BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE INGENIERÍA

COLEGIO DE INGENIERÍA MECÁNICA Y ELÉCTRICA

PROGRAMACIÓN EN PYTHON PARA LA AUTOMATIZACIÓN DEL ANÁLISIS DE SISTEMAS ELÉCTRICOS DE POTENCIA EN PSS/E

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIATURA INGENIERÍA MECÁNICA Y ELÉCTRICA

PRESENTA:

EDUARDO DANIEL PÉREZ HERNÁNDEZ

REVISOR:

DR. ISMAEL ALBINO PADILLA

PUEBLA, PUE. ENERO 2020

BUAP

Oficio D-SA 2940/2019

C. EDUARDO DANIEL PÉREZ HERNÁNDEZ
PASANTE DE LA CARRERA DE INGENIERÍA MECÁNICA Y ELÉCTRICA
P r e s e n t e.

En atención al Tema de Tesis que puso Usted a consideración de la Coordinación de Área y de esta Secretaría Académica en coordinación con la Dirección de ésta Facultad de Ingeniería, dentro del marco de Titulación por Examen Profesional, como medio de Titulación se dio revisión y se ha autorizado el tema denominado:

"PROGRAMACIÓN EN PYTHON PARA LA AUTOMATIZACIÓN DEL ANÁLISIS DE SISTEMAS ELÉCTRICOS DE POTENCIA EN PSS/E".

Por lo anterior hacemos de su conocimiento que se asigna como asesor al Dr. Ismael Albino Padilla docente de la Facultad de Ingeniería.

Sin más por el momento, le envío la seguridad de mi consideración más distinguida.

A t e n t a m e n t e
"Pensar bien, para vivir mejor"
H. Puebla de Z. a 11 de diciembre de 2019

M. en I. Fernando Daniel Lazcano Hernández
Director

M'FDLH/MJAJT/BARV
C.c.p. Interesado
C.c.p. Archivo

Facultad de Ingeniería
Blvd. Valsequillo y Av. San Claudio s/n, edif. ING 4, Col. San Manuel, Ciudad Universitaria, Puebla, Pue. C.P. 72570
01 (222) 229 55 00 Ext. 7610

M. en I. Fernando Daniel Lazcano Hernández
Director de la Facultad de Ingeniería
Benemérita Universidad Autónoma de Puebla
P r e s e n t e.

El que suscribe: Dr. Ismael Albino Padilla, Asesor del tema de tesis:

"PROGRAMACIÓN EN PYTHON PARA LA AUTOMATIZACIÓN DEL ANÁLISIS DE SISTEMAS ELÉCTRICOS DE POTENCIA EN PSS/E".

Presentada por el C. Eduardo Daniel Pérez Hernández, pasante del Colegio de Ingeniería Mecánica y Eléctrica, y en atención al oficio No. D– SA 2940/2019 con fecha de emisión 11 de diciembre de 2019, me permito informar a Usted que después de haber revisado cuidadosamente el contenido temático, metodología, redacción y ortografía de la tesis correspondiente, no tengo inconveniente en autorizar la impresión de la misma.

Sin otro particular, le reitero la seguridad de mi más atenta y distinguida consideración.

A t e n t a m e n t e
"Pensar bien, para vivir mejor"
H. Puebla de Z. a 07 de enero de 2020

Dr. Ismael Albino Padilla
Asesor

D'IAP/BARV
C.c.p. Interesado
C.c.p. Archivo

Dedicatoria:

A mi Madre.

La razón por la cual siempre querré ser mejor.

A Pedro, gracias por guiarme

Agradecimientos,

Al Doctor. Ismael Albino Padilla por compartir su conocimiento y tiempo, puesto que sin ellos este trabajo no serían posible.

A las personas que me apoyaron, brindando su tiempo y esfuerzo en la corrección de mis errores, y demostrando que puedo contar con ellos Violeta, Gabriela, Libreros.

A mis amigos que hasta ahora he visto triunfar.

"Si vas a intentarlo,

ve hasta el final.

De lo contrario no empieces siquiera.

Tal vez suponga perder novias,

esposas, familia, trabajos

y quizás hasta la cabeza.

Tal vez suponga no comer durante

tres o cuatro días,

tal vez suponga helarte

en el banco de un parque.

Tal vez suponga la cárcel, la humillación,

el desdén y el aislamiento.

El aislamiento es el premio.

Todo lo demás sólo sirve para poner

a prueba tu resistencia,

tus auténticas ganas de hacerlo.

Y lo harás.

A pesar del rechazo y

de las ínfimas probabilidades,

y será mejor que cualquier cosa

que pudieras imaginar.

Si vas a intentarlo,

ve hasta el final.

No existe una sensación igual.

Estarás sólo con los dioses

y las noches arderán en llamas.

Llevarás las riendas de la vida

hasta la risa perfecta,

es por lo único que vale

la pena luchar."

Charles Bukowski (1920-1994)

Índice

Capítulo 1 Protocolo De Tesis 8
1.1 Introducción 8
1.2 Planteamiento Del Problema 9
1.3 Justificación 9
1.4 Objetivo General 9
1.5 Objetivos particulares 9
1.6 Hipótesis 10
1.7 Limitaciones 10
Capítulo 2 Introducción A Python 11
2.1 Introducción 11
2.2 ¿Qué es Python? 11
2.3 ¿Qué es un estudio de estabilidad transitoria? 12
2.4 Datos Necesarios Para Un Estudio De Estabilidad Transitoria 13
2.5 Introducción a Python con PSS/E 15
2.6 Procedimiento para la construcción de las macros en Python 16
Capítulo 3 Descripción de los Sistemas de Prueba y Ejecución de Macros 21
3.1 Introducción 21
3.2 Caso1 Sistema Bus Infinito con generador tipo (GENCLS) 22
3.3 Caso 2 Sistema Bus Infinito con generador tipo (GENCLS Y GENROU) 28
3.4 Caso 3 Sistema 14 Bus de IEEE 34
3.5 Caso 4 Sistema 9 Buses de IEEE 45
Capítulo 4 Resultados 53
4.1 Introducción 53
4.2 Caso1 Sistema Bus Infinito con generador tipo (GENCLS) 53
4.3 Caso 2 Sistema Bus Infinito con generador tipo (GENCLS Y GENROU) 57
4.4 Caso 3 Sistema 14 Bus de IEEE 61
4.5 Caso 4 Sistema 9 Buses de IEEE 65
Capítulo 5 Conclusiones 70
5.1 Conclusiones 70

5.2 Contribuciones 71
5.3 Áreas de oportunidad 71
Bibliografía 72
Anexos 73

Capítulo 1 Protocolo De Tesis

1.1 Introducción

Conforme avanza la actualización y modernización del sistema eléctrico nacional, es necesario actualizar y optimizar las formas de trabajo, esto obliga a los ingenieros a tomar medidas para ahorrar tiempos que serían indispensables en cuanto a planeación y entrega de resultados fundamentales con los cuales se puedan tomar decisiones en la entrada o salida de los diferentes equipos en el Sistema Eléctrico Nacional (SEN), así como para proveedores de energía independientes.

Teniendo en cuenta que los cambios que se han presentado por reformas energéticas, en la reestructuración del sistema eléctrico nacional y sus participantes, podemos referenciar la normativa establecida, se pueden realimentar los estudios solicitados por las diferentes dependencias gubernamentales y privadas para poder brindar eficiencia, confiabilidad, seguridad y sustentabilidad a el sistema eléctrico del cual el participante o usuario se mantenga alimentado.

Estos diferentes estudios son parte vital del SEN, por ello los analistas encargados se ven obligados a indagar en las diferentes herramientas que los programas certificados más utilizados les brindan, a fin de hacer un trabajo mucho más confiable y eficiente, estos programas especializados como lo es PSS/E contienen recursos fundamentales con los cuales el trabajo que se realiza día a día se puede optimizar en tiempos resumidos, así como lo ha sido crear formas de trabajo tomando de ejemplo la creación de casos base y la utilización de modelos óptimos, y no solo eso, también la fusión de otras áreas como lo es la programación, que es útil teniendo en cuenta como se ha avanzado en los últimos años en la automatización de procesos, así se puede aspirar a un funcionamiento independiente, con el cual el usuario solo nos encargamos de verificar los datos ingresados y esperar resultados evitando errores a la hora de realizar pasos repetitivos en la búsqueda de resultados en flujos de potencia y modelos dinámicos para estudios del sistema.

Todo aquello que sea posible compartir en cuanto a la optimización de resolución de problemas para flujos de potencia y sistemas dinámicos es fundamental para que ingenieros en formación que se encuentran en situación de investigación puedan tomar de referencia y así contribuir a la creación de nuevas formas de trabajo óptimas, estables y más seguras y sustentables.

1.2 Planteamiento Del Problema

Dentro de los análisis de sistemas eléctricos de potencia pueden encontrarse cada vez más con situaciones donde los análisis deben hacerse de una manera más rápida y eficiente, estas situaciones obligan a generar métodos de trabajo como lo son utilizar programas especializados en la resolución de estos análisis, de igual manera, conforme los sistemas crecen se complican los pasos para la resolución de análisis en específico. Esta complicación demora al analista con respecto a la entrega del trabajo en consecuencia las resoluciones tardan más de lo esperado, por ello la optimización del proceso requiere que en la resolución los pasos que se repitan o sean muy recurrentes, así como la carga de datos, se hagan de una manera automática, de esa forma estudiar la optimización con respecto a las herramientas brindadas por el programa (en este caso PSS/E) nos generará la oportunidad de reducir tiempos.

1.3 Justificación

Teniendo en cuenta que la información existente acerca de las herramientas que contiene el programa PSS/E suele ser poca y por lo regular solo se encuentra en cursos o capacitaciones de empresas especializadas, se pretende aumentar la cantidad de información disponible para el sector estudiantil o laboral de modo que se generen oportunidades y se impulse a los ingenieros en formación a interesarse en temas especializados en el área de sistemas de potencia, de igual manera la mezcla de la programación genera una inclusión de otras áreas generando interés fuera de la ingeniería eléctrica, de tal forma las colaboraciones y grupos de trabajo pueden ir creando nuevas oportunidades así como una visión más amplia para el área eléctrica.

1.4 Objetivo General

Crear macros para distintos casos de estudio dentro de la IEEE estos para mostrar la utilización de las diferentes herramientas y la optimización del análisis en el programa PSS/E.

1.5 Objetivos particulares

- Crear macros con el lenguaje Python para modelos de sistemas en el programa PSS/E
- Obtener resultados de estudios de flujos de potencia en el programa PSS/E mediante las macros realizadas con el lenguaje de Python

- Crear macros para sistemas dinámicos en el programa PSS/E
- Hacer simulaciones de fallas utilizando macros creadas a partir de Python
- Obtener resultados gráficos a partir de macros creadas con Python para el programa de PSS/E

1.6 Hipótesis

Al crear macros básicas para la resolución de flujos de potencia, así como análisis de sistemas dinámicos utilizar el programa PSS/E será más rápido y se podrá ahorrar procesos y brindar una forma diferente de analizar sistemas eléctricos generando una alternativa para los nuevos usuarios de programa.

1.7 Limitaciones

Las últimas versiones del programa PSS/E tienen una diferencia en la programación de macros, por ello será necesario ajustar los valores de versión a versión. Los modelos que se utilizarán son modelos clásicos y las fallas a moderar serán sencillas, únicamente para comprobar la funcionalidad de las macros durante los procedimientos para el análisis dinámico.

Capítulo 2 Introducción A Python

2.1 Introducción

En el presente capítulo se analizará a grandes rasgos el lenguaje de programación que se utilizará en la creación de macros para la resolución de análisis de flujos de potencia y análisis dinámico, así como los puntos por los que se consideró este lenguaje y cómo es que a lo largo del desarrollo del proyecto será útil, puntualizando la compatibilidad con otros programas y la fluidez de estos al trabajar con este lenguaje.

Se analizará qué y para qué hacer un estudio de flujos de potencia y estabilidad transitoria, cuál es el beneficio de obtener gráficas y su información correspondiente a través de su diagrama de flujos, y de esta manera mostrar cómo es que los resultados se pueden interpretar y brindar cuándo se resuelve un caso.

Ya identificada la finalidad de los estudios de flujos de potencia se dan a conocer los datos esenciales para correr un análisis de flujos de potencia y estabilidad transitoria, de tal forma que se tenga una base de datos previa para poder tomar la información y utilizarla dentro de las macros propuestas.

Una vez que se conozcan los principales puntos de la programación y análisis de sistemas de potencia, se dará la introducción de cómo es que PSS/E brinda las herramientas y facilita al usuario los principales códigos con los que se puede trabajar optimizando la metodología y facilitando al usuario la elaboración de procesos específicos.

Cuando se haya trabajado con los documentos y la información necesaria para el análisis de sistemas, se procederá a la explicación de creación de las macros con el lenguaje ya descrito, de esa forma el usuario podrá tener una idea y referenciar cómo es que se fue resolviendo el procedimiento de flujos de potencia y estabilidad transitoria con las macros, el porqué de cada paso y su resolución.

2.2 ¿Qué es Python?

Raúl González Duque define que Python:
"Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos."[1]

Para comprender mejor esta definición se desglosará las características de este lenguaje:

- Lenguaje interpretado o de script:

Al trabajar con un lenguaje interpretado la característica principal es que el programador crea un código fuente y lo ejecuta directamente en el script, este tipo de lenguaje en la programación es más sencillo para el programador ya que en el ciclo de desarrollo es más rápido que un lenguaje compilado, aunque agrega procesos adicionales a la máquina.

- Tipado dinámico:

  Para tener una idea del tipado dinámico, es necesario conocer el tipado estático que obliga al programador a definir las variables desde el principio, a diferencia del dinámico, que nos brinda la oportunidad de no definir el tipo cuando declaramos una variable y esta irá cambiando el tipo conforme se le asignen valores de otro tipo.

- Lenguaje fuertemente tipado:

  Un lenguaje fuertemente tipado es aquel que no permite tratar el valor de una variable distinto al tipo que pertenece, ya que es necesario hacer una conversión de tipo si se desea manejar uno distinto.

- Multiplataforma:

  "El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no se utiliza librerías específicas de cada plataforma el programa podrá correr en todos estos sistemas sin grandes cambios."[1]

- Orientado a objetos:

  Un lenguaje orientado a objetos se caracteriza principalmente porque combina datos y acciones que comúnmente reciben el nombre de métodos dentro de una estructura lógica conocida como objeto y estos a su vez pertenecen a una clase en específico, con esta estructura podemos administrar y desarrollar con mayor facilidad aplicaciones y estructuras de datos de gran tamaño por la funcionalidad y la interacción entre objetos.

2.3 ¿Qué es un estudio de estabilidad transitoria?

La estabilidad transitoria estudia la capacidad de mantener el sincronismo durante o después de perturbaciones de gran magnitud, que pueden ser pérdidas de generación o cargas importantes.

El sistema, después de las causas ya descritas, responde mediante grandes variaciones, en ángulos de los generadores síncronos, así como oscilaciones en los flujos de potencia, tensiones, corrientes, etc. Si el separador angular permanece acotado en los generadores síncronos entonces se mantendrá sincronismo en el sistema. De no ser así se perderá el sincronismo notoriamente después de dos o tres segundos.

Las razones porque los análisis de estabilidad son necesarios son tener medidas de prevención o de acciones remediales después de perder interconexiones con redes de transmisión o con puntos cercanos de interconexión y de esa manera poder seguir brindando continuidad al sistema de manera eficiente y confiable.

Los estudios siempre se irán complicando ya sea por el tamaño que el sistema va tomando o simplemente por los avances tecnológicos, inclusive John J. Grainger y William D. Stevenson lo mencionan de esta manera: "La representación clásica de la máquina es conveniente para muchos de los estudios. Sin embargo, se pueden requerir modelos más elaborados para representar los modelos turboalternadores con características dinámicas determinadas por muchos de los avances tecnológicos en el diseño de la máquina y de sus sistemas de control." Con lo anterior podemos destacar que siempre que un estudio se hace de manera correcta tomando una nueva tecnología es necesario rehacer estudios para todo el sistema.[3]

2.4 Datos Necesarios Para Un Estudio De Estabilidad Transitoria

La información necesaria para los estudios de estabilidad transitoria se obtiene principalmente del caso base, y los primeros estudios que anteceden al análisis transitorio, estos datos requeridos se enlistaran enseguida: [10]

- Caso base:
  Compone toda la información general del sistema, se puede tomar como la base de datos que se cargará y de donde partirán los primeros análisis del sistema.

- Impedancia de líneas:
  Se genera por las propiedades que componen a la línea (longitud, material), la impedancia compuesta por R, X y B está dada en unidades PU.

- Tensión controlada en buses de generación:
  Es la tensión controlada desde las centrales de generación, por lo regular se usa para controlar todo el voltaje del sistema, se utiliza en unidades PU.

- Potencia real generada:

Es la potencia real que brindan los generadores se expresa en unidades de MW.

- Potencia reactiva generada:
  Es la potencia reactiva generada para el sistema esta se expresa en unidades Mvar.

- Impedancia en los generadores:
  Se refiere a la impedancia del generador que se asocia al sistema, se especifica la reactancia transitoria en unidades PU.

- Carga potencia real:
  Esta es la carga que se ajusta en cada bus (esta carga no incluye las cargas que demanda la estructura del sistema), las unidades de esta se expresan en MW.

- Carga de potencia reactiva:
  Es la carga en potencia reactiva en unidades de Mvar.

- Impedancia en los transformadores de dos devanados:
  Esta impedancia en transformadores de dos devanados que por lo general solo se expresa con la reactancia X en unidades PU para casos de estudio.

- Transformadores de tres devanados y su impedancia
  Este tipo de transformadores no son utilizados, aunque para el sistema en casos de estudio de pueden hacer equivalentes con arreglos de dos devanados, su reactancia X de igual manera se representará en unidades PU.

- El tipo de los buses del sistema.
  Swing bus o Slack bus: utilizado para equilibrar la potencia activa y potencia reactiva en los sistemas durante estudios de flujos de potencia.

  Bus de generación: para este bus se especifica el voltaje y la potencia real, estos dos valores serán constantes para cada bus existente de este tipo.

  Bus de carga: en este tipo de bus se tendrán especificadas la potencia real y reactiva, en este bus se tendrá una generación negativa originada por la fluidez de potencia en el sistema.

- Tensión nominal en líneas:
  Un dato que se recibe de sistema resultado de flujos de potencia. Las unidades son en kV

- Bancos de capacitores:
  En casos específicos se colocan bancos de capacitores para estudios corrigiendo el factor de potencia o modificando perfiles de voltaje herramienta que se usa para observar si es factible su instalación sus unidades se utilizan principalmente en Mvar.

- Tiempos antes, durante y después de la falla:
  Estos tiempos son aquellos con los que se mostrará cada etapa del análisis para su mayor comprensión en cada instante.

- Bus fallado:
  Este bus puede ser cualquiera que genera el transitorio y por el cual se hace el estudio

- Tipo de generador a modelar:
  Este puede variar entre los diferentes modelos que brinda el programa PSS/E junto a la información con la que se cuente del caso base.

2.5 Introducción a Python con PSS/E

Python al ser un lenguaje de script tiene la facilidad de trabajar con programas que tengan una interfaz que permita interpretar la sintaxis del su lenguaje, de esta forma PSS/E brinda la facilidad de trabajar con su entorno mediante la programación de macros, estas macros son procedimientos definidos para ser utilizados de manera que se optimice la forma de trabajo, el usuario puede crear procesos muy específicos con los cuales no sea necesario la interacción hasta la hora de mostrar o interpretar resaltados o bien el usuario puede crear programas complejos con los que algún otro usuario sin la necesidad de conocer más que los datos básicos del caso por estudiar, pueda manejar o realizar análisis sin algún inconveniente.

Todos estos procesos brevemente descritos son posibles gracias a una serie de códigos que el programa PSS/E tiene incorporados ,PSS/E incluso tiene herramientas donde se pueden generar estos códigos el por sí mismos, sin embargo la utilización del lenguaje de programación para poder crear nuestros propios procesos se vuelve más conveniente cuando la se requiere la interacción entre otros programas, por ejemplo la paquetería de office, los códigos previamente mencionados se encuentran en "PSS®E Application Program Interface (API)", información incorporada en la guía de PSS/E.

2.6 Procedimiento para la construcción de las macros en Python

PSS/E tiene la capacidad de ejecutar macros de algunos lenguajes de programación, y la información que se utiliza para poder crearlas se encuentra en su manual, con los siguientes ejemplos se mostrara como aplicar estas herramientas que brinda el programa PSS/E

Para el primer ejemplo se toma un cogido con el cual ingresaremos los datos de bus. Podemos observar que todos los códigos vienen acompañados de una breve introducción de las acciones a realizar.

"2.4 BUS_DATA_2
This API is the second release of the bus data specification engine. Use this API to modify the data of an existing bus in the working case or to add a new bus to the working case (see PSS®E Program Operation Manual, Bus Data)".[4]

Estas explicaciones suelen ser de gran ayuda ya que algunas acciones son similares dentro de la aplicación de PSS/E o se realizan en la misma ventana de opciones, sin embargo, para la programación se debe de especificar la acción a resolver.

Cuando se ubica la acción a realizar, PSS/E nos indica cómo es que se debe generar la sintaxis con la que se deberá escribir el código, estos bloques de información serán indispensables para que nuestra programación compile adecuadamente.

"Python syntax:
`ierr = bus_data(i, intgar, realar, name)`". [4]

Una vez que se localiza el código principal se tendrá que llenar cada uno de los bloques de información que están especificados en el mismo capítulo, la guía brinda como es que debe de introducir los datos dentro del bloque de información, es importante recordar que la guía al especificar más la sintaxis de Python nos describe cómo es que se interpretarán los bloques de datos.

"Python commands: Python is an open source language; its syntax is well documented. Some of are arguments are defined as arrays (in keeping with Fortran nomenclature, as mentioned above). Python does not have arrays, and in general you should interpret arrays to mean lists (although tuples will usually work on input). Arrays that have some elements as input and some as output are in the Fortran syntax represented in the Python syntax as two separate lists. Keywords can be used for arguments, and any argument can be omitted".[4]

were:

Integer	Is the bus number	
Integern INTGAR(4)	Is an array of four elements	
	INTGAR(1)	IDE, bus type code (1 by default).
	INTGAR(2)	AREA, area number (1 by default)
	INTGAR(3)	ZONE, zone number (1 by default)
	INTGAR(4)	OWNER, owner number (1 by default).
Real REALAR(3)	Is an array of three elements (input).	
	REALAR(1)	BASKV, bus base voltage in kV (0.0 by default)
	REALAR(2)	VM, bus voltage magnitude in pu. (1.0 by default)
	REALAR(3)	VA, bus voltage phase angle (0.0 by default)
Character NAME*12	Is the bus name (input; blank by default).	
Integer IERR	Is the error code (output).	
	IERR = 0	no errors occurred
	IERR > 0	fatal error occurred
	IERR = 1	invalid bus number
	IERR = 2	bus table is full.
	IERR = 3	working case is not initialized
	IERR = 4	prerequisite requirements for API are not met
IERR = -1	data error, warning and information messages; one or more of:	
	• Invalid base voltage.	• Invalid bus type code.
	• Invalid area, zone or owner number.	• Bus name is more than 12 characters.
	• New area number but bus is the area swing for its old area.	• Bus type code is 3 but is not the swing bus for its area.
	• Area table is full.	• Zone table is full.
	• Owner table is full.	

Tabla 2.1 Ejemplo de descripción de errores dentro del programa PSSE

Conociendo lo anterior se podrá tener una idea de cómo crear las sintaxis en nuestros códigos de cada macro, este procedimiento es bastante sencillo ya que los parámetros regularmente se encuentran establecidos en los casos base.

Una vez colocados los parámetros en los lugares correspondientes dentro de la plantilla descrita se observará de la siguiente manera:

```
Pss.bus_data_2(1,[2,11,13,12],[230.0,1.4,0.5],r""""BUS2""""
).[4]
##(bus,[code,area,zone,owner],[baseKV,voltaje(pu),angle]""""nombre del bus"""")
```

Cuando los datos se encuentran en el sitio correspondiente, al correr esta primera macro generan el bus antes descrito con los parámetros solicitados.

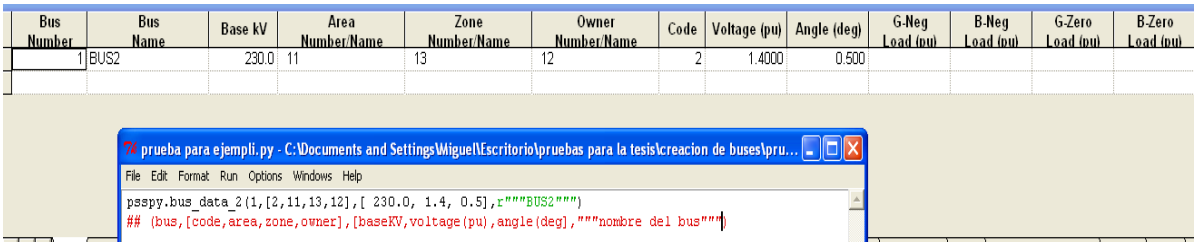

Figura 2.1 Ejemplo del ingreso de la información a PSSE

Esta es una forma del llenado de información, sin embargo, se complica cuando se desea trabajar con bases de datos de gran magnitud, estas bases de datos correspondientes pueden ser de cientos de buses dependiendo que parte del SEN se quiera analizar.
Un ejemplo puede ser hojas de cálculo donde se encuentre almacenada dicha información.

	B	C	D	E	F	G	H	I	J	K	L	M
1												
2	Bus Name	Base kV	Area Number/Name	Zone Num	Owner Nu	Code	Voltage (pu)	Angle (deg)	G-Neg Load (pu)	B-Neg Load (pu)	G-Zero Load (pu	B-Zero Load (pu)
3	BUS 1	230	1	1	1	3	1.06	0				
4	BUS 2	230	1	1	1	2	1.045	0				
5	BUS 3	230	1	1	1	2	1.01	0				
6	BUS 4	230	1	1	1	1	1	0				
7	BUS 5	230	1	1	1	1	1	0				
8	BUS 6	115	1	1	1	2	1.07	0				
9	BUS 7	1	1	1	1	1	1	0				
10	BUS 8	13.2	1	1	1	2	1.09	0				
11	BUS 9	115	1	1	1	1	1	0				
12	BUS 10	115	1	1	1	1	1	0				
13	BUS 11	115	1	1	1	1	1	0				
14	BUS 12	115	1	1	1	1	1	0				
15	BUS 13	115	1	1	1	1	1	0				
16	BUS 14	115	1	1	1	1	1	0				

Figura 2.2 Base de datos creada para el ingreso de la información desde una hoja de Excel

Sin embargo, cuando se tiene una base de datos se puede acelerar el proceso por medio de herramientas en la programación, de modo que si se desea introducir bastante información

es mejor no saturar las líneas de código con información que solo varia en algunos parámetros.

Por ejemplo:

```
def main():
  m1 = open("14barras.csv","r")
  m1_csv = csv.reader(m1)
  for v in m1_csv:

pssy.bus_data_2(float(v[0]),[float(v[6]),float(v[3]),float(v[4]),float(v[5]),
float(v[2]),float(v[7]),float(v[8]),str(v[1]))
  m1.close()
main()
```

Con lo anterior se puede observar que el proceso se agiliza, evitando estar escribiendo y modificando cada una de las líneas para cada uno de los buses a crear. Y así introducir bloques de información de gran tamaño.

	Bus Number	Bus Name	Base kV	Area Number/Name	Zone Number/Name	Owner Number/Name	Code	Voltage (pu)	Angle (deg)	G-Neg Load (pu)	B-Neg Load (pu
	1	BUS 1	230.0	1	1	1	3	1.0600	0.000		
	2	BUS 2	230.0	1	1	1	2	1.0450	0.000		
	3	BUS 3	230.0	1	1	1	2	1.0100	0.000		
	4	BUS 4	230.0	1	1	1	1	1.0000	0.000		
	5	BUS 5	230.0	1	1	1	1	1.0000	0.000		
	6	BUS 6	115.0	1	1	1	2	1.0700	0.000		
	7	BUS 7	1.0	1	1	1	1	1.0000	0.000		
	8	BUS 8	13.2	1	1	1	2	1.0900	0.000		
	9	BUS 9	115.0	1	1	1	1	1.0000	0.000		
	10	BUS 10	115.0	1	1	1	1	1.0000	0.000		
	11	BUS 11	115.0	1	1	1	1	1.0000	0.000		
	12	BUS 12	115.0	1	1	1	1	1.0000	0.000		
	13	BUS 13	115.0	1	1	1	1	1.0000	0.000		
	14	BUS 14	115.0	1	1	1	1	1.0000	0.000		
*											

14barraspy2.py - C:\Documents and Settings\Miguel\Escritorio\pruebas para la tesis\pruebas de programacion\prueba de armado de buses 2 esta ya sirve\14barraspy2.py

File Edit Format Run Options Windows Help

```
import csv
psspy.newdiagfile()
psspy.newcase_2([0,0], 100.0, 60.0,"","")

def main():
    m1 = open("14barras.csv", "r")
    m1_csv = csv.reader(m1)
    for v in m1_csv:
        psspy.bus_data_2(float(v[0]),[float(v[6]),float(v[3]),float(v[4]),float(v[5])],[float(v[2]),float(v[7]),float(v[8])],str(v[1]))
    m1.close()
main()
```

Figura 2.3 Ejemplo de bloque de programación para la utilización de la información de una hoja de Excel

Entonces, todo aquel componente que se utilizara dentro del análisis es necesario identificar junto a su código que le corresponde para poder introducir los parámetros con los que va a trabajar.

Es de gran importancia tener conocimientos previos del programa PPSS/E ya que muchas especificaciones dentro del manual están dirigidas a usuarios experimentados en el programa. Algunos parámetros pueden ser erróneos si no se conoce el tipo al que pertenece el valor por trabajar. Por ello, es de gran importancia a la hora de trabajar con bases de datos, tener bien establecida la tipología de estos datos, ya que si se desea una programación sin errores la tipología es fundamental.

Cuando no se respetan los parámetros que establece PSS/E se generan errores, estos también se encuentran en la tabla anterior de las especificaciones de la sintaxis de Python para PSS/E, de esa forma se identifica la acción a corregir dentro del código de programación de forma más rápida.

Capítulo 3 Descripción de los Sistemas de Prueba y Ejecución de Macros

3.1 Introducción

Parte importante del análisis dinámico es entender que con regularidad la información escasea por fines de confiabilidad de las empresas, esto genera que, al tratar de localizar los controles, así como especificaciones en análisis de nuevas tecnologías se imposibilite, sin embargo, estos principales modelos didácticos ayudan a la comprensión principal del ingeniero en formación, para poder enfrentarse a análisis más complejos cuando sea requerido.

En este capítulo se describen los ejemplos de sistemas de potencia utilizados para la creación de macros y la ejecución de estas en ellos, se muestran los parámetros utilizados en bases de datos, de esta forma se describe cómo es que cada código fue creado específicamente y diseñado para un proceso repetitivo o una acción que requiriera de múltiples operaciones.

Al ser casos tomados de la IEEE se respetaron los apuntes adquiridos tomando los datos que se muestran para mostrar la correcta aplicación de cada análisis, esto a fin de mostrar que las macros son una herramienta fundamental, de igual forma la creación del diagrama se vuele una cuestión opcional ya que los datos y las resoluciones no necesitan esa parte gráfica.

3.2 Caso1 Sistema Bus Infinito con generador tipo (GENCLS)

Los datos utilizados para la realizar macros fueros obtenidos en el “procedimiento para evaluar estabilidad transitoria usando PSSE”.[2][10]. Con este contenido se crearon todos los procedimientos a partir de macros que redujeran el tiempo de captura y ejecución de acciones necesarias para correr los diferentes estudios necesarios dentro de análisis y obtención de datos.

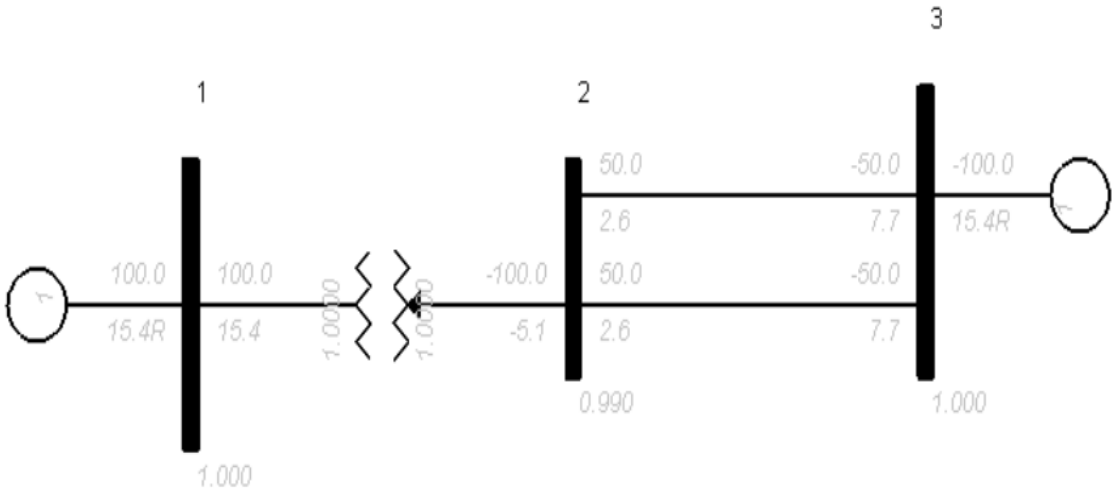

Figura 3.1 Diagrama utilizado para el sistema de bus infinito

DATOS DE LINEAS			
De Bus a Bus (number)	Impedancia R (su)	Reactancia X (pu)	B(pu)
2-3	0	0.4	0
2-3	0	0.4	0

Tabla 3.1 Datos de líneas requeridos para sistema de bus infinito

TENSION CONTROLADA	
Bus (numero)	Bus (numero)
1	1
3	3

Tabla 3.2 Datos de tensión controlada requeridos para el sistema de bus infinito

POTENCIA GENERADA		
Bus (numero)	P Gen (MW)	Q Gen (Mvar)
1	100	0
3	0	0

Tabla 3.3 Datos de potencia generada requeridos para el sistema de bus infinito

IMPEDANCIA DE LOS GENERADORES	
Bus (numero)	X Source (pu)
1	0.2
3	0.1

Tabla 3.4 Datos de impedancia en los generadores requeridos para el sistema de bus infinito

DATOS DE CORTOCIRCUITO EN LOS GENERADORES	
Bus (numero)	ZX-Pos, ZX-Neg, ZX-Zero (pu).
1	0.2
3	1

Tabla 3.5 Datos de cortocircuito de los generadores requerido para el sistema de bus infinito

INPEDANCIA EN TRANSFORMADORES DE DOS DEVANADOS	
De Bus a Bus (number)	Specified X (pu)
1-3	0.1

Tabla 3.6 Datos de impedancia en transformadores requeridos para el sistema de bus infinito

TIPOS DE BUS	
Bus (numero)	Code
1	2
2	1
3	3

Tabla 3.7 Datos de tipos de buses requeridos para el sistema de bus infinito

DATOS DINAMICOS DEL GENERADOR (GENCLS)			
Bus (numero)	Mbase (MVA)	Generador	H
1		GENCLS	5
3		GENCLS	100

Tabla 3.8 Datos dinámicos del generador GENCLS requeridos para el sistema de bus infinito

DATOS DE SIMULACION	
Bus donde ocurrirá la falla	2
Tiempo antes de la falla	1s
Tiempo de falla	0.05s (la falla dura 3 ciclos)
Línea que se libera	Ninguna
Tiempo total del evento	10s

Tabla 3.9 Datos de simulación requeridos para el sistema de bus infinito

Una vez listos los datos se generan archivos CSV como base de datos y con ellos se estructurará la programación. Se generan múltiples archivos CSV para ejemplificar de forma más detallada y específica cada proceso en la introducción de datos y ejecución de procesos.

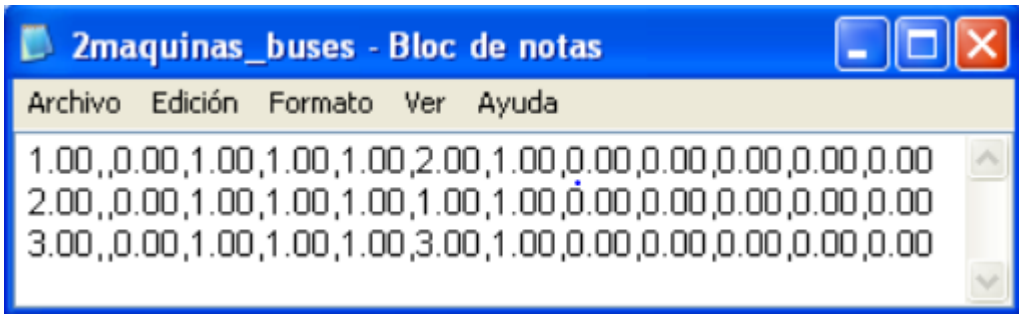

Figura 3.2 Base de datos en CSV para la programación de macros.

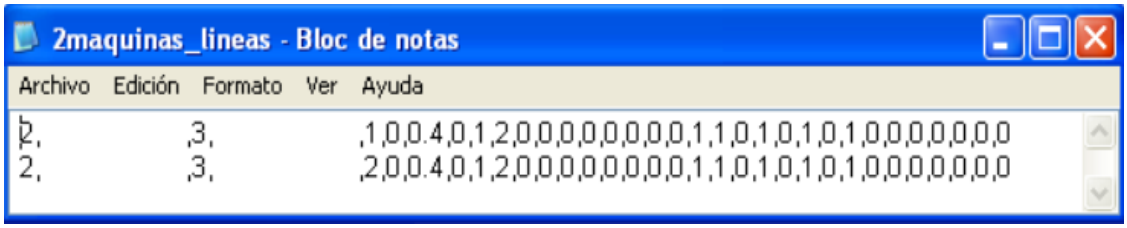

Figura 3.3 Base de datos en CSV para la programación de macros

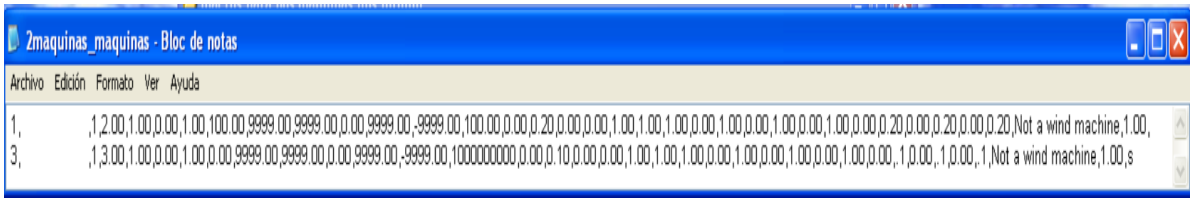

Figura 3.4 Base de datos en CSV para la programación de macros

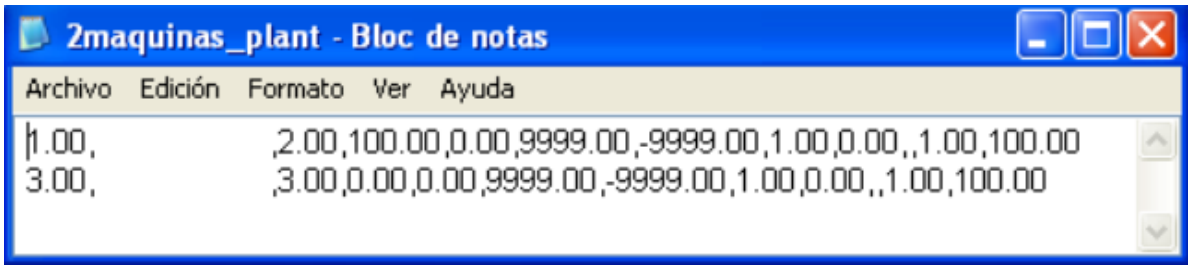

Figura 3.5 Base de datos en CSV para la programación de macros

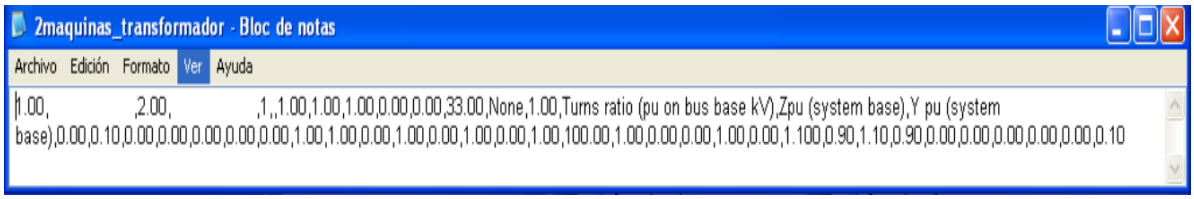

Figura 3.6 Base de datos en CSV para la programación de macros

```
import csv                                          % importar archivos csv
psspy.newdiagfile()                                 % crear nuevo diagrama
psspy.newcase_2([0,0], 100.0, 60.0,"","")           % crear nuevo caso con datos base

def bus():
    m1 = open("2maquinas_buses.csv", "r")           % introduce la información de bus
    m1_csv = csv.reader(m1)                         % a la hoja network data desde un
    for v in m1_csv:                                % archivo csv

psspy.bus_data_2(float(v[0]),[float(v[6]),float(v[3]),float(v[4]),float(v[5])],[flo
at(v[2]),float(v[7]),float(v[8])],str(v[1]))   %asigna los valores correspondientes
    m1.close()
bus()                                               %llama a la función para ejecutarla

def lineas():                                       % introduce la información de líneas
    m1 = open("2maquinas_lineas.csv", "r")          % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for b in m1_csv:

psspy.branch_data(float(b[0]),float(b[2]),b[4],[float(b[8]),float(b[9]),float(b[18]
),float(b[20]),float(b[22]),float(b[24])], [float(b[5]), float(b[6]), float(b[7]),
float(b[10]), float(b[11]), float(b[12]), float(b[13]), float(b[14]), float(b[15]),
float(b[16]), float(b[17]),float(b[19]),float(b[21]),float(b[23]),float(b[25])])
```

```
    m1.close()                                  % asigna los datos correspondientes
lineas()                                        % llama a la función para ejecutarla

def plant():                                    % introduce la información de voltaje
    m1 = open("2maquinas_plant.csv", "r")       % preparando las máquinas para
    m1_csv = csv.reader(m1)                     % asignar los valores correspondes
    for p in m1_csv                             % desde un archivo csv
        psspy.plant_data(float(p[0]),0,[ float(p[7]),100])
    m1.close()                                  % asigna valores correspondientes
plant()                                         % llama a la función para ejecutarla

def maquinas():                                 % asigna los valores de las maquinas
    m1 = open("2maquinas_maquinas.csv", "r")    % a la hoja network data desde un
    m1_csv = csv.reader(m1)                     % archivo csv
    for m in m1_csv:

psspy.machine_data_2(float(m[0]),r"""1""",[1,1,0,0,0,0],[float(m[7]),0,float(m[11])
,float(m[12]),9999,-9999,float(m[13]),0,float(m[15]),0,0,1,1,1,1,1,1])
    m1.close()                                  % asigna valores correspondientes
maquinas()                                      % llama a la función para ejecutarla

def transformador():                            % asigna valores del
    m1 = open("2maquinas_transformador.csv", "r") % transformador a la hoja
    m1_csv = csv.reader(m1)                     % network data desde un archivo
    for w in m1_csv:                            % csv

psspy.two_winding_data_3(float(w[0]),float(w[2]),w[4],[float(w[6]),float(w[0]),floa
t(w[24]),float(w[26]),float(w[28]),float(w[30]),33,float(w[45]),float(w[0]),0,1,0,1
,1,1],[float(w[17]),float(w[18]),100,float(w[33]),float(w[34]),float(w[35]),float(w
[36]),float(w[37]),float(w[19]),float(w[20]),float(w[21]),1,1,1,1,float(w[22]),floa
t(w[23]),float(w[38]),float(w[39]),float(w[40]),float(w[41]),0,0,0,],"")
    m1.close()                                  % asigna valores correspondientes
transformador()                                 % llama a la función para ejecutarla

psspy.newseq()                      % prepara los datos de cortocircuito

def seq():                                      % asigna los datos de cortocircuito
    m1 = open("2maquinas_maquinas.csv", "r")    % desde un archivo csv a la hoja
    m1_csv = csv.reader(m1)                     % network data
    for m in m1_csv:
        psspy.seq_machine_data(float(m[0]),r"""1""",[0,float(m[28]),0,
float(m[30]),0,float(m[32])])                   % asigna valores correspondientes
    m1.close()
seq()                                           % llama a la función para ejecutarla

psspy.fnsl([0,0,0,0,1,1,99,0]) % solución de flujos de potencia Newton Raphson

psspy.cong(0)
psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])  % ajustes de líneas y generadores
psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])  % para análisis dinámico
psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])

psspy.ordr(0)                                   % se asumen los elementos en
                                                % operación
psspy.fact()
psspy.tysl(1)                                   % solution for swiching studies
```

```
psspy.add_plant_model(1,r"""1""",1,r"""GENCLS""",0,"",0,[],[],2,[0.0,0.0])
psspy.change_plmod_con(1,r"""1""",r"""GENCLS""",1, 5.0)
psspy.add_plant_model(3,r"""1""",1,r"""GENCLS""",0,"",0,[],[],2,[0.0,0.0])
psspy.change_plmod_con(3,r"""1""",r"""GENCLS""",1, 100.0)
     % introducir generadores dentro de la hoja dynamics data

psspy.chsb(0,1,[-1,-1,-1,1,1,0])              % desactivar parámetros
psspy.chsb(0,1,[-1,-1,-1,1,2,0])              % dejando los necesarios
                                              % a graficar

psspy.strt(0,r"""C:\Documents and Settings\Miguel\Escritorio\macros para dos
máquinas bus infinito\grf.out""")  % asigna la dirección y nombre para archivos out

psspy.run(0, 1.0,1,1,0)                       % corre simulación dinámica

psspy.dist_bus_fault(2,1,0.0,[0.0,-0.2E+10])  % agrega la información de falla
                                              % dentro de un bus

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\macros para dos máquinas bus infinito\grf.out""")
                                        % refresca la información de las graficas

psspy.run(0, 1.05,1,1,0)                      % corre simulación dinámica

psspy.dist_clear_fault(1)                     % libera la falla

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\macros para dos máquinas bus infinito\grf.out""")
                                        % refresca la información de las graficas

psspy.run(0, 10,1,1,0)                        % corre simulación dinámica

pssplot.newplotbook()                      % abre un nuevo libro de dibujo
pssplot.setselectedpage(0)                 % configura el libro de dibujo activo

pssplot.openchandatafile(r"""C:\Documents and Settings\Miguel\Escritorio\macros
para dos máquinas bus infinito\grf.out""")   % carga un archivo direccionado out

pssplot.dragdropplotdata(r"""grf""",r"""1 - ANGL     1[                ]1""")
pssplot.dragdropplotdata(r"""grf""",r"""2 - ANGL     3[                ]1""")
pssplot.dragdropplotdata(r"""grf""",r"""3 - POWR     1[                ]1""")
pssplot.dragdropplotdata(r"""grf""",r"""4 - POWR     3[                ]1""")
                  % carga los archivos de salida dentro del libro activo
```

Las diferentes funciones mostradas anteriormente son las necesarias para correr el análisis y mostrar los resultados dentro del programa PSSE.

3.3 Caso 2 Sistema Bus Infinito con generador tipo (GENCLS Y GENROU)

Los datos utilizados para la realizar macros fueros obtenidos en el "procedimiento para evaluar estabilidad transitoria usando PSSE".[2][10]. Se modificaron los parámetros del generador conectado en bus 1 y se actualizan con un generador tipo GENROU con la intensión de notar el impacto en el cambio de tecnología en el sistema analizado

DATOS DE LINEAS			
De Bus a Bus (number)	Impedancia R (pu)	Reactancia X (pu)	B(pu)
2-3	0	0.4	0
2-3	0	0.4	0

Tabla 3.10 Datos de líneas requeridos para sistema de bus infinito GENCLS Y GENROU

TENSION CONTROLADA	
Bus (numero)	Tensión (pu)
1	1
3	1

Tabla 3.11 Datos de tensión controlada requeridos para el sistema de bus infinito GENCLS Y GENROU

POTENCIA GENERADA		
Bus (numero)	PGen (MW)	QGen (Mvar)
1	100	0
3	0	0

Tabla 3.12 Datos de potencia generada requeridos para el sistema de bus infinito GENCLS Y GENROU

IMPEDANCIA DE LOS GENERADORES	
Bus (numero)	X Source (pu)
1	0.205
3	0.1

Tabla 3.13 Datos de impedancia en los generadores requeridos para el sistema de bus infinito GENCLS Y GENROU

DATOS DE CORTOCIRCUITO EN LOS GENERADORES	
Bus (numero)	ZX-Pos, ZX-Neg, ZX-Zero (pu).
1	0.205
3	1

Tabla 3.14 Datos de cortocircuito de los generadores requerido para el sistema de bus infinito GENCLS Y GENROU

INPEDANCIA EN TRANSFORMADORES DE DOS DEVANADOS	
De Bus a Bus (number)	Specified X (pu)
1-3	0.1

Tabla 3.15 Datos de impedancia en transformadores requeridos para el sistema de bus infinito GENCLS Y GENROU

TIPOS DE BUS	
Bus (numero)	Code
1	2
2	1
3	3

Tabla 3.16 Datos de tipos de buses requeridos para el sistema de bus infinito GENCLS Y GENROU

DATOS DINAMICOS DEL GENERADOR (GENCLS)			
Bus (numero)	Mbase (MVA)	Generador	H
3		GENCLS	100

Tabla 3.17 Datos dinámicos del generador GENCLS requeridos para el sistema de bus infinito GENCLS Y GENROU

DATOS DINAMICOS DEL GENERADOR (GENROU)								
Bus (numero)	Generador	T´do	T´´do	T´q	T´´q	H	D	Xd
1	GENROU	0.5871	0.0248	0.1351	0.0267	2.656	2	1.67
Bus (numero)	Xq	X´d	X´q	X´´d=X´´q	Xl	S(1.0)	S(1.2)	
1	1.60	0.265	0.46	0.205	0.15	0.091	0.4	

Tabla 3.18 Datos dinámicos del generador GENROU requeridos para el sistema de bus infinito GENCLS Y GENROU

Datos de los excitadores								
Bus (numero)	Excitador	Tr	Ka	Ta	Vrmax	Vrmin	Ke	Te
1	IEEET1	0	50	0.06	1	-1	-0.0465	0.52
Bus (numero)	Kf	Tf	Switch	E1	SE(E1)	E2	SE(E2)	
1	0.0832	1	0	3.24	0.072	4.32	0.2821	

Tabla 3.19 Datos dinámicos de los excitadores requeridos para el sistema de bus infinito GENCLS Y GENROU

DATOS DE SIMULACION	
Bus donde ocurrirá la falla	2
Tiempo antes de la falla	1s
Tiempo de falla	0.05s (la falla dura 3 ciclos)
Línea que se libera	Ninguna
Tiempo total del evento	10s

Tabla 3.20 Datos de simulación requeridos para el sistema de bus infinito GENCLS Y GENROU

Para el siguiente caso se mantiene la base de datos ya que las modificaciones son realizadas directamente a partir de líneas de código dentro de la macro antes creada.

```
import csv                                          % importar archivos csv
psspy.newdiagfile()                                 % crear nuevo diagrama
psspy.newcase_2([0,0], 100.0, 60.0,"","")           % crear nuevo caso con datos base

def bus():                                          % introduce la información de bus
    m1 = open("2maquinas_buses.csv", "r")           % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for v in m1_csv:

psspy.bus_data_2(float(v[0]),[float(v[6]),float(v[3]),float(v[4]),float(v[5])],[flo
at(v[2]),float(v[7]),float(v[8])],str(v[1]))        % asigna los datos correspondientes
    m1.close()
bus()                                               % llama a la función para ejecutarla

def lineas():                                       % introduce la información de líneas
    m1 = open("2maquinas_lineas.csv", "r")          % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for b in m1_csv:

psspy.branch_data(float(b[0]),float(b[2]),b[4],[float(b[8]),float(b[9]),float(b[18]
),float(b[20]),float(b[22]),float(b[24])], [float(b[5]), float(b[6]), float(b[7]),
float(b[10]), float(b[11]), float(b[12]), float(b[13]), float(b[14]), float(b[15]),
float(b[16]), float(b[17]),float(b[19]),float(b[21]),float(b[23]),float(b[25])])
    m1.close()                                      % asigna los datos correspondientes
```

```
lineas()                                        % llama a la función para ejecutarla

def plant():                                    % introduce la información de voltaje
    m1 = open("2maquinas_plant.csv", "r")       % preparando las máquinas para
    m1_csv = csv.reader(m1)                     % asignar los valores correspondes
    for p in m1_csv:                            % desde un archivo csv

        psspy.plant_data(float(p[0]),0,[ float(p[7]),100])
    m1.close()                                  % asigna los datos correspondientes
plant()                                         % llama a la función para ejecutarla


def maquinas():                                 % asigna los valores de las maquinas
    m1 = open("2maquinas_maquinas.csv", "r")    % a la hoja network data desde un
    m1_csv = csv.reader(m1)                     % archivo csv
    for m in m1_csv:

psspy.machine_data_2(float(m[0]),r"""1""",[1,1,0,0,0,0],[float(m[7]),0,float(m[11])
,float(m[12]),9999,-9999,float(m[13]),0,float(m[15]),0,0,1,1,1,1,1,1])
    m1.close()                                  % asigna los datos correspondientes
maquinas()                                      % llama a la función para ejecutarla

def transformador():                            % asigna valores del
    m1 = open("2maquinas_transformador.csv", "r") % transformador a la hoja
    m1_csv = csv.reader(m1)                     % network data desde un archivo
    for w in m1_csv:                            % csv

psspy.two_winding_data_3(float(w[0]),float(w[2]),w[4],[float(w[6]),float(w[0]),floa
t(w[24]),float(w[26]),float(w[28]),float(w[30]),33,float(w[45]),float(w[0]),0,1,0,1
,1,1],[float(w[17]),float(w[18]),100,float(w[33]),float(w[34]),float(w[35]),float(w
[36]),float(w[37]),float(w[19]),float(w[20]),float(w[21]),1,1,1,1,float(w[22]),floa
t(w[23]),float(w[38]),float(w[39]),float(w[40]),float(w[41]),0,0,0,],"")
    m1.close()                                  % asigna los datos correspondientes
transformador()                                 % llama a la función para ejecutarla

psspy.newseq()                          % prepara los datos de cortocircuito

def seq():                                      % asigna los datos de cortocircuito
    m1 = open("2maquinas_maquinas.csv", "r")    % a la hoja network data desde un
    m1_csv = csv.reader(m1)                     % archivo csv
    for m in m1_csv:

        psspy.seq_machine_data(float(m[0]),r"""1""",[0,float(m[28]),0,
float(m[30]),0,float(m[32])])                   % asigna los datos correspondientes
    m1.close()
seq()                                           % llama a la función para ejecutarla

psspy.machine_data_2(1,r"""1""",[_i,_i,_i,_i,_i,_i],[_f,_f,_f,_f,_f,_f,_f,_f,
0.265,_f,_f,_f,_f,_f,_f,_f,_f])                 %ajuste para genrou

psspy.seq_machine_data(1,r"""1""",[0,0.265,0, 0.265,0,0.265]) %ajuste para genrou


psspy.fnsl([0,0,0,0,1,1,99,0])  % solución de flujos de potencia Newton Raphson

psspy.cong(0)
psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])  % ajustes de líneas y generadores
psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])  % para análisis dinámico
psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])
```

```
psspy.ordr(0)                                   % se asumen los elementos en
                                                % operación

psspy.fact()
psspy.tysl(1)                                   % solution for swiching studies


psspy.add_plant_model(1,r"""1""",1,r"""GENROU""",0,"",0,[],[],14,[0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]) % añade a la hoja dynamics data los datos
                                                % del genrou

psspy.change_plmod_con(1,r"""1""",r"""GENROU""",1, 0.5871) % modifica los datos
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",2, 0.0248) % de cada uno de
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",3, 0.1351) % los parámetros
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",4, 0.0267)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",5, 2.656)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",6, 2.0)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",7, 1.67)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",8, 1.6)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",9, 0.265)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",10, 0.46)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",11, 0.205)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",12, 0.15)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",13, 0.091)
psspy.change_plmod_con(1,r"""1""",r"""GENROU""",14, 0.4)

psspy.add_plant_model(1,r"""1""",6,r"""IEEET1""",0,"",0,[],[],14,[0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]) % añade a la hoja dynamics data los datos
                                                % del excitador IEEET1

psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",2, 50.0)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",3, 0.06)    % modifica parámetros
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",4, 1.0)     % por los nuevos datos
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",5,-1.0)     % de entrada
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",6,-0.0465)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",7, 0.52)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",8, 0.0832)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",9, 1.0)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",11, 3.24)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",12, 0.072)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",13, 4.32)
psspy.change_plmod_con(1,r"""1""",r"""IEEET1""",14, 0.2821)

psspy.add_plant_model(3,r"""1""",1,r"""GENCLS""",0,"",0,[],[],2,[0.0,0.0])
psspy.change_plmod_con(3,r"""1""",r"""GENCLS""",1, 100.0)
            % introduce datos del gencls dentro de la hoja dynamics data

psspy.chsb(0,1,[-1,-1,-1,1,1,0])                % desactivar parámetros
psspy.chsb(0,1,[-1,-1,-1,1,2,0])                % dejando los necesarios
                                                % a graficar

psspy.strt(0,r"""C:\Documents and Settings\Miguel\Escritorio\macros para dos
máquinas bus infinito\grf.out""") % asigna la dirección y nombre para archivos out

psspy.run(0, 1.0,1,1,0)                         % corre simulación dinámica

psspy.dist_bus_fault(2,1,0.0,[0.0,-0.2E+10])    % agrega la información de falla
                                                % dentro de un bus
```

```
psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\macros para dos máquinas bus infinito\grf.out""")
                                        % refresca la información de las graficas

psspy.run(0, 1.05,1,1,0)                          % corre simulación dinámica

psspy.dist_clear_fault(1)                         % libera la falla

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\macros para dos máquinas bus infinito\grf.out""")
                                         % refresca la información de las graficas

psspy.run(0, 10,1,1,0)                            % corre simulación dinámica

pssplot.newplotbook()                          % abre un nuevo libro de dibujo
pssplot.setselectedpage(0)                     % configura el libro de dibujo activo

pssplot.openchandatafile(r"""C:\Documents and Settings\Miguel\Escritorio\macros
para dos máquinas bus infinito\grf.out""")   % carga un archivo direccionado out


pssplot.dragdropplotdata(r"""grf""",r"""1 - ANGL     1[                  ]1""")
pssplot.dragdropplotdata(r"""grf""",r"""2 - ANGL     3[                  ]1""")
pssplot.dragdropplotdata(r"""grf""",r"""3 - POWR     1[                  ]1""")
pssplot.dragdropplotdata(r"""grf""",r"""4 - POWR     3[                  ]1""")
                    % carga los archivos de salida dentro del libro activo
```

Las diferentes funciones mostradas realizaran los pasos necesarios y al finalizar la ejecución entregaran los resultados y la gráfica en el programa PSSE.

3.4 Caso 3 Sistema 14 Bus de IEEE

Los datos de este caso de estudio que se presentaran a continuación fueron tomados de dos fuentes diferentes, el caso base de 14 Bus Power Flow Test Case, Christie. (1993), y los datos de los generadores de IEEE 14-bus modified test system, Asprou, Quirós-Tortós y Kyriakides (2017). [5][6][10]

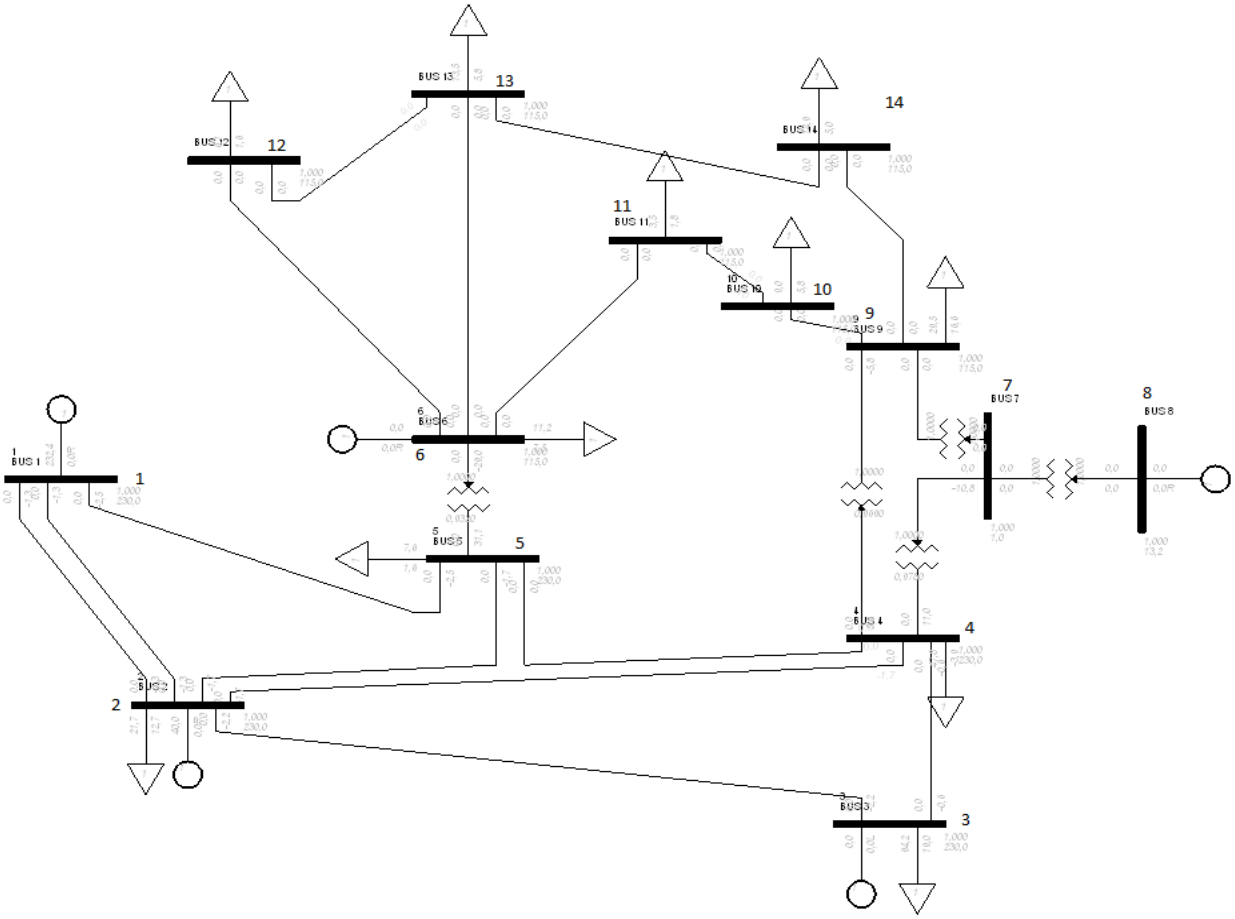

Figura 3.7 Diagrama de 14 barras utilizado dentro de la simulación

Datos de línea			
De Bus a Bus (number)	Impedancia R (pu)	Reactancia X (pu)	B (pu)
1-2	0.03876	0.11834	0.0264
1-2	0.03876	0.11834	0.0264
1-5	0.05403	0.22304	0.0492
2-3	0.04699	0.19797	0.0438
2-4	0.05811	0.17632	0.0340
2-5	0.05695	0.17388	0.0346
3-4	0.06701	0.17103	0.0128
4-5	0.01335	0.04211	0
6-11	0.09498	0.19890	0
6-12	0.12291	0.25581	0
6-13	0.06615	0.13027	0
9-10	0.03181	0.08450	0
9-14	0.12711	0.27038	0
10-11	0.08205	0.19207	0
12-13	0.22092	0.19988	0
13-14	0.17093	0.34802	0

Tabla 3.21 Datos de líneas requeridos para Sistema 14 Bus de IEEE

TENSIÓN CONTROLADA	
Bus (numero)	Tensión (pu)
1	1.060
2	1.045
3	1.010
6	1.070
8	1.090

Tabla 3.22 Datos de tensión controlada requeridos para Sistema 14 Bus de IEEE

Potencia generada		
Bus (numero)	PGen (MW)	QGen (Mvar)
1	-	0
2	40	0
3	0	0
6	0	0
8	0	0

Tabla 3.23 Datos de potencia generada requeridos para Sistema 14 Bus de IEEE

Impedancia en los generadores	
Bus (numero)	X Source (pu)
1	0.265
2	0.22
3	0.343
6	0.3
8	0.3

Tabla 3.24 Datos de impedancia en los generadores requeridos para Sistema 14 Bus de IEEE

Datos de cortocircuito	
Bus (numero)	ZX-Pos, ZX-Neg, ZX-Zero (pu).
1	0.265
2	0.22
3	0.343
6	0.3
8	0.3

Tabla 3.25 Datos de cortocircuito requeridos para Sistema 14 Bus de IEEE

Carga conectada al sistema		
Bus (numero)	Pload (MW)	Qload (Mvar)
2	21.7	12.7
3	94.2	19
4	47.8	-3.9
5	7.6	1.6
6	11.2	7.5
9	29.5	16.6
10	9	5.8
11	3.5	1.8
12	6.1	1.6
13	13.5	5.8
14	14.9	5

Tabla 3.26 Datos de carga conectada al sistema requeridos para Sistema 14 Bus de IEEE

Impedancia de transformador de dos devanados	
De Bus a Bus (number)	Specified X (pu)
4-7	0.20912
4-9	0.55618
5-6	0.25202
7-8	0.17615
7-9	0.11001

Tabla 3.27 Datos de impedancia de transformador de dos devanados requeridos para Sistema 14 Bus de IEEE

Banco de capacitores	
Bus (numero)	B-Shunt (Mvar)
9	19

Tabla 3.28 Datos de bancos de capacitores requeridos para Sistema 14 Bus de IEEE

Tipo de buses	
Bus (numero)	Code
1	3
2	2
3	2
4	1
5	1
6	2
7	1
8	2
9	1
10	1
11	1
12	1
13	1
14	1

Tabla 3.29 Datos de bancos de capacitores requeridos para Sistema 14 Bus de IEEE

Datos dinámicos del generador								
Bus (numero)	Generador	T´do	T´´do	T´q	T´´q	H	D	Xd
1	GENROU	0.5871	0.0248	0.1351	0.0267	2.656	2	1.67
2	GENROU	1.10	0.0277	0.1086	0.0351	4.985	2	1.18
3	GENROU	11.6	0.058	0.159	0.201	1.52	0	2.373
6	GENROU	4.57	0.042	0.5	0.3	5	0	1.9
8	GENROU	4.57	0.042	0.5	0.3	5	0	1.9
Bus (numero)	Xq	X´d	X´q	X´´d=X´´q	Xl	S(1.0)	S(1.2)	
1	1.60	0.265	0.46	0.205	0.15	0.091	0.4	
2	1.05	0.22	0.38	0.145	0.075	0.0933	0.4044	
3	1.172	0.343	1.172	0.231	0.132	0.295	0.776	
6	1.6	0.3	0.7	0.3	0.25	0	0	
8	1.6	0.3	0.7	0.3	0.25	0	0	

Tabla 3.30 Datos dinámicos del generador requeridos para Sistema 14 Bus de IEEE

Datos de los excitadores								
Bus (numero)	Excitador	Tr	Ka	Ta	Vrmax	Vrmin	Ke	Te
1	IEEET1	0	50	0.06	1	-1	-0.0465	0.52
2	IEEET1	0.06	25	0.2	1	-1	-0.0582	0.6544
3	IEEET1	0	400	0.05	6.63	-6.63	-0.17	0.95
6	IEEET1	0	400	0.05	4.407	-4.407	-0.17	0.95
8	IEEET1	0	400	0.05	4.407	-4.407	-0.17	0.95
Bus (numero)	Kf	Tf	Switch	E1	SE(E1)	E2	SE(E2)	
1	0.0832	1	0	3.24	0.072	4.32	0.2821	
2	0.105	0.35	0	2.5785	0.0889	3.438	0.3468	
3	0.04	1	0	6.375	0.2174	8.5	0.9388	
6	0.04	1	0	4.2375	0.2174	5.65	0.9386	
8	0.04	1	0	4.2375	0.2174	5.65	0.9386	

Tabla 3.31 Datos dinámicos del generador requeridos para Sistema 14 Bus de IEEE

Datos de simulación	
Bus donde ocurrirá la falla	13
Tiempo antes de la falla	1
Tiempo de falla	0.05s
Tiempo total, después de falla (liberada)	10s

Tabla 3.32 Datos dinámicos del generador requeridos para Sistema 14 Bus de IEEE

Voltaje nominal del bus		
Bus (numero)	Nombre del bus	kV base
1	BUS 1	230
2	BUS 2	230
3	BUS 3	230
4	BUS 4	230
5	BUS 5	230
6	BUS 6	115
7	BUS 7	1
8	BUS 8	13.2
9	BUS 9	115
10	BUS 10	115
11	BUS 11	115
12	BUS 12	115
13	BUS 13	115
14	BUS 14	115

Tabla 3.33 Datos de voltaje nominal de los bus requeridos para Sistema 14 Bus de IEEE

Una vez listos los datos se generan archivos CSV como base de datos y con ellos se estructurará la programación. Se generan múltiples archivos CSV para ejemplificar de forma más detallada y específica cada proceso en la introducción de datos y ejecución de procesos

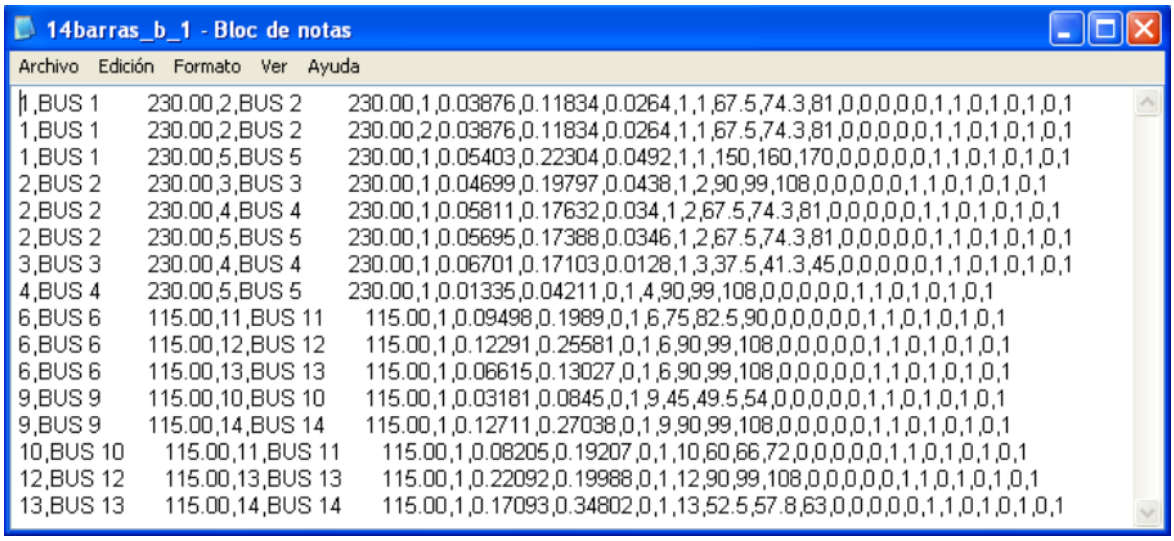

Figura 3.8 Base de datos en CSV para la programación de macros

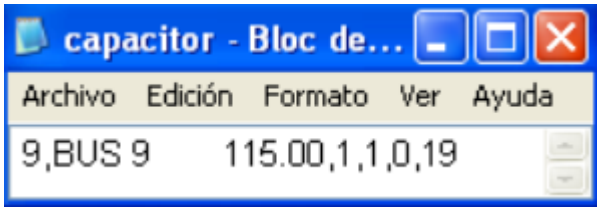

Figura 3.9 Base de datos en CSV para la programación de macros

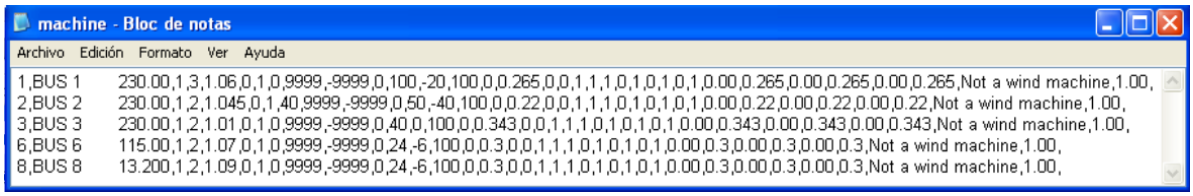

Figura 3.10 Base de datos en CSV para la programación de macros

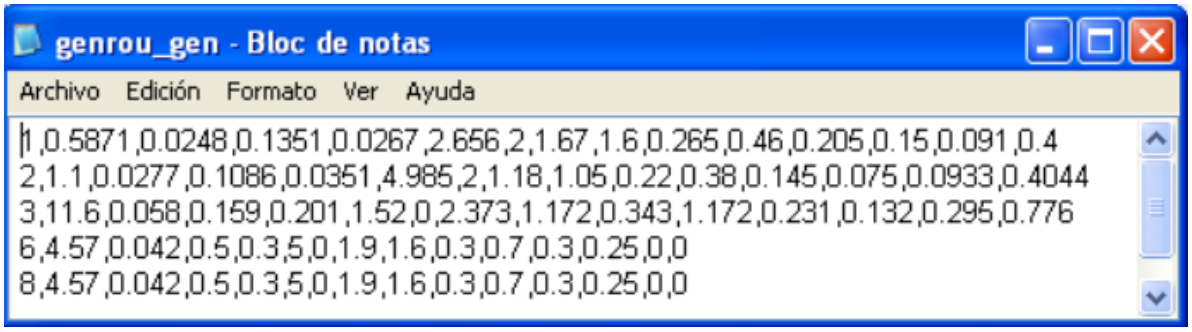

Figura 3.11 Base de datos en CSV para la programación de macros

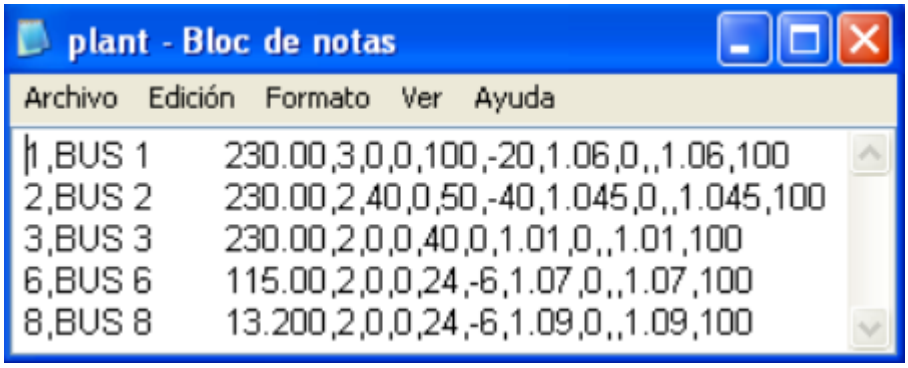

Figura 3.12 Base de datos en CSV para la programación de macros

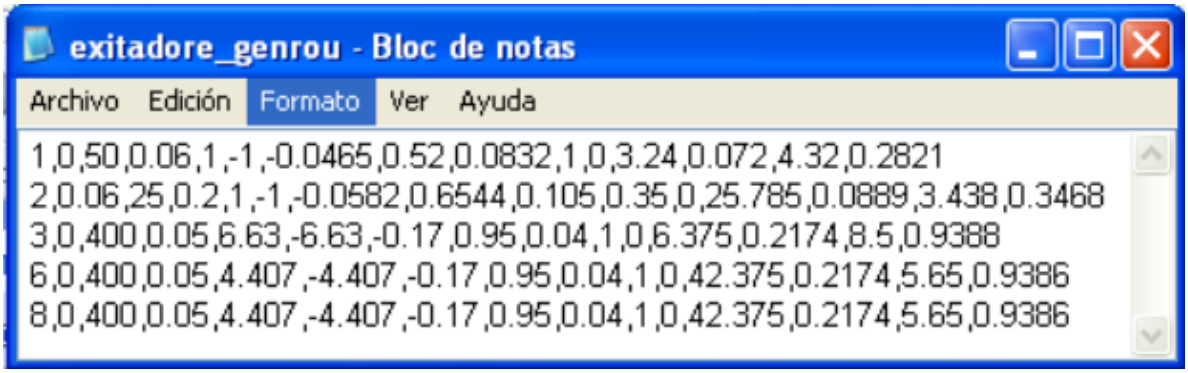

Figura 3.13 Base de datos en CSV para la programación de macros

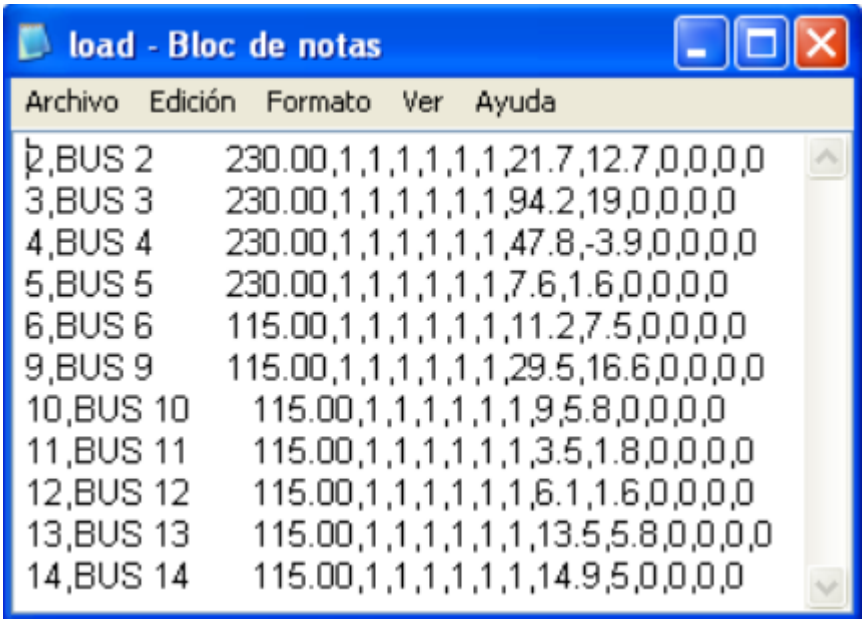

Figura 3.14 Base de datos en CSV para la programación de macros

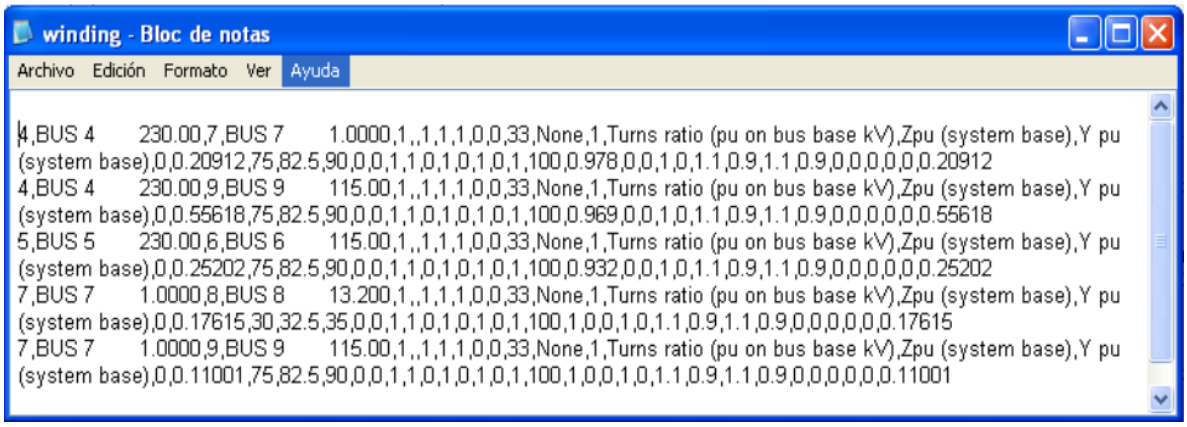

Figura 3.15 Base de datos en CSV para la programación de macros

```
import csv                                          % importar archivos csv
psspy.newdiagfile()                                 % crear nuevo diagrama
psspy.newcase_2([0,0], 100.0, 60.0,"","")           % crear nuevo caso con datos base

def buses():                                        % introduce la información de bus
    m1 = open("14barras.csv", "r")                  % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for v in m1_csv:

psspy.bus_data_2(float(v[0]),[float(v[6]),float(v[3]),float(v[4]),float(v[5])],[flo
at(v[2]),float(v[7]),float(v[8])],str(v[1])) % asigna los datos correspondientes
    m1.close()
buses()                                             %llama a la función para ejecutar


def plant():                                        % introduce la información de voltaje
    m1 = open("plant.csv", "r")                     % preparando las máquinas para
    m1_csv = csv.reader(m1)                         % asignar los valores correspondes
    for p in m1_csv:                                % desde un archivo csv

        psspy.plant_data(float(p[0]),0,[ float(p[7]),100])
    m1.close()                                  % asigna los datos correspondientes
plant()                                         %llama a la función para ejecutar

def machine():                                      % asigna los valores de las maquinas
    m1 = open("machine.csv", "r")                   % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for m in m1_csv:

psspy.machine_data_2(float(m[0]),r"""1""",[1,1,0,0,0,0],[float(m[7]),0,float(m[11])
,float(m[12]),9999,-9999,float(m[13]),0,float(m[15]),0,0,1,1,1,1,1,1])
    m1.close()                                  % asigna los datos correspondientes
machine()                                       %llama a la función para ejecutar

def load():                                         % asigna los valores de las cargas
    m1 = open("load.csv", "r")                      % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for l in m1_csv:

psspy.load_data_3(float(l[0]),r"""1""",[1,1,1,1,1],[float(l[8]),float(l[9]),0.0,0.0
,0.0,0.0])                                      % asigna los datos correspondientes
    m1.close()
load()                                          %llama a la función para ejecutar

def capacitor():                                    % asigna los valores de
    m1 = open("capacitor.csv", "r")                 % los capacitores a la hoja
    m1_csv = csv.reader(m1)                         % network data desde un archivo csv
    for c in m1_csv:

        psspy.shunt_data(float(c[0]),r"""1""",1,[float(c[4]),float(c[5])])
    m1.close()                                  % asigna los datos correspondientes
capacitor()                                     % llama a la función para ejecutar


def lineas():                                       % asigna los valores de
    m1 = open("14barras_b_1.csv", "r")              % las lineas  a la hoja
    m1_csv = csv.reader(m1)                         % network data desde un archivo csv
    for b in m1_csv:
```

```
psspy.branch_data(float(b[0]),float(b[2]),b[4],[float(b[8]),float(b[9]),float(b[18]
),float(b[20]),float(b[22]),float(b[24])], [float(b[5]), float(b[6]), float(b[7]),
float(b[10]), float(b[11]), float(b[12]), float(b[13]), float(b[14]), float(b[15]),
float(b[16]), float(b[17]),float(b[19]),float(b[21]),float(b[23]),float(b[25])])
    m1.close()                          % asigna los datos correspondientes
lineas()                                % llama a la función para ejecutar

psspy.newseq()                  % prepara los datos de cortocircuito

def sequ():                             % asigna los datos de cortocircuito
    m1 = open("machine.csv", "r")       % a la hoja network data desde un
    m1_csv = csv.reader(m1)             % archivo csv
    for m in m1_csv:

        psspy.seq_machine_data(float(m[0]),r"""1""",[0,float(m[28]),0,
float(m[30]),0,float(m[32])])       % asigna los datos correspondientes
    m1.close()
sequ()                                  % llama a la función para ejecutar

def transformador():                    % asigna valores del transformador
    m1 = open("winding.csv", "r")       % a la hoja network data desde un
    m1_csv = csv.reader(m1)             % archivo csv
    for w in m1_csv:

psspy.two_winding_data_3(float(w[0]),float(w[2]),w[4],[float(w[6]),float(w[0]),floa
t(w[24]),float(w[26]),float(w[28]),float(w[30]),33,float(w[45]),float(w[0]),0,1,0,1
,1,1],[float(w[17]),float(w[18]),100,float(w[33]),float(w[34]),float(w[35]),float(w
[36]),float(w[37]),float(w[19]),float(w[20]),float(w[21]),1,1,1,1,float(w[22]),floa
t(w[23]),float(w[38]),float(w[39]),float(w[40]),float(w[41]),0,0,0,],"")
    m1.close()                          % asigna los datos correspondientes
transformador()                         % llama a la función para ejecutar

psspy.fnsl([0,0,0,0,1,1,99,0])  % solución de flujos de potencia Newton Raphson

psspy.cong(0)
psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])  % ajustes de líneas y generadores
psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])  % para análisis dinámico
psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])

psspy.ordr(0)                                    % se asumen los elementos en
                                                 % operación

psspy.fact()
psspy.tysl(1)                                    % solution for swiching studies

def genrou():                                    % asigna los valores del modelo
    m1 = open("genrou_gen.csv", "r")             % del generador genrou desde
    m1_csv = csv.reader(m1)                      % un archivo csv
    for v in m1_csv:

psspy.add_plant_model(float(v[0]),r"""1""",1,r"""GENROU""",0,"",0,[],[],14,[float(v
[1]),float(v[2]),float(v[3]),float(v[4]),float(v[5]),float(v[6]),float(v[7]),float(
v[8]),float(v[9]),float(v[10]),float(v[11]),float(v[12]),float(v[13]),float(v[14])]
)                                       % asigna los datos correspondientes
    m1.close()
genrou()                                % llama a la función para ejecutar
```

```
def excitadores():                                  % asigna los valores del modelo
    m1 = open("exitadore_genrou.csv", "r")          % de excitador dentro de la hoja
    m1_csv = csv.reader(m1)                         % dynamics data desde un archivo
    for v in m1_csv:                                % csv

psspy.add_plant_model(float(v[0]),r"""1""",6,r"""IEEET1""",0,"",0,[],[],14,[float(v
[1]),float(v[2]),float(v[3]),float(v[4]),float(v[5]),float(v[6]),float(v[7]),float(
v[8]),float(v[9]),float(v[10]),float(v[11]),float(v[12]),float(v[13]),float(v[14])]
)                                       % asigna los datos correspondientes
    m1.close()
excitadores()                            % llama a la función para ejecutar

psspy.chsb(0,1,[-1,-1,-1,1,1,0])                % desactivar parámetros
psspy.chsb(0,1,[-1,-1,-1,1,2,0])                % dejando los necesarios
                                                % a graficar

psspy.strt(0,r"""C:\Documents and Settings\Miguel\Escritorio\ejemplo de catorce
barras  que yaa sirve\14grf.out""")% asigna la dirección y nombre para archivos out

psspy.run(0, 1.0,1,1,0)                         % corre simulación dinámica

psspy.dist_bus_fault(13,1,0.0,[0.0,-0.2E+10])   % añade la falla al bus

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\ejemplo de catorce barras  que yaa sirve\14grf.out""")
                                                % refresca la información

psspy.run(0, 1.05,1,1,0)                        % corre la simulación dinámica

psspy.dist_clear_fault(1)                       % libera la falla del bus

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\ejemplo de catorce barras que yaa sirve\14grf.out""")
                                                % refresca la información

psspy.run(0, 10,1,1,0)                          % corre la simulación dinámica

pssplot.newplotbook()                       % abre un nuevo libro de dibujo
pssplot.setselectedpage(0)                  % configura el libro de dibujo activo

pssplot.openchandatafile(r"""C:\Documents and Settings\Miguel\Escritorio\ejemplo de
catorce barras  que yaa sirve\14grf.out""")  % carga un archivo direccionado out

pssplot.dragdropplotdata(r"""14grf""",r"""1 - ANGL     1[BUS 1       230.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""2 - ANGL     2[BUS 2       230.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""3 - ANGL     3[BUS 3       230.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""4 - ANGL     6[BUS 6       115.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""5 - ANGL     8[BUS 8       13.200]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""6 - POWR     1[BUS 1       230.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""7 - POWR     2[BUS 2       230.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""8 - POWR     3[BUS 3       230.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""9 - POWR     6[BUS 6       115.00]1""")
pssplot.dragdropplotdata(r"""14grf""",r"""10 - POWR     8[BUS 8       13.200]1""")
                        % carga los archivos de salida dentro del libro
```

3.5 Caso 4 Sistema 9 Buses de IEEE

Los datos de este caso de estudio que se presentaran a continuación fueron tomados de dos fuentes diferentes, el caso base del libro Power System Control and Stability, Anderson y Fouda. (1977), y los datos de los generadores de IEEE 9-bus modified test system, Asprou, Quirós-Tortós y Kyriakides (2017).[7][8][10]

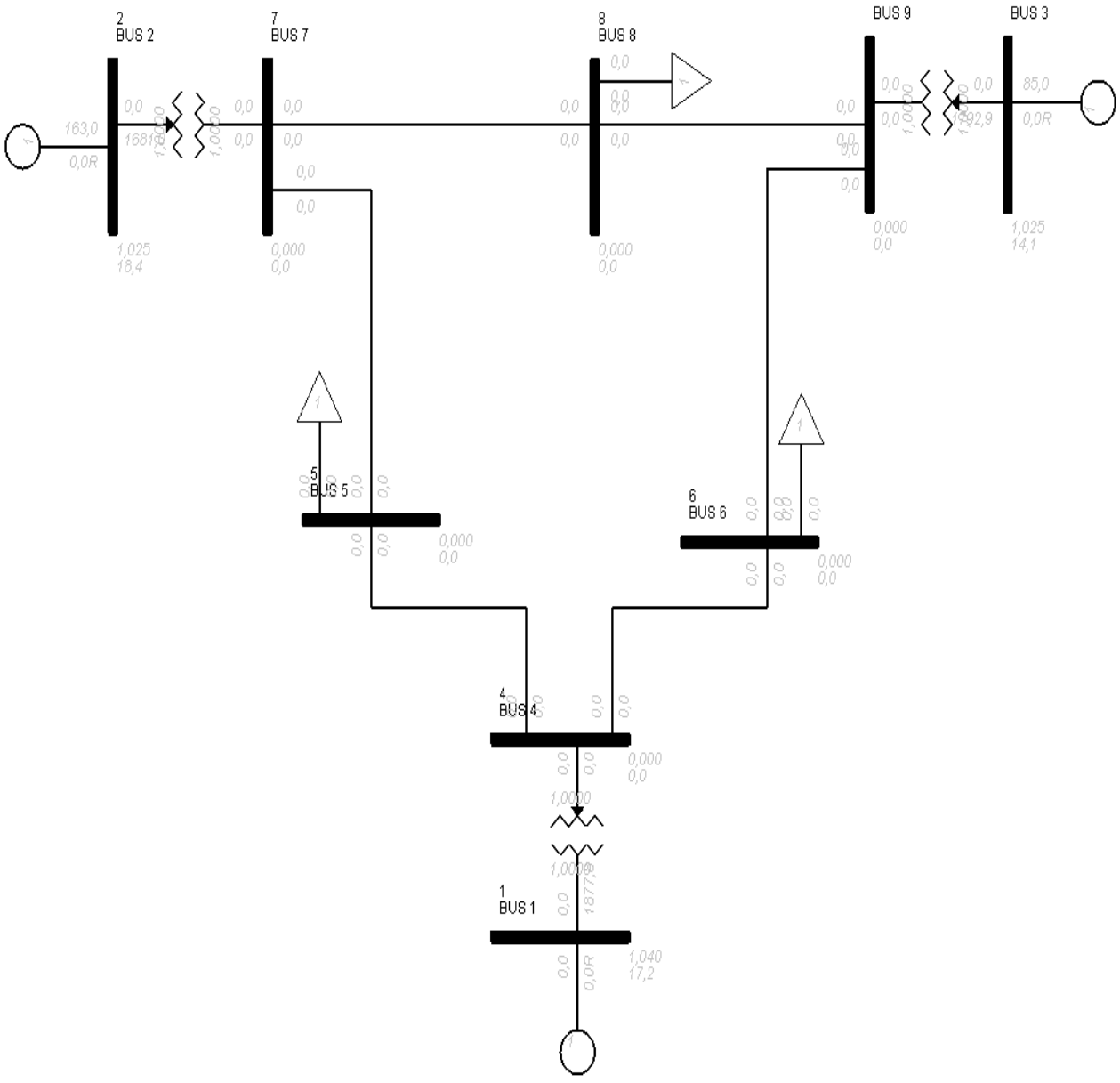

Figura 3.16 Diagrama utilizado para el Sistema 9 Buses de IEEE

Datos de líneas			
De Bus a Bus (numero)	Impedancia R (pu)	Reactancia X (pu)	B (pu)
4-5	0.0100	0.0850	0.176
4-6	0.0170	0.0920	0.158
5-7	0.0320	0.1610	0.306
6-9	0.0390	0.1700	0.358
7-8	0.0085	0.0720	0.149
8-9	0.0119	0.1008	0.209

Tabla 3.34 Datos de líneas requeridos para Sistema 9 Buses de IEEE

Tensión controlada	
Bus (numero)	Tensión (pu)
1	1.040
2	1.025
3	1.025

Tabla 3.35 Datos de tensión controlada requeridos para Sistema 9 Buses de IEEE

Potencia generada		
Bus (numero)	PGen (MW)	QGen (Mvar)
1	0	0
2	163	0
3	85	0

Tabla 3.36 Datos de potencia generada requeridos para Sistema 9 Buses de IEEE

Impedancia de los generadores	
Bus (numero)	X Source (pu)
1	0.0608
2	0.1198
3	0.1813

Tabla 3.37 Datos de impedancia de los generadores requeridos para Sistema 9 Buses de IEEE

Datos de cortocircuito en generadores	
Bus (numero)	ZX-Pos, ZX-Neg, ZX-Zero (pu).
1	0.0608
2	0.1198
3	0.1813

Tabla 3.38 Datos de cortocircuito en generadores requeridos para Sistema 9 Buses de IEEE

Datos de carga		
Bus (numero)	Pload (MW)	Qload (Mvar)
5	125	50
6	90	30
8	100	35

Tabla 3.39 Datos de carga requeridos para Sistema 9 Buses de IEEE

Impedancia de dos devanados	
De Bus a Bus (number)	Specified X (pu)
1-4	0.0576
2-7	0.0625
3-9	0.0586

Tabla 3.40 Datos de impedancia de dos devanados requeridos para Sistema 9 Buses de IEEE

Tipo de buses	
Bus (numero)	Code
1	3
2	2
3	2
4	1
5	1
6	1
7	1
8	1
9	1

Tabla 3.41 Datos de tipos de buses requeridos para Sistema 9 Buses de IEEE

Datos dinámicos del generador			
Bus (numero)	Mbase (MVA)	Generador	H
1	100	GENCLS	23.64
2	100	GENCLS	6.4
3	10000000	GENCLS	3.01

Tabla 3.42 Datos dinámicos del generador requeridos para Sistema 9 Buses de IEEE

Datos de simulación	
Bus donde ocurrirá la falla	7
Tiempo antes de la falla	1s
Tiempo de falla	0.08334s (la falla dura 5 ciclos)
Línea que se libera	5-7
Tiempo total, después de falla (liberada)	3.0s

Tabla 3.43 Datos dinámicos de simulación requeridos para Sistema 9 Buses de IEEE

Tensión nominal de los buses		
Bus (numero)	Pload (MW)	Qload (Mvar)
1	BUS 1	16.5
2	BUS 2	18
3	BUS 3	13.8
4	BUS 4	230
5	BUS 5	230
6	BUS 6	230
7	BUS 7	230
8	BUS 8	230
9	BUS 9	230

Tabla 3.44 Datos de tensión nominal de los buses requeridos para Sistema 9 Buses de IEEE

Una vez listos los datos se generan archivos CSV como base de datos y con ellos se estructurará la programación. Se generan múltiples archivos CSV para ejemplificar de forma más detallada y específica cada proceso en la introducción de datos y ejecución de procesos.

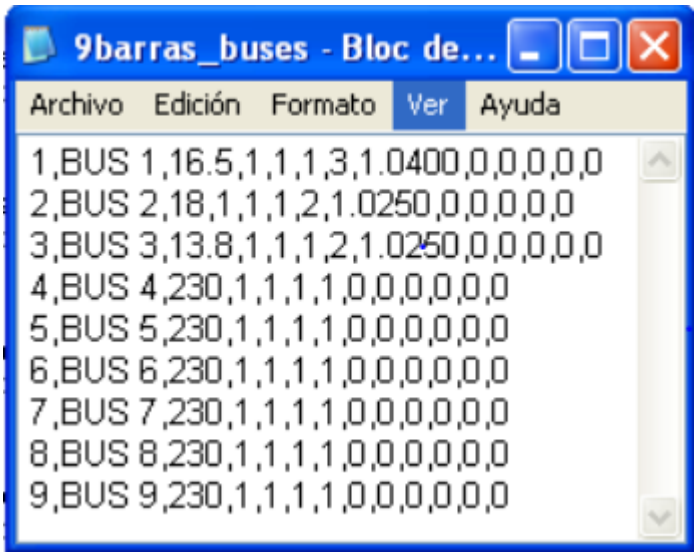

Figura 3.17 Base de datos en CSV para la programación de macros

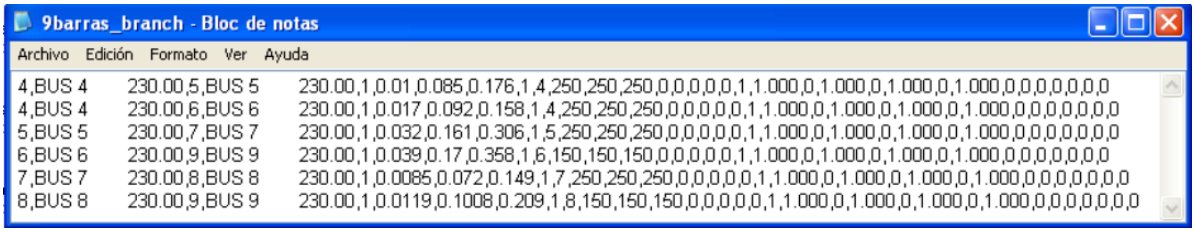

Figura 3.18 Base de datos en CSV para la programación de macros

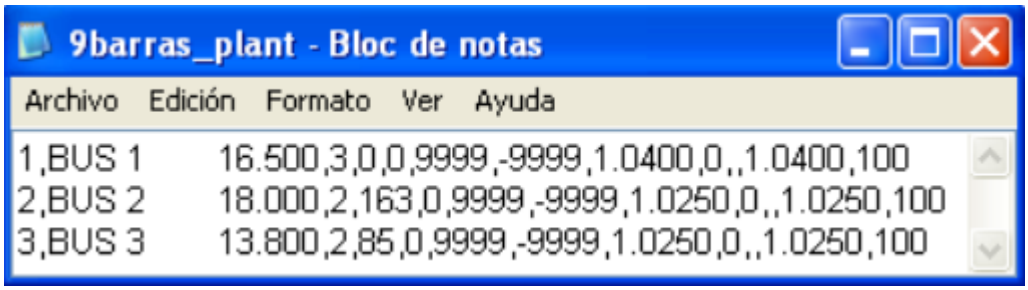

Figura 3.19 Base de datos en CSV para la programación de macros

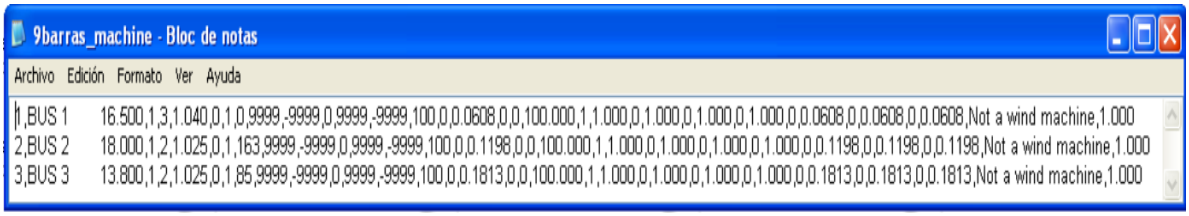

Figura 3.20 Base de datos en CSV para la programación de macros

Figura 3.21 Base de datos en CSV para la programación de macros

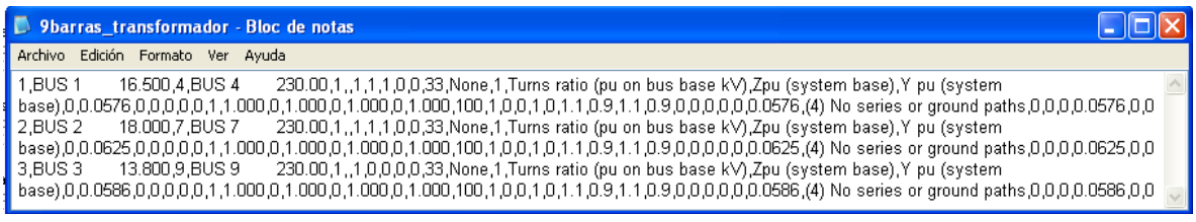

Figura 3.22 Base de datos en CSV para la programación de macros

```
import csv                                          % importar archivos csv
psspy.newdiagfile()                                 % crear nuevo diagrama
psspy.newcase_2([0,0], 100.0, 60.0,"","")           % crear nuevo caso con datos base

def buses():                                        % introduce la información de bus
    m1 = open("9barras_buses.csv", "r")             % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for v in m1_csv:

psspy.bus_data_2(float(v[0]),[float(v[6]),float(v[3]),float(v[4]),float(v[5])],[flo
at(v[2]),float(v[7]),float(v[8])],str(v[1])) % asigna los datos correspondientes
    m1.close()
buses()                                             %llama a la función para ejecutar

def lineas():                                       % introduce la información de lineas
    m1 = open("9barras_branch.csv", "r")            % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for b in m1_csv:

psspy.branch_data(float(b[0]),float(b[2]),b[4],[float(b[8]),float(b[9]),float(b[18]
),float(b[20]),float(b[22]),float(b[24])], [float(b[5]), float(b[6]), float(b[7]),
float(b[10]), float(b[11]), float(b[12]), float(b[13]), float(b[14]), float(b[15]),
float(b[16]), float(b[17]),float(b[19]),float(b[21]),float(b[23]),float(b[25])])
    m1.close()                                      % asigna los datos correspondientes
lineas()                                            % llama a la función para ejecutar

def plant():                                        % introduce la información de voltaje
    m1 = open("9barras_plant.csv", "r")             % preparando las máquinas para
    m1_csv = csv.reader(m1)                         % asignar los valores correspondes
    for p in m1_csv:                                % desde un archivo csv

        psspy.plant_data(float(p[0]),0,[ float(p[7]),100])
    m1.close()                                      % asigna los datos correspondientes
plant()                                             %llama a la función para ejecutar


def machine():                                      % asigna los valores de las maquinas
    m1 = open("9barras_machine.csv", "r")           % a la hoja network data desde un
    m1_csv = csv.reader(m1)                         % archivo csv
    for m in m1_csv:

psspy.machine_data_2(float(m[0]),r"""1""",[1,1,0,0,0,0],[float(m[7]),0,float(m[11])
,float(m[12]),9999,-9999,float(m[13]),0,float(m[15]),0,0,1,1,1,1,1,1])
    m1.close()                                      % asigna los datos correspondientes
machine()                                           % llama a la función para ejecutar

def transformador():                                % asigna los valores del
    m1 = open("9barras_transformador.csv", "r")     % transformador a la hoja network
    m1_csv = csv.reader(m1)                         % data desde un archivo csv
    for w in m1_csv:

psspy.two_winding_data_3(float(w[0]),float(w[2]),w[4],[float(w[6]),float(w[0]),floa
t(w[24]),float(w[26]),float(w[28]),float(w[30]),33,float(w[45]),float(w[0]),0,1,0,1
,1,1],[float(w[17]),float(w[18]),100,float(w[33]),float(w[34]),float(w[35]),float(w
[36]),float(w[37]),float(w[19]),float(w[20]),float(w[21]),1,1,1,1,float(w[22]),floa
t(w[23]),float(w[38]),float(w[39]),float(w[40]),float(w[41]),0,0,0,],"")
    m1.close()                                      % asigna los datos correspondientes
transformador()                                     % llama a la función para ejecutar
```

```
def load():                                    % asigna los valores de las cargas
    m1 = open("9barras_load.csv", "r")         % a la hoja network data desde un
    m1_csv = csv.reader(m1)                    % archivo csv
    for l in m1_csv:

psspy.load_data_3(float(l[0]),r"""1""",[1,1,1,1,1],[float(l[8]),float(l[9]),0.0,0.0
,0.0,0.0])                                     % asigna los datos correspondientes
    m1.close()
load()                                         % llama a la función para ejecutar

psspy.newseq()                  % prepara los datos de cortocircuito

def sequ():                                    % asigna los valores de cortocircuito
    m1 = open("9barras_machine.csv", "r")      % a la hoja network data desde un
    m1_csv = csv.reader(m1)                    % archivo csv
    for m in m1_csv:

        psspy.seq_machine_data(float(m[0]),r"""1""",[0,float(m[28]),0,
float(m[30]),0,float(m[32])])                  % asigna los datos correspondientes
    m1.close()
sequ()                                         % llama a la función para ejecutar

psspy.fnsl([0,0,0,0,1,1,99,0])  % solución de flujos de potencia Newton Raphson

psspy.cong(0)
psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])  % ajustes de líneas y generadores
psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])  % para análisis dinámico
psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])

psspy.ordr(0)                                  % se asumen los elementos en
                                               % operación

psspy.fact()
psspy.tysl(1)                                  % solution for swiching studies

psspy.add_plant_model(1,r"""1""",1,r"""GENCLS""",0,"",0,[],[],2,[0. 0,0.0])
psspy.change_plmod_con(1,r"""1""",r"""GENCLS""",1, 23.64)
psspy.add_plant_model(2,r"""1""",1,r"""GENCLS""",0,"",0,[],[],2,[0.0,0.0])
psspy.change_plmod_con(2,r"""1""",r"""GENCLS""",1, 6.4)
psspy.add_plant_model(3,r"""1""",1,r"""GENCLS""",0,"",0,[],[],2,[0.0,0.0])
psspy.change_plmod_con(3,r"""1""",r"""GENCLS""",1, 3.01)
              % introduce datos del gencls dentro de la hoja dynamics data

psspy.chsb(0,1,[-1,-1,-1,1,1,0])               % desactivar parámetros
psspy.chsb(0,1,[-1,-1,-1,1,2,0])               % dejando los necesarios
                                               % a graficar

psspy.strt(0,r"""C:\Documents and
Settings\Miguel\Escritorio\9barras\lineafuera.out""")
                            % asigna la dirección y nombre para archivos out

psspy.run(0, 1.0,1,1,0)                        % corre simulación dinámica

psspy.dist_bus_fault(7,1, 230.0,[0.0,-0.2E+10])  % añade la falla al bus

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\9barras\lineafuera.out""") % refresca la información

psspy.run(0, 1.0833,1,1,0)                     % corre simulación dinámica
```

```
psspy.dist_clear_fault(1)                          % libera la falla

psspy.dist_branch_trip(5,7,r"""1""")               %libera la línea

psspy.change_channel_out_file(r"""C:\Documents and
Settings\Miguel\Escritorio\9barras\lineafuera.out""") % refresca la información

psspy.run(0, 3.0,1,1,0)                            % corre simulación dinámica

pssplot.newplotbook()                          % abre un nuevo libro de dibujo
pssplot.setselectedpage(0)                     % configura el libro de dibujo activo

pssplot.openchandatafile(r"""C:\Documents and
Settings\Miguel\Escritorio\9barras\lineafuera.out""")
                                  % carga un archivo direccionado out

pssplot.dragdropplotdata(r"""lineafuera""",r"""1 - ANGL     1[BUS 1
16.500]1""")
pssplot.dragdropplotdata(r"""lineafuera""",r"""2 - ANGL     2[BUS 2
18.000]1""")
pssplot.dragdropplotdata(r"""lineafuera""",r"""3 - ANGL     3[BUS 3
13.800]1""")
pssplot.dragdropplotdata(r"""lineafuera""",r"""4 - POWR     1[BUS 1
16.500]1""")
pssplot.dragdropplotdata(r"""lineafuera""",r"""5 - POWR     2[BUS 2
18.000]1""")
pssplot.dragdropplotdata(r"""lineafuera""",r"""6 - POWR     3[BUS 3
13.800]1""")
                  % carga los archivos de salida dentro del libro
```

Capítulo 4 Resultados

4.1 Introducción

Cuando se realizan los estudios dinámicos en sistemas eléctricos de potencia es necesario brindar todos los recursos necesarios para la interpretación de los resultados, el siguiente capítulo aporta las gráficas de potencia y ángulo de respuesta dinámica, de igual manera las tablas de flujos de potencia que se obtuvieron en los diferentes ejemplos anteriormente explicados, teniendo como objetivo mostrar la eficiencia de las macros dentro de los análisis en sistemas eléctricos de potencia.

4.2 Caso1 Sistema Bus Infinito con generador tipo (GENCLS)

Se presentan los resultados después de utilizar las macros antes descritas, con fines de mostrar los datos obtenidos por PSS/E después de la simulación y verificar que para los ejemplos puedan ser útiles, y el usuario que las utilizara en sus análisis dinámicos tenga la información requerida.

Resultados del network después el análisis dinámico.

	Bus Number	Bus Name	Id	Code	VSched (pu)	Remote Bus	In Service	Pgen (MW)	Pmax (MW)	Pmin (MW)	Qgen (Mvar)
	1		1	2	1.0000	0	☑	100.0000	9999.0000	-9999.0000	15.3536
	3		1	2	1.0000	0	☑	-100.0000	9999.0000	-9999.0000	-0.0000
*							☑				

Figura 4.1 Resultados del network después el análisis dinámico

Resultados del network después el análisis dinámico.

	Bus Number	Bus Name	Base kV	Area Number/Name	Zone Number/Name	Owner Number/Name	Code	Voltage (pu)	Angle (deg)
	1		0.0	1	1	1	2	1.0160	11.854
	2		0.0	1	1	1	1	1.0059	7.951
	3		0.0	1	1	1	2	1.0000	-0.000
*									

Figura 4.2 Resultados del network después el análisis dinámico

Observaciones dentro de las gráficas potencia de generadores y ángulos en sincronismo:

En la Figura 4.3 se puede observar la maquina en sincronismo con una potencia constante durante el tiempo de simulación antes de ocurrir la perturbación del sistema, para este caso

la falla en el bus 2. Cuando se genera la perturbación se puede apreciar que la potencia suministrada en el sistema cae a cero sin embargo la potencia generada sigue aumentándose, esto causa que se genere una desigualdad entre la potencia mecánica del motor y la potencia eléctrica del mismo por lo cual la potencia de aceleración comienza a variar oscilando y provocando que la potencia eléctrica también comience una oscilación, esto se puede apreciar durante todo el tiempo en el que ocurre la falla hasta el momento de la liberación. Después de generarse la liberación de la falla, la potencia eléctrica continuara con las oscilaciones que se generan a partir de una falta de par de amortiguamiento.

En la Figura 4.4 se puede apreciar durante el tiempo antes de la falla como es que se mantiene constante el ángulo hasta el momento de la perturbación. Después de generarse la perturbación en el sistema el ángulo comienza a aumentar debido a la potencia de aceleración provocada por la potencia eléctrica de igual manera un factor importante es la constante de inercia que determinara la velocidad con la que aumentara el valor de dicho ángulo, después de aplicarse la liberación de la falla, el ángulo tardara unos ciclos en mostrar sincronismo tomando un valor máximo con el que las siguientes oscilaciones se verán reflejadas si no es que se genera un amortiguamiento.

Con los datos obtenidos gracias a las macros podemos ir modificando ciertas características para hacer más eficiente el programa realizado, las macros realizadas siempre arrojaran los resultados de las gráficas para poder observar el comportamiento en los cambios de estado del sistema, de esta forma es de suma importancia tener en cuenta que debemos de contar con un conocimiento previo sobre los comportamientos mencionados, para poder tener una mayor comprensión de la relación programación-resolución del sistema descrito.

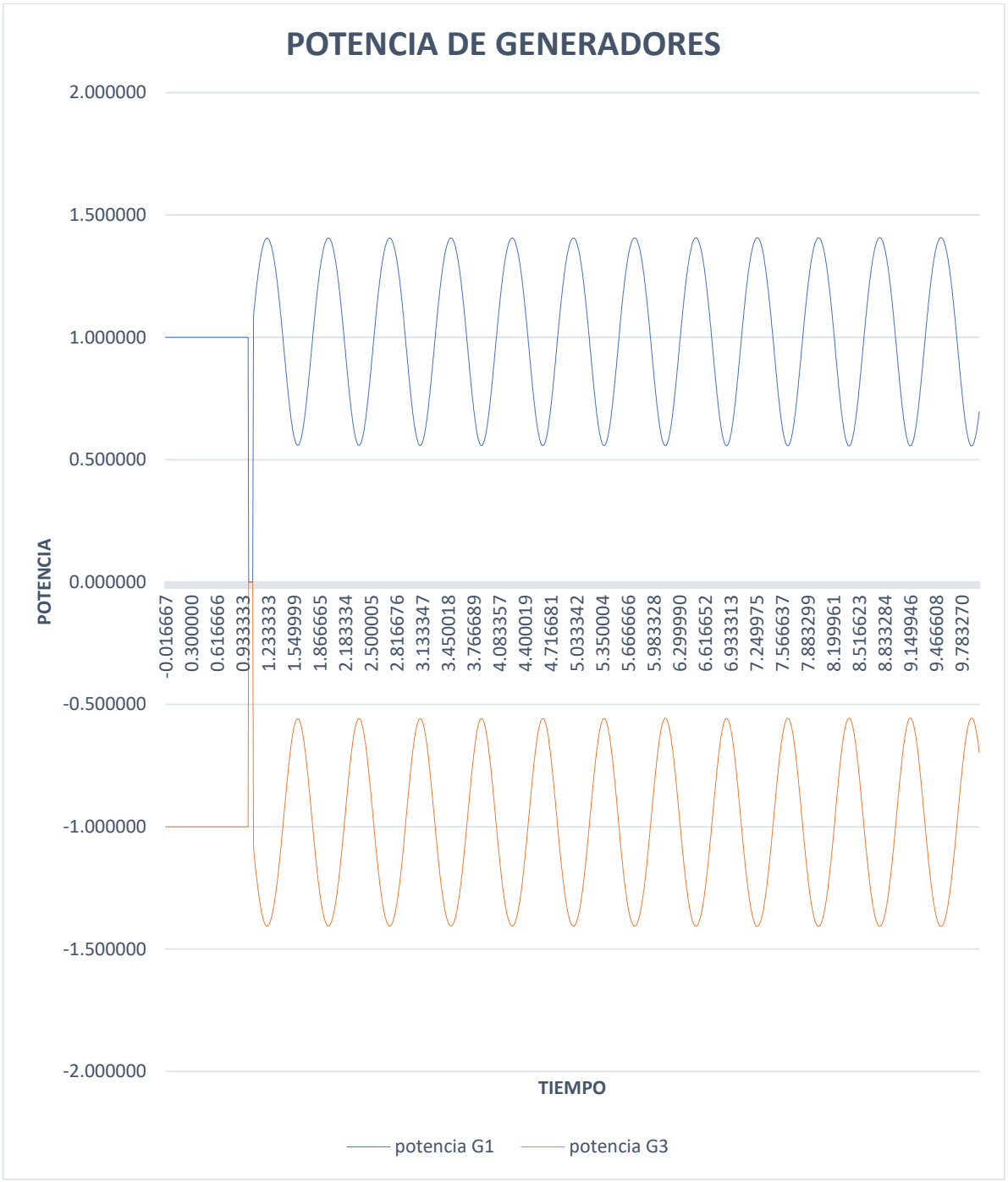

Figura 4.3 Grafica de potencia, generada con los datos arrojados por las macros después de la simulación del Sistema Bus Infinito con generador tipo (GENCLS)

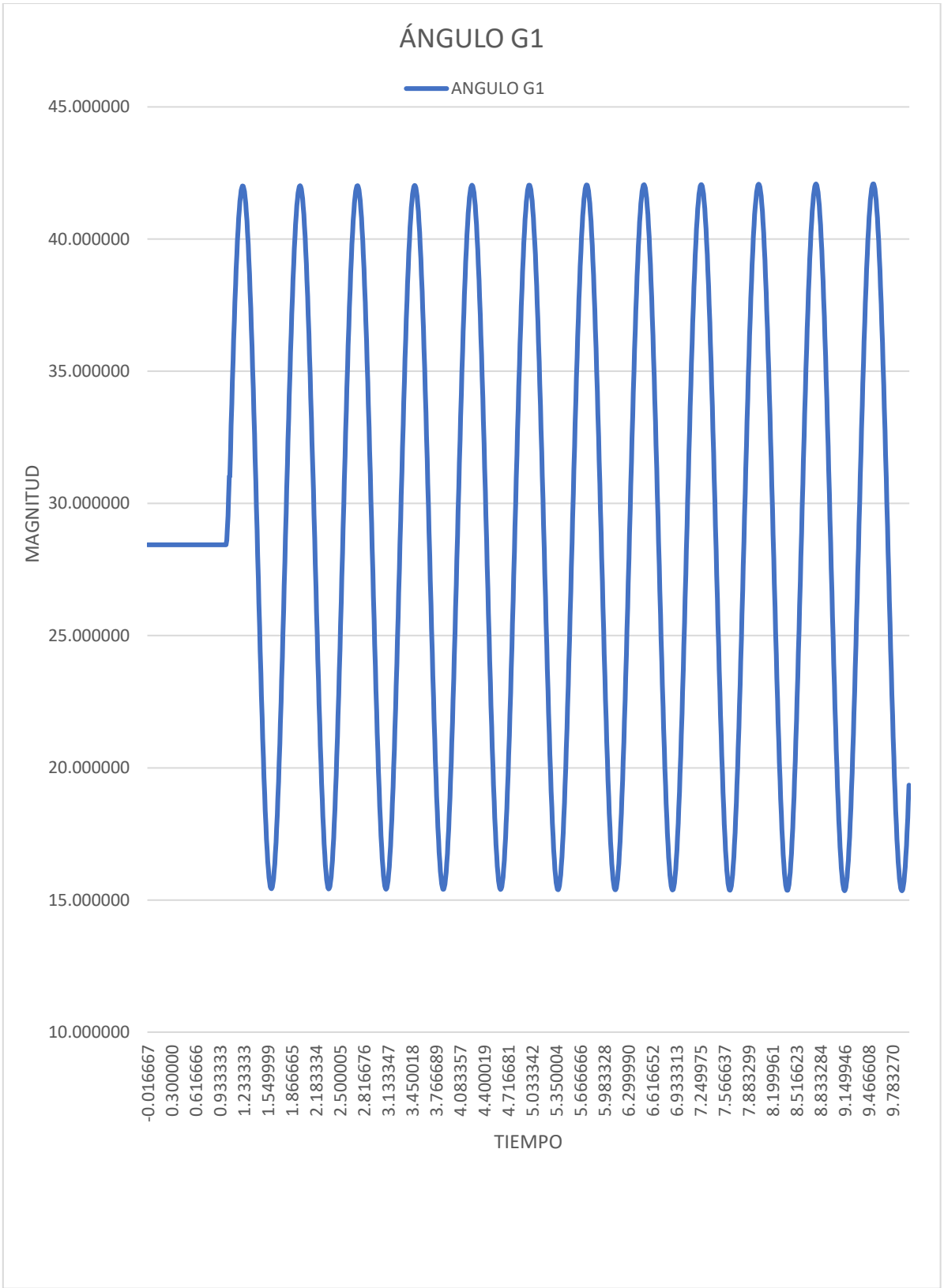

Figura 4.4 Grafica de ángulo del generadorG1, generada con los datos arrojados por las macros después de la simulación del Sistema Bus Infinito con generador tipo (GENCLS)

4.3 Caso 2 Sistema Bus Infinito con generador tipo (GENCLS Y GENROU)

Se presentan los resultados después de utilizar las macros antes descritas, con fines de mostrar los datos obtenidos por PSS/E después de la simulación y verificar que para los ejemplos puedan ser útiles, y el usuario que las utilizara en sus análisis dinámicos tenga la información requerida.

Resultados del network después el análisis dinámico.

	Bus Number	Bus Name	Base kV	Area Number/Name	Zone Number/Name	Owner Number/Name	Code	Voltage (pu)	Angle (deg)
	1		0.0	1	1	1	2	1.0000	17.457
	2		0.0	1	1	1	1	0.9897	11.658
	3		0.0	1	1	1	2	1.0000	0.000

Figura 4.5 Resultados del network después el análisis dinámico

Resultados del network después el análisis dinámico.

	Bus Number	Bus Name	Id	Code	VSched (pu)	Remote Bus	In Service	Pgen (MW)	Pmax (MW)	Pmin (MW)	Qgen (Mvar)
	1		1	2	1.0000	0	☑	100.0000	9999.0000	-9999.0000	15.3536
	3		1	2	1.0000	0	☑	-100.0000	9999.0000	-9999.0000	0.0000
*							☑				

Figura 4.6 Resultados del network después el análisis dinámico

Observaciones dentro de las gráficas potencia de generadores y ángulos en sincronismo:

En la Figura 4.7 se puede observar la máquina en sincronismo con una potencia constante durante el tiempo de simulación antes de ocurrir la perturbación del sistema, para este caso la falla en el bus 2 cuando se genera la perturbación se puede apreciar que la potencia suministrada en el sistema cae a cero, sin embargo la potencia generada sigue aumentando, esto causa que se genere una desigualdad entre la potencia mecánica del motor y la potencia eléctrica del mismo por lo cual la potencia de aceleración comienza a variar oscilando y provocando que la potencia eléctrica también comience una oscilación, esto se puede apreciar durante todo el tiempo en el que ocurre la falla hasta el momento de la liberación. Después de generarse la liberación de la falla la potencia eléctrica continuara con las oscilaciones que se generan a partir de una falta de par de amortiguamiento.

En la gráfica de la Figura 4.8 se puede apreciar durante el tiempo antes de la falla como es que se mantiene constante el ángulo hasta el momento de la perturbación, después de

generarse la perturbación en el sistema el ángulo comienza a aumentar debido a la potencia de aceleración provocada por la potencia eléctrica de igual manera un factor importante es la constante de inercia que determinara la velocidad con la que aumentara el valor de dicho ángulo, en este caso donde el tipo de generador cambia se tienen parámetros más específicos con los cuales se trabajara dentro del programa para el tipo de maquia deseado, después de aplicarse la liberación de la falla, el ángulo tardara unos ciclos en mostrar sincronismo tomando un valor máximo con el que las siguientes oscilaciones se verán reflejadas. Al tener un sistema de excitación se puede apreciar cómo se genera un amortiguamiento en las oscilaciones que tiende a recuperar el punto de operación similar al anterior o uno cercano al mismo.

Dentro del sistema se agrega un control de tipo excitador IEEET1, este sistema de excitación se encarga de brindar corriente directa al devanado de campo de la máquina. La función de control de este sistema de excitación son modificar el comportamiento del voltaje y el flujo de potencia reactiva puesto que es el principal sistema de control encargado de mantener la estabilidad en el sistema analizado porque actúa de forma más rápida que otro sistema de control.[9]

Una vez resuelto el caso con GENROU podemos apreciar que trabajar con valores distintos en los generadores y controles nos arrojara distintos valores, aunque se generen las mismas contingencias, por esa razón es importante el modelaje con ciertas características distintas entre un caso y otro, y con macros se podría cambiar los valores a simular de una manera más rápida.

Con los datos obtenidos gracias a las macros podemos ir modificando ciertas características para hacer más eficiente el programa realizado, las macros realizadas siempre arrojaran los resultados de las gráficas para poder observar el comportamiento en los cambios de estado del sistema, de esta forma es de suma importancia tener en cuenta que debemos de contar con un conocimiento previo sobre los comportamientos mencionados, para poder tener una mayor comprensión de la relación programación-resolución del sistema descrito.

Figura 4.7 Grafica de potencia en G1, G2, generada con los datos arrojados por las macros después de la simulación del Sistema Bus Infinito con generador tipo (GENCLS Y GENROU)

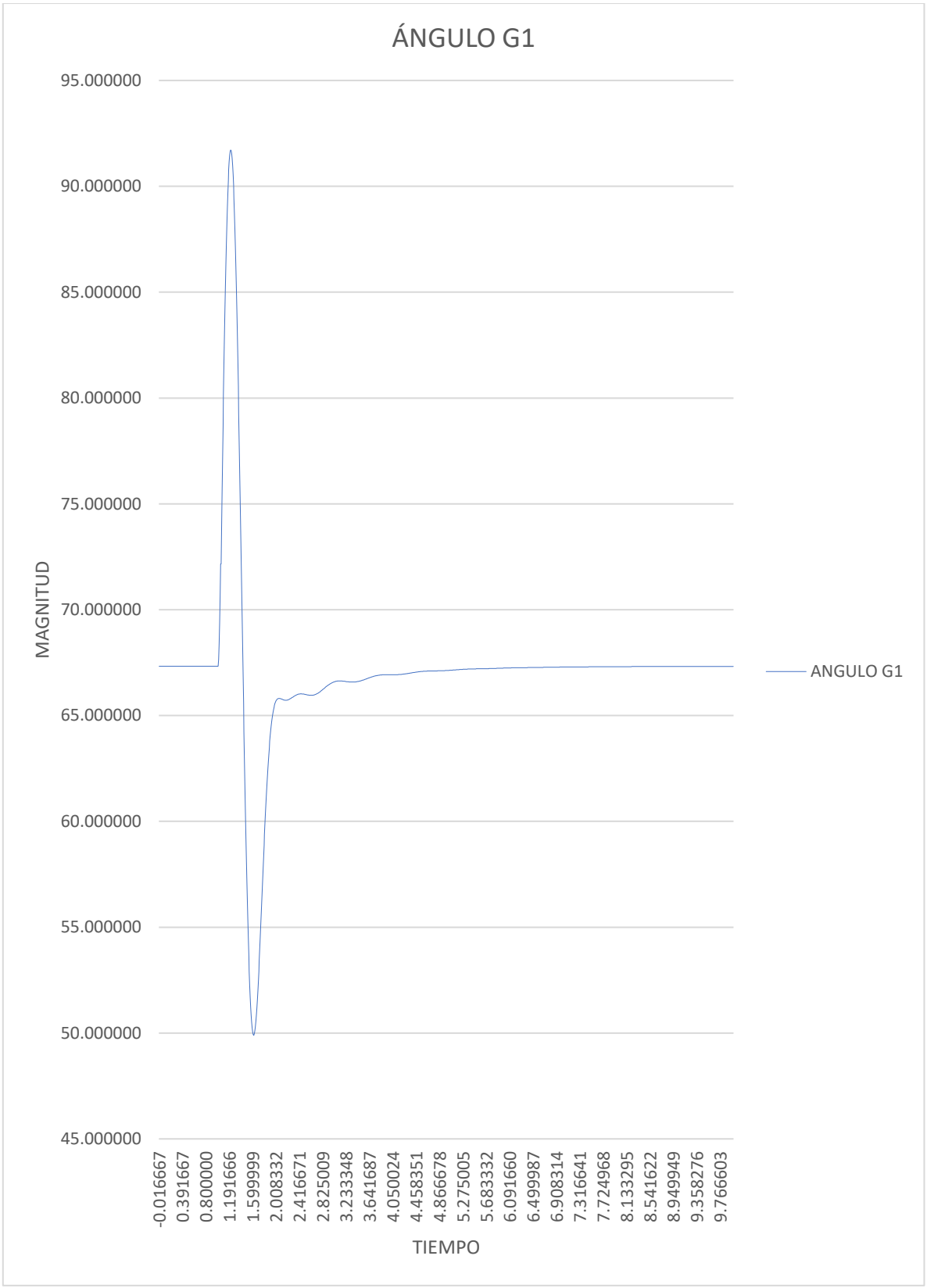

Figura 4.8 Grafica de ángulo del generador G1, generada con los datos arrojados por las macros después de la simulación del Sistema Bus Infinito con generador tipo (GENCLS Y GENROU)

4.4 Caso 3 Sistema 14 Bus de IEEE

Se presentan los resultados después de utilizar las macros antes descritas, con fines de mostrar los datos obtenidos por PSS/E después de la simulación y verificar que para los ejemplos puedan ser útiles, y el usuario que las utilizara en sus análisis dinámicos tenga la información requerida.

Resultados del network después el análisis dinámico

	Bus Number	Bus Name	Id	Code	VSched (pu)	Remote Bus	In Service	Pgen (MW)	Pmax (MW)	Pmin (MW)	Qgen (Mvar)	Qmax (Mvar)	Qmin (Mvar)
	1	BUS 1 230.00	1	2	1.0600	0	☑	232.3454	9999.0000	-9999.0000	-16.5310	100.0000	-20.0000
	2	BUS 2 230.00	1	2	1.0450	0	☑	40.0103	9999.0000	-9999.0000	43.5316	50.0000	-40.0000
	3	BUS 3 230.00	1	2	1.0100	0	☑	0.0101	9999.0000	-9999.0000	25.0717	40.0000	0.0000
	6	BUS 6 115.00	1	2	1.0700	0	☑	-0.0000	9999.0000	-9999.0000	12.7241	24.0000	-6.0000
	8	BUS 8 13.200	1	2	1.0900	0	☑	-0.0012	9999.0000	-9999.0000	17.6258	24.0000	-6.0000

Figura 4.9 Resultados del network después el análisis dinámico

Resultados del network después el análisis dinámico

	Bus Number	Bus Name	Base kV	Area Number/Name	Zone Number/Name	Owner Number/Name	Code	Voltage (pu)	Angle (deg)
	1	BUS 1	230.0	1	1	1	2	1.0600	-53.993
	2	BUS 2	230.0	1	1	1	2	1.0459	-58.988
	3	BUS 3	230.0	1	1	1	2	1.0097	-66.704
	4	BUS 4	230.0	1	1	1	1	1.0179	-64.304
	5	BUS 5	230.0	1	1	1	1	1.0198	-62.766
	6	BUS 6	115.0	1	1	1	2	1.0699	-68.213
	7	BUS 7	1.0	1	1	1	1	1.0616	-67.353
	8	BUS 8	13.2	1	1	1	2	1.0899	-67.356
	9	BUS 9	115.0	1	1	1	1	1.0560	-68.932
	10	BUS 10	115.0	1	1	1	1	1.0510	-69.091
	11	BUS 11	115.0	1	1	1	1	1.0569	-68.784
	12	BUS 12	115.0	1	1	1	1	1.0551	-69.068
	13	BUS 13	115.0	1	1	1	1	1.0503	-69.149
	14	BUS 14	115.0	1	1	1	1	1.0355	-70.027

Figura 4.10 Resultados del network después el análisis dinámico

Observaciones dentro de las gráficas potencia de generadores y ángulos en sincronismo:

En la Figura 4.11 se puede observar la maquina en sincronismo con una potencia constante durante el tiempo de simulación antes de ocurrir la perturbación del sistema, para este caso la falla en el bus 13 cuando se genera la perturbación se puede apreciar que la potencia suministrada en el sistema cae a cero, sin embargo la potencia generada sigue aumentando, esto causa que se genere una desigualdad entre la potencia mecánica del motor y la potencia

eléctrica del mismo por lo cual la potencia de aceleración comienza a variar oscilando y provocando que la potencia eléctrica también comience una oscilación, esto se puede apreciar durante todo el tiempo en el que ocurre la falla hasta el momento de la liberación. Después de provocarse la liberación de la falla podemos observar que la potencia sufre un incremento notorio en las maquinas G2, G3, G4, G5, a comparación de la maquina G1 que al liberarse la falla tomo un valor menor al de operación antes de iniciar la falla y elevándolo para después de unos ciclos de oscilaciones restablecer un punto de operación o cercano a él.

Dentro del sistema se agregan unos controles de excitador tipo IEEET1, este sistema de excitación se encarga de brindar corriente directa al devanado de campo de la máquina. La función de control de este sistema de excitación son modificar el comportamiento del voltaje y el flujo de potencia reactiva ya que es el principal sistema de control encargado de mantener la estabilidad en el sistema analizado porque actúa de forma más rápida que otro sistema de control.[9]

Sin embargo, con respecto al ángulo de generadores que se muestra en la Figura 4.13 Comparación angular, donde claramente se observa cómo se rebasa la diferencia angular a más de 90 grados lo que provoca una desincronización de la máquina, sin embargo, al agregarse los controles se genera un restablecimiento de sincronización, de esta forma se observa el amortiguamiento y con respecto a la gráfica se observa un acercamiento al punto inicial de separación angular o cercano a él.

Con los datos obtenidos gracias a las macros podemos ir modificando ciertas características para hacer más eficiente el programa realizado, las macros realizadas siempre arrojaran los resultados de las gráficas para poder observar el comportamiento en los cambios de estado del sistema, de esta forma es de suma importancia tener en cuenta que debemos de contar con un conocimiento previo sobre los comportamientos mencionados, para poder tener una mayor comprensión de la relación programación-resolución del sistema descrito.

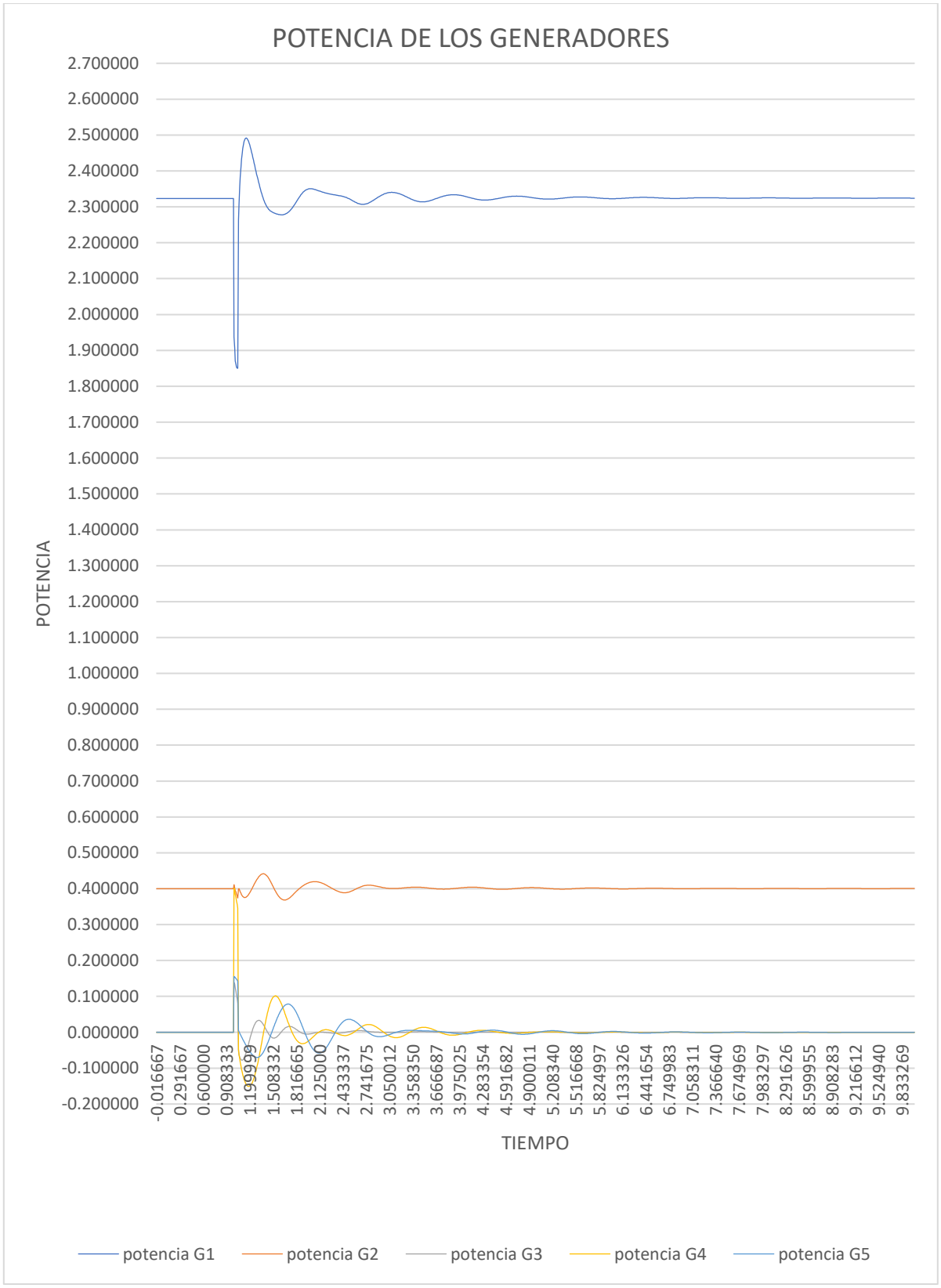

Figura 4.11 Grafica de potencia en G1, G2, G3, G4, G5, generada con los datos arrojados por las macros después de la simulación del Sistema 14 Bus de IEEE

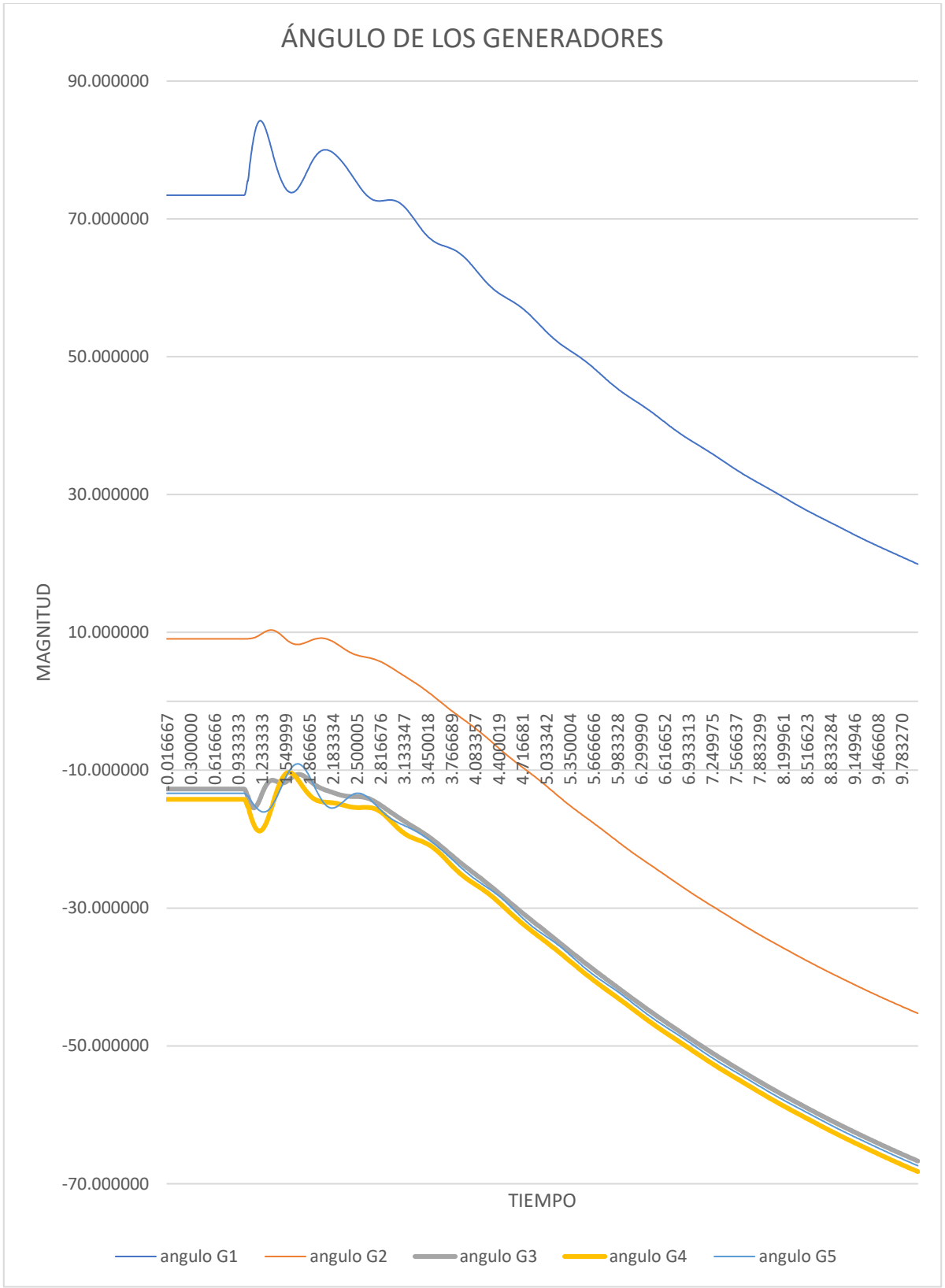

Figura 4.12 Grafica de ángulos de los generadores G1, G2, G3, G4, G5, generada con los datos arrojados por las macros después de la simulación del Sistema 14 Bus de IEEE

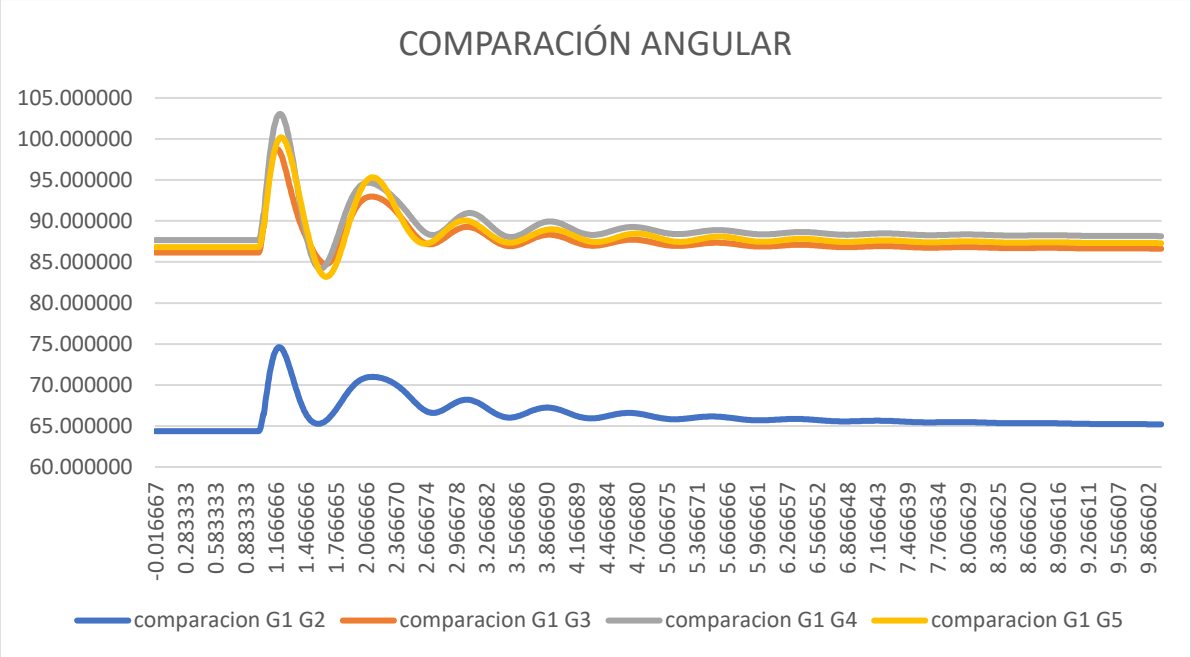

Figura 4.13 Grafica de comparación angular de las maquinas G2, G3, G4, G5 contra G1 para verificar que la diferencia angular sea menor a 90 grados eléctricos y determinar el sincronismo

4.5 Caso 4 Sistema 9 Buses de IEEE

Se presentan los resultados después de utilizar las macros antes descritas, con fines de mostrar los datos obtenidos por PSS/E después de la simulación y verificar que para los ejemplos puedan ser útiles, y el usuario que las utilizara en sus análisis dinámicos tenga la información requerida.

Resultados del network después el análisis dinámico

	Bus Number	Bus Name	Id	Code	VSched (pu)	Remote Bus	In Service	Pgen (MW)	Pmax (MW)	Pmin (MW)	Qgen (Mvar)	Qmax (Mvar)	Qmin (Mvar)
	1	BUS 1 16.500	1	2	1.0400	0	☑	71.6410	9999.0000	-9999.0000	27.0460	9999.0000	-9999.0000
	2	BUS 2 18.000	1	2	1.0250	0	☑	163.0000	9999.0000	-9999.0000	6.6537	9999.0000	-9999.0000
	3	BUS 3 13.800	1	2	1.0250	0	☑	85.0000	9999.0000	-9999.0000	-10.8597	9999.0000	-9999.0000

Figura 4.14 Resultados del network después el análisis dinámico

Resultados del network después el análisis dinámico

	Bus Number	Bus Name	Base kV	Area Number/Name	Zone Number/Name	Owner Number/Name	Code	Voltage (pu)	Angle (deg)
	1	BUS 1	16.5	1	1	1	2	1.0231	174.345
	2	BUS 2	18.0	1	1	1	2	1.0342	-178.828
	3	BUS 3	13.8	1	1	1	2	1.0204	177.977
	4	BUS 4	230.0	1	1	1	1	0.9976	169.805
	5	BUS 5	230.0	1	1	1	1	0.9463	163.892
	6	BUS 6	230.0	1	1	1	1	0.9915	168.785
	7	BUS 7	230.0	1	1	1	1	1.0298	178.187
	8	BUS 8	230.0	1	1	1	1	1.0154	174.744
	9	BUS 9	230.0	1	1	1	1	1.0261	175.341

Figura 4.15 Resultados del network después el análisis dinámico

Observaciones dentro de las gráficas potencia de generadores y ángulos en sincronismo:

En la Figura 4.16 se puede observar la maquina en sincronismo con una potencia constante durante el tiempo de simulación antes de ocurrir la perturbación del sistema, para este caso la falla en el bus 7. Cuando se genera la perturbación se puede apreciar que la potencia suministrada en el sistema cae a cero sin embargo la potencia generada sigue aumentándose para todos los generadores del sistema, al generarse la aceleración por la desigualdad de la potencia mecánica y eléctrica comienza a oscilar provocando que cuando ocurra el restablecimiento el comportamiento de los tres generadores sea completamente diferente, los generadores G2 y G3 al restablecerse la falla y liberar la línea 5-7 tomaron una potencia menor de la que se tenía en el valor inicial de la falla a diferencia del G1 que mostro una elevación significativa al punto respecto al punto de operación inicial, los tres generadores presentan oscilaciones no amortiguadas después de presentarse la falla.

Con respecto al ángulo podemos observar en la Figura 4.18 que al no haber una diferencia angular de más de 90 grados eléctricos el sincronismo se mantiene, se sabe que al no tener controles el amortiguamiento del incremento al igual que las oscilaciones continuara en aumento sin embargo no se presenta una tendencia a desincronización de las maquinas.[7]

Con los datos obtenidos gracias a las macros podemos ir modificando ciertas características y así hacer mucho más eficiente el programa realizado, las macros realizadas siempre arrojaran los resultados de las gráficas para poder observar el comportamiento en los cambios de estado del sistema, de esta forma es de suma importancia tener en cuenta que debemos de contar con un conocimiento previo de los dichos comportamientos, para tener una mayor comprensión de la relación programación-resolución del sistema descrito.

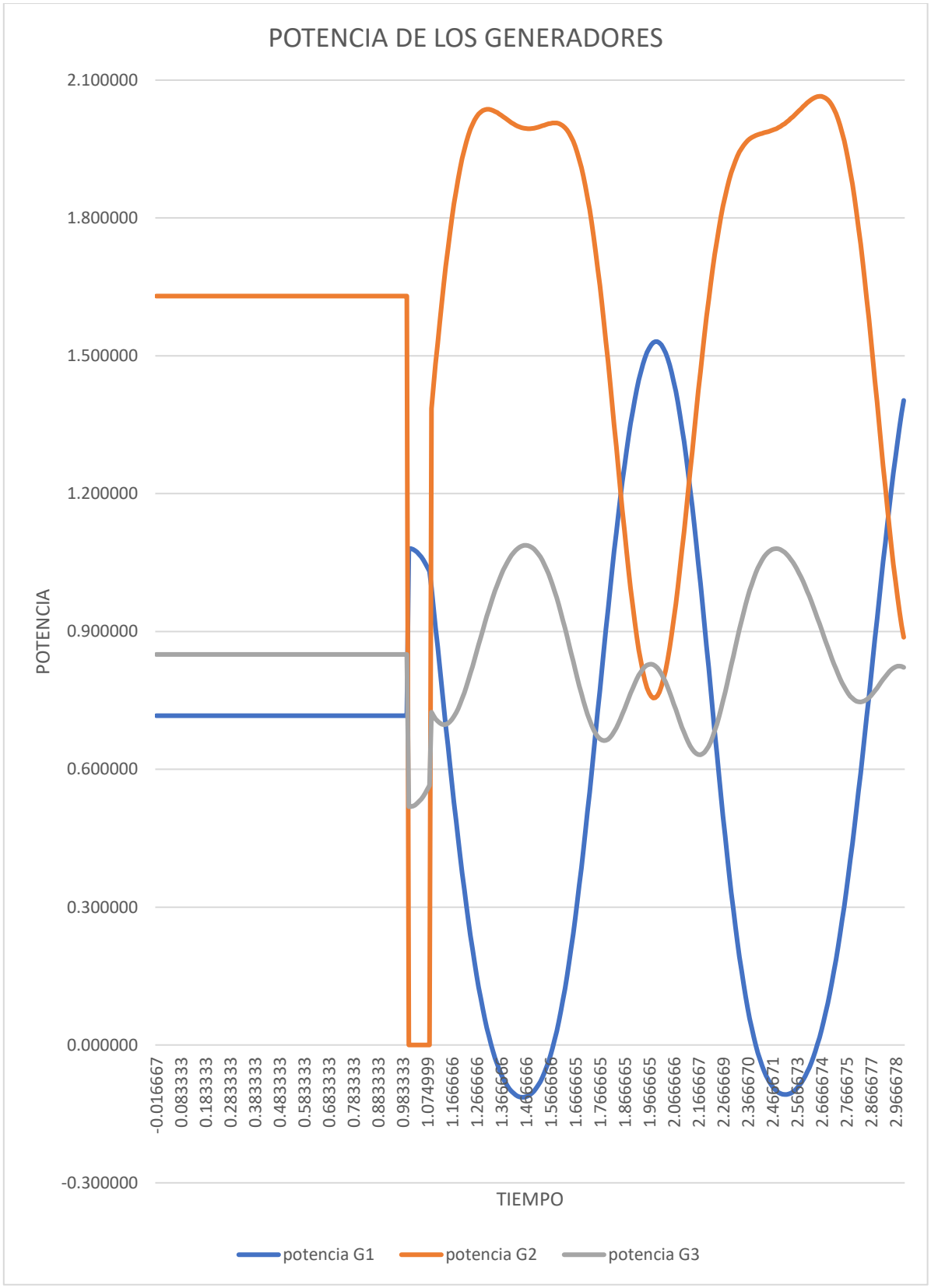

Figura 4.16 Grafica de potencia en G1, G2, G3, generada con los datos arrojados por las macros después de la simulación del Sistema 9 Buses de IEEE

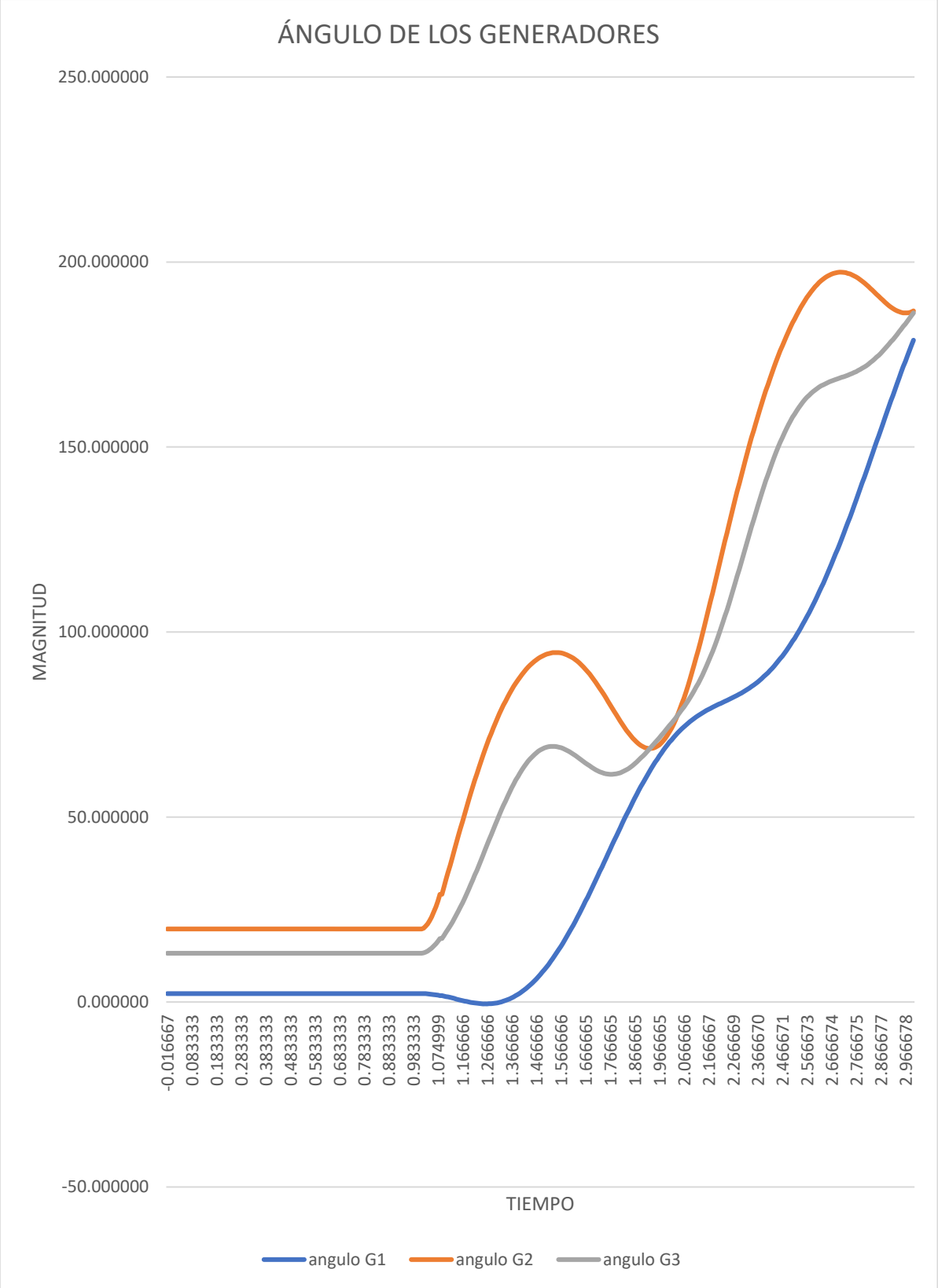

Figura 4.17 Grafica de ángulos de los generadores G1, G2, G3, generada con los datos arrojados por las macros después de la simulación del Sistema 9 Buses de IEEE

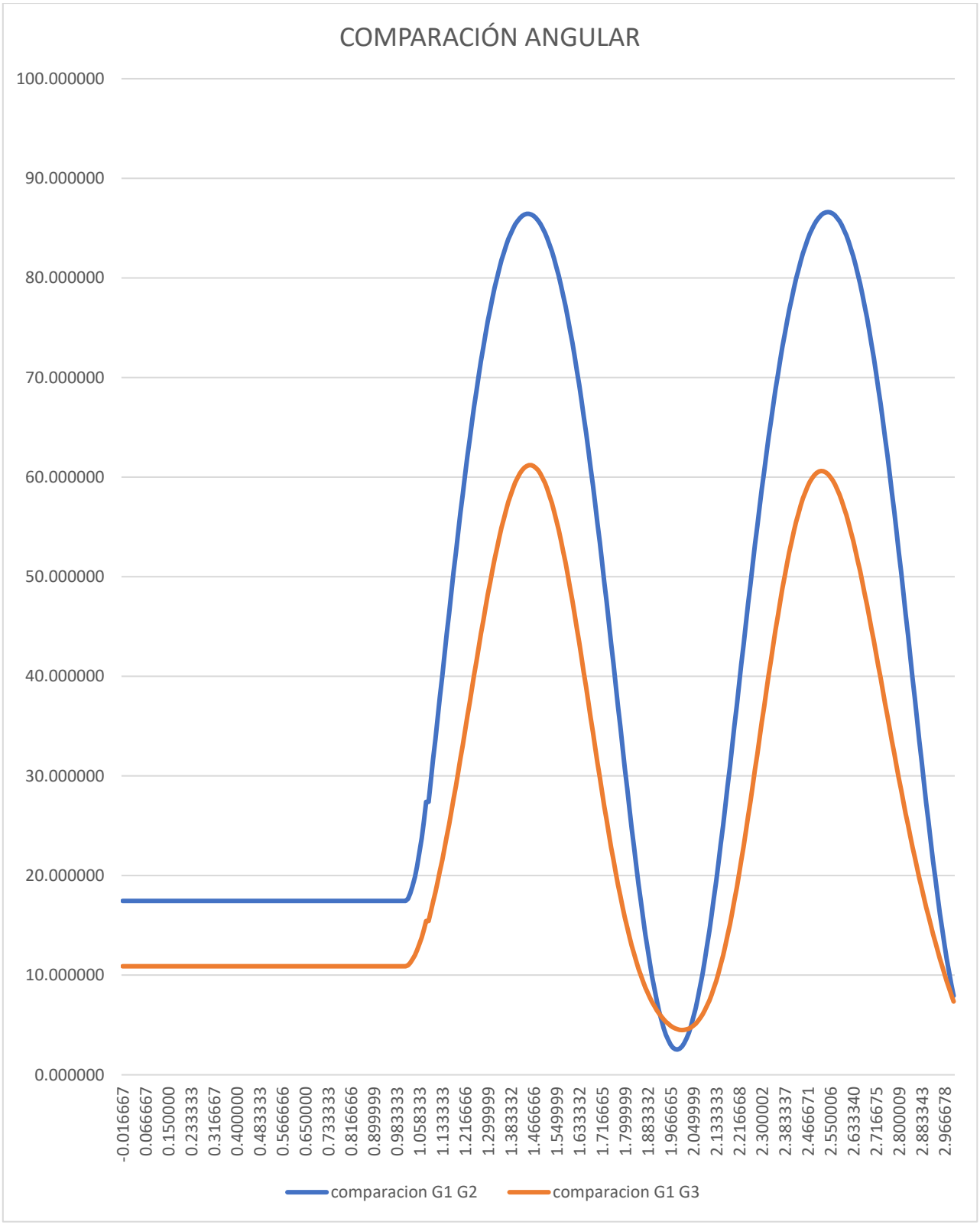

Figura 4.18 Grafica de comparación angular de las maquinas G2, G3 contra G1 para verificar que la diferencia angular sea menor a 90 grados eléctricos y determinar el sincronismo

Capítulo 5 Conclusiones

5.1 Conclusiones

Después de llevar a cabo el presente trabajo y realizar las respectivas simulaciones de cada caso a analizando se puede llegar a las siguientes conclusiones:

1. La programación de las macros facilita la introducción de datos, porque al tener una base de datos firme el usuario puede manipular la información antes de realizar la simulación.

2. Cuando el usuario corre la simulación ahorrar tiempo, porque evita los procedimientos repetitivos dentro de la simulación haciendo una eficiencia de tiempo con respecto a la inicial

3. El usuario puede tener un mejor manejo de la información en la simulación, no será necesario tener que hacer simulaciones en prefalla, falla y liberación de falla, puesto que al establecer los parámetros de cada una se ellas las macros se encargará de realizarlas y asignar los datos brindados por el usuario

4. Al correr las macros los resultados gráficos se muestran de forma automática a diferencia de hacer el procedimiento manual para mostrar los mismos.

5. Regularmente cuando el usuario comienza la simulación de forma manual se sugiere guardar la información en los respectivos archivos de salida, sin embargo al perder alguno de la secuencia de archivos genera un problema esto ocurre por qué se tendría que comenzar desde el archivo eliminado y rehacerlo, en comparación las macros no obligan al usuario a conservar los resultados ya que lo más importante será la única base de datos a utilizar, los procesos siempre se realizarán de manera automática y no se utilizan archivos antes creados en PSSE.

5.2 Contribuciones

1. Se desarrolló un programa con el cual el usuario puede introducir información a partir de una base de datos ordenada de una manera específica, para realizar simulaciones dinámicas pre falla, en falla y liberación de falla, con fines de obtener resultados de forma gráfica necesarios para entender el comportamiento de las máquinas en estos tres estados del sistema.

2. Se muestra el comportamiento de las macros, la efectividad de la programación al no modificar los resultados obtenidos porque la secuencia se conserva en todo momento.

5.3 Áreas de oportunidad

1. Reducir la programación puesto que las funciones aíslan los procesos de introducción de datos.

2. Generar macros para introducir los datos de los diferentes tipos de controles porque para algunos de los casos mostrados solo se introdujeron los datos de excitadores.

3. Generar macros para introducir otros diferentes tipos de generación (solar, eólica) así como sus diferentes controles.

4. Generar macros para diferentes tipos de estudio (análisis de contingencias).

5. Generar macros para introducir los diferentes compensadores de potencia o voltaje (capacitores, reactores, entre otros).

Bibliografía

[1] González, R. (2008). *Python PARA TODOS*. España: Creative Commons .

[2] Albino, I.(2017). *Procedimiento de estabilidad transitoria*. Benemérita Universidad Autónoma de Puebla. Puebla, México.

[3] Grainger, J & Stevenson, W.(1994). *Power System Analysis*. McGraw-Hill. New York. USA.

[4] Siemens Power Technologies International. (October 2010). *PSS®E Application Program Interface (API)*. New York, USA: Siemens Energy.

[5] Christie, R., *14 Bus Power Flow Test Case*, consultado en agosto, 2019, de University of Washington, 1993. Recuperado de: https://www2.ee.washington.edu/research/pstca/pf14/pg_tca14bus.htm

[6] Asprou, M, Quirós-Tortós, J. and Kyriakides E., "IEEE 14-bus modified test system", consultado en agosto, 2019, de University of Cyprus, 2017. Recuperado de: http://www.kios.ucy.ac.cy/testsystems/index.php/dynamic-ieee-test-systems/ieee-14-busmodified-test-system

[7] Anderson, P., & Fouad, A., "Power System Control and Stability", The Iowa state university press, Ames Iowa, USA, 1977.

[8] Asprou, M, Quirós-Tortós, J. and Kyriakides E., "IEEE 9-bus modified test system", consultado en agosto 2019, de University of Cyprus 2017. Recuperado de: http://www.kios.ucy.ac.cy/testsystems/index.php/dynamic-ieee-test-systems/ieee-9-busmodified-test-system

[9] Hermilo, J. (Agosto 2008). Modelo de la máquina síncrona modificada incluyendo sus lazos de control para estudios de estabilidad angular. México, D.F: Instituto Politécnico Nacional.

[10] Castelán, R.. (Agosto 2018). Estudio De Estabilidad Transitoria En Sistemas Eléctricos De Potencia Utilizando El Programa Pss/E. Puebla, Puebla, México.: Benemérita Universidad Autónoma De Puebla.

Anexos

ANEXO 1 :ESTRUCTURA DEL PROGRAMA PARA RESOLUCIÓN DE LOS CASOS A SIMULAR

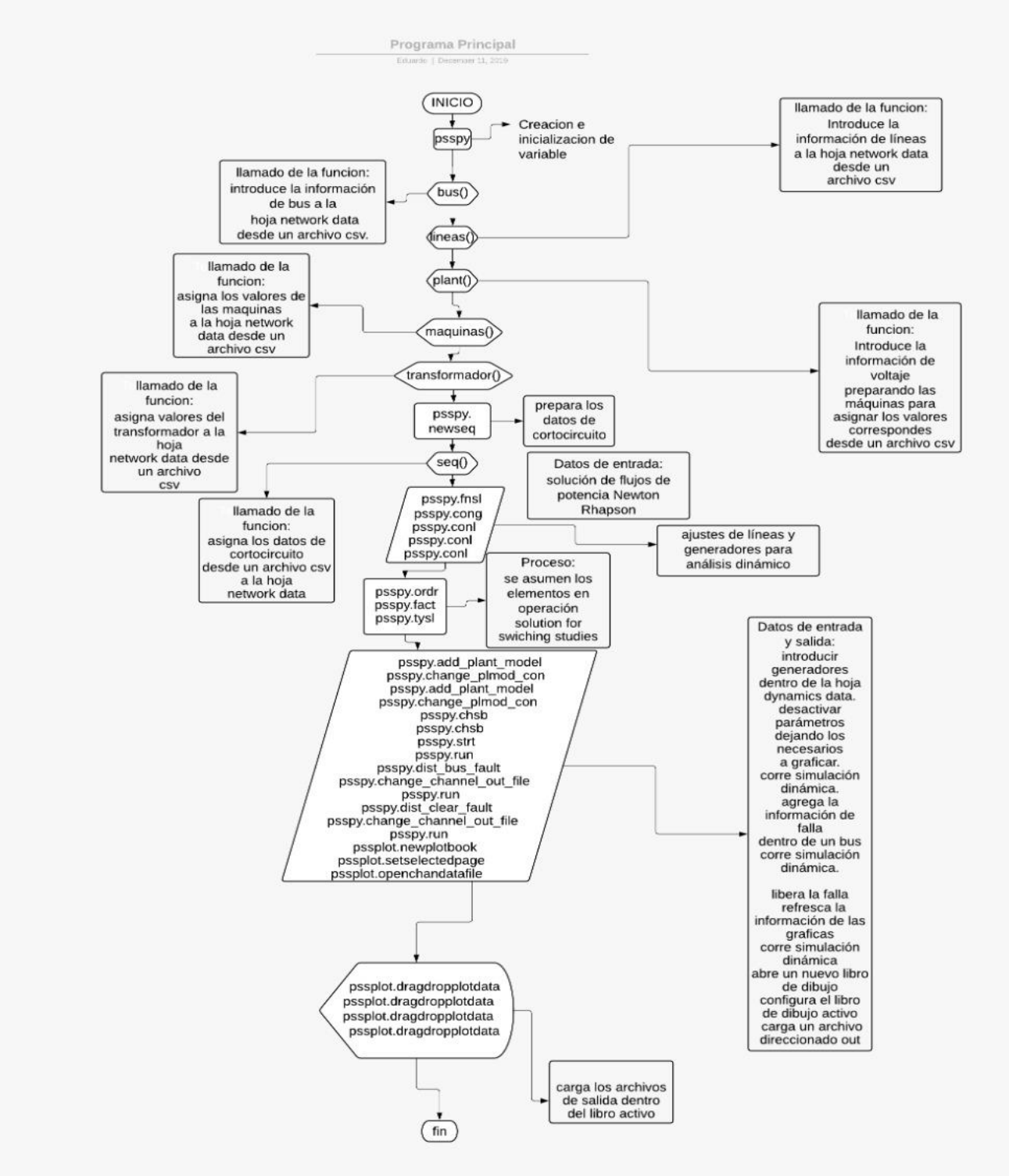

Figura A.1 Diagrama de flujo que muestra la resolución de una simulación utilizado para la creación de las macros

En la Figura A.1 podemos observar como se genera la estructura de la resolución de un sistema dinámico, para este caso en particular, son sistemas que se mantienen estables aplicándoles una falla que será liberada mostrando unas graficas de comportamiento de potencia y ángulo

ANEXO 2 :ESTRUCTURA INTERNA DE LAS FUNCIONES DE ASIGNACIÓN DE INFORMACIÓN

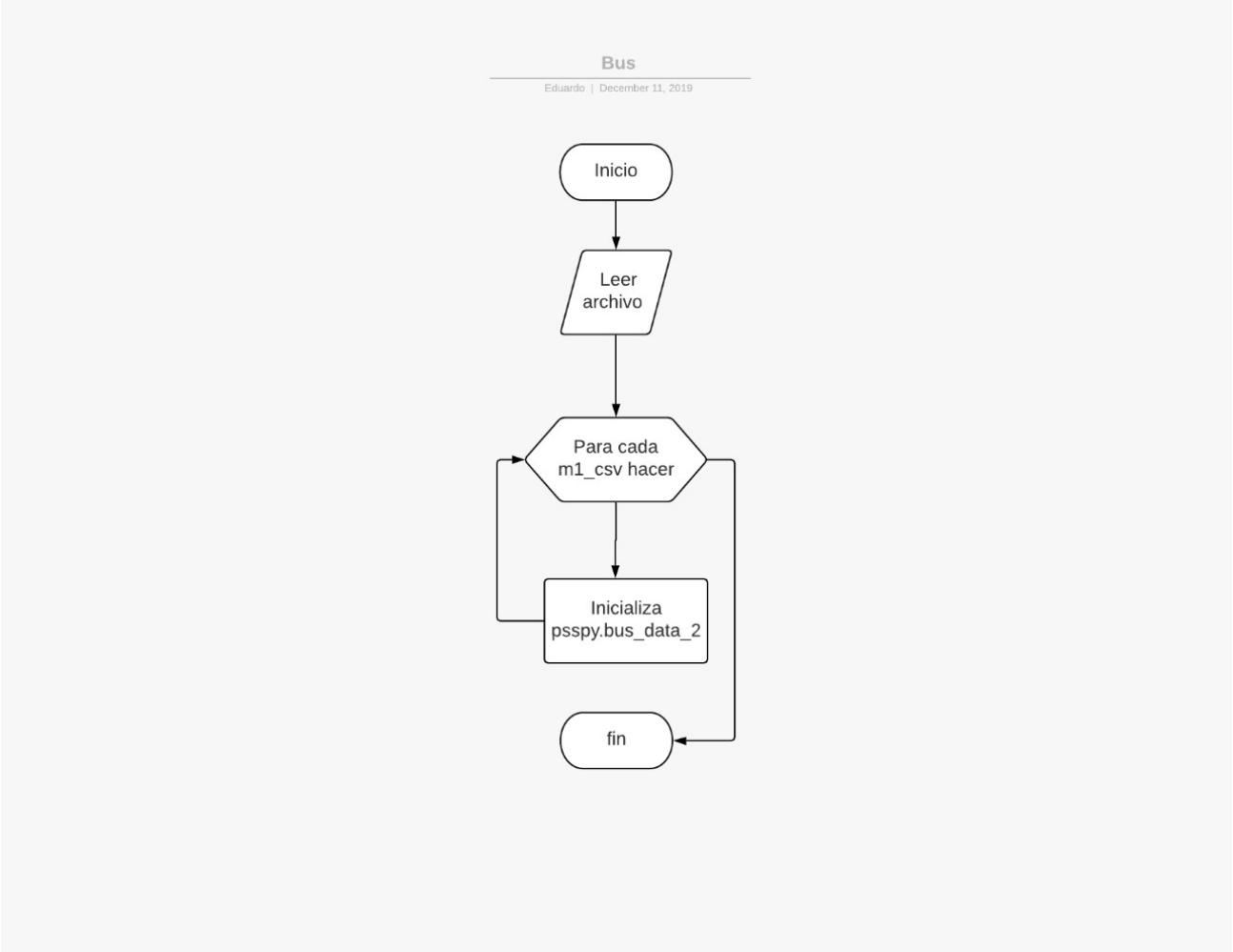

Figura A.2 Diagrama de flujo utilizado para la creación de macros correspondientes a la asignación de valores desde un archivo CSV

En la Figura A2 se observa la estructura de las funciones con que se realizó el llenado de la información, estas funciones fueron creadas especialmente para que nuevos usuarios pudieran incorporar información de nuevos casos utilizando la misma estructura, con el fin de brindar una forma de trabajo diferente.