ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN MÓDULO DE SOFTWARE EN EL LENGUAJE DE PROGRAMACIÓN FORTRAN PARA SIMULACIÓN DE EVENTOS TRANSITORIOS ELECTROMAGNÉTICOS EN UNIDADES DE GENERACIÓN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

JOSÉ ANTONIO CARRILLO NIETO
jose.carrillo@epn.edu.ec

DIRECTOR: MSc. – ING. JUAN CARLOS PLAZARTE ACHIG.
juan.plazarte@celec.gob.ec

CODIRECTOR: DR. – ING. HUGO NEPTALÍ ARCOS MARTÍNEZ.
hugo.arcos@epn.edu.ec

Quito, junio 2019

AVAL

Certificamos que el presente trabajo fue desarrollado por José Antonio Carrillo Nieto, bajo nuestra supervisión.

Msc. – ING. JUAN PLAZARTE
DIRECTOR DEL TRABAJO DE TITULACIÓN

DR. – ING. HUGO ARCOS
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Jose Antonio Carrillo Nieto, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

JOSÉ ANTONIO CARRILLO NIETO

DEDICATORIA

A MI MADRE, HERMANOS y ABUELITO ANTONIO NIETO

A mi Madre por ser mi motor, mi norte, mi guía, por impulsarme, por enseñarme a ser perseverante. Gracias por sus desvelos, lágrimas y fortaleza, por mantenerme firme en la fe de que los tiempos de Dios son perfectos.

A esos seres mágicos que son mis hermanos Daniel, Toty y Nicolay gracias por existir, por formar parte de mi vida, por su apoyo incondicional, por ser mis cómplices, por nuestras travesuras y amor filial, a ustedes dedico este trabajo.

A mi “Ser de luz”, que siempre siento su presencia y su bendición, a usted que me enseñó los valores de lealtad, honradez y respeto, para usted va dedicado mis logros Abuelito Antonio Nieto.

También va grabado en mi ser el cariño y consideración a mi familia, un agradecimiento especial a mis tíos Washington y Margoth Nieto Cortez.

Gracias por todo y por tanto.

AGRADECIMIENTO

Cuando un individuo ostenta un título profesional, no es el reflejo solamente de su esfuerzo, sacrificio y preparación académica, también implica quienes son los mentores de esos grandes ideales de libertad, verdad y conocimiento que son bases espirituales que dignifican al ser humano.

Mi mensaje de gratitud y afecto a mis respetados Profesores, Director y Codirector del presente trabajo de titulación, por su legado y dedicación en transmitirme los conocimientos elementales y por permitirme ser parte de sus preceptos.

Es un privilegio y me congratulo el haber culminado mi carrera universitaria y mi compromiso de ser ejemplo de respeto, responsabilidad y dignidad; como, tantas personas que se han destacado profesionalmente, que un día egresaron de la ESCUELA POLITÉCNICA NACIONAL.

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1. INTRODUCCIÓN 1
1.1. Objetivos 2
1.1.1. Objetivo general 2
1.1.2. Objetivos específicos 2
1.2. Alcance 2
1.3. Marco Teórico 3
1.3.1. Dinámica del Generador Sincrónico 3
1.3.1.1. Transformada de Park – sistema qd0 8
1.3.1.2. Transformada de las concatenaciones de flujo ψ al sistema qd0 9
1.3.1.3. Transformada del voltaje del estator al sistema qd0 11
1.3.1.4. Transformada del torque eléctrico al sistema qd0 13
1.3.1.5. Parámetros en por unidad 14
1.3.2. Modelos de Generador Sincrónico Según el Tipo de Rotor 16
1.3.2.1. Aplicación del modelo 2.2 del estándar IEEE 1001 17
1.3.2.2. Aplicación del modelo 2.1 del estándar IEEE 1001 19
1.3.3. Sistema de Excitación y Regulador de Voltaje – AVR 20
1.3.4. Sistema de Control de Velocidad – GOV 24
1.3.4.1. Turbina hidráulica 24
1.3.4.2. Control de velocidad (gobernador) 25
1.3.5. Método Runge-Kutta de Cuarto Orden para Resolver Sistemas de Ecuaciones Diferenciales 26
1.3.6. Introducción a FORTRAN 30
1.3.6.1. Ventajas 30

1.3.6.2. Desventajas 30
2. METODOLOGÍA 31
2.1. Arreglo de Ecuaciones del Sistema Máquina Sincrónica – Barra Infinita 31
2.2. Arreglo de Ecuaciones del Regulador de Voltaje – AVR 35
2.3. Arreglo de Ecuaciones del Regulador de Velocidad – GOV 38
2.4. Inicialización del Sistema. 40
2.4.1. Condiciones iniciales del sistema generador-barra infinita 40
2.4.2. Condiciones iniciales del regulador de voltaje 42
2.4.3. Condiciones iniciales del regulador de velocidad 43
2.5. Instalación del Software para Programar en Fortran 43
2.5.1. Visual Studio 44
2.5.2. Intel Parallel Studio XE 46
2.5.3. Gnuplot 46
2.6. Simulación del Sistema en Fortran 48
2.7. Simulación del Sistema en DIgSILENT PowerFactory 55
3. RESULTADOS 57
3.1. Comparación de Simulaciones de Eventos entre Fortran y DIgSILENT 57
3.1.1. Variación en el voltaje de referencia 57
3.1.2. Variación en la velocidad de referencia 60
3.1.3. Variación en la potencia de referencia 63
3.1.4. Falla trifásica 66
3.2. Análisis de Resultados 68
4. CONCLUSIONES 70
5. REFERENCIAS BIBLIOGRÁFICAS 71
6. ANEXOS 73
ANEXO I 74
ANEXO II 79
ANEXO III 85
ORDEN DE EMPASTADO 98

RESUMEN

En el presente trabajo se desarrolla un módulo de software que simula la respuesta de una unidad de generación hidráulica ante diferentes eventos transitorios. Dicha unidad está compuesta de un generador sincrónico de polos salientes, un regulador de voltaje IEEE DC1A (AVR) y un regulador de velocidad y turbina hidráulica IEEE G2 (GOV). El sistema eléctrico incluye además una línea de transmisión conectada al generador sincrónico y una barra infinita. Los eventos disponibles para las simulaciones son: cortocircuitos trifásicos (en la barra infinita, línea de transmisión y terminales del generador), variación en el voltaje de referencia del AVR y variaciones en la velocidad y potencia de referencia del GOV.

Las señales de salida del generador para el análisis de eventos transitorios son: voltaje y corriente en terminales, voltaje de excitación, potencia activa y reactiva generada, ángulo de potencia, velocidad angular y torque eléctrico del rotor. Para comprobar la validez del software desarrollado, dichas señales se comparan con simulaciones en el programa comercial DIgSILENT PowerFactory.

También se incluye la deducción de las ecuaciones empleadas, en base a los estándares publicados por la IEEE, así como indicaciones fundamentales sobre el uso del lenguaje de programación Fortran.

PALABRAS CLAVE: Dinámica de generador sincrónico, simulación AVR, gobernador, programación en Fortran

ABSTRACT

In this work, a software module is developed that simulates the response of a hydraulic generation unit to different transitory events. Said unit is composed of a salients pole synchronous generator, an IEEE DC1A voltage regulator (AVR) and an IEEE G2 hydraulic turbine and speed governor (GOV). Also, the electrical system includes a transmission line connected to the synchronous generator and an infinite bus bar, the events available for the simulations are: three-phase short circuits (at infinite bus bar, at transmission line and at generator terminals), variation in the AVR voltage reference and speed and power reference variations in the GOV.

The output signals of the generator for the analysis of transitory events are: voltage and current at terminals, excitation voltage, active and reactive power generated, power angle, angular speed and electric torque of the rotor. To check the validity of the software developed, these signals are compared with simulations made in DIgSILENT PowerFactory commercial program.

Also is included the deduction of the equations used, based on the standards published by the IEEE, as well as fundamental indications refered to use of Fortran programming language.

KEYWORDS: Synchronous generator dynamics, AVR simulation, governor, Fortran Programming

1. INTRODUCCIÓN

El estudio de estabilidad transitoria y la modelación de unidades de generación en sistemas de potencia es de gran importancia en las prácticas docentes, por lo tanto, se han realizado diferentes intentos en la modelación de generadores y de sistemas de regulación de velocidad y voltaje, sin embargo, no se ha tenido el impulso para contar con un software de código abierto, de fácil acceso y de ejecución didáctica; por este motivo en este trabajo se ha investigado y analizado los modelos matemáticos apropiados para el desarrollo de una herramienta computacional para la simulación de unidades de generación.

La máquina síncrona desempeña un papel fundamental en todo SEP, inclusive se podría considerar como el vínculo entre el mundo eléctrico y el mecánico. Por lo tanto, es importante una comprensión precisa de su dinámica. Consecuentemente, esta tesis proporciona un modelo detallado de la máquina síncrona y controladores además de simulaciones con eventos en los diferentes parámetros del sistema.

Para conseguir que el software didáctico sea bastante preciso es necesario una fuerte basé teórica que respalde las ecuaciones implementadas, para ello, el análisis matemático de la dinámica del generador y la deducción de las ecuaciones diferenciales de los reguladores, se basan en bibliografía relacionada con estándares de la IEEE.

La programación en Fortran tiene una notable ventaja en cuanto a velocidad de procesamiento en comparación a otros lenguajes como Python, Matlab o C++. Además, los programas actuales que emplean una interface gráfica tienen altos tiempos de ejecución y muchas veces no se tiene acceso a la programación básica dentro de los bloques o paquetes que se emplean en la modelación de máquinas eléctricas; debido a lo cual no se puede contrastar ningún tipo de información parcial del sistema.

Una de las mayores dificultades en la simulación de sistemas eléctricos es la implementación del método para resolver los sistemas de ecuaciones que surgen durante la modelación, en este trabajo se presenta una de las mejores alternativas: el método Runge-Kutta de cuarto orden, este algoritmo es sencillo de aplicar y sobretodo es bastante preciso en la resolución de ecuaciones diferenciales ordinarias.

1.1. Objetivos

1.1.1. Objetivo general:

- Desarrollar simulaciones de la respuesta transitoria de una unidad de generación ante diferentes eventos, con el fin de lograr mejores resultados en cuanto a velocidad de procesamiento y consumo de memoria mediante el uso del lenguaje de programación Fortran.

1.1.2. Objetivos específicos:

- Estructurar una herramienta de software para el estudio de fenómenos transitorios en máquinas eléctricas mediante programación en Fortran; mejorando la velocidad de procesamiento, consumo de memoria, eficiencia y portabilidad.
- Observar la evolución transitoria de las variables en un generador hidroeléctrico, pudiendo elegir los eventos a analizar como variación del voltaje de referencia, potencia de referencia y velocidad de referencia.
- Modelar un regulador de velocidad y un regulador de voltaje; realizar pruebas y validar las respuestas.
- Analizar el comportamiento del generador ante diferentes perturbaciones y comparar los resultados con el software comercial DIgSILENT

1.2. Alcance

Desarrollar un software de simulación programado en lenguaje Fortran; que permitirá la interacción usuario – máquina para la modelación y análisis de respuesta transitoria de una unidad de generación hidráulica; esta unidad incluye: un generador sincrónico, un regulador de velocidad y un regulador de voltaje.

Para realizar las simulaciones de los diferentes eventos se emplea un sistema simple que consiste en el generador sincrónico conectado a una línea de transmisión y a una barra infinita. Los eventos transitorios disponibles son variaciones en: voltaje de referencia, potencia de referencia y velocidad de referencia; además el usuario podrá ingresar los parámetros del generador y de los reguladores.

El resultado de las pruebas se compara con las simulaciones obtenidas en DIgSILENT, de esta manera se demuestra la validez del software desarrollado. A su vez este programa sirve para analizar el efecto de los reguladores de voltaje y velocidad sobre el generador ante fenómenos transitorios, mediante la observación y estudio de los resultados obtenidos.

1.3. Marco Teórico

1.3.1. Dinámica del Generador Sincrónico [1]

La representación de la máquina sincrónica para el estudio dinámico se indica en la figura 1.1, allí se muestra al estator o armadura con los tres devanados de fase ($a, b\ y\ c$) y al rotor con cuatro devanados; uno de campo (f) y tres de amortiguamiento (g, kd, kq). La cantidad de devanados de amortiguamiento pueden variar desde cero en los modelos más simples, hasta más de cinco en modelos muy detallados, además el número de devanados se considera dependiendo del tipo de rotor sea este cilíndrico o de polos salientes.

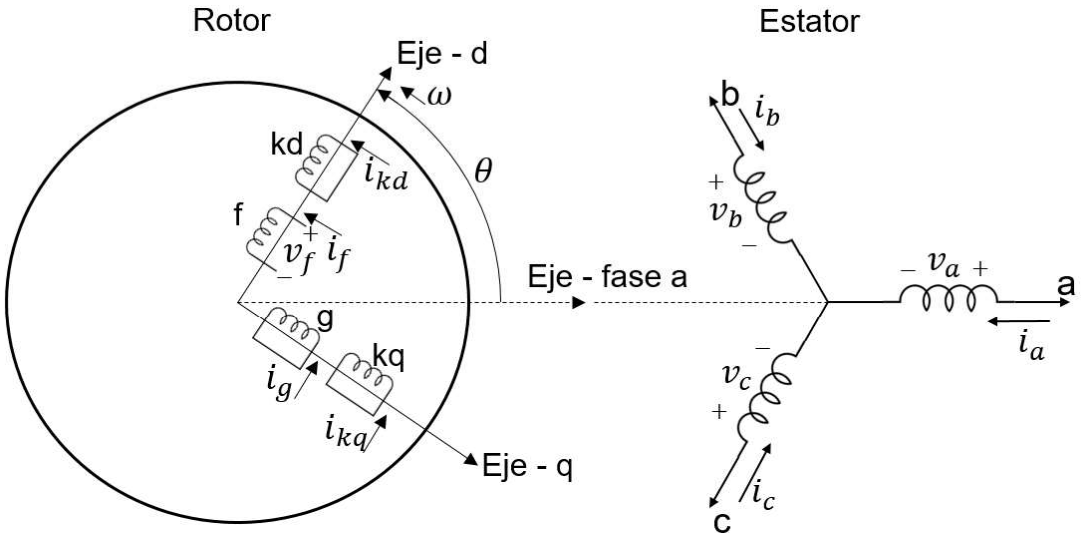


Figura 1.1 Diagrama de la máquina sincrónica [1]

Para la deducción de las ecuaciones de la máquina sincrónica se hacen las siguientes consideraciones:

- La fuerza magneto-motriz (fmm) en el entrehierro es distribuida sinusoidalmente, por lo tanto, los armónicos son despreciables.
- No se considera el efecto de las ranuras del estator sobre las reactancias del rotor
- Se ignora la histéresis y saturación magnética.
- Se asume que la máquina tiene dos polos. Con más polos simplemente se debe multiplicar la ecuación del torque eléctrico por el factor P/2, donde P es el número de polos.

Las ecuaciones 1.1, 1.9 y 1.14 rigen el comportamiento dinámico de un generador sincrónico. A partir de estas fórmulas es posible calcular los parámetros de la máquina que se usan en los flujos de potencia de un sistema eléctrico [1].

$$\begin{bmatrix}\psi_s\\ \psi_r\end{bmatrix} = \begin{bmatrix}L_{ss} & L_{sr}\\ L_{rs} & L_{rr}\end{bmatrix}\begin{bmatrix}i_s\\ i_r\end{bmatrix}$$

Ecuación 1.1 Concatenaciones de flujo del estator y rotor

Donde:

$$\psi_s = \begin{bmatrix}\psi_a\\ \psi_b\\ \psi_c\end{bmatrix}$$

Ecuación 1.2 Concatenaciones de flujo del estator

$$\psi_r = \begin{bmatrix}\psi_f\\ \psi_{kd}\\ \psi_g\\ \psi_{kq}\end{bmatrix}$$

Ecuación 1.3 Concatenaciones de flujo del rotor

$$[L_{ss}] = \begin{bmatrix} L_{aa0} & L_{ba0} & L_{ba0} \\ L_{ba0} & L_{aa0} & L_{ba0} \\ L_{ba0} & L_{ba0} & L_{aa0} \end{bmatrix} + L_{aa2} \begin{bmatrix} \cos 2\theta & \cos\left(2\theta - \frac{2\pi}{3}\right) & \cos\left(2\theta + \frac{2\pi}{3}\right) \\ \cos\left(2\theta - \frac{2\pi}{3}\right) & \cos\left(2\theta + \frac{2\pi}{3}\right) & \cos 2\theta \\ \cos\left(2\theta + \frac{2\pi}{3}\right) & \cos 2\theta & \cos\left(2\theta - \frac{2\pi}{3}\right) \end{bmatrix}$$

Ecuación 1.4 Matriz de auto-inductancia del estator

Donde:

$L_{aa0} = L_{al} + L_{g0}$

$L_{ab0} = L_{abl} + L_{g0}$

$L_{aa2} = L_{gaa} - L_{g0}$

L_{al} Inductancia de dispersión de una de las fases

L_{abl} Inductancia de dispersión mutua entre fases

L_{g0} Inductancia del entrehierro

L_{gaa} Inductancia mutua entre fases y entrehierro

$$[L_{rr}] = \begin{bmatrix} L_f & L_{fkd} & 0 & 0 \\ L_{fkd} & L_{kd} & 0 & 0 \\ 0 & 0 & L_g & L_{gkq} \\ 0 & 0 & L_{gkq} & L_{kq} \end{bmatrix}$$

Ecuación 1.5 Matriz de auto-inductancia del rotor

$$[L_{sr}] = \begin{bmatrix} L_{af}\cos\theta & L_{akd}\cos\theta & L_{ag}\,\mathrm{sen}\,\theta & L_{akq}\,\mathrm{sen}\,\theta \\ L_{af}\cos\left(\theta - \frac{2\pi}{3}\right) & L_{akd}\cos\left(\theta - \frac{2\pi}{3}\right) & L_{ag}\mathrm{sen}\left(\theta - \frac{2\pi}{3}\right) & L_{akq}\mathrm{sen}\left(\theta - \frac{2\pi}{3}\right) \\ L_{af}\cos\left(\theta + \frac{2\pi}{3}\right) & L_{akd}\cos\left(\theta + \frac{2\pi}{3}\right) & L_{ag}\mathrm{sen}\left(\theta + \frac{2\pi}{3}\right) & L_{akq}\mathrm{sen}\left(\theta + \frac{2\pi}{3}\right) \end{bmatrix}$$

Ecuación 1.6 Matriz de inductancia mutua rotor - estator

El subíndice a re refiere a una de las fases del estator

$$i_s = \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}$$

Ecuación 1.7 Matriz de corrientes del estator

$$i_r = \begin{bmatrix} i_f \\ i_{kd} \\ i_g \\ i_{kq} \end{bmatrix}$$

Ecuación 1.8 Matriz de corrientes del rotor:

$$\begin{bmatrix} v_s \\ v_r \end{bmatrix} = \begin{bmatrix} -R_s & 0 \\ 0 & -R_r \end{bmatrix} \begin{bmatrix} i_s \\ i_r \end{bmatrix} - \begin{bmatrix} d\psi_s/dt \\ d\psi_r/dt \end{bmatrix}$$

Ecuación 1.9 Voltaje del estator y rotor

Donde:

$$v_s = \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}$$

Ecuación 1.10 Matriz de voltajes del estator

$$v_r = \begin{bmatrix} -v_f \\ 0 \\ 0 \end{bmatrix}$$

Ecuación 1.11 Matriz de voltajes en el rotor

$$[R_s] = \begin{bmatrix} R_a & 0 & 0 \\ 0 & R_a & 0 \\ 0 & 0 & R_a \end{bmatrix}$$

Ecuación 1.12 Matriz de resistencia del estator de la fase a

$$[R_r] = \begin{bmatrix} R_f & 0 & 0 & 0 \\ 0 & R_{kd} & 0 & 0 \\ 0 & 0 & R_g & 0 \\ 0 & 0 & 0 & R_{kq} \end{bmatrix}$$

Ecuación 1.13 Matriz de resistencias del rotor

$$T_m - T_e = J\frac{d^2\theta_m}{dt^2} + D\frac{d\theta_m}{dt}$$

Ecuación 1.14 Fuerza motriz del rotor

Donde:

T_m Torque mecánico en dirección de la rotación

T_e Torque eléctrico opuesto al torque mecánico

J Momento de inercia total del rotor

D Coeficiente de amortiguamiento

θ_m Ángulo mecánico del rotor

θ Ángulo eléctrico del rotor, si P=2 entonces $\theta = \theta_m$

$$T_e = -\frac{\partial W}{\partial \theta}$$

Ecuación 1.15 Torque eléctrico

$$W = \frac{1}{2}\begin{bmatrix} i_s^T & i_r^T \end{bmatrix}\begin{bmatrix} L_{ss} & L_{sr} \\ L_{rs} & L_{rr} \end{bmatrix}\begin{bmatrix} i_s \\ i_r \end{bmatrix}$$

Ecuación 1.16 Matriz de co-energía

Substituyendo 1.16 en 1.15 y considerando que la matriz L_{rr} es constante, se obtiene la expresión desarrollada del torque eléctrico.

$$T_e = -\frac{1}{2}\left[i_s^T\left[\frac{\partial L_{ss}}{\partial\theta}\right]i_s + 2i_s^T\left[\frac{\partial L_{sr}}{\partial\theta}\right]i_r\right]$$

Ecuación 1.17 Torque eléctrico

De las ecuaciones 1.1 y 1.9 se forma el siguiente sistema:

$$\begin{cases}\begin{bmatrix}d\psi_s/dt\\ d\psi_r/dt\end{bmatrix} = \begin{bmatrix}-R_s & 0\\ 0 & -R_r\end{bmatrix}\begin{bmatrix}L_{ss} & L_{sr}\\ L_{rs} & L_{rr}\end{bmatrix}^{-1}\begin{bmatrix}\psi_s\\ \psi_r\end{bmatrix} - \begin{bmatrix}v_s\\ v_r\end{bmatrix}\\ \begin{bmatrix}i_s\\ i_r\end{bmatrix} = \begin{bmatrix}L_{ss} & L_{sr}\\ L_{rs} & L_{rr}\end{bmatrix}^{-1}\begin{bmatrix}\psi_s\\ \psi_r\end{bmatrix}\end{cases}$$

Ecuación 1.18 Sistema de ecuaciones diferenciales de la máquina sincrónica

1.3.1.1. Transformada de Park – sistema qd0

El sistema de ecuaciones 1.18 podría resolverse mediante métodos numéricos, pero como se aprecia, las ecuaciones 1.4 y 1.6 varían en el tiempo al estar en función de θ, por lo tanto la solución analítica es muy difícil de obtener, inclusive con complejos algoritmos ya que los tiempos de ejecución serían muy elevados [1].

Por ese motivo es muy conveniente convertir las ecuaciones cuyas inductancias varían con el tiempo en ecuaciones invariantes en el tiempo. Para lograr esto R. H. Park introdujo la siguiente trasformación [1]:

$$[f_\alpha] = [C_P]\begin{bmatrix}f_d\\ f_q\\ f_0\end{bmatrix}$$

Ecuación 1.19 Transformada de Park de f_α

Donde f_α pude ser voltajes, corrientes o concatenaciones de flujo

$$[C_P] = \frac{1}{\sqrt{3}}\begin{bmatrix}\sqrt{2}\cos\theta & \sqrt{2}\text{sen}\,\theta & 1\\ \sqrt{2}\cos\left(\theta-\frac{2\pi}{3}\right) & \sqrt{2}\,\text{sen}\left(\theta-\frac{2\pi}{3}\right) & 1\\ \sqrt{2}\cos\left(\theta+\frac{2\pi}{3}\right) & \sqrt{2}\,\text{sen}\left(\theta+\frac{2\pi}{3}\right) & 1\end{bmatrix}$$

Ecuación 1.20 Matriz de transformación a coordenadas arbitrarias qd0

Las funciones en las coordenadas qd0 se obtienen invirtiendo la ecuación 1.20

$$\begin{bmatrix} f_d \\ f_q \\ f_0 \end{bmatrix} = [C_P]^{-1}[f_\alpha]$$

Ecuación 1.21 Transformada de f_α en las coordenadas arbitrarias qd0

$$[C_P]^{-1} = \frac{1}{\sqrt{3}}\begin{bmatrix} \sqrt{2}\cos\theta & \sqrt{2}\cos\left(\theta - \frac{2\pi}{3}\right) & \sqrt{2}\cos\left(\theta + \frac{2\pi}{3}\right) \\ \sqrt{2}\,\text{sen}\,\theta & \sqrt{2}\,\text{sen}\left(\theta - \frac{2\pi}{3}\right) & \sqrt{2}\,\text{sen}\left(\theta + \frac{2\pi}{3}\right) \\ 1 & 1 & 1 \end{bmatrix}$$

Ecuación 1.22 Matriz inversa de C_P

1.3.1.2. Transformada de las concatenaciones de flujo ψ al sistema qd0

En base a la ecuación 1.19 se determina la transformada de Park de las ecuaciones del generador

$$\begin{bmatrix} \psi_s \\ \psi_r \end{bmatrix} = \begin{bmatrix} C_P & 0 \\ 0 & U_4 \end{bmatrix}\begin{bmatrix} \psi_{dq0} \\ i_r \end{bmatrix}$$

Ecuación 1.23 Aplicación de la transforma de Park para ψ_s

$$\begin{bmatrix} \psi_s \\ \psi_r \end{bmatrix} = \begin{bmatrix} L_{ss} & L_{sr} \\ L_{rs} & L_{rr} \end{bmatrix}\begin{bmatrix} C_P & 0 \\ 0 & U_4 \end{bmatrix}\begin{bmatrix} i_{dq0} \\ i_r \end{bmatrix}$$

Ecuación 1.24 Aplicación de la transforma de Park para i_s

Donde: $\psi_{dq0} = \begin{bmatrix}\psi_d\ \psi_q\ \psi_0\end{bmatrix}^T, i_{dq0} = \begin{bmatrix} i_d\ i_q\ i_0 \end{bmatrix}^T$, U_4: Matriz unitaria de orden 4

Reemplazando la ecuación 1.24 en la ecuación 1.23 se obtiene:

$$\begin{bmatrix} \psi_{dq0} \\ \psi_r \end{bmatrix} = \begin{bmatrix} C_P^{-1} & 0 \\ 0 & U_4 \end{bmatrix}\begin{bmatrix} L_{ss} & L_{sr} \\ L_{rs} & L_{rr} \end{bmatrix}\begin{bmatrix} C_P & 0 \\ 0 & U_4 \end{bmatrix}\begin{bmatrix} i_{dq0} \\ i_r \end{bmatrix} = \begin{bmatrix} L'_{ss} & L'_{sr} \\ L'_{rs} & L_{rr} \end{bmatrix}\begin{bmatrix} i_{dq0} \\ i_r \end{bmatrix}$$

Ecuación 1.25 Concatenaciones de flujo en el sistema qd0

Donde:

$$L'_{ss} = C_P^{-1} L_{ss} C_P = \begin{bmatrix} L_d & 0 & 0 \\ 0 & L_q & 0 \\ 0 & 0 & L_0 \end{bmatrix}$$

Ecuación 1.26 Matriz de impedancia del estator en el sistema qd0

$$L_d = L_{aa0} - L_{ab0} + \frac{3}{2} L_{aa2}$$
$$L_q = L_{aa0} - L_{ab0} - \frac{3}{2} L_{aa2}$$
$$L_0 = L_{aa0} - 2L_{ab0}$$

Ecuaciones 1.27 Impedancias en el sistema qd0

$$L'_{sr} = \begin{bmatrix} \sqrt{\frac{3}{2}} L_{af} & \sqrt{\frac{3}{2}} L_{akd} & 0 & 0 \\ 0 & 0 & \sqrt{\frac{3}{2}} L_{ag} & \sqrt{\frac{3}{2}} L_{akq} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ecuación 1.28 Matriz de impedancia mutua en el sistema qd0

$$L'_{rs} = \begin{bmatrix} \sqrt{\frac{3}{2}} L_{af} & 0 & 0 \\ \sqrt{\frac{3}{2}} L_{akd} & 0 & 0 \\ 0 & \sqrt{\frac{3}{2}} L_{ag} & 0 \\ 0 & \sqrt{\frac{3}{2}} L_{akq} & 0 \end{bmatrix}$$

Ecuación 1.29 Matriz de impedancia mutua en el sistema qd0

1.3.1.3. Transformada del voltaje del estator al sistema qd0

Aplicando la transformada de Park a la ecuación 1.9 se obtiene:

$$[C_P]v_{dq0} = -\frac{d}{dt}\left[C_P\psi_{dq0}\right] - [R_s][C_P]i_{dq0}$$

Ecuación 1.30 Aplicación de la transformada de Park para v_s

La derivada con respecto al tiempo de $C_P\psi_{dq0}$ puede representarse como:

$$-\frac{d}{dt}\left[C_P\psi_{dq0}\right] = -\dot{\theta}\frac{d[C_P]}{d\theta}\psi_{dq0} - [C_P]\frac{d}{dt}\psi_{dq0}$$

Ecuación 1.31 Derivada del producto $C_P \cdot \psi_{dq0}$

Donde $\dot{\theta}$ es la velocidad angular del rotor

$$\frac{d[C_P]}{d\theta} = \frac{1}{\sqrt{3}}\begin{bmatrix} -\sqrt{2}\,\mathrm{sen}\,\theta & \sqrt{2}\cos\theta & 0 \\ -\sqrt{2}\,\mathrm{sen}\left(\theta - \frac{2\pi}{3}\right) & \sqrt{2}\cos\left(\theta - \frac{2\pi}{3}\right) & 0 \\ -\sqrt{2}\,\mathrm{sen}\left(\theta + \frac{2\pi}{3}\right) & \sqrt{2}\cos\left(\theta + \frac{2\pi}{3}\right) & 0 \end{bmatrix}$$

Ecuación 1.32 Derivada de la matriz C_P con respecto a θ

La ecuación 1.32 puede rescribirse en función de $[C_P]$ de la siguiente manera:

$$\frac{d[C_P]}{d\theta} = [C_P]\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = [C_P][P_1]$$

Ecuación 1.33 Derivada de la matriz C_P

Nótese que $[P_1] = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Substituyendo la ecuación 1.31 en 1.30 y operando se obtiene la ecuación 1.34. Se debe considerar que $[R_s] = R_a[U_3]$ (ecuación 1.12). Donde U_3 es una matriz unitaria de orden 3.

$$v_{dq0} = -\frac{d}{dt}\psi_{dqo} - \dot{\theta}[P_1]\psi_{dq0} - R_a i_{dq0}$$

Ecuación 1.34 Voltaje del estator en el sistema dq0

Al expandir la ecuación 1.34 da como resultado:

$$v_d = -\frac{d}{dt}\psi_d - \dot{\theta}\psi_q - R_a i_d$$

$$v_q = -\frac{d}{dt}\psi_q - \dot{\theta}\psi_d - R_a i_q$$

$$v_0 = -\frac{d}{dt}\psi_0 - R_a i_0$$

Ecuaciones 1.35 Voltajes del estator en el sistema qd0

El voltaje en el rotor no cambia, por lo que este es igual a:

$$v_f = \frac{d}{dt}\psi_f + R_f i_f$$

$$0 = \frac{d}{dt}\psi_{kd} + R_{kd} i_{kd}$$

$$0 = \frac{d}{dt}\psi_g + R_g i_g$$

$$0 = \frac{d}{dt}\psi_{kq} + R_{kq} i_{kq}$$

Ecuaciones 1.36 Voltajes del rotor

La componente '*0*' de la transformada de Park, por la cual fluye la corriente de secuencia cero i_0, no tiene acoplamiento con los devanados del rotor, por lo tanto, es despreciable y ya no será tomada en cuenta para este estudio.

1.3.1.4. Transformada del torque eléctrico al sistema qd0

Aplicando la transformada de Park en la ecuación 1.17 se obtiene el torque eléctrico en coordenadas qd0:

$$T_e = -\frac{1}{2}\left[i_{dq0}[C_P]^T\left[\frac{\partial L_{ss}}{\partial\theta}\right][C_P]i_{dq0} + 2i_{dq0}^T[C_P]^T\left[\frac{\partial L_{sr}}{\partial\theta}\right]i_r\right]$$

Ecuación 1.37 Torque eléctrico en el sistema dq0

Los valores de las matrices L_{ss} y L_{sr} se indican en la ecuaciones 1.4 y 1.6 respectivamente, entonces al aplicar las derivadas parciales correspondientes y operar se obtiene:

$$T_e = i_q\left(\sqrt{\frac{3}{2}}L_{af}i_f + \sqrt{\frac{3}{2}}L_{akd}i_{kd} + \frac{3}{2}L_{aa2}i_d\right) - i_d\left(\sqrt{\frac{3}{2}}L_{ag}i_g + \sqrt{\frac{3}{2}}L_{akq}i_{kq} - \frac{3}{2}L_{aa2}i_q\right)$$

Ecuación 1.38 Torque eléctrico en el sistema dq0

Si al expandir la ecuación 1.25 se tiene que:

$$\psi_d = L_d i_d + \sqrt{\frac{3}{2}}L_{af}i_f + \sqrt{\frac{3}{2}}L_{akd}i_{kd}$$

$$\psi_q = L_q i_q + \sqrt{\frac{3}{2}}L_{ag}i_g + \sqrt{\frac{3}{2}}L_{akq}i_{kq}$$

Ecuaciones 1.39 Concatenaciones de flujo en el sistema dq0

Entonces para lograr una mejor expresión del torque eléctrico se remplaza las ecuaciones 1.39 en la ecuación 1.38 y se opera, considerando que $L_d - \frac{3}{2}L_{aa2} = L_q + \frac{3}{2}L_{aa2}$:

$$T_e = i_q\psi_d - i_d\psi_q$$

Ecuación 1.40 Torque eléctrico en el sistema dq0

1.3.1.5. Parámetros en por unidad [1]

Es conveniente expresar las magnitudes de voltaje, corriente e impedancia en por unidad escogiendo una base adecuada. Las ventajas de usar este sistema son:

- Los valores numéricos de corriente y voltaje están relacionados con sus valores nominales independientemente del tamaño de la máquina.
- Las impedancias en por unidad de una máquina tienen valores muy cercanos a impedancias de máquinas de diseño similar.
- Se requiere menor número de parámetros.

Los valores base de los parámetros del estator son:

- Potencia base, S_B = Potencia $3\emptyset$ nominal
- Voltaje base, V_B = Voltaje de línea a línea nominal (RMS)
- Corriente base, I_B = $\sqrt{3} \times$ Corriente de línea nominal
- Impedancia base, $Z_B = \frac{V_B}{I_B}$
- Velocidad angular base, ω_B = frecuencia angular nominal en rad/seg.
- Concatenaciones de flujo base, $\psi_B = \frac{V_B}{\omega_B}$
- Inductancia base, $L_B = \frac{\psi_B}{I_B} = \frac{Z_B}{\omega_B}$

En base a estas cantidades, se reescriben las ecuaciones 1.35 como:

$$v_d = -\frac{1}{\omega_B}\frac{d}{dt}\psi_d - \frac{\omega}{\omega_B}\psi_q - R_a i_d$$

$$v_q = -\frac{1}{\omega_B}\frac{d}{dt}\psi_q - \frac{\omega}{\omega_B}\psi_d - R_a i_q$$

Ecuaciones 1.41 Voltajes del estator en el sistema qd0 en por unidad

Donde $\omega = \dot{\theta}$, y $v_d, v_q, \psi_d, \psi_q, i_d, i_q, R_a$ representan ahora cantidades en por unidad.

Nótese que si la frecuencia de operación es la misma que la frecuencia base, entonces las inductancias en por unidad son idénticas a sus correspondientes reactancias en por unidad, por lo tanto se cumple que $x_{pu} = L_{pu}$, como consecuencia en adelante se empleará el símbolo de reactancia en lugar del de inductancia, ejemplo: $x_d = L_d$, $x_{akd} = L_{akd}$, etc.; donde x es la reactancia en por unidad.

Para determinar los valores base de los parámetros de rotor se debe considerar que la potencia base y la frecuencia base son las mismas que en el estator, entonces se obtiene:

$$I_{fB} = \frac{x_{ad}}{x_{df}} I_B$$

$$I_{kdB} = \frac{x_{ad}}{x_{dkd}} I_B$$

$$\psi_{fB} = \frac{I_B}{I_{fB}} \psi_B$$

$$\psi_{kdB} = \frac{I_B}{I_{kdB}} \psi_B$$

Ecuaciones 1.42 Valores base del rotor, eje-d

$$I_{gB} = \frac{x_{aq}}{x_{qg}} I_B$$

$$I_{kqB} = \frac{x_{aq}}{x_{qkq}} I_B$$

$$\psi_{gB} = \frac{I_B}{I_{gB}} \psi_B$$

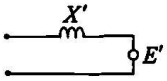
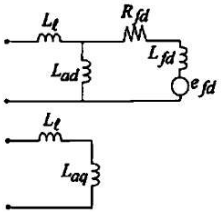
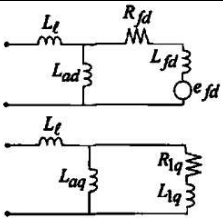
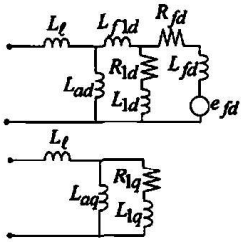
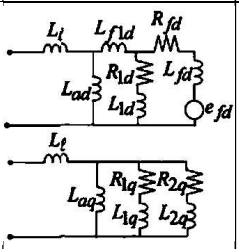
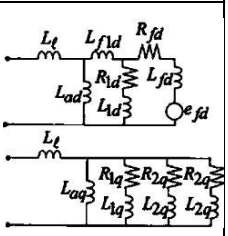
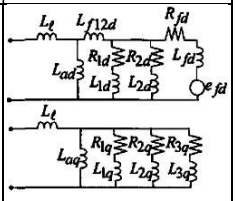
$$\psi_{kqB} = \frac{I_B}{I_{kqB}} \psi_B$$

Ecuaciones 1.43 Valores base del rotor, eje-q

1.3.2. Modelos de Generador Sincrónico Según el Tipo de Rotor [1] [2]

En [2] se sugiere varios modelos para los circuitos de los devanados del rotor dependiendo del grado de detalle y de la aplicación del generador, básicamente en la norma se detalla seis modelos los cuales se muestra en la Tabla 1.1. Se recomienda que para generadores de rotor cilíndrico se use el modelo 2.2 y para rotor de polos salientes se use el modelo 2.1

Tabla 1.1. Modelos del generador según grado de complejidad [2]

Concatena-ciones de flujo del rotor	Equivalente de Thevenin 			
Eje – Q → / Eje - D ↓	Sin circuitos de amortiguamiento equivalentes	Un circuito de amortiguamiento equivalente	Dos circuitos de amortiguamiento equivalentes	Tres circuitos de amortiguamiento equivalentes
Un solo circuito de campo	 Modelo 1.0	 Modelo 1.1	No Considerado	No considerado
Circuito de campo + Un circuito de amortigua-miento	No considerado	 Modelo 2.1	 Modelo 2.2	 Modelo 2.3
Circuito de campo + Dos circuitos de amortigua-miento	No considerado	No considerado	No considerado	 Modelo 3.3

1.3.2.1. Aplicación del modelo 2.2 del estándar IEEE 1001

A partir de la ecuación 1.39 se consigue:

$$\psi_d = x_d i_d + x_{df} i_f + x_{dh} i_h$$

$$\psi_q = x_q i_q + x_{qg} i_g + x_{qk} i_k$$

Ecuaciones 1.44 Flujos del estator en los ejes q-d

Donde: $x_{df} = \sqrt{\frac{3}{2}} L_{af}, x_{dh} = \sqrt{\frac{3}{2}} L_{akd}, x_{qg} = \sqrt{\frac{3}{2}} L_{ag}, x_{qk} = \sqrt{\frac{3}{2}} L_{akq}$

Para eliminar las corrientes del rotor i_f,i_h, i_g, i_k se las debe expresar en términos de ψ. Como se muestra a continuación:

$$\begin{bmatrix} i_f \\ i_h \end{bmatrix} = \begin{bmatrix} x_f & x_{fkd} \\ x_{fkd} & x_{kd} \end{bmatrix}^{-1} \left(\begin{bmatrix} \psi_f \\ \psi_{kd} \end{bmatrix} - \begin{bmatrix} x_{df} \\ x_{dkd} \end{bmatrix} i_d \right)$$

$$\begin{bmatrix} i_g \\ i_k \end{bmatrix} = \begin{bmatrix} x_g & x_{gkq} \\ x_{gkq} & x_{kq} \end{bmatrix}^{-1} \left(\begin{bmatrix} \psi_g \\ \psi_{kq} \end{bmatrix} - \begin{bmatrix} x_{qg} \\ x_{qkq} \end{bmatrix} i_q \right)$$

Ecuaciones 1.45 Corrientes del rotor

Reemplazando las ecuaciones 1.45 en 1.44 y operando da como resultado:

$$\psi_d = x_d'' i_d + E_q''$$

$$\psi_q = x_q'' i_q - E_d''$$

Ecuaciones 1.46 Flujos del estator

Donde:

$$E_q'' = \left(\frac{x_{df} x_{kd} - x_{dkd} x_{fkd}}{x_f x_{kd} - x_{fkd}^2} \right) \psi_f + \left(\frac{x_{dkd} x_f - x_{df} x_{fkd}}{x_f x_{kd} - x_{fkd}^2} \right) \psi_{kd}$$

$$E_d'' = -\left(\frac{x_{qg} x_{kq} - x_{qkq} x_{gkq}}{x_g x_{kq} - x_{gkq}^2} \right) \psi_g - \left(\frac{x_{qkq} x_g - x_{qg} x_{gkq}}{x_g x_{kq} - x_{gkq}^2} \right) \psi_{kq}$$

Ecuaciones 1.47 Valores de E_s''

Es conveniente expresar las ecuaciones en función de las reactancias transitoria y subtransitoria, para ello se emplea as siguientes ecuaciones:

$$x'_d = x_d - \frac{x_{ad}^2}{x_f}$$

$$x'_q = x_q - \frac{x_{aq}^2}{x_g}$$

$$x''_d = x_d - \left(\frac{x_{df}x_{kd} - x_{dkd}x_{fkd}}{x_f x_{kd} - x_{fkd}^2}\right)x_{df} - \left(\frac{x_{dkd}x_f - x_{df}x_{fkd}}{x_f x_{kd} - x_{fkd}^2}\right)x_{dkd}$$

$$x''_q = x_q - \left(\frac{x_{qg}x_{kq} - x_{qkq}x_{gkq}}{x_g x_{kq} - x_{gkq}^2}\right)x_{qg} - \left(\frac{x_{qkq}x_g - x_{qg}x_{gkq}}{x_g x_{kq} - x_{gkq}^2}\right)x_{qkq}$$

Ecuaciones 1.48 Valores de reactancia transitorias y subtransitorias

Reemplazando las ecuaciones 1.48 en 1.47 y operando se obtiene:

$$E''_q = \left(\frac{x_d - x'_d}{x_d}\right)\left(\frac{x''_d}{x'_d}\right)\psi_f + \left(\frac{x'_d - x''_d}{x'_d}\right)\psi_{kd}$$

$$E''_d = \left(\frac{x'_q - x_q}{x_q}\right)\left(\frac{x''_q}{x'_q}\right)\psi_g + \left(\frac{x''_q - x'_q}{x'_q}\right)\psi_{kq}$$

Ecuaciones 1.49 Valores de E'' del estator modelo 2.2

Las ecuaciones básicas de las concatenaciones de flujo del rotor en por unidad son:

$$\frac{d\psi_f}{dt} = \omega_B\left[-R_f i_f + \frac{x'_d}{x_d - x'_d}E_{fd}\right]$$

$$\frac{d\psi_{kd}}{dt} = -\omega_B R_{kd} i_{kd}$$

$$\frac{d\psi_g}{dt} = -\omega_B R_g i_g$$

$$\frac{d\psi_{kq}}{dt} = -\omega_B R_{kq} i_{kq}$$

Ecuaciones 1.50 Flujos del rotor

$$i_f = \frac{\psi_f - \psi_d}{x_{df}} \; ; \quad i_{kd} = \frac{\psi_{kd} - \psi_d}{x_{dkd}}$$
$$i_g = \frac{\psi_g - \psi_d}{x_{qg}} \; ; \quad i_{kq} = \frac{\psi_{kq} - \psi_d}{x_{qkq}}$$

Ecuaciones 1.51 Corrientes del rotor en función del flujo

Las constantes de tiempo T pueden expresarse mediante las siguientes ecuaciones:

$$T_d' = \frac{x_{df}}{\omega_B R_f} \; ; \quad T_d'' = \frac{x_{dkd}}{\omega_B R_{kd}}$$
$$T_q' = \frac{x_{qg}}{\omega_B R_g} \; ; \quad T_d'' = \frac{x_{qkq}}{\omega_B R_{kq}}$$

Ecuaciones 1.52 Constantes de tiempo transitorio y subtransitorio

Se reemplaza las ecuaciones 1.51 y 1.52 en las ecuaciones 1.50 para conseguir las siguientes expresiones para flujo del rotor:

$$\frac{d\psi_f}{dt} = \frac{1}{T_d'}\left(\psi_d - \psi_f + \frac{x_d'}{x_d - x_d'} E_{fd}\right)$$

$$\frac{d\psi_{kd}}{dt} = \frac{1}{T_d''}(\psi_d - \psi_{kd})$$

$$\frac{d\psi_g}{dt} = \frac{1}{T_q'}(\psi_q - \psi_g)$$

$$\frac{d\psi_{kq}}{dt} = \frac{1}{T_q''}(\psi_q - \psi_{kq})$$

Ecuaciones 1.53 Flujos del rotor modelo 2.2

1.3.2.2. Aplicación del modelo 2.1 del estándar IEEE 1001

Como se aprecia en la tabla 1.1 el modelo 2.1 consta de solo dos devanados de amortiguamiento, uno en cada eje, entonces las ecuaciones son similares a la del modelo 2.2 con la diferencia que en este no existe el flujo ψ_{kq} ni la reactancia x'_q.

$$E_q'' = \left(\frac{x_d - x_d'}{x_d}\right)\left(\frac{x_d''}{x_d'}\right)\psi_f + \left(\frac{x_d' - x_d''}{x_d'}\right)\psi_{kd}$$
$$E_d'' = \left(\frac{x_q'' - x_q}{x_q}\right)\psi_g$$

Ecuaciones 1.54 Valores de E'' del estator modelo 2.1

$$\frac{d\psi_f}{dt} = \frac{1}{T_d'}\left(\psi_d - \psi_f + \frac{x_d'}{x_d - x_d'}E_{fd}\right)$$
$$\frac{d\psi_{kd}}{dt} = \frac{1}{T_d''}(\psi_d - \psi_{kd})$$
$$\frac{d\psi_g}{dt} = \frac{1}{T_q''}(\psi_q - \psi_g)$$

Ecuaciones 1.55 Flujos del rotor modelo 2.1

1.3.3. Sistema de Excitación y Regulador de Voltaje – AVR [3] [4] [5]

El objetivo de los sistemas de excitación (AVR) es ajustar automáticamente la corriente de campo del generador sincrónico cuanto exista perturbaciones tanto transitorias como estacionarias, con el fin de mantener estable el voltaje en los terminales del generador. Los componentes básicos del sistema son: amplificador, excitatriz, estabilizador y comparador. La figura 1.2 representa la configuración básica del sistema AVR

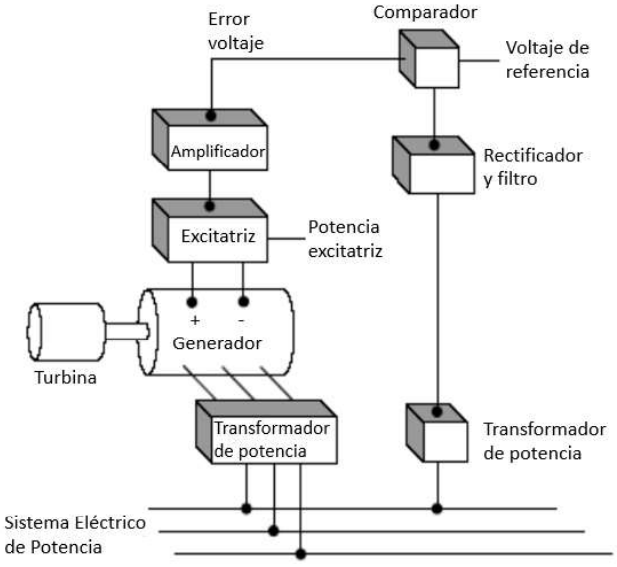


Figura 1.2 Modelo real del sistema AVR [3]

Los sistemas de excitación se clasifican de acuerdo al tipo de fuente primaria: dc, ac y estático, en muchos de los sistemas de excitación dc la energía proviene de un generador de corriente continua cuyos devanados de campo están montados en el mismo eje del rotor del generador sincrónico, en el presente trabajo el estudio del avr se enfoca en el modelamiento de un sistema de excitación dc. La figura 1.3 representa el esquema básico de conexión del sistema de control de excitación.

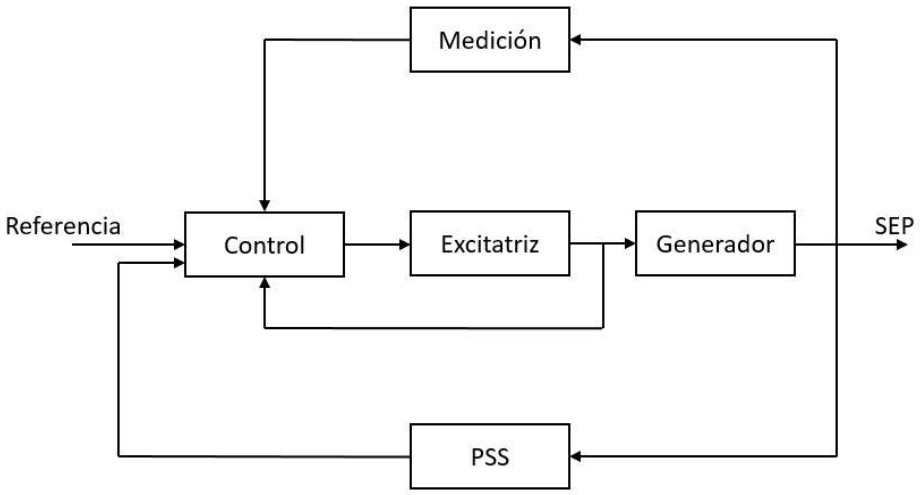


Figura 1.3 Diagrama de bloques funcional del control de excitación [3].

A continuación, se detalla la función de los bloques básicos del AVR:

Comparador. – También se lo conoce como transductor y su función es compensar la caída de voltaje que se produce en la medición por la impedancia de los transformadores de corriente y potencia (TC y TP)

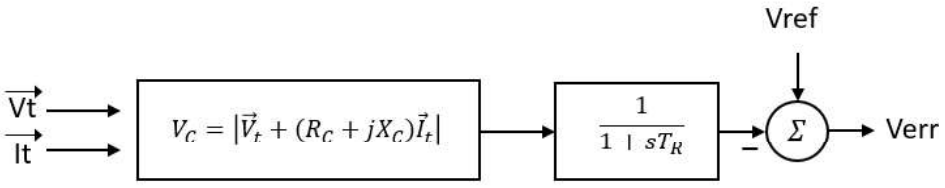


Figura 1.4 Función de transferencia del compensador de voltaje [1]

Amplificador. – Incrementa la señal de entrada a un nivel adecuado, de tal manera que actúa como respaldo de la excitatriz, además esta unidad incluye límites para el máximo y mínimo voltaje de campo.

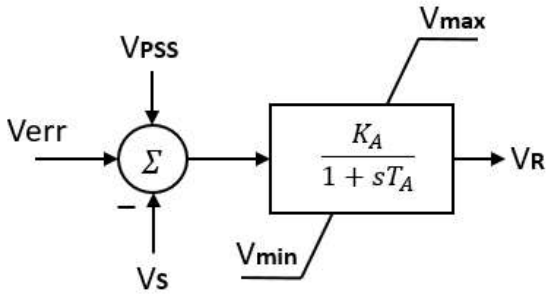


Figura 1.5 Función de transferencia del amplificador [1]

Estabilizador. – Su objetivo es proveer del adelanto de fase necesario para lograr el correcto margen de ganancia y fase en la respuesta de frecuencia de lazo abierto de la excitatriz, para ello actúa de dos maneras; una es habilitar una alta ganancia del regulador mediante la compensación de las grandes constantes de tiempo de la excitatriz, la otra función es contrarrestar el amortiguamiento negativo que se introduce por la respuesta inicial del sistema de excitación.

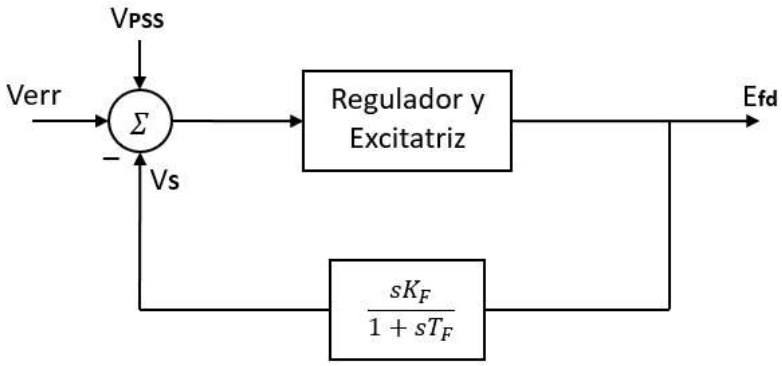


Figura 1.6 Función de transferencia del estabilizador (V_F) [1]

Excitatriz. – Constituye la fuente de poder del regulador ya que esta unidad proporciona la energía eléctrica que alimenta el campo rotatorio de la máquina sincrónica. El devanado de armadura por lo general tiene un pequeño número de vueltas a comparación del devanado de campo, como consecuencia la pequeña impedancia del devanado de armadura a menudo se desprecia. El voltaje de salida de la excitatriz es una función no

lineal debido a las corrientes de campo y armadura. En la ecuación 1.56, se expresa la corriente de excitación en función del voltaje del devanado de armadura (v_x).

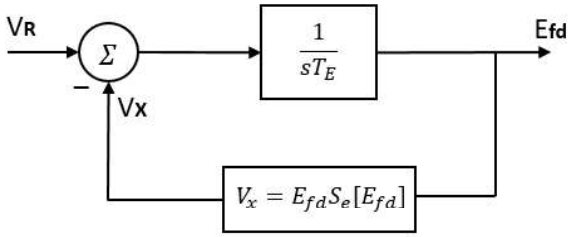


Figura 1.7 Función de transferencia de la excitatriz [1]

$$i_{fd} = \frac{v_x}{R_{ag}} + S_e v_x$$

Ecuación 1.56 Corriente de campo de la excitatriz

Donde:

$$S_e = A_{ex} e^{B_{ex} E_{fd}}$$

Ecuación 1.57 Función de saturación de la excitatriz.

Entonces combinando todos los componentes descritos y considerando que las mediciones de voltaje y corriente son ideales (Rc=0, Xc=0), se obtiene el regulador DC1A propuesto en la norma IEEE - 421.5 [6]:

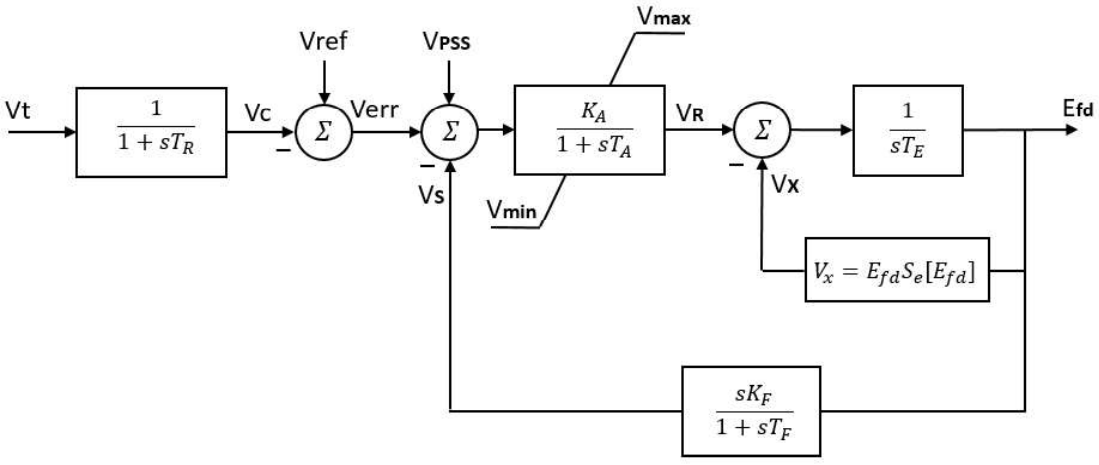


Figura 1.8 AVR Tipo DC1A [6]

1.3.4. Sistema de Control de Velocidad – GOV

Para la regulación de la frecuencia en un sistema eléctrico es necesario el control de velocidad de fuerza motriz, para ello se emplea un "gobernador" el cual es un sistema de control auxiliar que responde a eventos de desbalance de carga y que además sirve para ajustar la generación de acuerdo al despacho económico [1]. Los diferentes tipos de control de fuerza motriz se clasifican en:

- Primario. – Regulador de velocidad (gobernador)
- Secundario. - Regulación de frecuencia – potencia (LFC)
- Terciario. - Despacho económico

Con el incremento del tamaño de los sistemas de potencia debido a las interconexiones, las variaciones de frecuencia (en condiciones normales) se han vuelto cada vez menos frecuentes, sin embargo, el rol de los reguladores de velocidad en el control rápido de frecuencia no puede ser descartado [1].

En estudios de estabilidad, generalmente no se considera los controles secundario y terciario, solo se necesita representar los reguladores de velocidad, incluyendo el modelo de turbina [1]. En esta sección se presenta la turbina y el gobernador basados en la norma IEEE Hydro Turbines Committee Report [7].

1.3.4.1. Turbina hidráulica

La turbina hidráulica se representa mediante la función de transferencia de la figura 1.9 donde T_W se conoce como constante de tiempo de arranque del agua y su valor se muestra en la ecuación 1.58.

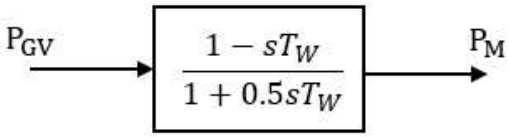


Figura 1.9 Función de transferencia de la turbina hidráulica [1]

$$T_W = \frac{LV}{H_T g}$$

Ecuación 1.58 Constante de tiempo del agua

Donde:

L Longitud de la tubería de presión

V Velocidad del agua

H_T Caída total

g Aceleración debido a la gravedad

Algunos modelos de turbinas hidráulicas, con el fin de lograr mayor precisión, incluyen los fenómenos que sufre el agua durante su trayectoria por la tubería de presión, sin embargo, eso no es necesario para estudios de estabilidad [1].

Los valores de T_W oscilan en el rango de 0.5 a 5.0 segundos, pero el valor típico que se usa es de 1.0 segundo [1]. La entrada $\mathrm{P_{GV}}$ proviene del controlador de velocidad (gobernador).

1.3.4.2. Control de velocidad (gobernador)

Existen dos tipos de controles de velocidad: Mecánico-Hidráulico y Electro-Hidráulico. En ambos casos se emplea motores hidráulicos para posicionar la válvula o compuerta que controla el flujo de agua [1].

El gobernador electro-hidráulico tiene básicamente el mismo funcionamiento dinámico que el mecánico-hidráulico, por lo tanto, en estudios de estabilidad se emplea una sola función de estado simplificada (figura 1.10) que representa ambos tipos de controles [1].

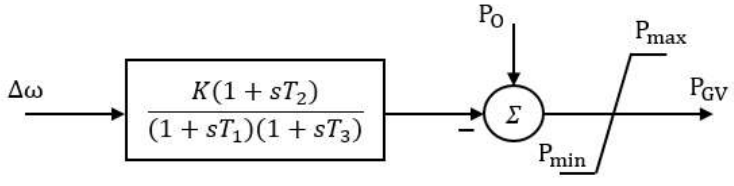


Figura 1.10 Función de estado del control de velocidad [1]

Donde:

P_O	Potencia inicial o potencia de referencia
K	Ganancia del control de velocidad
T_1	Constante de tiempo de retardo
T_2	Constante de tiempo de adelanto
T_3	Constante de tiempo del actuador de la compuerta
P_{max}, P_{min}	Límites de la compuerta
$\Delta\omega$	Diferencia entre la velocidad de referencia y la velocidad de operación

Para resolver la función de transferencia de la figura 1.10, es útil separarla en dos bloques, entonces combinando el gobernador con la turbina hidráulica se obtiene el modelo general simplificado de un GOV, el cual según la norma IEEE Hydro Turbines Committee Report [7] se lo denomina Type IEEEG2.

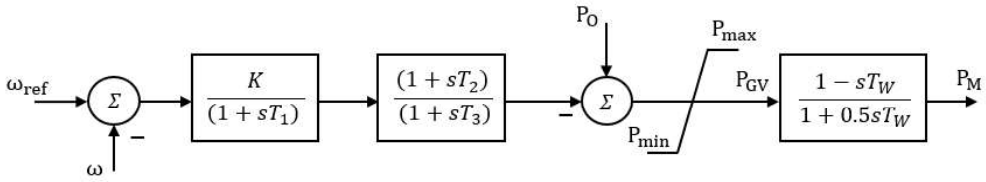


Figura 1.11 GOV – IEEEG2 [7]

1.3.5. Método Runge-Kutta de Cuarto Orden para Resolver Sistemas de Ecuaciones Diferenciales [8] [9]

De la familia de métodos Runge-Kutta, el de cuarto orden es el más usado debido a la gran precisión y velocidad del algoritmo para encontrar la solución de una ecuación diferencial (ecuación 1.59) conociendo el valor inicial; usualmente se abrevia como RK4. Básicamente se trata de un método numérico que conociendo y_n en el punto inicial t_n, da la solución aproximada de la función y_{n+1} en el punto $t_n + h$. Donde h es un punto consecutivo de la función y, por lo tanto h puede ser considerado como la precisión del método ya que mientras menor sea este valor más exacta será la solución. La fórmula que se emplea para resolver una sola ecuación diferencial se muestra en la ecuación 1.60.

$$\frac{dy}{dx} = f(x, y)$$

Ecuación 1.59 Ecuación Diferencia Ordinaria

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Ecuación 1.60 Valor de la variable dependiente método RK4

Donde las constantes k están en función de la variable dependiente y/o independiente, y se las conoce como etapas del método RK4.

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

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

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

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

Ecuaciones 1.61 Etapas del método RK4

Si se considera ahora un sistema de p ecuaciones diferenciales de primer orden, entonces existirá p variables que darán la solución al sistema. En las ecuaciones del presente trabajo la variable independiente es el tiempo t. A las variables de estado se las representa con la letra x.

$$\begin{aligned}
\dot{x}_1 &= \frac{d}{dt}x_1(t) = f(t, x_1, x_2, \dots, x_p) \\
\dot{x}_2 &= \frac{d}{dt}x_2(t) = f(t, x_1, x_2, \dots, x_p) \\
&\vdots \\
\dot{x}_p &= \frac{d}{dt}x_p(t) = f(t, x_1, x_2, \dots, x_p)
\end{aligned}$$

Ecuaciones 1.62 Sistema de Ecuaciones Diferenciales Ordinarias

$$x_{1_{n+1}} = x_{1_n} + \frac{1}{6}[K_1(1) + 2K_2(1) + 2K_3(1) + K_4(1)]$$
$$x_{2_{n+1}} = x_{2_n} + \frac{1}{6}[K_1(2) + 2K_2(2) + 2K_3(2) + K_4(2)]$$
$$\vdots$$
$$x_{p_{n+1}} = x_{p_n} + \frac{1}{6}[K_1(p) + 2K_2(p) + 2K_3(p) + K_4(p)]$$

Ecuaciones 1.63 Variable dependiente método RK4 para un sistema EDO

Donde $K_{1,2,3,4}$ pasan a ser vectores de p términos como se muestra en las siguientes ecuaciones:

$$K_1 = h\begin{bmatrix} f_1(t_n, x_{1_n}, x_{2_n}, \dots, x_{p_n}) \\ f_2(t_n, x_{1_n}, x_{2_n}, \dots, x_{p_n}) \\ \vdots \\ f_p(t_n, x_{1_n}, x_{2_n}, \dots, x_{p_n}) \end{bmatrix}$$

Ecuación 1.64 Componente K_1

$$K_2 = h\begin{bmatrix} f_1\left(t_n + \frac{h}{2}, x_{1_n} + \frac{K_1(1)}{2}, x_{2_n} + \frac{K_1(2)}{2}, \dots, x_{p_n} + \frac{K_1(p)}{2}\right) \\ f_2\left(t_n + \frac{h}{2}, x_{1_n} + \frac{K_1(1)}{2}, x_{2_n} + \frac{K_1(2)}{2}, \dots, x_{p_n} + \frac{K_1(p)}{2}\right) \\ \vdots \\ f_p\left(t_n + \frac{h}{2}, x_{1_n} + \frac{K_1(1)}{2}, x_{2_n} + \frac{K_1(2)}{2}, \dots, x_{p_n} + \frac{K_1(p)}{2}\right) \end{bmatrix}$$

Ecuación 1.65 Componente K_2

$$K_3 = h\begin{bmatrix} f_1\left(t_n + \frac{h}{2}, x_{1_n} + \frac{K_2(1)}{2}, x_{2_n} + \frac{K_2(2)}{2}, \dots, x_{p_n} + \frac{K_2(p)}{2}\right) \\ f_2\left(t_n + \frac{h}{2}, x_{1_n} + \frac{K_2(1)}{2}, x_{2_n} + \frac{K_2(2)}{2}, \dots, x_{p_n} + \frac{K_2(p)}{2}\right) \\ \vdots \\ f_p\left(t_n + \frac{h}{2}, x_{1_n} + \frac{K_2(1)}{2}, x_{2_n} + \frac{K_2(2)}{2}, \dots, x_{p_n} + \frac{K_2(p)}{2}\right) \end{bmatrix}$$

Ecuación 1.66 Componente K_3

$$K_4 = h\begin{bmatrix} f_1\left(t_n + h, x_{1_n} + K_3(1), x_{2_n} + K_3(2), \dots, x_{p_n} + K_3(p)\right) \\ f_2\left(t_n + h, x_{1_n} + K_3(1), x_{2_n} + K_3(2), \dots, x_{p_n} + K_3(p)\right) \\ \vdots \\ f_p\left(t_n + h, x_{1_n} + K_3(1), x_{2_n} + K_3(2), \dots, x_{p_n} + K_3(p)\right) \end{bmatrix}$$

Ecuación 1.67 Componente K_4

Si se diera el caso en que una o más ecuaciones diferenciales sean de un orden superior entonces lo que se debe hacer es convertir la derivada de orden n a n derivadas de primer orden, ejemplo:

$$\ddot{x} = \frac{d^2x}{dt^2} = f(t, x)$$

Ecuación 1.68 Derivada de segundo orden

Se hace el siguiente cambio de variables:

$$x_1 = x = f(t, x)$$

$$x_2 = \frac{dx}{dt}$$

Ecuaciones 1.69 Cambio de variables

Entonces:

$$\dot{x}_1 = \frac{dx}{dt} = x_2$$

$$\dot{x}_2 = \frac{d^2x}{dt^2} = \ddot{x}$$

Ecuaciones 1.70 Derivadas de primer orden

1.3.6. Introducción a FORTRAN

El lenguaje de programación Fortran fue el primer lenguaje de alto nivel, se creó en la década de los 50 por John Backus para la IBM 704, el nombre viene de la unión de dos palabras: **For**mula **Tran**slation. En el año de 1966 este lenguaje fue estandarizado con el nombre de FORTRAN IV, con el transcurso de los años se han hacho revisiones del lenguaje dando como resultado FORTRAN 66, 77, 90/95, 2003, 2008 y 2018. Fortran fue diseñado por científicos e ingenieros y ha dominado este campo, por más de 50 años Fortran ha sido usado en numerosos proyectos como diseños de estructuras y aeronaves, se usa en el control automático en fábricas, control de drenaje pluvial, análisis de datos científicos, por citar algunos ejemplos [10] [11] [13].

Los elementos y comandos básicos para programar en Fortran se detallan en el ANEXO I

1.3.6.1. Ventajas

- Dispone de un gran conjunto de paquetes con las mejores bibliotecas numéricas como: BLAS, PAW o SOFA [11]
- Es un lenguaje compacto y ligero, es decir que es fácil de aprender, es portable y de sencilla depuración.
- Tiene buen soporte para arreglos y matrices.
- Fácil de paralelizar en memoria compartida, razón por la cual la suite de Intel Parallel Studio incluye herramientas de desarrollo en Fortran [12]

1.3.6.2. Desventajas

- La comunidad de usuarios es pequeña debido a que solo se emplea en el campo científico, por lo tanto, no hay muchas clases y/o libros sobre Fortran.
- Está lejos de los enfoques modernos de programación, de hecho, Fortran puede ser considerado arcaico por aquellos programadores que usan los lenguajes más actuales.

2. METODOLOGÍA

En este capítulo se implementa en Fortran las ecuaciones de la máquina sincrónica, así como los correspondientes reguladores. Se realiza la modelación del sistema generador-barra infinita, las ecuaciones a programar corresponden al modelo 2.1 de la IEEE (sección 1.3.2.2). Adicionalmente se incluye el proceso de inicialización de variables de estado, tema de capital importancia para la implementación del algoritmo Runge-Kutta, y se da una rápida guía sobre la correcta instalación del software utilizado, así como una revisión al modelo desarrollado en DigSilent, el cual sirve para validar el proyecto.

2.1. Arreglo de Ecuaciones del Sistema Máquina Sincrónica – Barra Infinita [21]

Para analizar la estabilidad transitoria se ilustra un sistema simple que consiste en una Máquina Sincrónica (generador) conectado a una Barra Infinita, abreviado como MSBI. A pesar que este sistema es un ejemplo sencillo no significa que no exista en la realidad; puede darse por ejemplo que una unidad de generación remota se conecte a un centro de carga a través de una línea de transmisión larga, en ese caso se tendría el sistema MSBI.

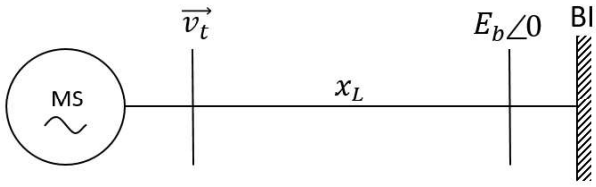


Figura 2.1. Sistema MSBI

Del flujo de potencia de la figura 2.1 se obtiene:

$$\overrightarrow{v_t} = jx_L\vec{I} + E_b\angle 0$$

Ecuación 2.1. Voltaje en terminales sistema MSBI

Escribiendo la ecuación 2.1 referida al frame (sistema dq0) de la máquina se tiene:

$$v_q + jv_d = jx_L\left(i_q + ji_d\right) + E_b[\cos(-\delta) + j\sin(-\delta)]$$

Ecuación 2.2. Voltaje en terminales referido al frame de la máquina

Donde δ es el ángulo entre el voltaje en terminales y el voltaje interno de la máquina. Se lo conoce también como ángulo de potencia.

Separando la parte real e imaginaria de la ecuación 2.2 resulta:

$$v_q = E_b \cos\delta - i_d x_L$$

$$v_d = -E_b \operatorname{sen}\delta + i_q x_L$$

Ecuaciones 2.3. Parte real e imaginaria de $\overrightarrow{v_t}$

Si se ignoran los términos $d\psi_d/dt$ y $d\psi_q/dt$ de la ecuación 1.41 por ser valores muy cercanos a cero, y reemplazando la ecuación 1.46 en 1.41, considerando que no existe variaciones de velocidad ($\omega = \omega_B$) se consigue la siguiente expresión:

$$v_d = E_d'' - x_q'' i_q - R_a i_d$$
$$v_q = E_q'' + x_d'' i_d - R_a i_q$$

Ecuaciones 2.4. Voltajes del estator

Al escribir las ecuaciones 2.4 en forma de matriz da como resultado:

$$\begin{bmatrix} R_a & x_q'' \\ -x_d'' & R_a \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} E_d'' - v_d \\ E_q'' - v_q \end{bmatrix}$$

Ecuación 2.5 Voltajes del estator

Se reemplaza las ecuaciones 2.3 en 2.5 y se opera para despejar las corrientes del estator:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} R_a & (x_q'' + x_L) \\ -(x_d'' + x_L) & R_a \end{bmatrix}^{-1} \begin{bmatrix} E_d'' + E_b \operatorname{sen}\delta \\ E_q'' - E_b \cos\delta \end{bmatrix}$$

Ecuación 2.6 Corrientes del estator

Puesto que, en por unidad, la constante de inercia es: $H = \frac{1}{2}J$, y $\omega = \frac{d\theta}{dt}$, se puede escribir la ecuación 1.14 como:

$$T_m - T_e = 2H\frac{d\omega}{dt} + D\omega$$

Ecuación 2.7. Fuerza motriz del rotor

Despejando la deriva en la ecuación 2.7:

$$\frac{d\omega}{dt} = \frac{1}{2H}(T_m - T_e - D\omega)$$

Ecuación 2.8. Derivada de la velocidad angular del rotor

Por definición el ángulo de potencia δ es igual a la diferencia entre el ángulo del rotor (θ) y el ángulo del eje de referencia de rotación sincrónica θ_0 [5]:

$$\delta = \theta - \theta_0$$

Ecuación 2.9. Angulo de potencia

Para tener δ en función de la velocidad angular del rotor se deriva la ecuación 2.9:

$$\frac{d\delta}{dt} = \frac{d\theta}{dt} - \frac{d\theta_0}{dt} = \omega - \omega_0$$

Ecuación 2.10. Derivada de δ

Se debe tomar en cuenta que en el programa los valores de ω están en por unidad, por lo tanto, para calcular δ en radianes se debe multiplicar la ecuación 2.10 por la velocidad angular base ω_B, además si se considera que la velocidad de operación es la misma que la velocidad de referencia entonces $\omega_0 = 1$ (en pu):

$$\frac{d\delta}{dt} = (\omega - \omega_0)\omega_B = (\omega - 1)\omega_B$$

Ecuación 2.11. Derivada de δ

En conclusión, basándose en las ecuaciones dadas en la sección 1.3.2 se resume, en la tabla 2.1, las ecuaciones necesarias para simular la máquina sincrónica dependiendo del tipo de rotor.

Tabla 2.1. Ecuaciones de la máquina sincrónica según el tipo de rotor [1]

Tipo de rotor	**Polos salientes** Modelo 2.1	**Rotor Cilíndrico** Modelo 2.2
Sistema EDO	$\begin{cases} \frac{d\psi_f}{dt} = \frac{1}{T'_d}\left(\psi_d - \psi_f + \frac{x'_d}{x_d - x'_d}E_{fd}\right) \\ \frac{d\psi_{kd}}{dt} = \frac{1}{T''_d}(\psi_d - \psi_{kd}) \\ \frac{d\psi_g}{dt} = \frac{1}{T''_q}(\psi_q - \psi_g) \\ \frac{d\omega}{dt} = \frac{1}{2H}(T_m - T_e - D\omega) \\ \frac{d\delta}{dt} = (\omega - 1)\omega_B \end{cases}$	$\begin{cases} \frac{d\psi_f}{dt} = \frac{1}{T'_d}\left(\psi_d - \psi_f + \frac{x'_d}{x_d - x'_d}E_{fd}\right) \\ \frac{d\psi_{kd}}{dt} = \frac{1}{T''_d}(\psi_d - \psi_{kd}) \\ \frac{d\psi_g}{dt} = \frac{1}{T'_q}(\psi_q - \psi_g) \\ \frac{d\psi_{kq}}{dt} = \frac{1}{T''_q}(\psi_q - \psi_{kq}) \\ \frac{d\omega}{dt} = \frac{1}{2H}(T_m - T_e - D\omega) \\ \frac{d\delta}{dt} = (\omega - 1)\omega_B \end{cases}$
E''_{dq}	$E''_q = \left(\frac{x_d - x'_d}{x_d}\right)\left(\frac{x''_d}{x'_d}\right)\psi_f + \left(\frac{x'_d - x''_d}{x'_d}\right)\psi_{kd}$ $E''_d = \left(\frac{x''_q - x_q}{x_q}\right)\psi_g$	$E''_q = \left(\frac{x_d - x'_d}{x_d}\right)\left(\frac{x''_d}{x'_d}\right)\psi_f + \left(\frac{x'_d - x''_d}{x'_d}\right)\psi_{kd}$ $E''_d = \left(\frac{x'_q - x_q}{x_q}\right)\left(\frac{x''_q}{x'_q}\right)\psi_g + \left(\frac{x''_q - x'_q}{x'_q}\right)\psi_{kq}$
ψ_{dq}	$\psi_d = x''_d i_d + E''_q$ $\psi_q = x''_q i_q - E''_d$	
i_{dq}	$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} R_a & (x''_q + x_L) \\ -(x''_d + x_L) & R_a \end{bmatrix}^{-1} \begin{bmatrix} E''_d + E_b \operatorname{sen}\delta \\ E''_q - E_b \cos\delta \end{bmatrix}$	

Para cualquier tipo de modelo siempre debe cumplirse que: $(\mathrm{x_d} \geq \mathrm{x_q} > \mathrm{x'_q} \geq \mathrm{x'_d} > \mathrm{x''_q} \geq \mathrm{x''_d})$, $(\mathrm{T'_{d0}} > \mathrm{T'_d} > \mathrm{T''_{d0}} > \mathrm{T''_d})$ y $(\mathrm{T'_{q0}} > \mathrm{T'_q} > \mathrm{T''_{q0}} > \mathrm{T''_q})$ [5]

A continuación, la tabla 2.2 indica los valores límites de los parámetros para ambos modelos de rotor:

Tabla 2.2. Rangos típicos de los parámetros de la máquina sincrónica [5]

Parámetro		Modelo 2.1	Modelo 2.2
Reactancia sincrónica (p.u.)	x_d	0,6 – 1,5	1,0 – 2,3
	x_q	0,4 – 1,0	1,0 – 2,3
Reactancia transitoria (p.u.)	x'_d	0,2 – 0,5	0,15 – 0,4
	x'_q	–	0,3 – 1,0
Reactancia subtransitoria (p.u.)	x''_d	0,15 – 0,35	0,12 – 0,25
	x''_q	0,2 – 0,45	0,12 – 0,25
Constate de tiempo transitoria de Circ. Abierto. (seg.)	T'_{d0}	1,5 – 9,0	3,0 – 10,0
	T'_{q0}	–	0,5 – 2,0
Constate de tiempo subtransitoria de Circ. Abierto. (seg.)	T''_{d0}	0,01 – 0,05	0,02 – 0,05
	T''_{q0}	0,01 – 0,09	0,02 – 0,05
Resistencia del estator (p.u.)	R_a	0,002 – 0,02	0,0015 – 0,005

2.2. Arreglo de Ecuaciones del Regulador de Voltaje – AVR

Para simular el regulador IEEE DC1A, visto en la sección 1.3.3, se debe convertir las funciones de transferencia a ecuaciones en el dominio del tiempo, para ello se emplea la transformada inversa de Laplace de la ecuación 2.12.

$$\mathcal{L}^{-1}\left[\frac{1}{s}F(s)\right] = \int_0^t f(t)dt$$

Ecuación 2.12. Transformada inversa de Laplace

Entonces según el diagrama de la figura 1.8 (sección 1.3.3) se tiene 4 funciones de trasferencia que resultan en las ecuaciones diferenciales de ese regulador en particular, dichas funciones se identifican en la figura 2.2:

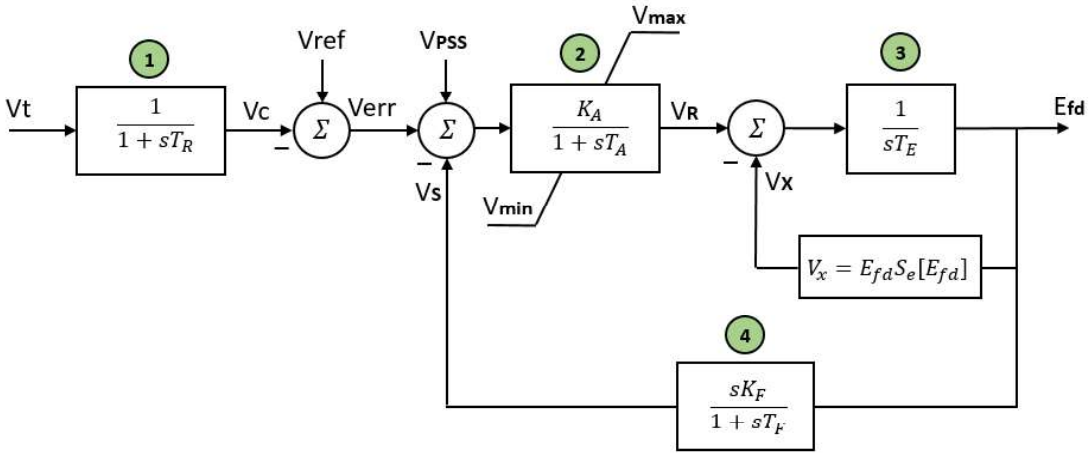


Figura 2.2. Funciones de transferencia del AVR IEEE DC1A

A continuación se realizan las operaciones necesarias para que cada función de transferencia tenga el factor $1/s$. No se incluye el parámetro V_{PSS} ya que en este trabajo no se simulará el estabilizador de potencia.

$$V_C = \frac{V_t}{1+sT_R} = \frac{1}{sT_R}(V_t - V_C)$$

$$V_R = (V_{ref} - V_C - V_S)\left(\frac{K_A}{1+sT_A}\right) = \frac{1}{sT_A}\left[(V_{ref} - V_C - V_S)K_A - V_R\right]$$

$$E_{fd} = \frac{V_R - V_x}{sT_E} = \frac{1}{sT_E}(V_R - V_x)$$

$$V_S = \frac{E_{fd}sK_F}{1+sT_F} = E_{fd}\frac{K_F}{T_F} - \frac{V_S}{sT_F}$$

Ecuaciones 2.13. Funciones de transferencia

Luego se aplica la transforma inversa (ecuación 2.12) a cada función de transferencia:

$$V_C = \frac{1}{T_R}\int_0^t (V_t - V_C)\,dt$$

$$V_R = \frac{1}{T_A}\int_0^t \left[(V_{ref} - V_C - V_S)K_A - V_R\right]dt$$

$$E_{fd} = \frac{1}{T_E}\int_0^t (V_R - V_x)\,dt$$

$$V_S = E_{fd}\frac{K_F}{T_F} - \frac{1}{T_F}\int_0^t V_S\,dt$$

Ecuaciones 2.14. Inversa de la Laplace de las ecuaciones 2.13

Finalmente se derivan las ecuaciones 2.14 para poder resolverlas mediante Runge-Kutta, con lo que se consigue el siguiente sistema de ecuaciones diferenciales:

$$\begin{cases} \frac{dV_C}{dt} = \frac{1}{T_R}(V_t - V_C) \\ \frac{dV_R}{dt} = \frac{1}{T_A}\left[(V_{ref} - V_C - V_S)K_A - V_R\right] \\ \frac{dE_{fd}}{dt} = \frac{1}{T_E}(V_R - V_x) \\ \frac{dV_S}{dt} = \frac{dE_{fd}}{dt}\frac{K_F}{T_F} - \frac{1}{T_F}V_S = \left[\frac{1}{T_E}(V_R - V_x)\right]\frac{K_F}{T_F} - \frac{1}{T_F}V_S \end{cases}$$

Ecuaciones 2.15. Sistema EDO del AVR IEEE DC1A

La saturación del bloque 2 de la figura 2.2 se programa con el siguiente algoritmo:

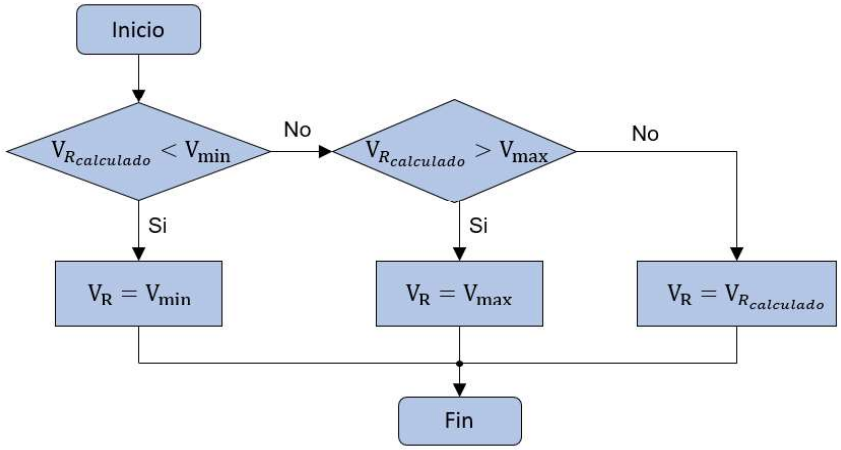


Figura 2.3. Algoritmo para la saturación del AVR

Los rangos en que pueden variar los parámetros del regulador se detalla en la tabla 2.3

Tabla 2.3. Valores límite de los parámetros del AVR

Retraso de medición	T_R	0 – 0,5
Ganancia del regulador	K_A	10 – 500
Constante de tiempo ganancia	T_A	0 – 1
Constante de tiempo excitatriz	T_E	0,04 – 1
Ganancia estabilizador	K_F	0 – 0,3
Conste de tiempo estabilizador	T_F	0,04 – 1,5
Control de voltaje mínimo	V_{min}	-10 – 0
Control de voltaje máximo	V_{max}	0,5 – 10

2.3. Arreglo de Ecuaciones del Regulador de Velocidad – GOV

El control de velocidad y turbina hidráulica que se modela es el IEEE G2 (sección 1.3.4). Sus funciones de transferencia se indican en la figura 2.4:

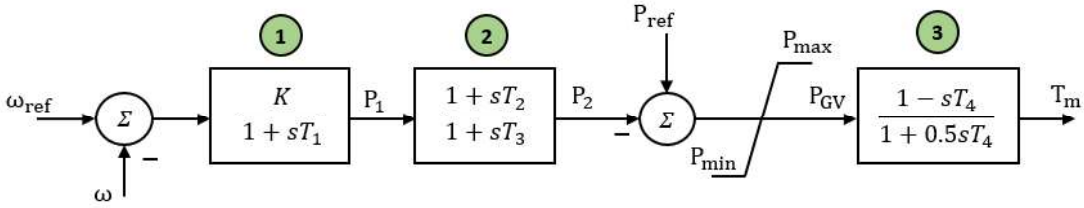


Figura 2.4. Funciones de transferencia GOV IEEEG2

Al igual que con el avr, se realizan las operaciones necesarias para que cada función de transferencia quede en función del factor $1/s$:

$$P_1 = \frac{(\omega_{ref} - \omega)K}{1 + sT_1} = \frac{1}{sT_1}\left[(\omega_{ref} - \omega)K - P_1\right]$$

$$P_2 = \frac{P_1(1 + sT_2)}{1 + sT_3} = \frac{1}{T_3}\left[\frac{1}{s}(P_1 - P_2) + T_2P_1\right]$$

$$T_m = \frac{(P_{ref} - P_2)(1 - sT_4)}{1 + 0.5T_4} = 2\left[\frac{1}{s}\left(\frac{P_{ref} - P_2 - T_m}{T_4}\right) - P_{ref} + P_2\right]$$

Ecuaciones 2.16. Funciones de transferencia

Aplicando la transformada inversa de Laplace a las ecuaciones 2.16 se tiene:

$$P_1 = \frac{1}{T_1}\int_0^t \left[(\omega_{ref} - \omega)K - P_1\right] dt$$

$$P_2 = \frac{1}{T_3}\left[\int_0^t (P_1 - P_2)\, dt + T_2P_1\right]$$

$$T_m = 2\left[\int_0^t \left(\frac{P_{ref} - P_2 - T_m}{T_4}\right) dt - P_{ref} + P_2\right]$$

Ecuaciones 2.17. Inversa de la Laplace de las ecuaciones 2.16

El sistema de ecuaciones diferenciales se consigue derivando las ecuaciones 2.17:

$$\begin{cases} \frac{dP_1}{dt} = \frac{1}{T_1}\left[(\omega_{ref} - \omega)K - P_1\right] \\ \frac{dP_2}{dt} = \frac{1}{T_3}\left[(P_1 - P_2) + T_2\frac{dP_1}{dt}\right] \\ \frac{dT_m}{dt} = 2\left[\left(\frac{P_{ref} - P_2 - T_m}{T_4}\right) - \frac{dP_{ref}}{dt} + \frac{dP_2}{dt}\right] \end{cases}$$

Ecuaciones 2.18. Sistema EDO del GOV IEEE G2

La saturación (Pmax/Pmin) se programa con el siguiente algoritmo:

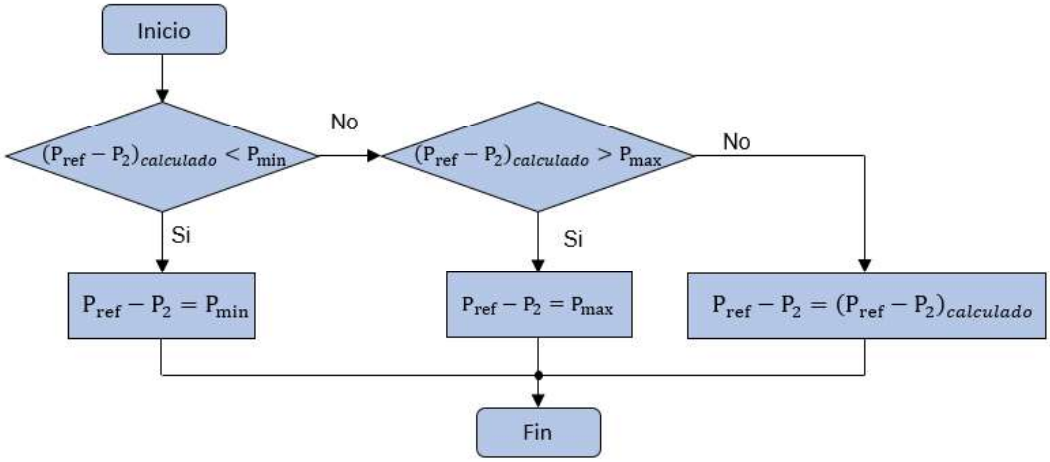


Figura 2.5. Algoritmo para la saturación del GOV

Los rangos en que pueden variar los parámetros del regulador se detalla en la tabla 2.4

Tabla 2.4. Valores límite de los parámetros del GOV

Ganancia gobernador	K	5 – 30
Constante de tiempo de retardo	T_1	0 – 100
Constante de tiempo de adelanto	T_2	0 – 10
Constante de tiempo actuador compuerta	T_3	0,04 – 1
Constante de tiempo arranque agua	T_4	0,04 – 5
Límite mínimo de la compuerta	P_{min}	0 – 0,5
Límite máximo de la compuerta	P_{max}	0,5 – 1,5

2.4. Inicialización del Sistema.

Establecer las condiciones iniciales del sistema de ecuaciones diferenciales es uno de los temas de mayor importancia del programa, ya que como se menciona en la sección 1.3.5, el método Runge-Kutta parte de un punto inicial $t_{n=0}$.

En sistemas de potencia, el punto inicial corresponde a la operación en estado estable que se determina resolviendo el flujo de potencia de la red, en cambio para determinar las condiciones iniciales de las funciones de transferencia de los reguladores, se debe igualar a cero las derivadas.

2.4.1. Condiciones iniciales del sistema generador-barra infinita [1] [21].

El procedimiento para calcular las condiciones iniciales es el siguiente:

1. Calcular el voltaje y corriente en los terminales del generador

De la red de la figura 2.1 los datos conocidos son:

$\overline{E_b}$ Voltaje y ángulo de la barra infinita: $1\angle 0$

x_L Reactancia de la línea de transmisión

P_t Potencia en los terminales del generador

f_p Factor de potencia del generador

De las ecuaciones de flujos de potencia se tiene:

$$\begin{cases} P_t = \dfrac{V_t E_b}{x_L} \operatorname{sen}\theta \\ Q_t = -\dfrac{V_t E_b}{x_L} \cos\theta + \dfrac{V_t^2}{x_L} \end{cases}$$

Ecuaciones 2.19 Flujo de potencia en estado estable

Ya que el sistema de ecuaciones 2.19 es sencillo se puede resolver sin iteraciones, en cambio en redes más complejas habría que resolver el flujo de potencia con métodos numéricos, por ejemplo, con Newton-Raphson.

Con $E_b = 1$ el valor de voltaje y ángulo es:

$$\begin{cases} V_t = \sqrt{\dfrac{E_b^2 + 2Q_t x_L + \sqrt{E_b^4 + 4E_b^2 Q_t x_L - 4P_t^2 x_L^2}}{2}} \\ \theta = \sin^{-1}\left(\dfrac{P_t x_L}{V_t E_b}\right) \end{cases}$$

Ecuaciones 2.20 Voltaje y ángulo en terminales

Una vez conocido $\overline{V_t} = V_t(\cos\theta + j\,\mathrm{sen}\,\theta)$, se determina el valor de $\overline{I_t}$

$$\overline{I_t} = \left(\frac{P_t + jQ_t}{\overline{V_t}}\right)^*$$

Ecuación 2.21. Corriente en terminales

2. Calcular el voltaje de excitación E_{fd}

$$\overline{E_{q_0}} = E_{q_0}\angle\delta_0 = \overline{V_t} + (R_a + jx_q)\overline{I_t}$$

Ecuación 2.22. Voltaje interno en estado estable

$$i_{q_0} + ji_{d_0} = \overline{I_t}(\cos\delta_0 - j\,\mathrm{sen}\,\delta_0)$$

Ecuación 2.23. Corriente del estator en estado estable

$$E_{fd_0} = E_{q_0} + (x_q - x_d)i_{d_0}$$

Ecuación 2.24. Voltaje de excitación en estado estable

3. Calcular las concatenaciones de flujo ψ

$$\psi_{d_0} = x_d i_{d_0} + E_{fd_0}$$
$$\psi_{q_0} = x_q i_{q_0}$$

Ecuación 2.25. Flujos del estator en estado estable

$$\psi_{f_0} = \psi_{d_0} + \left(\frac{x'_d}{x_d - x'_d}\right) E_{fd_0}$$
$$\psi_{kd_0} = \psi_{d_0}$$
$$\psi_{g_0} = \psi_{q_0}$$
$$\psi_{kq_0} = \psi_{q_0}$$

Ecuación 2.26. Flujos del rotor en estado estable

4. Calcular el torque eléctrico y mecánico

$$T_{e_0} = \psi_{d_0} i_{q_0} - \psi_{q_0} i_{d_0}$$

Ecuación 2.27. Torque eléctrico

$$T_{m_0} = T_{e_0}$$

Ecuación 2.28. Torque mecánico en estado estable

2.4.2. Condiciones iniciales del regulador de voltaje

Para encontrar el valor de las variables de estado en condiciones iniciales se iguala a cero las derivadas de las ecuaciones 2.15, dando como resultado el siguiente sistema:

$$\begin{cases} 0 = \frac{1}{T_R}\left(V_t - V_{C_0}\right) \\ 0 = \frac{1}{T_A}\left[\left(V_{ref_0} - V_{C_0} - V_{S_0}\right) K_A - V_{R_0}\right] \\ 0 = \frac{1}{T_E}\left(V_{R_0} - V_x\right) \\ 0 = -\frac{1}{T_F} V_{S_0} \end{cases}$$

Ecuaciones 2.29

Se despejan las variables de estado de las ecuaciones 2.29:

$$V_{C_0} = V_t$$
$$V_{ref_0} = \frac{V_x}{K_A} + V_t$$
$$V_{R_0} = V_x$$
$$V_{S_0} = 0$$

Ecuaciones 2.30. Condiciones iniciales del AVR

2.4.3. Condiciones iniciales del regulador de velocidad

Derivadas del sistema de ecuaciones 2.18 igual a cero:

$$\begin{cases} 0 = \frac{1}{T_1}\left[\left(\omega_{ref_0} - \omega\right)K - P_{1_0}\right] \\ 0 = \frac{1}{T_3}\left[\left(P_{1_0} - P_{2_0}\right)\right] \\ 0 = 2\left(\frac{P_{ref_0} - P_{2_0} - T_{m_0}}{T_4}\right) \end{cases}$$

Ecuaciones 2.31

Despejando las variables de estado, considerando que $\omega_{ref_0} = \omega$, da como resultado las condiciones iniciales:

$$P_{1_0} = 0$$
$$P_{2_0} = 0$$
$$P_{ref_0} = T_{m_0}$$

Ecuaciones 2.32. Condiciones iniciales del GOV

2.5. Instalación del Software para Programar en Fortran

Para que un código o script pueda ejecutarse en un computador debe ser traducido a un formato legible por la máquina, por ende, los lenguajes de programación se clasifican en dos grupos de acuerdo al tipo de traducción a código de máquina, estos son: lenguajes compilados y lenguajes interpretados. [15]

En los lenguajes compilados se emplean un compilador o ensamblador que traduce todo el script al lenguaje de máquina y se crea un archivo ejecutable que posteriormente se

puede correr sin la necesidad del compilador, lo que brinda mayor velocidad de ejecución del programa. Algunos de estos lenguajes son: C/C++, Basic, Pascal, Fortran [15][16].

En cambio, en los lenguajes interpretados, se usa un programa interprete que hace la traducción línea por línea a medida que se ejecuta el script, como consecuencia son más lentos que los lenguajes compilados, además el intérprete debe estar instalado en la máquina para poder correr el programa. La ventaja de estos lenguajes es que son más flexibles y portables debido a que pueden ejecutarse en cualquier sistema operativo siempre que tengan disponible al interprete. Ejemplos: Python, Matlab, Java, Excel [15][16].

Actualmente existen algunos compiladores para trabajar con Fortran, de paga y gratuitos, un buen compilador brinda al usuario diversas herramientas que facilitan la programación como librerías, subrutinas o ventanas de inspección para identificar errores en el código y examinar el uso de memoria. Entre los principales compiladores se tiene: Absoft-Fortran95, GNU gfortran, PathScale, HP Fortran Compiler y Microsoft Visual Studio [17].

En el presente trabajo se utiliza el compilador Visual Studio 2017 con la suite Intel(R) Visual Fortran; este módulo dispone de una de las librerías más completas de Fortran, además la IDE de VS permite escribir códigos de manera precisa y eficiente.

El programa escrito en Fortran se utiliza mediante línea de comandos en una terminal, es decir que no tiene interfaz gráfica, entonces los resultados de las simulaciones se exportan en archivos .dat para poder ser graficados en otro programa como Excel, sin embargo, se puede graficar directamente desde el programa mediante el software Gnuplot, el cual es ligero y gratuito, pero debe estar instalado en el computador, en la sección 2.5.3 se dan más detalles.

2.5.1. Visual Studio

Visual Studio (figura 2.6) dispone de un depurador que permite controlar la ejecución, es decir, decidir exactamente dónde pausar todos los subprocesos durante la ejecución del script e inspeccionar el estado del programa en ese punto. Las principales funciones son: detener todo en cualquier momento (break all), ejecutar comandos línea por línea (Step Over statements), Editar y Continuar (Edit and Continue), y establecer puntos de interrupción (Breakpoints) [18].

Una vez que hace una pausa en la ejecución de una aplicación, Visual Studio ofrece muchas formas para inspeccionar el valor de las variables locales y evaluar expresiones

complejas, todo sin salir del depurador. Incluso se puede consultar de forma interactiva estructuras de datos [18].

Figura 2.6. Logo de Visual Studio 2017

Microsoft Visual Studio es un software de paga, pero también está disponible en una versión gratuita para estudiantes, con algunas limitaciones, esta versión se llama Visual Studio Community, y es lo suficientemente completa para desarrollar cualquier aplicación de ámbito científico. Una vez descargado el instalador de la página oficial se debe instalar por lo menos (para programar en Fortran) los tres componentes que se indican en la figura 2.7.

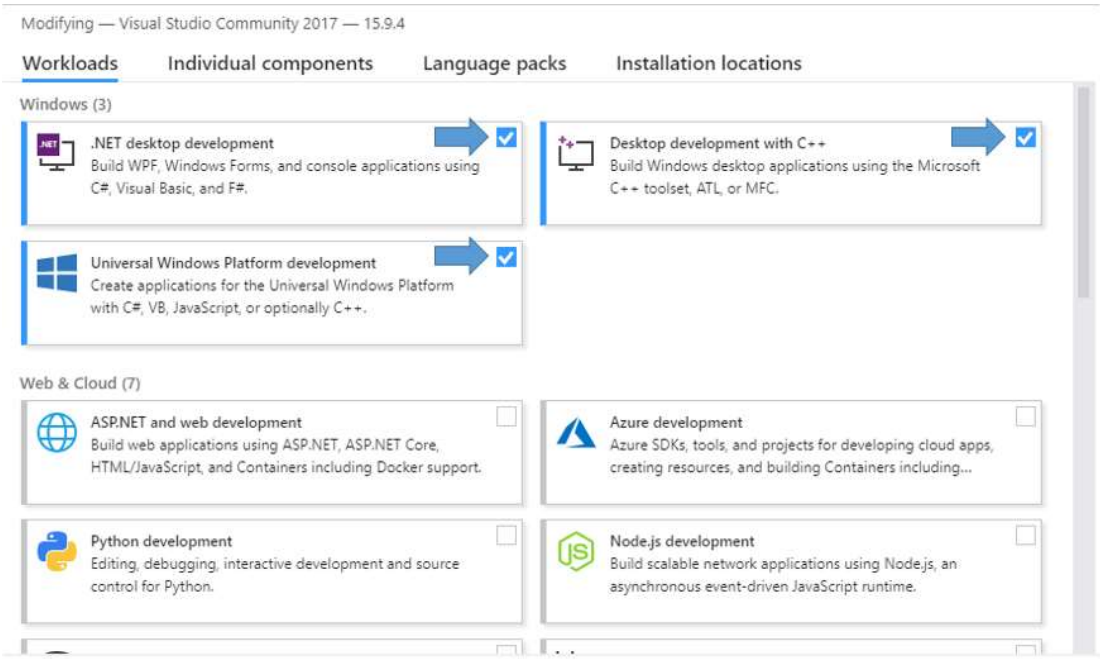


Figura 2.7. Instalación de Visual Studio 2017

Visual Studio dispone de varios lenguajes como C++, C#/VB, JavaScript y Python; pero inicialmente no incluye Fortran, es por eso que se requiere una extensión desarrollada por Intel llamada *Parallel Studio XE* .

2.5.2. Intel Parallel Studio XE

El compilador Fortran Visual de Intel (figura 2.8) es un complemento para Microsoft Visual Studio, permite la depuración de proyectos definidos por esa IDE, este conjunto completo de herramientas de desarrollo facilita la creación y modernización de código con las últimas técnicas de vectorización, paralelización multinodo y optimización de memoria [19].

Permite a los desarrolladores de software en C, C ++, Fortran y Python crear códigos más rápidos ya que aumenta el rendimiento de las aplicaciones que se ejecutan en las plataformas Intel actuales [19].

Este complemento se descarga desde la página oficial: *Intel Developer Zone*, y entre las licencias disponibles hay una sin costo para estudiantes que contiene la versión completa de Fortran 2008 y futuros lanzamientos incluirán Fortran 2018. La instalación es sencilla pero solo podrá hacerse si el equipo ya cuenta con Visual Studio 2013, 2015 o 2017.

Figura 2.8. Logo de Intel® Parallel Studio XE

2.5.3. Gnuplot

Gnuplot (figura 2.9) es una herramienta de gráficos portátil dirigida por línea de comandos para Linux, iOS, MS Windows, OSX, VMS y muchas otras plataformas. El código fuente tiene derechos de autor, pero se distribuye libremente, es decir, no se tiene que pagar por ello. Originalmente fue creado para permitir a los científicos y estudiantes visualizar las funciones matemáticas y los datos de manera interactiva, pero ha crecido para admitir muchos usos no interactivos, como los scripts web. También se utiliza como un motor de trazado por aplicaciones de terceros como Octave. Gnuplot ha estado en desarrollo activo desde 1986 [20].

Este motor de gráficos se puede utilizar de forma independiente o como complemento de cualquier lenguaje de programación. Admite también muchos tipos de gráficos en 2D y 3D y se puede dibujar utilizando líneas, puntos, cuadros, contornos, campos vectoriales, superficies, etc., además permite muchos tipos diferentes de salida: terminales de pantalla interactiva con entrada de mouse y acceso directo desde cualquier directorio, salida directa a los trazadores de lápiz o impresoras modernas. Las gráficas se pueden exportar a muchos formatos de archivo como: eps, emf, fig, jpeg, LaTeX, pdf, png, postscript [20].

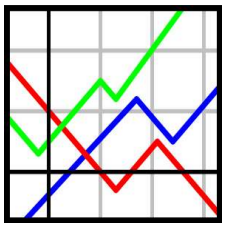


Figura 2.9. Logo de Gnuplot

Para poder ejecutar un script de Gnuplot desde Fortran es importante que durante la instalación se marque la opción de añadir el directorio de Gnuplot a la ruta de las variables de entorno del sistema, como se muestra en la figura 2.10.

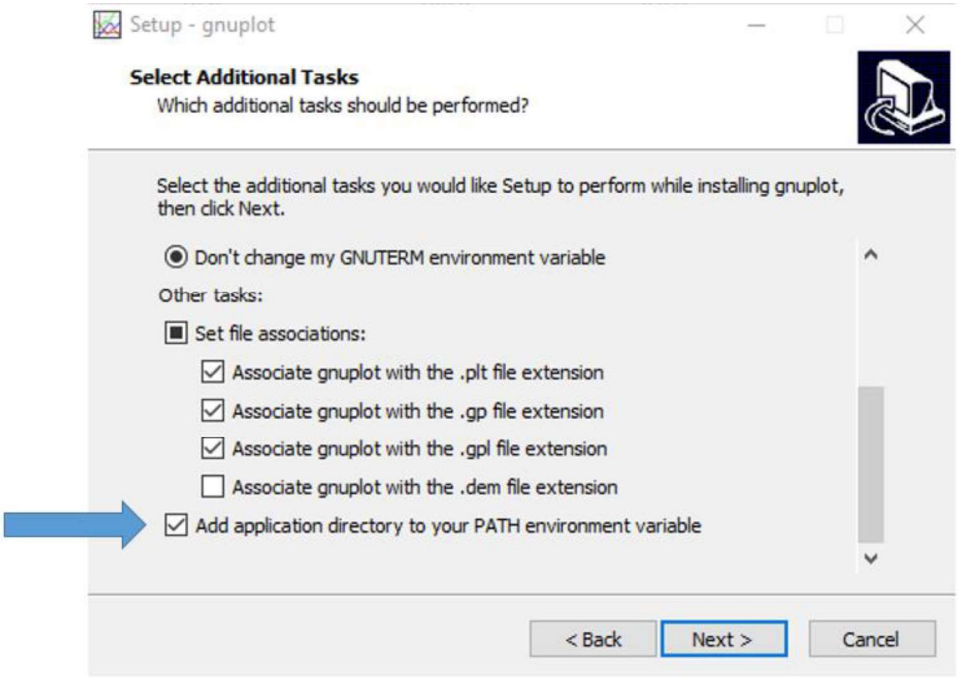


Figura 2.10. Instalación de Gnuplot

2.6. Simulación del Sistema en Fortran

Una vez determinadas todas las ecuaciones diferenciales con sus respectivas condiciones iniciales se implementa el programa en Fortran considerando las instrucciones y comandos detallados en el ANEXO I. En la definición de variables se recomienda asignar el valor de `pi` y del número imaginario `j` de la siguiente forma:

```
real,parameter::pi=4.0*atan(1.0)
complex,parameter::j=(0.0,1.0)
```

Como el programa es interactivo, el usuario puede ingresar los parámetros del generador y de los reguladores, para ello se incluye en el directorio del ejecutable (MSBI_avr_gov_2_1.exe) los archivos que contienen los datos mencionados y que pueden ser modificados con el editor de textos de Windows

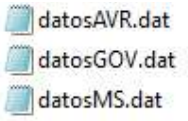


Figura 2.11. Archivos de datos

Los datos que contienen los archivos de la figura 2.11 se detallan a continuación:

Tabla 2.5. Parámetros del generador sincrónico [22]

Potencia nominal	Sn	MVA	100,0
Potencia generada	Pg	MW	80,0
Factor de potencia	fp	Ind.	0,9
Frecuencia	f	Hz	60,0
Resistencia de armadura	Ra	p.u.	0,00265
Constante de Inercia	H	seg.	3,77
Amortiguamiento mecánico	D	p.u.	0,0
Reactancias sincrónicas	Xd	p.u.	1,0225
	Xq	p.u.	0,6334
Reactancia transitoria	Xd'	p.u.	0,355
Reactancias subtransitorias	Xd"	p.u.	0,275
	Xq"	p.u.	0,275
Constante de tiempo transitoria c.c.	Tdo'	seg.	7,9
Constantes de tiempo subtransitorias c.c.	Tdo"	seg.	0,032
	Tqo"	seg.	0,055

Tabla 2.6. Parámetros del regulador de voltaje AVR

Retraso de medición	T_R	seg.	0,02
Ganancia del regulador	K_A	p.u.	200
Constante de tiempo ganancia	T_A	seg.	0,03
Constante de tiempo excitatriz	T_E	seg.	0,2
Ganancia estabilizador	K_F	p.u.	0,05
Conste de tiempo estabilizador	T_F	seg.	1,5
Factor de saturación 1	E_1	p.u.	3,9
Factor de saturación 2	Se_1	p.u.	0,1
Factor de saturación 3	E_2	p.u.	5,2
Factor de saturación 4	Se_2	p.u.	0,5
Control de voltaje mínimo	V_{min}	p.u.	-10
Control de voltaje máximo	V_{max}	p.u.	10

Tabla 2.7. Parámetros del regulador de velocidad GOV

Ganancia gobernador	K	p.u.	5,0
Constante de tiempo de retardo	T_1	seg.	0,5
Constante de tiempo de adelanto	T_2	seg.	0,1
Constante de tiempo actuador compuerta	T_3	seg.	0,95
Constante de tiempo arranque agua	T_4	seg.	0,8
Límite mínimo de la compuerta	P_{min}	p.u.	0,0
Límite máximo de la compuerta	P_{max}	p.u.	1,0

Para crear vectores de gran tamaño (ej. tiempo) se recomienda usar un bucle DO implícito de la siguiente manera:

```
Ndatos=nint(tstop/e)
t=(/(i,i=0,Ndatos)/)*e
```

Donde `tstop` es el tiempo que dura la simulación y `e` representa el error del algoritmo RK4, un valor aceptable es `e=0,005`

Una vez seleccionado el tipo de evento se procede a calcular los factores k1, k2, k3 y k4 del algoritmo Runge-Kutta (sección 1.3.5) para cada paso de tiempo t_n y se resuelve el sistema EDO con la ecuación 1.63. Concluida las iteraciones se almacenan los resultados en archivos .dat, estos archivos se guardan automáticamente en el mismo directorio del programa.

Figura 2.12. Nombre de los archivos de resultados

Cada archivo de la figura 2.12 contiene diferentes señales en forma de arreglos incluido el vector tiempo, por lo que es posible importar los datos desde Excel o Matlab para graficarlos, pero como ya se había mencionado, en este trabajo se emplea Gnuplot para mayor versatilidad.

Entonces para graficar en Gnuplot primero se crea archivos de texto con la extensión .plt, en este trabajo se usa un script para cada señal (figura 2.13), estos documentos deben ir en el mismo directorio del programa.

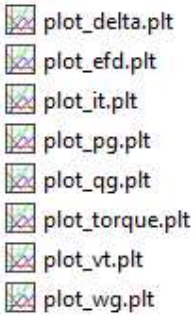


Figura 2.13. Archivos de Gnuplot

Luego con el editor de textos se escribe los comandos necesarios para representar la señal como se desee (título del gráfico, etiqueta de los ejes, cuadrícula, tipo de línea, etc.), con el comando `plot` se toma los datos de las respectivas magnitudes (figura 2.12). Como ejemplo se muestra el script para graficar el voltaje en terminales:

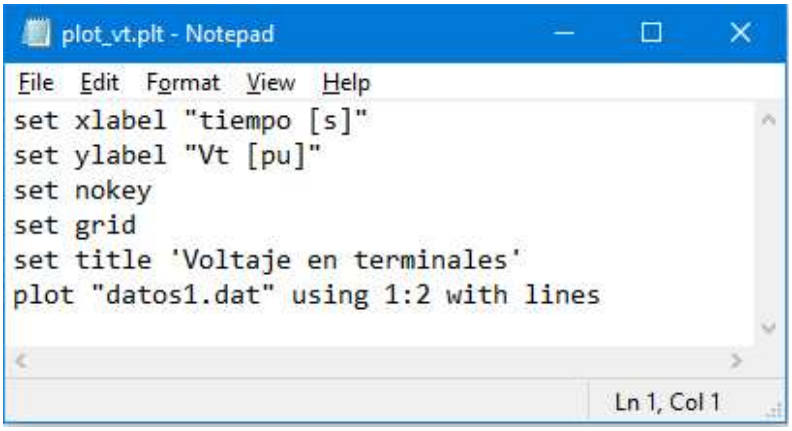


Figura 2.14. Script para graficar la señal v_t

En el código del programa en Fortran se llama al script de Gnuplot para ejecutarlo directamente desde ahí, empleando la siguiente sintaxis:

```
call system('wgnuplot -p plot_vt.plt')
```

El archivo ejecutable (figura 2.16) se crea automáticamente al iniciar la depuración y compilación en Visual Studio seleccionando la siguiente opción:

Figura 2.15. Depuración y compilación del programa

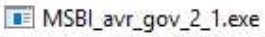


Figura 2.16. Programa ejecutable

Al ejecutar el programa que se muestra en la figura 2.16 se abre la siguiente ventana:

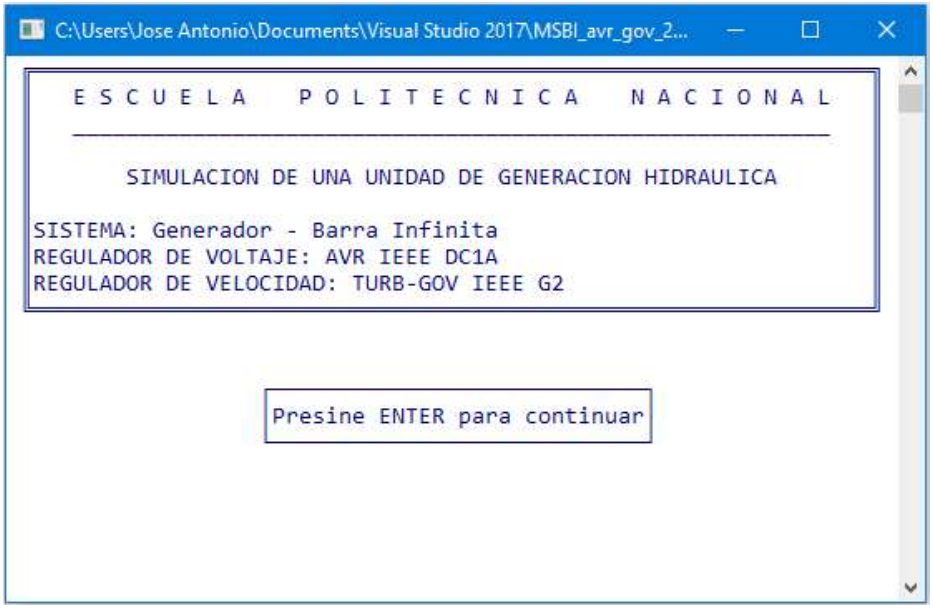


Figura 2.17. Ventana de inicio del programa

A continuación, se ingresa el tiempo que dura la simulación:

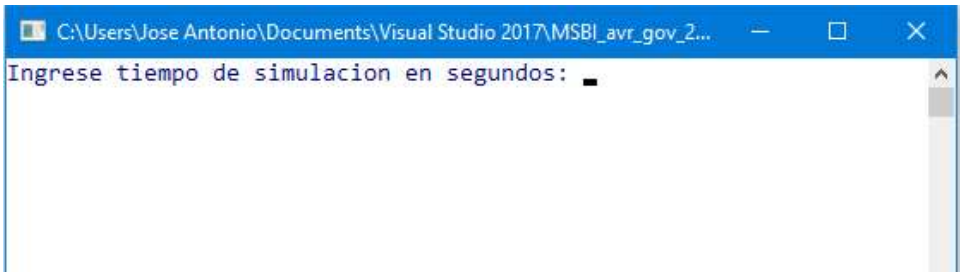


Figura 2.18. Lectura del tiempo de simulación

Después se escoge el tipo de evento a simular:

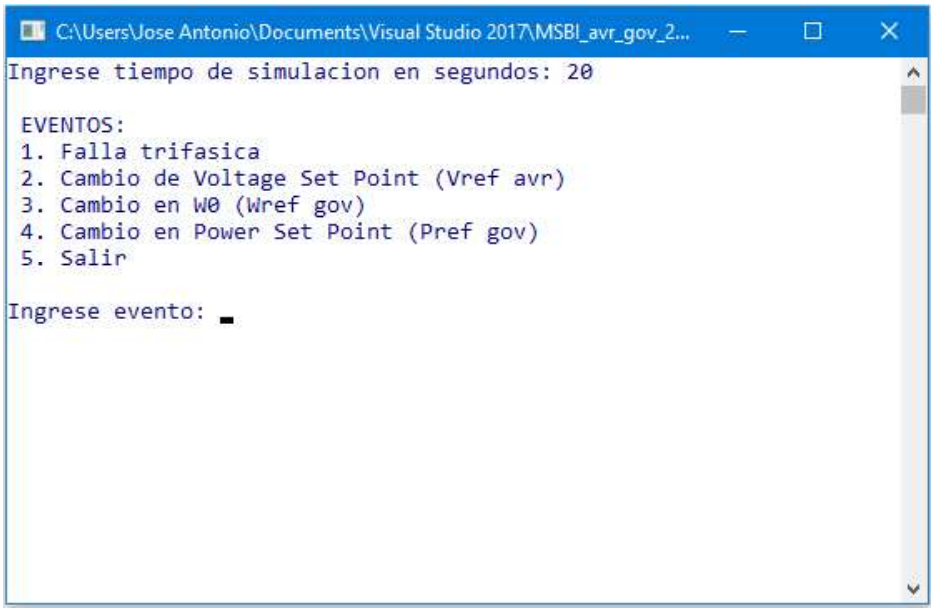


Figura 2.19. Ingreso de evento

Como ejemplo se realiza la simulación de variación en la potencia de referencia, entonces el programa pide que se ingrese el tiempo en que ocurre el cambio del parámetro:

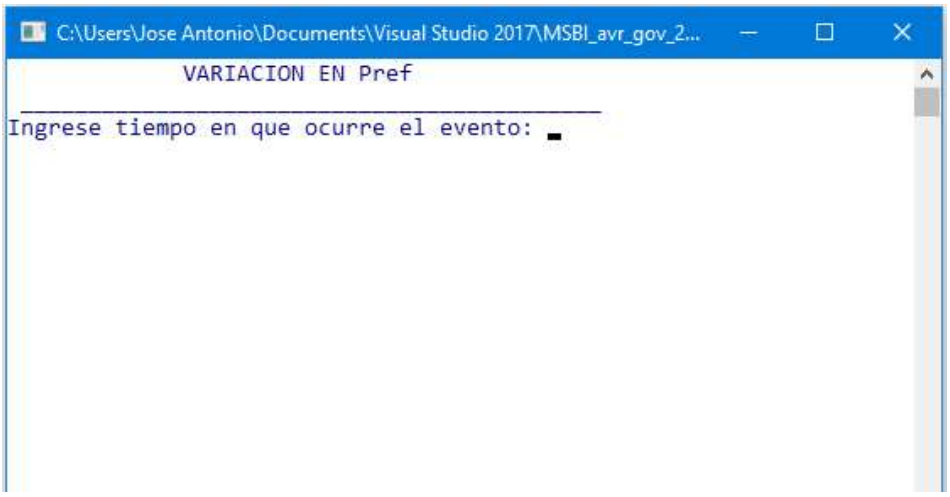


Figura 2.20. Lectura del tiempo en que varía la Pref

Se muestra el valor del parámetro en estado estable (set point), en base a esto el usuario sabe cuánto se puede modificar esa magnitud.

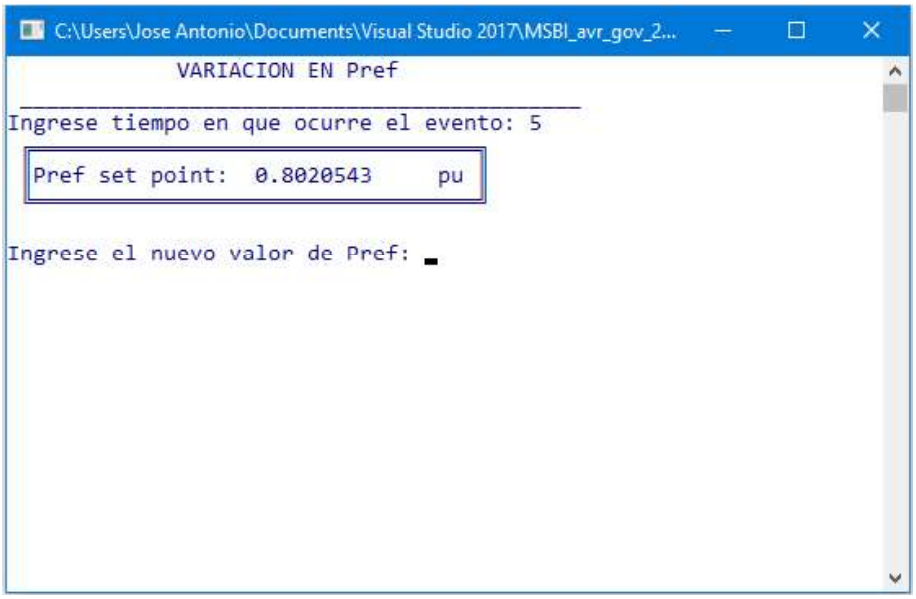


Figura 2.21. Ingreso del nuevo valor del parámetro

Se indica que la simulación se completó satisfactoriamente, y se dan las opciones para visualizar las diferentes señales de salida del generador.

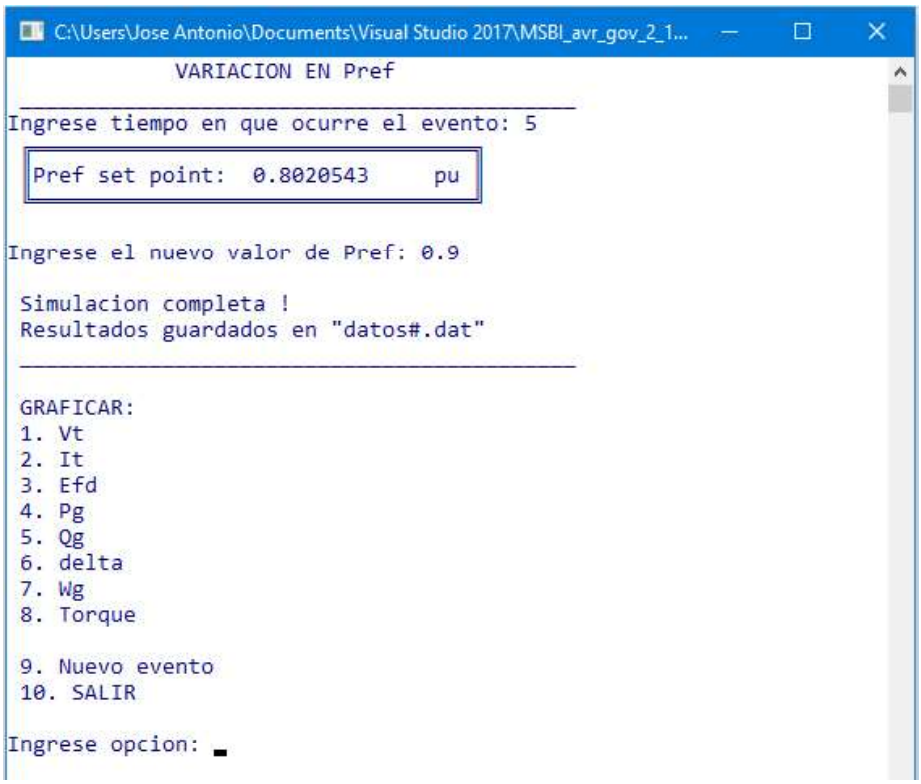


Figura 2.22. Selección del gráfico

Como ejemplo se indica a continuación el gráfico de una de las señales obtenidas:

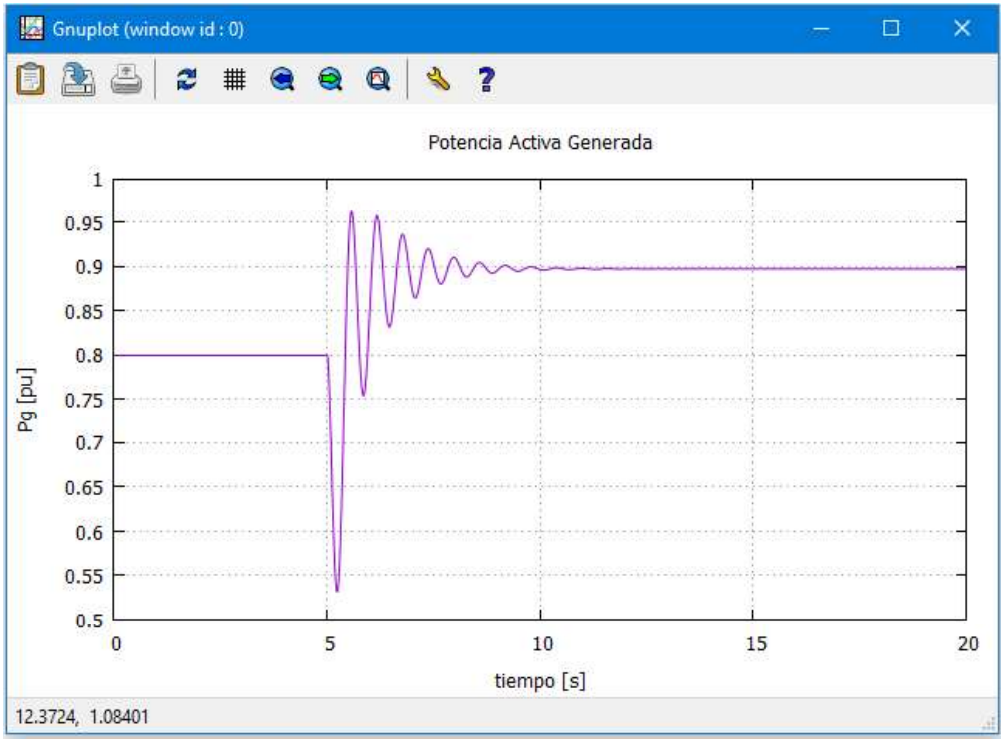


Figura 2.23. Variación de la potencia de referencia

En la ventana de Gnuplot (figura 2.23) se puede exportar el gráfico a diferentes formatos de imagen, imprimir, hacer zoom a la gráfica, cambiar el tamaño de la ventana, mostrar coordenadas, etc.

El código completo del programa se muestra en el ANEXO III

2.7. Simulación del Sistema en DIgSILENT PowerFactory

La red para modelar el sistema generador - barra infinita (MSBI) se muestra en la figura 2.24. La línea de transmisión tiene una reactancia de 0,48736 ohm/km y una longitud de 10 km, el voltaje nominal del generador es de 138 kV. La red externa se configura como slack bus 1∠0.

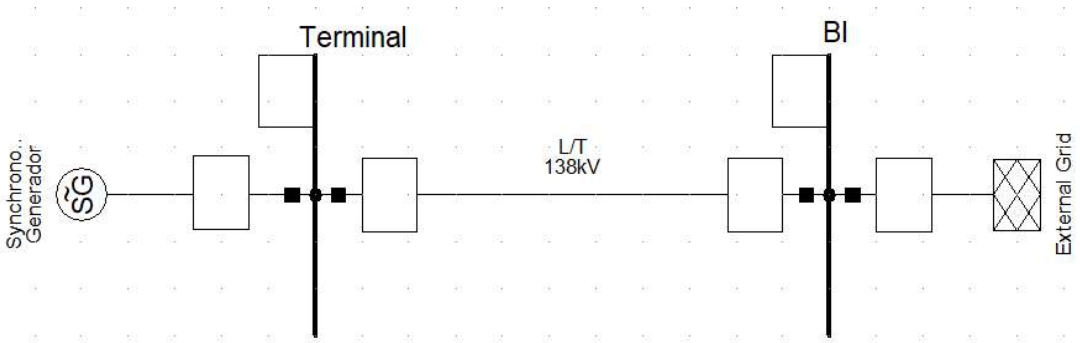


Figura 2.24 Red MSBI

Los parámetros del generados se establecen de acuerdo a la tabla 2.5. El procedimiento para crear el frame del sistema, así como los modelos de los reguladores, se indica en el ANEXO II.

3. RESULTADOS

3.1. Comparación de Simulaciones de Eventos entre Fortran y DIgSILENT

Para validar los resultados del software didáctico desarrollado en este proyecto se realizan diferentes simulaciones dinámicas del generador sincrónico ante eventos transitorios, dicha simulación se compara con resultados que se obtienen en el software DIgSILENT PowerFactory. Los eventos o maniobras son: Variación en el valor set point del voltaje en terminales (voltaje de referencia), variación en el valor set point de la potencia generada (potencia de referencia), variación en el valor set point de la velocidad angular del rotor en estado estable (velocidad de referencia) y fallas trifásicas en diferentes puntos del sistema (terminales del generador, línea de transmisión y barra infinita).

3.1.1. Variación en el voltaje de referencia

Tiempo de simulación: 20 seg.

Evento ocurre en el tiempo: 1 seg.

Incremento de Vref a: 1.02 p.u.

Resultados:

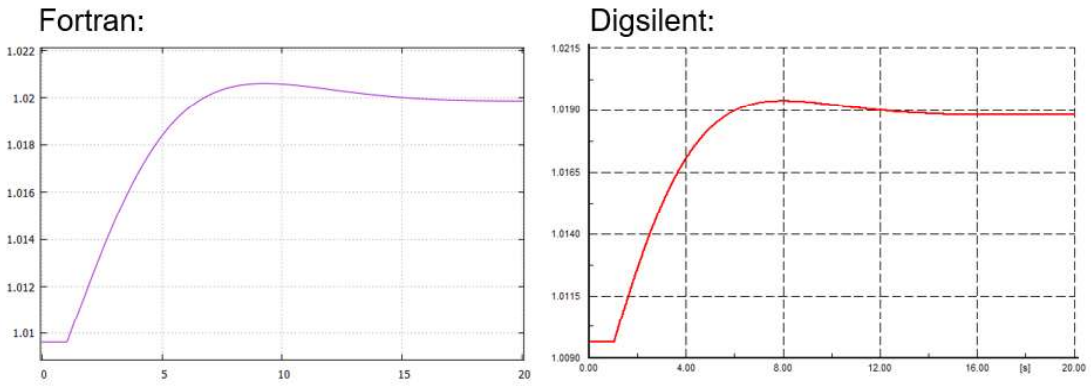


Figura 3.1. Voltaje en terminales (Vt)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta V_t = 0.196$ %

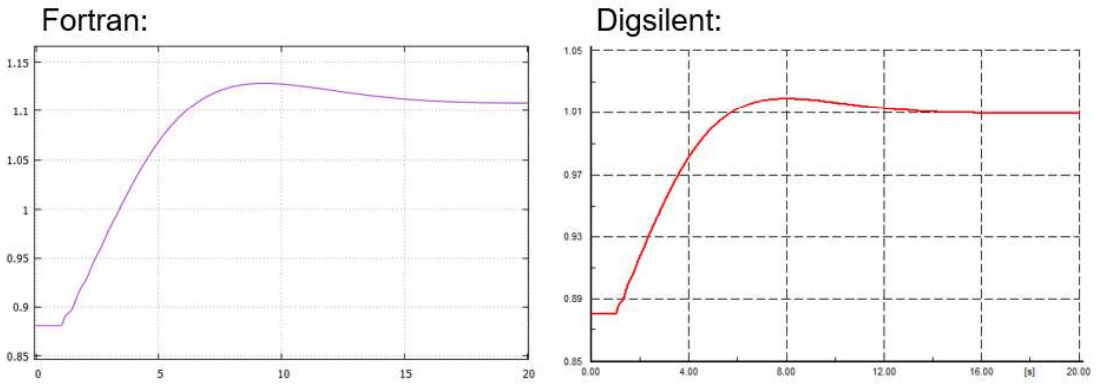


Figura 3.2. Corriente en terminales (It)

Variación máxima de la señal en Fortran con respecto a Digsilent: ΔI_t = 9.812 %

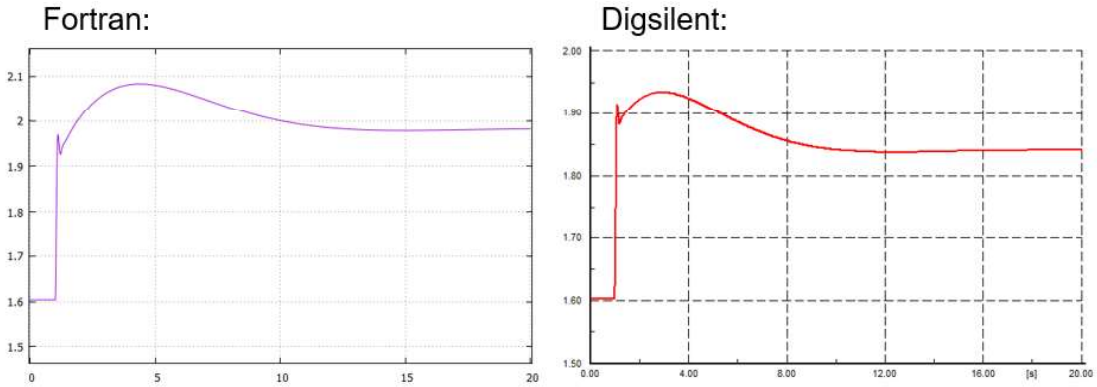


Figura 3.3. Voltaje de excitación (Efd)

Variación máxima de la señal en Fortran con respecto a Digsilent: ΔE_{fd} = 7.70 %

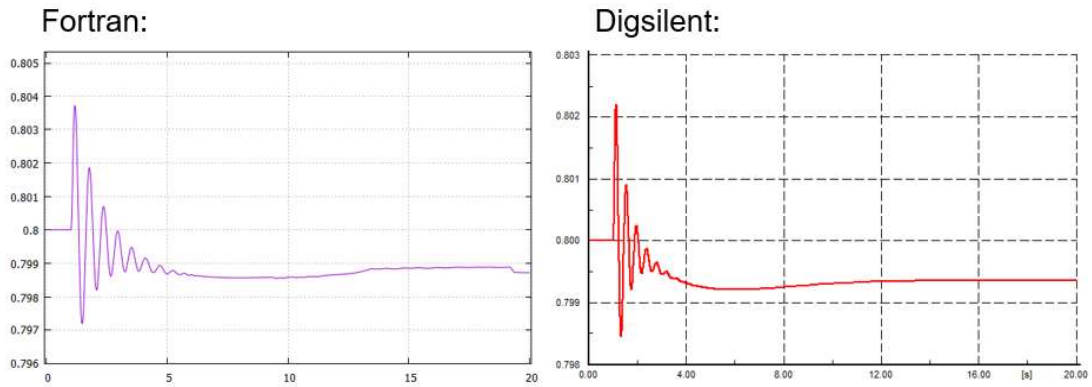


Figura 3.4. Potencia activa generada (Pg)

Variación máxima de la señal en Fortran con respecto a Digsilent: ΔP_g = 0.12%

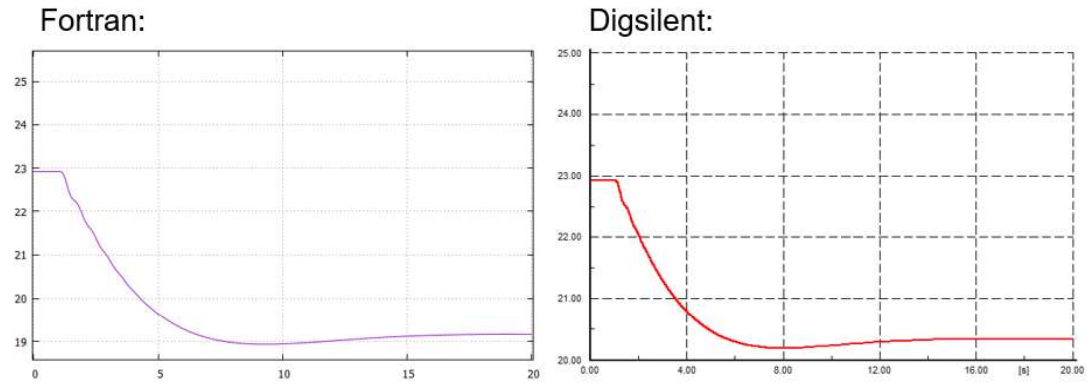


Figura 3.5. Ángulo de potencia (δ)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\delta$ = 6.24 %

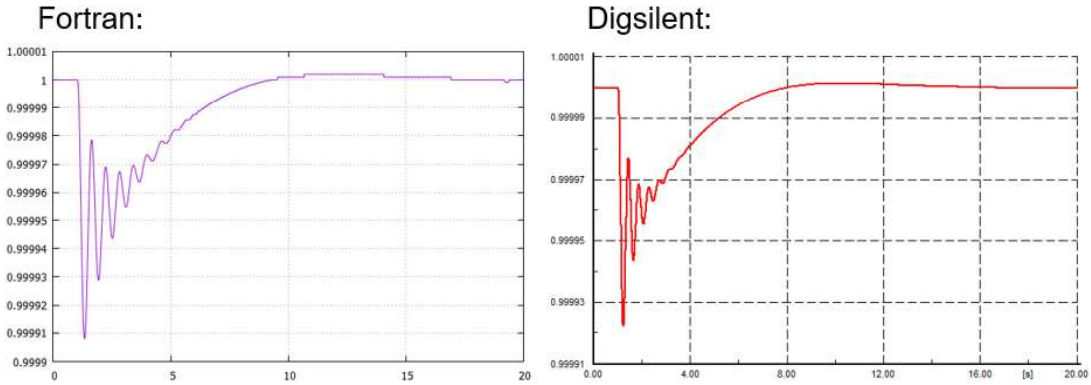


Figura 3.6. Velocidad angular del rotor (ω)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\omega$ = 0.01 %

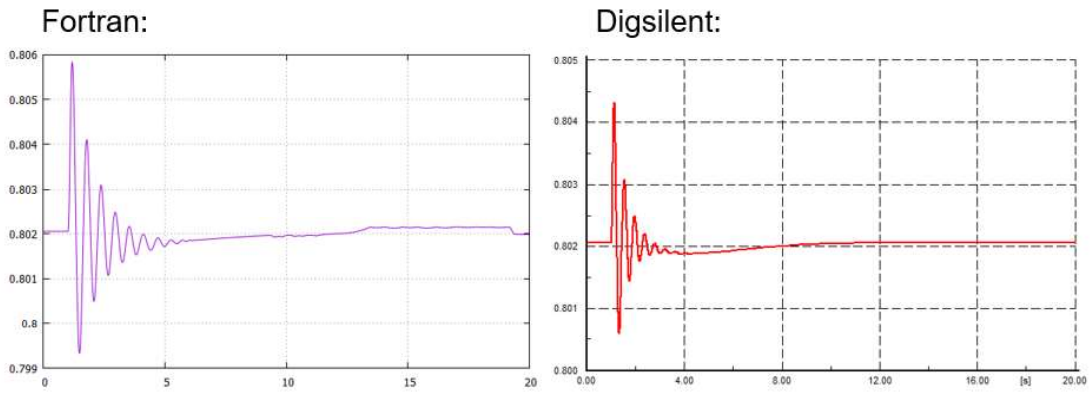


Figura 3.7. Torque eléctrico (Te)

Variación máxima de la señal en Fortran con respecto a Digsilent: ΔTe = 0.22%

3.1.2. Variación en la velocidad de referencia

Tiempo de simulación: 20 seg.

Evento ocurre en el tiempo: 1 seg.

Decremento de Wref a: 0.98 p.u.

Resultados:

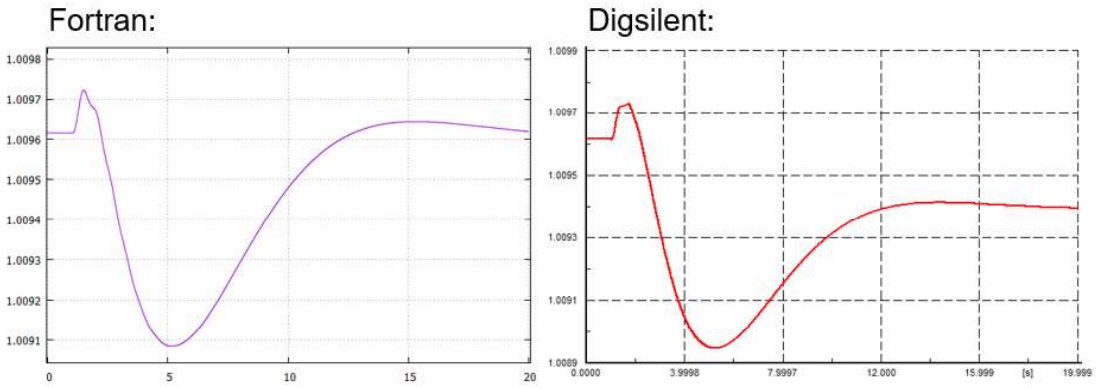


Figura 3.8. Voltaje en terminales (Vt)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta V_t = 0.06$ %

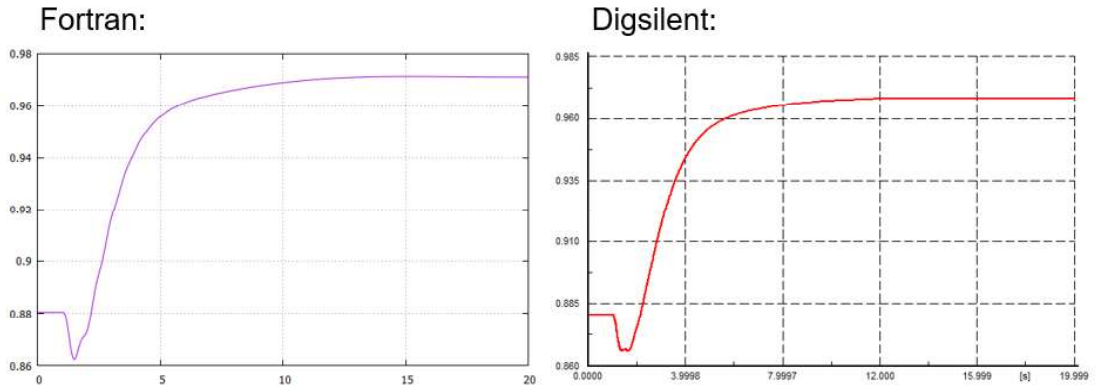


Figura 3.9. Corriente en terminales (It)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta I_t = 0.46$ %

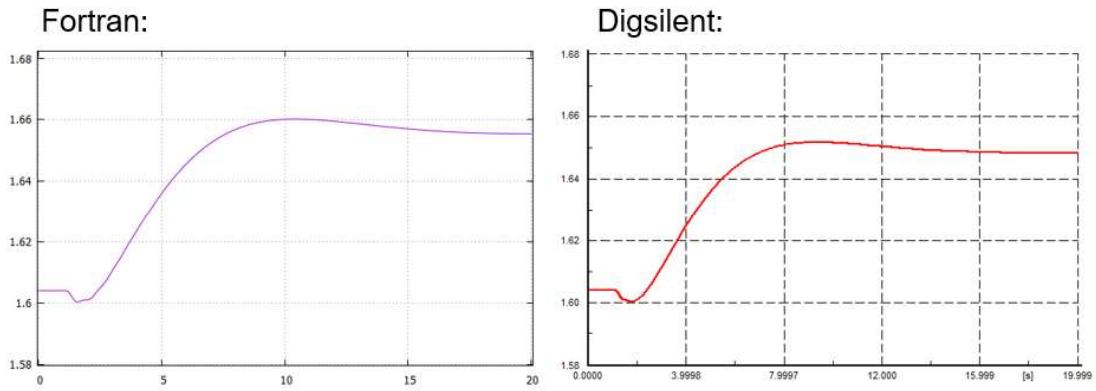


Figura 3.10. Voltaje de excitación (Efd)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta E_{fd} = 0.42$ %

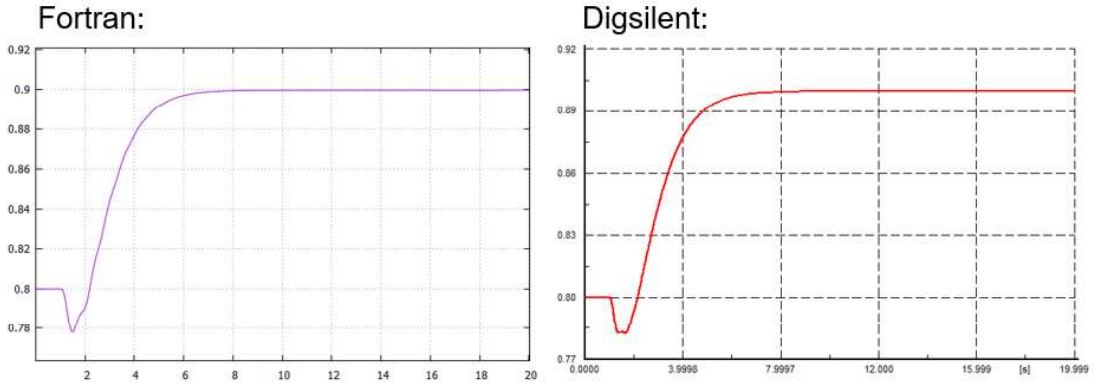


Figura 3.11. Potencia activa generada (Pg)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta P_g = 0.51$ %

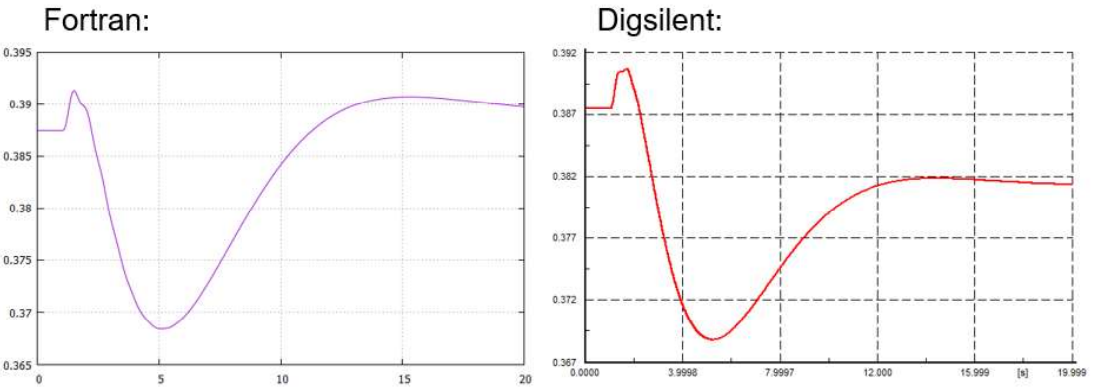


Figura 3.12. Potencia reactiva generada (Qg)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta Q_g = 2.3$ %

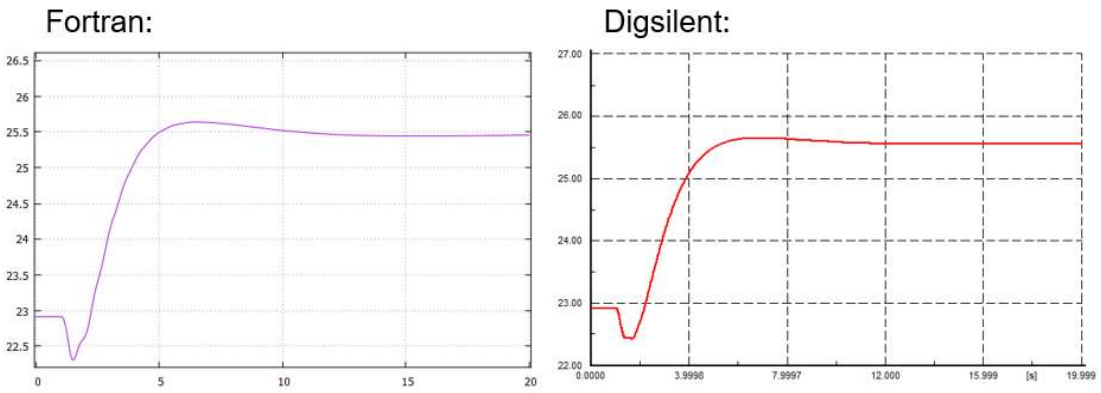


Figura 3.13. Ángulo de potencia (δ)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\delta$ = 0.54 %

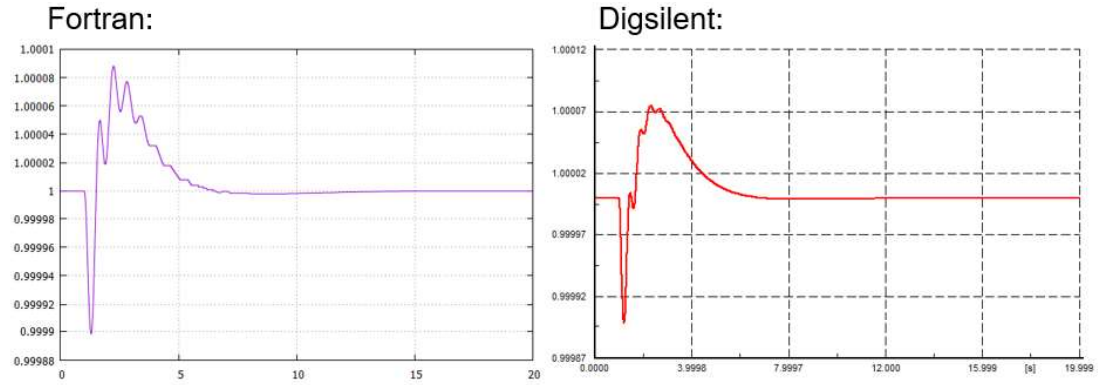


Figura 3.14. Velocidad angular del rotor (ω)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\omega$ = 0.01 %

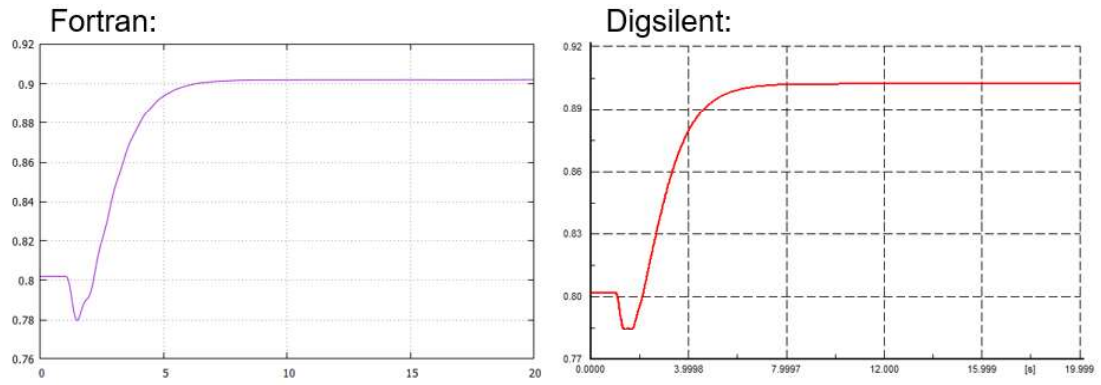


Figura 3.15. Torque eléctrico (Te)

Variación máxima de la señal en Fortran con respecto a Digsilent: ΔTe = 0.52 %

3.1.3. Variación en la potencia de referencia

Tiempo de simulación: 20 seg.

Evento ocurre en el tiempo: 1 seg.

Incremento de Pref a: 0.9 p.u.

Resultados:

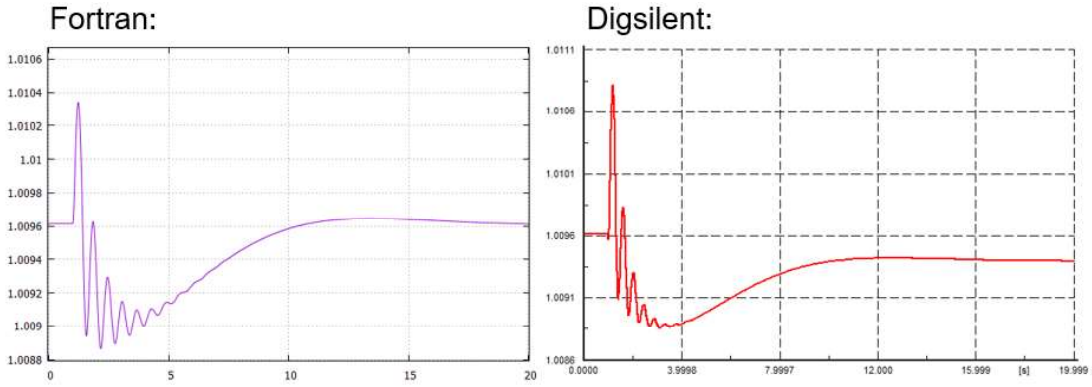


Figura 3.16. Voltaje en terminales (Vt)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta V_t = 0.09$ %

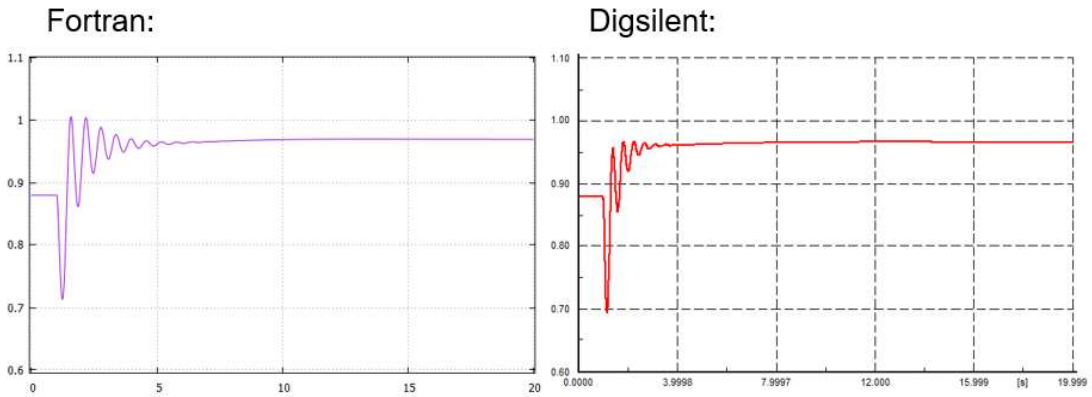


Figura 3.17. Corriente en terminales (It)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta I_t = 3.78$ %

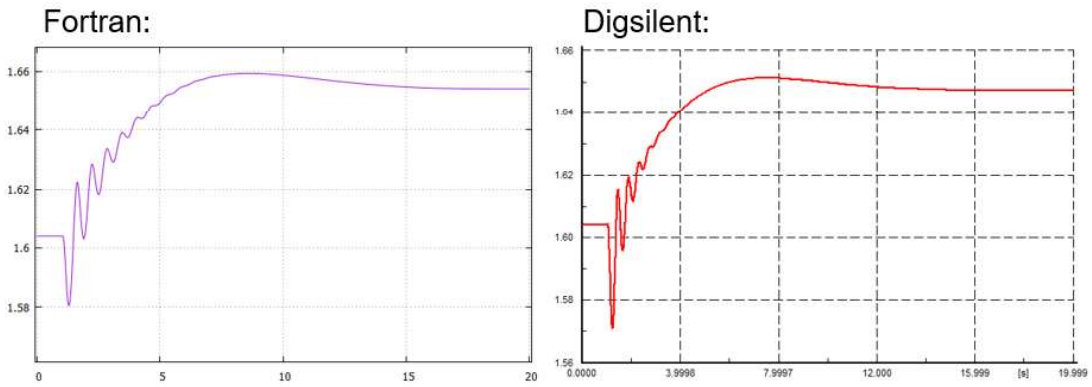


Figura 3.18. Voltaje de excitación (Efd)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta E_{fd} = 0.43$ %

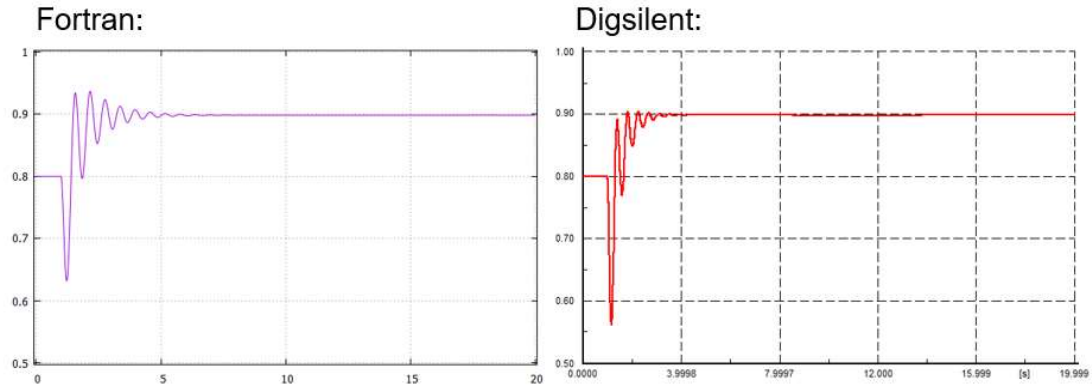


Figura 3.19. Potencia activa generada (Pg)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta P_g = 4.65$ %

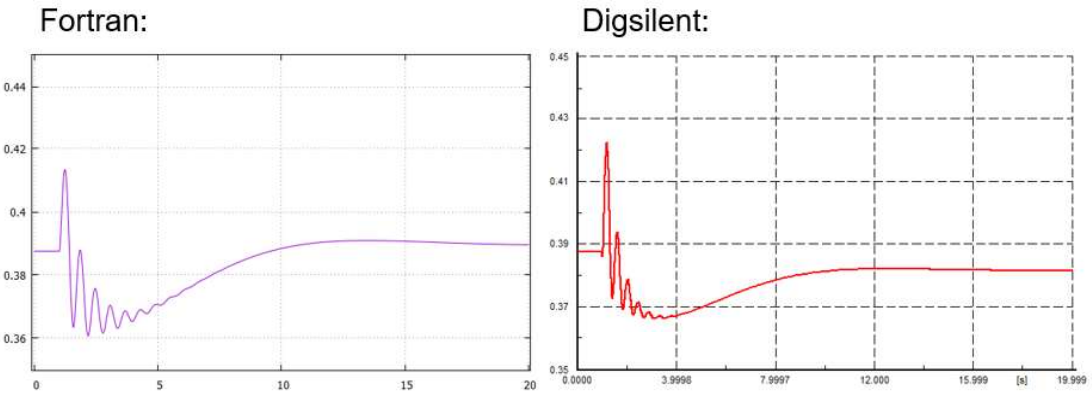


Figura 3.20. Potencia reactiva generada (Qg)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta Q_g = 0.02$ %

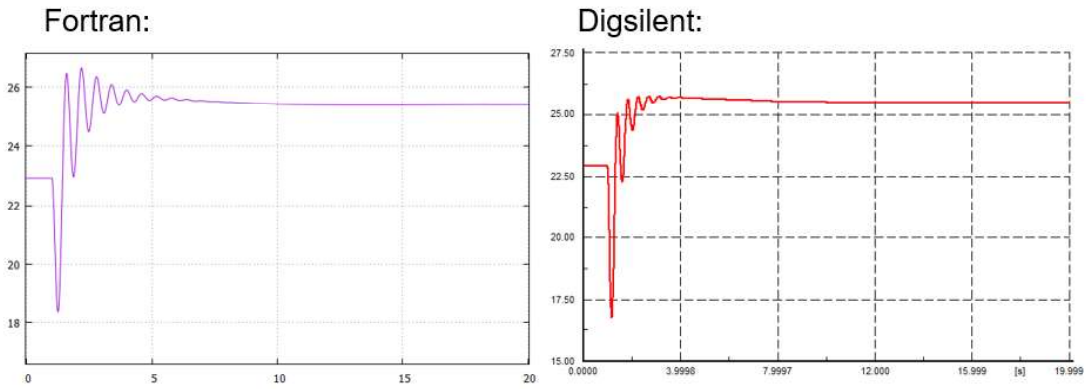


Figura 3.21. Ángulo de potencia (δ)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\,\delta$ = 4.31 %

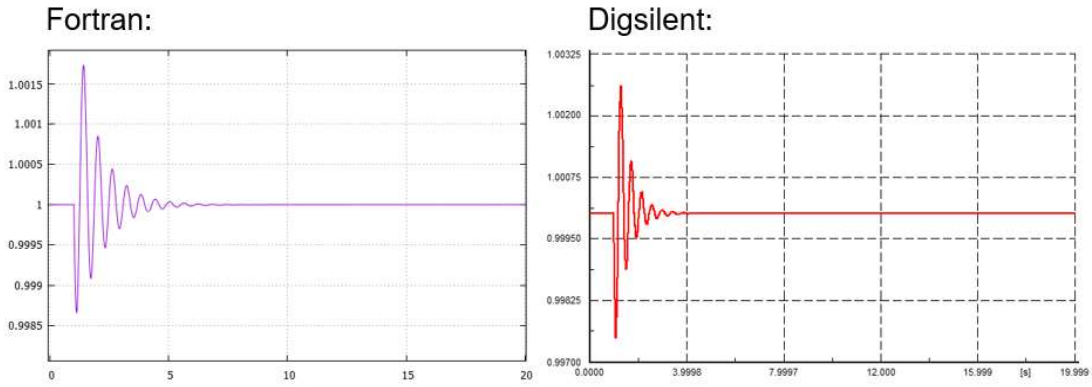


Figura 3.22. Velocidad angular del rotor (ω)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\omega$ = 0.09 %

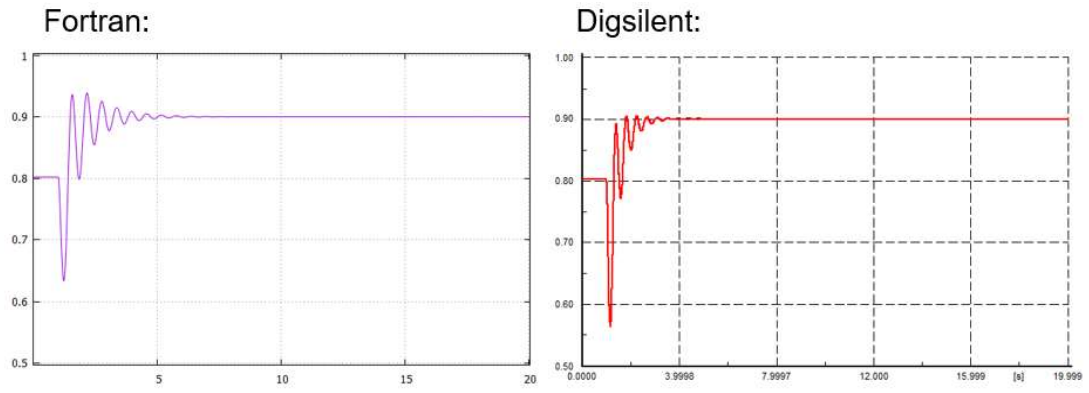


Figura 3.23. Torque eléctrico (Te)

Variación máxima de la señal en Fortran con respecto a Digsilent: ΔTe = 4.97 %

3.1.4. Falla trifásica

Tiempo de simulación: 10 seg.

Tiempo en que ocurra falla: 1 seg.

Tiempo que dura falla: 1 ms

Ubicación de falla: Barra infinita

Resultados:

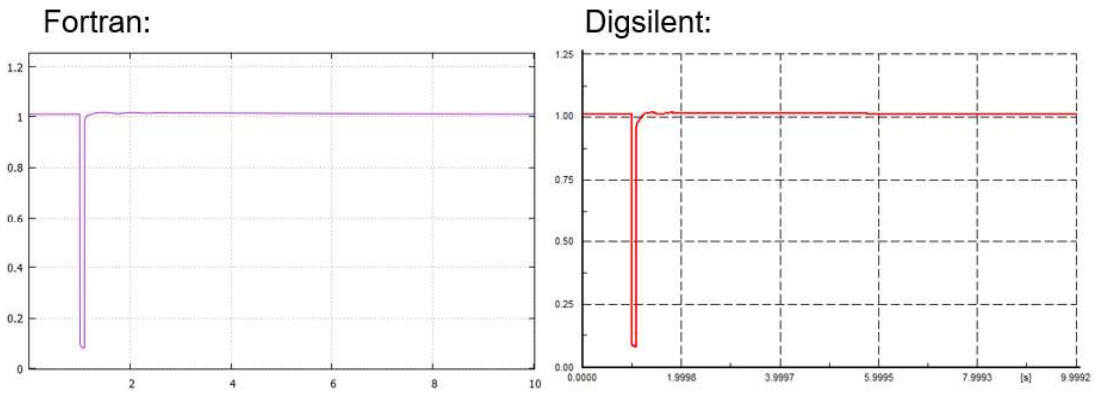


Figura 3.24. Voltaje en terminales (Vt)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta V_t = 2.5\ \%$

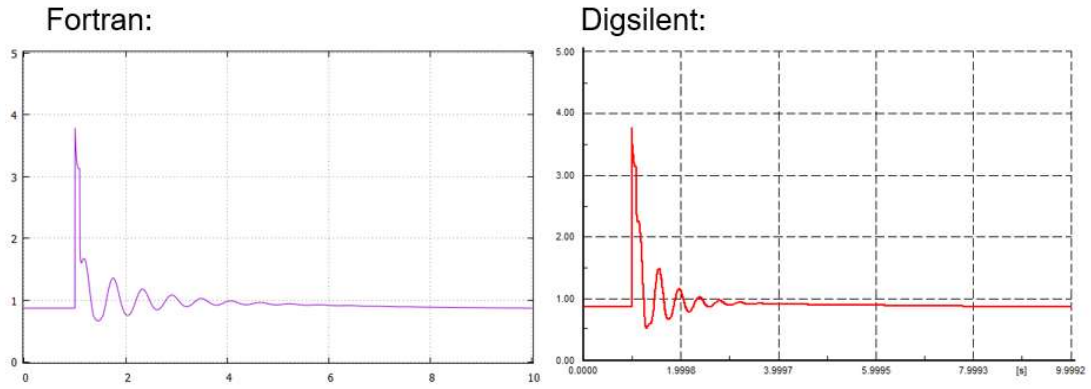


Figura 3.25. Corriente en terminales (It)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta I_t = 0.5\ \%$

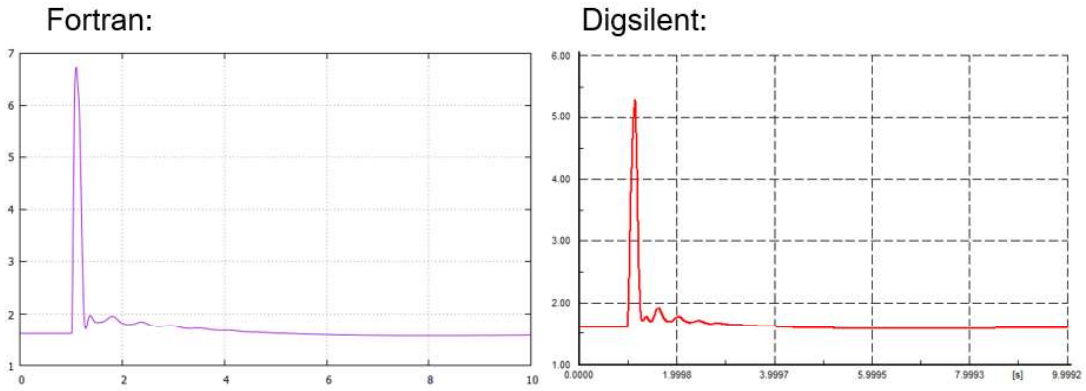


Figura 3.26. Voltaje de excitación (Efd)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta E_{fd} = 4.22$ %

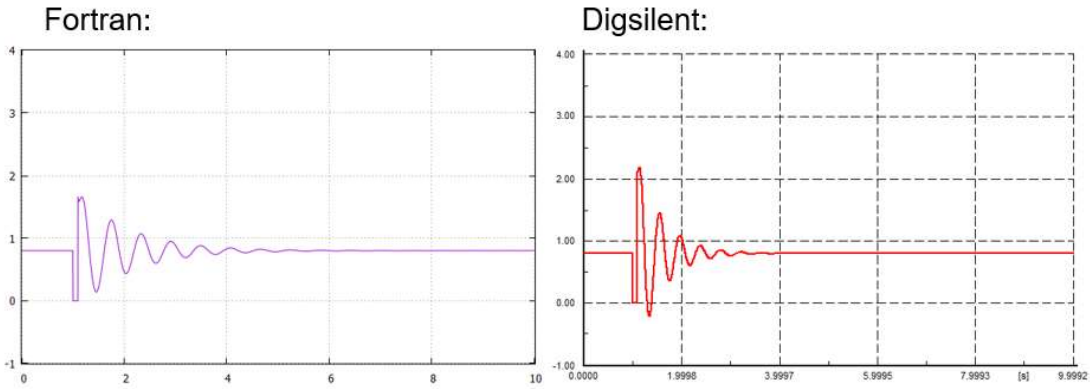


Figura 3.27. Potencia activa generada (Pg)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta P_g = 9.31$ %

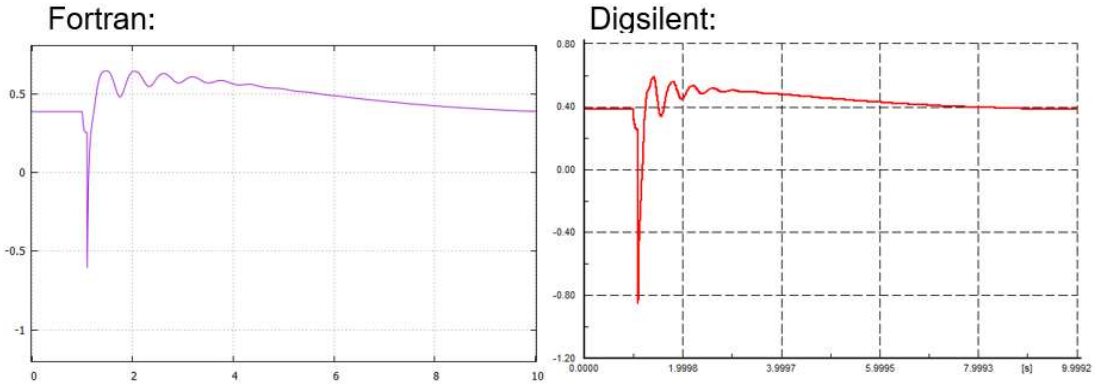


Figura 3.28. Potencia reactiva generada (Qg)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta Q_g = 9.19$ %

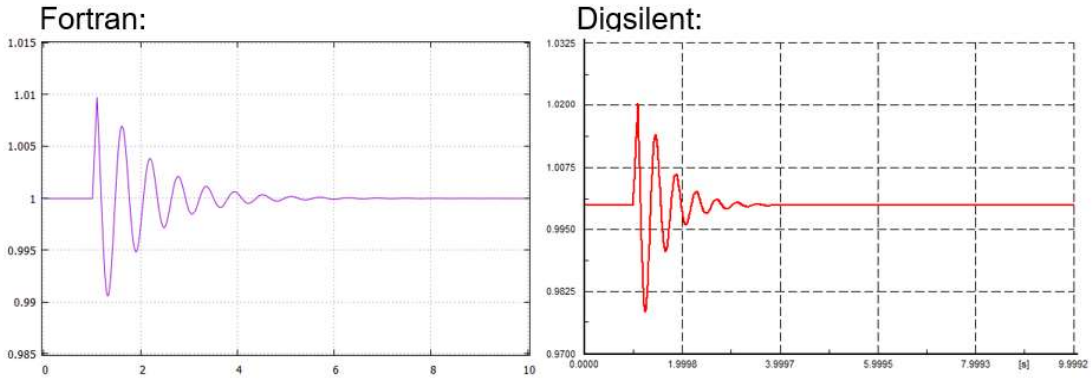


Figura 3.29. Velocidad angular del rotor (ω)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta\ \omega = 0.98\ \%$

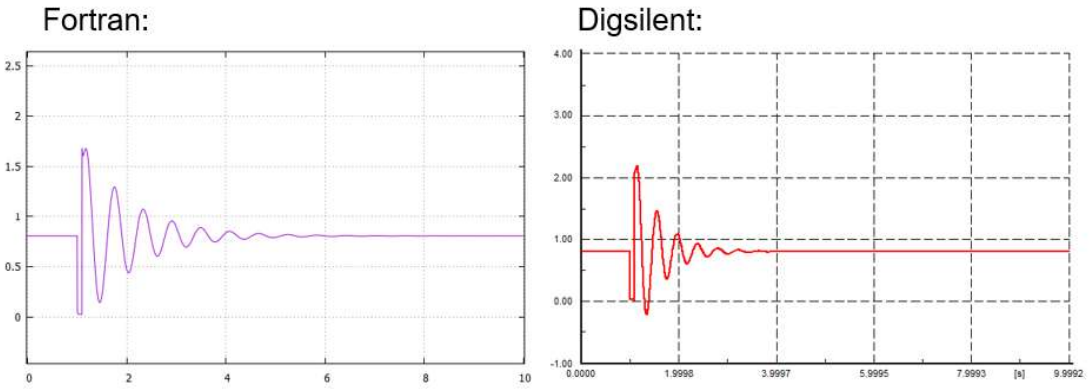


Figura 3.30. Torque eléctrico (Te)

Variación máxima de la señal en Fortran con respecto a Digsilent: $\Delta Te = 11.13\ \%$

3.2. Análisis de Resultados

Como se puede apreciar en las gráficas de este capítulo, en todos los eventos transitorios, las señales obtenidas en Fortran son bastantes similares a las de Digsilent, aunque se puede ver que hay una mínima diferencia en el valor de las magnitudes, en general, las señales siguen la misma tendencia en ambos programas.

Las variaciones en las magnitudes se deben a que en Digsilent se considera otros parámetros como variables de estado, sin embargo los resultados son prácticamente los mismos que en el software didáctico, por ende se concluye que los resultados empleando

los modelos matemáticos vistos en el capítulo 1 son aceptables para la simulación de la máquina sincrónica y que además, la programación en Fortran fue realizada satisfactoriamente, así como la deducción de las ecuaciones para los reguladores y la implementación del algoritmo Runge-Kutta como una alternativa para resolver sistemas complejos de ecuaciones diferenciales ordinarias.

Al implementar el software programado en Fortran es notable el poco tiempo en que este tarda en completar la simulación y en dibujar las gráficas, en comparación a otros softwares que se emplean, por ejemplo, Matlab, inclusive la resolución tarda menos que Digsilent y aunque esta diferencia de tiempo es poca, hay que considerar que se trata de un sistema simple, por lo que en redes más complejas que incluyan sistemas multimáquina y con modelos completos de líneas de transmisión y transformadores, se evidenciaría las enormes ventajas de Fortran en el ahorro de consumo de memoria y tiempo de simulación.

4. CONCLUSIONES

- La comparación de los resultados de las señales entre el software comercial Digsilent y el software didáctico hecho en Fortran reflejan la confiabilidad del modelo que se implementó, por lo tanto, la metodología de programación presentada en este proyecto sirve como referencia para el diseño de futuros programas referentes a estudios de estabilidad en sistemas de potencia.

- Habiéndose cumplido el objetivo de modelar un sistema eléctrico mediante la programación en Fortran como una alternativa para optimizar recursos de hardware, se recomienda la utilización de este lenguaje para futuros proyectos orientados al uso del software libre y de código abierto.

- Si bien la red generador-barra infinita es simple, también constituye un sistema real y muy útil para el estudio de estabilidad en máquinas que se encuentran alejadas del centro de carga. Inclusive si se representa a una porción grande del SEP como un generador ficticio, sería conveniente la implementación de este sistema como una aproximación plausible.

- Aunque inicialmente la programación en Fortran puede parecer complicada debido a la limitada información referente a este, basta con aprender los comandos básicos indicados en este trabajo para que el desarrollador encuentre en este lenguaje una poderosa herramienta para la creación de aplicaciones en el campo científico.

- Es muy importante considerar los rangos de valores de los parámetros de los reguladores al momento de parametrizarlos, porque al configurar ganancias demasiado elevadas o constantes de tiempo fuera de los límites, se producen resultados no deseados como por ejemplo amortiguamiento negativo o sobrexcitación lo que causa perdida de estabilidad del sistema.

- Se verificó la importancia de los reguladores de velocidad y voltaje para la estabilidad transitoria de un sistema eléctrico, ya que con el correcto modelamiento y parametrización de estos controles se consigue un notable amortiguamiento en las oscilaciones producto de eventos mecánicos o eléctricos.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] K. R. Padiyar, *Power System Dynamics. Stability and Control*, 2 ed., Hyderabad: BS Publications, 2008, cap. 3, 4, 6.

[2] IEEE Power Engineering Society, *Std 1110™-2002. Guide for Synchronous Generator Modeling Practices and Applications in Power System Stability Analyses*, Revision of IEEE Std 1110-1991, New York: The Institute of Electrical and Electronics Engineers Inc., 2003, p. 15 - 21

[3] H. Shayeghi, J. Dadashpour, *Anarchic Society Optimization Based PID Control of an Automatic Voltage Regulator (AVR) System*, Ardabil: Technical Engineering Department, University of Mohaghegh Ardabili, 2012, p. 202

[4] CM. Ong, *Dynamic Simulation of Electrical Machinery Using Matlab/Simulink*, New Jersey: Prentice Hall PTR, 1998, p. 487 - 490

[5] P. Kundur. *Power System Stability and Control*, California: McGraw-Hill Inc., p. 99, 152, 153, 318-320

[6] IEEE Power Engineering Society, *Std 421.5™-2005. Recommended Practice for Excitation System Models for Power System Stability Studies*, Revision of IEEE Std 421.5-1992, New York: Energy Development and Power Generation Committee, 2006, p.7

[7] IEEE Power Engineering Society, *Paper T-73-089. Dynamic Models for Steam and Hydro Turbines in Power System Studies*, New York: IEEE Committee Report, 1973, p. 1905-1909

[8] P. De León, *Métodos Numéricos para Ecuaciones Diferenciales Ordinarias. Métodos Runge–Kutta Explícitos*, [Tesis], Universidad de La Laguna, Departamento de Análisis Matemático, Santa Cruz de Tenerife, España, 2015, p. 5 - 8

[9] H. Martínez, *Métodos Numéricos*, Monterrey: Centro de Sistemas Inteligentes - Tecnológico de Monterrey, 2004, p. 32 - 40.

[10] Department of Chemistry - University of Oxford (Feb. 2001), Programming in Fortran, [Online], Disponible en:http://www.chem.ox.ac.uk/fortran/fortran1.html

[11] J. Fernández, P. Guerrón and D. Zarruk, *Scientific Computing Languages*, Pennsylvania: University of Pennsylvania, 2018, p. 31 - 34

[12] Intel Developer Zone (Sep. 2018), Intel Parallel Studio XE, [online], Disponible en: https://software.intel.com/en-us/parallel-studio-xe

[13] P. Bernardos (Ene. 2008), Aprenda a Programar en Fortran, [online], Disponible en: https://ocw.unican.es/course/view.php?id=205&section=1

[14] M. Alcubierre, *Introducción a Fortran*, México: Instituto de Ciencias Nucleares – UNAM, 2005, p. 5

[15] N. Williams (Apr. 2004), What does a compiler actually do?, [online], Disponible en: https://www.codehelp.co.uk/html/code10.html

[16] Open Directory DMOZ (Feb. 2017), Computers Programming Languages, [online], Disponible en: http://dmoztools.net/Computers/Programming/Languages/

[17] The Fortran Company (Mar. 2014), Fortran Compilers, [online], Disponible en: https://www.fortran.com/compilers.html

[18] Microsoft (Mar. 2019), Visual Studio Community 2017, [online], Disponible en: https://visualstudio.microsoft.com/

[19] Intel Developer Zone (Jun. 2018), Intel Parallel Studio XE for Windows, [online], https://software.intel.com/en-us/parallel-studio-xe/choose-download/student-windows

[20] Gnuplot (Jan. 2019), An Interactive Plotting Program, [online], Disponible en: http://www.gnuplot.info/

[21] K.N. Shubhanga, *Small-signal Stability Analysis of SMIB System*, Karnataka: Department of Electrical Engineering – NITK, 2013, p. 2 - 6

[22] Laboratorio de Sistemas Eléctricos de Potencia, *Práctica N1. Análisis de Sensitividad de Generadores Sincrónicos*, Quito: Escuela Politécnica Nacional, 2016, p. 1 - 10

6. ANEXOS

ANEXO I. Elementos y comandos básicos de Fortran

ANEXO II. Modelamiento del sistema en DIgSILENT

ANEXO III. Código del Programa

ANEXO I

Elementos y comandos básicos de Fortran [14]

Las reglas básicas para programar en Fortran son:

- Usualmente cada línea de comandos no debe superar la cantidad de 32 caracteres, pero esta longitud máxima puede variar de acuerdo al compilador. El signo '&' indica que el comando continúa en la siguiente línea.
- Al igual que en la mayoría de lenguajes, los espacios en blanco al inicio de una línea de comandos se ignoran, lo que es muy útil para estructurar el programa de forma ordenada.
- Los comentarios deben iniciar con el signo '!'.
- Varios comandos en una misma línea deben ser separados con ';'
- No se distingue entre mayúsculas y minúsculas.

Las constantes y variables del programa deben ser declaradas según su tipo de dato, como se muestra en la tabla 6.1.

Tabla 6.1. Tipos de datos en Fortran

`integer`	Números enteros positivos y negativos ej. `0, -1, 12`. Se almacenan en 4 bytes
`real`	Números positivos y negativos con decimales ej. `-0.8, 1.24, 1E-3`. Se almacena con 8 cifras significativas en 4 bytes.
`double, real(8)`	Reales de doble precisión, es decir que se almacenan en 8 bytes con 16 cifras significativas
`complex`	Números complejos cuya sintaxis es: `(parte_real,parte_imaginaria)` ej. `(0.0,-1.4), (1E-2,1E-1)`
`logical`	Solo puede tener 2 valores, `.true.` o `.false.`
`parameter`	Variable que se le asigna un valor en particular, ej. `pi=3.1416`
`character(N)`	Cadena de caracteres que debe ir entre comillas simples ej. `'hola mundo'`. `N` es el número de caracteres incluido espacios

A las variables que no han sido declaradas Fortran les asigna un tipo implícito de variable, cuya regla es: nombres que empiecen con i, j, k, l, m, n se asignan como variables enteras (`integer`), al resto se asigna como variables reales (`real`), el uso de estas variables implícitas a menudo causa errores de escritura en el código, por lo tanto, se debe escribir el comando `implicit none` al inicio del programa.

Cada código de programación escrito Fortran debe empezar con la palabra reservada `program` y escribirse en el siguiente orden:

```
program Nombre_del_programa
 implicit none
  ! Declaración de Variables
  ! Cuerpo del programa
  ! Sub-programas (si es requerido)
end program Nombre_del_programa
```

Cuando se trabaja con vectores o matrices se debe declarar sus dimensiones con el comando `dimension`, algunas formas de sintaxis son:

```
real,dimension(3,3)::matriz       !Matriz 3x3
real,dimension(:,:)::matriz(2,3)  !Matriz 2x3
real,dimension(10)::vector        !Vector de 10 elementos
real,dimension(:)::v1(5),v2(10)   !Dos vectores de 5 y 10 elementos
```

Pero si se pretende que las dimensiones se definan durante el programa entonces se debe hacer una declaración dinámica de variables con el comando `allocatable, ej.`

```
real,allocatable::arreglo(:)     !Crea el arreglo sin definir número de datos
```

Después, en cualquier parte del programa se debe asignar las dimensiones al arreglo una vez conocido su tamaño con el comando `allocate`, ej:

```
allocate(arreglo(100))           !Asigna la cantidad de datos del arreglo
deallocate(arreglo)              !Borra la dimensión del arreglo si es que
                                 !se desea volver a dimensionarlo
```

Las operaciones matemáticas siguen el siguiente orden:

1. Operaciones dentro paréntesis empezando por los que están más adentro
2. Exponenciación
3. Multiplicación y división de izquierda a derecha
4. Suma y resta de izquierda a derecha

Es recomendable emplear la multiplicación en lugar de la división siempre que sea posible, ya que multiplicar le toma menos tiempo a Fortran ej. Escribir `8.0*0.5` en lugar de `8.0/2.0`

En la tabla 6.2 se indican las operaciones y funciones matemáticas más empleadas en Fortran

A continuación, se muestra ejemplos de cómo emplear los comandos condicionales IF, DO y DO WHILE

IF:

```
if (condición_lógica_1) then
       ! programa1
else if (condición_lógica_2) then
       ! programa2
else
        ! programa3
end if
```

DO:

```
do i=inicio,fin,incremento
       !programa
end do
```

DO WHILE:

```
do while (condición_lógica)
        !programa
end do
```

Para leer y mostrar datos por pantalla se usan los comandos `read(type,type)` y `write(type,type)` respectivamente. Los argumentos `type` se emplean para cambiar el tipo de formato de entrada y salida de datos, pero la forma más sencilla de emplear estos comandos es usar el símbolo * que representa el formato estándar, ej:

```
read(*,*) datos         !Lectura de la variable datos

read*,datos             !Forma simplificada de read(*,*)

write(*,*) datos        !Imprime en pantalla los valores de la variable datos

print*,'Hola mundo'     !Forma simplificada de write(*,*)
```

Para la lectura y escritura de archivos se usan los comandos `open` y `close` como se muestra en el siguiente ejemplo:

```
!Lectura horizontal de datos:
open(n,file='archivo.txt',status='unknown')
read(n,*) (datos(i),i=1,n_datos)!La información se almacena en el vector datos
close(n)

!Lectura vertical de datos:
open(n,file='archivo.txt',status='unknown')
do i=1,n_datos
read(n,*) datos(i)!La información se almacena en este vector
end do
close(n)

!Escritura de datos:
open(m,file='nuevo_archivo.dat',status='unknown')
do i=1,n_datos !Por filas
    write(m,*) datos_columna1(i),datos_columna2(i) !Por columnas
end do
close(m)
```

En este ejemplo `n` y `m` indican dirección del fichero en el programa (pueden ser cualquier numero entero positivo), El archivo .txt o .dat debe estar en el mismo directorio del programa. En el argumento `status` se especifica el estado actual del archivo (unknown, new, old).

Otros dos comandos muy útiles son:

`call`: Para llamar subrutinas o ejecutar programas del sistema

`goto n`: Para saltarse a la dirección `n` del programa.

Tabla 6.2. Operaciones matemáticas básicas en Fortran

Operación	Ejemplo	Comando/Sintaxis
Suma	$a + b$	`a+b`
Resta	$a - b$	`a-b`
Multiplicación	$a \times b$	`a*c`
División	$a \div b$	`a/b`
Exponenciación	x^n	`x**n`
Raíz cuadrada	$\sqrt[2]{x}$	`sqrt(x)`
Exponencial	e^x	`exp(x)`
Logaritmo neperiano	$\ln x$	`log(x)`
Logaritmo decimal	$\log_{10} x$	`log10(x)`
Seno	$\sin x$	`sin(x)`
Coseno	$\cos x$	`cos(x)`
Tangente	$\tan x$	`tan(x)`
Arco-seno	$\sin^{-1} x$	`asin(x)`
Arco-coseno	$\cos^{-1} x$	`acos(x)`
Arco-tangente	$\tan^{-1} x$	`atan(x)`
Valor absoluto	$\|x\|$, $\|a + jb\|$	`abs(x), abs((a,b))`
Valor máximo	Máximo entre a y b	`max(a,b)`
Valor mínimo	Mínimo entre a y b	`min(a,b)`
Igual que	$a = b$	`a==b, a.eq.b`
No igual que	$a \neq b$	`a/=b, a.ne.b`
Mayor que	$a > b$	`a>b , a.gt.b`
Menor que	$a < b$	`a<b , a.lt.b`
Mayor o igual que	$a \geq b$	`a>=b, a.ge.b`
Menor o igual que	$a \leq b$	`a<=b, a.le.b`
Disyunción lógica	$A \vee B$	`A.or.B`
Conjunción lógica	$A \wedge B$	`A.and.B`
Producto de matrices	$[A] \times [B]$	`matmul(A,B)`

ANEXO II

Modelamiento del sistema en DIgSILENT

Para crear el frame del sistema hay que ubicarse en la sección *User Defined* Models de la librería del proyecto, dar click derecho y seleccionar *New - Block/Frame Diagram* como se muestra en la figura 6.1, se escribe el nombre, luego *ok* y se abre una sección donde se debe colocar los slots del generador y reguladores (figura 6.2).

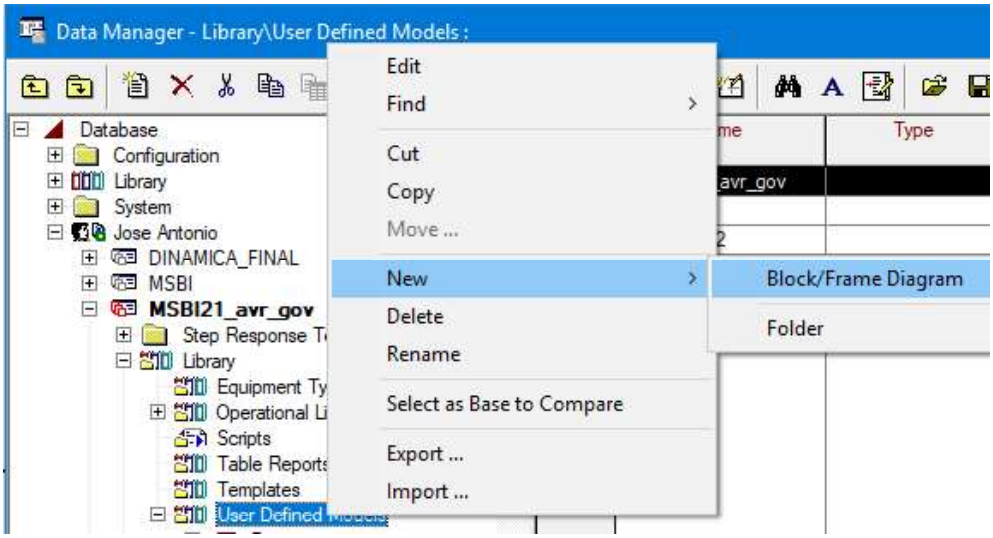


Figura 6.1. Creación del frame

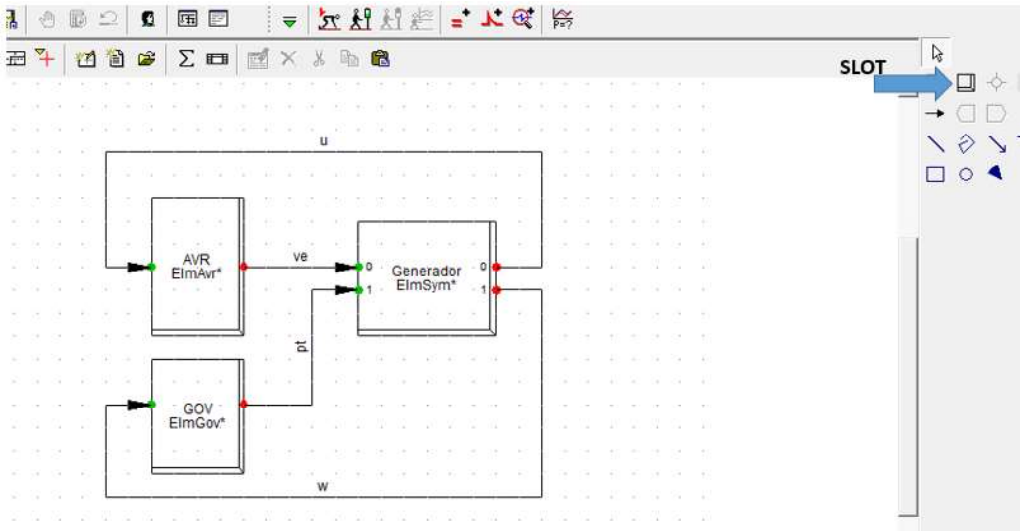


Figura 6.2. Diseño del frame

A continuación, se muestra la configuración de cada slot

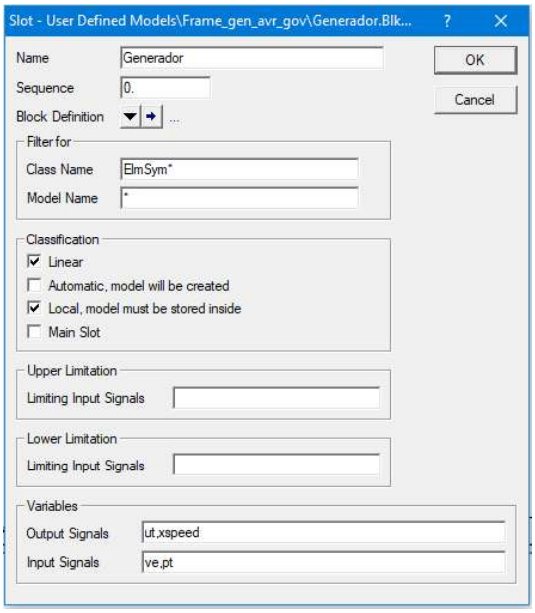


Figura 6.3. Configuración slot generador

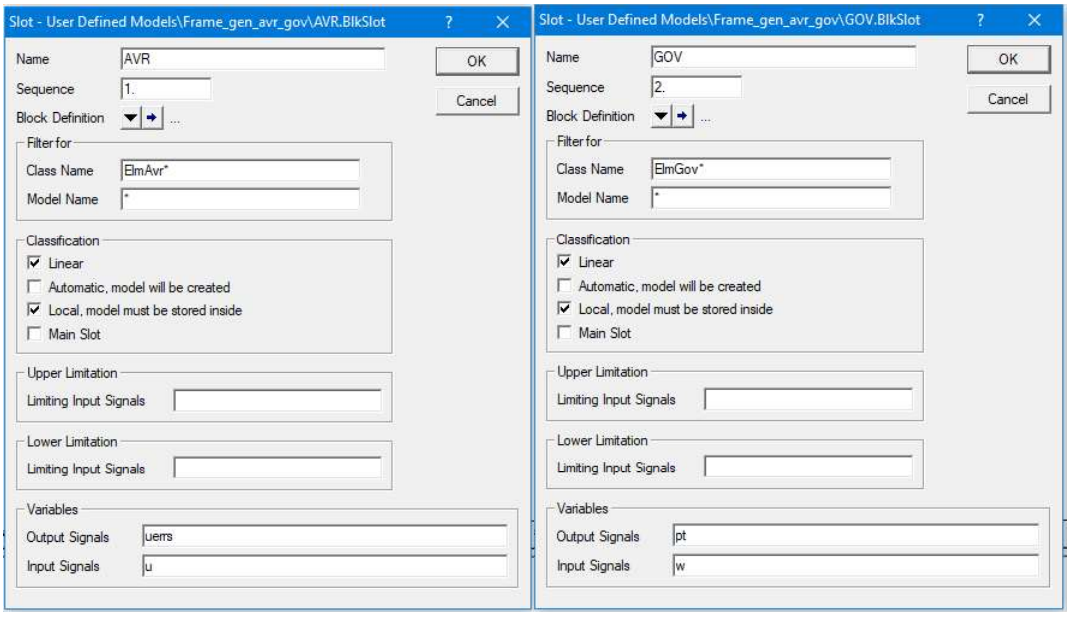


Figura 6.4. Configuración slot reguladores

Se copian los modelos de los reguladores desde la librería general a la librería del proyecto en la sección *User Defined Models*.

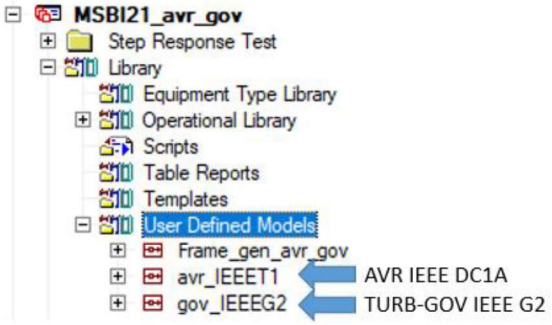


Figura 6.5. Selección de los reguladores

En los datos de la red se crea el *Composite Model* seleccionando la opción *New Object*, luego se escoge *ElmComp* en la lista de opciones (figura 6.6), click en *ok*, se le da un nombre (ej. CM generador) y en la opción *Select Project Type* se selecciona el Frame previamente creado en la biblioteca del usuario (figura 6.7)

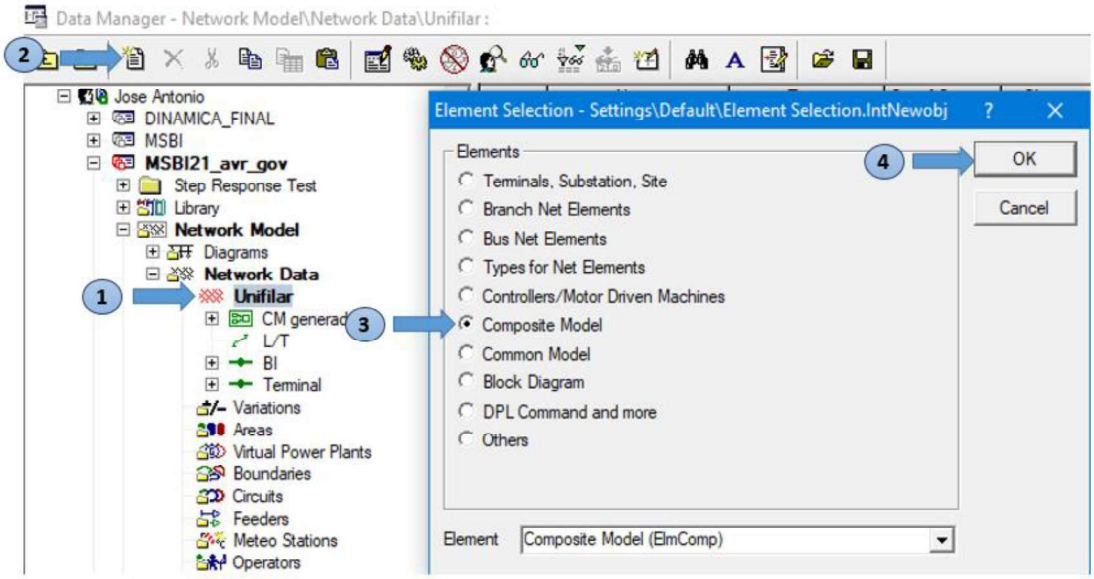


Figura 6.6. Creación del Composite Model

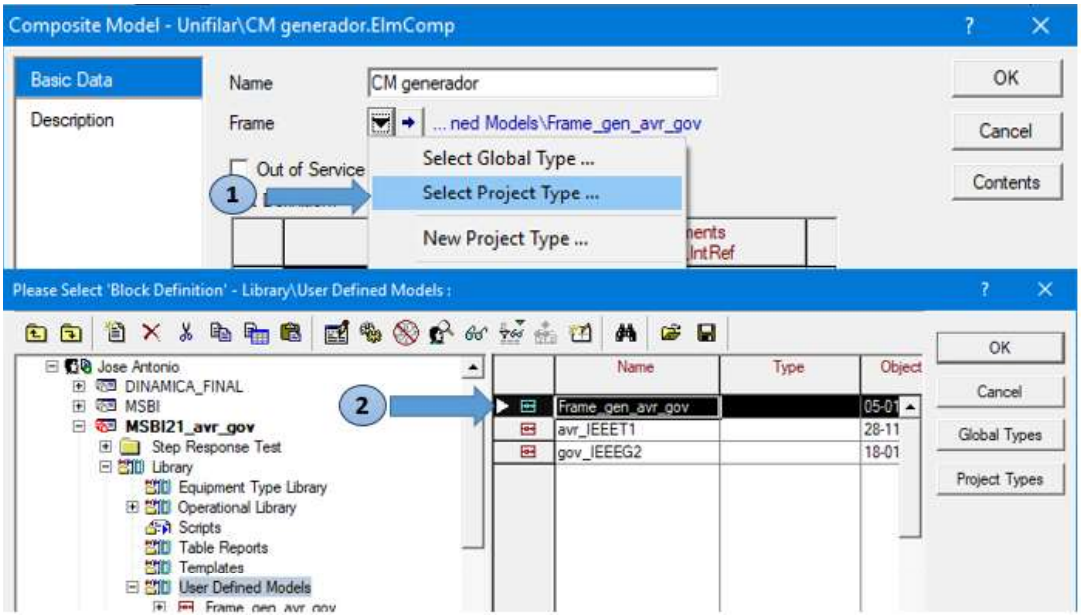


Figura 6.7. Asignación del Frame al Composite Model

En la sección del composite model creado van los *DSL* de los reguladores (*Common Model*), para lo cual se realiza el proceso de la figura 6.8, después se debe seleccionar el correspondiente regulador que debe estar en la biblioteca del usuario (figura 6.9)

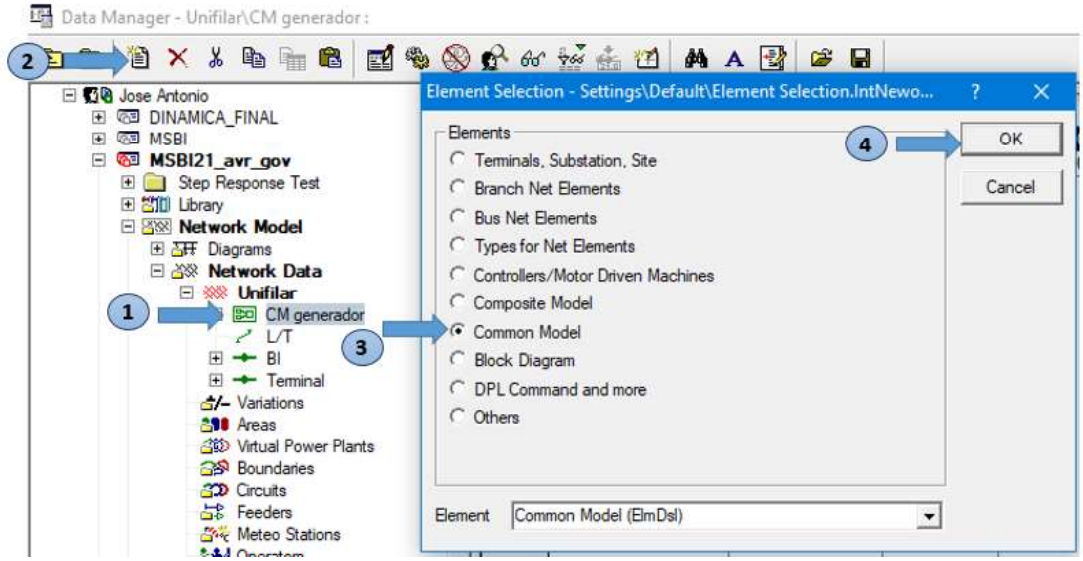


Figura 6.8. Creación de los DSL de los reguladores

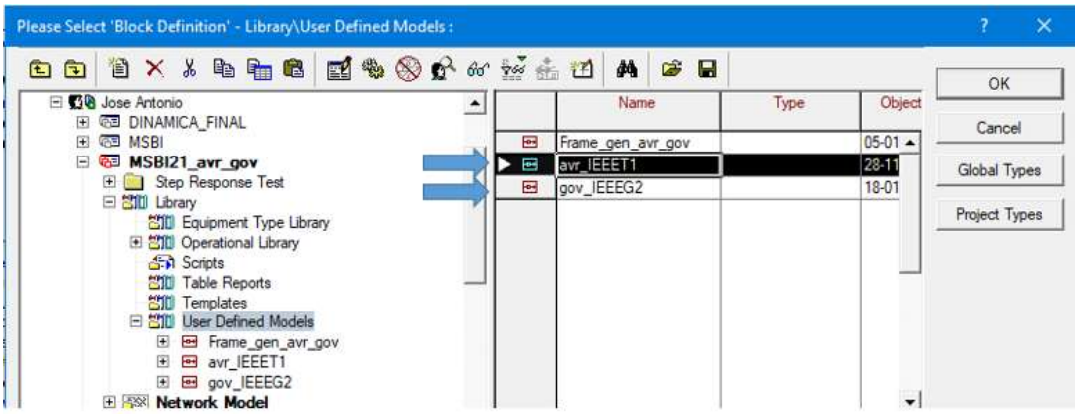


Figura 6.9. Asignación de los DSL de cada regulador

En estos DSL se puede modificar los parámetros de los reguladores si se desea

Finalmente se vuelve a abrir en Composite Model (CM generador) para establecer el generador y los dsl de los reguladores que actúan en la simulación

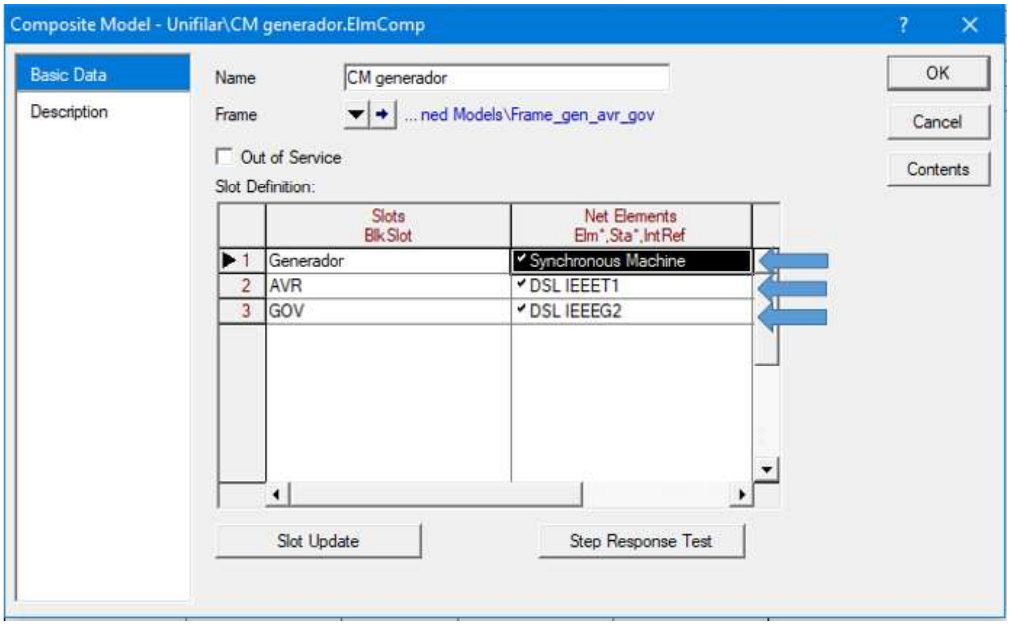


Figura 6.10. Asignación del generador y reguladores al composite model

Para realizar la simulación de eventos transitorios se debe escoger la opción *RMS/EMT Simulation* en el toolbox, se cran los eventos (figura 6.11) y las gráficas que se deseen observar.

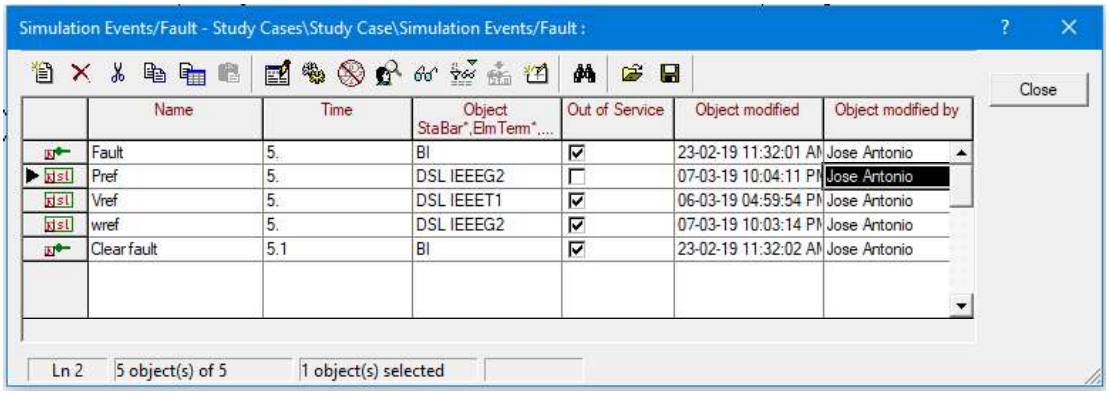


Figura 6.11. Eventos a simular

ANEXO III

Código del Programa

```
!**************************************************************************
!
!  PROGRAM: MSBI_avr_gov_2_1
!
!  PURPOSE:     Simulación de un generador sincrónico 2.1 conectado a una barra
!               infinita con AVR y GOV.
!
!               Tesis de Grado
!               Realizado por: José Carrillo Nieto
!               Director: Ing. Juan Plazarte
!               Codirector: Dr. Hugo Arcos
!**************************************************************************

    program MSBI_avr_gov_2_1

    implicit none

     ! Variables
real psif0,psikd0,psig0,psikq0,wg0,delta0,Vc0,Vr0,Efd0,Vs0,P10,P20,Tm0  !Condiciones iniciales
integer Ndatos,i,evento                                                 !Variable para control de
simulacion
real,allocatable::t(:),Eo(:),vref(:),wref(:),pref(:)                    !Eventos
character(17)::nevento                                                  !Nombre del evento
real,allocatable::Ymat11(:),Ymat12(:),Ymat21(:),Ymat22(:)               !Matriz de impedancia
real tstop,tclear,tpaso,nuevo                                           ! "
real,dimension(19)::datosSM                                             !Parámetros de la máquina
real,dimension(12)::datosAVR                                            !Parámetros del AVR
real,dimension(7)::datosGOV                                             !Parámetros del GOV
real xd,xdp,xdpp,Td0p,Td0pp,xq,xqp,xqpp,Tq0p,Tq0pp,H,D,Ra,frec,wb       ! "
real a,b,c,Tdpp1,Tdpp2,Tdpp,Tdp,Tqpp1,Tqpp2,Tqpp,Tqp,tfp,fd,fefd,ft     ! "
real Pgen,Sbase,Sn,fp,fpg,Pg0,vt0,Vl,xL,xL2,theta0,thetaL,Qg0,Eq0       ! "
complex Sgv,Vt0v,It0v,Eq0v,It0dqv                                       ! "
real iq0,id0,psid0,psiq0,Te0,w0,vref0,wref0,pref0                       ! "
real Edppx,Eqppx,idx,iqx,psidx,psiqx,Temx,Vtx,Vx                        ! "
real,allocatable::Edpp(:),Eqpp(:),id(:),iq(:),psid(:),psiq(:)           ! "
real,allocatable::psif(:),psikd(:),psig(:),psikq(:)                     ! "
real,allocatable::wg(:),delta(:),Efd(:),Tem(:)                          ! "
real,allocatable::Vt(:),It(:),Pg(:),Qg(:),Torque(:)                     ! "
real KA,TA,Vmax,Vmin,TR,TE,KF,TF,E1,E2,Se1,Se2,Bex,Aex                  !Parámetros AVR DC1A
real K,T1,T2,T3,T4,Pmax,Pmin                                            !Parámetros GOV IEEEG2
real,allocatable::f1(:),f2(:),f3(:),f4(:),f5(:),f6(:),f7(:)             !Sistema de EDO
real,allocatable::f8(:),f9(:),f10(:),f11(:),f12(:),Dp(:),U(:)           ! "
real::x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12                            ! "
real,parameter::e=0.005                                 !Precisión del algoritmo Runge-Kutta
real,dimension(12)::F,k1,k2,k3,k4                       !Variables del algoritmo Runge-Kutta 4to orden
real,parameter::pi=4.0*atan(1.0)                                        !Número pi
complex,parameter::j=(0.0,1.0)                                          !Asignación del imaginario j

    ! Cuerpo del programa
!Lectura parámetros de la maquina sincrónica
open(100,file='datosMS.dat',status='old')
read(100,*)
read(100,*) (datosSM(i),i=1,19)
```

```
close(100)
!Eje directo
xd=datosSM(1)
xdp=datosSM(2)
xdpp=datosSM(3)
Td0p=datosSM(4)
Td0pp=datosSM(5)

!Eje en cuadratura
xq=datosSM(6)
xqpp=datosSM(7)
Tq0pp=datosSM(8)

H=datosSM(9)
D=datosSM(10)
Ra=datosSM(11)
frec=datosSM(12)

wb=2.0*pi*frec

!Cálculo de Td' y Td":
a=(1.0-xd/xdp+xd/xdpp)
b=-(Td0p+Td0pp)
c=(xdpp/xdp)*Td0p*Td0pp
Tdpp1=(-b+sqrt(b**2-4.0*a*c))/(2.0*a)
Tdpp2=(-b-sqrt(b**2-4.0*a*c))/(2.0*a)
Tdpp=min(Tdpp1,Tdpp2)
Tdp=Td0p*Td0pp*(xdpp/xd)/Tdpp

!Cálculo de Tq":
Tqpp=Tq0pp*(xqpp/xq)

!Cálculo de condiciones iniciales:
Sn=datosSM(13)
fpg=datosSM(14)
Pgen=datosSM(15)
fp=datosSM(16)
tfp=datosSM(17)
VL=datosSM(18)
xL=datosSM(19)
Sbase=100.0 !MVA
Pg0=Pgen/Sbase

theta0=acos(fp)
if (tfp>0) then
Qg0=Pg0*tan(theta0)
    else
     Qg0=-Pg0*tan(theta0)
end if
Sgv=Pg0+j*Qg0

Vt0=sqrt(0.5*(VL**2+2.0*Qg0*xL+sqrt(VL**4+4.0*Qg0*XL*VL**2-4.0*(xL**2)*(Pg0**2))))
thetaL=asin(Pg0*xL/(Vt0*VL))
Vt0v=Vt0*(cos(thetaL)+j*sin(thetaL))
It0v=conjg(Sgv/Vt0v)

Eq0v=Vt0v+(Ra+j*xq)*It0v
delta0=atan(imag(Eq0v)/real(Eq0v))
```

```
Eq0=abs(Eq0v)

It0dqv=It0v*(cos(delta0)-j*sin(delta0))

iq0=real(It0dqv)
id0=imag(It0dqv)

Efd0=Eq0-(xd-xq)*id0
psid0=xd*id0+Efd0
psiq0=xq*iq0
psif0=psid0+(xdp/(xd-xdp))*Efd0
psikd0=psid0
psig0=psiq0

Te0=psid0*iq0-psiq0*id0
Tm0=Te0

w0=wb
wg0=w0/wb

!-----------------AVR------------------
!Lectura de parámetros
open(200,file='datosAVR.dat',status='old')
read(200,*)
read(200,*) (datosAVR(i),i=1,12)
close(200)
TR=datosAVR(1)
KA=datosAVR(2)
TA=datosAVR(3)
Vmax=datosAVR(4)
Vmin=datosAVR(5)
TE=datosAVR(6)
KF=datosAVR(7)
TF=datosAVR(8)

!Saturation Se
E1=datosAVR(9)
Se1=datosAVR(10)
E2=datosAVR(11)
Se2=datosAVR(12)
Bex=1.0/(E2-E1)*log(Se2/Se1)
Aex=Se1*exp(-Bex*E1)

Vref0=(Efd0*Aex*exp(Bex*Efd0))/KA+Vt0
Vc0=Vt0
Vr0=Efd0*Aex*exp(Bex*Efd0)
Vs0=0.0
!---------------------------------------
!---Hydraulic Governor GOV -------------
!Lectura de parámetros
open(300,file='datosGOV.dat',status='old')
read(300,*)
read(300,*) (datosGOV(i),i=1,7)
close(300)
K=datosGOV(1)
T1=datosGOV(2)
T2=datosGOV(3)
T3=datosGOV(4)
```

```
T4=datosGOV(5)
Pmax=datosGOV(6)
Pmin=datosGOV(7)

P10=0.0
P20=0.0

wref0=wg0
pref0=Tm0
!------Carátula del programa---------------------------------
print*,achar(201),repeat(achar(205),63),achar(187)
print*,achar(186),'   E S C U E L A    P O L I T E C N I C A    N A C I O N A L   ',achar(186)
print*,achar(186),'   ____________________________________________________________   ',achar(186)
print*,achar(186),'                                                                  ',achar(186)
print*,achar(186),'       SIMULACION DE UNA UNIDAD DE GENERACION HIDRAULICA       ',achar(186)
print*,achar(186),'                                                                  ',achar(186)
print*,achar(186),'SISTEMA: Generador - Barra Infinita                              ',achar(186)
print*,achar(186),'REGULADOR DE VOLTAJE: AVR IEEE DC1A                              ',achar(186)
print*,achar(186),'REGULADOR DE VELOCIDAD: TURB-GOV IEEE G2                         ',achar(186)
print*,achar(200),repeat(achar(205),63),achar(188)
print*,''
print*,''
print*,repeat(' ',18),achar(218),repeat(achar(196),28),achar(191)
print*,repeat(' ',18),achar(179),'Presine ENTER para continuar',achar(179)
print*,repeat(' ',18),achar(192),repeat(achar(196),28),achar(217)
read(*,*)
call system('CLS')
!Lectura del tiempo de simulación
write (*,'(''Ingrese tiempo de simulacion en segundos: '',\)')
read*,tstop
Ndatos=nint(tstop/e) !Para calcular la longitud de los datos

!Asiganción del tamaño de los datos
allocate(t(Ndatos+1),Eo(Ndatos+1),Vref(Ndatos+1),wref(Ndatos+1),pref(Ndatos+1))
allocate(Ymat11(Ndatos+1),Ymat12(Ndatos+1),Ymat21(Ndatos+1),Ymat22(Ndatos+1))
allocate(f1(Ndatos+1),f2(Ndatos+1),f3(Ndatos+1),f4(Ndatos+1),f5(Ndatos+1),f6(Ndatos+1),f7(Ndatos+1))
allocate(f8(Ndatos+1),f9(Ndatos+1),f10(Ndatos+1),f11(Ndatos+1),f12(Ndatos+1),Dp(Ndatos+1),U(Ndatos+1))
allocate(Edpp(Ndatos+1),Eqpp(Ndatos+1),id(Ndatos+1),iq(Ndatos+1),psid(Ndatos+1),psiq(Ndatos+1))
allocate(psif(Ndatos+1),psikd(Ndatos+1),psig(Ndatos+1),psikq(Ndatos+1))
allocate(wg(Ndatos+1),delta(Ndatos+1),Efd(Ndatos+1),Tem(Ndatos+1))
allocate(Vt(Ndatos+1),It(Ndatos+1),Pg(Ndatos+1),Qg(Ndatos+1),Torque(Ndatos+1))

t=(/(i,i=0,Ndatos)/)*e !Creación del vector tiempo

!Valores iniciales de los parametros que cambian
Eo(1)=1.0
Vref(1)=Vref0
wref(1)=wref0
pref(1)=pref0
U(1)=0.0

!Matriz de impedancia del estator inicial
Ymat11(1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
Ymat12(1)=(-xqpp-xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
Ymat21(1)=(xdpp+xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
Ymat22(1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))

!Selección de eventos
```

```
10 print*,''
print*,'EVENTOS:'
print*,'1. Falla trifasica'
print*,'2. Cambio de Voltage Set Point (Vref avr)'
print*,'3. Cambio en W0 (Wref gov)'
print*,'4. Cambio en Power Set Point (Pref gov)'
print*,'5. Salir'
print*,''
write (*,'(''Ingrese evento: '',\)')
read*,evento
if (evento==1)then
    nevento='FALLA TRIFASICA'
    goto 11
else if (evento==2)then
    nevento='VARIACION EN Vref'
    goto 12
else if (evento==3)then
    nevento='VARIACION EN Wref'
    goto 13
else if (evento==4)then
    nevento='VARIACION EN Pref'
    goto 14
else if (evento==5)then
    goto 15
else
    print*,' *** OPCION INCORRECTA ***'
    goto 10
endif

!--------------------Falla 3f----------------------
11 call system('CLS')
print*,repeat(' ',15),nevento
print*,repeat(achar(196),45)
print*,'Ubicacion de la falla:'
print*,'1. Terminales del generador'
print*,'2. Barra infinita'
print*,'3. Linea de transmision (50%)'
print*,''
write (*,'(''Ingrese eleccion: '',\)')
read*,nuevo
if (nuevo==1) then
    xL2=0.0
else if (nuevo==2) then
    xL2=xL
else if (nuevo==3) then
    xL2=xL/2.0
    else
    print*,' *** OPCION INCORRECTA ***'
    goto 11
end if
write (*,'(''Ingrese tiempo en que ocurre la falla: '',\)')
read*,tpaso
write (*,'(''Ingrese tiempo que dura la falla: '',\)')
read*,tclear
    do i=1,Ndatos
        if (t(i+1)>tpaso .and. t(i+1)<(tpaso+tclear)) then
        Eo(i+1)=0.0
        Ymat11(i+1)=Ra/(Ra**2+(xqpp+xL2)*(xdpp+xL2))
```

```
        Ymat12(i+1)=(-xqpp-xL2)/(Ra**2+(xqpp+xL2)*(xdpp+xL2))
        Ymat21(i+1)=(xdpp+xL2)/(Ra**2+(xqpp+xL2)*(xdpp+xL2))
        Ymat22(i+1)=Ra/(Ra**2+(xqpp+xL2)*(xdpp+xL2))
        else
            Eo(i+1)=1.0
            Ymat11(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
            Ymat12(i+1)=(-xqpp-xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
            Ymat21(i+1)=(xdpp+xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
            Ymat22(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
        end if
    end do
do i=1,Ndatos
    Vref(i+1)=Vref0
    wref(i+1)=wref0
    pref(i+1)=pref0
    U(i+1)=0.0
end do
goto 16

!--------------------Cambio Vsetp----------------------
12 call system('CLS')
print*,repeat(' ',15),nevento
print*,repeat(achar(196),45)
write (*,'(''Ingrese tiempo en que ocurre el evento: '',\)')
read*,tpaso
print*,achar(201),repeat(achar(205),34),achar(187)
print*,achar(186),'Vref set point:',Vref0,'pu ',achar(186)
print*,achar(200),repeat(achar(205),34),achar(188)
print*,''
write (*,'(''Ingrese el nuevo valor de Vref: '',\)')
read*,nuevo
    do i=1,Ndatos
        if (t(i+1)>tpaso) then
        Vref(i+1)=nuevo
        else
            Vref(i+1)=Vref0
        end if
    end do
do i=1,Ndatos
    Eo(i+1)=1.0
    wref(i+1)=wref0
    pref(i+1)=pref0
    U(i+1)=0.0
    !Matriz de impedancia del estator
    Ymat11(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat12(i+1)=(-xqpp-xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat21(i+1)=(xdpp+xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat22(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
end do
goto 16
!--------------------Cambio Wref----------------------
13 call system('CLS')
print*,repeat(' ',15),nevento
print*,repeat(achar(196),45)
write (*,'(''Ingrese tiempo en que ocurre el evento: '',\)')
read*,tpaso
print*,achar(201),repeat(achar(205),34),achar(187)
print*,achar(186),'Wref set point:',Wref0,'pu ',achar(186)
```

```
print*,achar(200),repeat(achar(205),34),achar(188)
print*,''
write (*,'(''Ingrese el nuevo valor de Wref: '',\)')
read*,nuevo
    do i=1,Ndatos
        if (t(i+1)>tpaso) then
        wref(i+1)=nuevo
        else
            wref(i+1)=wref0
        end if
    end do
do i=1,Ndatos
    Eo(i+1)=1.0
    Vref(i+1)=Vref0
    pref(i+1)=pref0
    U(i+1)=0.0

    Ymat11(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat12(i+1)=(-xqpp-xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat21(i+1)=(xdpp+xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat22(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
end do
goto 16
!--------------------Cambio Pref----------------------
14 call system('CLS')
print*,repeat(' ',15),nevento
print*,repeat(achar(196),45)
write (*,'(''Ingrese tiempo en que ocurre el evento: '',\)')
read*,tpaso
print*,achar(201),repeat(achar(205),34),achar(187)
print*,achar(186),'Pref set point:',Pref0,'pu ',achar(186)
print*,achar(200),repeat(achar(205),34),achar(188)
print*,''
write (*,'(''Ingrese el nuevo valor de Pref: '',\)')
read*,nuevo
    do i=1,Ndatos
        if (t(i+1)>tpaso) then
        pref(i+1)=nuevo
        else
            pref(i+1)=pref0
        end if
        if (t(i+1)==tpaso) then
            U(i+1)=(nuevo-Pref0)*Sbase
        else
            U(i+1)=0.0
            end if
    end do
do i=1,Ndatos
    Eo(i+1)=1.0
    Vref(i+1)=Vref0
    wref(i+1)=wref0

    Ymat11(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat12(i+1)=(-xqpp-xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat21(i+1)=(xdpp+xL)/(Ra**2+(xqpp+xL)*(xdpp+xL))
    Ymat22(i+1)=Ra/(Ra**2+(xqpp+xL)*(xdpp+xL))
end do
```

```
!Inicio Runge-Kutta cuarto orden
!condiciones iniciales
16 f1(1)=psif0
f2(1)=psikd0
f3(1)=psig0
f4(1)=wg0
f5(1)=delta0
f6(1)=Vc0
f7(1)=Vr0
f8(1)=Efd0
f9(1)=Vs0
f10(1)=P10
f11(1)=P20
f12(1)=Tm0
Dp(1)=pref(1)-f11(1)

do i=1,Ndatos

    !Cálculo del factor k1
    x1=f1(i)
    x2=f2(i)
    x3=f3(i)
    x4=f4(i)
    x5=f5(i)
    x6=f6(i)
    x7=f7(i)
    x8=f8(i)
    x9=f9(i)
    x10=f10(i)
    x11=f11(i)
    x12=f12(i)

    !Ecuaciones de la máquina sincrónica (se repite para cada factor k):
    Eqppx=x2*((xdp-xdpp)/xdp)+x1*((xd-xdp)/xd)*xdpp/xdp
    Edppx=-x3*((xq-xqpp)/xq)
    idx=Ymat11(i)*(Edppx+Eo(i)*sin(x5))+Ymat12(i)*(Eqppx-Eo(i)*cos(x5))
    iqx=Ymat21(i)*(Edppx+Eo(i)*sin(x5))+Ymat22(i)*(Eqppx-Eo(i)*cos(x5))
    psidx=Eqppx+xdpp*idx
    psiqx=-Edppx+xqpp*iqx
    Temx=iqx*psidx-idx*psiqx
    Vtx=sqrt((psidx-Ra*iqx)**2+(-psiqx-Ra*idx)**2)
    Vx=x8*(Aex*exp(Bex*x8))

    !Sistema de ecuaciones diferenciales (se repite para cada factor k):
    F(1)=(1.0/Tdp)*(psidx+x8*(xdp/(xd-xdp))-x1)
    F(2)=(1.0/Tdpp)*(psidx-x2)
    F(3)=(1.0/Tqpp)*(psiqx-x3)
    F(4)=(1.0/(2.0*H))*(x12-Temx-D*x4)
    F(5)=(x4-1.0)*wb
    F(6)=(1.0/TR)*(Vtx-x6)
    F(7)=(1.0/TA)*(KA*(Vref(i)-x6-x9)-x7)
    F(8)=(1.0/TE)*(x7-Vx)
    F(9)=(KF/TF)*(x7-Vx)/TE-x9/TF
    F(10)=(1.0/T1)*(K*(wref(i)-x4)-x10)
    F(11)=(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10)))
    F(12)=2.0*((Dp(i)-x12)/T4-U(i)+(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10))))

    K1=e*F
```

```
!Cálculo del factor  k2
x1=f1(i)+K1(1)/2.0
x2=f2(i)+K1(2)/2.0
x3=f3(i)+K1(3)/2.0
x4=f4(i)+K1(4)/2.0
x5=f5(i)+K1(5)/2.0
x6=f6(i)+K1(6)/2.0
x7=f7(i)+K1(7)/2.0
x8=f8(i)+K1(8)/2.0
x9=f9(i)+K1(9)/2.0
x10=f10(i)+K1(10)/2.0
x11=f11(i)+K1(11)/2.0
x12=f12(i)+K1(12)/2.0

Eqppx=x2*((xdp-xdpp)/xdp)+x1*((xd-xdp)/xd)*xdpp/xdp
Edppx=-x3*((xq-xqpp)/xq)
idx=Ymat11(i)*(Edppx+Eo(i)*sin(x5))+Ymat12(i)*(Eqppx-Eo(i)*cos(x5))
iqx=Ymat21(i)*(Edppx+Eo(i)*sin(x5))+Ymat22(i)*(Eqppx-Eo(i)*cos(x5))
psidx=Eqppx+xdpp*idx
psiqx=-Edppx+xqpp*iqx
Temx=iqx*psidx-idx*psiqx
Vtx=sqrt((psidx-Ra*iqx)**2+(-psiqx-Ra*idx)**2)
Vx=x8*(Aex*exp(Bex*x8))

F(1)=(1.0/Tdp)*(psidx+x8*(xdp/(xd-xdp))-x1)
F(2)=(1.0/Tdpp)*(psidx-x2)
F(3)=(1.0/Tqpp)*(psiqx-x3)
F(4)=(1.0/(2.0*H))*(x12-Temx-D*x4)
F(5)=(x4-1.0)*wb
F(6)=(1.0/TR)*(Vtx-x6)
F(7)=(1.0/TA)*(KA*(Vref(i)-x6-x9)-x7)
F(8)=(1.0/TE)*(x7-Vx)
F(9)=(KF/TF)*(x7-Vx)/TE-x9/TF
F(10)=(1.0/T1)*(K*(wref(i)-x4)-x10)
F(11)=(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10)))
F(12)=2.0*((Dp(i)-x12)/T4-U(i)+(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10))))

K2=e*F

!Cálculo del factor k3
x1=f1(i)+K2(1)/2.0
x2=f2(i)+K2(2)/2.0
x3=f3(i)+K2(3)/2.0
x4=f4(i)+K2(4)/2.0
x5=f5(i)+K2(5)/2.0
x6=f6(i)+K2(6)/2.0
x7=f7(i)+K2(7)/2.0
x8=f8(i)+K2(8)/2.0
x9=f9(i)+K2(9)/2.0
x10=f10(i)+K2(10)/2.0
x11=f11(i)+K2(11)/2.0
x12=f12(i)+K2(12)/2.0

Eqppx=x2*((xdp-xdpp)/xdp)+x1*((xd-xdp)/xd)*xdpp/xdp
Edppx=-x3*((xq-xqpp)/xq)
idx=Ymat11(i)*(Edppx+Eo(i)*sin(x5))+Ymat12(i)*(Eqppx-Eo(i)*cos(x5))
iqx=Ymat21(i)*(Edppx+Eo(i)*sin(x5))+Ymat22(i)*(Eqppx-Eo(i)*cos(x5))
```

```
    psidx=Eqppx+xdpp*idx
    psiqx=-Edppx+xqpp*iqx
    Temx=iqx*psidx-idx*psiqx
    Vtx=sqrt((psidx-Ra*iqx)**2+(-psiqx-Ra*idx)**2)
    Vx=x8*(Aex*exp(Bex*x8))

    F(1)=(1.0/Tdp)*(psidx+x8*(xdp/(xd-xdp))-x1)
    F(2)=(1.0/Tdpp)*(psidx-x2)
    F(3)=(1.0/Tqpp)*(psiqx-x3)
    F(4)=(1.0/(2.0*H))*(x12-Temx-D*x4)
    F(5)=(x4-1.0)*wb
    F(6)=(1.0/TR)*(Vtx-x6)
    F(7)=(1.0/TA)*(KA*(Vref(i)-x6-x9)-x7)
    F(8)=(1.0/TE)*(x7-Vx)
    F(9)=(KF/TF)*(x7-Vx)/TE-x9/TF
    F(10)=(1.0/T1)*(K*(wref(i)-x4)-x10)
    F(11)=(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10)))
    F(12)=2.0*((Dp(i)-x12)/T4-U(i)+(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10))))

    K3=e*F

    !Cálculo del factor k4
    x1=f1(i)+K3(1)
    x2=f2(i)+K3(2)
    x3=f3(i)+K3(3)
    x4=f4(i)+K3(4)
    x5=f5(i)+K3(5)
    x6=f6(i)+K3(6)
    x7=f7(i)+K3(7)
    x8=f8(i)+K3(8)
    x9=f9(i)+K3(9)
    x10=f10(i)+K3(10)
    x11=f11(i)+K3(11)
    x12=f12(i)+K3(12)

    Eqppx=x2*((xdp-xdpp)/xdp)+x1*((xd-xdp)/xd)*xdpp/xdp
    Edppx=-x3*((xq-xqpp)/xq)
    idx=Ymat11(i)*(Edppx+Eo(i)*sin(x5))+Ymat12(i)*(Eqppx-Eo(i)*cos(x5))
    iqx=Ymat21(i)*(Edppx+Eo(i)*sin(x5))+Ymat22(i)*(Eqppx-Eo(i)*cos(x5))
    psidx=Eqppx+xdpp*idx
    psiqx=-Edppx+xqpp*iqx
    Temx=iqx*psidx-idx*psiqx
    Vtx=sqrt((psidx-Ra*iqx)**2+(-psiqx-Ra*idx)**2)
    Vx=x8*(Aex*exp(Bex*x8))

    F(1)=(1.0/Tdp)*(psidx+x8*(xdp/(xd-xdp))-x1)
    F(2)=(1.0/Tdpp)*(psidx-x2)
    F(3)=(1.0/Tqpp)*(psiqx-x3)
    F(4)=(1.0/(2.0*H))*(x12-Temx-D*x4)
    F(5)=(x4-1.0)*wb
    F(6)=(1.0/TR)*(Vtx-x6)
    F(7)=(1.0/TA)*(KA*(Vref(i)-x6-x9)-x7)
    F(8)=(1.0/TE)*(x7-Vx)
    F(9)=(KF/TF)*(x7-Vx)/TE-x9/TF
    F(10)=(1.0/T1)*(K*(wref(i)-x4)-x10)
    F(11)=(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10)))
    F(12)=2.0*((Dp(i)-x12)/T4-U(i)+(1.0/T3)*(x10-x11+T2*((1.0/T1)*(K*(wref(i)-x4)-x10))))
```

```
    K4=e*F

    !Solución del algoritmo RK4-------------------------------------
    f1(i+1)=f1(i)+(1.0/6.0)*(K1(1)+2.0*K2(1)+2.0*K3(1)+K4(1))
    f2(i+1)=f2(i)+(1.0/6.0)*(K1(2)+2.0*K2(2)+2.0*K3(2)+K4(2))
    f3(i+1)=f3(i)+(1.0/6.0)*(K1(3)+2.0*K2(3)+2.0*K3(3)+K4(3))
    f4(i+1)=f4(i)+(1.0/6.0)*(K1(4)+2.0*K2(4)+2.0*K3(4)+K4(4))
    f5(i+1)=f5(i)+(1.0/6.0)*(K1(5)+2.0*K2(5)+2.0*K3(5)+K4(5))
    f6(i+1)=f6(i)+(1.0/6.0)*(K1(6)+2.0*K2(6)+2.0*K3(6)+K4(6))
    f7(i+1)=f7(i)+(1.0/6.0)*(K1(7)+2.0*K2(7)+2.0*K3(7)+K4(7))
    f8(i+1)=f8(i)+(1.0/6.0)*(K1(8)+2.0*K2(8)+2.0*K3(8)+K4(8))
    f9(i+1)=f9(i)+(1.0/6.0)*(K1(9)+2.0*K2(9)+2.0*K3(9)+K4(9))
    f10(i+1)=f10(i)+(1.0/6.0)*(K1(10)+2.0*K2(10)+2.0*K3(10)+K4(10))
    f11(i+1)=f11(i)+(1.0/6.0)*(K1(11)+2.0*K2(11)+2.0*K3(11)+K4(11))
    f12(i+1)=f12(i)+(1.0/6.0)*(K1(12)+2.0*K2(12)+2.0*K3(12)+K4(12))
    Dp(i+1)=pref(i+1)-f11(i+1)

    !----Saturación AVR----
    if (f7(i+1)<Vmin) then
        f7(i+1)=Vmin
        else if (f7(i+1)>=Vmax) then
                f7(i+1)=Vmax
        end if

    !----Saturación GOV----
    if (Dp(i+1)<Pmin) then
        Dp(i+1)=Pmin
        else if (Dp(i+1)>=Pmax) then
                Dp(i+1)=Pmax
        end if

end do

!Señales de salida:
psif=f1
psikd=f2
psig=f3
wg=f4
delta=f5
Efd=f8

Eqpp=((xdp-xdpp)/xdp)*psikd+(((xd-xdp)/xd)*xdpp/xdp)*psif
Edpp=-((xq-xqpp)/xq)*psig
id=Ymat11*(Edpp+Eo*sin(delta))+Ymat12*(Eqpp-Eo*cos(delta))
iq=Ymat21*(Edpp+Eo*sin(delta))+Ymat22*(Eqpp-Eo*cos(delta))
psid=Eqpp+xdpp*id
psiq=-Edpp+xqpp*iq
Tem=iq*psid-id*psiq

!Cambio de base del frame del generador:
It0v=conjg((Sgv/Vt0v)*(Sbase/Sn))
Eq0v=Vt0v+(Ra+j*xq)*It0v
fd=atan(imag(Eq0v)/real(Eq0v))/delta0
It0dqv=It0v*(cos(delta0*fd)-j*sin(delta0*fd))
fefd=(abs(Eq0v)-(xd-xq)*imag(It0dqv))/Efd0
psid0=xd*imag(It0dqv)+Efd0*fefd
psiq0=xq*real(It0dqv)
ft=((psid0*real(It0dqv)-psiq0*imag(It0dqv))/fpg)/Te0
```

```
!Cálculo de parámetros:
Vt=sqrt((psid-Ra*iq)**2+(-psiq-Ra*id)**2)
It=sqrt(id**2+iq**2)*(Sbase/Sn)
Pg=(psid-Ra*iq)*iq+(-psiq-Ra*id)*id
Qg=-(psid-Ra*iq)*id+(-psiq-Ra*id)*iq
Torque=Tem*ft

!Almacenamiento de los datos de resultado:
open(30,file='datos1.dat',status='unknown')
open(40,file='datos2.dat',status='unknown')
open(50,file='datos3.dat',status='unknown')
open(60,file='datos4.dat',status='unknown')
write(30,*)'#Tiempo Vt It'
write(40,*)'#Tiempo Pg Qg'
write(50,*)'#Tiempo delta speed Torque'
write(60,*)'#Tiempo Efd'

do i=1,Ndatos+1
    write(30,*) t(i),Vt(i),It(i)
    write(40,*) t(i),Pg(i),Qg(i)
    write(50,*) t(i),delta(i)*180.0*fd/pi,wg(i),torque(i)
    write(60,*) t(i),Efd(i)*fefd
end do
close(30)
close(40)
close(50)
close(60)
print*,''
print*,achar(7)
print*,'Simulacion completa !'
print*,'Resultados guardados en "datos#.dat"'
print*,repeat(achar(196),45)
print*,''
80 write (*,'(''Presione: 1.Graficos / 2.Salir: '',\)')
read*,evento
if (evento==1) then
    goto 70
else if (evento==2) then
    goto 15
else
    print*,'Opcion incorrecta !'
    goto 80
end if

!Código para graficar:
70 call system('CLS')
print*,repeat(' ',10),'RESULTADOS ',nevento
print*,repeat(achar(196),45)
print*,'1. Vt'
print*,'2. It'
print*,'3. Efd'
print*,'4. Pg'
print*,'5. Qg'
print*,'6. Delta'
print*,'7. Wg'
print*,'8. Torque'
print*,' '
```

```
print*,'9. Nuevo evento'
print*,'10. SALIR'
print*,''
write (*,'(''Ingrese grafico: '',\)')
read*,evento
if (evento==1)then
    call system('wgnuplot -p plot_vt.plt')
    goto 70
else if (evento==2)then
    call system('wgnuplot -p plot_it.plt')
    goto 70
else if (evento==3)then
    call system('wgnuplot -p plot_efd.plt')
    goto 70
else if (evento==4)then
    call system('wgnuplot -p plot_pg.plt')
    goto 70
else if (evento==5)then
    call system('wgnuplot -p plot_qg.plt')
    goto 70
else if (evento==6)then
    call system('wgnuplot -p plot_delta.plt')
    goto 70
else if (evento==7)then
    call system('wgnuplot -p plot_wg.plt')
    goto 70
else if (evento==8)then
    call system('wgnuplot -p plot_torque.plt')
    goto 70
else if (evento==9)then
    call system('CLS')
    goto 10
else if (evento==10)then
   goto 15
else
    print*,' *** OPCION INCORRECTA ***'
    goto 70
endif

15    end program MSBI_avr_gov_2_1
```

ORDEN DE EMPASTADO