apoyándome en la realización de esta meta.

Índice general

Índice de figuras

Índice de tablas

Resumen

TÍTULO: MODELADO Y SIMULACIÓN DE LA MÁQUINA SÍNCRONA CONECTADA A UNA BARRA INFINITA EMPLEANDO PROCESAMIENTO PARALELO.*

AUTOR: GERSON DAVID JAIMES SANTOS.**

PALABRAS CLAVE: Estabilidad, procesamiento paralelo, método de integración implícita, unidad de procesamiento gráfico.

DESCRIPCIÓN:
El análisis de estabilidad comprende el comportamiento los generadores y su capacidad para mantener el sincronismo ante variaciones repentinas en el sistema. La respuesta de estos sistemas es de carácter no lineal y en su análisis se deben incluir las desviaciones de los ángulos entre rotores, en las velocidades de rotor y en las tensiones en los terminales de la máquina.

Debido a que el tiempo de simulación para sistemas de gran tamaño es alto, en este proyecto se hace uso de la tecnología de procesamiento paralelo para comparar los tiempos de cómputo. El método numérico empleado para la solución de las ecuaciones del modelo es el de integración implícita, y se adaptó para que los cálculos puedan ser procesados en el dispositivo gráfico (GPU). El programa fue implementado en MATLAB, ya que también tiene soporte para transferir la información a la GPU. Para tal fin, se organizan los parámetros en forma vectorial, ya que la GPU esta diseñado para procesar este tipo de datos. El sistema simulado corresponde al ejemplo 13.2 de [1] y es la base para desarrollar este proyecto.

Por otra parte, la forma de implementar la GPU consiste en transferir la información desde la RAM de la CPU hacia la RAM del dispositivo gráfico. Esta tarea se hace por medio de instrucciones predefinidas del lenguaje (Matlab). Una vez realizados los cálculos, la información se transfiere nuevamente a la RAM de la CPU.

Al final se presenta el ejemplo 13.2 de [1], comparando los tiempos de simulación para entender las ventajas y desventajas de implementar esta tecnología.

*Trabajo de grado

**Facultad de Ingenierías Fisicomecánicas. Escuela de ingenierías Eléctrica, electrónica y de Telecomunicaciones. Hermann Raúl Vargas Torres

Resume

TITLE: MODELING AND SIMULATION OF SYNCHRONOUS MACHINE CONNECTED TO AN INFINITE BUS USING PARALLEL PROCESSING.*

AUTHOR: GERSON DAVID JAIMES SANTOS**

KEYWORDS: Stability, parallel processing, implicit integration method, graphics processing unit.

ABSTRACT:
The stability analysis includes the behavior of generators and its ability to maintain synchronism under sudden changes in the system. The response of these systems is non-linear, and its analysis must include deviations between rotor angles, on rotor speeds and voltages on the machine's terminals.

Ought to the simulation time for large systems is high, this project uses the parallel processing technology in order to compare computation times. The numerical method used to solve the model's equations is the implicit integration, and it's adapted to the calculations can be processed in the graphics device (GPU). The program was implemented in MATLAB, since it also has support to transfer information to the GPU. To reach this goal, the parameters are arranged in vector form, ought to the GPU is designed to process such data. The simulated system corresponds to the example 13.2 of [1] and it is the basis for developing this project.

On the other hand, the way for implementing the GPU consist in transferring information from CPU's RAM to the Graphics Device's RAM. This work is done through instructions pre-defined by the computer language (Matlab). Once the calculations are done, the information is transferred again to the CPU's RAM.

Finally, it's presented the example 13.2 from [1], comparing the simulation times in order to understand the advantages and disadvantages of implementing this technology.

*Degree project

**Faculty of Physic-mechanical Engineering, Electrical, Electronics and Telecommunications Engineering School

Introducción

El presente libro describe el modelo matemático de la dinámica de la máquina síncrona y los conceptos relacionados con la estabilidad del sistema. Posteriormente se plantea la solución del modelo matemático mediante procesamiento paralelo.

Se inicia con los conceptos básicos de estabilidad de la máquina síncrona, como el criterio de áreas equivalente y la respuesta a una falla de corto circuito. luego, se presentan las ecuaciones relacionadas al rotor, el estator, el sistema de excitación y la respectiva solución de la ecuaciones. Posteriormente, se presentan los conceptos básicos de procesamiento paralelo en lenguaje MATLAB y la forma en la que se trabaja en el dispositivo desde este lenguaje. Después se da la solución al problema de estabilidad empleando el dispositivo gráfico. Por último se presentan las conclusiones dadas por este desarrollo.

Capítulo 1

Objetivo general

Modelar y simular la máquina síncrona conectada a una barra infinita ante grandes perturbaciones por medio de procesamiento paralelo.

1.1. Objetivos específicos

1. Modelar la máquina utilizando software de procesamiento paralelo permitiendo incluir otras máquinas.
2. Modelar las líneas y transformadores utilizando software de procesamiento paralelo permitiendo incluir otras líneas y transformadores.
3. Elaborar la herramienta software que permita simular el comportamiento de la máquina utilizando procesamiento paralelo.
4. Presentar los resultados.

1.2. Descripción del problema

En la actualidad las herramientas software resuelven los problemas de simulación dinámica de sistemas eléctricos de potencia fuera de línea debido a que los tiempos de simulación son muy largos comparados con la realidad.

Por tal motivo se propone el procesamiento paralelo. Para dar inicio en este campo, se modelará el comportamiento de la máquina síncrona ante la presencia de grandes perturbaciones.
Además, el software será versátil con el fin permitir la conexión de varias máquinas síncronas.

Capítulo 2

Marco de referencia

En este capítulo, se presenta una breve reseña histórica del fenómeno de estabilidad y la de los dispositivos gráficos. Además, se dan los conceptos básicos del comportamiento ante una gran perturbación, de la máquina síncrona conectada a través de dos líneas a una barra infinita. Adicionalmente, se presenta una breve teoría acerca del procesamiento de la información utilizando el dispositivo gráfico del equipo de cómputo.

2.1. Reseña histórica

Los problemas de estabilidad surgieron en centrales hidroeléctricas interconectadas a grandes cargas, las cuales estaban conectadas através de sistemas de transmisión débiles y operados en los límites de la estabilidad. Adicionalmente, se tenían sistemas de protección lentos que originaban inestabilidades no oscilatorias, ya que el torque sincronizante entre los generadores era deficiente. Dada la importancia del fenómeno eléctrico, empezaron sus estudios a partír de modelos simples debido a las limitaciones tecnológicas se enfocaban en los sistemas de transmisión.

Una de las alternativas que surgieron para la solución de este problema fue la de instalar líneas de interconexión para darle rigidez al sistema, o colocar en los sistemas de excitación de los generadores, reguladores automáticos con ganancias elevadas. Por razones económicas, la segunda opción resultó ser la mejor. Aunque la instalación de estos sistemas de compensación trajo como problema, oscilaciones poco amortiguadas o crecientes en amplitud[2].

Por otra parte, han habido avances tecnológicos en el área de la computación, que permiten analizar el fenómeno de manera mas compleja y centrándose en el comportamiento de la máquina síncrona. Unos de ellos, es el generado por NVIDIA [5] y sus dispositivos de procesamiento gráfico, que culminó en 1999 con la primer GPU. Inmediatamente, investigadores de otras áreas empezaron a aprovechar su rendimiento en punto flotante, surgiendo así la GPGPU (unidad de procesamiento gráfico de propósito general). En sus inicios era difícil de programar, ya que había que utilizar OpenGL, una plataforma que permitía ejecutar el código sobre el dispositivo. Fue hasta el 2003, que investigadores lanzaron el primer módulo de programación, que extendía el lenguaje C para procesar datos de manera paralela, llevando la GPU a lenguajes de alto nivél, siendo fáciles de escribir y con gran poder

de procesamiento[4].

A partír de estos avances, NVIDIA desarrolló una herramienta software que aprovechará de la mejor manera el dispositivo gráfico, creando así CUDA[5].

2.2. Estabilidad transitoria

En la actualidad se tiene como fin, garantizar suministro de energía a la totalidad de usuarios que se encuentran distribuidos en las diferentes zonas del país. La energía se transcurre a través de diferentes elementos que conforman el sistema, que van desde una central eléctrica que genera a partír de fuentes de energía primaria, luego se transmite y posteriormente se distribuye en los diferentes usuarios, ya sean comerciales, industriales o domésticos. Debido a estas características, además de ser no lineales, los sistemas de potencia son dinámicos, complejos.

Partiendo de una condición de equilibrio, la estabilidad del sistema esta dada por la capacidad de retornar a este punto o a otro punto aceptable de equilibrio, luego de la eventualidad. Además, la estabilidad del sistema está influenciada por la no linealidad, el tipo de perturbación y las condiciones en las que se encontraba el sistema antes de ocurrida la eventualidad.

La estabilidad se analiza a partír de la separación resultante entre el ángulo de las máquinas en el sistema. Si este valor está dentro de ciertos límites, se dice que el sistema está en sincronismo.

2.2.1. Conceptos básicos de estabilidad transitoria

Para analizar el tema, se considera como ejemplo un generador que entrega potencia a una barra infinita a través de dos líneas de transmisión. El efecto de las resistencias en el sistema se considera despreciable al igual que el efecto de la velocidad del gobernador. Se cumple el modelo de la máquina síncrona con cuatro variables de estado denominado modelo clásico. El sistema descrito anteriormente se muestra en la figura 2.1.

La tensión inducida y la reactancia transitoria se denotan por $E^{'}$ y $X_d^{'}$ respectivamente. El ángulo δ representa el ángulo entre $E^{'}$ y E_B. La magnitud de $E^{'}$ durante la pre-falla permanece constante, en el momento de la falla, la magnitud $E^{'}$ se mantiene constante en el valor de prefalla y la magnitud del ángulo δ varía de acuerdo con la desviación de la velocidad del rotor con respecto a la de sincronismo ω_0[1]. Las ecuaciones son tomadas de [1].

La ecuaciones que modelan el sistema son:

$$P_e = \frac{E^{'} E_B}{X_T} \sin(\delta) = P_{MAX} \sin(\delta) \tag{2.2.1}$$

donde:

$$P_{MAX} = \frac{E^{'} E_B}{X_T} \tag{2.2.2}$$

El circuito simplificado se muestra en la figura 2.2.

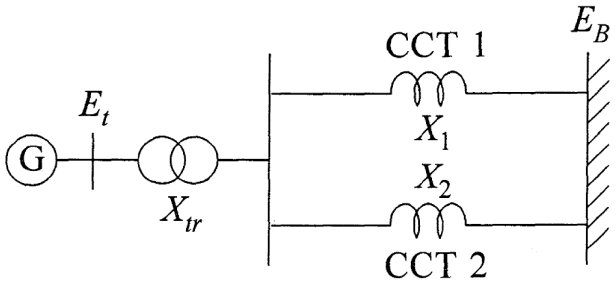


Figura 2.1: Máquina conectada a una barra infinita [1].

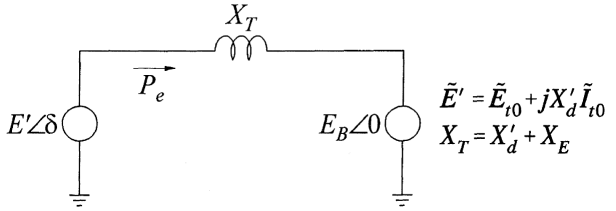


Figura 2.2: Circuito simplificado [1].

Ya que se desprecia la resistencia del estator, P_e representa la potencia en el entrehierro como la potencia en terminales. Ahora, la potencia mecánica de entrada a la máquina P_m, en condiciones de estado estable, es igual a la potencia eléctrica de salida P_e. La relación entre la potencia y el ángulo a través de las líneas y las condiciones de operación para el punto a, se representa en la gráfica 2.3.

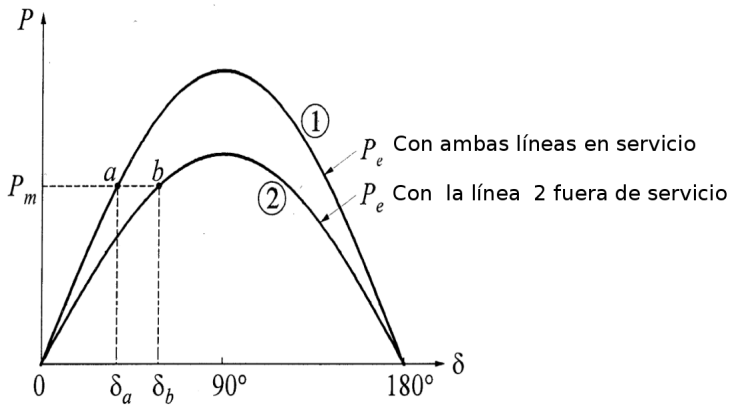


Figura 2.3: Relación entre la potencia y el ángulo [1].

Si una de las líneas esta fuera de servicio, la reactancia efectiva X_T es mayor. La relación potencia ángulo con la línea 2 fuera de servicio se muestra en la figura 2.3. La potencia máxima es ahora mas baja. Con una potencia mecánica de entrada P_m, el ángulo del rotor es ahora δ_b que corresponde al punto de operación del punto b en la curva 2 de la figura 2.3. Con una mayor reactancia, el ángulo del rotor es mayor con el fin de transmitir la misma potencia en estado estable.

En la perturbación, la oscilación de δ se impone sobre la velocidad de sincronismo ω_0. pero la desviación de la velocidad ($\Delta\omega_r = d\delta/dt$) es mucho mas pequeña que ω_0. lo que se considera a ω_r aproximadamente igual a ω_0[1].

La ecuación del movimiento esta determinada por:

$$\frac{2H}{\omega_o}\frac{d^2\delta}{dt^2} = P_m - P_{max}\sin\delta \tag{2.2.3}$$

donde

P_m = Potencia mecánica de entrada en pu.
P_{max} = Máxima potencia eléctrica de salida en pu.
H = Constante de inercia, en $MW \cdot s/MVA$.
δ = Ángulo del rotor. rad eléctricos.
t = Tiempo, dado en s.

Respuesta un cambio de paso en P_m

Ahora se examina el comportamiento transitorio del sistema, con ambos circuitos en servicio, considerando un incremento repentino en la potencia mecánica de entrada de un valor inicial P_{m0} a P_{m1} como se muestra en la figura. Debido a la inercia del rotor, el ángulo del rotor no puede cambiar repentinamente de un valor inicial δ_0 a δ_1 correspondiente al nuevo punto de equilibrio b, en el cual $P_e = P_{m1}$. La potencia mecánica ahora excede a la potencia eléctrica. La resultante es un torque acelerante que causa que el rotor acelere desde la condición inicial de operación en el punto a, hacia el nuevo punto de equilibrio b. Trazando la curva $P_e - \delta$ la variación está determinada por la ecuación de oscilación. La diferencia entre P_{m1} y P_e e cualquier instante representa la potencia de aceleración.

Cuando se alcanza el punto b, la potencia de aceleración es cero, pero la velocidad del rotor es mas alta que la velocidad de sincronismo ω_0. Por lo tanto, el ángulo del rotor continua incrementándose. Para valores de δ mayores que δ_1, P_e es mayor que P_{m1} y el rotor desacelera. Para algún pico δ_m, la velocidad del rotor recupera el valor de sincronismo ω_0, pero P_e es más alta que P_{m1}. El rotor continua desacelerando y la velocidad cae por debajo de ω_0. El punto de operación trazado en la curva $P_e - \delta$ pasa de c a b y luego a c. El ángulo del rotor oscila indefinidamente cerca del nuevo ángulo de equilibrio δ_1 con una amplitud constante como se muestra en la gráfica del tiempo en la figura.

En la representación del sistema de potencia anteriormente descrito, se despreciaron todas las resistencias y el modelo clásico se emplea para representar al generador. Se desprecian todas fuentes de amortiguamiento. Por lo tanto, las oscilaciones del rotor se mantienen después de la perturbación.

Criterio de áreas equivalentes

Para el sistema considerado anteriormente, no es necesario resolver formalmente la ecuación de oscilación para determinar si el ángulo del rotor se incrementa indefinidamente u oscila alrededor de un punto de equilibrio. La información sobre la máxima desviación angular δ_m, y los límites pueden ser obtenidos gráficamente usando el diagrama de potencia-ángulo como se muestra en la figura 2.4. Aunque este método no aplica a múltiples máquinas con representación detallada de la máquina síncrona, si ayuda a entender los factores básicos que influencian la estabilidad transitoria para cualquier sistema. De la ecuación (2.2.3), despejando $\frac{d\delta}{dt}$ e igualándolo a

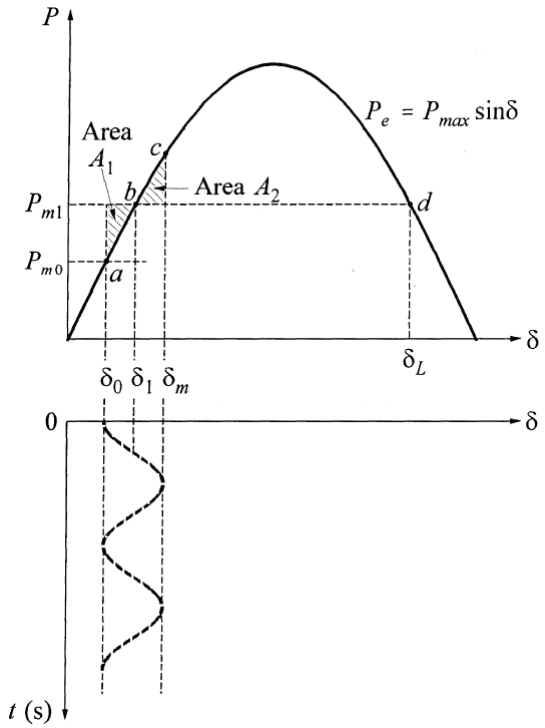


Figura 2.4: Variación potencia-ángulo y respuesta del ángulo del rotor [1].

cero. La expresión resultante es:

$$\left[\frac{d\delta}{dt}\right]^2 = \int \frac{\omega_0(P_m - P_e)}{H} d\delta \tag{2.2.4}$$

La desviación de la velocidad es el inicialmente cero. El cambio es resultado de la perturbación. Para la operación estable, la desviación del ángulo δ se puede delimitar, alcanzando un valor máximo en el punto c de la figura 2.4 y luego cambiando de dirección. Esto requiere que la desviación de la velocidad $d\delta/dt$ llegue a ser cero algún tiempo después de la perturbación. Por lo tanto, la ecuación (2.2.4) como criterio de estabilidad se puede escribir como:

$$\int_{\delta_0}^{\delta_m} \frac{\omega_0}{H}(P_m - P_e)\, dt = 0 \tag{2.2.5}$$

Donde δ_0 es el ángulo inicial del rotor y δ_m es el máximo ángulo del rotor, como se muestra en la figura 2.4. Asi, el área bajo la función $P_m - P_e$ trazada contra δ necesita ser cero para que el sistema sea estable. De la figura 2.4, esto se satisface cuando el área A_1 es igual al A_2. La energía cinética ganada durante la aceleración, cuando δ cambia de δ_0 a δ_1 es:

$$E_1 = \int_{\delta_0}^{\delta_1} (P_m - P_e) d\delta = A_1 \tag{2.2.6}$$

La energía durante la desaceleración cuando δ cambia de δ_1 a δ_m es:

$$E_2 = \int_{\delta_1}^{\delta_m} (P_e - P_m) d\delta = A_2 \tag{2.2.7}$$

Sin considerar las perdidas, El área A_1 es igual al área A_2.

Respuesta a una falla de corto circuito

Se considera la respuesta del sistema ante una falla trifásica ubicada en el punto F en la línea de transmisión de la figura 2.5a. El correspondiente circuito equivalente, suponiendo el modelo clásico del generador, como se muestra en la figura 2.5b. La falla se despeja por apertura de las protecciones de las líneas en ambas terminaciones del circuito fallado, el tiempo de despeje de la falla depende del tiempo que tarde en actuar la protección.

Si la localización de falla F está en el extremo emisor de el circuito fallado, no se transmite potencia a la barra infinita. La corriente de corto circuito fluye a través de las reactancias puras de la falla. Por lo tanto, solo la potencia reactiva fluye y la potencia P_e y el correspondiente torque eléctrico en el entrehierro son cero durante la falla. Si se incluyen las resistencias del estator y transformadores, P_e puede tener un valor muy pequeño que corresponde a las pérdidas resistivas.

Si la falla está ubicada a una distancia lejana del extremo de envió como se muestra en la figura 2.5a y 2.5b, se puede transmitir alguna potencia a la barra infinita mientras la falla este activa.

Las figuras 2.5c y 2.5d, muestran el gráfico de $P_e - \delta$ para las tres condiciones de la red:

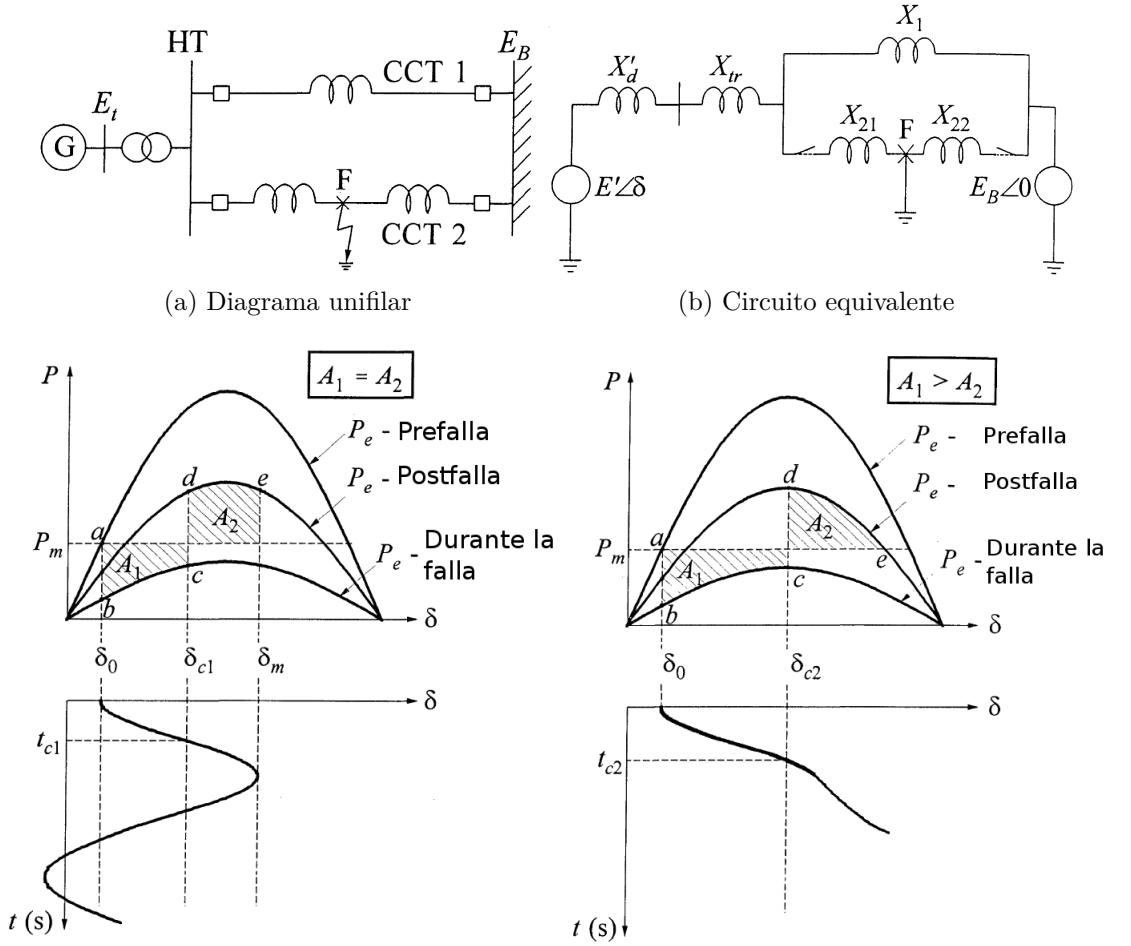
1. Prefalla, con ambos circuitos en servicio.

2. Con una falla trifásica en el circuito 2, ubicada a una distancia del extremo emisor.

3. Postfalla, circuito 2 fuera de servicio.

La figura 2.5c considera el rendimiento del sistema con un tiempo de despeje de t_{c1} y representa el caso estable. La figura 2.5d considera un tiempo t_{c2} mayor de despeje de la falla, tal que que el sistema es inestable. En ambos casos P_m se asume constante.

Se examina el caso estable representado en la figura 2.5c. Inicialmente el sistema está operando con ambos circuitos en servicio tal que $P_w = P_e$ y $\delta = \delta_0$. Cuando la falla ocurre, el punto de operación cambia repentinamente de a a b. Debido a la inercia, el ángulo δ no puede cambiar repentinamente. Ya que P_m es ahora mayor que P_e, el rotor acelera hasta alcanzar el nuevo punto c de operación, cuando la falla se despeja aislando el circuito 2 del sistema. El nuevo punto de operación cambia de repente al punto d. Ahora P_e es mayor que P_m, causando desaceleración del rotor. Desde que la velocidad del rotor es mayor que la de sincronismo ω_0, δ continua incrementándose hasta que la energía cinética ganada durante el periodo de aceleración se gasta por transferencia de energía al sistema. El punto de operación se mueve de d a e, la velocidad es igual a ω_0 y δ alcanza el máximo valor δ_m. Ya que P_e es todavía mayor que P_m, el rotor va más despacio con la velocidad cayendo por debajo de ω_0. El ángulo del rotor decrece, y el punto de operación traza la trayectoria de e a d y sigue la curva $P_e - \delta$ para el sistema de postfalla. Debido a la ausencia de cualquier fuente de amortiguamiento, el rotor continua oscilando con amplitud constante. Para la figura 2.5d, la energía cinética ganada durante el periodo de aceleración no puede ser completamente entregada, como consecuencia, la velocidad



(a) Diagrama unifilar

(b) Circuito equivalente

(c) Respuesta estable en caso de falla

(d) Respuesta inestable en caso de falla

Figura 2.5: Fenómeno de la estabilidad transitoria [1].

es todavía mayor que ω_0 y δ continua incrementándose. Más allá del punto e, P_e es menor que P_m, y el rotor empieza a acelerarse nuevamente. La velocidad del rotor y el ángulo continúan incrementándose, provocándose la pérdida de sincronismo.

De lo mencionado anteriormente, y de la figura 2.5, se concluye que los factores que influyen en la estabilidad de la máquina síncrona dependen de:

- La carga que tiene el generador.
- El lugar de la falla y el tipo de falla.
- El tiempo de despeje de la falla.
- La reactancia del sistema de transmisión en la postfalla.
- La reactancia del generador, ya que aumenta el pico de potencia y disminuye el ángulo inicial del rotor.
- La inercia del generador, a mayor inercia menor razón de cambio del ángulo. Porque se reduce la energía cinética ganada durante la falla.

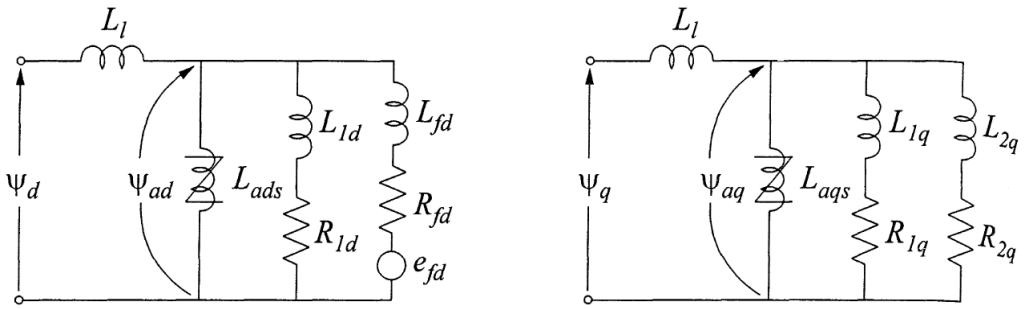


Figura 2.6: Circuitos equivalentes de los ejes d y q [1].

- La magnitud de la tensión interna $E^{'}$ del generador. Este depende del flujo de excitación.
- La magnitud de la tensión de la barra infinita.

2.2.2. Modelado de la máquina síncrona

Para el análisis de la estabilidad transitoria, el generador se representa por un devanado directo d y dos devanados amortiguadores q, de acuerdo a la figura 2.6. Las unidades de las siguientes ecuaciones están dadas en por unidad pu, con el tiempo t en segundos, y el ángulo δ en radianes eléctricos[1].

ecuaciones del movimiento

$$p\Delta\omega_r = \frac{T_m - T_e - K_D\Delta\omega_r}{2H} \tag{2.2.8}$$

$$p\delta = \omega_0\Delta\omega_r \tag{2.2.9}$$

Donde
$\omega_0 = 2\pi f_0\ rad/s$
$\Delta\omega_r = pu$ variación de la velocidad del rotor.
p = Operador derivador d/dt

Ecuaciones del circuito del rotor

Las ecuaciones que modelan la dinámica del circuito del rotor son:

$$p\psi_{fd} = \omega_0\left(e_{fd} + \frac{(\psi_{ad} - \psi_{fd})R_{fd}}{L_{fd}}\right) \tag{2.2.10}$$

$$p\psi_{1d} = \omega_0\left(\frac{\psi_{ad} - \psi_{1d}}{L_{1d}}\right)R_{1d} \tag{2.2.11}$$

$$p\psi_{1q} = \omega_0\left(\frac{\psi_{aq} - \psi_{1q}}{L_{1q}}\right)R_{1q} \tag{2.2.12}$$

$$p\psi_{2q} = \omega_0\left(\frac{\psi_{aq} - \psi_{2q}}{L_{2q}}\right)R_{2q} \tag{2.2.13}$$

Los acoples de los ejes d y q están dados:

$$\psi_{ad} = -L_{ads}i_d + L_{ads}i_{fd} + L_{ads}i_{1d} = L_{ads}^{"}\left(-i_d + \frac{\psi_{fd}}{L_{fd}} + \frac{\psi_1 d}{L_{1d}}\right) \quad (2.2.14)$$

$$\psi_{aq} = L_{aqs}^{"}\left(-i_q + \frac{\psi_{1q}}{L_{1q}} + \frac{\psi_{2q}}{L_{2q}}\right) \quad (2.2.15)$$

Donde las inductancias están definidas de la siguiente manera:

$$L_{ads}^{"} = \frac{1}{\frac{1}{L_{ads}}+\frac{1}{L_{fd}}+\frac{1}{L_{1d}}} \quad (2.2.16)$$

$$L_{aqs}^{"} = \frac{1}{\frac{1}{L_{aqs}}+\frac{1}{L_{1q}}+\frac{1}{L_{2q}}} \quad (2.2.17)$$

Los términos L_{ads} y L_{aqs} son los valores saturadas del efecto mutuo de las inductancias del eje d y eje q

$$L_{ads} = K_{sd}L_{adu}L_{aqs} = K_{sq}L_{aqu} \quad (2.2.18)$$

Las constantes K_{sd} y K_{sq} dependen de la relación del flujo en el entre hierro.

Ecuaciones de la tensión en el estator

Para el análisis, las tensiones en el estator se modelan de la siguiente manera.

$$e_d = -R_a i_d + (\overline{\omega}L_q^{"})i_q + E_d^{"} \quad (2.2.19)$$

$$e_q = -R_a i_q + (\overline{\omega}L_q^{"})i_q + E_q^{"} \quad (2.2.20)$$

Con

$$E_d^{"} = -\overline{\omega}L_{aqs}^{"}\left(\frac{\psi_{1q}}{L_{1q}} + \frac{\psi_{2q}}{L_{2q}}\right) \quad (2.2.21)$$

$$E_q^{"} = \overline{\omega}L_{ads}^{"}\left(\frac{\psi_{fd}}{L_{fd}} + \frac{\psi_{1d}}{L_{1d}}\right) \quad (2.2.22)$$

Donde

$$L_d^{"} = L_l + L_{ads}^{"} \quad (2.2.23)$$

$$L_q^{"} = L_l + L_{aqs}^{"} \quad (2.2.24)$$

Ya que se han despreciado los efectos de las variaciones de la velocidad sobre los efectos de la tensión en el estator, $\overline{w} = \frac{\omega}{\omega_0} = 1$, por lo tanto, $\overline{w}L_d^{''} = X_d^{''}$ y $\overline{w}L_q^{''} = X_q^{''}$. Las ecuaciones anteriormente expuestas están en un marco de referencia $d-q$, el cual rota con el rotor de la máquina. Para la solución de las ecuaciones de la interconexión de una red de transmisión se emplea una referencia común que rota sincronizadamente. La transformación de un marco de referencia a otro, se muestra en la figura 2.7.

La variación de la velocidad $\overline{\omega} = \omega/\omega_0 = 1$ del rotor y su incidencia en la tensión del estator no se considera. A partír de estas definiciones, Las reactancias son $\overline{\omega}L_d^{"} = X_d^{"}$ y $\overline{\omega}L_q^{"} = X_q^{"}$. Las ecuaciones están especificadas para una máquina individual con ejes de referencia $d-q$ que rota de acuerdo con el ángulo de la máquina . Para considerar la interconexión de líneas de transmisión, se transforman

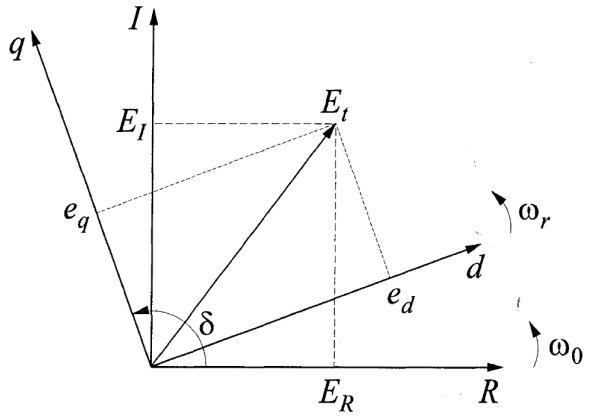


Figura 2.7: Marco de referencia y definición del ángulo delta [1].

la variables a un eje de referencia $R-I$, el cual permite analizar el ángulo δ de cada máquina.

Las ecuaciones de acuerdo con el gráfico 2.7 se definen de la siguiente manera.

$$E_R = e_d \sin\delta + e_q \cos\delta \tag{2.2.25}$$
$$E_I = e_q \sin\delta - e_d \cos\delta \tag{2.2.26}$$
$$e_d = E_R \sin\delta - E_l \cos\delta \tag{2.2.27}$$
$$e_q = E_L \sin\delta + E_R \cos\delta \tag{2.2.28}$$

Reorganizando las ecuaciones del estator con respecto al eje de referencia común $R-I$, la matriz resultante es:

$$\begin{bmatrix} E_R \\ E_I \end{bmatrix} = \begin{bmatrix} -R_{RR} & X_{RI} \\ -X_{IR} & -R_{II} \end{bmatrix} \begin{bmatrix} I_R \\ I_I \end{bmatrix} + \begin{bmatrix} E_R^{"} \\ E_I^{"} \end{bmatrix} \tag{2.2.29}$$

De la ecuación (2.2.29), se define:

$$I_D = \begin{bmatrix} I_R \\ I_I \end{bmatrix} \tag{2.2.30}$$

Los elementos de la matriz de impedancia y las tensiones E_R'' y E_I'' son:

$$R_{RR} = (X_d^{"} - X_q^{"}) \sin\delta\cos\delta + R_a \tag{2.2.31}$$
$$R_{II} = (X_q^{"} - X_d^{"}) \sin\delta\cos\delta + R_a \tag{2.2.32}$$
$$X_{RI} = X_d^{"} \cos^2\delta + X_q^{"} \sin^2\delta \tag{2.2.33}$$
$$X_{IR} = X_d^{"} \sin^2\delta + X_q^{"} \cos^2\delta \tag{2.2.34}$$
$$E_R^{"} = E_d^{"} \sin\delta + E_q^{"} \cos\delta \tag{2.2.35}$$
$$E_I^{"} = E_q^{"} \sin\delta - E_d^{"} \sin\delta \tag{2.2.36}$$

La potencia activa y reactiva del estator es:

$$P_t = e_d i_d + e_q i_q \tag{2.2.37}$$
$$Q_t = e_q i_d - e_d i_q \tag{2.2.38}$$

El torque en el entre hierro para resolver la ecuación de oscilación es

$$T_e = \psi_d i_q - \psi_q i_d = \psi_{ad} i_q - \psi_{aq} i_d \quad (2.2.39)$$
$$T_e = P_e = P_t + R_a I_t^2 \quad (2.2.40)$$

En sistema *pu* la corriente de campo esta dada por:

$$i_{fd} = \frac{\psi_{fd} - \psi_{ad}}{L_{fd}} \quad (2.2.41)$$

Y la corriente de excitación de salida I_{fd} es:

$$I_{fd} = L_{adu} i_{fd} \quad (2.2.42)$$

El modelo considerado tiene un eje d y dos eje q de circuitos amortiguadores.

Valores iniciales

Los valores de prefalla involucrados en el análisis de estabilidad se obtienen a partír de un flujo de cargas del sistema. De igual manera, las tensiones, potencia activa y reactiva.

2.2.3. Representación del sistema de excitación

Se considera un sistema de excitación retro-alimentado por tiristor llamado (STA1), junto a un regulador automático de la tensión (AVR) y un estabilizador de sistemas de potencia (PSS). Además se tiene una alta ganancia de excitación (K_A) sin reducir la ganancia transitoria. Para un excitador de tiristor retro-alimentado, la tensión varía con la tensión de los terminales del generador E_t y el excitador con la salida de la corriente del flujo directo I_{fd}. El diagrama correspondiente se muestra a continuación:

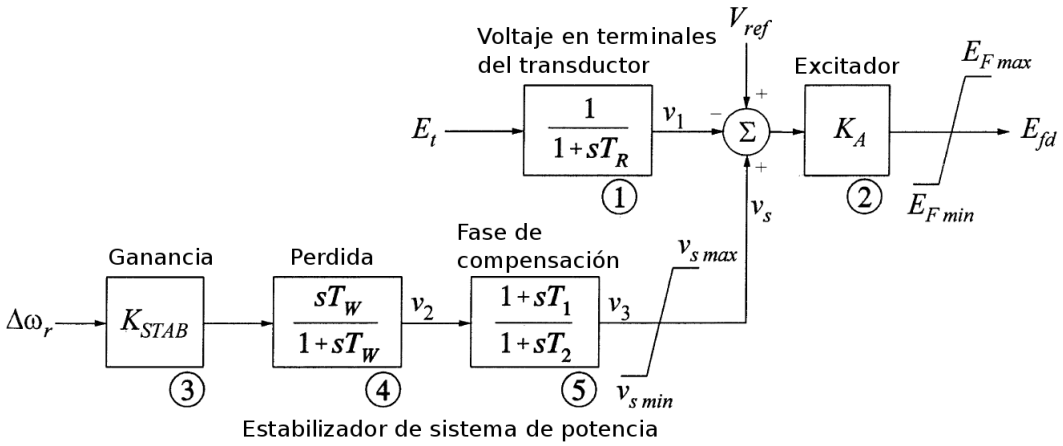


Figura 2.8: Sistema de excitación con AVR y PSS [1].

$$E_{Fmax} = V_{Rmax} E_t - K_c I_{fd}$$
$$E_{Fmin} = V_{Rmin} E_t \quad (2.2.43)$$

Del bloque 1 de la figura 2.8, la ecuaciones es:

$$pv_1 = \frac{E_t - v_1}{T_R} \tag{2.2.44}$$

y de los bloques 3 y 4 nos queda:

$$pv_2 = K_{STAB} p\Delta\omega_r - \frac{v_2}{T_w} \tag{2.2.45}$$

El bloque 5 y la ecuación [2.2.9] el resultado es:

$$pv_3 = \frac{T_1 pv_2 + v_2 - v_3}{T_2} \tag{2.2.46}$$

El término pv_2 dado en la ecuación (2.2.45). Se obtiene la salida del estabilizador v_s

$$v_s = v_3 \tag{2.2.47}$$

v_s se encuentra en el rango:

$$v_{smax} \geq v_s \geq v_{smin} \tag{2.2.48}$$

Para el bloque 2, la salida del excitador es

$$E_{fd} = K_A[V_{ref} - v_1 + v_s] \tag{2.2.49}$$

el rango en el que se encuentra E_{fd} es:

$$E_{Fmax} \geq E_{fd} \geq E_{Fmin} \tag{2.2.50}$$

De la ecuación (2.2.51), la tensión de campo de generador e_{fd} en pu, se relaciona con E_{fd} y son proporcionales.

$$e_{fd} = \frac{R_{fd}}{L_{adu}} E_{fd} \tag{2.2.51}$$

Para el estado estable del generador, la tensión de campo e_{fd} se determina por las ecuaciones del generador de la sección 2.2.2. Los valores del sistema de excitación se determinan de la siguiente manera.

$$\begin{aligned} E_{fd} &= \frac{L_{adu}}{R_{fd}} e_{fd} \\ v_1 &= E_t \quad v_2 = 0 \quad v_s = 0 \end{aligned} \tag{2.2.52}$$

La tensión de referencia para el AVR es:

$$V_{ref} = \frac{E_{fd}}{K_A} + v_1 \tag{2.2.53}$$

El valor de referencia toma valores de acuerdo con las condiciones de carga del generador, en estado estable.

2.2.4. Red de transmisión y representación de la carga

Los transitorios relacionados con la red de transmisión decaen muy rapidamente. Por lo tanto, es usual adaptar la red, durante las condiciones de transitorio electromecánico, como si pasara de un estado estable a otro. La frecuencia fundamental se puede considerar como microprocesos, y solo las variaciones sobre la forma de onda de la tensión y corriente se consideran para el análisis de estabilidad. Para análisis de condiciones balanceadas, la representación monofásica de las tres fases es usada. La representación de las cargas estáticas hacen parte de las ecuaciones de la red. Las cargas con una admitancia característica constante son simples para manejar y se incluyen en la matriz admitancia. Las cargas no lineales se representan como una función exponencial o polinómica de la magnitud de la tensión y frecuencia. El efecto neto, es que el modelo de las cargas estáticas no lineales se trata como una inyección de corriente en la propia ecuación de la red. El valor de la corriente de nodo de tierra en la red es.

$$\widetilde{I}_L = -\frac{P_L - jQ_L}{\widetilde{V_L}^*} \tag{2.2.54}$$

Donde $\widetilde{V_L}^*$ es el conjugado de las tensiones en las barras, y P_L y Q_L son parte de la potencia activa y reactiva de la carga que varía como una función no lineal de V_L y la desviación de la frecuencia. Para una carga inductiva, Q_L es positiva.

La representación general de la carga y la red comprende una gran matriz dispersa de admitancia con una estructura similar al del flujo de carga. La matriz se denota por:

$$\widetilde{I} = Y_N \widetilde{V} \tag{2.2.55}$$

La matriz admitancia Y_N es simétrica, excepto por la asimetría introducida por los transformadores desfasadores

Simulación de la falla

Para una falla trifásica, la impedancia de falla es cero y la barra afectada tiene el mismo potencial que la tierra. Esto implica ubicar una admitancia infinita en paralelo para que la tension en la barra afectada sea cero. Esto se realiza colocando una G igual a $10^6 pu$. La falla se elimina restaurando la admitancia en paralelo a un valor apropiado dependiendo de la configuración del sistema de postfalla.

2.3. Solución mediante integración implícita

Para el análisis, no se tienen en cuenta los transitorios electromagnéticos en la solución de la ecuaciones algebraico-diferencial de tipo no lineal de primer orden. En cuanto al algoritmo implementado en la solución de este sistema se optó por la regla trapezoidal. El algoritmo implementado se presenta a continuación:

$$x_{n+1} = x_n + \frac{\Delta t}{2}\left[f(x_{n+1}, V_{n+1}) + f(x_n, V_n)\right] \tag{2.3.1}$$

$$I(x_{n+1}, V_{n+1}) = Y_N V_{n+1} \tag{2.3.2}$$

Igualando las anteriores expresiones a cero y definiendo los vectores F y G iguales a cero se obtiene:

$$F(x_{n+1}, V_{n+1}) = x_{n+1} - x_n - \frac{\Delta t}{2}\left[f(x_{n+1}, V_{n+1}) + f(x_n, V_n)\right] \quad (2.3.3)$$

$$G(x_{n+1}, V_{n+1}) = Y_n V_{n+1} - I(x_{n+1}, V_{n+1}) \quad (2.3.4)$$

Para resolver la ecuaciones de $F(x_{n+1}, v_{n+1}) = 0$ y $G(x_{n+1}, v_{n+1}) = 0$ se emplea el método de Newton. La ecuación (4.1.76) se resuelve obteniendo x^k_{n+1} y V^k_{n+1}, a partír de la ecuación (2.3.6).

$$\begin{bmatrix} x^{k+1}_{n+1} \\ V^{k+1}_{n+1} \end{bmatrix} = \begin{bmatrix} x^k_{n+1} \\ V^k_{n+1} \end{bmatrix} + \begin{bmatrix} \Delta x^k_{n+1} \\ \Delta V^k_{n+1} \end{bmatrix} \quad (2.3.5)$$

$$\begin{bmatrix} -F(x^k_{n+1}, V^k_{n+1}) \\ -G(x^k_{n+1}, V^k_{n+1}) \end{bmatrix} = \begin{bmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial V} \\ \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta x^k_{n+1} \\ \Delta V^k_{n+1} \end{bmatrix} \quad (2.3.6)$$

El jacobiano tiene la siguiente estructura:

$$J = \begin{bmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial V} \\ \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial V} \end{bmatrix} = \begin{bmatrix} A_D & B_D \\ C_D & Y_N + Y_D \end{bmatrix} \quad (2.3.7)$$

Donde las matrices A_D, B_D, C_D y Y_D están asociados a los modelos dinámicos de los dispositivos y cargas no lineales. Para un sistema con m dispositivos, la estrucutura es la siguiente:

$$A_D = \begin{bmatrix} A_{d1} & 0 & \cdots & 0 \\ 0 & A_{d2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & A_{dm} \end{bmatrix} \qquad B_D = \begin{bmatrix} B_{d1} \\ B_{d2} \\ \vdots \\ B_{dm} \end{bmatrix} \quad (2.3.8)$$

$$C_D = \begin{bmatrix} C_{d1} & C_{d2} & \cdots & C_{dm} \end{bmatrix} \qquad Y_D = \begin{bmatrix} Y_{d1} & 0 & \cdots & 0 \\ 0 & Y_{d2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & Y_{dm} \end{bmatrix} \quad (2.3.9)$$

Las ecuaciones $f(x, V) = px$ hacen referencia a las condiciones dinámicas del sistema, como los generadores. El término referente a $0 = g(x, V)$ son términos algebraicos no lineales que representan las líneas y transformadores.

2.4. Software existente para la simulación dinámica de sistemas de potencia y procesamiento paralelo

Con el desarrollo en los años 50 en el ámbito de la computación, se diseñaron los primeros programas para el análisis de la estabilidad en sistemas de potencia.

Adicionalmente, los avances en los métodos numéricos permitió desarrollar software de grandes prestaciones. El análisis se ha centrado enormemente en el fenómeno de la estabilidad de ángulo, ya que se ha tomado este criterio como límite de la estabilidad. De acuerdo con las características del sistema, se aborda el problema de estabilidad empleando técnicas especiales, como optimización de los algoritmos existentes[2].

Las características del software son:

- Interfaces gráficas que permite modificar la topología.
- Técnicas para la solución numérica del sistema en la que resultan ecuaciones algebraico-diferenciales, y la dimensión depende de la topología del sistema eléctrico.
- Herramientas gráficas para la visualización de resultados de simular el comportamiento de la máquina síncrona.
- Herramientas para el estudio especializado de sistemas de control.

Las herramientas software utilizadas actualmente en el mercado son el DIgSILENT, NEPLAN y SIMPOW. Estos programas tienen las características anteriormente mencionadas y se utiliza para analizar el programa fuera de línea. Debido a que los sistemas de potencia se trabajan cerca de los límites de estabilidad, se hace necesario trabajar con herramientas que analizan el sistema en línea, pero los centros de control tienen una limitada capacidad de cómputo por cuestiones económicas y técnicas[2].

En la Universidad Industrial de Santander se ha seguido un camino en el análisis de estabilidad de gran perturbación. En [2] se resuelve el sistema de la forma explicita recurriendo a métodos numéricos para aumentar la convergencia. Aquí la matriz J, es una matriz de gran tamaño. Por lo tanto, se propuso en [3] trabajar la matriz de forma particionada operando por separado cada máquina, siendo esta modelada por una red neuronal. Se remitía a resolver la ecuación $I = YV$, esto hizo que la solución del sistema no fuera tan compleja. El problema surgía al cambiar la operación de la máquina lo que implicaba nuevamente entrenar la red neuronal. Por lo tanto, una solución a este inconveniente pueden ser los HPC (High Perfomance Computer), ya que la forma de procesar la información resulta ser mas eficiente en comparación al procesamiento realizado en la CPU. La técnica empleada para que el software se ejecute sobre estas máquinas, se le llama programación en paralelo, en el cuál se inicia paralelizando el algoritmo empleado. Dadas las diferentes formas de procesamiento paralelo, el empleado en este trabajo es el de paralelismo sobre los datos y la tecnología de las tarjetas gráficas de NVIDIA y su arquitectura unificada de dispositivos de cómputo (CUDA). La arquitectura está especializada en el procesamiento de matrices y vectores. Por tal motivo, estos dispositivos han sido empleados en aplicaciones de diferentes propósitos que no son unicamente los videojuegos. Para este caso, se empleará la tecnología al fenómeno de la estabilidad de máquina síncrona[5].

Actualmente los lenguajes que tienen soporte para trabajar en el dispositivo gráfico son C, C++, FORTRAN, JAVA, PYTHON y el empleado en este proyecto, MATLAB, ya que se especializa en el manejo de matrices y vectores.

Capítulo 3

Conceptos básicos del procesamiento paralelo

El lenguaje empleado en este proyecto es MATLAB, debido a que posee instrucciones definidas para trabajar matrices y vectores. A continuación se presenta lo que es GPGPU, y el modelo de programación para ejecutar software en la GPU. Además, se muestran las instrucciones para trabajar la GPU con MATLAB, y el concepto de paralelizar un algoritmo.

3.1. El inicio de las GPGPU

Dada la necesidad de ejecutar aplicaciones en el menor tiempo posible, la GPU se fue transformando hasta convertirse en un dispositivo de computo especializado en cálculo numérico altamente paralelo. La GPU en comparación con la CPU, destina mas ALU para procesar la información, y su arquitectura permite ejecutar de manera simultánea la misma tarea sobre una gran cantidad de datos [5].

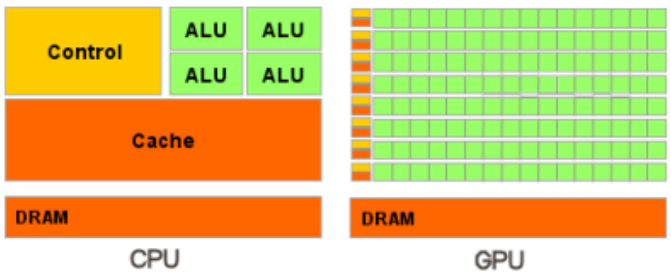


Figura 3.1: Comparación de arquitecturas [5].

Con los inicios CUDA en noviembre del 2006, que es un SDK[1] diseñado para aprovechar el poder de cálculo de las GPU de NVIDIA para diversas aplicaciones, se ofrece un modelo de programación escalable que aprovecha los núcleos de procesamiento. Entre sus propiedades se tienen, los grupos de hilos, memoria compartida y barrera de sincronización, que se agregan como extensiones al lenguaje. Estas

[1]Kit de desarrollo de software

propiedades proveen paralelismo de granularidad[2] fina de datos. Esto permite particionar un problema en sub-problemas mas pequeños que pueden resolver de manera independiente por bloques y que a la vez se pueden descomponer en partes mas pequeñas[5].

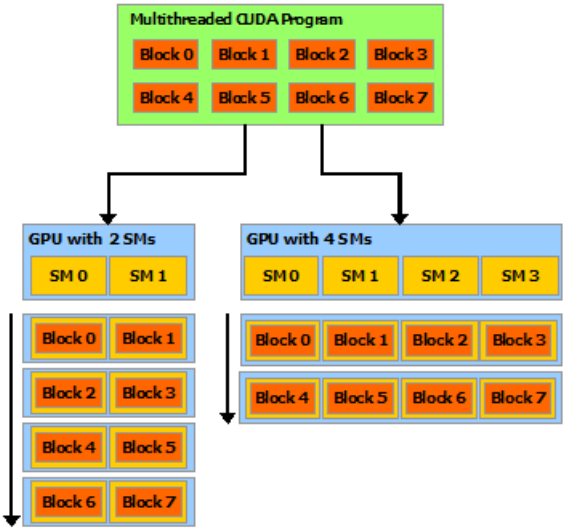


Figura 3.2: Escalabilidad automática [5].

[2]Número de tareas independientes en las que se puede descomponer un ejercicio

3.2. Modelo de programación

El esquema de programación, consiste en crear funciones encargadas de hacer la reserva de memoria de la RAM de la unidad gráfica (GPU) y transferirle la información desde la RAM de la unidad de procesamiento central (CPU) a la GPU. Luego se define una función, en donde el código se procesa en el dispositivo gráfico. Una vez procesada la información, se procede nuevamente a transferir de la RAM de la GPU a la RAM de la CPU.

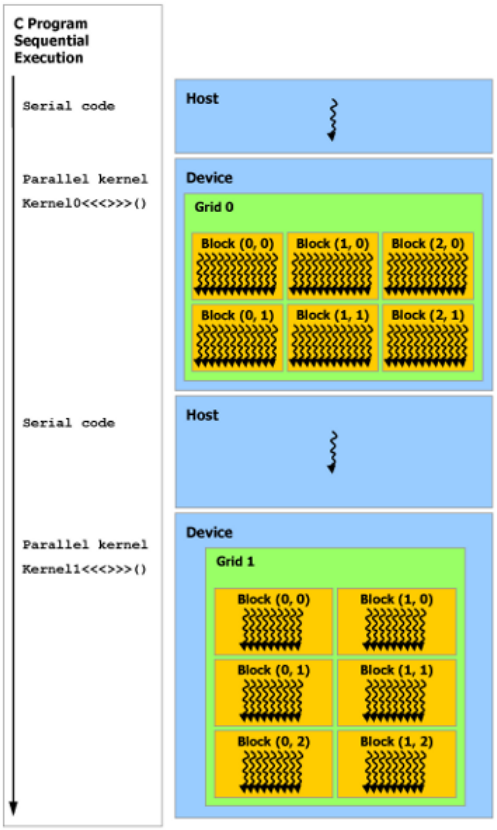


Figura 3.3: Programación heterogénea [5].

3.2.1. Procesamiento en paralelo en MATLAB

La plataforma en la que se desarrolla este proyecto es MATLAB, y se presenta el esquema de programación en esta plataforma. De lo expuesto anteriormente en 3.2, se entiende que es un modelo de programación heterogénea. Por lo tanto, MATLAB también requiere transferir información de la RAM de la CPU a la RAM de la GPU y viceversa. Para tal fin, las instrucciones de MATLAB encargadas de esta tarea son *gpuArray*, *gather*, *gpuDevice*, *reset* y *wait*. Aunque hay otras instrucciones, estas son las requeridas para la programación. A continuación se presentarán estas y otras instrucciones utilizadas para el cálculo en la GPU.

gpuArray: Transfiere la información de la memoria principal a la memoria del dispositivo.

gather: Transfiere la información de la memoria del dispositivo a la memoria principal.

reset: Borra la información de la RAM del dispositivo gráfico.

gpuDevice: Muestra las características del dispositivo o lo Selecciona.

gpuDeviceCount: Número de dispositivos presentes en el equipo.

arrayfun: Aplica la función a cada elemento de la matriz.

setConstantMemory: Mantiene variables en la memoria del dispositivo.

Las instrucciones anteriormente expuestas, en su mayoría, se encargan de la gestión de la información entre al memoria principal y la del dispositivo. Depende del programador conocer el problema a paralelizar, para desarrollar como es debido el respectivo código para que este se se ejecute adecuadamente en el dispositivo. En el capítulo 5 se explica como se desarrolló el software del proyecto para la ejecución en la GPU. Adicionalmente, MATLAB maneja las operaciones entre vectores y tiene soporte para trabajar en la GPU para diferentes funciones que tiene definidas. Álgunas de ellas son:

Tabla 3.1: Funciones definidas por MATLAB

abs	acos	acosh	acot
acoth	acsc	acsch	all
any	arrayfun	asec	asech
asin	asinh	atan	atan2
atanh	complex	cond	conj
conv	conv2	convn	cos
cosh	cot	coth	sin
csc	csch	length	find
fft	fft2	fftn	fix

3.2.2. Paralelizar un algoritmo

Debido a que las tecnologías de procesamiento paralelo son diferentes, es necesario adaptar el código de manera que se pueda aprovechar eficientemente el poder de procesamiento gráfico. Siendo una tecnología especializada para trabajo matricial y vectorial, se hace evidente que la información hay que adaptarla. Para tal fin, es importante conocer el problema a simular, ya que desde su análisis se puede determinar que parte del código es adaptable para ejecutarlo sobre el dispositivo gráfico.

Si se parte que un algoritmo, es una serie de pasos a seguir para realizar una tarea y que se deben desarrollar en determinado orden para poder cumplirla, entonces, se tiene que es una tarea netamente secuencial, y por lo tanto a la hora de paralelizar un algoritmo la secuencia en la que se hacen los cálculos no se altera en lo absoluto. Lo que se hace es que en cada paso, se desarrolle el mismo cálculo para todos los elementos de la matriz o vector de manera simultánea. En el presente caso son N máquinas, a las cuáles en cada paso se determinan los mismos parámetros de manera simultánea para las N máquinas del sistema.

A continuación se presenta un ejemplo, para resolver las raíces de una ecuación de segundo orden empleando la formula cuadrática $X_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. Los vectores cada uno con 1000 elementos.

1. Parámetros de entrada.
2. Cálculo del término asociado a x_1 para las primeras posiciones de los respectivos vectores.
3. Cálculo del término asociado a x_2 para las primeras posiciones de los respectivos vectores.
4. El proceso se repite dentro de un bucle *for* hasta recorrer las 1000 posiciones.

El código empleado tiene la siguiente forma:

```
% Tamaño del vector
tam=1e3;
%% Definición de vectores
% vectores en la memoria RAM de la CPU
ac =rand(tam,1);
bc =rand(tam,1);
cc =rand(tam,1);
% vectores en la memoria RAM de la GPU
ag=gpuArray(ac);
bg=gpuArray(bc);
cg=gpuArray(cc);

x1=[];
x2=[];
x1g=gpuArray(x1);
x2g=gpuArray(x2);
```

%% Programación del código de manera secuencial

$$
\begin{aligned}
&for \quad j = 1 : tam \\
&\quad y1(j) = (-bc(j) + (bc(j) \wedge (2) - 4 * ac(j) * cc(j)) \wedge (1/2))/(2 * ac(j)); \\
&\quad y2(j) = (-bc(j) - (bc(j) \wedge (2) - 4 * ac(j) * cc(j)) \wedge (1/2))/(2 * ac(j)); \\
&end
\end{aligned}
$$

Ejecutar la misma labor, paralelizando el código pero sin procesarlo en la GPU. Queda de la siguiente manera:
comp= find(bc<sqrt(4*ac.*cc));

$$
\begin{aligned}
&if \quad any(comp) \\
&\quad x1(comp, 1) = (-bc(comp, 1))./(2 * ac(comp, 1)) \\
&\quad +i * ((-bc(comp, 1). \wedge (2) \qquad (3.2.1) \\
&\quad +4. * ac(comp, 1). * cc(comp, 1)). \wedge (1/2))./(2. * ac(comp, 1)); \\
&\quad x2(comp, 1) = (-bc(comp, 1))./(2 * ac(comp, 1)) \\
&\quad -i * ((-bc(comp, 1). \wedge (2) \qquad (3.2.2) \\
&\quad +4. * ac(comp, 1). * cc(comp, 1)). \wedge (1/2))./(2 * ac(comp, 1)); \\
&end
\end{aligned}
$$

real= find(bc>=sqrt(4*ac.*cc));

$$
\begin{aligned}
&if \quad any(real) \\
&\quad x1(real, 1) = (-bc(real, 1))./(2 * ac(real, 1)) \\
&\quad +((bc(real, 1). \wedge (2) \qquad (3.2.3) \\
&\quad -4. * ac(real, 1). * cc(real, 1)). \wedge (1/2))./(2. * ac(real, 1)); \\
&\quad x2(real, 1) = (-bc(real, 1))./(2 * ac(real, 1)) \\
&\quad -((bc(real, 1). \wedge (2) \qquad (3.2.4) \\
&\quad -4. * ac(real, 1). * cc(real, 1)). \wedge (1/2))./(2 * ac(real, 1)); \\
&end
\end{aligned}
$$

El próximo código es muy similar al anterior, pero los vectores empleados están en la memoria de dispositivo gráfico. Por lo tanto el procesamiento se realiza en este dispositivo. La principal diferencia de estos dos códigos es la memoria en la que está guardada la información de los vectores.
%% Programación del código de empleando procesamiento paralelo

comp= find(bg<sqrt(4*ag.*cg));

$$
\begin{aligned}
&\textit{if} \quad any(comp) \\
&\quad x1g(comp,1) = (-bg(comp,1))./(2 * ag(comp,1)) \\
&\quad +i * ((-bg(comp,1). \wedge (2) \qquad (3.2.5) \\
&\quad +4. * ag(comp,1). * cg(comp,1)). \wedge (1/2)).(2. * ag(comp,1)); \\
&\quad x2g(comp,1) = (-bg(comp,1))./(2 * ag(comp,1)) \\
&\quad -i * ((-bg(comp,1). \wedge (2) \qquad (3.2.6) \\
&\quad +4. * ag(comp,1). * cg(comp,1)). \wedge (1/2)).(2 * ag(comp,1)); \\
&\textit{end}
\end{aligned}
$$

real= find(bg>=sqrt(4*ag.*cg));

$$
\begin{aligned}
&\textit{if} \quad any(real) \\
&\quad x1g(real,1) = (-bg(real,1))./(2 * ag(real,1)) \\
&\quad +((bg(real,1). \wedge (2) \qquad (3.2.7) \\
&\quad -4. * ag(real,1). * cg(real,1)). \wedge (1/2)).(2. * ag(real,1)); \\
&\quad x2g(real,1) = (-bg(real,1))./(2 * ag(real,1)) \\
&\quad -((bg(real,1). \wedge (2) \qquad (3.2.8) \\
&\quad -4. * ag(real,1). * cg(real,1)). \wedge (1/2)).(2 * ag(real,1)); \\
&\textit{end}
\end{aligned}
$$

$x1c = gather(x1g);$
$x2c = gather(x2g);$

Se tiene que el orden en el que se realiza dicha labor no puede intercambiarse o alterarse, lo que se realiza es la tarea para la N posiciones al mismo tiempo. Por lo tanto, para un código de programación, paralelizar un algoritmo consiste en encontrar esas tareas que se realizan con un bucle y realizar la misma actividad pero eliminando el bucle. Esta modificación se puede realizar cuando se tienen operaciones entre matrices o vectores.

Capítulo 4

Metodología

Este capítulo se divide en dos partes, la primera trata sobre el desarrollo de las ecuaciones que modelan la dinámica de la máquina síncrona ante una gran perturbación y el método empleado en su solución. La segunda, es el desarrollo del software y la metodología empleada para que el código se ejecute correctamente sobre el dispositivo gráfico.

4.1. Solución de las ecuaciones

En esta sección se presentan las fórmulas que modelan la dinámica del rotor y el conexionado del estator a la red. También, se presenta su respectiva solución para extraer las expresiones que se emplearán en el desarrollo del software.

4.1.1. Solución de las ecuaciones del rotor

Para empezar, de la sección (2.2.2) se procede a despejar i_d e i_q de las ecuaciones (2.2.19) y (2.2.20). El sistema a resolver es del estilo $x = Ab$ y se presenta a continuación.

$$\begin{bmatrix} e_d - E_d^{"} \\ e_q - E_q^{"} \end{bmatrix} = \begin{bmatrix} -R_a & X_q^{"} \\ -X_d^{"} & -R_a \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{4.1.1}$$

Donde A es la matriz del sistema. Se resuelve el sistema de ecuaciones por medio del cálculo del determinante de A y su inversa.

$$det(A) = R_a^2 + X_d^{"} X_q^{"} \tag{4.1.2}$$

$$inv(A) = \frac{\begin{bmatrix} -R_a & -X_q^{"} \\ X_d^{"} & -R_a \end{bmatrix}}{det(A)} \tag{4.1.3}$$

Las corrientes resultantes al resolver el sistema son:

$$i_d = -\frac{R_a}{det(A)} e_d + \frac{R_a}{det(A)} E_d^{"} - \frac{R_a}{det(A)} e_q + \frac{X_q^{"}}{det(A)} E_q^{"} \tag{4.1.4}$$

$$i_q = \frac{X_d^{"}}{det(A)} e_d - \frac{X_d^{"}}{det(A)} E_d^{"} - \frac{R_a}{det(A)} e_q + \frac{R_a}{det(A)} E_q^{"} \tag{4.1.5}$$

los términos $\varpi L_{ads}^{"} = X_{ads}^{"}$ y $\varpi L_{aqs}^{"} = X_{aqs}^{"}$ se reemplazan en las ecuaciones (2.2.21) y (2.2.22), las expresiónes resultantes son:

$$E_d^{"} = -\frac{X_{aqs}^{"}}{L_{1q}} \psi_{1q} - \frac{X_{aqs}^{"}}{L_{2q}} \psi_{2q} \tag{4.1.6}$$

$$E_q^{"} = \frac{X_{ads}^{"}}{L_{fd}} \psi_{fd} + \frac{X_{ads}^{"}}{L_{1d}} \psi_{1d} \tag{4.1.7}$$

Por último, las ecuaciones (2.2.19), (2.2.20),(4.1.6) y (4.1.7) se reemplazan en las ecuaciones (4.1.4) y (4.1.5). Las expresiones resultantes son:

$$\begin{aligned} i_d &= \frac{X_q^{"} X_{ads}^{"}}{det(A) * L_{fd}} \psi_{fd} + \frac{X_q^{"} X_{ads}^{"}}{det(A) * L_{1d}} \psi_{1d} - \frac{R_a X_{qs}^{"}}{det(A) * L_{1q}} \psi_{1q} - \frac{R_a X_{qs}^{"}}{det(A) * L_{2q}} \psi_{2q} \\ &- \frac{R_a}{det(A)} E_R \sin(\delta) - \frac{X_q^{"}}{det(A)} E_R \cos(\delta) + \frac{R_a}{det(A)} E_I \cos(\delta) - \frac{X_q^{"}}{det(A)} E_I \sin(\delta) \end{aligned} \tag{4.1.8}$$

$$\begin{aligned} i_q &= \frac{R_a^{"} X_{ads}^{"}}{det(A) * L_{fd}} \psi_{fd} + \frac{R_a^{"} X_{ads}^{"}}{det(A) * L_{1d}} \psi_{1d} - \frac{X_d X_{qs}^{"}}{det(A) * L_{1q}} \psi_{1q} - \frac{X_d X_{qs}^{"}}{det(A) * L_{2q}} \psi_{2q} \\ &- \frac{R_a}{det(A)} E_R \cos(\delta) - \frac{X_q^{"}}{det(A)} E_R \sin(\delta) + \frac{R_a}{det(A)} E_I \sin(\delta) - \frac{X_q^{"}}{det(A)} E_I \cos(\delta) \end{aligned} \tag{4.1.9}$$

Luego de calculadas las corrientes, se procede a reemplazarlas en las ecuaciones (2.2.14) y (2.2.15). Luego se efectúa la factorización y organización correspondiente de las expresiones. El resultado que se obtiene es:

$$\begin{aligned} \psi_{ad} &= \left(\frac{L_{ads}^{"}}{L_{fd}} - \frac{L_{ads}^{"} X_q^{"} X_{ads}^{"}}{det(A) * L_{fd}} \right) \psi_{fd} + \left(\frac{L_{ads}^{"}}{L_{1d}} - \frac{L_{ads}^{"} X_q^{"} X_{ads}^{"}}{det(A) * L_{d1}} \right) \psi_{1d} \\ &+ \left(\frac{L_{ads}^{"} R_a X_{aqs}^{"}}{det(A) * L_{1q}} \right) \psi_{1q} + \left(\frac{L_{ads}^{"} R_a X_{aqs}^{"}}{det(A) * L_{2q}} \right) \psi_{2q} + \frac{L_{ads}^{"} R_a}{det(A)} E_R \sin(\delta) \\ &+ \frac{L_{ads}^{"} X_q^{"}}{det(A)} E_R \cos(\delta) + \frac{L_{ads}^{"} R_a}{det(A)} E_I \cos(\delta) + \frac{L_{ads}^{"} X_q^{"}}{det(A)} E_I \sin(\delta) \end{aligned} \tag{4.1.10}$$

y

$$\begin{aligned} \psi_{aq} &= -\frac{L_{aqs}^{"} R_a^{"} X_{ads}^{"}}{det(A) * L_{fd}} \psi_{fd} - \frac{L_{aqs}^{"} R_a X_{ads}^{"}}{det(A) * L_{d1}} \psi_{1d} + \left(\frac{L_{aqs}^{"}}{L_{1d}} - \frac{L_{aqs}^{"} X_d^{"} X_{aqs}^{"}}{det(A) * L_{1q}} \right) \psi_{1q} \\ &+ \left(\frac{L_{aqs}^{"}}{L_{1d}} - \frac{L_{aqs}^{"} X_d^{"} X_{aqs}^{"}}{det(A) * L_{2q}} \right) \psi_{2q} + \frac{L_{aqs}^{"} R_a}{det(A)} E_R \cos(\delta) - \frac{L_{aqs}^{"} X_d^{"}}{det(A)} E_R \sin(\delta) \\ &+ \frac{L_{aqs}^{"} R_a}{det(A)} E_I \sin(\delta) + \frac{L_{aqs}^{"} X_d^{"}}{det(A)} E_I \cos(\delta) \end{aligned} \tag{4.1.11}$$

Las anteriores expresiones se utilizan en las ecuaciones del circuito del rotor (2.2.2).

$$
\begin{aligned}
p\psi_{fd} &= \omega_0 e_{fd} + \frac{\omega_0 R_{fd}}{L_{fd}}\left[\left(\frac{L^{"}_{ads}}{L_{fd}} - 1 - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{fd}}\right)\psi_{fd}\right. \\
&+ \left(\frac{L^{"}_{ads}}{L_{1d}} - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{1d}}\right)\psi_{1d} + \frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L_{1q}}\psi_{1q} + \frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L_{2q}}\psi_{2q} \\
&+ \frac{L^{"}_{ads}R_a E_R}{det(A)}\sin(\delta) + \frac{L^{"}_{ads}X^{"}_{q}E_R}{det(A)}\cos(\delta) \\
&- \left.\frac{L^{"}_{ads}R_a E_I}{det(A)}\cos(\delta) + \frac{L^{"}_{ads}X^{"}_{q}E_I}{det(A)}\sin(\delta)\right]
\end{aligned}
\tag{4.1.12}
$$

$$
\begin{aligned}
p\psi_{1d} &= \frac{\omega_0 R_{1d}}{L_{1d}}\left[\left(\frac{L^{"}_{ads}}{L_{fd}} - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{fd}}\right)\psi_{fd} + \left(\frac{L^{"}_{ads}}{L_{1d}} - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{1d}} - 1\right)\psi_{1d}\right. \\
&+ \frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L_{1q}}\psi_{1q} + \frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L_{2q}}\psi_{2q} \\
&+ \frac{L^{"}_{ads}R_a E_R}{det(A)}\sin(\delta) + \frac{L^{"}_{ads}X^{"}_{q}E_R}{det(A)}\cos(\delta) \\
&- \left.\frac{L^{"}_{ads}R_a E_I}{det(A)}\cos(\delta) + \frac{L^{"}_{ads}X^{"}_{q}E_I}{det(A)}\sin(\delta)\right]
\end{aligned}
\tag{4.1.13}
$$

$$
\begin{aligned}
p\psi_{1q} &= \frac{\omega_0 R_{1d}}{L_{1q}}\left[-\frac{L^{"}_{aqs}R_a X^{"}_{ads}}{det(A) * L_{fd}}\psi_{fd} - \frac{L^{"}_{aqs}R_a X^{"}_{ads}}{det(A) * L_{1d}}\psi_{1d}\right. \\
&+ \left(\frac{L_{aqs}}{L_{1q}} - 1 - \frac{L^{"}_{aqs}X^{"}_{d}X^{"}_{aqs}}{det(A) * L_{1q}}\right)\psi_{1q} + \left(\frac{L^{"}_{aqs}}{L_{1q}} - \frac{L^{"}_{aqs}X^{"}_{d}X^{"}_{aqs}}{det(A) * L_{2q}}\right)\psi_{2q} \\
&+ \frac{L^{"}_{aqs}R_a E_R}{det(A)}\cos(\delta) - \frac{L^{"}_{aqs}X^{"}_{d}E_R}{det(A)}\sin(\delta) \\
&+ \left.\frac{L^{"}_{aqs}R_a E_I}{det(A)}\sin(\delta) + \frac{L^{"}_{aqs}X^{"}_{d}E_I}{det(A)}\cos(\delta)\right]
\end{aligned}
\tag{4.1.14}
$$

$$
\begin{aligned}
p\psi_{2q} &= \frac{\omega_0 R_{2d}}{L_{2q}}\left[-\frac{L^{"}_{aqs}R_a X^{"}_{ads}}{det(A) * L_{fd}}\psi_{fd} - \frac{L^{"}_{aqs}R_a X^{"}_{ads}}{det(A) * L_{1d}}\psi_{1d}\right. \\
&+ \left(\frac{L_{aqs}}{L_{1q}} - \frac{L^{"}_{aqs}X^{"}_{d}X^{"}_{aqs}}{det(A) * L_{1q}}\right)\psi_{1q} + \left(\frac{L^{"}_{aqs}}{L_{1q}} - 1 - \frac{L^{"}_{aqs}X^{"}_{d}X^{"}_{aqs}}{det(A) * L_{2q}}\right)\psi_{2q} \\
&+ \frac{L^{"}_{aqs}R_a E_R}{det(A)}\cos(\delta) - \frac{L^{"}_{aqs}X^{"}_{d}E_R}{det(A)}\sin(\delta) \\
&+ \left.\frac{L^{"}_{aqs}R_a E_I}{det(A)}\sin(\delta) + \frac{L^{"}_{aqs}X^{"}_{d}E_I}{det(A)}\cos(\delta)\right]
\end{aligned}
\tag{4.1.15}
$$

$$
p\delta = \omega_o \Delta\omega_r \tag{4.1.16}
$$

4.1.2. Solución de las ecuaciones del estator

Los valores de $E_d^{''}$ y $E_q^{''}$ hallados de las ecuaciones (4.1.6) y (4.1.7), se reemplazan en (2.2.35) y (2.2.36) para hallar $E_R^{''}$ y $E_I^{''}$. Estos valores se reemplazan en la ecuación (2.2.29), dado que E_R y E_I son tensiones de entrada, se puede calcular I_R e I_I, dando como resultado las siguientes expresiones:

$$\begin{aligned} I_R &= -\frac{(X_q^{"} - X_d^{"})}{det(A)}\sin(\delta)\cos(\delta)E_R - \frac{R_a}{det(A)}E_R - \frac{X_d^{"}\cos^2(\delta) + X_q^{"}sin^2(\delta)}{det(A)}E_I \\ &+ \frac{R_a X_{ads}^{"}}{det(A) * L_{fd}}\psi_{fd}\cos(\delta) + \frac{R_a X_{ads}^{"}}{det(A) * L_{1d}}\psi_{1d}\cos(\delta) - \frac{R_a X_{aqs}^{"}}{det(A) * L_{1q}}\psi_{1q}\sin(\delta) \\ &- \frac{R_a X_{aqs}^{"}}{det(A) * L_{2q}}\psi_{2q}\sin(\delta) + \frac{X_{RI} X_{ads}^{"}}{det(A) * L_{fd}}\psi_{fd}\sin(\delta) + \frac{X_{RI} X_{ads}^{"}}{det(A) * L_{1d}}\psi_{1d}\sin(\delta) \\ &+ \frac{X_{RI} X_{aqs}^{"}}{det(A) * L_{1q}}\psi_{1q}\cos(\delta) + \frac{X_{RI} X_{aqs}^{"}}{det(A) * L_{2q}}\psi_{2q}\cos(\delta) \end{aligned} \tag{4.1.17}$$

$$\begin{aligned} I_I &= \frac{X_d^{"}sin^2(\delta) + X_q^{"}\cos^2(\delta)}{det(A)}E_R - \frac{X_{IR} X_{ads}^{"}}{det(A) * L_{fd}}\psi_{fd}\cos(\delta) - \frac{X_{IR} X_{ads}^{"}}{det(A) * L_{1d}}\psi_{1d}\cos(\delta) \\ &- \frac{X_{IR} X_{aqs}^{"}}{det(A) * L_{1q}}\psi_{1q}\sin(\delta) + \frac{X_{IR} X_{aqs}^{"}}{det(A) * L_{2q}}\psi_{2q}\sin(\delta) - \frac{(X_d^{"} - X_q^{"})}{det(A)}\sin(\delta)\cos(\delta)E_I \\ &- \frac{R_a}{det(A)}E_I + \frac{R_a X_{ads}^{"}}{det(A) * L_{fd}}\psi_{fd}\sin(\delta) + \frac{R_a X_{ads}^{"}}{det(A) * L_{1d}}\psi_{1d}\sin(\delta) \\ &+ \frac{R_a X_{aqs}^{"}}{det(A) * L_{1q}}\psi_{1q}\cos(\delta) + \frac{R_a X_{aqs}^{"}}{det(A) * L_{2q}}\psi_{2q}\cos(\delta) \end{aligned} \tag{4.1.18}$$

4.1.3. Solución de las ecuaciones mediante integración implícita

Las ecuaciones (4.1.12) a la (4.1.16), se resuelven por medio del método de integración implícita. Las ecuaciones resultantes son:

$$
\begin{aligned}
f_1 &= \psi_{fd(n+1)} - \psi_{fd(n)} - \frac{\Delta t * \omega_0 * e_{fd}}{2} \\
&- \frac{\omega_0 * R_{fd} * \Delta t}{2 * L_{fd}} \Bigg[\left(\frac{L_{ads}^{"}}{L_{fd}} - 1 - \frac{L_{ads}^{"} X_q^{"} X_{ads}^{"}}{det(A) * L_{fd}} \right) \left(\psi_{fd(n+1)} + \psi_{fd(n)} \right) \\
&+ \left(\frac{L_a^{"} ds}{L1d} - \frac{L_{ads}^{"} X_q^{"} X_{ads}^{"}}{det(A) * L_{1d}} \right) \left(\psi_{1d(n+1)} + \psi_{1d(n+1)} \right) + \frac{L_{ads}^{"} R_a X_{aqs}^{"}}{det(A) * L_{1q}} \left(\psi_{1q(n+1)} + \psi_{1q(n)} \right) \\
&+ \frac{L_{ads}^{"} R_a X_{aqs}^{"}}{det(A) * L_{2q}} \left(\psi_{2q(n+1)} + \psi_{2q(n)} \right) + \frac{L_{ads}^{"} R_a}{det(A)} \left(E_{R(n+1)} \sin(\delta_{n+1}) + E_{R(n)} \sin(\delta_n) \right) \\
&+ \frac{L_{ads}^{"} X_q^{"}}{det(A)} \left(E_{R(n+1)} \cos(\delta_{n+1}) + E_{R(n)} \cos(\delta_n) \right) \\
&- \frac{L_{ads}^{"} R_a}{det(A)} \left(E_{I(n+1)} cos(\delta_{n+1}) + E_{I(n)} cos(\delta_n) \right) \\
&+ \frac{L_{ads}^{"} X_q^{"}}{det(A)} \left(E_{I(n+1)} sin(\delta_{n+1}) + E_{I(n)} sin(\delta_n) \right) \Bigg] \qquad (4.1.19) \\
f_2 &= \psi_{1d(n+1)} - \psi_{1d(n)} - \frac{\omega_0 R_{1d} \Delta t}{2 * L_{1d}} \Bigg[\left(\frac{L_{ads}^{"}}{L_{fd}} - \frac{L_{ads}^{"} X_q^{"} X_{ads}^{"}}{det(A) * L_{fd}} \right) \left(\psi_{fd(n+1)} + \psi_{fd(n)} \right) \\
&+ \left(\frac{L_{ads}^{"}}{L_{1d}} - \frac{L_{ads}^{"} X_q^{"} X_{ads}^{"}}{det(A) * L_{1d}} - 1 \right) \left(\psi_{1d(n+1)} + \psi_{1d(n)} \right) + \frac{L_{ads}^{"} R_a X_{aqs}^{"}}{det(A) * L_{1q}} \left(\psi_{1q(n+1)} + \psi_{1q(n)} \right) \\
&+ \frac{L_{ads}^{"} R_a X_{aqs}^{"}}{det(A) * L_{2q}} \left(\psi_{2q(n+1)} + \psi_{2q(n)} \right) + \frac{L_{ads}^{"} R_a}{det(A)} \left(E_{R(n+1)} \sin(\delta_{(n+1)}) + E_{R(n)} \sin(\delta_{(n)}) \right) \\
&+ \frac{L_{ads}^{"} X_q^{"}}{det(A)} \left(E_{R(n+1)} \cos(\delta_{n+1}) + E_{R(n)} \cos(\delta_n) \right) \\
&- \frac{L_{ads}^{"} R_a}{det(A)} \left(E_{I(n+1)} \cos(\delta_{n+1}) + E_{I(n)} \cos(\delta_n) \right) \\
&+ \frac{L_{ads}^{"} X_q^{"}}{det(A)} \left(E_{I(n+1)} \sin(\delta_{n+1}) + E_{I(n)} \sin(\delta_n) \right) \Bigg] \qquad (4.1.20)
\end{aligned}
$$

$$
\begin{aligned}
f_3 &= \psi_{1q(n+1)} - \psi_{1q(n)} - \frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\Bigg[- \frac{L^{"}_{aqs} R_a X^{"}_{ads}}{det(A) * L_{fd}}\Big(\psi_{fd(n+1)} + \psi_{fd(n)}\Big) \\
&- \frac{L^{"}_{aqs} R_a X^{"}_{ads}}{det(A) * L_{1d}}\Big(\psi_{1d(n+1)} + \psi_{1d(n)}\Big) \\
&+ \left(\frac{L^{"}_{aqs}}{L_{1q}} - 1 - \frac{L^{"}_{aqs} X^{"}_{d} X^{"}_{aqs}}{det(A) * L_{1q}}\right)\Big(\psi_{1q(n+1)} + \psi_{1q(n)}\Big) \\
&+ \left(\frac{L^{"}_{aqs}}{L_{1q}} - \frac{L^{"}_{aqs} X^{"}_{d} X^{"}_{aqs}}{det(A) * L_{2q}}\right)\Big(\psi_{2q(n+1)} + \psi_{2q(n)}\Big) \\
&+ \frac{L^{"}_{aqs} R_a}{det(A)}\Big(E_{R(n+1)}\cos(\delta_{n+1}) + E_{R(n)}\cos(\delta_n)\Big) \\
&- \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)}\Big(E_{R(n+1)}\sin(\delta_{n+1}) + E_{R(n)}\sin(\delta_n)\Big) \\
&+ \frac{L^{"}_{aqs} R_a}{det(A)}\Big(E_{I(n+1)}\sin(\delta_{n+1}) + E_{I(n)}\sin(\delta_n)\Big) \\
&+ \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)}\Big(E_{I(n+1)}\cos(\delta_{n+1}) + E_{I(n)}\cos(\delta_n)\Big)\Bigg] \qquad (4.1.21)
\end{aligned}
$$

$$
\begin{aligned}
f_4 &= \psi_{2q(n+1)} - \psi_{2q(n)} - \frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\Bigg[- \frac{L^{"}_{aqs} R_a X^{"}_{ads}}{det(A) * L_{fd}}\Big(\psi_{fd(n+1)} + \psi_{fd(n)}\Big) \\
&- \frac{L^{"}_{aqs} R_a X^{"}_{ads}}{det(A) * L_{1d}}\Big(\psi_{1d(n+1)} + \psi_{1d(n)}\Big) + \left(\frac{L^{"}_{aqs}}{L_{1q}} - \frac{L^{"}_{aqs} X^{"}_{d} X^{"}_{aqs}}{det(A) * L_{1q}}\right)\Big(\psi_{1q(n+1)} + \psi_{1q(n)}\Big) \\
&+ \left(\frac{L^{"}_{aqs}}{L_{1q}} - 1 - \frac{L^{"}_{aqs} X^{"}_{d} X^{"}_{aqs}}{det(A) * L_{2q}}\right)\Big(\psi_{2q(n+1)} + \psi_{2q(n+1)}\Big) \\
&+ \frac{L^{"}_{aqs} R_a}{det(A)}\Big(E_{R(n+1)}\cos(\delta_{n+1}) + E_{R(n)}\cos(\delta_n)\Big) \\
&- \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)}\Big(E_{R(n+1)}\sin(\delta_{n+1}) + E_{R(n)}\sin(\delta_n)\Big) \\
&+ \frac{L^{"}_{aqs} R_a}{det(A)}\Big(E_{I(n+1)}\sin(\delta_{n+1}) + E_{I(n)}\sin(\delta_n)\Big) \\
&+ \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)}\Big(E_{I(n+1)}\cos(\delta_{n+1}) + E_{I(n)}\cos(\delta_n)\Big)\Bigg] \qquad (4.1.22)
\end{aligned}
$$

$$
f_5 = \delta_(n+1) - \delta(n) - \frac{\omega_o \Delta t}{2}\Big(\Delta\omega_{r(n+1)} + \Delta\omega_{r(n)}\Big) \qquad (4.1.23)
$$

Las ecuaciones (4.1.17), (4.1.18) hacen parte de la expresión (4.1.25), el método de integración implícita quedan de la siguiente manera.

$$
G(x_{n+1}, V_{n+1}) = Y_n V_{n+1} - I(x_{n+1}, V_{n+1}) \qquad (4.1.24)
$$

$$
\begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} Y_{nR} & -Y_{nI} \\ Y_{nI} & Y_{nR} \end{bmatrix}\begin{bmatrix} E_R \\ E_I \end{bmatrix} - \begin{bmatrix} I_R \\ I_I \end{bmatrix} \qquad (4.1.25)
$$

Donde los elementos de la matriz Y_N corresponden a un arreglo de los componentes de la parte real e imaginaria de la matriz Y_{barra}. De igual forma el vector V son las

tensiones de las diferentes barras, organizadas en sus componentes real e imaginaria.

$$
\begin{aligned}
g_1 &= E_{R(n+1)}Y_{NR} - E_{I(n+1)}Y_{NI} - \frac{X_{RI}X_{ads}^{"}}{det(A)*L_{fd}}\psi_{fd(n+1)}\sin(\delta_{n+1}) \\
&- \frac{R_aX_{ads}^{"}}{det(A)*L_{fd}}\psi_{fd(n+1)}\cos(\delta_{n+1}) \\
&- \frac{X_{RI}X_{ads}^{"}}{det(A)*L_{1d}}\psi_{1d(n+1)}\sin(\delta_{n+1}) - \frac{R_aX_{ads}^{"}}{det(A)*L_{1d}}\psi_{1d(n+1)}\cos(\delta_{n+1}) \\
&+ \frac{R_aX_{aqs}^{"}}{det(A)*L_{1q}}\psi_{1q(n+1)}\sin(\delta_{n+1}) - \frac{X_{RI}X_{aqs}^{"}}{det(A)*L_{1q}}\psi_{1q(n+1)}\cos(\delta_{n+1}) \\
&+ \frac{R_aX_{aqs}^{"}}{det(A)*L_{2q}}\psi_{2q(n+1)}\sin(\delta_{n+1}) - \frac{X_{RI}X_{aqs}^{"}}{det(A)*L_{2q}}\psi_{2q(n+1)}\cos(\delta_{n+1}) \\
&+ \frac{(X_q^{"} - X_d^{"})}{2*det(A)}E_{R(n+1)}\sin(2\delta_{n+1}) + \frac{R_aE_{R(n+1)}}{det(A)} \\
&+ \frac{X_d^{"}\cos^2(\delta_{n+1}) + X_q^{"}\sin^2(\delta_{n+1})}{det(A)}E_{I(n+1)} \qquad (4.1.26)
\end{aligned}
$$

$$
\begin{aligned}
g_2 &= E_{R(n+1)}Y_{NI} + E_{I(n+1)}Y_{NR} - \frac{R_aX_{ads}^{"}}{det(A)*L_{fd}}\psi_{fd(n+1)}\sin(\delta_{n+1}) \\
&+ \frac{X_{IR}X_{ads}^{"}}{det(A)*L_{fd}}\psi_{fd(n+1)}\cos(\delta_{n+1}) - \frac{R_aX_{ads}^{"}}{det(A)*L_{1d}}\psi_{1d(n+1)}\sin(\delta_{n+1}) \\
&+ \frac{X_{IR}X_{ads}^{"}}{det(A)*L_{1d}}\psi_{1d(n+1)}\cos(\delta_{n+1}) - \frac{X_{IR}X_{aqs}^{"}}{det(A)*L1q}\psi_{1q(n+1)}\sin(\delta_{n+1}) \\
&- \frac{R_aX_{aqs}^{"}}{det(A)*L_{1q}}\psi_{1q(n+1)}\cos(\delta_{n+1}) - \frac{X_{IR}X_{aqs}^{"}}{det(A)*L_{2q}}\psi_{2q(n+1)}\sin(\delta_{n+1}) \\
&- \frac{R_aX_{ads}^{"}}{det(A)*L_{1d}}\psi_{2q(n+1)}\cos(\delta_{n+1}) + \frac{R_a}{det(A)}E_{I(n+1)} \\
&- \frac{X_d^{"}\sin^2(\delta_{n+1}) + X_q^{"}\cos^2(\delta_{n+1})}{det(A)}E_{R(n+1)} \\
&+ \frac{X_d^{"} - X_q^{"}}{2*det(A)}E_{I(n+1)}\sin(2\delta_{n+1}) \qquad (4.1.27)
\end{aligned}
$$

En las ecuaciones de I_D, se consideran la corrientes debidas a las barras infinitas y carga estáticas.

4.1.4. Representación del jacobiano en términos de derivada parciales

Las ecuaciones (4.1.19) a la (4.1.23) se derivan parcialmente con respecto a las expresiones (4.1.12) a la (4.1.16) para armar el jacobiano como se expresa en la ecuación (2.3.7). Se empieza con los términos de A_D.

$$
\frac{\partial f_1}{\partial x_1} = A_{11} = 1 - \frac{\omega_0 R_{fd}\Delta t}{2*L_{fd}}\left[\left(\frac{L_{ads}^{"}}{L_{fd}} - 1 - \frac{L_{ads}^{"}X_q^{"}X_{ads}^{"}}{det(A)*L_{fd}}\right)\right] \qquad (4.1.28)
$$

$$\frac{\partial f_1}{\partial x_2} = A_{12} = -\left(\frac{\omega_0 R_{fd}\Delta t}{2 * L_{fd}}\right)\left(\frac{L^{"}_{ads}}{L_{1d}} - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{1d}}\right) \tag{4.1.29}$$

$$\frac{\partial f_1}{\partial x_3} = A_{13} = -\left(\frac{\omega_0 R_{fd}\Delta t}{2 * L_{fd}}\right)\left(\frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L_{1q}}\right) \tag{4.1.30}$$

$$\frac{\partial f_1}{\partial x_4} = A_{14} = -\left(\frac{\omega_0 R_{fd}\Delta t}{2 * L_{fd}}\right)\left(\frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L^{"}_{2q}}\right) \tag{4.1.31}$$

$$\frac{\partial f_1}{\partial x_5} = A_{15} = -\frac{\omega_0 R_{fd}\Delta t}{2 * L_{fd}}\left(\frac{L^{"}_{ads}R_a}{det(A)}E_{R(n+1)}\cos(\delta_{n+1}) - \frac{L^{"}_{ads}X^{"}_{q}}{det(A)}E_{R(n+1)}\sin(\delta_{n+1})\right.$$
$$\left. + \frac{L^{"}_{ads}X^{"}_{q}}{det(A)}E_{I(n+1)}\cos(\delta_{n+1}) + \frac{L^{"}_{ads}R_a}{det(A)}E_{I(n+1)}\sin(\delta_{n+1})\right) \tag{4.1.32}$$

$$\frac{\partial f_2}{\partial x_1} = A_{21} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(\frac{L^{"}_{ads}}{L_{fd}} - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{fd}}\right) \tag{4.1.33}$$

$$\frac{\partial f_2}{\partial x_2} = A_{22} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(\frac{L^{"}_{ads}}{L_{1d}} - \frac{L^{"}_{ads}X^{"}_{q}X^{"}_{ads}}{det(A) * L_{1d}} - 1\right) \tag{4.1.34}$$

$$\frac{\partial f_2}{\partial x_3} = A_{23} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(\frac{L^{"}_{ads}R_a X^{"}_{aqs}}{det(A) * L_{1q}}\right) \tag{4.1.35}$$

$$\frac{\partial f_2}{\partial x_4} = A_{24} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(\frac{L^{"}_{ads}R_a X^{"}_{aqs}}{L_{2q}det(A)}\right) \tag{4.1.36}$$

$$\frac{\partial f_2}{\partial x_5} = A_{25} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(\frac{L^{"}_{ads}R_a}{det(A)}E_{R(n+1)}\cos(\delta_{n+1}) - \frac{L^{"}_{ads}X^{"}_{q}}{det(A)}E_{R(n+1)}\sin(\delta_{n+1})\right.$$
$$\left. + \frac{L^{"}_{ads}R_a}{det(A)}E_{I(n+1)}\sin(\delta_{n+1}) + \frac{L^{"}_{ads}X^{"}_{q}}{det(A)}E_{n+1}\cos(\delta_{n+1})\right) \tag{4.1.37}$$

$$\frac{\partial f_3}{\partial x_1} = A_{31} = \frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(\frac{L^{"}_{aqs}R_a X^{"}_{ads}}{det(A) * L_{fd}}\right) \tag{4.1.38}$$

$$\frac{\partial f_3}{\partial x_2} = A_{32} = \frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(\frac{L^{"}_{aqs}R_a X^{"}_{ads}}{det(A) * L_{fd}}\right) \tag{4.1.39}$$

$$\frac{\partial f_3}{\partial x_3} = A_{33} = -\frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(\frac{L^{"}_{aqs}}{L_{1q}} - 1 - \frac{L^{"}_{aqs}X^{"}_{d}X^{"}_{aqs}}{det(A) * L_{1q}}\right) \tag{4.1.40}$$

$$\frac{\partial f_3}{\partial x_4} = A_{34} = -\frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(\frac{L^{"}_{aqs}}{L_{1q}} - \frac{L^{"}_{aqs}X^{"}_{d}X^{"}_{aqs}}{det(A) * L_{2q}}\right) \tag{4.1.41}$$

$$\frac{\partial f_3}{\partial x_5} = A_{35} = -\frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(-\frac{L^{"}_{aqs}R_a}{det(A)}E_{R(n+1)}\sin(\delta_{n+1}) - \frac{L^{"}_{aqs}X^{"}_{d}}{det(A)}E_{R(n+1)}\cos(\delta_{n+1})\right.$$
$$\left. + \frac{L^{"}_{aqs}R_a}{det(A)}E_{I(n+1)}\cos(\delta_{n+1}) - \frac{L^{"}_{aqs}X^{"}_{d}}{det(A)}E_{I(n+1)}\sin(\delta_{n+1})\right) \tag{4.1.42}$$

$$\frac{\partial f_4}{\partial x_1} = A_{41} = \frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\left(\frac{L^{"}_{aqs} R_a X^{"}_{ads}}{det(A) * L_{fd}}\right) \quad (4.1.43)$$

$$\frac{\partial f_4}{\partial x_2} = A_{42} = \frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\left(\frac{L^{"}_{aqs} R_a X^{"}_{ads}}{det(A) * L_{1d}}\right) \quad (4.1.44)$$

$$\frac{\partial f_4}{\partial x_3} = A_{43} = -\frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\left(\frac{L^{"}_{aqs}}{L_{1q}} - \frac{L^{"}_{aqs} X^{"}_{d} X^{"}_{aqs}}{det(A) * L_{1q}}\right) \quad (4.1.45)$$

$$\frac{\partial f_4}{\partial x_4} = A_{44} = -\frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\left(\frac{L^{"}_{aqs}}{L_{1q}} - 1 - \frac{L^{"}_{aqs} X^{"}_{d} X^{"}_{aqs}}{det(A) * L_{2q}}\right) \quad (4.1.46)$$

$$\frac{\partial f_4}{\partial x_5} = A_{45} = -\frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\Bigg(- \frac{L^{"}_{aqs} R_a}{det(A)} E_{R(n+1)} \sin(\delta_{n+1}) - \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)} E_{R(n+1)} \cos(\delta_{n+1}) + \frac{L^{"}_{aqs} R_a}{det(A)} E_{I(n+1)} \cos(\delta_{n+1}) - \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)} E_{I(n+1)} \sin(\delta_{n+1})\Bigg) \quad (4.1.47)$$

Las ecuaciones de f_5 se derivan parcialmente con respecto a x_1, x_2, x_3, x_4 y corresponden a $A_{51}, A_{52}, A_{53}, A_{54}$ y tienen como valor cero. La expresión $\frac{\partial f_5}{\partial x_5} = A_{55} = 1$. Ahora se procede a expresar las ecuaciones de B_D.

$$\frac{\partial f_1}{\partial v_1} = B_{11} = -\frac{\omega_0 R_{fd}\Delta t}{2 * L_{fd}}\left(\frac{L^{"}_{ads} R_a}{det(A)} \sin(\delta_{n+1}) + \frac{L^{"}_{ads} X^{"}_{q}}{det(A)} \cos(\delta_{n+1})\right) \quad (4.1.48)$$

$$\frac{\partial f_1}{\partial v_2} = B_{12} = -\frac{\omega_0 R_{fd}\Delta t}{2 * L_{fd}}\left(-\frac{L^{"}_{ads} R_a}{det(A)} \cos(\delta_{n+1}) + \frac{L^{"}_{ads} X^{"}_{q}}{det(A)} \sin(\delta_{n+1})\right) \quad (4.1.49)$$

$$\frac{\partial f_2}{\partial v_1} = B_{21} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(\frac{L^{"}_{ads} R_a}{det(A)} \sin(\delta_{n+1}) + \frac{L^{"}_{ads} X^{"}_{q}}{det(A)} \cos(\delta_{n+1})\right) \quad (4.1.50)$$

$$\frac{\partial f_2}{\partial v_2} = B_{22} = -\frac{\omega_0 R_{1d}\Delta t}{2 * L_{1d}}\left(-\frac{L^{"}_{ads} R_a}{det(A)} \cos(\delta_{n+1}) + \frac{L^{"}_{ads} X^{"}_{q}}{det(A)} \sin(\delta_{n+1})\right) \quad (4.1.51)$$

$$\frac{\partial f_3}{\partial v_1} = B_{31} = -\frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(\frac{L^{"}_{aqs} R_a}{det(A)} \cos(\delta_{n+1}) - \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)} \sin(\delta_{n+1})\right) \quad (4.1.52)$$

$$\frac{\partial f_3}{\partial v_2} = B_{32} = -\frac{\omega_0 R_{1q}\Delta t}{2 * L_{1q}}\left(\frac{L^{"}_{aqs} R_a}{det(A)} \sin(\delta_{n+1}) + \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)} \cos(\delta_{n+1})\right) \quad (4.1.53)$$

$$\frac{\partial f_4}{\partial v_1} = B_{41} = -\frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\left(\frac{L^{"}_{aqs} R_a}{det(A)} \cos(\delta_{n+1}) - \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)} \sin(\delta_{n+1})\right) \quad (4.1.54)$$

$$\frac{\partial f_4}{\partial v_2} = B_{42} = -\frac{\omega_0 R_{2q}\Delta t}{2 * L_{2q}}\left(\frac{L^{"}_{aqs} R_a}{det(A)} \sin(\delta_{n+1}) + \frac{L^{"}_{aqs} X^{"}_{d}}{det(A)} \cos(\delta_{n+1})\right) \quad (4.1.55)$$

Las expresiones $\frac{\partial f_5}{\partial v_1} = B_{51}$ y $\frac{\partial f_5}{\partial v_2} = B_{52}$ son iguales a cero. Queda por definir las expresiones de C_D e Y_D. Las ecuaciones (4.1.17) y (4.1.18) hacen referencia a las líneas y transformadores por medio de los cuales se conectan la máquina, estas expresiones hacen parte de las ecuaciones g_1 y g_2, luego se derivan parcialmente con respecto a los términos x_1, x_2, x_3, x_4, x_5 para obtener $C_{11}, C_{12}, C_{13}, C_{14}, C_{15}$ y

$C_{21}, C_{22}, C_{23}, C_{24}, C_{25}$. Los resultados son los siguientes:

$$\frac{\partial g_1}{\partial x_1} = C_{11} = -\frac{X_{RI}X^{"}_{ads}\sin(\delta_{n+1})}{det(A)*L_{fd}} - \frac{R_aX^{"}_{ads}\cos(\delta_{n+1})}{det(A)*L_{fd}} \tag{4.1.56}$$

$$\frac{\partial g_1}{\partial x_2} = C_{12} = -\frac{X_{RI}X^{"}_{ads}\sin(\delta_{n+1})}{det(A)*L_{1d}} - \frac{R_aX^{"}_{ads}\cos(\delta_{n+1})}{det(A)*L_{1d}} \tag{4.1.57}$$

$$\frac{\partial g_1}{\partial x_3} = C_{13} = \frac{R_aX^{"}_{aqs}\sin(\delta_{n+1})}{det(A)*L_{1q}} - \frac{X_{RI}X^{"}_{aqs}\cos(\delta_{n+1})}{det(A)*L_{1q}} \tag{4.1.58}$$

$$\frac{\partial g_1}{\partial x_4} = C_{14} = \frac{R_aX^{"}_{aqs}\sin(\delta_{n+1})}{det(A)*L_{2q}} - \frac{X_{RI}X^{"}_{aqs}\cos(\delta_{n+1})}{det(A)*L_{2q}} \tag{4.1.59}$$

$$\begin{aligned}\frac{\partial g_1}{\partial x_5} = C_{15} &= -\frac{X_{RI}X^{"}_{ads}}{det(A)*L_{fd}}\psi_{fd(n+1)}\cos(\delta_{n+1}) + \frac{R_aX^{"}_{ads}}{det(A)*L_{fd}}\psi_{fd(n+1)}\sin(\delta_{n+1}) \\ &- \frac{X_{RI}X^{"}_{ads}}{det(A)*L_{1d}}\psi_{1d(n+1)}\cos(\delta_{n+1}) + \frac{R_aX^{"}_{ads}}{det(A)*L_{1d}}\psi_{1d(n+1)}\sin(\delta_{n+1}) \\ &+ \frac{R_aX^{"}_{aqs}}{det(A)*L_{1q}}\psi_{1q(n+1)}\cos(\delta_{n+1}) + \frac{X_{RI}X^{"}_{aqs}}{det(A)*L_{1q}}\psi_{1q(n+1)}\sin(\delta_{n+1}) \\ &+ \frac{R_aX^{"}_{aqs}}{det(A)*L_{2q}}\psi_{2q(n+1)}\cos(\delta_{n+1}) + \frac{X_{RI}X^{"}_{aqs}}{det(A)*L_{2q}}\psi_{2q(n+1)}\sin(\delta_{n+1}) \\ &+ \frac{X^{"}_q - X^{"}_d}{det(A)}E_{R(n+1)}\cos(2\delta(n+1)) \\ &+ \frac{(X^{"}_q\sin(2\delta_{n+1}) - X^{"}_d\sin(2\delta_{n+1}))}{det(A)}E_{I(n+1)}\end{aligned} \tag{4.1.60}$$

$$\frac{\partial g_2}{\partial x_1} = C_{21} = -\frac{R_aX^{"}_{ads}}{det(A)*L_{fd}}\sin(\delta_{n+1}) + \frac{X_{IR}X^{"}_{ads}}{det(A)*L_{fd}}\cos(\delta_{n+1}) \tag{4.1.61}$$

$$\frac{\partial g_2}{\partial x_2} = C_{22} = -\frac{R_aX^{"}_{ads}}{det(A)*L_{1d}}\sin(\delta_{n+1}) + \frac{X_{IR}X^{"}_{ads}}{det(A)*L_{1d}}\cos(\delta_{n+1}) \tag{4.1.62}$$

$$\frac{\partial g_2}{\partial x_3} = C_{23} = -\frac{X_{IR}X^{"}_{aqs}}{det(A)*L_{1q}}\sin(\delta_{n+1}) - \frac{R_aX^{"}_{aqs}}{det(A)*L_{1q}}\cos(\delta_{n+1}) \tag{4.1.63}$$

$$\frac{\partial g_2}{\partial x_4} = C_{24} = -\frac{X_{IR}X^{"}_{aqs}}{det(A)*L_{2q}}\sin(\delta_{n+1}) - \frac{R_aX^{"}_{aqs}}{det(A)*L_{2q}}\cos(\delta_{n+1}) \tag{4.1.64}$$

$$
\begin{aligned}
\frac{\partial g_2}{\partial x_5} &= C_{25} = -\frac{R_a X^{"}_{ads}}{det(A) * L_{fd}} \psi_{fd(n+1)} \cos(\delta_{n+1}) - \frac{X_{IR} X^{"}_{ads}}{det(A) * L_{fd}} \psi_{fd(n+1)} \sin(\delta_{n+1}) \\
&- \frac{R_a X^{"}_{ads}}{det(A) * L_{1d(n+1)}} \psi_{1d(n+1)} \cos(\delta_{n+1}) - \frac{X_{IR} X^{"}_{ads}}{det(A) * L_{1d}} \psi_{1d(n+1)} \sin(\delta_{n+1}) \\
&- \frac{X_{IR} X^{"}_{aqs}}{det(A) * L_{1q}} \psi_{1q(n+1)} \cos(\delta_{n+1}) + \frac{R_a X^{"}_{aqs}}{det(A) * L_{1q}} \psi_{1q(n+1)} \sin(\delta_{n+1}) \\
&- \frac{X_{IR} X^{"}_{aqs}}{det(A) * L_{2q}} \psi_{2q(n+1)} \cos(\delta_{n+1}) + \frac{R_a X^{"}_{aqs}}{det(A) * L_{2q}} \psi_{2q(n+1)} \sin(\delta_{n+1}) \\
&- \frac{X^{"}_d \sin(2\delta_{n+1}) - X^{"}_q \sin(2\delta_{n+1})}{det(A)} E_{R(n+1)} \\
&+ \frac{(X^{"}_d - X^{"}_q)}{det(A)} E_{I(n+1)} \cos(2\delta_{n+1}) \qquad (4.1.65)
\end{aligned}
$$

Las ecuaciones de Y_D, corresponden a los parámetros internos de conexionado con la red.

$$
\begin{aligned}
\frac{\partial g_1}{\partial v_1} &= Y_{11} = \frac{X^{"}_q - X^{"}_d}{2 * det(A)} \sin(2\delta_{n+1}) + \frac{R_a}{det(A)} \qquad (4.1.66) \\
\frac{\partial g_1}{\partial v_2} &= Y_{12} = Y_n + \frac{X^{"}_d \cos^2(\delta_{n+1}) + X^{"}_q \sin^2(\delta_{n+1})}{det(A)} \qquad (4.1.67) \\
\frac{\partial g_2}{\partial v_1} &= Y_{21} = -Y_n - \frac{X^{"}_d \sin^2(\delta_{n+1}) + X^{"}_q \cos^2(\delta_{n+1})}{det(A)} \qquad (4.1.68) \\
\frac{\partial g_2}{\partial v_2} &= Y_{22} = \frac{X^{"}_d - X^{"}_q}{2 * det(A)} \sin(2\delta_{n+1}) + \frac{R_a}{det(A)} \qquad (4.1.69)
\end{aligned}
$$

La matriz resultante A_D es:

$$
A_D = \begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\
A_{21} & A_{22} & A_{23} & A_{24} & A_{25} \\
A_{31} & A_{32} & A_{33} & A_{34} & A_{35} \\
A_{41} & A_{42} & A_{43} & A_{44} & A_{45} \\
A_{51} & A_{52} & A_{53} & A_{54} & A_{55}
\end{bmatrix} \qquad (4.1.70)
$$

La matriz resultante B_D es:

$$
B_D = \begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22} \\
B_{31} & B_{32} \\
B_{41} & B_{42} \\
B_{51} & B_{52}
\end{bmatrix} \qquad (4.1.71)
$$

La matriz resultante C_D es:

$$
C_D = \begin{bmatrix}
C_{11} & C_{12} & C_{13} & C_{14} & C_{15} \\
C_{21} & C_{22} & C_{23} & C_{24} & C_{25}
\end{bmatrix} \qquad (4.1.72)
$$

La matriz resultante Y_D es:

$$Y_D = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \tag{4.1.73}$$

La matriz Y_N resulta de la topología del sistema. el jacobiano del sistema resultante es:

$$J = \begin{bmatrix} A_D & B_D \\ C_D & Y_D + Y_N \end{bmatrix} \tag{4.1.74}$$

Una vez se obtienen los diferentes componentes, se resuelve el sistema de ecuaciones compuesto de la siguiente manera. Se despeja Δx y ΔV.

$$\begin{bmatrix} -F(x_{n+1}^k, V_{n+1}^k) \\ -G(x_{n+1}^k, V_{n+1}^k) \end{bmatrix} = [J] \begin{bmatrix} \Delta x_{n+1}^k \\ \Delta V_{n+1}^k \end{bmatrix} \tag{4.1.75}$$

Luego, se resuelve la siguiente expresión (4.1.76) para obtener los nuevos valores de las variables de estado y tensiones.

$$\begin{bmatrix} x_{n+1}^{k+1} \\ V_{n+1}^{k+1} \end{bmatrix} = \begin{bmatrix} x_{n+1}^k \\ V_{n+1}^k \end{bmatrix} \begin{bmatrix} \Delta x_{n+1}^k \\ \Delta V_{n+1}^k \end{bmatrix} \tag{4.1.76}$$

Capítulo 5

Solución del problema mediante procesamiento paralelo

En esta sección se presenta la metodología con la cual se elabora el software. Para empezar, se organizan las ecuaciones presentadas anteriormente, en un algoritmo con el cual se pueda entender mejor el orden de los cálculos a realizar. Posteriormente se dará explicación acerca de como se paraleliza el algoritmo relacionado con el problema. Además, una corta explicación sobre algunas funciónes del software.

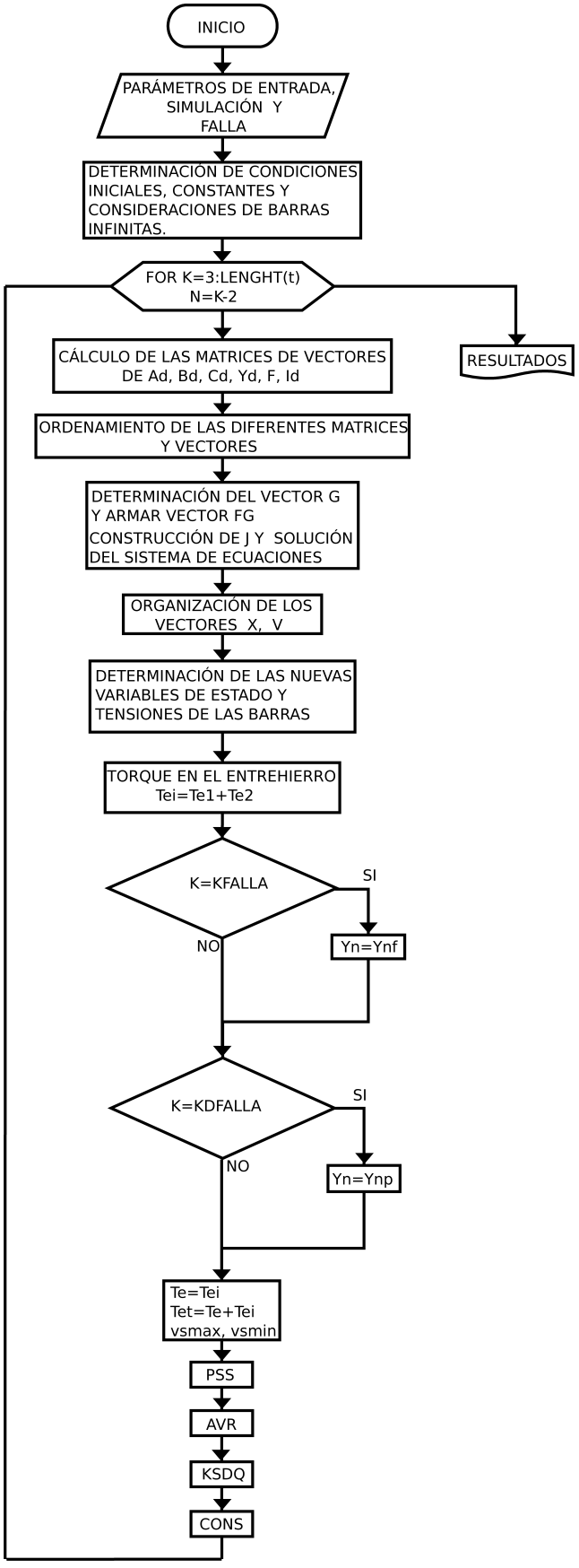


Figura 5.1: Algoritmo general del programa. Autor

5.1. Parámetros de entrada

Los datos de entrada están organizados en matrices. Cada fila representa una máquina y las columnas son los diferente parámetros de cada una.
Las entradas al sistema son:

gen: Matriz con los parámetros de los generadores.

genexct: Matriz con los parámetros de excitación de los generadores.

ngs: Vector de posiciones de los generadores de la matriz gen.

Y_{bus}*:* Es la Y_{barra} o matriz de admitancias del sistema.

V_{bus}*:* Vector con las tensiones de las diferentes barras del sistema.

N: Es el número de barras del sistema.

baseMVA: Es el valor de la potencia base del sistema.

bus: Define las potencias en cada barra.

Los parámetros de simulación son:

dt: Representa el paso de integración.

tsim: Tiempo de simulación.

tfalla: Tiempo en el cual ocurre la falla.

tdfalla: Tiempo de duración de la falla.

Parámetros de falla:

z: Impedancia de la rama afectada.

bf: Barra en la cual ocurre la falla.

bai y baf: Barras entre las cuales se origina la falla y posteriormente se despeja el circuito.

5.2. Paralelización del algoritmo

La figura 5.2, muestra el bucle que recorre las matrices de entrada para el cálculo de los parámetros de cada máquina.

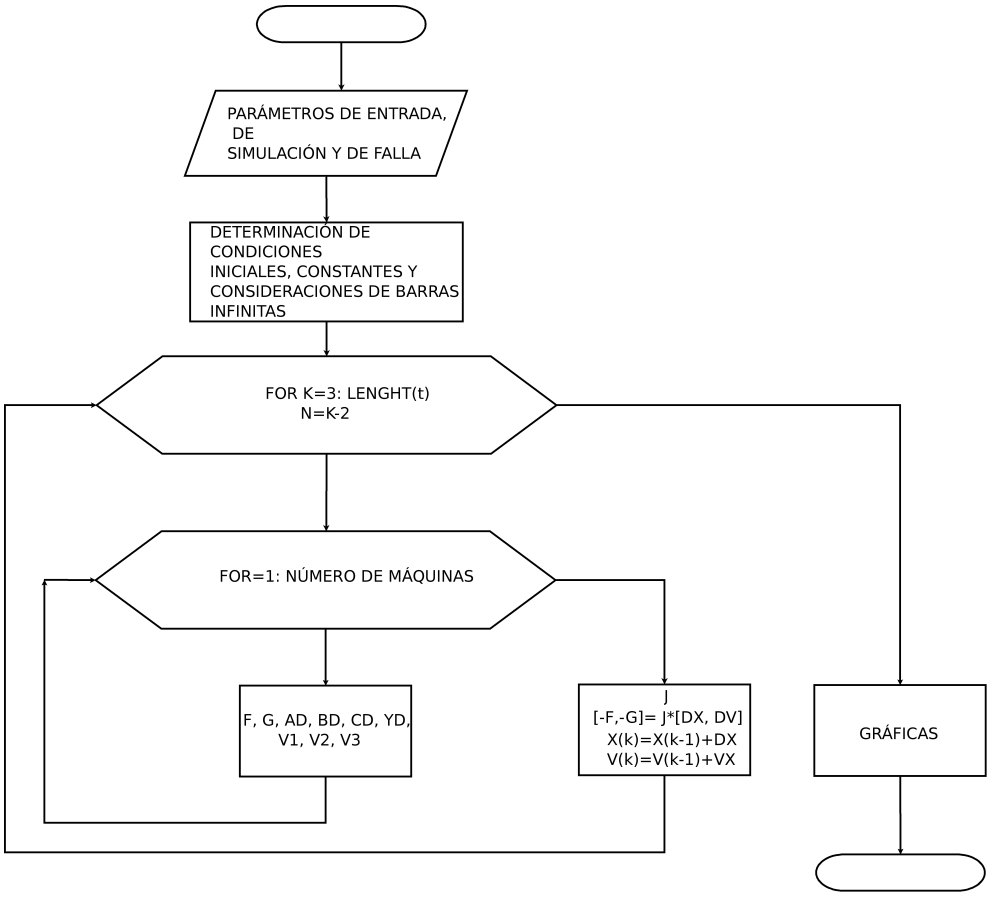


Figura 5.2: Algoritmo con bucles internos. Autor

La figura 5.3, muestra el algoritmo sin el bucle interno que recorre las diferentes matrices de entrada de cada máquina. Esta operación se puede realizar

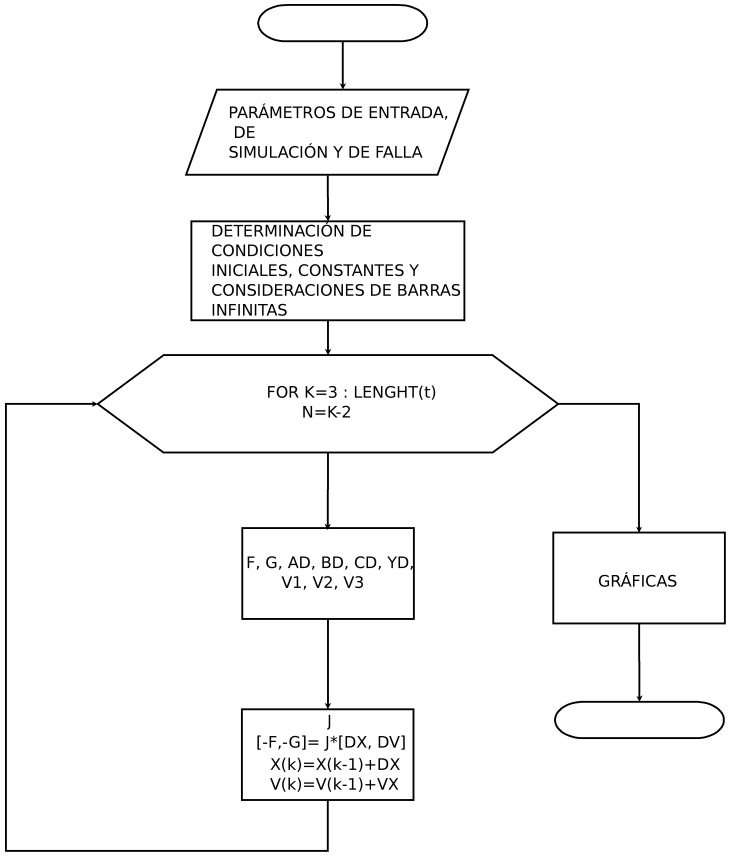


Figura 5.3: Algoritmo sin bucles internos. Autor

porque MATLAB tiene definidas instrucciones dato a dato, y permite tomar todos los datos y operarlos sin necesidad de un bucle for. Estas instrucciones, aunque son conocidas, permiten operar los datos de manera simultánea en la GPU, y se presentan a continuación.

5.3. Instrucciones utilizadas y modo de uso

Ahora se presentan instrucciones utilizadas para poder realizar operaciones dato a dato, y así eliminar el bucle for:

./ División dato a dato entre vectores o matrices.

.* Multiplicación dato a dato entre vectores o matrices.

.$\wedge$ Eleva cada dato de un vector o matriz, al exponente especificado.

find Encuentra las posiciones del vector o matriz que cumplen cierta condición y almacena las posiciones en otro vector o matriz dependiendo del caso.

any Opera junto a $find$ y se declara if *any*. Significa que para el dato existente en un vector o matriz, le aplica la operación especificada.

Los datos de entrada se toman como $gen(:, x)$ y de igual manera con otros archivos de entrada. Esta nomenclatura permite operar sobre todos los valores de la columna correspondiente. Existen mas instrucciones que pueden operar con los datos en memoria de la GPU, pero para el desarrollo del software estas fueron las más utilizadas.

5.4. Análisis del código y su implementación en la GPU

En esta sección se explican tres funciones del software, para dar a entender el uso de la GPU.

5.4.1. Función *consyci*

Los parámetros de entrada presentados en sección 5.1, ya han sido previamente enviados a la RAM de la gpu con la instrucción *gpuArray*. La función *consyci*, determina las diferentes constantes y condiciones iniciales necesarias para hallar las diferentes expresiones del sistema de ecuaciones. Como lo son: el jacobiano, los vectores F y G. Para comenzar se cálcula las constantes de saturación del eje directo y de cuadratura ksd y ksq respectivamente.

$$\begin{aligned} Et &= abs(Vbus(ngs, 1)); \\ Sg &= (complex(gen(ngs, 2), gen(ngs, 3)))/baseMVA; \\ Ig &= conj(Sg./Vbus(gen(ngs, 1))); \\ afpr &= angle(Sg); \\ P &= real(Sg); \\ Q &= imag(Sg); \\ Eat &= Vbus(gen(ngs, 1)) + (gen(ngs, 30) + i * gen(ngs, 28)). * Ig; \\ Fiat &= abs(Eat); \end{aligned}$$

Se aprecia, que la información a cálcular se encuentra organizada en vectores y al estar en el dispositivo gráfico, este la procesa de manera simultánea. E_t es un vector de tensiónes en los terminales de las máquinas que están conectadas al sistema. La potencia compleja se determina a partír de las columnas 2 y 3 de la matriz *gen*, tomando todos los valores que se encuentran en estas columnas. De igual manera, las corrientes entregadas por cada generador, se calculan a partír de 2 vectores por medio de [./]. Haciendo que la operación se realice dato a dato entre los vectores columna. De esta manera se trabaja la información a lo largo de este fragmento y también de este proyecto.

Siguiendo con las constantes de saturación, se procede a determinar los flujos del eje directo y cuadratura $FiId$ y $FiIq$. aunque se explica solo para $FiId$, ya que el otro flujo se determina exactamente igual.

$$
\begin{aligned}
FiId &= [];\\
FiId &= gpuArray(FiId);\\
xd &= find(Fiat <= gen(ngs, 31));\\
FiId(xd, 1) &= 0;\\
yd &= find((Fiat > gen(ngs, 31))\&(Fiat <= gen(ngs, 32)));\\
FiId(yd, 1) &= gen(ngs(yd), 34). * exp(gen(ngs(yd), 35). * (Fiat(yd, 1)\\
&- gen(ngs(yd), 31)));\\
zd &= find(Fiat > gen(ngs, 32));\\
FiId(zd, 1) &= gen(ngs(zd), 33) + gen(ngs(zd), 36). * (Fiat(zd, 1) - gen(ngs(zd), 32))\\
&- Fiat(zd, 1);
\end{aligned}
$$

En este fragmento del código, hay un primer uso de la instrucción *find*. Con esta instrucción se encuentran todas las posiciones de la columna 31 de la matriz gen *gen((ngs),31)*, que son menores que el flujo mutuo $Fiat$ y los guarda en un nuevo vector xd. Lo mismo sucede con yd y zd. Para poder trabajar FiId en la memoria del dispositivo, primero hay que declararlo y posteriormente con la instrucción *gpuArray*, se transfiere a la memoria de dispositivo gráfico. El cálculo de *ksd* y *ksq* se define como:

$$
\begin{aligned}
ksd(ngs, 1) &= (Fiat./(Fiat + FiId));\\
ksq(ngs, 1) &= (Fiat./(Fiat + FiIq));
\end{aligned}
$$

Donde las expresiones $Fiat$, $FiId$ y $FiIq$ están en la memoria del dispositivo gráfico, y por medio del operador [./] se cálculan las constantes de saturación de manera simultánea para las diferentes máquinas.

Cálculo de los parámetros fundamentales, saturados e insaturados:

$$
\begin{aligned}
ra &= gen(:,30);\\
ll &= gen(:,28);\\
ladu &= gen(:,13) - ll;\\
laqu &= gen(:,21) - ll;\\
lfd &= ((gen(:,14) - ll).*ladu)./((ll - gen(:,14)) + ladu);\\
l1q &= ((gen(:,22) - ll).*ladu)./((ll - gen(:,22)) + laqu);\\
l1d &= ((gen(:,15) - ll).*(ladu.*lfd))./((ll - gen(:,15)).*(ladu + lfd) + ladu.*lfd);\\
l2q &= ((gen(:,23) - ll).*laqu.*l1q)./((ll - gen(:,23)).*(laqu + l1q) + laqu.*l1q);\\
rfd &= (ladu + lfd)./(w0*gen(:,17));\\
r1q &= (ladu + l1q)./(w0*gen(:,25));\\
r1d &= (uno./(w0*gen(:,18))).*(l1d + (ladu.*lfd)./(ladu + lfd));\\
r2q &= (uno./(w0*gen(:,26))).*(l2q + (laqu.*l1q)./(laqu + l1q));\\
H &= genexct(ngs,43);\\
lads &= ksd(:,1).*ladu;\\
laqs &= ksq(:,1).*laqu;\\
ld &= lads + ll;\\
lq &= laqs + ll;\\
ladss &= (lads.*lfd.*l1d)./(lads.*lfd + lads.*l1d + lfd.*l1d);\\
laqss &= (laqs.*l1q.*l2q)./(laqs.*l1q + laqs.*l2q + l1q.*l2q);\\
lds &= ll + ladss;\\
lqs &= ll + laqss;
\end{aligned}
$$

Las constantes calculadas anteriormente, se determinan a partír matrices cuya información se encuentra en la RAM del dispositivo gráfico. Por lo tanto, los nuevos parámetros quedan automáticamente en la memoria de la GPU. Las constantes para el flujo de campo, eje directo, los 2 ejes de cuadratura y constantes que acompañan las diferentes variables de estado en el cálculo del vector F, se determinan a partír de parámetros anteriormente calculados, por ende también se guardan automáticamente en la memoria del dispositivo. Al final, se ordenan en matrices para facilitar su uso en las funciones. Las matrices de salida son:

$$
\begin{aligned}
mconsf1 &= [hwtfd\ \ kfdfd\ \ k1dfd\ \ k1qfd\ \ k2qfd\ \ ldsdra\ \ ldsdxq];\\
mconsf2 &= [hwt1d\ \ kfd1d\ \ k1d1d\ \ k1q1d\ \ k2q1d\ \ ldsdra\ \ ldsdxq];\\
mconsf3 &= [hwt1q\ \ kfd1q\ \ k1d1q\ \ k1q1q\ \ k2q1q\ \ lqsdxd\ \ lqsdra];\\
mconsf4 &= [hwt2q\ \ kfd2q\ \ k1d2q\ \ k1q2q\ \ k2q2q\ \ lqsdxd\ \ lqsdra];\\
mconsg &= [xdqslfd\ \ xdrslfd\ \ xdqsl1d\ \ xdrsl1d\ \ xqrsl1q\ \ xqdsl1q\ \ xqrsl2q\\
&\quad xqdsl2q\ \ xqmd\ \ xdmq\ \ ra\ \ xds\ \ xqs\ \ xadss\ \ lfd\ \ l1d\ \ xaqss\ \ l1q\ \ l2q];
\end{aligned}
$$

Este arreglo es una matriz de vectores, Porque cada columna representa un vector de constantes para las N máquinas.

Para las condiciones iniciales y los parámetros de compensación, la metodología es exactamente la misma y la respectiva matriz de vectores para el sistema de compensación es:

$$mconssc = [dtms2H \;\; KD \;\; mask1 \;\; Kstab \;\; kv2mas \;\; kv2men \;\; kv3mas \;\; kv3men \;\; T1sT2 \;\; k22 \;\; dtmsTr \;\; invdf9 \;\; KAtfd];$$

5.4.2. Función *mvtj*

Esta función determina la matriz de vectores *mvF, mvA, mvB, mvC, mvYd, mvId*, ecuaciones necesarias para construir el jacobiano del sistema. La primer matriz es *mvF*:

$$\begin{aligned}
f1 &= fifdn1 - fifdn - dtw0 * efd - mconsf1(:,1). * (mconsf1(:,2). * (fifdn1 + fifdn) \\
&+ mconsf1(:,3). * (fi1dn1 + fi1dn) + mconsf1(:,4). * (fi1qn1 + fi1qn) \\
&+ mconsf1(:,5). * (fi2qn1 + fi2qn) + mconsf1(:,6). * (Er(ngs,n+1). * sin(deltan1) \\
&+ Er(ngs,n). * sin(deltan)) + mconsf1(:,7). * (Er(ngs,n+1). * cos(deltan1) \\
&+ Er(ngs,n). * cos(deltan)) - mconsf1(:,6). * (Ei(ngs,n+1). * cos(deltan1) \\
&+ Ei(ngs,n). * cos(deltan)) + mconsf1(:,7). * (Ei(ngs,n+1). * sin(deltan1) \\
&+ Ei(ngs,n). * sin(deltan)));
\end{aligned}$$

$$\begin{aligned}
f2 &= fi1dn1 - fi1dn - mconsf2(:,1). * (mconsf2(:,2). * (fifdn1 + fifdn) \\
&+ mconsf2(:,3). * (fi1dn1 + fi1dn) + mconsf2(:,4). * (fi1qn1 + fi1qn) \\
&+ mconsf2(:,5). * (fi2qn1 + fi2qn) + mconsf2(:,6). * (Er(ngs,n+1). * sin(deltan1) \\
&+ Er(ngs,n). * sin(deltan)) + mconsf2(:,7). * (Er(ngs,n+1). * cos(deltan1) \\
&+ Er(ngs,n). * cos(deltan)) - mconsf2(:,6). * (Ei(ngs,n+1). * cos(deltan1) \\
&+ Ei(ngs,n). * cos(deltan)) + mconsf2(:,7). * (Ei(ngs,n+1). * sin(deltan1) \\
&+ Ei(ngs,n). * sin(deltan)));
\end{aligned}$$

El anterior código pertenece a las expresiones de la sección (4.1.3), para determinar las funciones F. Las ecuaciones f_3, f_4, f_5 son determinadas de manera semejante.

Las contantes y condiciones iniciales calculadas previamente en (5.4.1), se utilizan en las expresiones de la sección (4.1.3) para determinar diferentes funciones de las variables de estado. Estas funciones, al calcularse a partír de parámetros que están en la memoria del dispositivo, quedan guardadas automáticamente en la memoria de la GPU. Los vectores columna operan entre si dato a dato, por medio de la instrucción [.∗]. Por lo tanto, no es necesario crear un bucle para recorrer los vectores. Luego se ordenan las diferentes funciones en una matriz, con el fin de facilitar la manipulación de esta información. El arreglo queda de la siguiente manera:

$mvF = [f1 \;\; f2 \;\; f3 \;\; f4 \;\; f5]$;

Los siguientes cálculos son las diferentes componentes de A_D, B_D, C_D, Y_D que conforman el jacobiano. El primer parámetro es A_D y el fragmento del código se

muestra a continuación.

$$\begin{aligned}
A11 &= unos - mconsf1(:,1).*mconsf1(:,2);\\
A12 &= -mconsf1(:,1).*mconsf1(:,3);\\
A13 &= -mconsf1(:,1).*mconsf1(:,4);\\
A14 &= -mconsf1(:,1).*mconsf1(:,5);\\
A15 &= -mconsf1(:,1).*(mconsf1(:,6).*Er(ngs,n+1).*cos(deltan1)\\
&- mconsf1(:,7).*sin(deltan1).*Er(ngs,n+1)\\
&+ mconsf1(:,6).*Ei(ngs,n+1).*sin(deltan1)\\
&+ mconsf1(:,7).*Ei(ngs,n+1).*cos(deltan1));
\end{aligned}$$

Los parámetros A_{11}, A_{12}, A_{13} A_{14}, A_{15}, son los correspondientes a todas las máquinas conectadas al sistema, calculados al mismo tiempo. Debido a que determinan a partír de la columna 1 matriz $mconsf1$ y la columna 2 de la misma matriz, cuyos valores se encuentra en la memoria del dispositivo y por medio del operador $.*$ se realiza el cálculo, dato a dato de las respectivas columnas de la matriz $mconsf1$.

De igual, manera se procede con los parámetros de los componentes A_D, B_D, C_D, Y_D con sus correspondientes constantes y valores iniciales. Una vez hallados los diferentes componentes del jacobiano, se procede a organizarlos de la siguiente manera:

$$\begin{aligned}
mvA &= [A11\ \ A12\ \ A13\ \ A14\ \ A15\ \ A21\ \ A22\ \ A23\ \ A24\ \ A25\\
&\quad\ A31\ \ A32\ \ A33\ \ A34\ \ A35\ \ A41\ \ A42\ \ A43\ \ A44\ \ A45\\
&\quad\ A51\ \ A52\ \ A53\ \ A54\ \ A55];\\
mvB &= [B11\ \ B12\ \ B21\ \ B22\ \ B31\ \ B32\ \ B41\ \ B42\ \ B51\ \ B52];\\
mvC &= [C11\ \ C12\ \ C13\ \ C14\ \ C15\ \ C21\ \ C22\ \ C23\ \ C24\ \ C25];
\end{aligned}$$

Hay que resaltar que cada columna, son los diferentes parámetros de todos los generadores.

Por último, se define el parámetro Y_D. Para empezar hay que declarar los vectores en el dispositivo gráfico. Esta tarea se puede realizar directamente, ya que MATLAB dispone de funciones que cumplen esta labor.

$$\begin{aligned}
Yd11 &= gpuArray.zeros(2*N,1);\\
Yd12 &= gpuArray.zeros(2*N,1);\\
Yd21 &= gpuArray.zeros(2*N,1);\\
Yd22 &= gpuArray.zeros(2*N,1);
\end{aligned}$$

Otras instrucciones definidas por MATLAB son *gpuArray.ones, gpuArray.rand*, que permiten crear directamente la información en la memoria del dispositivo.

El cálculo de los parámetros

$$\begin{aligned}
Yd11(ngs,1) &= gen(:,30)./det - mconsg(:,10)./det. * sin(2 * deltan1)/2; \\
Yd12(ngs,1) &= (mconsg(:,12). * cos(deltan1). \wedge 2 \\
&+ mconsg(:,13). * sin(deltan1). \wedge 2)./det; \\
Yd21(ngs,1) &= -(mconsg(:,12). * sin(deltan1). \wedge 2 \\
&+ mconsg(:,13). * cos(deltan1). \wedge 2)./det; \\
Yd22(ngs,1) &= (gen(:,30) + mconsg(:,10). * sin(2 * deltan1)./2)./det;
\end{aligned}$$

Estos parámetros representan la conexión de la máquina con el sistema, y cada uno son un vector columna. En el cuál cada fila de la columna representa el parámetro de la máquina. Posteriormente, se encuentran las modificaciones hechas considerando las barras infinita y las carga estáticas. El ordenamiento de las Y_D es:

$$mvYd = [Yd11 \;\; Yd12 \;\; Yd21 \;\; Yd22];$$

5.4.3. Solución del sistema de ecuaciones

Ahora se procede a armar el sistema de ecuaciones.

$$\begin{aligned}
G &= Yn * E - Id; \\
FG &= [-F; -G]; \\
J &= [Ad \;\; Bd; Cd \;\; Yd + Yn]; \\
deltaXV &= J \setminus FG; \\
[deltX, deltV] &= deltaxv(deltaXV, ngs, N);
\end{aligned}$$

El vector de corrientes G se construye a partír de las tensiones en las diferentes barras, y está organizado en componentes real e imaginario. Adicionalmente, se tiene la matriz Y_N del sistema y el vector I_D de corrientes inyectadas por los generadores, cargas estáticas y barras infinitas. El vector FG se forma como se muestra en el código. El sistema de ecuaciones se resuelve por medio de la instrucción [\].

Capítulo 6

Resultados de la simulación

La herramienta software fue empleada con dos sistemas, el primero es el ejemplo 13.2 del libro [1], en el que se muestran los conceptos básicos de estabilidad. Este sistema representa a la máquina síncrona puesta a una barra infinita. El segundo sistema también es un ejemplo del mismo libro, el ejemplo 12.6.

Las pruebas consisten en ejecutar el programa, y comparar los tiempos de simulación, entre el código ejecutado en la CPU y el código procesado sobre la GPU. Los códigos se ejecutan cada uno 10 veces y los tiempos se promedian para establecer un tiempo aproximado de simulación y poder comparar las respectivas respuestas. El código empleado para procesar la información en la CPU es el mismo que se utiliza para procesar la información en la GPU con la salvedad, de que este no transfiere la información a la memoria del dispositivo gráfico. El sistema a ejecutar es el del generador puesto a la barra infinita de la figura 6.1a y el sistema de 10 barras con 4 generadores figura 6.2b.

Los sistemas se presentan en la figura 6.1. Los tiempos de simulación de los sistemas se muestran a la tabla 6.1. Los resultados de promediar los respectivos

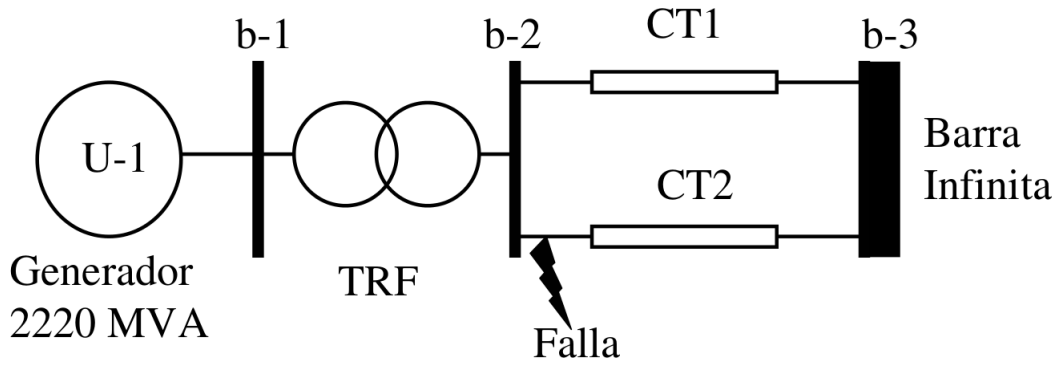
Tabla 6.1: Tiempos de simulación

(a) Barra infinita

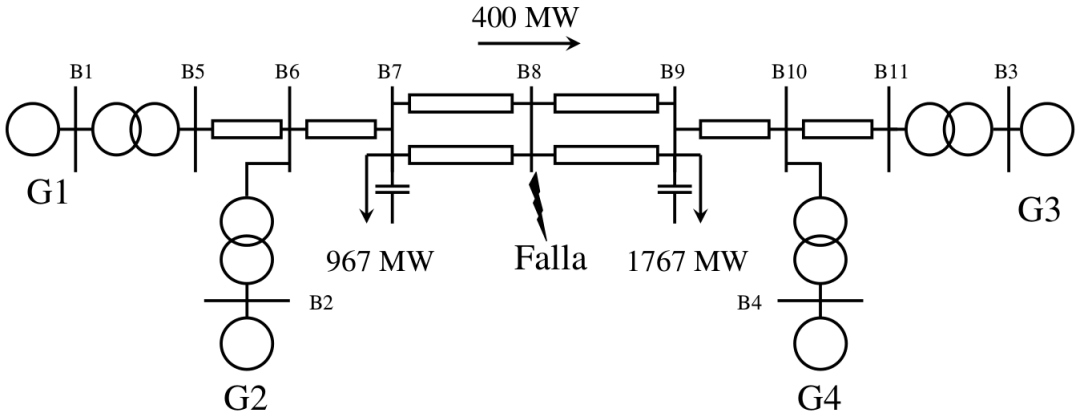
TIEMPOS		
CPU(s)	GPU(s)	GPU(min)
0,99	180	3,02
0,96	179,4	2,99
0,98	178,8	2,98
0,97	180	3,02
0,97	180,6	3,01
0,97	180,6	3,01
0,96	182,4	3,04
0,98	183	3,05
0,99	184,2	3,07
0,97	180	3,02

(b) Sistemas de 4 generadores

TIEMPOS		
CPU(s)	GPU(s)	GPU(min)
1,64	157,8	2,63
1,65	159	2,65
1,65	160,8	2,68
1,68	159	2,65
1,66	160,2	2,67
1,66	158,4	2,64
1,64	160,8	2,68
1,68	161,4	2,69
1,64	157,8	2,63
1,66	160,2	2,67



(a) Sistema de barra infinita [2].



(b) Sistema de 4 generadores [2].

Figura 6.1: Sistemas simulados

tiempos se presentan en la tabla 6.2.

Tabla 6.2: Promedio de los tiempos de simulación

(a) Barra infinita

CPU(s)	GPU(s)	GPU(MIN)
0,974	181,26	3,021

(b) Sistema de 10 barras 4 generadores

CPU(s)	GPU(s)	GPU(MIN)
1,656	159,7	2,659

6.1. Análisis de la propagación del error

El error en los datos se origina por la falta de sincronismo entre los núcleos que procesan la información, provocando que a través de las iteraciones el error aumente.

Tabla 6.3: Propagación del error

Tamaño del vector	% error x_1	% error x_2
1	0	0
10	0	0
100	$4{,}4408921*10^{-14}$	$4{,}4408921*10^{-14}$
1000	$8{,}4376949*10^{-13}$	$7{,}1054275*10^{-13}$
10000	$8{,}6597396*10^{-12}$	$5{,}6843419*10^{-12}$
100000	$4{,}1089354*10^{-11}$	$4{,}5474735*10^{-11}$
1000000	$8{,}0091489*10^{-10}$	$3{,}6379788*10^{-10}$
10000000	$1{,}4795121*10^{-8}$	$4{,}6566129*10^{-8}$
20000000	$2{,}5810765*10^{-7}$	$3{,}7252903*10^{-7}$

6.2. Estimación de los tiempos de retardo debido a la transferencia de información

Debido a que la información se encuentra almacenada inicialmente en la memoria de la CPU, transferir la información a la memoria de la GPU representa un tiempo de retardo para la simulación. Adicionalmente, si la transferencia de información se realiza al interior de la iteración, el tiempo de transferencia aumenta de acuerdo al número de iteraciones. Por tal motivo, la cantidad de veces que se requiera para transferir la información, debe ser el menor posible. En el presente trabajo, el número de veces que se transfiere la información es constante y no depende de la cantidad de máquinas del sistema.

El tiempo de retardo estimado es de 1.194806 s

Capítulo 7

Conclusiones

La finalidad de este proyecto era la de diseñar un software mediante el cuál, se modelara y simulara ante una gran perturbación, el comportamiento de una máquina síncrona conectada a una barra infinita, y que permitiera incluir otros generadores, líneas y transformadores. Adicionalmente, los cálculos fueran realizados empleando programación paralela utilizando la GPU. Por lo tanto, de acuerdo con las simulaciones anteriormente expuestas, se puede decir que el objetivo de este proyecto se han cumplido de manera satisfactoria. Las conclusiones acerca de este proyecto se presentan a continuación:

- De acuerdo con el análisis acerca del fenónemo de estabilidad, los parámetro para cada generador se cálculan de manera independiente a los demás generadores y se utiliza un mismo modelo matemático para su solución. Se puede decir que, el fenómeno de estabilidad es altamente paralelizable y que se puede solucionar bajo diferentes métodos numéricos empleando procesamiento paralelo.

- Se aprecia que los tiempos de simulación para el sistema de barra infinita, son menores en la unidad de procesamiento central. Esto se debe a que la velocidad de procesamiento de la CPU es mayor al de la GPU, y se esta procesando una pequeña cantidad de información. Por lo tanto, para sistemas pequeños no es eficiente utilizar la GPU para procesar información.

- De acuerdo con el segundo sistema, aunque aún existe una diferencia ventajosa de la CPU con respecto a la GPU. Se aprecia que el programa ejecutado sobre la CPU, aumento ligeramente su tiempo de simulación debido a que la cantidad de información procesada es mayor. En cambio, el mismo problema ejecutado sobre la GPU disminuyó su tiempo de simulación, ya que la GPU esta diseñada para manejar grandes volúmenes de información, haciéndola más eficiente trabajando con vectores. Ademas, para una mayor cantidad de datos, los tiempos de simulación tienden a aumentar mas lentamente.

- Ya que el procesamiento en paralelo permite trabajar con gran volumen de datos y se que se pueden aplicar diferentes métodos numéricos al fenómeno de la estabilidad en sistemas de potencia. Se puede deducir que, se pueden incorporar modelos matemáticos que consideren muchas mas variables de la

máquina síncrona, para tener una mayor aproximación del fenómeno de estudio.

- Paralelizar un algoritmo, consiste en eliminar todos los bucles que realizan tareas sobre vectores o matrices y que pueden ser procesadas de manera simultánea en la GPU.

- Realizar la misma operación sobre los diferentes núcleos de procesamiento, el tiempo de cómputo de cada núcleo no va a ser el mismo. Por lo tanto, se originan pequeños margenes de error que se propagan a través de las iteraciones. Por tal motivo es necesario realizar pruebas de confiabilidad de resultados. De acuerdo con la tabla 6.3, se observa que el error para la simulación es cero, en parte porque la pequeña cantidad de máquinas en el sistema. Para una gran cantidad de datos el error empieza a ser considerable.

- Para ejecutar el código sobre la GPU, debe tenerse en cuenta que el tiempo de transferencia de información entre dispositivos puede ser considerable, por lo tanto, el número de veces que se transfiera la información debe ser la menor posible. Para el proyecto, el número de veces que se transfiere la información es constante y no depende de la cantidad de máquinas del sistema. Por lo tanto, se puede decir que para un sistema de una dimensión determinada, el tiempo de computo en ambos dispositivos será el mismo y, a partir de este punto será mas eficiente el trabajo en la GPU.

Aportes

Resolver el problema de estabilidad por medio de procesamiento paralelo. Además de presentar las bases para un posterior desarrollo sobre el fenómeno de estabilidad, en el que se estudien modelos mas completos de la máquina síncrona, la representación de elementos dinámicos, otros sistemas de compesación, además de que se puedan estudiar sistemas de mayor tamaño y teniendo en cuenta otros métodos numéricos. Todo esto, empleando procesamiento paralelo.

Sugerencias

Sugerencias para posteriores desarrollos se tienen:

- Implementación de procesamiento en paralelo en un lenguaje de bajo nivel para poder tener mayor control sobre la GPU, y disminuir aún mas los tiempos de cómputo en problemas con gran volumen de información.

- Análisis de sistemas de potencia de mayor tamaño para tener una mejor aproximación de la unidad gráfica en el fenómeno de estabilidad.

- Desarrollo de una interfaz gráfica.

- Implementación de otros métodos numéricos que permitan reducir la cantidad de transferencias de información entre dispositivos.

Bibliografía

[1] KUNDUR, Prabha. Power System Stability and Control. McGraw-Hill, Inc. 1 st. edition 1994.

[2] RODRÍGUEZ SIERRA,Carlos. Herramienta software para el modelado y simulación de sistemas elétricos de potencia ante grandes perturbaciones. Bucaramanga, 2007, 171p. Trabajo de investigación (magíster en en Ingeniería eléctrica).

[3] BARRERA CÁRDENAS, René Alexander. Modelado de la máquina síncrona mediante redes neuronales. Bucaramanga, 2010, 88p. Trabajo de investigación (magíster en en Ingeniería eléctrica).

[4] NVIDIA, CUDA PROGRAMACIÓN PARALELA FACILITADA. wed-site: `http://la.nvidia.com/object/cuda_home_new_la.html`

[5] CUDA TOOLKIT DOCUMENTATION. wed-site: `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`

[6] R. Zimmerman and D. Gan. *MATPOWER version 2.0, User's manual.* PSERC, School of Electrical Engineening, Cornell University, Ithaca, NY, 1997.

[7] mathworks (2012, mayo) gpucomputing toolbox (online) http://www.mathworks.com/help/distcomp/graphics-processing-unit-gpu-computing.html

Anexo A

Manual de usuario, maqgpu.

Introducción

Maqgpu, es un software en el cuál se modela y simula ante una gran perturbación, las condiciones de estabilidad de una máquina síncrona puesta a una barra infinita. Adicionalmente el software permite incluir otras máquinas, líneas y generadores. El aporte del maqgpu, es el de procesar la información de los diferentes parámetros, de manera simultánea en la unidad gráfica del equipo de cómputo[2].

El software es un prototipo para desarrollo posteriores, en el que se modelan máquinas de orden superior, sistemas de mayor dimensión y diferentes sistemas de compesación, entre otros. Maqgpu, muestra ciertas pautas en la metodología a seguir, en el diseño se software de procesamiento paralelo.

Adicionalmente, la herramienta aprovecha el software MATPOWER [6], para determinar el flujo de cargas y la condiciones iniciales en las que opera el sistema de potencia.

Requisitos e instalación

Para la ejecución de la herramienta software, se requiere de versiones de MATLAB 8.0 en adelante, con la toolbox GPU computing. Es recomendable instalar el software en equipos con especificaciones técnicas superiores a 1024 MB de memoria RAM, 1.4 GHz de procesamiento central y 512 MB de RAM en la unidad de procesamiento gráfica. Además el dispositivo gráfico debe tener soporte CUDA.

pasos para la instalación:

1. Descomprimir la carpeta maqgpu.
2. Agregar la carpeta maqgpu al *path* de MATLAB.
3. La carpeta con los casos de estudio se deben agregar de igual manera al *path* de MATLAB.

Ejecución

Para la ejecución del software, se debe escribir en la consola de MATLAB la instrucción *maqgpu(` barinf `)* o *maqgpu(` st1a `)*, name es el nombre del archivo a simular. En el se encuentran los parámetros del sistema de potencia, estabilidad y las condiciones de simulación. Otra forma, es ejecutando directamente el archivo *maqgpu* y simulará por defecto, el ejemplo 13.2 del libro Kundur [1]. Una vez finalizada la simulación, se muestra el tiempo de procesamiento, y un menú de resultados. Con este menú se puede visualizar:

0. *Salir del programa.* Permite iniciar una nueva simulación.

1. *Ángulos de los rotores.* Permite visualizar el ángulo δ de las diferentes máquinas y su evolución en el tiempo.

2. *Desviación de las velocidades rotóricas.* Permite visualizar la variación de la velocidad de los rotores de las diferentes máquinas y su evolución en el tiempo.

3. *Tensiones de todas las barras.* Permite visualizar las tensiones en todas las barras y su evolución en el tiempo.

4. *Tensiones de las barras con generadores.* Permite visualizar las tensiones en la barras con generación y su evolución en el tiempo.

5. *Torque en el entrehierro.* Permite visualizar el torque en el entrehierro y dado que esta en *pu*, resulta ser igual a la potencia de la máquina, y su variación en el tiempo.

6. *Jacobiano del sistema.* Permite visualizar la estructura del jacobiano del sistema.

Parámetros de entrada

Los datos de entrada están en formato MATLAB (.m). Contienen los parámetros de los generadores, topología del sistema, sistema de excitación, parámetros de simulación y de falla. Los parámetros de simulación y falla son:

tsim: Tiempo de simulación, iniciando desde 0.

dt: Paso de integración.

bf: Barra que entra en estado de falla.

bai y baf: Barras entre las cuáles ocurre la falla.

z: Impedancia de la rama afectada.

tfalla: Tiempo en el cuál se origina la falla, después de iniciada la simulación.

tdfalla: Tiempo después de originada la falla, que se produce el despeje.

Tabla A.1: Información de las barras

Columna	Parámetro	Descripción	Unidades
1	nb	Número de barra	-
2	tipo	Tipo de barra	1-PQ 2-PV 3-Referencia 4-Aislada 5-Infinita
3	P_d	Demanda potencia activa	MW
4	Q_d	Demanda potencia activa	MWAr
5	P_{d0}	Demanda potencia activa sec. cero	MW
6	Q_{d0}	Demanda potencia reactiva sec. cero	MWAr
7	G_s	Conductancia shunt	MW a V_{nb}=1 pu
8	B_s	Susceptancia shunt	MVAr a V_{nb}=1 pu
9	G_{s0}	Conductancia shunt	MW a V_{nb}=1 pu
10	B_{s0}	Susceptancia shunt	MWAr a V_{nb}=1 pu
11	área	Número de área	-
12	V_m	Magnitud inicial de tensión	pu
13	V_a	Ángulo inicial de tensión	grados
14	basekV	Tensión	kV
15	zona	Número de zona	-
16	maxV	Magnitud máxima de tensión	pu
17	minV	Magnitud mínima de tensión	pu

baseMVA: Valor base del sistema.

Datos acerca de las barras.
Notas:

- En el flujo de cargas, las barras infinitas se modelan como tipo PV.
- Barras tipo PV sin generadores, se modelan como barra PQ.
- Las tensiones V_m y V_a se utilizan para dar inicio a la solución al flujo de cargas.

Parámetros de la máquina síncrona

El modelo tomado es el del cuarto orden. El modelo puede representar los efectos de saturación y se le pueden asociar sistemas de control, como AVR y PSS. El modelo representa el comportamiento de tres circuitos rotóricos, dos para el eje directo y uno para el eje de cuadratura, también se tiene en cuenta la inercia mecánica del rotor.

Tabla A.2: Parámetros modelo clásico de la máquina síncrona

Columna	Parámetro	Descripción	Unidades
1	ngs	Número de generadores del sistema	-
2	ng	Número del generador	-
3	P_g	Potencia activa	MW
4	Q_g	Potencia reactiva	MVAr
5	V_{sp}	Tesión objetivo	MWAr
6	reg	Regulación	%
7	S_n	Potencia nominal	MVA
8	st	Estado	1- En servicio 0- Fuera de servicio
9	R_a	Resistencia de estator	pu
10	X_l	Reactancia de dispersión estator	pu
11	X_{pl}	Reactancia de dispersión estator	pu
12	X_d	Reactancia síncrona	pu
13	$X_d^{'}$	Reactancia transitoria	pu
14	$X_d^{''}$	Reactancia subtransitoria	pu
15	$T_{d0}^{'}$	Consrante de tiempo transitoria	s
16	$T_{d0}^{''}$	Constante de tiempo subtransitoria	s
17	X_q	Reactencia síncrona	pu
18	$X_q^{''}$	Reactancia subtransitoria	pu
19	$T_{q0}^{''}$	Constante de tiempo subtransitoria	s
20	ψ_{fd}	Constante de saturación eje directo	pu
21	ψ_{1d}	Constante de saturación eje directo	pu
22	ψ_{G1d}	Constante de saturación eje directo	pu
23	A_{satd}	Constante de saturación eje directo	pu
24	B_{satd}	Constante de saturación eje directo	pu
25	L_{ratiod}	Constante de saturación eje directo	pu
26	ψ_{1q}	Constante de saturación eje cuadratura	pu
27	ψ_{2q}	Constante de saturación eje cuadratura	pu
28	ψ_{G2q}	Constante de saturación eje cuadratura	pu
29	A_{satq}	Constante de saturación eje cuadratura	pu
30	B_{satq}	Constante de saturación eje cuadratura	pu
31	L_{ratioq}	Constante de saturación eje cuadratura	pu
32	H	Constante de inercia	s
33	K_d	Coeficiente de torque amortiguante	pu
34	R_2	Resistencia de secuencia negativa	pu
35	X_2	Reactancia de secuencia negativa	pu
36	R_0	Resistencia de secuencia cero	pu
37	X_0	Reactancia de secuencia cero	pu
38	R_e	Resistencia de puesta a tierra	pu
39	X_e	Reactancia de puesta a tierra	pu

La máquina síncrona y su modelo de saturación

Las saturación se representa variando las inductancias mutuas de los diferentes circuitos, por medio de los factores de saturación. Un análisis mas detallado se presenta en la sección (B).

$$L_{ad} = K_{sd} L_{adu} \tag{A.0.1}$$
$$L_{aq} = K_{sq} L_{aqu} \tag{A.0.2}$$

se tiene que:

K_{sd} = factor de saturación del eje directo.

K_{sq} = factor de saturación del eje en cuadratura.

L_{adu} = valor insaturado de L_{ad}.

L_{aqu} = valor insaturado de L_{aq}.

A partir de las curva de saturación en circuito abierto, se determinan los factores de saturación. La relación obtenida a partir de la figura (A.1) es la siguiente:

$$K_{sd} = \frac{\psi_{at}}{\psi_{at} + \psi_I} \tag{A.0.3}$$

con:
$\psi_I = \ si\ \psi_{at} \leqslant \psi_{T1}$ (región I).
$\psi_I = A_{sat} e^{B_{sat}(\psi_{at} - \psi_{T1})} \ \ si \ \ \psi_{T1} < \psi_{at} \leqslant \psi_{T2}$ (región II).
$\psi_I = \psi_{G2} + L_{ratio}(\psi_{at} - \psi_{T2}) - \psi_{at} \ \ si \ \ \psi_{at} > \psi_{T2}$ (región III).
$L_{ratio} = \frac{L_{adu}}{L_{incr}}$

Los anteriores términos representan lo siguiente:

ψ_{at} Flujo Resultante en el entrehierro.

i_{fd} Corriente de campo.

K_{sd} Factor de saturación eje directo.

K_{sq} Factor de saturación eje de cuadratura. Se determina de manera similar a K_{sd}.

Sistemas de excitación - *AVR*

Maqgpu tienes como sistemas de compensación, un regulador de tensión y un estabilizador de sistema de potencia. Las variables empleadas es el sistema de compensación son las siguientes:

- V_{ref} = Ajuste referencia.

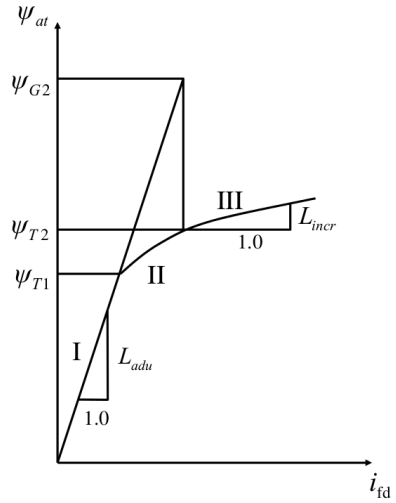


Figura A.1: Saturación de la máquina síncrona [1].

- V_1 = Compensador de carga y señal del transductor de tensión.
- V_{ss} = Señal de estabilidad.
- v_{oxl} = Señal limitadora de la corriente de campo.
- E_{fd} = Tensión de campo del flujo directo.
- I_{fd} = Corriente de campo de flujo directo.
- E_t = Tensión en terminales.
- I_t = Corriente en terminales.
- $\Delta\omega$ = Variación de la velocidad del rotor.

Sistema de regulación estático

La figura A.2, representa al diagrama de bloques del sistema de excitación. Sus parámetros se especifican en la tabla.

Tabla de parámetros del sistema de excitación estático.

Estabilizador del sistema de potencia -*PSS*

El software maqgpu incluye un modelo que compensa la variación de la velocidad de síncronismo $\Delta\omega$. El diagrama de bloques correspondiente se representa en la figura A.3.

Tabla de parámetros del *PSS*. *Nota:*
A los estabilizadores se les debe asociar un regulador de tensión. Las tablas y figuras presentadas en este apéndice fueron tomadas de la tesis [2].

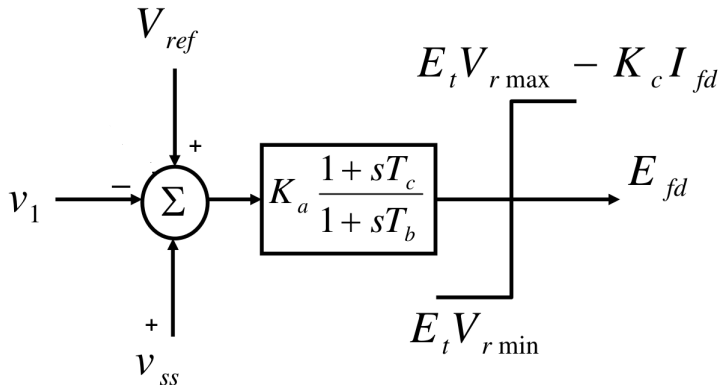


Figura A.2: Regulador estático [1].

Tabla A.3: Parámetros sistema de excitación estático

Columna	Parámetro	Descripción	Unidades
1	ng	Número de máquina síncrona	-
2	st	estado	1- activo 0-inactivo
3	K_a	Ganancia regulador	pu
4	T_b	Constante de tiempo compensador	s
5	T_c	Constante de tiempo compensador	s
6	K_c	Constante rectificador	pu
7	V_{rmax}	Límite superior del regulador	pu
8	V_{rmin}	Límite inferior del regulador	pu
9	T_r	Constante de tiempo del transductor	s
10	R_c	Parte real de la impedancia de compensación	pu
11	X_c	Parte imaginaria de la impedancia de compensación	pu

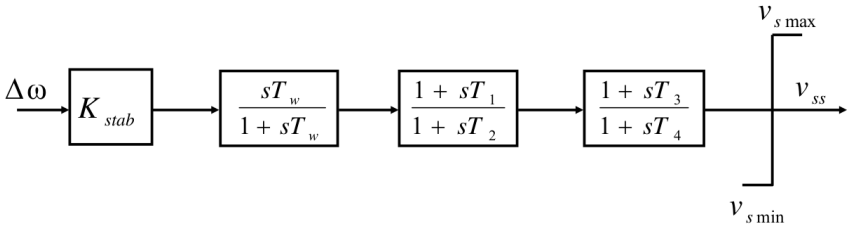


Figura A.3: Estabilizador del sistema de potencia [1].

Tabla A.4: Estabilizador del sistema de potencia

Columna	Paámetro	Descripción	Unidades
1	ng	Número de máquina síncrona	-
2	st	Estado	1-En servicio 0-fuera de servicio
3	K_{stab}	Ganancia del estabilizador	pu
4	T_W	Constante de tiempo filtro washout	s
5	T_1	Constante de tiempo compensador	s
6	T_2	Constante de tiempo compensador	s
7	T_3	Constante de tiempo compensador	s
8	T_4	Constante de tiempo compensador	s
9	v_{smax}	Límite superior estabilizador	pu
10	v_{smin}	Límite inferior estabilizador	pu

Anexo B

Saturación en el análisis de la estabilidad transitoria

Características de corto circuito y circuito abierto

Para el tratamiento de la saturación es necesario analizar el circuito magnético en la condiciones de circuito abierto (OCC). Por lo tanto, para condiciones de vacío, a velocidad nominal se tiene que:

$$i_d = i_q = \psi_q = e_d = 0 \tag{B.0.1}$$

$$E_T = e_q = \psi_d = L_{ad} = i_{fd} \tag{B.0.2}$$

En las pruebas de circuito abierto (OCC), se relacionan E_t e i_{fd} dando como resultado la curva características de el eje d. Las imágenes mostradas en esta sección son tomadas del capítulo 3 de [1].

Como se muestra en la figura B.1, la línea tangente a la parte baja de la curva se denomina línea del entrehierro, e indica la corriente de campo requerida para superar la reluctancia del entrehierro. La salida de la característica de circuito abierto indica el grado de saturación de los hierros del rotor y del estator.

La característica de corto circuito (SCC), también se muestra en la figura B.1. Este es un tramo de la corriente de armadura *vs* corriente de campo, con el generador operando en estado estable a velocidad nominal, con un corto circuito trifásico ubicado en terminales de la armadura. La característica de corto circuito (SCC) es lineal y crece más allá de la corriente de armadura nominal. Desde este punto, es pequeña o no hay ningún grado de saturación en el hierro, debido al efecto desmagnetizante de la reacción de la armadura.

En la figura B.2 se aprecia, que la tensión interna del generador es igual al producto de la corriente de corto circuito y la reactancia de sincronismo. Si se tiene la corriente en *pu*, la tensión es proporcional a I_{fsc}. De acuerdo a la figura B.1 se obtiene:

$$KI_{FSC} = 1{,}0X_{s(unsat)}$$

Corresponde la tensión en *pu* en la línea del entrehierro.

$$KI_{FNL} = 1{,}0$$

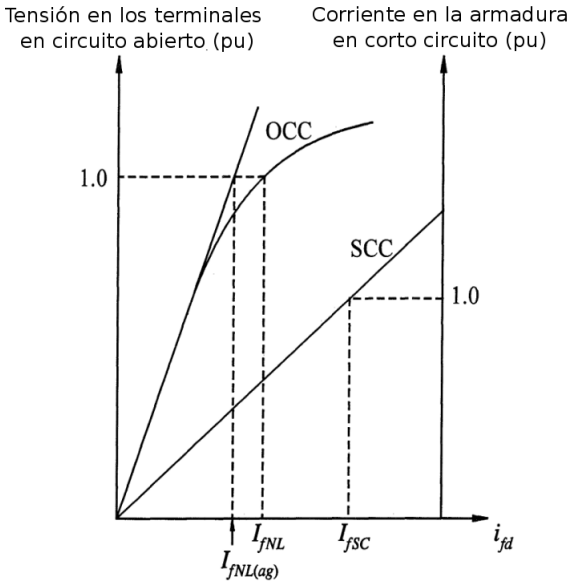


Figura B.1: Características de corto circuito y circuito abierto. [1].

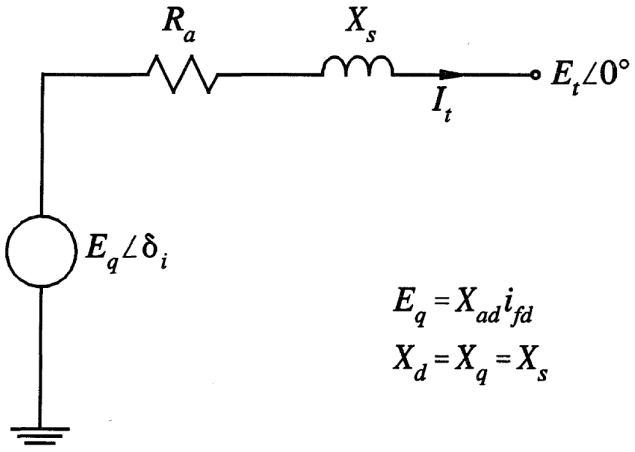


Figura B.2: Circuito equivalente en estado estable [1].

El valor no saturado de X_s es:

$$X_{s(unsat)} = \frac{I_{FSC}}{I_{FNL(ag)}} \tag{B.0.3}$$

El valor saturado de X_s que corresponde a la tensión nominal esta dado por:

$$X_{s(sat)} = \frac{I_{FSC}}{I_{FNL}} \tag{B.0.4}$$

La relación de corto circuito está definido como la razón de la corriente de campo requerida para producir una tensión a velocidad nominal y sin carga entre la corriente de campo requerida para producir una corriente de campo nominal bajo condiciones de corto circuito trifásico en estado estable.

$$SCR = \frac{I_{NFL}}{I_{FSC}} = \frac{1}{X_{s(sat)}} \tag{B.0.5}$$

El factor SCR refleja el grado de saturación y tiene relevancia con respecto al rendimiento y costo. Un bajo SCR indica que se requiere un gran cambio en la corriente de campo para mantener constante el tensión en los terminales de la máquina. Por lo tanto se requiere de una gran sistema de compensación para que la máquina se mantenga estable. Por otra parte, una máquina con un (SCR) bajo implica un menor peso, tamaño y costo. Por lo tanto la tendencia son máquina con un sistema de control y excitación asociados, con bajos (SCR).

Análisis de la saturación en el estudio de estabilidad

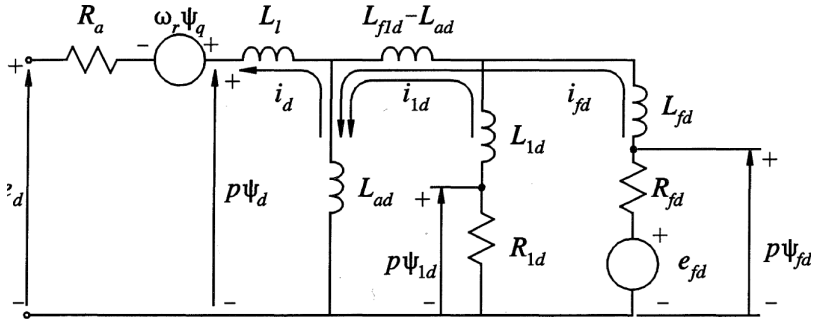
Consideraciones a tener en cuenta para la representación de la saturación magnética, en el estudio de estabilidad.

- Las inductancias de dispersión y saturación son independientes, Los flujos de dispersión se van por el aire y no se afectan de manera considerable por la saturación del hierro. Por lo tanto, solo las inductancias mutuas L_{ad} y L_{aq} del circuito de la figura B.3a y B.3b respectivamente, son las que se saturan.

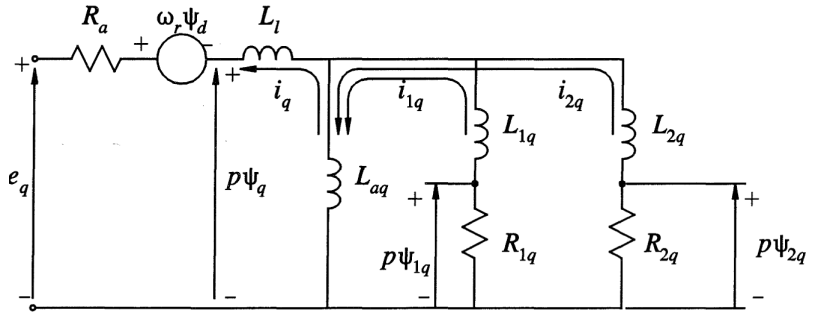
- Los flujos de dispersión no contribuyen a la saturación del hierro. Estas fugas son pequeñas y su trayectoria coincide en un pequeño tramo con el flujo principal. La saturación se determina por el acoplamiento magnético del entrehierro.

- La relación de saturación entre el flujo de entrehierro magnético y la mmf en condiciones de carga, es igual que en condiciones sin carga. Lo cual permite que las características de saturación se puedan representar con la curva de saturación de circuito abierto, siendo el dato del cual se dispone.

- No hay acoplamiento magnético entre los ejes d y q dado a la no linealidad introducido por la saturación.



(a)



(b)

Figura B.3: Circuito equivalente de los ejes d y q [1].

Partiendo de los suposiciones hechas anteriormente, se puede representar los efectos de la saturación mediante las siguientes expresiones.

$$L_{ad} = K_{sd}L_{adu} \tag{B.0.6}$$

$$L_{aq} = K_{sq}L_{aqu} \tag{B.0.7}$$

Donde, L_{adu} y L_{aqu} son inductancias no saturadas. L_{ad} y L_{aq} son los valores de inductancias saturadas. Los valores de K_{sd} y K_{sq}, son el grado de saturación de los respectivos ejes d y q.

El grado de saturación se determina de OCC. En la figura B.4, esta definido el punto de operación en "a" de la gráfica de OCC, el factor de saturación esta definido por:

$$K_{sd} = \frac{\psi_{at}}{\psi_{at0}} \tag{B.0.8}$$

Otra definición para K_{sd} es

$$K_{sd} = \frac{I_0}{I} \tag{B.0.9}$$

La expresión a utilizar es (B.0.8). Esta expresión da el grado de saturación para cualquier valor especificado de flujo de entrehierro o tensión.

La expresión matemática a utilizar para representar la variación de OCC de la línea del entrehierro es:

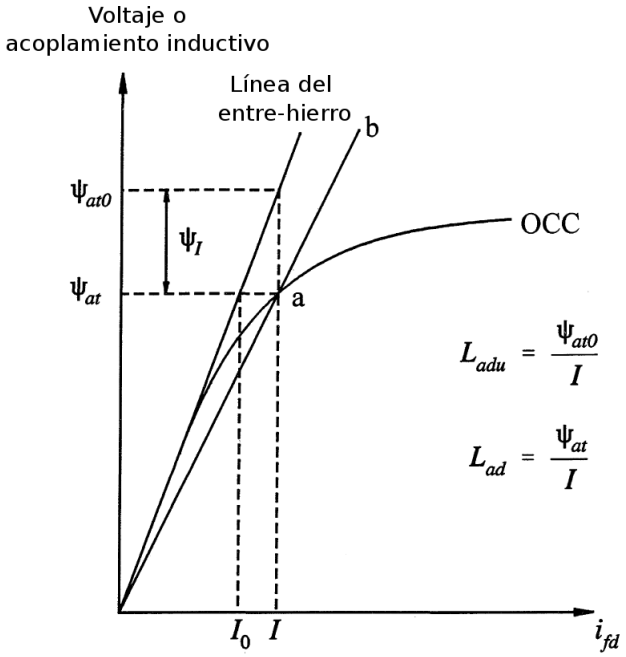


Figura B.4: Caracteristica de circuito abierto mostrando efectos de saturación [1].

$$\psi_I = \psi_{at0} - \psi_{at} \tag{B.0.10}$$

Por lo tanto la expresión del factor de saturación es:

$$K_{sd} = \frac{\psi_{at}}{\psi_{at} + \psi_I} \tag{B.0.11}$$

La curva de saturación se divide en tres segmentos:

Segmento I No saturado.

Segmento II No lineal.

Segmento III Completamente saturado.

En el segmento I se tiene $\psi_{at} \leqslant \psi_{TI}$.

$$\psi_I = 0 \tag{B.0.12}$$

En el segmento II se tiene $\psi_{at} < \psi_{at} \leqslant \psi_{T2}$. Donde, ψ_I se puede expresar en términos de una función exponencial

$$\psi_I = A_{sat} e^{B_{sat}(\psi_{at} - \psi_{TI})} \tag{B.0.13}$$

Donde, A_{sat} y B_{sat} son constantes que dependen de las características de saturación del segmento II. Cuando $\psi_{at} = \psi_{TI}$, de la expresión (B.0.13), entonces $\psi_I = A_{sat}$. Por lo tanto se presenta una pequeña discontinuidad en la frontera entre el segmento I y II. Sin embargo es muy pequeña por lo tanto la discontinuidad no tiene relevancia.

El segmento III esta definido pata $\psi_{at} > \psi_{T2}$,

$$\psi_I = \psi_{G2} + L_{ratio}(\psi_{at} - \psi_{T2}) - \psi_{at} \tag{B.0.14}$$

Donde L_{ratio}, se representa como se define en la figura B.5, la pendiente de la línea del entrehierro y la pendiente del segmento III de OCC.

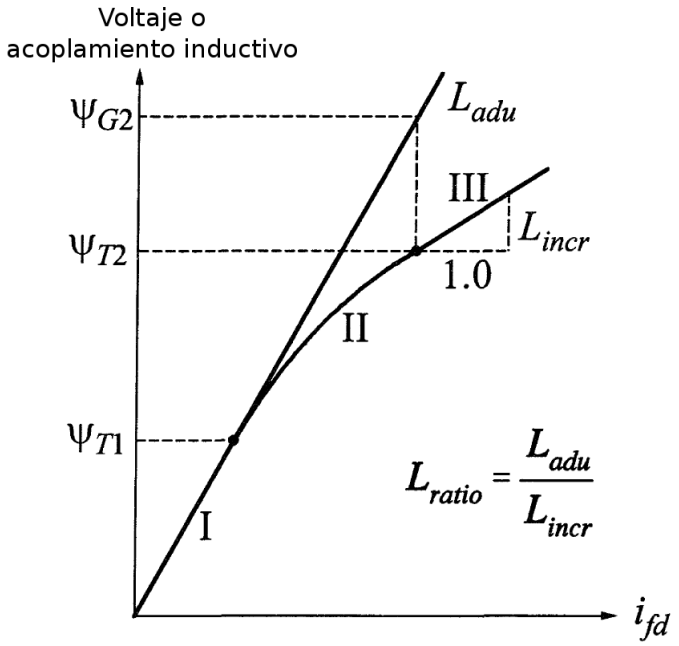


Figura B.5: Caracteristica de saturación [1].

La representación de la saturación para cualquier máquina se realiza por medio de ψ_{T1}, ψ_{T2}, ψ_{G2}, A_{sat}, B_{sat} y L_{ratio}. Las constantes de saturación para cualquier condición de operación en función del acople del flujo del entrehierro esta dado por

$$\psi_{at} = \sqrt{\psi_{ad}^2 + \psi_{ad}^2} \tag{B.0.15}$$

Donde ψ_{ad} y ψ_{aq}, son los componentes de los ejes d y q del entrehierro. El circuito se muestra en la figura (B.6a) y (B.6b) respectivamente.

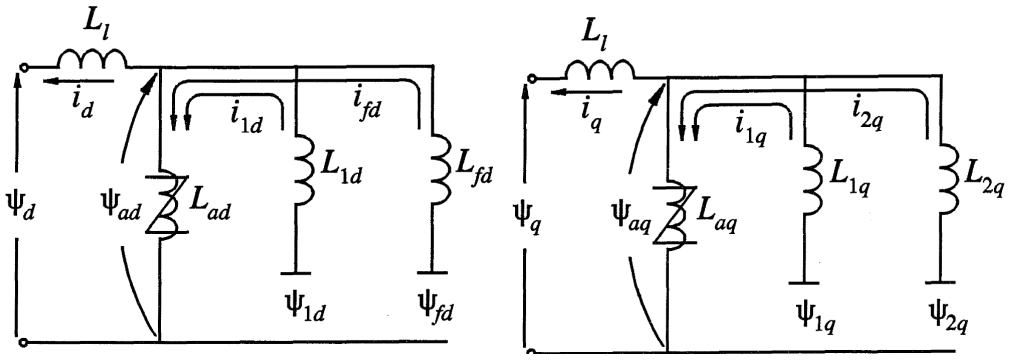
Los flujos de acople del entrehierro de los eje d y q están dado por:

$$\psi_{ad} = \psi_d + L_l i_d = e_q + R_a i_q + L_l i_d \tag{B.0.16}$$
$$\psi_{aq} = \psi_q + L_l i_q = -e_d - R_a i_d + L_l i_q \tag{B.0.17}$$

Por lo tanto ψ_{at} en pu es igual al tensión en el entrehierro

$$\widetilde{E}_a = \widetilde{E}_t + (R_a + jX_l)\widetilde{I}_t \tag{B.0.18}$$

El factor de saturación K_{sd} se determina para cualquier valor de tensión y corriente calculando $\widetilde{E}_a$ y luego usando las ecuaciones (B.0.11), (B.0.12), (B.0.13) y (B.0.14).



(a) (b)

Figura B.6: Circuito equivalente con elementos no lineales y acoplamiento del flujo del entrehierro [1].

Para máquinas de polos salientes, gran parte de la trayectoria del flujo del eje q se dispersa en el aire, L_{aq} no varia de manera significativa con la saturación de una parte del hierro. Por lo tanto, se asume que K_{sd} es igual a 1 para condiciones de plena carga.

Para máquinas de rotor cilíndrico, la saturación ocurre en ambos ejes. K_{sd} se determina a partir de la saturación característica sin carga del eje q. Sin embargo los datos de saturación del eje q no esta disponible, entonces se supone que k_{sq} es igual a k_{sd}. Se asume que la reluctancia de la trayectoria magnética es homogénea alrededor de la periferia del rotor.

Anexo C

Definición de conceptos a partir del lenguaje C

El kernel

Los kernels son funciones que cuando son llamadas, son ejecutadas N veces en paralelo por N hilos diferentes de CUDA. la función se declara con la palabra __global__ void y cuando es invocada se utilizan los símbolos <<< ... >>> para especificarle el número de hilos en los cuales ejecutarse. Cada subproceso en su interior, se le asigna un identificador que puede ser leído desde el interior del kernel por medio del la variable threadIdx.x. El siguiente ejemplo consiste en sumar dos vectores y guardar el valor en un tercero.

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

Figura C.1: Declaración del kernel [5].

Jerarquía de hilos

Para una mayor facilidad el hilo es un vector de tres componentes, que pueden ser identificadas por medio de una, dos o tres dimensiones, formando un bloque de hilos de uno, dos o tres dimensiones, y de esta manera poder hacer cálculos a través de un vector, matriz o volumen. La relación es directa entre el índice de un hilo y su ID. La cuál es la siguiente:

- Bloque de una dimensión se mantiene igual a x.

- Bloque de dos dimensiones D_x, D_y teniendo como índices (x, y) es $x + yD_x$
- Bloque de tres dimensiones D_x, D_y, D_z con índices (x, y, z) es $x+yD_x+zD_xD_y$

Ejemplo para una dimensión.

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
                       float C[N][N])
{
    int i = threadIdx.x;
    int j = threadIdx.y;
    C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation with one block of N * N * 1 threads
    int numBlocks = 1;
    dim3 threadsPerBlock(N, N);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

Figura C.2: Suma de 2 matrices con bloque de una dimensión [5].

Bloques de múltiple dimensión.

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation
    dim3 threadsPerBlock(16, 16);
    dim3 numBlocks(N / threadsPerBlock.x, N / threadsPerBlock.y);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

Figura C.3: Suma de 2 matrices con bloques de múltiple dimensión [5].

Un kernel puede ejecutar múltiples bloques de hilos en el cuál, el número de hilos es igual al número de hilos por bloque y por el número de bloques. Los bloques tienen 3 dimensiones, el número de bloques en una malla está determinado por la cantidad de datos a procesar o la cantidad de procesos del sistema. Sin embargo, hay un número límite de hilos por bloque. Actuálmente se tiene un máximo 1024 hilos por bloque.

Los bloques por malla e hilos por bloque que se especifican en <<< ... >>> son de tipo int o dim3. Cada bloque dentro de la malla se identifica por un índice que puede tener una, dos o tres dimensiones, al cuál se accede desde el kernel con la variable blockIdx. También se accede desde el kernel a la dimensión de los bloques de

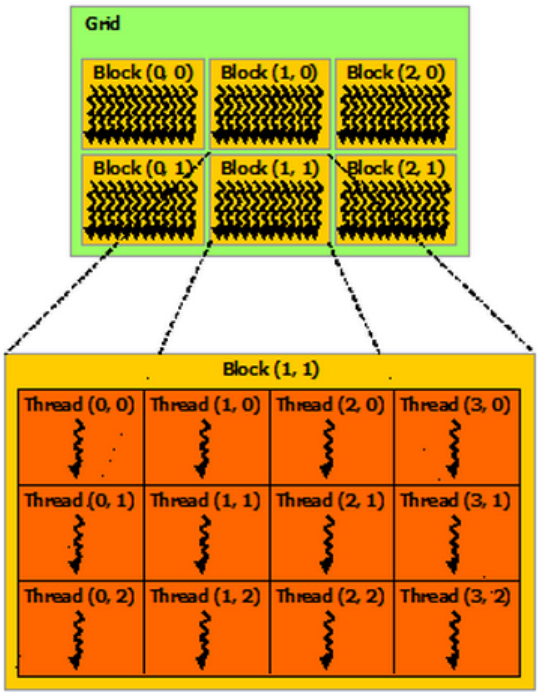


Figura C.4: Malla de bloques de hilo [5].

hilos por medio de la variable blockDim. La ejecución de bloques de hilos se puede realizar en cualquier orden, ya sea en serie o peralelo. Esta independencia permite ser programadas en cualquier número de núcleos y escalarse de manera automática.

Los hilos dentro de un bloque pueden compartir información por medio de la isntrucción __shared__ y sisncronizar la información para coordinar el acceso a la memoria. La instrucción encargada de la sincronización es __syncthreads, que actúa como una barrera para todos los hilos del bloque esperen en la respectiva tarea, y así poder continuar.

Jerarquía de memoria

El modelo de programación CUDA se trabaja con dos dispositivos físicamente separados, conocidos como el host y el device. En le host se ejecuta el código secuencial y en el device se ejecuta la parte del código paralelizable. cada una cuenta con su propia memoria DRAM separadas que se les conoce como memoria principal y memoria del dispositivo. Esto implica que al ejecutarse el respectivo código, se necesita realizar la respectiva gestión de los espacios de la memoria. Por lo tanto, hay que reservar el espacio de la memoria del dispositivo y luego transferir la información desde la memoria alfitrión a la memoria del dispositivo y viceversa, una vez procesada la información. Las instrucciones encargadas de esta tarea son:

cudaMalloc: Encargada de reservar el espacio de memoria especificado, en la DRAM del dispositivo y se declara como cudaMalloc(&d_A, tamaño de la memoria a reservar).

cudaMemcpy: Encargada de transferir la información entre la memoria principal y la del dispositivo. La forma en la que se declara es la siguiente:

- cudaMemcpy(d_A, h_A, tamaño, cudaMemcpyHostToDevice), en el cuál se transfiere los datos desde la memoria del host al device.
- cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost), en el que se hace la operación de transferir los datos desde la memoria del device al host.

cudaFree: Encargada de borrar la información del dispositivo y su escritura es cudaFree(d_A).

Capacidad de cómputo

Se define con un número de revisión que índica la arquitectura del núcleo. Los dispositivos de igual número significa que son de igual arquitectura, los que tienen el número 3 corresponden a la arquitectura keepler, los que tienen el número 2 corresponden a la arquitectura fermi, y por último, las correspondientes a la serie tesla con el número 1.