ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA METODOLOGÍA HÍBRIDA PARA LA SINTONIZACIÓN DE CONTROLADORES DE POTENCIA PARA AMORTIGUAR OSCILACIONES DE BAJA FRECUENCIA DE MANERA ADAPTATIVA, UTILIZANDO DESCOMPOSICIÓN DE LA SEÑAL, TÉCNICAS HEURÍSTICAS Y EL MÉTODO DEL RESIDUO

DESARROLLO DE UNA METODOLOGÍA HÍBRIDA PARA LA SINTONIZACIÓN DE ESTABILIZADORES DE POTENCIA USANDO LOS ALGORITMOS DE OPTIMIZACIÓN MVMO Y WOA, MÉTODO DEL RESIDUO Y LA DESCOMPOSICIÓN DE LA SEÑAL

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRICIDAD

RAMIRO JOHAO BENAVIDES ENCALADA

ramiro.benavides@epn.edu.ec

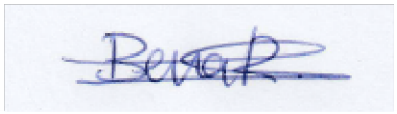
DIRECTOR: PROF. DR. CARLOS FABIÁN GALLARDO QUINGATUÑA (PhD)

carlos.gallardo@epn.edu.ec

DMQ, febrero 2022

CERTIFICACIONES

Yo, Ramiro Johao Benavides Encalada declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



RAMIRO JOHAO BENAVIDES ENCALADA

Certifico que el presente trabajo de integración curricular fue desarrollado por Ramiro Johao Benavides Encalada, bajo mi supervisión.

DR. CARLOS FABIÁN GALLARDO QUINGATUÑA
DIRECTOR

Certificamos que revisamos el presente trabajo de integración curricular.

NOMBRE_REVISOR1	**NOMBRE_REVISOR2**
REVISOR1 DEL TRABAJO DE INTEGRACIÓN CURRICULAR	**REVISOR2 DEL TRABAJO DE INTEGRACIÓN CURRICULAR**

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

RAMIRO JOHAO BENAVIDES ENCALADA

PROF. DR. CARLOS FABIÁN GALLARDO QUINGATUÑA (PhD)

DEDICATORIA

A Dios, a mi madre Miriam Encalada, a mi amigo Ángel Sisalema y a todas las personas que hicieron posible que pueda llegar hasta aquí.

AGRADECIMIENTO

A mi director, al Dr. Carlos Gallardo y al Mgtr. Mauricio Soria por su asesoramiento y paciencia para la realización de este trabajo.

ÍNDICE DE CONTENIDO

RESUMEN

El propósito de este trabajo es desarrollar una metodología híbrida que permita sintonizar estabilizadores de potencia previamente ya ubicados con dos algoritmos de optimización MVMO (Optimización de Mapeo por Media y Varianza) y WOA (Algoritmo de Optimización de Ballenas), con el fin de que se pueda amortiguar oscilaciones de baja frecuencia tanto para el Sistema New England como para el Sistema Nacional Interconectado. Ahora bien, se parte de la premisa de que se está realizando la sintonización de Estabilizadores de Potencia en simulación y esos parámetros que se obtuvo se desea aplicarlos en la vida real, cuando eso sucede el sistema presenta cambios a lo que se obtuvo en simulación, si esos cambios que se obtienen tienen que ver con un bajo amortiguamiento de modos de oscilación del sistema menores al 5% establecido es necesario realizar un ajuste.

Este ajuste se lo realizará con el Método del Residuo, el cual será llamado residuo estimado y será explicado en los siguientes capítulos. Después de realizar el ajuste para la sintonización de los dos algoritmos, la estabilidad de pequeña señal del sistema mejorará ya que no tendrá ningún modo que tenga un amortiguamiento menor al 5%.

Para escoger al algoritmo que presente una mejor estabilidad transitoria se utilizará la descomposición de la señal, obteniendo los modos oscilatorios de las señales provenientes de Unidades de Medición Sincrofasorial Virtual, con estos modos se comparará el amortiguamiento que tienen y se elegirá al mayor.

PALABRAS CLAVE: amortiguamiento, Residuo, Estimado, MVMO, WOA, Estabilizador de Sistemas de Potencia (PSS).

ABSTRACT

The purpose of this work is to develop a hybrid methodology that allows to tune previously located power stabilizers with two optimization algorithms MVMO (Mean Variance Mapping Optimization) and WOA (Whale Optimization Algorithm), to damp low oscillations. frequency for both the New England System and the National Interconnected System. Now, it is based on the premise that the tuning of Power Stabilizers is being carried out in simulation and those parameters that were obtained are to be applied in real life, when that happens the system presents changes to what was obtained in simulation, If these changes that are obtained have to do with a low damping of oscillation modes of the system less than the established 5%, it is necessary to make an adjustment.

This adjustment will be made with the Residual Method, which will be called estimated residual and will be explained in the following chapters. After adjusting for the tuning of the two algorithms, the small-signal stability of the system will improve as it will not have any mode that has less than 5% damping.

To choose the algorithm that presents the best transient stability, the failure of the signal will be ensured, obtaining the oscillatory modes of the signals coming from Virtual Phasor Measurement Units, with these modes the damping they have will be compared and the highest will be chosen.

KEYWORDS: damping, Residue, Estimated, MVMO, WOA, Power System Stabilizers (PSS).

1 INTRODUCCIÓN

El componente se desarrolló de la siguiente manera:

Para el Sistema New England se estableció tres puntos de operación y de esos tres puntos se realizó un análisis modal sin PSS´s (Estabilizadores de Potencia), obteniendo que para la demanda mínima no es necesario PSS´s y para demanda media y máxima si son necesarios.

Para este sistema se hizo una investigación bibliográfica para saber en que generadores están ubicados los estabilizadores. Posteriormente, se utilizó el algoritmo MVMO y WOA programados en DPL para sintonizar los estabilizadores de potencia en demanda media y demanda máxima.

Después se determina los modos de oscilación electromecánicos de las señales de las unidades de medición sincrofasorial (virtuales) utilizando Descomposición de la Señal (Método Prony) con el programa en Matlab llamado “ringdown”. Lo que se busca con esto es encontrar en una señal el modo oscilatorio que se parezca a los modos que no tienen el amortiguamiento mayor al 5%, en demanda media se tiene 2 modos locales que no cumplen con el amortiguamiento y en demanda máxima hay 1 modo entre área y 2 locales. Después de buscar que señal descomponer se llegó a que cuando se tiene un modo entre área que tiene un bajo amortiguamiento la señal que descomponiendo se encontrará a un modo parecido es la Potencia Activa de la línea de Interconexión de las dos áreas que se forman, esto se explica mejor en los resultados. Esto sirve para poder comparar la estabilidad transitoria del MVMO y WOA, en otras palabras, que tanto se amortigua la señal después de que ocurra una perturbación.

Posteriormente, se simula la premisa de que se realiza la sintonización de Estabilizadores de Potencia en simulación y esos parámetros que se obtuvo se desea aplicarlos en la vida real. En el caso del Sistema New England la demanda media se la toma como simulación y la demanda máxima como operación real. Dando como resultado después que en demanda máxima se tiene un modo que no cumple con el amortiguamiento; aquí es necesario realizar un ajuste el cual se lo realizará con el Método del Residuo para mover el modo hacia un amortiguamiento mayor al 5%. Cabe aclarar que se utilizará un residuo estimado y no uno calculado con matrices.

Después de realizar el ajuste tanto para el MVMO como para el WOA, es necesario escoger cual es el mejor en el dominio del tiempo y aquí se utiliza lo que se mencionó antes, la descomposición de la señal.

Para el Sistema Nacional Interconectado se hizo lo mismo, la única diferencia es que la demanda mínima se la toma como simulación y la demanda máxima como operación real.

1.1 Objetivo general

Desarrollar una metodología híbrida que permita sintonizar estabilizadores de potencia utilizando dos técnicas heurísticas de optimización, que son corregidas utilizando el método del residuo y validadas a través de la descomposición de la señal obtenida de la tecnología de medición sincrofasorial virtual, a fin de amortiguar las oscilaciones de baja frecuencia en el Sistema New England y en el Sistema Nacional Interconectado.

1.2 Objetivos específicos

1. Sintonizar Estabilizadores de Potencia utilizando dos algoritmos de optimización: MVMO (Mean Variance Mapping Optimization) y WOA (Whale Optimization Algorithm) en el dominio del tiempo.
2. Determinar los modos de oscilación electromecánicos de las señales de las unidades de medición sincrofasorial (virtuales) utilizando Descomposición de la Señal (Método Prony).
3. Utilizar los resultados de los algoritmos de optimización y mediante el método del residuo ajustar los parámetros de los Estabilizadores de Potencia. Aplicar la metodología tanto al Sistema New England como al Sistema Nacional Interconectado.

1.3 Alcance

En el presente componente se planea realizar la sintonización de Estabilizadores de Potencia previamente ya ubicados con los algoritmos de optimización MVMO y WOA para los sistemas New England y para el Sistema Nacional Interconectado, con el fin de amortiguar las oscilaciones de baja frecuencia.

Teniendo los parámetros de los Estabilizadores de Potencia se modificará el punto de operación inicial a uno mayor, para probar la sintonización y posteriormente realizar el ajuste. En el caso del Sistema New England se modificará de Demanda Media a Demanda

Máxima. Para el caso del Sistema Nacional Interconectado se modificará de Demanda Mínima a Demanda Máxima.

Esto se hace porque se tiene como premisa que el punto de operación inicial es llamado de “simulación” y el punto de operación mayor se lo considera como “real”. Esto sucede principalmente porque en el mundo simulación todo es ideal y al momento de llevar los resultados al mundo real se tiene ciertos fallos o en casos muy extremos no cumple.

Ya en el punto de operación mayor se utilizará el Método del Residuo Estimado para poder mover los autovalores (valores propios o modos) hacia un amortiguamiento deseado.

Como se está utilizando dos algoritmos de optimización se busca escoger cual sintonización es mejor en el dominio del tiempo, en otras palabras, cual sintonización presenta una mejor estabilidad transitoria después de haber ocurrido una perturbación. Para esto se determina los modos de oscilación electromecánicos de las señales de las unidades de medición sincrofasorial (PMU´s virtuales) utilizando Descomposición de la Señal (Método Prony). Después de encontrar el modo de oscilación electromecánico que describe perfectamente a la señal de estudio, se verificará con cual algoritmo se tiene un mayor amortiguamiento en el dominio del tiempo. Para realizar la descomposición de la señal se utilizó el software de Matlab “ringdown”.

Cabe mencionar que la sintonización de Estabilizadores de Potencia se la realiza en el software DigSilent PowerFactory en lenguaje DPL (DigSilent Programming Language) y el Método del Residuo Estimado se lo programó en Python.

1.4 Marco teórico

1.4.1 Estabilidad de Sistemas Eléctricos de Potencia

Según [1] me indica que es la propiedad de un Sistema de Potencia que le permite permanecer en un estado de equilibrio operativo en condiciones normales de funcionamiento y también le permite recuperar un estado de equilibrio aceptable después de estar sujeto el sistema a una perturbación. Esta perturbación puede ser severa, como cortocircuitos o suave como variaciones pequeñas de carga.

La estabilidad de Sistema de Potencia se puede clasificar en:

- Estabilidad de Voltaje
- Estabilidad de Frecuencia

- Estabilidad de Ángulo

En este trabajo se hará especial énfasis en la estabilidad del ángulo del rotor.

1.4.2 Estabilidad de Ángulo

En [2] dice que la estabilidad del ángulo del rotor es la capacidad de las máquinas síncronas interconectadas de un sistema de potencia para permanecer en sincronismo, después de haber estado sujeto a una perturbación. La inestabilidad de ángulo provoca un incremento en las oscilaciones de ángulo, perdiendo el sincronismo de algunos generadores del sistema.

La estabilidad de ángulo se puede clasificar en:

- Estabilidad de Pequeña Señal
- Estabilidad de Gran Señal o Transitoria

1.4.3 Estabilidad de Pequeña Señal

Según [1] es la capacidad del Sistema de Potencia para mantener el sincronismo bajo pequeñas perturbaciones. Estas perturbaciones ocurren continuamente en el sistema debido a pequeñas variaciones en las cargas y generación. Una perturbación se considera pequeña si es posible la linealización de las ecuaciones del sistema para fines analíticos.

Según [3] la inestabilidad de pequeña señal está presente en algunos de los principales colapsos de un sistema que son atribuidos a oscilaciones electromecánicas que son descritas a continuación:

Oscilaciones de modos entre áreas

Este fenómeno complejo se observa sobre una gran parte de la red e involucra muchas partes del sistema con comportamiento dinámico no lineal.

Involucra dos grupos coherentes de grupos de generadores que se balancean entre sí a 0.8 Hz o menos. La variación en la potencia de la línea de interconexión puede ser grande, por lo cual este sería el mejor lugar para poder visualizar el impacto de este fenómeno.

La característica de amortiguamiento del modo inter-área es impuesta por la fuerza de la línea de enlace, la naturaleza de las cargas y el flujo de potencia a través de la interconexión, además de la interacción de las cargas con la dinámica de los generadores.

En [3] me indica que el funcionamiento de un sistema con la presencia de un modo inter-área ligeramente amortiguado es muy difícil, por lo cual se tiene que presentar mucha atención para poder amortiguarlo.

Oscilaciones de modos locales

En un modo local, un generador oscila contra el resto del sistema entre las frecuencias de 0.8 a 2 Hz. El impacto de este tipo de oscilación se localiza en el generador y la línea con la que se conecta a la red. El resto del sistema normalmente se modela como una fuente de voltaje constante cuya frecuencia se supone que permanece constante. Este tipo de oscilación se puede eliminar con la entrada de uno o dos estabilizadores de potencia (PSS).

Oscilaciones de modos entre máquinas

En un modo entre máquinas, en el mismo sitio de generación de energía las máquinas oscilan entre las frecuencias de entre 2 a 3 Hz, dependiendo de los valores nominales de la unidad y la reactancia que los conecta. Estas oscilaciones se llaman de esta forma porque se manifiestan dentro del complejo de la planta de generación; el resto del sistema no se ve afectado.

Oscilaciones de modos de control

Estas oscilaciones están asociadas con generadores y la mala sintonización de excitadores, reguladores de voltaje y de velocidad. Para frecuencias mayores de 4 Hz están presentes estas oscilaciones.

Oscilaciones de modos de torsión

Están asociadas con oscilaciones en el eje de la turbina – generador de la máquina síncrona. Las frecuencias en las que se presentan están en el rango de 10 a 46 Hz.

1.4.4 Estabilidad de Gran Señal o Transitoria

Es la capacidad del Sistema de Potencia para mantener el sincronismo bajo una perturbación transitoria severa. A diferencia de la estabilidad de pequeña señal, en ésta se utiliza el modelo no lineal porque el sistema al ser alterado provoca que el estado de operación después de la perturbación sea diferente al estado de operación antes de la perturbación.

1.4.5 Análisis Modal

Según [4] un sistema eléctrico de potencia consiste en una interconexión de diferentes elementos eléctricos (generadores, cargas dinámicas, reguladores, etc.), estos elementos con base en algunos supuestos se pueden describir mediante ecuaciones diferenciales.

Para poder representar un sistema de potencia (un modelo no lineal) en un espacio de estados (modelo lineal), se realiza una linealización sobre un punto de operación determinado; la linealización de un sistema de potencia multi máquina puede ser representado por las siguientes ecuaciones:

$$\Delta\dot{x} = A\Delta x + B\Delta u \tag{1.1}$$

$$\Delta y = C\Delta x + D\Delta u \tag{1.2}$$

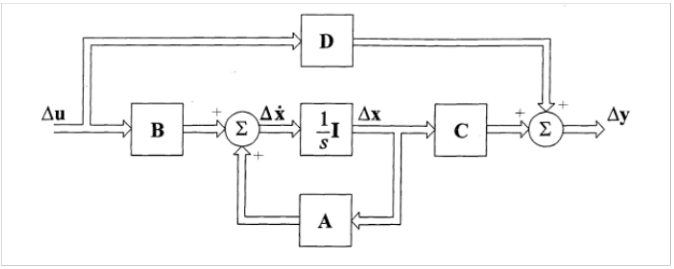


Figura 1.1. Diagrama de bloque de la representación de espacio de estados

Autovalores o valores propios y estabilidad

Para un sistema alrededor de un punto de equilibrio elegido es caracterizado por su matriz de estado A y más precisamente por sus autovalores $\lambda_i, i = 1 \dots n$ de A que corresponden a los modos del sistema. Estos autovalores se los calcula de la siguiente manera:

$$\det(A - \lambda I_n) = 0 \tag{1.3}$$

Cada autovalor está dado por su parte real e imaginaria $\lambda_i = \sigma_i \pm j\omega_i$ donde i representa el i - ésimo valor propio de la matriz de estado A.

Para cada autovalor se puede obtener la frecuencia en Hz y el radio de amortiguamiento que tiene, así respectivamente:

$$f_i = \frac{\omega_i}{2\pi} \tag{1.4}$$

$$\zeta_i = -\frac{\sigma_i}{|\lambda_i|} = -\frac{\sigma_i}{\sqrt{{\sigma_i}^2 + {\omega_i}^2}} \tag{1.5}$$

Según [5] la estabilidad de un sistema se la puede clasificar según la ubicación que tengan sus valores propios en el plano complejo. Los valores propios que se encuentran en la parte izquierda del semiplano complejo se asocian a que son estables y los que están en el semiplano complejo de la parte derecha se los considera inestables.

Autovectores o vectores propios y Matrices Modales

Para cada valor propio λ_i, el vector de n columnas ϕ_i cumple con la ecuación:

$$A\phi_i = \lambda_i \phi_i \quad i = 1,2,\dots n \tag{1.6}$$

Donde:

ϕ_i es el auto vector o vector propio derecho de la matriz de estado A asociado con el autovalor λ_i.

De la misma forma, un vector de n filas ψ_i, que cumple con la ecuación:

$$\psi_i A = \lambda_i \psi_i \quad i = 1,2,\dots,n \tag{1.7}$$

Donde:

ψ_i es el autovector izquierdo asociado con el autovalor λ_i.

Para expresar las propiedades propias de la matriz A, es conveniente introducir las siguientes matrices:

$$\Phi = [\Phi_1\ \Phi_2 \dots \Phi_n] \tag{1.8}$$

$$\Psi = [\Psi_1{}^T\ \Psi_2{}^T \dots \Psi_n{}^T]^T \tag{1.9}$$

Las dos matrices son la matriz de auto vectores derechos e izquierdos y son de dimensión nxn.

Factor de Participación

La matriz de participación (P) combina los vectores propios derecho e izquierdo como medida de la asociación entre las variables de estado y los modos de oscilación.

$$P = [p_1\ p_2 \dots p_n] \tag{1.10}$$

Con

$$p_i = \begin{bmatrix} p_{1i} \\ p_{2i} \\ \vdots \\ p_{ni} \end{bmatrix} = \begin{bmatrix} \phi_{1i}\psi_{i1} \\ \phi_{2i}\psi_{i2} \\ \vdots \\ \phi_{ni}\psi_{in} \end{bmatrix} \quad (1.11)$$

El elemento $p_{ki} = \phi_{ki}\psi_{ki}$ es el factor de participación. Es una medida de la participación relativa de la k-ésima variable de estado en el i-ésimo modo y viceversa.

Controlabilidad y Observabilidad

La matriz de controlabilidad y observabilidad modal se muestra en las siguientes ecuaciones respectivamente.

$$B' = \Phi^{-1}B \quad (1.12)$$

$$C' = C\Phi \quad (1.13)$$

La matriz de controlabilidad tiene dimensiones n x r y la matriz de observabilidad tiene dimensiones m x n. Si la i-ésima fila de la matriz de controlabilidad es cero, las entradas no tienen efecto sobre el i-ésimo modo, en este caso se dice que el i-ésimo modo no es controlable. De forma similar, si la columna de la matriz de observabilidad es un vector cero, se dice que un modo no es observable.

1.4.6 Estabilizador de Potencia (Power System Stabilizer [PSS])

Es un dispositivo que se agrega al sistema de excitación de un generador para mejorar la estabilidad de pequeña señal del sistema eléctrico de potencia. El objetivo de un PSS es compensar el pobre amortiguamiento inicial del sistema de potencia. Para proporcionar el amortiguamiento, el estabilizador debe producir un componente de par eléctrico en fase con las desviaciones de velocidad del rotor Δw. Esto se lo puede ver en la siguiente ecuación:

$$\dot{\omega} = \frac{1}{2H}(T_m - T_e) \quad (1.14)$$

Donde:

H es la constante de Inercia del generador

T_m es el torque mecánico

T_e es el torque electromecánico

Si el PSS produce una señal en adelanto a un torque eléctrico adicional en fase con $\Delta\omega$ entonces se puede reescribir la ecuación 1.14 se puede reescribir en:

$$\dot{\omega} = \frac{1}{2H}(T_m - T_{e,PSS})$$

$$\dot{\omega} = \frac{1}{2H}(T_m - T_e - k\Delta\omega) \quad (1.15)$$

En [4] me indica que para el modelo simplificado si la velocidad del rotor aumenta el torque de aceleración resultante disminuirá y también la velocidad del rotor (y viceversa). Si no hay desviación de velocidad del rotor $(\Delta\omega = 0)$ la acción del PSS es nula.

Estructura de un PSS Convencional

Este estabilizador de potencia es uno de los más usados por su simpleza en cuanto a la estructura ya que dispone de una ganancia, un filtro washout, dos filtros adelanto – atraso y un limitador de voltaje.

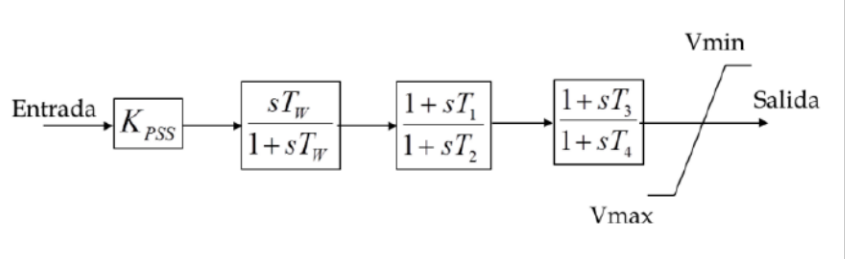


Figura 1.2. Estructura de un PSS Convencional

Métodos para la Sintonización de Estabilizadores de Potencia

Existen diferentes métodos para la sintonización de Estabilizadores de Potencia entre ellos lo que se van a utilizar en este trabajo son los siguientes:

- Algoritmo de Optimización Mean Variance Mapping Optimization (MVMO)
- Algoritmo de Optimización de Ballenas (WOA)
- Método del Residuo (cabe mencionar que este método solo se lo utilizará para realizar un ajuste de Ganancias de los Estabilizadores).

1.4.7 Mean Variance Mapping Optimization (MVMO)

Según [6] el MVMO es un algoritmo de optimización evolutiva. La idea principal del MVMO se basa en una función de mapeo especial aplicada para mutar la descendencia sobre la base de la media y la varianza del conjunto que comprende las n mejores soluciones obtenidas hasta ahora y guardadas en un archivo que se actualiza constantemente.

El procedimiento de este algoritmo se ilustra en la figura 1.3.

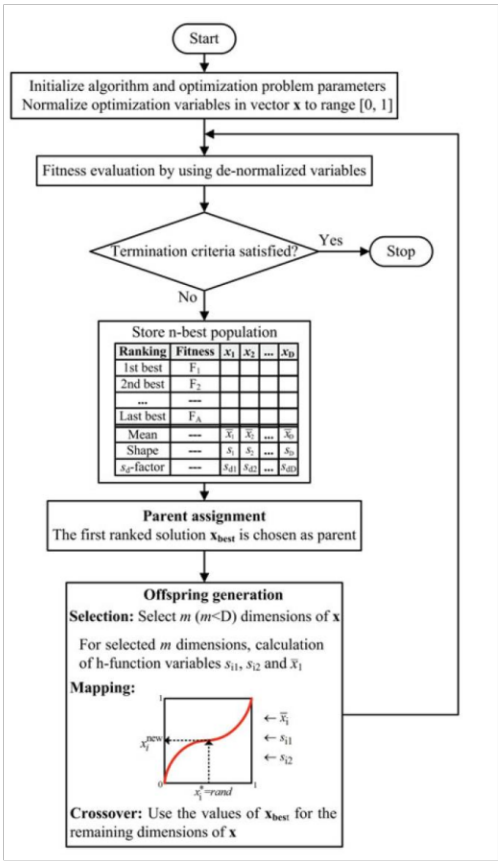


Figura 1.3. Procedimiento de Búsqueda del MVMO

El procedimiento se realiza de la siguiente manera:

1. Comienza con una etapa de inicialización, en este apartado se configuran los parámetros del algoritmo. La solución inicial se genera muestreando aleatoriamente las variables decisión dentro de sus límites [min,max], y cada variable decisión se normaliza a [0,1].
2. Ingresa al siguiente ciclo iterativo, que comprende la evaluación de la aptitud de la solución candidata, el llenado / actualización del archivo solución, aquí se de normalizan las variables para probar la función objetivo.
3. Asigna la mejor solución encontrada hasta el momento y se la coloca en la primera fila de la tabla de mejores soluciones.
4. Creación de nuevas soluciones candidatas, en donde se aplica operadores de cruce y mutación.
5. Finaliza cuando se cumple con el criterio de finalización predefinido, este puede ser el número determinado de iteraciones o que la solución global encontrada cumple con un error muy bajo.

Inicialización

Los parámetros que se necesitan configurar en el MVMO están en la siguiente tabla, después de la aplicación en varios problemas de optimización se llegaron a estos valores de referencia.

Tabla 1.1. Valores Referencia para el Algoritmo MVMO para la sintonización de PSS´s

Nombre	Símbolo	Valor
Tamaño de archivo	A_{sz}	[2,5]
Factor de escala	f_s	[0.9,10]
Factor de asimetría	AF	[1,10]
-	s_d	[10,75]
-	k_d	0.0505/D + 1
Número de variables decisión	D	-

También para la inicialización se necesita la solución candidata inicial la cual se genera muestreando aleatoriamente las variables decisión dentro de sus límites como en la siguiente ecuación:

$$x_i{}^{ini} = x_i{}^{min} + rand\left(x_i{}^{max} - x_i{}^{min}\right), \quad i = 1 \dots D \tag{1.16}$$

Donde:

$x_i{}^{ini}$ es la solución candidata inicial

$x_i{}^{min}$ es el límite inferior de la solución candidata inicial

$x_i{}^{max}$ es el límite superior de la solución candidata inicial

Evaluación de la aptitud

En este caso para realizar la evaluación de la aptitud las variables se de normalizan a su rango normal [min, max] original y se evalúan en las ecuaciones que determinan los valores de la función objetivo y las restricciones (en el caso que las hubiera).

Para un caso general, en donde la aptitud se la evalúa con una estrategia de manejo de restricciones se la puede aproximar a:

$$f^*(x) = f(x) + \sum_{i=1}^{N_{ic}} \gamma_i \cdot max\{0, g_i(x)\} + \sum_{j=1}^{N_{ec}} \vartheta_j \cdot max\{0, |h_j(x)| - \varepsilon\} \tag{1.17}$$

Como en el caso de la sintonización de estabilizadores de potencia no existen restricciones aparentes la aptitud es igual a la función objetivo.

Archivo Solución

El tamaño de este archivo solución es fijo y en este caso es de 4 soluciones. Para que un individuo ingrese a la tabla de mejores soluciones tiene que ser mejor que todos los que están ahí, si es mejor que el primero toma su posición y el primero anterior pasa a ser el segundo.

Las variables de forma, media y varianza se calculan después de cada actualización del archivo para cada variable de optimización x_i, mediante el uso de las ecuaciones (1.18) (1.19) (1.20).

$$\overline{x_i} = \frac{1}{n}\sum_{j=1}^{n} x_i(j) \tag{1.18}$$

$$s_i = -\ln(v_i) \cdot f_s \tag{1.19}$$

donde la varianza v_i se calcula de la siguiente manera:

$$v_i = \frac{1}{n}\sum_{j=1}^{n}(x_i(j) - \overline{x_i})^2 \tag{1.20}$$

Al principio, $\overline{x_i}$ corresponde con el valor inicializado de x_i, y v_i se establece en 1. La variable de forma s_i es una de las entradas de la función de mapeo, que influye fuertemente en la forma de su característica geométrica. Por eso el cálculo de s_i considera el factor de escala f_s, que permite controlar la forma de la función de mapeo y por tanto el proceso de búsqueda.

Designación de Padres, Selección de variables, Mutación y Cruce

El mejor individuo de la tabla (primer lugar), va a ser la base para crear otro individuo nuevo. La diferencia con otros métodos como el de enjambre, crea una población inicial y después otra más, encuentra los mínimos locales y luego globales. Cada vez que muto un individuo es uno solo, no son todos los 4 o 5 que pueden estar en la tabla. Encuentra la solución en el menor tiempo posible.

$$x_i = h_x + (1 - h_1 + h_0) \cdot x_i^* - h_0 \tag{1.21}$$

Xi* este es un valor que vino que está entre 0 y 1, a este valor tenemos una correspondiente función objetivo y algo que le hizo entrar a la tabla.

A partir de esta función matemática se tiene un nuevo valor mutado de esta variable xi.

$$h(\overline{x_i}, s_1, s_2, x) = \overline{x_i} \cdot (1 - e^{-x \cdot s_{i1}}) + (1 - \overline{x_i}) \cdot e^{-(1-x) \cdot s_{i2}} \tag{1.22}$$

Los términos h vienen de ésta en función que es función de la media y de función de s que es la varianza es el logaritmo natural. S1 y s2 son factores de forma.

$$h_x = h(x = x_i{}^*), h_0 = h(x = 0), h_1 = h(x = 1) \quad (1.23)$$

1.4.8 WOA (Whale Optimization Algorithm)

Conforme a [7] Este algoritmo está inspirado en la estrategia de búsqueda de redes de burbujas. Este comportamiento está representado por las siguientes ecuaciones:

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X^*}(t) - \vec{X}(t)\right| \quad (1.24)$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} \quad (1.25)$$

Donde:

t indica la iteración actual

$\vec{A}$ y $\vec{C}$ son vectores coeficientes

$\overrightarrow{X^*}$ es el vector posición de la mejor solución obtenida hasta ahora

$\vec{X}$ es el vector posición

$\overrightarrow{X^*}$ debe actualizarse en cada iteración si hay una mejor solución.

Los vectores $\vec{A}$ y $\vec{C}$ se calculan de la siguiente manera:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \quad (1.26)$$

$$\vec{C} = 2 \cdot \vec{r} \quad (1.27)$$

Donde:

$\vec{a}$ es un valor que decrece linealmente de 2 a 0 en el transcurso de las iteraciones en las fases de exploración y explotación

$\vec{r}$ es un vector aleatorio en [0,1]

Método de ataque con red de burbujas (fase de explotación)

Para modelar matemáticamente el comportamiento de la red de burbujas de las ballenas jorobadas, se diseñan dos enfoques de la siguiente manera:

Posición de actualización en espiral:

Luego se crea una ecuación en espiral entre la posición de la ballena y la presa para imitar el movimiento en forma de hélice de las ballenas jorobadas de la siguiente manera:

$$\vec{X}(t+1) = \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) \tag{1.28}$$

Donde:

$\overrightarrow{D'} = \left|\overrightarrow{X^*}(t) - \vec{X}(t)\right|$ e indica la distancia de la i - ésima ballena a la presa (la mejor solución obtenida hasta ahora).

B es una constante para definir la forma de la espiral logarítmica.

l es un número aleatorio en $[-1,1]$ y es una multiplicación elemento por elemento.

Se debe tener en cuenta que las ballenas jorobadas nadan alrededor de la presa dentro de un círculo que se encoge y a lo largo de un camino en forma de espiral simultáneamente. Para modelar este comportamiento simultáneo, asumimos que hay una probabilidad del 50% de elegir entre el mecanismo de cerco contracción o el modelo en espiral para actualizar la posición de las ballenas durante la optimización.

El modelo matemático es el siguiente:

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} & if\ p < 0.5 \\ \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) & if\ p \geq 0.5 \end{cases} \tag{1.29}$$

Donde:

p es un número aleatorio en $[0,1]$

Además del método de la red de burbujas, las ballenas jorobadas buscan presas al azar. El modelo matemático de la búsqueda es el siguiente.

Búsqueda de presas (fase de exploración)

A diferencia de la fase de explotación, se actualiza la posición de un agente de búsqueda en la fase de exploración de acuerdo con un agente de búsqueda elegido al azar en lugar del mejor agente de búsqueda encontrado hasta ahora.

Este mecanismo y $\left|\vec{A}\right| > 1$ enfatiza la exploración y permite el algoritmo WOA para realizar una búsqueda global. El modelo matemático es el siguiente:

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_{rand}} - \vec{X}\right| \quad (1.30)$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}} - \vec{A} \cdot \vec{D} \quad (1.31)$$

Donde:

$\overrightarrow{X_{rand}}$ es el vector de posición aleatorio (una ballena aleatoria) elegido de la población actual.

1.4.9 Método del Residuo

En [4] como antecedente, se conoce que una función de realimentación de una ganancia K puede cambiar los valores propios de un sistema en el caso de una sola entrada y salida. Esto es usado para incrementar el amortiguamiento de un modo crítico moviendo la parte real de este modo hacia la izquierda.

Este método permite tanto la sintonización de PSS´s como también un ajuste a las ganancias de estos PSS´s. Considerando a un sistema usando a un PSS (representado en la figura 1.4) de una entrada y una salida únicas, la forma general de su función de transferencia T_{PSS} es:

$$T_{PSS} = K_{PSS}|H_{PSS}|e^{jarg(H_{PSS})} \quad (1.32)$$

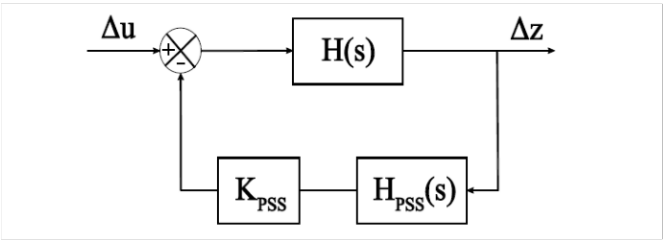


Figura 1.4. Representación en Diagrama de bloques de un sistema de una entrada/salida usando un PSS

De acuerdo con 1.16, si la función de transferencia de retroalimentación es solo una ganancia de relativamente valor pequeño, entonces el desplazamiento del valor propio es opuesto a la dirección del residuo asociados como se muestra en la figura 1.5.

$$\frac{\partial \lambda_i}{\partial k} = -r_i \rightarrow \Delta\lambda_i = -r_i k \quad (1.33)$$

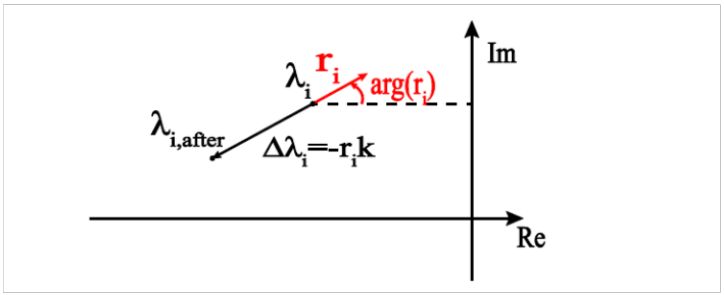


Figura 1.5. Desplazamiento del valor propio debido a una función de transferencia de retroalimentación de ganancia K

Tomando en cuenta toda la función de transferencia del PSS, con la consideración de 1.17 cambia a:

$$\Delta\lambda_i = -r_i K_{PSS} H_{PSS}$$

$$\Delta\lambda_i = -K_{PSS}|r_i||H_{PSS}|e^{j(\arg(r_i)+\arg(H_{PSS}))} \tag{1.34}$$

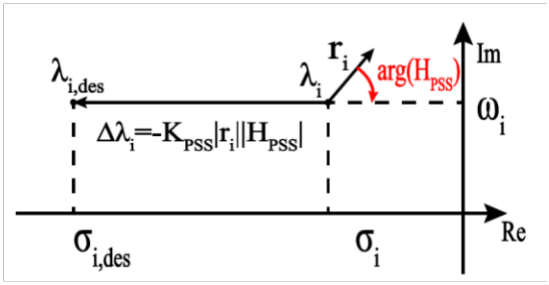


Figura 1.6. Desplazamiento del valor propio con una buena sintonización del PSS

$$\arg(r_i) + \arg(H_{PSS}) = 0 \leftrightarrow \arg(H_{PSS}) = -\arg(r_i) \tag{1.35}$$

Una vez que el desplazamiento de λ_i está en la dirección deseada, el nuevo objetivo es obtener el amortiguamiento deseado $\zeta_{i,des}$ para este modo.

$$\zeta_{i,des} = -\frac{\sigma_{i,des}}{\sqrt{{\sigma_{i,des}}^2+{\omega_i}^2}} \leftrightarrow \sigma_{i,des} = -\frac{\zeta_{i,des}\omega_i}{\sqrt{1-{\zeta_{i,des}}^2}} \tag{1.36}$$

Para valores pequeños de $K_{PSS}|H_{PSS}|$ se tiene:

$$|\Delta\lambda_i| = |\sigma_i - \sigma_{i,des}| = K_{PSS}|H_{PSS}||r_i|$$

$$K_{PSS}|H_{PSS}| = \frac{|\Delta\lambda_i|}{|r_i|} \tag{1.37}$$

Las ecuaciones más importantes que se utilizará posteriormente son 1.20 y 1.21.

2 METODOLOGÍA

Para explicar de una mejor forma la metodología híbrida se planea decir explícitamente que se realizó en el Sistema New England y en la parte de resultados se va a comentar todo lo que se obtuvo del Sistema Nacional Interconectado.

Para el Sistema New England se establece tres puntos de operación Demanda Mínima, Demanda Media y Demanda Máxima. Los datos de Generación y Carga se los visualiza en las tablas 3.1 y 3.2.

Tabla 2.1. Generación New England

DEMANDA	MINIMA	MEDIA	MAXIMA
Name	Act.Pow.	Act.Pow.	Act.Pow.
	[MW]	[MW]	[MW]
G 01	165	493.92	543.312
G 02	145	406.62	447.282
G 03	180	517.32	569.052
G 04	165	473.22	520.542
G 05	125	355.32	390.852
G 06	190	545.22	599.742
G 07	195	560.52	616.572
G 08	165	473.22	520.542
G 09	245	708.12	778.932
G 10	315	997.92	1097.712

Tabla 2.2. Cargas New England

DEMANDA	MINIMA	MEDIA	MAXIMA
Name	Act.Pow.	Act.Pow.	Act.Pow.
	[MW]	[MW]	[MW]
Load 03	80	241.2	265.32
Load 04	75	225.9	248.49
Load 07	90	272.7	299.97
Load 08	160	450	495
Load 12	30	90.9	99.99
Load 15	100	301.5	331.65
Load 16	135	406.8	447.48
Load 18	40	120.6	132.66
Load 20	205	482.4	530.64
Load 21	110	333	366.3
Load 23	65	196.2	215.82
Load 24	125	378	415.8

Load 25	55	165.6	182.16
Load 26	30	85.5	94.05
Load 27	70	212.4	233.64
Load 28	45	135.9	149.49
Load 29	75	225.9	248.49
Load 31	60	180.9	198.99
Load 39	340	1026	1128.6

2.1 Análisis Modal Sin PSS

Para cada punto de operación se realiza un análisis Modal sin la presencia de estabilizadores de potencia, esto se hace para conocer en que escenarios se van a utilizar los estabilizadores y cuáles no.

Tabla 2.3. Análisis Modal Demanda Mínima sin PSS´s

Name	**Real part**	**Imaginary part**	**Damped Frequency**	**Damping Ratio**
	1/s	**rad/s**	**Hz**	**%**
Mode 00024	-0.60535108	7.32088816	1.16515554	8.240695982
Mode 00041	-0.77168673	7.08003638	1.12682279	10.83530322
Mode 00044	-0.85990288	7.79372089	1.2404092	10.96672904
Mode 00039	-0.76044842	6.88852485	1.09634278	10.97269264
Mode 00028	-0.62075008	4.25903728	0.67784684	14.42251003

En este caso al momento de realizar el análisis modal todos los modos cumplen con el objetivo de ser mayor al 5% por lo tanto en demanda mínima del Sistema New England no son necesarios los PSS´s.

Tabla 2.4. Análisis Modal Demanda Media sin PSS´s

Name	**Real part**	**Imaginary part**	**Damped Frequency**	**Damping Ratio**
	1/s	**rad/s**	**Hz**	**%**
Mode 00012	-0.1556663	6.79807425	1.08194712	2.2892587
Mode 00019	-0.29121593	6.71714433	1.06906672	4.33134411
Mode 00015	-0.22982378	3.75938655	0.59832495	6.10194022
Mode 00028	-0.49763921	7.67205028	1.22104473	6.47278916
Mode 00031	-0.58537051	8.573822	1.36456615	6.81155905

De este análisis modal es necesario que para modos que no cumplan con el amortiguamiento si son locales se obtenga el factor de participación de estos modos para conocer cual generador es el que más participa. Y para un modo entre área es necesario

obtener la observabilidad para verificar como se comparta el sistema y que generadores oscilan con otros.

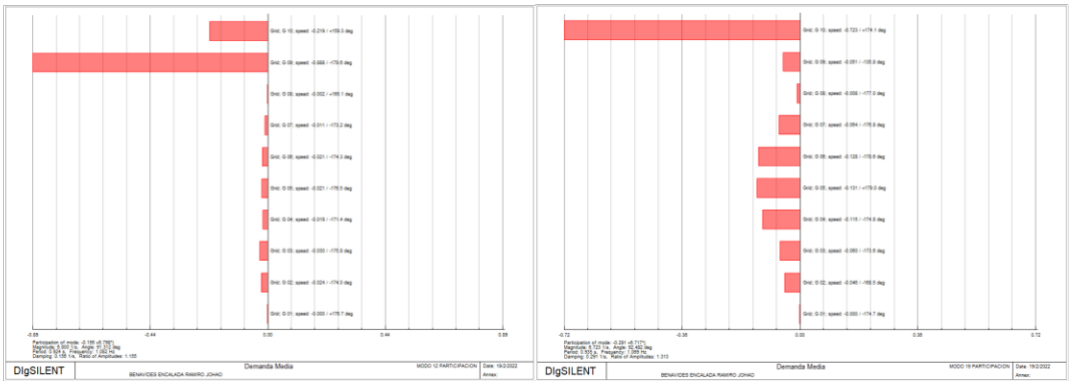


Figura 2.1. Demanda Participación media modo 12 y 19

En el análisis modal se puede ver que tiene 2 modos locales que no cumplen con el 5% de amortiguamiento, el modo 12 y 19, estos modos tienen una participación del Generador 9 y el Generador 10 respectivamente.

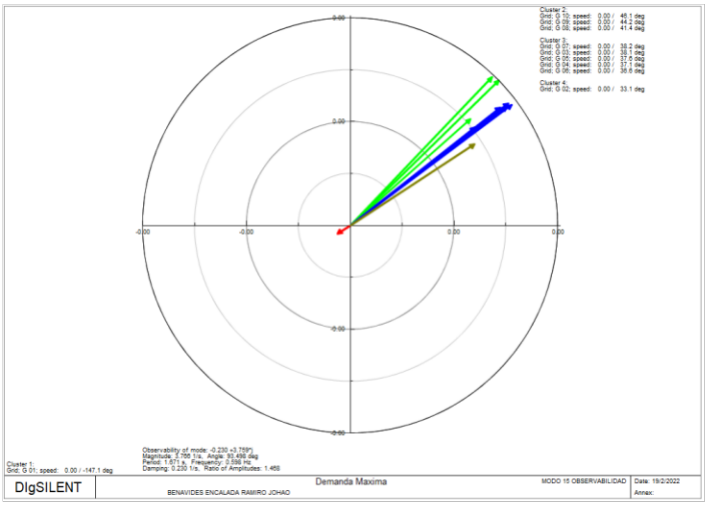


Figura 2.2. Demanda media Observabilidad modo inter-área 15

En cambio, como se puede ver en la gráfica este modo inter-área el Generador 1 (el fasor de color rojo), si hubiera una falla oscilaría en contra de todos los demás generadores.

Tabla 2.5. Análisis Modal Demanda Máxima sin PSS´s

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00014	-0.03641868	3.55026802	0.5650427	1.02574684
Mode 00016	-0.09046247	6.67197267	1.06187743	1.35573326
Mode 00018	-0.14593293	6.48893472	1.03274604	2.24838194
Mode 00023	-0.42025957	7.62514772	1.21357995	5.50314217

Mode 00035	-0.55527466	8.63422168	1.37417906	6.41783306

En este punto de operación al momento de realizar el análisis modal se obtuvo que hay 3 modos que no cumplen con el amortiguamiento del 5%, 1 modo inter-área y dos modos locales.

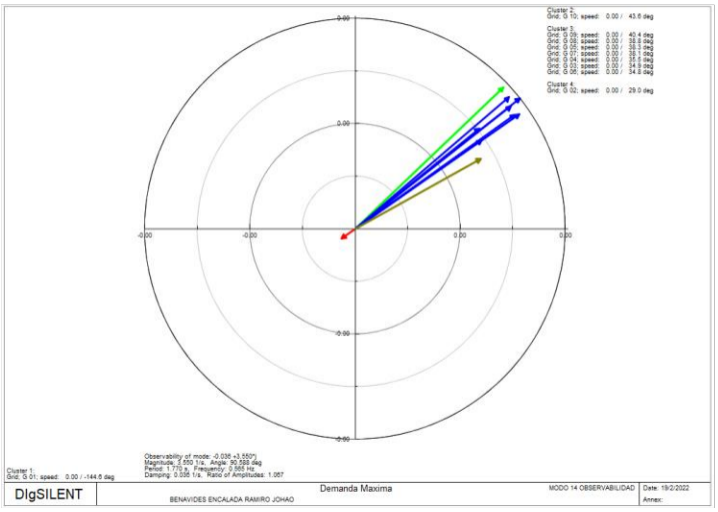


Figura 2.3. Demanda máxima Observabilidad modo inter-área 14

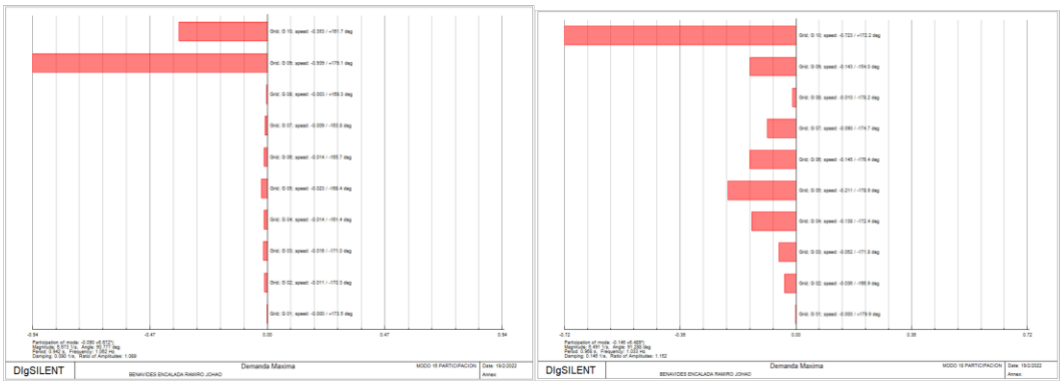


Figura 2.4. Demanda Participación media modo 16 y 18

Como se puede ver en las gráficas anteriores la observabilidad del modo inter-área muestra lo mismo que el de demanda media en que el generador 1 oscila en contra los demás generadores. Se forman dos áreas una en contra de otra.

Con los resultados anteriores los escenarios a los que se aplicará los algoritmos de optimización son demanda media y máxima.

2.2 Sintonización Estabilizadores de Potencia

Para la ubicación de los estabilizadores de potencia se consultó de [8] y se tiene que se deben instalar en los Generadores 5,6,9 y 10.

2.2.1 Función Objetivo

La función objetivo que se utiliza para la sintonización de PSS´s es la siguiente:

$$minOF = |\zeta_{th} - \zeta_{sys}| \tag{2.1}$$

$$\zeta_{sys} = \min\left(\zeta_p\right)_{p=1\ldots nm} \tag{2.2}$$

Y está sujetado a los límites:

$$x_{min} \leq x \leq x_{max} \tag{2.3}$$

Donde:

ζ_{th} es el límite del amortiguamiento mínimo aceptable (5%)

ζ_{sys} es el amortiguamiento global mínimo del sistema

x es el vector que constituye la solución al problema de optimización (ganancias y constantes de tiempo).

La constante de tiempo Tw es igual a 10 s.

Los límites para la ganancia Kpss es de 0 a 100.

Y por último los límites de las constantes de tiempo de T1 y T3 van de 0.01 a 0.2 y T1/T2 y T3/T4 va de 1 a 15.

Estos límites y esta función objetivo se utilizarán con los algoritmos MVMO y WOA.

2.2.2 Resultados Algoritmos Demanda Media

Tabla 2.6. Demanda Media Resultados MVMO

Name	Kpss	Tw	T1	T2	T3	T4
PSS 5	71.64883	10	0.075776	0.007807	0.115736	0.025876
PSS 6	45.38297	10	0.116597	0.070105	0.107783	0.02586
PSS 9	32.47004	10	0.054721	0.00622	0.025664	0.008617
PSS 10	54.97789	10	0.419967	1.38332	0.08865	0.019604

Tabla 2.7. Análisis Modal Demanda Media MVMO

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%

Mode 00017	-0.3736038	7.31782688	1.16466832	5.09875212
Mode 00031	-0.59414148	8.59983004	1.36870546	6.89232914
Mode 00039	-0.74779103	7.97289097	1.26892501	9.33818688
Mode 00045	-0.78912907	8.47800399	1.34931624	9.26789721
Mode 00055	-0.88955573	4.67946031	0.74475924	18.6753502

Tabla 2.8. Demanda Media Resultados WOA

Name	Kpss	Tw	T1	T2	T3	T4
PSS 5	70.43061	10	0.137038	0.019757	0.089996	0.012077
PSS 6	43.70357	10	0.103303	0.014825	0.119323	0.01057
PSS 9	8.968803	10	0.046153	0.006854	0.036483	0.002674
PSS 10	4.102581	10	0.158888	0.065289	0.129016	0.017078

Tabla 2.9. Análisis Modal Demanda Media WOA

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00019	-0.36954207	7.24249035	1.15267814	5.09578804
Mode 00029	-0.5383355	7.08309166	1.12730905	7.57843324
Mode 00033	-0.59866888	8.59667979	1.36820408	6.94712942
Mode 00053	-0.96020178	8.10190578	1.28945835	11.7691881
Mode 00039	-0.77175574	4.23125948	0.67342586	17.9433622

2.2.3 Resultados Algoritmos Demanda Máxima

Tabla 2.10. Demanda Máxima Resultados MVMO

Name	Kpss	Tw	T1	T2	T3	T4
PSS 5	98.28391	10	0.100186	0.007791	0.011967	0.000977
PSS 6	65.50983	10	0.193041	0.016702	0.083431	0.007676
PSS 9	99.64141	10	0.030198	0.003547	0.11926	0.016298
PSS 10	63.69583	10	0.199688	0.472024	0.175527	0.035043

Tabla 2.11. Análisis Modal Demanda Máxima MVMO

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00019	-0.39397248	7.71004666	1.22709204	5.10320056
Mode 00033	-0.56097876	8.65483443	1.37745968	6.46810823
Mode 00041	-0.78026209	9.69251947	1.54261238	8.02418887
Mode 00047	-0.84319676	8.21907227	1.30810598	10.2054616
Mode 00059	-1.23859144	5.24043442	0.83404104	23.0015483

Tabla 2.12. Demanda Máxima Resultados WOA

Name	Kpss	Tw	T1	T2	T3	T4
PSS 5	24.17279	10	0.063741	0.008989	0.110284	0.032354
PSS 6	35.2321	10	0.126212	0.012394	0.059411	0.012484
PSS 9	51.60434	10	0.169132	0.012096	0.137945	0.023662
PSS 10	20.80956	10	0.160557	0.112781	0.025726	0.001924

Tabla 2.13. Análisis Modal Demanda Máxima WOA

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00021	-0.36892795	7.22397347	1.14973109	5.10034795
Mode 00031	-0.55680531	8.64050578	1.37517921	6.43078997
Mode 00035	-0.64086784	7.78479147	1.23898804	8.20455124
Mode 00046	-0.88402301	9.36157578	1.48994106	9.40127696
Mode 00057	-1.1579798	4.3826724	0.69752398	25.5451492

Como se puede ver en los análisis modales para las dos demandas la media y la máxima ya cumple con el amortiguamiento que se colocó así que todo funciona bien.

2.3 Respuesta Transitoria Demanda Media sin PSS, MVMO y WOA

Para probar la estabilidad transitoria del sistema se hizo una falla con las siguientes características:

Cortocircuito trifásico al 50% en la línea 26-29 a 0.01 [s] cercana al generador 9 y apertura de la línea 0.1 [s] limpiando el cortocircuito.

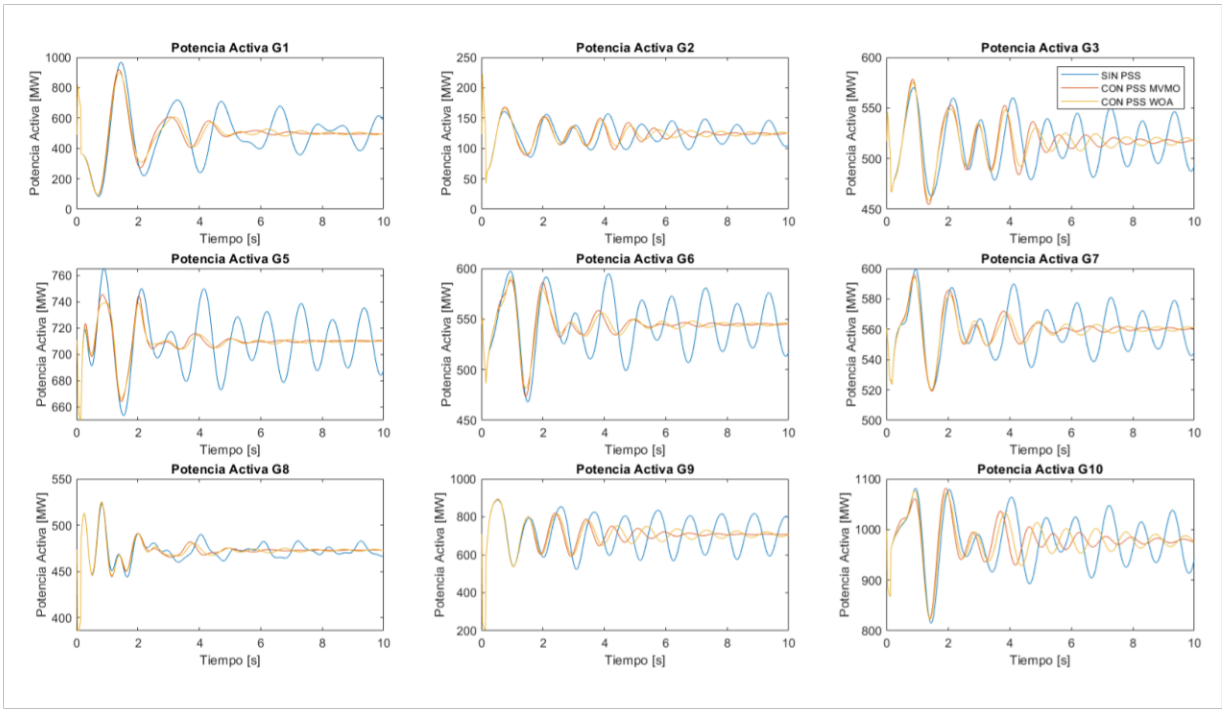


Figura 2.5. Potencia Activa de Generadores

2.3.1 Descomposición de la Señal en Demanda Media

Como en demanda media sin PSS´s el modo con menor amortiguamiento es un modo local a este modo se le tiene que revisar cuál es su factor de participación, y como se pueden ver en la figura 1.3 el Generador 9 es el que más participa. Para saber cual señal tomar siempre para obtener los modos oscilatorios parecidos a un modo local (frecuencia 0.8 a 2 Hz) se descompuso 6 señales Potencia Activa 9 [MW], Velocidad 9 [pu], Frecuencia 9 [Hz], Firot 9 [deg] y Voltaje Barra 9 [pu].

Para la Potencia Activa del Generador 9 se cargará los datos al programa ringdown y se le pondrá los tiempos: tiempo inicial (ti) cuando la señal llega a su valor pico más alto y el tiempo final (tf) se colocará cuando la señal toma su menor valor pico ósea cuando la onda ya se está acabando.

Por ejemplo, para la Potencia Activa del Generador 9, el tiempo inicial es 0.5 [s] y el tiempo final es 9.9 [s]

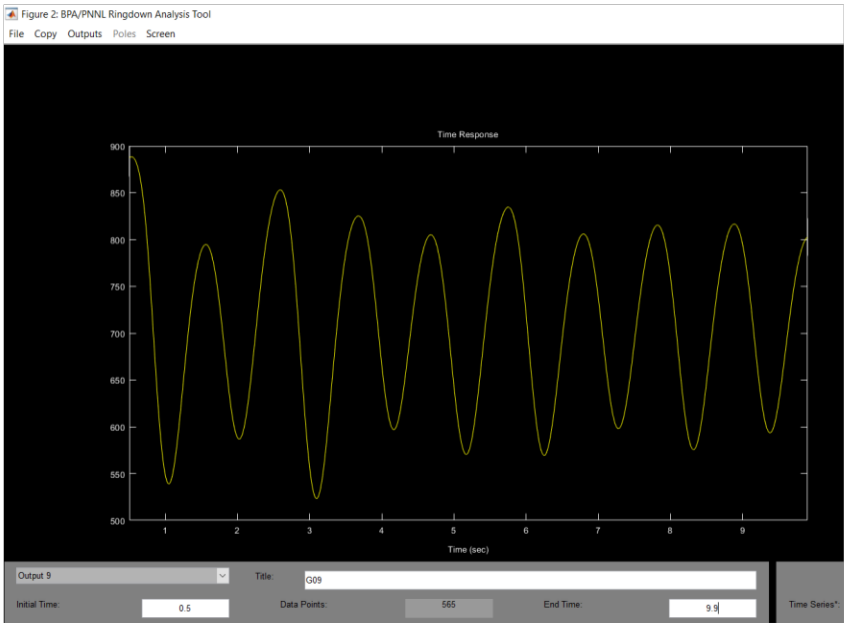


Figura 2.6. Cargar la señal Potencia Activa 9 en ringdown colocando el ti y el tf

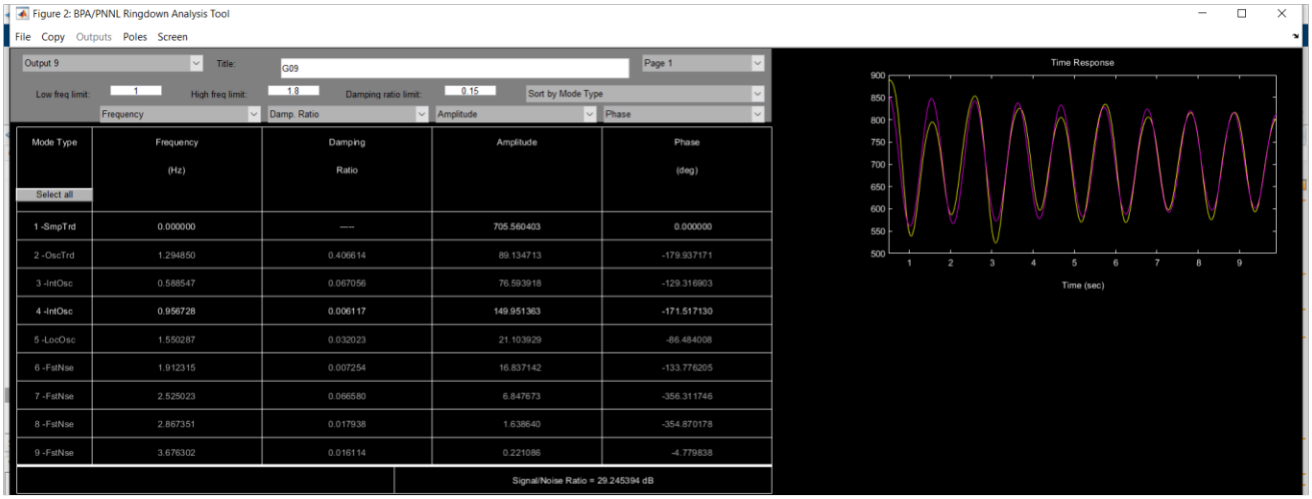


Figura 2.7. Onda parecida a la Potencia Activa 9

Ya en la otra pantalla del software de Matlab ringdown después de haber presionado en la opción de Análisis Prony, mostrará todos los modos oscilatorios que conforman la señal original, cada modo oscilatorio tiene su propia frecuencia, amortiguamiento, amplitud, ángulo y algunos datos más. Los datos de interés son la frecuencia y el amortiguamiento, en este caso para esta señal, primero se da clic en la componente DC de la señal (siempre es el primer modo oscilatorio), después se irá probando entre todos los modos oscilatorios y el modo oscilatorio que se parezca más a la señal original será el elegido, el resultado para Potencia del Generador 9 SIN PSS es:

Tabla 2.14. Demanda Media Resultados WOA

SIN PSS	tiempo inicial [s]	tiempo final [s]
	0.5	9.9
POTENCIA GENERADOR 9	FRECUENCIA [Hz]	AMORTIGUAMIENTO [pu]
	0.956727817	0.006117221

Estos tiempos, el tiempo inicial y el tiempo final se utilizarán con las señales Potencia Generador 9 con PSS MVMO y con PSS WOA.

Después de hacer el mismo procedimiento con las demás señales se obtuvo lo siguiente:

Tabla 2.15. Resultados Modos Oscilatorios señales Generador 9

SEÑAL		FRECUENCIA [Hz]	AMORTIGUAMIENTO [pu]
POTENCIA GENERADOR 9	SIN PSS	0.956727817	0.006117221
	MVMO	1.149520845	0.080223322
	WOA	1.028736214	0.047347573
VELOCIDAD GENERADOR 9	SIN PSS	0.956511768	0.006200592
	MVMO	1.133656451	0.05822476
	WOA	1.028610927	0.047420493
FRECUENCIA GENERADOR 9	SIN PSS	0.956071275	0.00542318
	MVMO	1.050408673	0.087541324
	WOA	1.027808261	0.047035266
FIROT GENERADOR 9	SIN PSS	0.957591324	0.005697991
	MVMO	1.152766625	0.062666264
	WOA	1.032957362	0.048996151
VOLTAJE BARRA GENERADOR 9	SIN PSS	0.956865127	0.005523842
	MVMO	1.022735543	0.085014858
	WOA	1.029091741	0.047341139

Con los resultados anteriores se puede ver que con cualquiera de estas señales se asemeja al modo local (0.8 a 2 Hz) que se está buscando. De aquí en adelante se realizará la descomposición solo de la potencia del generador que más participa en el modo local de estudio.

Tabla 2.16. Modo Local vs Modos Oscilatorios Potencia Activa Generador 9

SEÑAL Y MODO		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL	SIN PSS	1.08194712	2.2892587
	MVMO	1.16466832	5.09875212
	WOA	1.15267814	5.09578804
POTENCIA GENERADOR 9	SIN PSS	0.956727817	0.6117221
	MVMO	1.149520845	8.0223322

	WOA	1.028736214	4.7347573

Como se puede ver en la tabla anterior el modo local vs Modos Oscilatorios de la Potencia Activa Generador 9, las frecuencias y amortiguamientos obtenidos mediante la descomposición de la señal se asemejan bastante a las frecuencias y amortiguamientos del modo local.

Ahora bien, para escoger cual algoritmo es mejor en el dominio del tiempo se verifica simplemente que valor tiene en el amortiguamiento de la tabla anterior, en este caso MVMO es el mejor ya que tiene un amortiguamiento de 8% comparado con el 4.7% del WOA.

2.4 Respuesta Transitoria Demanda Máxima sin PSS, MVMO y WOA

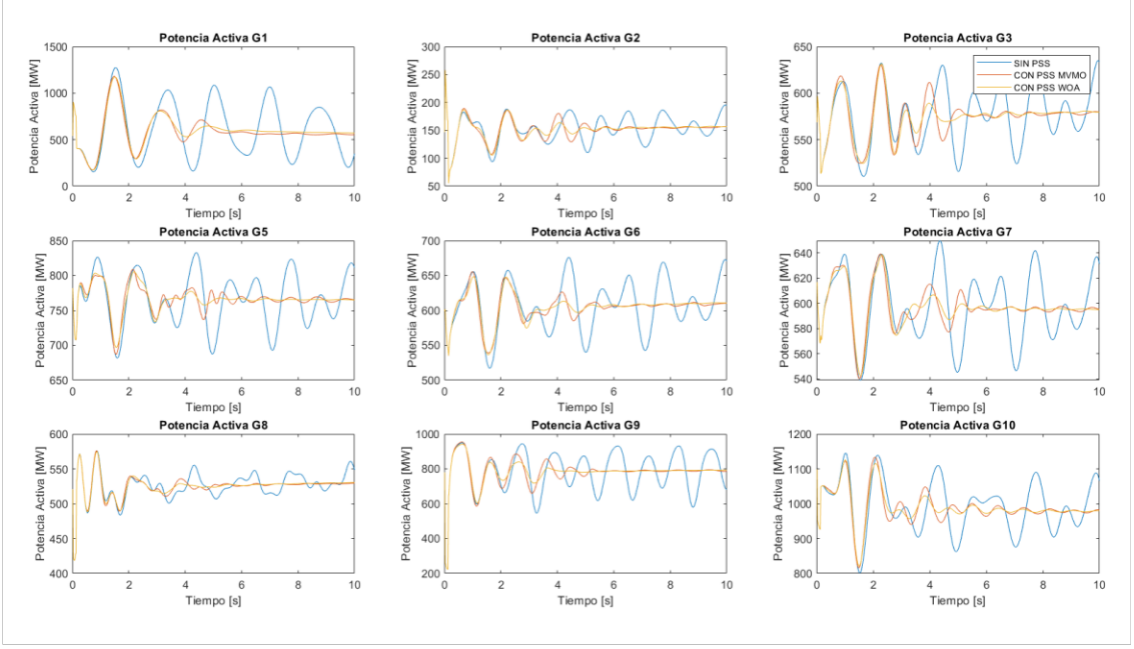


Figura 2.8. Potencia Activa Generadores

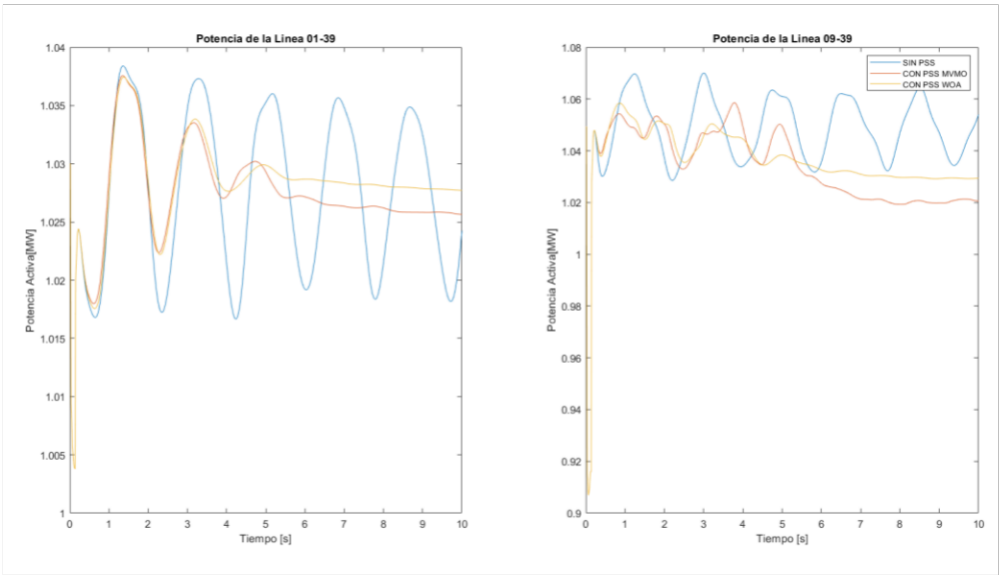


Figura 2.9. Potencia Activa Línea 01-39 y Línea 09-39

2.4.1 Descomposición de la Señal Demanda Máxima

Para este punto de operación de demanda máxima como se mencionó anteriormente existen 3 modos que no cumplen con el amortiguamiento, 1 inter-área y 2 locales.

Del modo inter-área se obtuvo la observabilidad y que el generador 1 oscila en contra de todos los demás lo que quiere decir que para encontrar un modo inter-área, se tiene que ver en la interconexión de las áreas tal como lo dice la teoría, existen entonces dos señales que se pueden descomponer la potencia activa de la Línea 01-39 y la potencia activa de la línea 09-39, porque estas dos líneas salen de la barra del generador 1, en este caso se optó por descomponer señal de la Potencia Activa de la línea 01-39 aunque da igual cualquiera de las dos señales vale.

Para el modo local con menor amortiguamiento se descompuso la señal de la potencia activa del generador 9, porque de ese modo el generador que más participa es el 9.

Tabla 2.17. Modo Inter-área vs Modos Oscilatorios Potencia Activa LT01-39

SEÑAL Y MODO		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO INTERAREA	SIN PSS	0.5650427	1.02574684
	MVMO	0.83404104	23.0015483
	WOA	0.69752398	25.5451492
POTENCIA LT01-39	SIN PSS	0.554349574	1.424750354
	MVMO	0.633371872	13.80274333
	WOA	0.615297641	25.41314745

Tabla 2.18. Modo Local vs Modos Oscilatorios Potencia Activa Generador 9

SEÑAL Y MODO		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL	SIN PSS	1.06187743	1.35573326
	MVMO	1.22709204	5.10320056
	WOA	1.14973109	5.10034795
POTENCIA GENERADOR 9	SIN PSS	0.906626221	0.388797905
	MVMO	0.993927807	8.802897572
	WOA	0.970776972	12.76712135

De los resultados anteriores para el modo inter-área (0.2 a 0.8 Hz) se obtuvo que para la potencia activa de la línea 01-39 se consigue frecuencias y amortiguamientos muy parecidos, por lo tanto, sí se puede encontrar un modo inter-área descomponiendo la señal.

En este caso, para escoger cual algoritmo presenta una mejor estabilidad transitoria (un mayor amortiguamiento en la señal) se lo ve en el modo inter-área y no en el local, por la razón de que el modo inter-área es mucho más importante que esté amortiguado. El algoritmo con mayor amortiguamiento de la señal es el WOA con un 12.76% comparado con el 8.80% del MVMO.

Con esto se llega a la conclusión, de que, si solo aparecen modos locales, se toma al modo local con menor amortiguamiento y de este se obtiene el factor de participación. Teniendo este factor se descompondrá la señal de la Potencia Activa del Generador que más participa y por último se decidirá cual algoritmo presenta mejor amortiguamiento.

Ahora bien, si aparece un modo inter-área poco amortiguado cuando no están activos los PSS´s; de este modo inter-área se obtiene la observabilidad y se verifica como oscilan los generadores los unos con los otros formando así áreas, entonces para saber cuál señal descomponer se la tiene hacer a la potencia de la interconexión entre áreas, el caso anterior era la Línea 01-39 porque de la barra del generador 1 sale esta línea.

2.5 Ajuste Demanda Media a Demanda Máxima

Para realizar el ajuste de las Ganancias de los Estabilizadores de Potencia, se llevará la sintonización de demanda media a máxima y se realizará un ajuste suponiendo que el sistema está mal modelado y al momento de llevarlo al mundo real con la sintonización inicial no cumple con el amortiguamiento del 5%.

Tabla 2.19. Análisis Modal Demanda Máxima con los datos de MVMO Demanda Media

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00019	-0.312871445	7.197505412	1.145518564	4.342841665
Mode 00033	-0.555031645	8.652364752	1.37706662	6.401639387
Mode 00037	-0.682416027	7.930723538	1.262213853	8.573034042
Mode 00043	-0.755395744	8.403452373	1.337450984	8.953012731
Mode 00051	-0.861204716	4.504850152	0.71696917	18.77723079

Tabla 2.20. Análisis Modal Demanda Máxima con los datos de WOA Demanda Media

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00019	-0.324472333	7.154956356	1.138746672	4.530274988
Mode 00031	-0.502133769	6.931899304	1.103246039	7.224881524
Mode 00035	-0.561533667	8.65019414	1.376721156	6.477938159
Mode 00050	-0.868567026	9.610777254	1.529602707	9.000745407
Mode 00041	-0.687085415	4.022597167	0.640216223	16.8368021

Como se puede ver en las tablas al momento de comprobar si la sintonización de Demanda Media en Demanda Máxima cumple con el objetivo del 5% no cumple un modo, por lo que se tiene que realizar un ajuste, en este caso con el método del residuo.

La metodología del método del residuo estimado es la siguiente:

1. Se obtiene la participación del modo que no cumple con el amortiguamiento. Teniendo esta información se conoce a que Estabilizador de Potencia se va a modificar el parámetro de la ganancia. De este estabilizador se obtiene las constantes de tiempo para calcular HPSS y también del análisis modal se obtiene la parte real, imaginaria y el amortiguamiento que tiene en este momento.

$$HPSS = \frac{Tw*\sigma}{1+\sigma*Tw} * \frac{1+\sigma*T1}{1+\sigma*T2} * \frac{1+\sigma*T3}{1+\sigma*T4} \quad (2.4)$$

   Donde: σ es la parte real del modo. Todas las Ts son las constantes de tiempo del Estabilizador de Potencia.

2. A la ganancia del estabilizador previamente escogido se le multiplicará por 1.01 y 0.99. Esto se hace para saber si a la ganancia de este estabilizador se le tiene que sumar o restar. Por ejemplo, se multiplica por 1.01 se realiza nuevamente un análisis modal y se verifica si el amortiguamiento se incrementó, si es así a la ganancia de este estabilizador se le tendrá que sumar. También de ese análisis modal se obtendrá la parte real e imaginaria del modo de estudio. Para poder

saber la distancia entre la parte real inicial y la parte real después de multiplicarle por 0.99 o 1.01.

3. A partir de los datos de HPSS, K, y la distancia entre la parte real inicial y la parte real después de multiplicar ya sea 1.01 o 0.99, se obtiene el residuo.

$$|r_i| = \frac{|\Delta\lambda_i|}{|1.01 * K_{PSS} - K_{PSS}||H_{PSS}|}$$

4. Teniendo el residuo simplemente se aplica las fórmulas y se podrá mover al polo hasta el amortiguamiento deseado de 5%.

Tabla 2.21. Valores MVMO Demanda Media Ajustado

Name	Kpss	Tw	T1	T2	T3	T4
PSS 5	71.64883	10	0.075776	0.007807	0.115736	0.025876
PSS 6	45.38297	10	0.116597	0.070105	0.107783	0.02586
PSS 9	32.47004	10	0.054721	0.00622	0.025664	0.008617
PSS 10	6.203356	10	0.419967	1.38332	0.08865	0.019604

Después de ocupar el método del residuo se cambió la Ganancia del PSS 10 de 54.97789 a 6.203356.

Tabla 2.22. Análisis Modal Demanda Máxima con los datos de MVMO Demanda Media ajustado

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00019	-0.392882034	6.878429767	1.094736098	5.702503758
Mode 00033	-0.555103734	8.652284165	1.377053794	6.402526833
Mode 00037	-0.676688518	7.894531209	1.256453665	8.540294729
Mode 00046	-0.772982613	9.514581814	1.514292727	8.097510215
Mode 00041	-0.707121378	4.294486548	0.683488762	16.24702197

Tabla 2.23. Valores WOA Demanda Media Ajustado

Name	Kpss	Tw	T1	T2	T3	T4
PSS 5	70.43061	10	0.137038	0.019757	0.089996	0.012077
PSS 6	43.70357	10	0.103303	0.014825	0.119323	0.01057
PSS 9	11.74784	10	0.046153	0.006854	0.036483	0.002674
PSS 10	4.102581	10	0.158888	0.065289	0.129016	0.017078

Después de ocupar el método del residuo se cambió la Ganancia del PSS 9 de 8.968803 a 11.74784.

Tabla 2.24. Análisis Modal Demanda Máxima con los datos de WOA Demanda Media ajustado

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00019	-0.368099242	7.265551736	1.156348473	5.059873551
Mode 00031	-0.518671258	6.932257241	1.103303007	7.461142006
Mode 00035	-0.561550258	8.650215364	1.376724534	6.478112925
Mode 00051	-0.868505034	9.610901908	1.529622546	8.999992418
Mode 00041	-0.723281219	4.044416383	0.643688859	17.60416019

2.5.1 Respuesta en el Dominio del Tiempo Algoritmo Seteado vs Ajustado

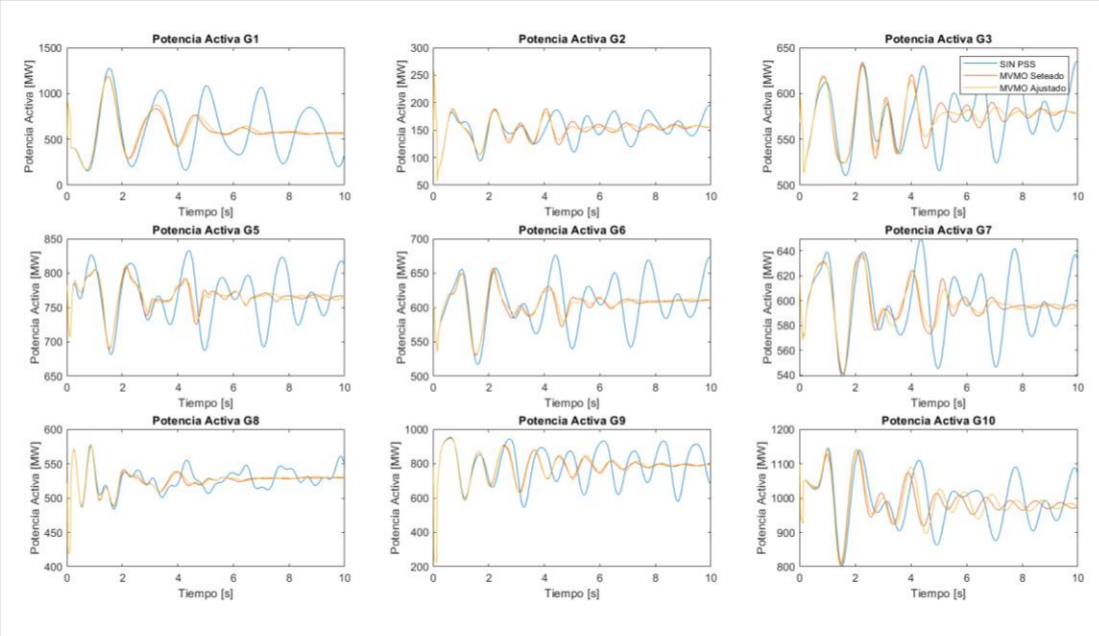


Figura 2.10. Potencias Activas de Generadores MVMO Seteado vs MVMO Ajustado

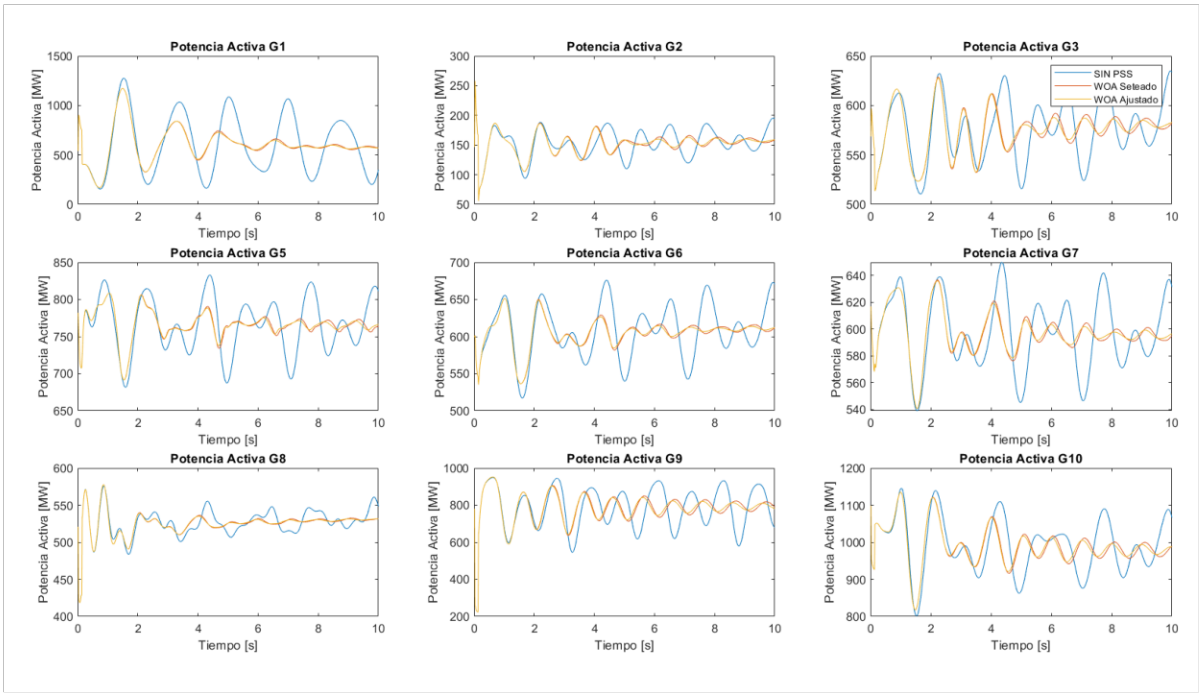


Figura 2.11. Potencias Activas de Generadores WOA Seteado vs WOA Ajustado

De acuerdo con las gráficas se puede visualizar de que al momento de realizar el ajuste la respuesta en el dominio del tiempo se vio mejorada.

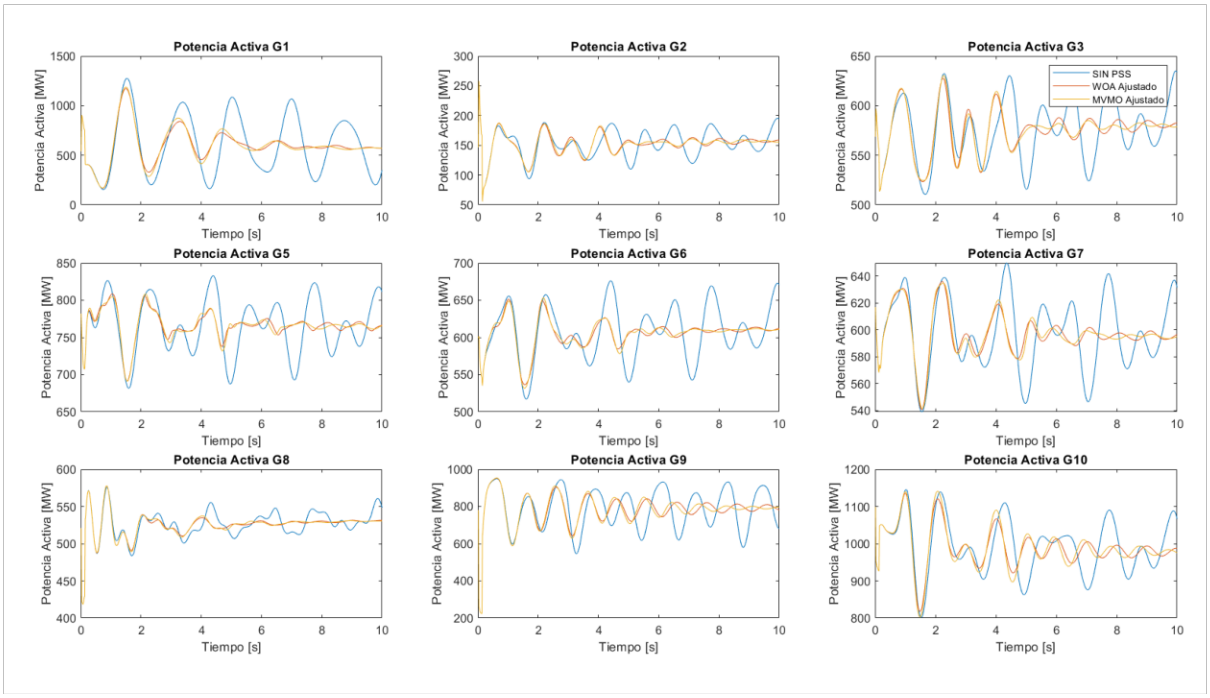


Figura 2.12. Potencias Activas de Generadores MVMO Ajustado vs WOA Ajustado

2.5.2 Descomposición de la Señal MVMO Ajustado vs WOA Ajustado

Se realiza la descomposición de la señal ya con los valores de los PSS´s Ajustados.

Y se obtuvo los siguientes resultados:

Tabla 2.25. Modo inter-área vs Modos Oscilatorio Potencia LT01-39

SEÑAL Y MODO		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO INTERAREA	MVMO	0.843023256	23.77220811
	WOA	0.652085326	19.84136039
POTENCIA LT01-39	MVMO	0.606974604	12.22335455
	WOA	0.600843381	13.64125727

Tabla 2.26. Modo local vs Modos Oscilatorio Potencia Generador 9

SEÑAL Y MODO		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL 1	MVMO	1.239708145	5.101713617
	WOA	1.053576472	5.500969647
POTENCIA GENERADOR 9	MVMO	1.061014085	8.112367941
	WOA	1.000695144	4.286062135

Para escoger el mejor algoritmo con mayor amortiguamiento en el dominio del tiempo se situará en el punto de operación actual ósea Demanda Máxima, en este punto existe un modo inter-área, por lo tanto, el algoritmo con mayor amortiguamiento es el WOA con un 13.64% contrario al 12.22% del MVMO.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

3.1.1 Sistema Nacional Interconectado Análisis Modal sin PSS

Tabla 3.1. Análisis Modal Demanda Mínima sin PSS´s

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00100	-0.319462171	7.309188657	1.163293505	4.366523959
Mode 00138	-0.46961228	8.439894137	1.343250871	5.555602822
Mode 00152	-0.565769086	7.396536717	1.17719538	7.626828742

Mode 00170	-0.686280312	8.335211328	1.326590085	8.205741597
Mode 00095	-0.270099038	3.136886728	0.499251029	8.578675209

Existe un modo local que no cumple con el amortiguamiento del 5%.

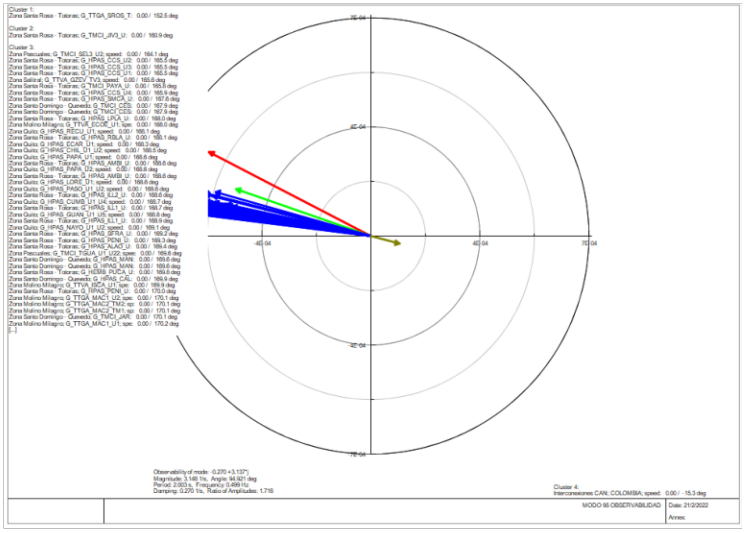


Figura 3.1. Observabilidad modo entre área 95

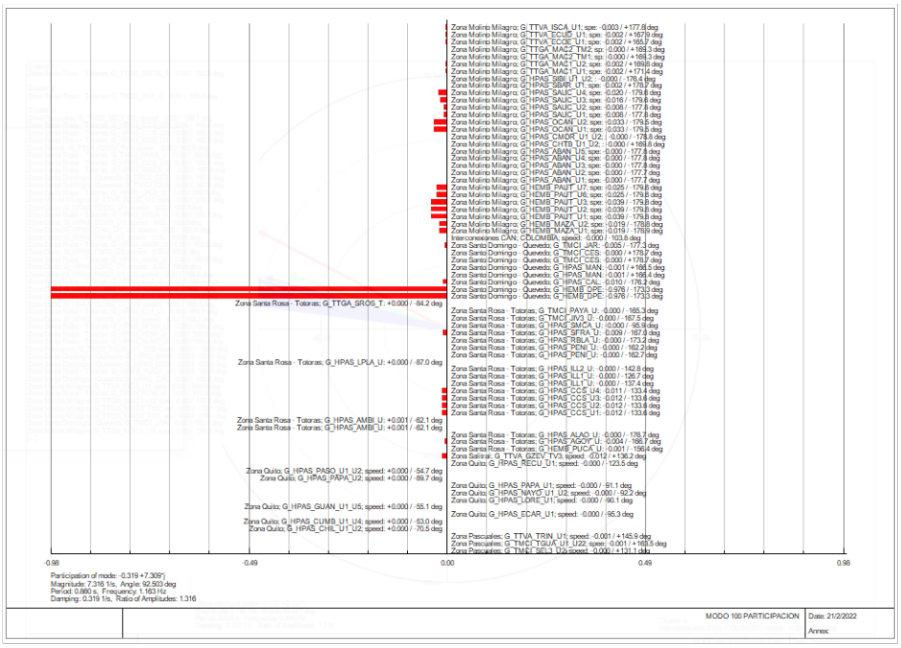


Figura 3.2. Participación modo local 100

Como se puede ver en la gráfica de la observabilidad del modo entre área el generador COLOMBIA (que es una simplificación de todo el sistema colombiano) oscila en contra de todos los generadores ecuatorianos.

Y también del modo local poco amortiguado podemos notar que el generador que más participa es Daule Peripa 1.

Tabla 3.2. Análisis Modal Demanda Media sin PSS´s

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00121	-0.276439895	7.289120422	1.160099546	3.789775592
Mode 00167	-0.459628475	8.503780578	1.353418714	5.397112036
Mode 00169	-0.472067993	6.541375296	1.041092213	7.197928376
Mode 00178	-0.50999622	7.037396318	1.120036411	7.227989503
Mode 00098	-0.194358299	2.902693102	0.461977956	6.68083266

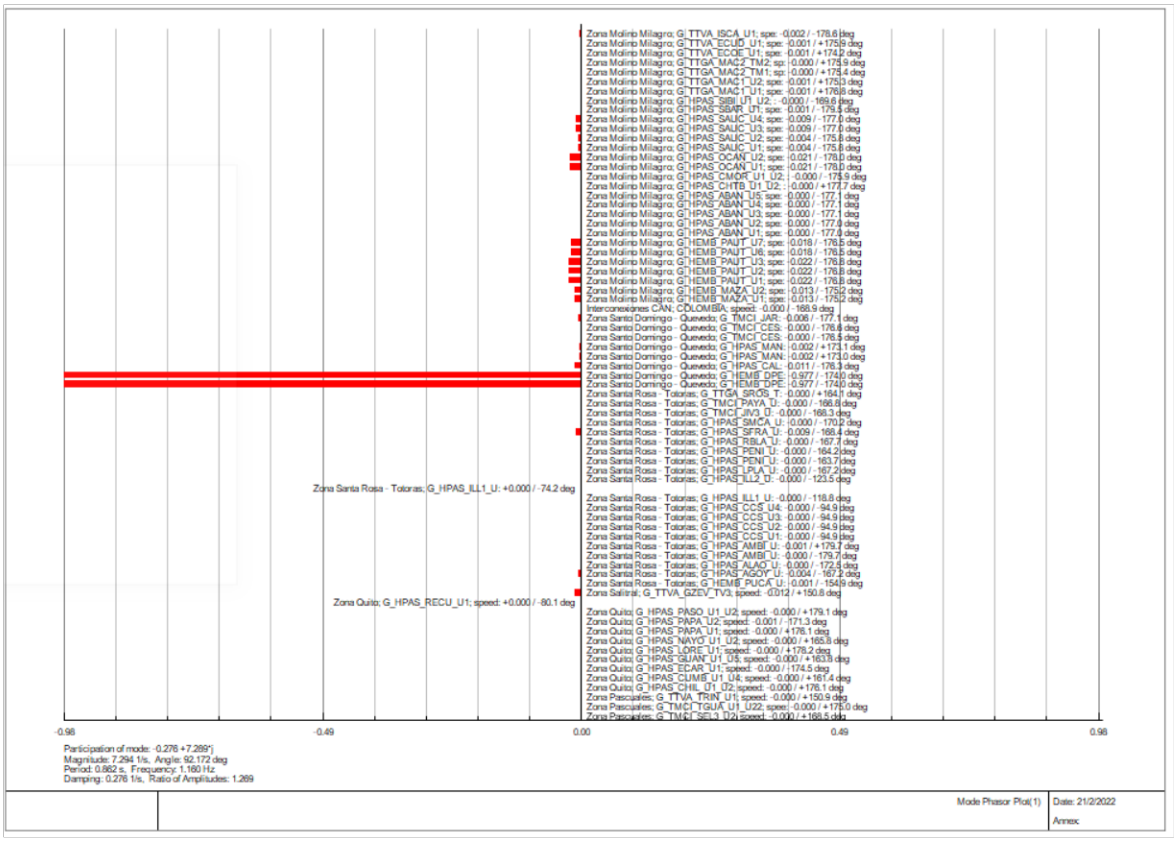


Figura 3.3. Participación modo local 121

Sucedió una respuesta muy parecida a la demanda mínima solo existe un modo que no cumple con el amortiguamiento del 5% y a este modo el generador que más participa es Daule Peripa.

Tabla 3.3. Análisis Modal Demanda Máxima sin PSS´s

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%

Mode 00127	-0.242811522	7.286695239	1.159713566	3.330409979
Mode 00184	-0.439461572	8.631552171	1.373754195	5.084752867
Mode 00196	-0.477441319	7.206211906	1.146904246	6.610919469
Mode 00189	-0.459852973	6.437694049	1.02459083	7.124977596
Mode 00108	-0.209720355	2.878981998	0.458204216	7.265280768

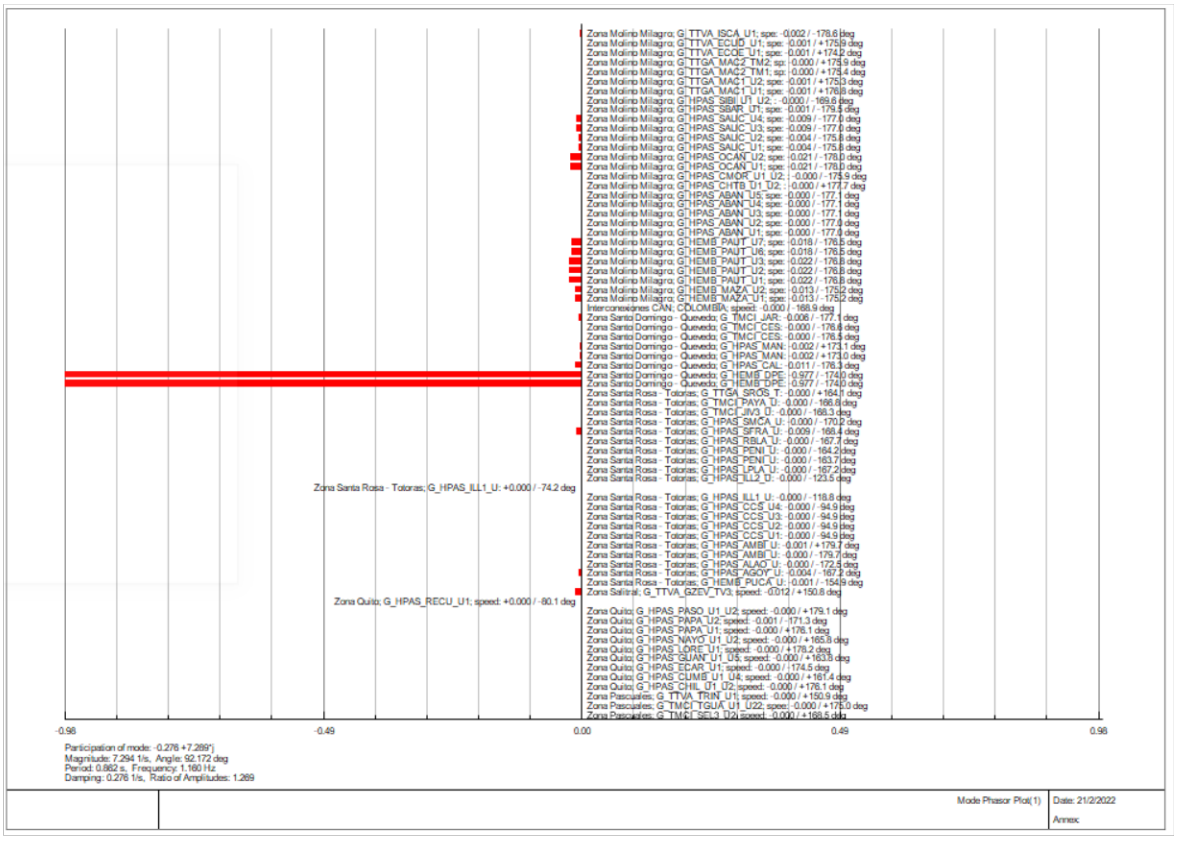


Figura 3.4. Participación modo local 127

Como se puede ver en los tres puntos de operación es necesario utilizar los algoritmos de optimización y se obtuvo los siguientes resultados.

Para la ubicación de los estabilizadores de potencia se consultó de [9] y se tiene que se deben instalar en los Generadores Paute C, Sopladora 1, Machala, Agoyán, San Francisco, Coca Codo Sinclair, Egzevallos y Daule Peripa.

3.1.2 Resultados Demanda Mínima SNI

Tabla 3.4. Resultados Demanda Mínima MVMO

Name	Kw	Tw	T2	T1	T4	T3
PSS_08_Paute_1_10	14.466	10	0.013107	0.185775	0.028871	0.185745
PSS_13_Sopl_1_3	17.17204	10	0.011146	0.099549	0.019138	0.124104

PSS_17_MACH2	34.77177	10	0.011273	0.057188	0.011813	0.080875
PSS_19_MACH_II	39.98752	10	0.019821	0.191928	0.006552	0.091714
PSS_29_TV3	6.58655	10	0.04761	0.099284	0.026765	0.198682
PSS_30_Ago1	25.65432	10	0.082415	0.198137	0.002443	0.014984
PSS_32_CCS_1_4	35.24528	10	0.004772	0.065424	0.001386	0.012719
PSS_37_SF2	42.76362	10	0.002135	0.016217	0.030586	0.141903
PSS_42_DPR1	2.618495	10	0.027394	0.159653	0.025097	0.127378

Tabla 3.5. Análisis Modal Demanda Mínima MVMO

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00139	-0.420134215	7.154252195	1.138634601	5.101410586
Mode 00158	-0.565689042	7.396561042	1.177199252	7.625731058
Mode 00166	-0.636532636	8.085000721	1.28676783	7.84871928
Mode 00180	-0.685028588	8.335462217	1.326630015	8.190630552
Mode 00104	-0.275443144	3.136945467	0.499260377	8.746961824

Tabla 3.1. Resultados Demanda Mínima WOA

Name	Kw	Tw	T2	T1	T4	T3
PSS_08_Paute_1_10	7.622597	10	0.015459	0.101031	0.005415	0.058129
PSS_13_Sopl_1_3	0	10	0.005096	0.02077	0.005227	0.057244
PSS_17_MACH2	82.18121	10	0.009582	0.05841	0.14681	0.182302
PSS_19_MACH_II	77.01137	10	0.035633	0.167785	0.007303	0.104135
PSS_29_TV3	0	10	0.006339	0.070248	0.002051	0.01
PSS_30_Ago1	0	10	0.01041	0.140907	0.000936	0.011657
PSS_32_CCS_1_4	35.24528	10	0.004772	0.065424	0.001386	0.012719
PSS_37_SF2	42.76362	10	0.002135	0.016217	0.030586	0.141903
PSS_42_DPR1	2.618495	10	0.027394	0.159653	0.025097	0.127378

Tabla 3.6. Análisis Modal Demanda Mínima WOA

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00138	-0.417906287	7.15475677	1.138714907	5.098018983
Mode 00158	-0.565632049	7.396548245	1.177197215	7.624980357
Mode 00166	-0.636393864	8.084958642	1.286761133	7.847059284
Mode 00180	-0.686029234	8.334535057	1.326482453	8.203420803
Mode 00103	-0.270658658	3.134428591	0.498859804	8.60300988

Ya en el análisis modal se puede notar que todos los modos cumplieron con el amortiguamiento por que se puede ver que los algoritmos cumplen con su trabajo.

Respuesta en el dominio del tiempo demanda mínima SNI

Para comprobar la respuesta en el dominio del tiempo se realiza la siguiente falla:

Cortocircuito trifásico línea de transmisión 1 Daule Peripa – Quevedo a 0.01 [s] y apertura de las dos líneas a 0.1[s]

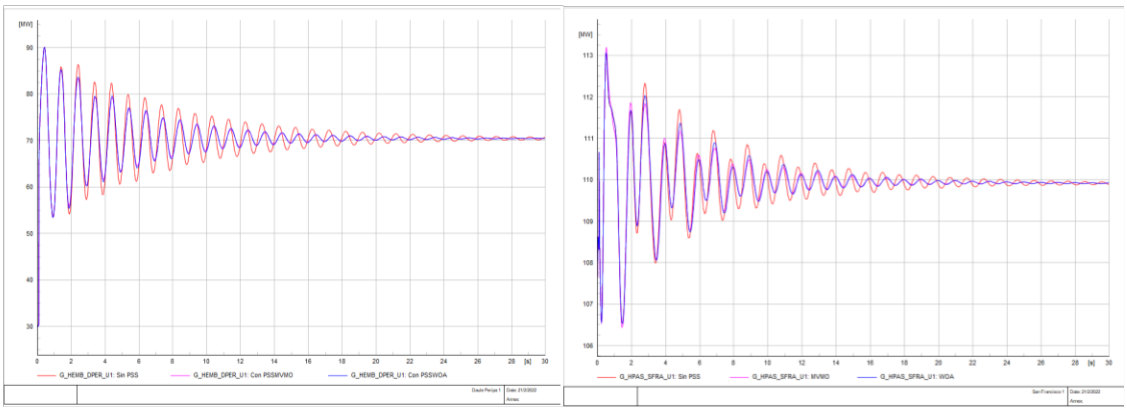


Figura 3.5. Potencia Activa Daule Peripa 1 Demanda Mínima SIN PSS´s, MVMO y WOA, y Potencia Activa San Francisco 1 Demanda Mínima SIN PSS´s, MVMO y WOA

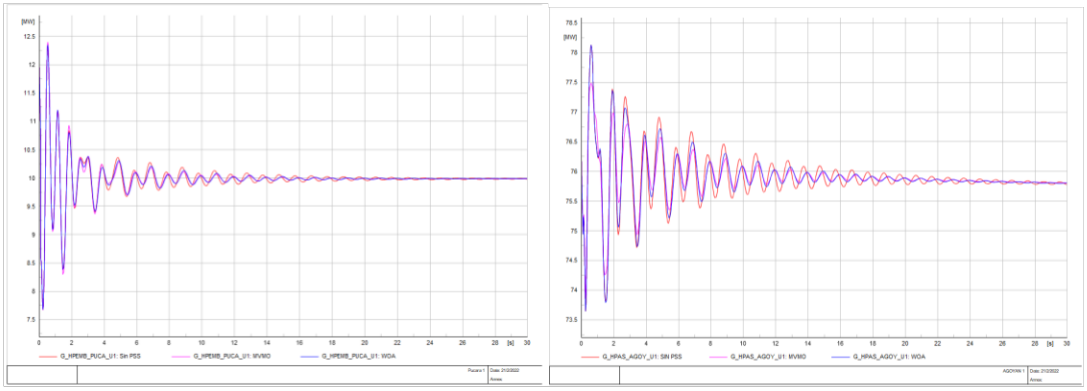


Figura 3.6. Potencia Activa Pucará 1 Demanda Mínima SIN PSS´s, MVMO y WOA, y Potencia Activa Agoyán 1 Demanda Mínima SIN PSS´s, MVMO y WOA

Descomposición de la señal Demanda Mínima SNI

Se procede a realizar la descomposición de la señal de la Potencia Activa Daule Peripa 1 ya que el modo que tiene menor amortiguamiento es local y el generador que más participa es Daule Peripa.

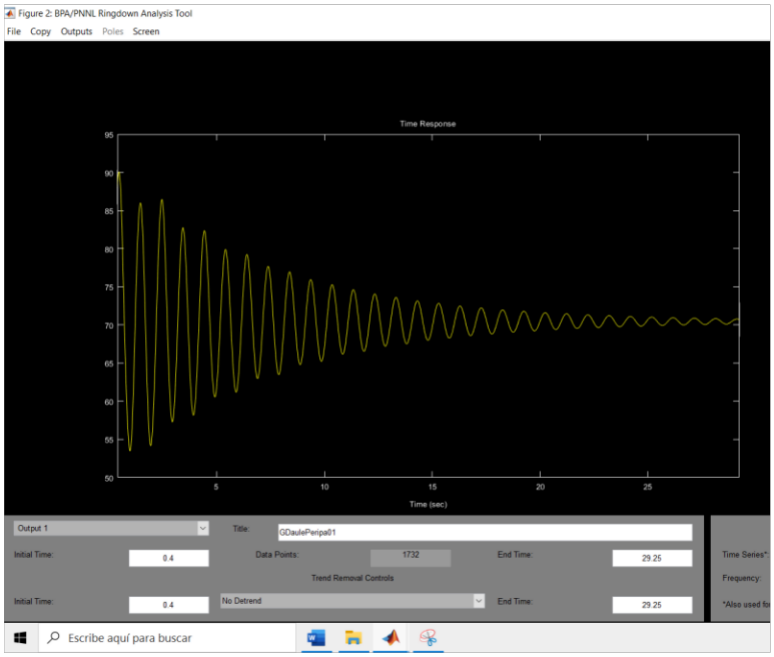


Figura 3.7. Potencia Activa Daule Peripa 1 Sin PSS´s

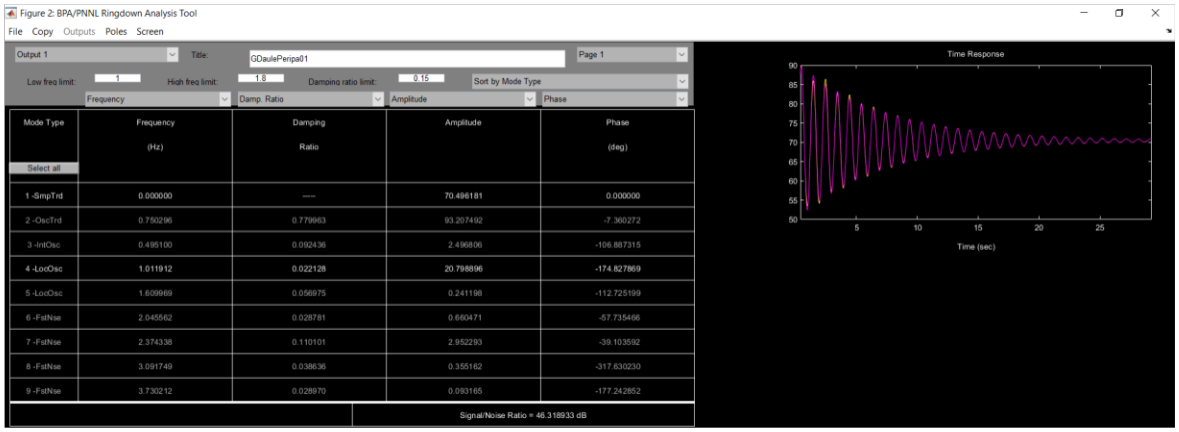


Figura 3.8. Descomposición de la Potencia Activa Daule Peripa 1

Tabla 3.7. Modo local vs Modo oscilatorio de la señal

SEÑAL Y MODO DEMANDA MINIMA		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL	SIN PSS	1.163293505	4.366523959
	MVMO	1.138634601	5.101410586
	WOA	1.138714907	5.098018983
POTENCIA GENERADOR DAULE PERIPA U1	SIN PSS	1.011911903	2.212840625
	MVMO	0.995593803	3.330445004
	WOA	0.996132944	3.228330275

De estos resultados se puede ver que se escoge al algoritmo MVMO porque tiene un amortiguamiento del 3.33% frente a un 3.22% del WOA en el dominio del tiempo.

3.1.3 Resultados Demanda Media SNI

Tabla 3.8. Resultados Demanda Media MVMO

Name	Kw	Tw	T2	T1	T4	T3
PSS_08_Paute_1_10	71.29004	10	0.01647	0.159439	0.018596	0.193551
PSS_13_Sopl_1_3	46.01829	10	0.013305	0.184661	0.014665	0.167803
PSS_17_MACH2	66.02328	10	0.0181	0.095269	0.00952	0.132011
PSS_19_MACH_II	54.70112	10	0.017214	0.159661	0.013584	0.199746
PSS_29_TV3	57.04942	10	0.015923	0.153508	0.006402	0.092616
PSS_30_Ago1	99.87982	10	0.00921	0.130456	0.014139	0.19357
PSS_32_CCS_1_4	37.42671	10	0.004772	0.065424	0.001386	0.012719
PSS_37_SF2	55.01467	10	0.002135	0.016217	0.030586	0.141903
PSS_42_DPR1	67.24974	10	0.027394	0.159653	0.025097	0.127378

Tabla 3.9. Análisis Modal Demanda Media MVMO

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00291	-1.126281739	19.7731537	1.120051923	5.096796872
Mode 00235	-0.780064813	10.82901902	1.723491905	7.184850062
Mode 00184	-0.510536719	7.037493783	3.146995153	7.235510038
Mode 00195	-0.571973549	6.56128073	1.044260261	8.684471472
Mode 00129	-0.266661924	2.909841623	0.463115678	9.12589856

Tabla 3.10. Resultados Demanda Media WOA

Name	Kw	Tw	T2	T1	T4	T3
PSS_08_Paute_1_10	100	10	0.013625	0.166231	0.013333	0.2
PSS_13_Sopl_1_3	48.61568	10	0.006775	0.097354	0.104169	0.16062
PSS_17_MACH2	100	10	0.010549	0.142725	0.010564	0.129692
PSS_19_MACH_II	74.03861	10	0.039204	0.2	0.058772	0.2
PSS_29_TV3	75.02465	10	0.01756	0.193899	0.018541	0.178984
PSS_30_Ago1	88.32912	10	0.017471	0.2	0.023227	0.2
PSS_32_CCS_1_4	37.42671	10	0.004772	0.065424	0.001386	0.012719
PSS_37_SF2	55.01467	10	0.002135	0.016217	0.030586	0.141903
PSS_42_DPR1	67.24974	10	0.027394	0.159653	0.025097	0.127378

Tabla 3.11. Análisis Modal Demanda Media WOA

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00293	-1.115531411	19.77078677	1.120022548	5.097361794

Mode 00183	-0.510252703	7.037309215	3.146618443	7.231694588
Mode 00189	-0.524631453	6.550844497	1.042599283	7.983048108
Mode 00229	-0.776844974	10.82457261	1.722784238	7.158269769
Mode 00119	-0.231870484	2.902577387	0.461959539	7.963066391

Respuesta en el Dominio del Tiempo Demanda Media

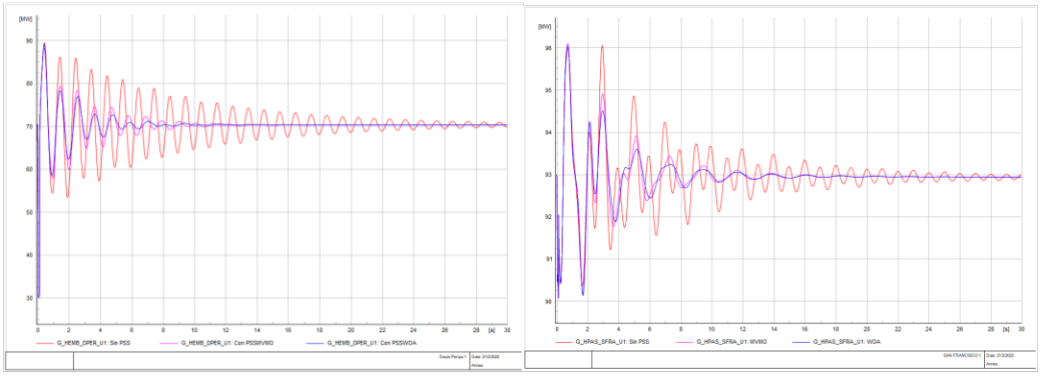


Figura 3.9. Potencia Activa Daule Peripa 1 Demanda Media SIN PSS´s, MVMO y WOA, y Potencia Activa San Francisco 1 Demanda Media SIN PSS´s, MVMO y WOA

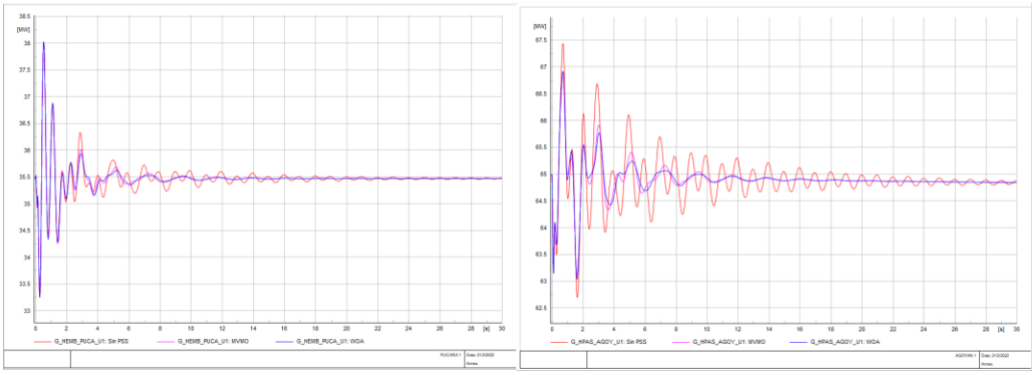


Figura 3.10. Potencia Activa Pucará 1 Demanda Media SIN PSS´s, MVMO y WOA, y Potencia Activa Agoyán 1 Demanda Media SIN PSS´s, MVMO y WOA

Descomposición de la señal demanda media

Tabla 3.12. Modo local vs Modo oscilatorio de la señal

SEÑAL Y MODO DEMANDA MEDIA		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL	SIN PSS	1.160099546	3.789775592
	MVMO	1.120051923	5.096796872
	WOA	1.120022548	5.097361794
POTENCIA GENERADOR DAULE PERIPA U1	SIN PSS	0.59869217	1.933130367
	MVMO	0.552602532	5.35138844
	WOA	0.534423856	8.738746637

De estos resultados se puede ver que se escoge al algoritmo WOA porque tiene un amortiguamiento del 8.738% frente a un 5.35% del MVMO en el dominio del tiempo.

3.1.4 Resultados demanda máxima SNI

Tabla 3.13. Resultados Demanda Máxima MVMO

Name	Kw	Tw	T2	T1	T4	T3
PSS_08_Paute_1_10	67.65134	10	0.013398	0.099732	0.046547	0.186448
PSS_13_Sopl_1_3	84.32145	10	0.011587	0.16833	0.019089	0.196603
PSS_17_MACH2	35.02697	10	0.011611	0.114592	0.01434	0.180191
PSS_19_MACH_II	96.6186	10	0.001381	0.017675	0.009035	0.134158
PSS_29_TV3	99.54878	10	0.007977	0.048964	0.039985	0.165533
PSS_30_Ago1	52.3015	10	0.010611	0.136672	0.010858	0.157519
PSS_32_CCS_1_4	35.24528	10	0.004772	0.065424	0.001386	0.012719
PSS_37_SF2	42.76362	10	0.002135	0.016217	0.030586	0.141903
PSS_42_DPR1	20.61849	10	0.027394	0.159653	0.025097	0.127378

Tabla 3.14. Análisis Modal Demanda Máxima MVMO

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00320	-1.075101104	18.23950055	1.14679065	5.098143144
Mode 00202	-0.477503724	7.205498164	1.55274768	6.612431857
Mode 00245	-0.672297006	9.756201408	2.902906672	6.874667949
Mode 00210	-0.500326943	6.502936058	1.034974418	7.671190756
Mode 00144	-0.281488459	2.894381288	0.460655089	9.679672783

Tabla 3.15. Resultados Demanda Máxima WOA

Name	Kw	Tw	T2	T1	T4	T3
PSS_08_Paute_1_10	100	10	0.00823	0.123451	0.017958	0.164351
PSS_13_Sopl_1_3	100	10	0.011098	0.166472	0.014497	0.182675
PSS_17_MACH2	57.85558	10	0.005805	0.068522	0.054461	0.2
PSS_19_MACH_II	100	10	0.012031	0.17994	0.015436	0.2
PSS_29_TV3	68.18807	10	0.034067	0.2	0.011773	0.145756
PSS_30_Ago1	100	10	0.020509	0.2	0.012243	0.183646
PSS_32_CCS_1_4	35.24528	10	0.004772	0.065424	0.001386	0.012719
PSS_37_SF2	42.76362	10	0.002135	0.016217	0.030586	0.141903
PSS_42_DPR1	20.61849	10	0.027394	0.159653	0.025097	0.127378

Tabla 3.16. Análisis Modal Demanda Máxima WOA

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00320	-1.077542054	18.24006537	1.146794724	5.097274331
Mode 00202	-0.477505387	7.205523759	1.552466828	6.612431389
Mode 00245	-0.673271695	9.754436765	2.902996566	6.885827121
Mode 00213	-0.514435984	6.507952446	1.035772801	7.880147583
Mode 00144	-0.283325921	2.887362841	0.459538069	9.76571605

Respuesta en el dominio del tiempo demanda máxima

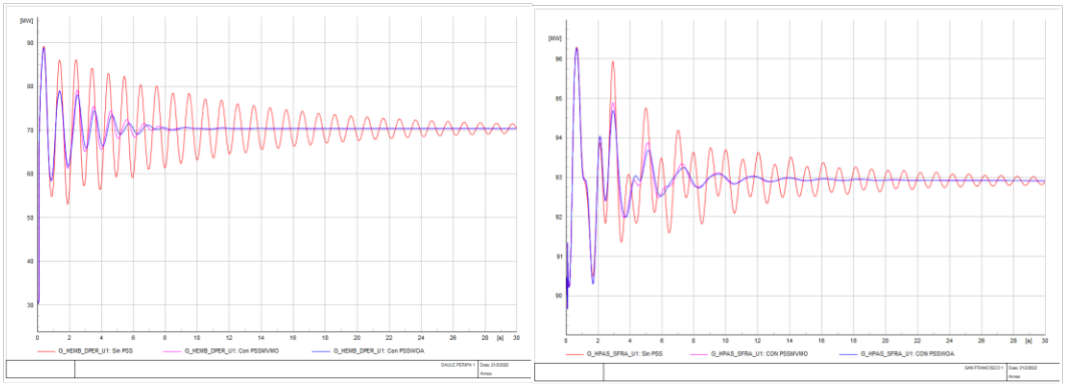


Figura 3.11. Potencia Activa Daule Peripa 1 Demanda Máxima SIN PSS´s, MVMO y WOA, y Potencia Activa San Francisco 1 Demanda Máxima SIN PSS´s, MVMO y WOA

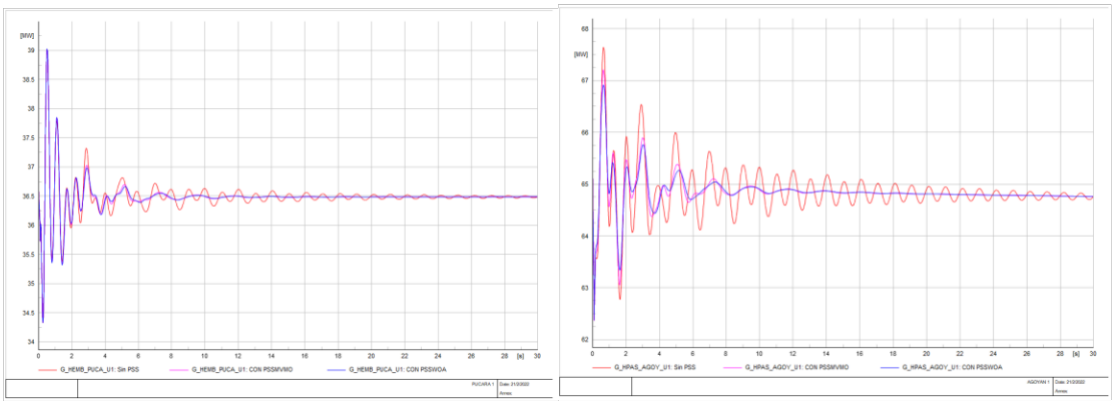


Figura 3.12. Potencia Activa Pucará 1 Demanda Máxima SIN PSS´s, MVMO y WOA, y Potencia Activa Agoyán 1 Demanda Máxima SIN PSS´s, MVMO y WOA

Descomposición de la señal demanda máxima

Tabla 3.17. Modo local vs Modo oscilatorio de la señal

SEÑAL Y MODO DEMANDA MAXIMA		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL	SIN PSS	1.159713566	3.330409979
	MVMO	1.14679065	5.098143144
	WOA	1.146794724	5.097274331
POTENCIA GENERADOR DAULE PERIPA U1	SIN PSS	0.59282858	1.531918167
	MVMO	0.562959178	5.303389741
	WOA	0.541993518	7.770603451

De estos resultados se puede ver que se escoge al algoritmo WOA porque tiene un amortiguamiento del 7.77% frente a un 5.3% del WOA en el dominio del tiempo.

3.1.4 Ajuste demanda mínima a demanda máxima

Para el ajuste se pasará de Demanda Mínima a Máxima.

Con los datos de MVMO en Demanda Mínima aplicados a Demanda Máxima se obtuvo después de realizar el análisis modal.

Tabla 3.18. Análisis Modal Resultados MVMO Demanda Mínima en Demanda Máxima

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00150	-0.324615339	7.149065228	1.13780907	4.535994637
Mode 00322	-1.07081806	18.23679422	2.902475946	5.861649016
Mode 00206	-0.477865814	7.20645696	1.146943247	6.616547515
Mode 00203	-0.466710688	6.450827133	1.026681025	7.216035767
Mode 00128	-0.219209414	2.885759924	0.459282957	7.574423813

Después de Realizar el Ajuste con el Método del Residuo Estimado se obtuvo que para el PSS Daule Peripa 1 tiene que cambiar el valor de 2.618495 a 21.2147 y se obtuvo el siguiente análisis modal.

Tabla 3.19. Análisis Modal Resultados MVMO Demanda Mínima Ajustado en Demanda Máxima

Name	Real part	Imaginary part	Damped Frequency	Damping Ratio
	1/s	rad/s	Hz	%
Mode 00176	-0.378505146	7.016281004	1.116675804	5.386836327
Mode 00322	-1.070821983	18.23679479	2.902476037	5.861670234
Mode 00206	-0.478033403	7.206058684	1.14687986	6.619222018
Mode 00202	-0.466314154	6.449684812	1.026499219	7.211206981

Mode 00130	-0.222249276	2.883445667	0.458914631	7.68497282

Con los datos de WOA en Demanda Mínima aplicados a Demanda Máxima se obtuvo después de realizar el análisis modal.

Tabla 3.20. Análisis Modal Resultados WOA Demanda Mínima en Demanda Máxima

Name	**Real part**	**Imaginary part**	**Damped Frequency**	**Damping Ratio**
	1/s	**rad/s**	**Hz**	**%**
Mode 00150	-0.322300845	7.149329116	1.137851069	4.503553115
Mode 00324	-1.070084826	18.23621243	2.902383351	5.85783531
Mode 00206	-0.477872104	7.206429892	1.146938939	6.616658978
Mode 00199	-0.46083748	6.441589967	1.025210885	7.135857477
Mode 00120	-0.212158584	2.881879177	0.458665316	7.341945544

Después de Realizar el Ajuste con el Método del Residuo Estimado se obtuvo que para el PSS Daule Peripa 1 tiene que cambiar el valor de 2.618495 a 20.95483 y se obtuvo el siguiente análisis modal.

Tabla 3.21. Análisis Modal Resultados WOA Demanda Mínima Ajustado en Demanda Máxima

Name	**Real part**	**Imaginary part**	**Damped Frequency**	**Damping Ratio**
	1/s	**rad/s**	**Hz**	**%**
Mode 00172	-0.374945492	7.018052292	1.116957713	5.334977732
Mode 00324	-1.070088701	18.23621284	2.902383417	5.857856319
Mode 00206	-0.478033007	7.206047526	1.146878084	6.619226765
Mode 00199	-0.460556369	6.440613999	1.025055555	7.132601918
Mode 00126	-0.215162457	2.879637899	0.458308606	7.451087808

Respuesta en el dominio del tiempo ajuste demanda mínima a demanda máxima

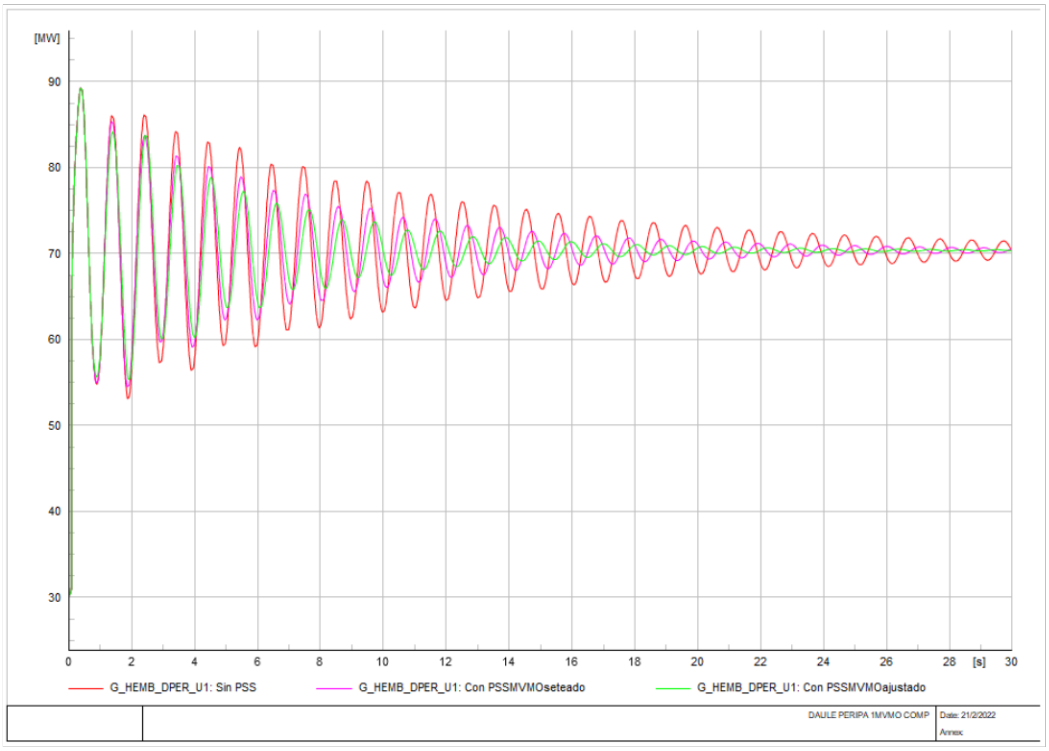


Figura 3.13. Potencia Activa Daule Peripa 1 Demanda Máxima SIN PSS´s, Demanda Mínima MVMO seteado y Demanda Mínima MVMO Ajustado

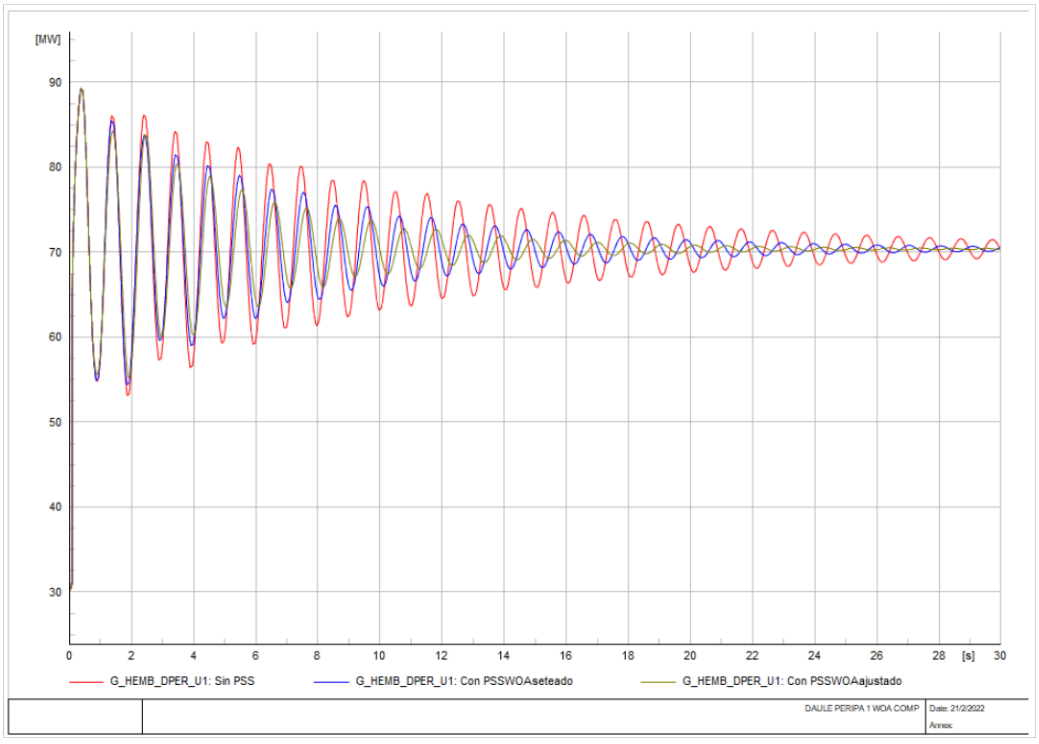


Figura 3.14. Potencia Activa Daule Peripa 1 Demanda Máxima SIN PSS´s, Demanda Mínima WOA seteado y Demanda Mínima WOA Ajustado

De las gráficas de Demanda Mínima seteado vs ajustado se puede ver claramente que la señal se vuelve un poco más amortiguada, lo que ayuda bastante a la estabilidad transitoria.

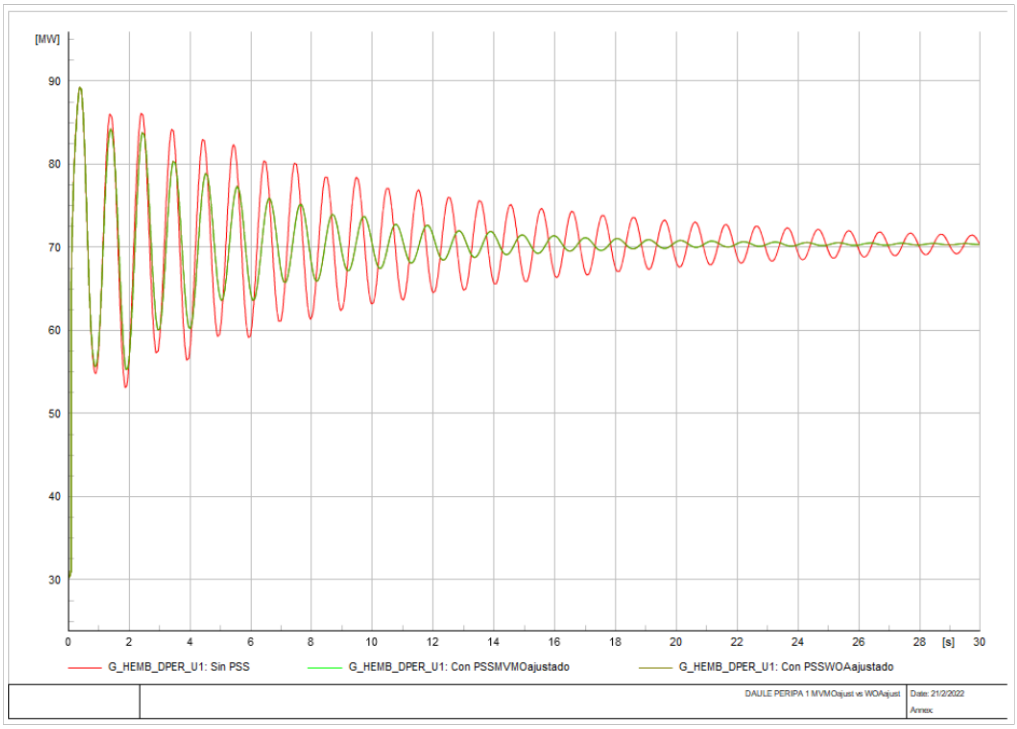


Figura 3.15. Potencia Activa Daule Peripa 1 Demanda Máxima SIN PSS´s, Demanda Mínima MVMO ajustado y Demanda Mínima WOA Ajustado

Descomposición de la señal MVMO ajustado vs WOA ajustado

Tabla 3.22. Modo local vs Modo oscilatorio de la señal

SEÑAL Y MODO DEMANDA MINIMA A MAXIMA		FRECUENCIA [Hz]	AMORTIGUAMIENTO [%]
MODO LOCAL	MVMO	1.116675804	5.386836327
	WOA	1.116957713	5.334977732
POTENCIA GENERADOR DAULE PERIPA U1	MVMO	0.577272935	2.990703118
	WOA	0.577549822	2.985863469

De estos resultados se puede ver que se escoge al algoritmo MVMO porque tiene un amortiguamiento del 2.99% frente a un 2.98% del WOA en el dominio del tiempo.

3.2 Conclusiones

- Los algoritmos de optimización MVMO y WOA mejoraron notablemente la estabilidad de pequeña señal para los Sistemas New England y Sistema Nacional Interconectado, haciendo que el modo oscilatorio del sistema con menor amortiguamiento sea mayor al 5%.
- El método del Residuo resultó ser una forma bastante efectiva y sencilla para poder mover un modo a través del plano complejo hasta alcanzar un amortiguamiento establecido.
- Se concluye que para los Sistemas New England y SNI, si se lleva la sintonización de estabilizadores de potencia de un punto de operación mayor a otro punto de operación menor no se tiene que realizar un ajuste ya que por lo general no se presentan modos con amortiguamiento menor a 5%. En el caso contrario de un punto de operación menor se lo lleva a uno mayor, sí existen modos que no cumplen con el amortiguamiento del 5%, y en este caso se tiene que realizar el ajuste con el método del Residuo.
- Como se pudo ver en las gráficas de Potencia Activa, después de realizar el ajuste con el Método del Residuo se mejoró la estabilidad transitoria de la señal de Potencia Activa haciéndola un poco más amortiguada.
- Las señales que se deben escoger para realizar la descomposición de la señal con el método Prony dependen de conocer si el modo con menor amortiguamiento del sistema es entre área o es un local. Si el modo es entre área se obtiene la observabilidad de este modo y se descompone la potencia activa de la línea de interconexión de las dos grandes áreas que se forman y si el modo es local se obtiene el factor participación de éste y se descompondrá la potencia activa del generador que más participa.
- La metodología propuesta para la sintonización de estabilizadores de potencia con dos técnicas heurísticas, la corrección con el método del residuo y la validación con la descomposición de la señal resultó ser muy efectiva ya que se mejoró tanto la estabilidad de pequeña señal como la estabilidad transitoria para los Sistemas New England como SNI.
- Los dos algoritmos de optimización son muy buenos para realizar la sintonización de estabilizadores de potencia, la gran diferencia entre estos dos algoritmos es que

para el WOA se tiene que definir solo dos parámetros, en cambio para el MVMO se tiene que definir mínimo seis parámetros, por lo que por simpleza gana el algoritmo de ballenas.

3.3 Recomendaciones

- Se recomienda realizar un estudio exhaustivo con respecto a la ubicación de los Estabilizadores de Potencia para el Sistema de Prueba al cual se quiera implementar esta Metodología Híbrida ya que en este trabajo no se lo realizó, es muy importante este punto porque podrá significar un menor esfuerzo para los algoritmos de optimización y también un menor tiempo para que converjan en una solución.

4 REFERENCIAS BIBLIOGRÁFICAS

1. Bibliografía

[1] M. A. Pai y A. Stankovic, Robust Control in Power Systems, New York: Springer, 2005.

[2] F. DUSSAUD, An application of modal analysis in, STOCKHOLM, SWEDEN: KTH ROYAL INSTITUTE OF TECHNOLOGY, 2015.

[3] G. Arguello y H. Flores, ESTUDIO DE ESTABILIDAD DE PEQUEÑA SEÑAL EN EL SISTEMA NACIONAL INTERCONECTADO APLICANDO EL MÉTODO DE ANÁLSIS MODAL, Quito, Ecuador: Escuela Politécnica Nacional, 2005.

[4] E. F. Guanichanga Collaguazo y M. G. Ocaña Frutos , Ubicación de un control de modos deslizantes y PSS´s con el objetivo de amortiguar oscilaciones electromecánicas utilizando el método del residuo y fuzzy, Quito: Escuela Politécnica Nacional, 2018.

[5] J. A. Oscullo Lala, Sintonización adaptativa de estabilizadores de sistemas de potencia en tiempo real utilizando tecnología sincrofasorial, Quito : Escuela Politécnica Nacional, 2021.

[6] P. Kundur, Power System Stability and Control, New York : McGraw-Hill, Inc, 1994.

[7] C. F. Gallardo Quingatuña, Estabilidad y Amortiguamiento de Oscilaciones en Sistemas Eléctricos con Alta Penetración Eólica, Leganés: Universidad Carlos III de Madrid, 2009.

[8] J. Cepeda y J. L. Rueda, Power Factory Applications for power system analysis, 2014.

[9] S. Mirjalili y A. Lewis, Advances in Engineering Software, Brisbane, Australia: Elsevier, 2016.

5 ANEXOS

ANEXO I. Código DPL MVMO (Mean Variance Mapping Optimization)

```
!!! MVMO internal variable declaration
int nparam,ele,dddd,Ddddd,asymetry,Fx_best,no_in,no_inin,ele1,numparam;
int scaling1,scaling2,scaling,fitsize,xtfeas,isbetter,ixbetter;
int n_sim,nnn,nnn1,nnp,changed,i_position,njump,Time_o,Time_f,Dpl_time;
double ran_n,x_n,x_den,mslope,Nrandomly,ffx,xtobjective,xtfitness;
double sumfit,vartemp,weight,var1,var2;
double vartemp1,vartemp2,vartemp3,vartemp4,vartemp5,H,H0,H1;
string folder,file1,file2,parametros;
int contador;
set OldMons;
object mon;
object GrB,ViPg,Plot;

!!! SPECIFICATION OF FOLDER TO SAVE RESULTS
folder=sprintf('%s','C:\Users\LENOVO\Desktop\Archivos OSEP\Resultados MVMO New
England');
!!!----------------------------------------------------------------

!*********************************************************************
!!! CREATION OF TWO RESULT FILES
file1=sprintf('%s%s',folder,'\results.txt'); ! File for saving convergence results
file2=sprintf('%s%s',folder,'\parameters.txt'); ! File for saving optimized variables
!*************************************************************

Time_o=GetTime(4); ! Defining the initial processor time
EchoOff(); ! Freezing the user-interface

!************************************
! Saving convergence results.

OldMons = Convergence.GetContents();
mon=OldMons.FirstFilt('Xvar.IntMon');

if(mon = NULL){
mon = Convergence.CreateObject('IntMon','Xvar');
}

if(mon){
mon:obj_id=this;
mon.AddVar('b:ffx_eval');
mon.AddVar('b:ffx_conv');
}

!************************************

! Parameter definition
contador=1;
```

```
sumfit=0;
dddd=25;
asymetry=2.5;
Fx_best=1e50;
no_in=0; !Initial number of variables selected for mutation
no_inin=0;
Nrandomly=Nrandomly_in;

! Memory space definition
nparam=Paralim.NCol(); ! returns the no of columns in matrix
Ddddd=0.0505/nparam+1; ! Initial value of alternative shape
bests.Resize(Nsave,nparam); ! Nsave rows (no of population) and n columns
Mobjective.Resize(Nsave); ! vector of Nsave rows (no of population)
Feas.Resize(Nsave); ! Feas resize (no of population)
! for Nsave population size, set Ftness vector to 1e10.
Fitness.Resize(Nsave);
for(ele=1;ele<=Nsave;ele+=1){
Fitness.Set(ele,1e10);
}
!*************************************

considered.Resize(nparam); ! Considered vector with n values from nparam
variance.Resize(nparam); ! variance vector with n values from nparam
Meann.Resize(nparam); ! mean vector with n values from nparam
for(ele=1;ele<=nparam;ele+=1){ ! for all n values
considered.Set(ele,1); !set value at index i (i,v) to V
variance.Set(ele,1);
}
! set considered and variance vector to 1 ( n vector values)
!*********************************************

!*************************************

vsel.Resize(Nrandomly); ! Vsel vector Number of variable changed randomly
vv.Resize(Nrandomly); ! vv vector with Number of variable changed randomly
for(ele=1;ele<=Nrandomly;ele+=1){
vsel.Set(ele,ele);
vv.Set(ele,ele);
}
! set vsel and vv vector to Number of variable changed randomly
!*************************************************************

!*****************************************************

v_isfeas.Resize(Nrestric); ! vector v_isfeas equal to number of restrictions
for(ele=1;ele<=Nrestric;ele+=1){
v_isfeas.Set(ele,ele);
}
!*************************************************
!*********************************************
if (Nrandomly > nparam) { ! if # var. changed randomly > # parameters to be optimized
Nrandomly=nparam; !variable changed randomly = Number of parameters to be optimzed
}
!*********************************************
```

```
!*************************************************
x_norm.Resize(nparam); ! make vector x_norm equal to nparam
x_denorm.Resize(nparam); ! make vector x_denorm equal to nparam
idcontrol.Resize(nparam); ! make vector idcontrol equal to nparam
mean_temp.Resize(nparam); ! make vector mean_temp equal to nparam
var_temp.Resize(nparam); ! make vector var_temp equal to nparam
V_temp.Resize(nparam); ! make vector V_temp equal to nparam

!*************************************************

!*************************************************************
! Initialization
for(ele=1;ele<=nparam;ele+=1){ ! for every parameters to be optimized (n)
ran_n= fRand(0); !generate a random number (uniform distribution)
!0= uniform , 1= normal , 2 = weibull.
x_norm.Set(ele,ran_n); ! set for each parameters to be optimized, the x-norm values to
random numbers (n) (normalze all the values between 0 to 1)
x_norm_best.Set(ele,ran_n); ! set for each parameters to be optimized, the x-norm_best
values to random numbers (n)
x_denorm_best.Set(ele,ran_n); ! set for each parameters to be optimized, the x-
denorm_best are the best de-normalized parameters (n)
Meann.Set(ele,ran_n); ! set for each parameters to be optimized, the Meann values to
random numbers (n)
}

!*****************************************************************************************

!*****************************************************************
!Main loop: Optimization, Objective function and Restrictions

for(n_sim=1;n_sim<=Maxeval;n_sim+=1){ ! Running optimization for each iteration to
Maxeval

for(ele=1;ele<=nparam;ele+=1){ ! for each parameters to be optimized,
scaling1=Paralim.Get(2,ele); ! get the max parameters values and store them in scaling 1
(n)
! for matrix !Get (Row in matrix, col in matrix) = returns the row value from the matrix
scaling2=Paralim.Get(1,ele); ! get the min parameters values and store them in scaling 2
(n)
scaling=scaling1-scaling2; ! get the diff between max and min limits of the parameters
values
x_n=x_norm.Get(ele); !for vector ! returns the element stored at given position in container
! store all the n random values from x_norm in x_n
!Get(position in the container). return value is the element stored at that position.
x_den=x_n*scaling+scaling2; ! denormalize all the values (n)
x_denorm.Set(ele,x_den); !set value at index i (i,v) to V (write all the values in x_denorm
vector)
}

!*************************************************
!-----------------------------------------------
!!!! Llamar a la funcion objetivo
Obj_function.Execute(ffx);
```

```
no_in+=1; ! for each iteration, increase no_in by 1
changed=0;
ixbetter=0;
vartemp=x_norm.Size(); ! set vartemp equal to n values (randomly generated) from
x_norm
if(ixbetter=1){ ! if true
i_position=ele; ! change i_position with each iteration
changed=1; ! make changed equal to 1 and
no_inin+=1; ! incease no_inin by 1
}
}
}
!**************************************************************************
if(changed=1){
if(Nsave>=i_position+1){
for(ele=Nsave;ele>=i_position+1;ele-=1){ ! run for loop from Nsave to 1
vartemp=Mobjective.Get(ele-1); ! set vartemp equal to value of Mobjective
Mobjective.Set(ele,vartemp); ! set Mobjective value equal to vartemp
vartemp=x_norm.Size(); ! make vartemp equal to x_norm (n)
}
Mobjective.Set(i_position,xtobjective); ! set Mobjective with values of objective function
Fitness.Set(i_position,xtfitness); ! same for fitness with xfitness
Feas.Set(i_position,xtfeas); ! same for Feas with feas
!*****************************************************************************
!**************************************************************************
!Actualize x_norm_best
vartemp=x_norm.Size(); ! make vartemp equal to x_norm size (n)
for (ele=1;ele<=vartemp;ele+=1){
vartemp1=bests.Get(1,ele); ! set vartemp1 with first row and n columns of bests
x_norm_best.Set(ele,vartemp1); ! set x_norm_best with vartemp1
}
!*****************************************************************************
!*****************************************************************************
!! MEAN
for(ele=1;ele<=nparam;ele+=1){
vartemp=0;
for(ele1=1;ele1<=Nsave;ele1+=1){
vartemp1=bests.Get(ele1,ele);
vartemp+=vartemp1; ! add each variable in best
}
vartemp2=weight*vartemp; ! mean formula
mean_temp.Set(ele,vartemp2); ! mean_temp is equal to mean value
}
for(ele=1;ele<=nparam;ele+=1){
vartemp=idcontrol.Get(ele);
if(vartemp=1){
vartemp1=mean_temp.Get(ele);
Meann.Set(ele,vartemp1);
}
}
!! Variance
for(ele=1;ele<=nparam;ele+=1){
```

```
vartemp=0;
for(ele1=1;ele1<=Nsave;ele1+=1){
vartemp1=bests.Get(ele1,ele); ! for each variable in best
vartemp2=mean_temp.Get(ele);
vartemp3=weight*sqr(vartemp1-vartemp2); !variance formula
vartemp+=vartemp3;
}
var_temp.Set(ele,vartemp);
if(vartemp>0){
V_temp.Set(ele,1);
}else{
V_temp.Set(ele,0);
}
}
for(ele=1;ele<=nparam;ele+=1){
vartemp=idcontrol.Get(ele);
vartemp2=V_temp.Get(ele);
vartemp3=min(vartemp,vartemp2);
if(vartemp3=1){
vartemp1=var_temp.Get(ele);
variance.Set(ele,vartemp1);
}
}
}
}

vartemp=Mobjective.Get(1); ! get first value of objective function value and print it
vartemp2=Feas.Get(1);
ffx_conv=vartemp;
ffx_eval=n_sim;
Convergence.WriteDraw(); ! Internally save convergence results (for plotting)
fopen(file1,'a',1); ! Open file for saving convergence results
fprintf(1,'%g %g %g',n_sim,vartemp,vartemp2); ! Save convergence results
fclose(1); ! Close file for saving convergence results
if({n_sim=contador*Nprint}.or.{n_sim=1}){ ! Print convergence in Output Window
parametros=sprintf('%g %g %g',n_sim,vartemp,vartemp2);
printf('%s',parametros);
if(n_sim>1){
contador+=1;
}
}
fopen(file2,'a',2); ! Open file for saving optimized variables
for(ele=1;ele<=nparam;ele+=1){
scaling1=Paralim.Get(2,ele);
scaling2=Paralim.Get(1,ele);
scaling=scaling1-scaling2;
x_n=bests.Get(1,ele);
x_den=x_n*scaling+scaling2; ! denormalize
x_denorm_best.Set(ele,x_den); ! Save denormalized parameters
if(ele=1){
parametros=sprintf('%g',x_den);
}else{
parametros=sprintf('%s %g',parametros,x_den);
}
```

```
}
fprintf(2,'%s',parametros); ! Save optimized variables results
fclose(2); ! Close file for saving optimized variables

!*********************************************************************
!*********************************************************************
!!! MUTATION 1
for(ele=1;ele<=nparam;ele+=1){
vartemp=x_norm_best.Get(ele);
x_norm.Set(ele,vartemp); ! Set x_norm with x_beast norm
considered.Set(ele,0);
}

for(ele=2;ele<=Nrandomly;ele+=1){
vartemp=irand.Get(ele-1);
vsel.Set(ele,vartemp);
}
for(ele=1;ele<=Nrandomly;ele+=1){
vartemp=vsel.Get(ele);
considered.Set(vartemp,1);
}
for(ele=1;ele<=Nrandomly;ele+=1){
vartemp=vsel.Get(ele);
vartemp+=njump;
vsel.Set(ele,vartemp);
}
!*****************************************************
!*****************************************************
!!!MUTATION 2
for(ele=1;ele<=nparam;ele+=1){
vartemp=considered.Get(ele);
if(vartemp){
vartemp1=fRand(0);
x_norm.Set(ele,vartemp1);
}
}
for(ele=1;ele<=nparam;ele+=1){
vartemp=considered.Get(ele);
if(vartemp){
vartemp3=variance.Get(ele);
vartemp1=-ln(vartemp3)*fs_factor; !!!ss1
vartemp2=vartemp1; !!!ss2
vartemp4=x_norm_best.Get(ele);
vartemp5=Meann.Get(ele);
if(vartemp4<vartemp5){
vartemp2=asymetry*vartemp2;
}else if(vartemp4>vartemp5){
vartemp1=asymetry*vartemp1;
}else{
if(vartemp2>=dddd){
dddd=dddd*Ddddd;
}else{
dddd=dddd/Ddddd;
}
```

```
vartemp1=dddd;
}
!! Function mutation
vartemp3=Meann.Get(ele);
vartemp4=x_norm.Get(ele);
H=vartemp3*(1-exp(-vartemp4*vartemp1))+(1-vartemp3)*exp((vartemp4-1)*vartemp2); !
functions for mapping
H0=(1-vartemp3)*exp(-vartemp2); ! functions for mapping
H1=vartemp3*exp(-vartemp1); ! functions for mapping
vartemp5=H+H1*vartemp4+H0*(vartemp4-1); ! functions for mapping
x_norm.Set(ele,vartemp5); ! update the x_norm vector with new values of functions
}
}
}

EchoOn(); ! Re-activating the user interface
Time_f=GetTime(4); ! Defining the final processor time
Dpl_time=Time_f-Time_o; ! Computing the elapsed time
printf('%s: %g','Processor Time',Dpl_time); ! Printing the elapsed time

!-----Load Best Param

numparam=x_denorm_best.Size();
for(ele=1;ele<=numparam/5;ele+=1){
if (ele=1){
PSS1:Kpss=x_denorm_best.Get(1);
var1=x_denorm_best.Get(2);
var2=x_denorm_best.Get(3);
PSS1:T1=var2;
PSS1:T2=var2/var1;
var1=x_denorm_best.Get(4);
var2=x_denorm_best.Get(5);
PSS1:T3=var2;
PSS1:T4=var2/var1;
} else if (ele=2){
PSS2:Kpss=x_denorm_best.Get(6);
var1=x_denorm_best.Get(7);
var2=x_denorm_best.Get(8);
PSS2:T1=var2;
PSS2:T2=var2/var1;
var1=x_denorm_best.Get(9);
var2=x_denorm_best.Get(10);
PSS2:T3=var2;
PSS2:T4=var2/var1;
} else if (ele=3){
PSS3:Kpss=x_denorm_best.Get(11);
var1=x_denorm_best.Get(12);
var2=x_denorm_best.Get(13);
PSS3:T1=var2;
PSS3:T2=var2/var1;
var1=x_denorm_best.Get(14);
var2=x_denorm_best.Get(15);
PSS3:T3=var2;
PSS3:T4=var2/var1;
```

```
} else if (ele=4){
PSS4:Kpss=x_denorm_best.Get(16);
var1=x_denorm_best.Get(17);
var2=x_denorm_best.Get(18);
PSS4:T1=var2;
PSS4:T2=var2/var1;
var1=x_denorm_best.Get(19);
var2=x_denorm_best.Get(20);
PSS4:T3=var2;
PSS4:T4=var2/var1;
}
}
```

ANEXO II. Código DPL WOA (Whale Optimization Algorithm)

```
!!! WOA internal variable declaration
int dim,leader_score,i,j,a,b,m,n,nparam;
int Time_o,Time_f,Dpl_time,ele,ele1,scaling,scaling1,scaling2,t,z,contador;
double ran_n,vartemp,vartemp1,vartemp2,vartemp3,vartemp4;
double ffx,fitness,a1,a2,r1,r2,A,C,l,p,var1,var2;
double rand_leader_index,D_X_rand,D_Leader,distance2Leader;
set OldMons;
object mon;
object GrB,ViPg,Plot;

Time_o=GetTime(4); ! Defining the initial processor time
EchoOff(); ! Freezing the user-interface

! Saving convergence results.

OldMons = Convergence.GetContents();
mon=OldMons.FirstFilt('Xvar.IntMon');

if(mon = NULL){
mon = Convergence.CreateObject('IntMon','Xvar');
}

if(mon){
mon:obj_id=this;
mon.AddVar('b:ffx_eval');
mon.AddVar('b:ffx_conv');
}


!searchagents_no = 20; !Numero de agentes
!max_iteration = 20; !Numero maximo de iteraciones
dim = Paralim.NCol(); ! returns the no of columns in matrix numero de variables

z=0;
leader_pos.Resize(dim);
leader_best.Resize(dim);
leader_score = 1e50;
positions.Resize(searchagents_no,dim);
v_vartemp.Resize(dim); ! make vector x_norm equal to nparam


for(ele=1;ele<=dim;ele+=1){ ! for each parameters to be optimized,
for(ele1=1;ele1<=searchagents_no;ele1+=1){
scaling1=Paralim.Get(2,ele); ! get the max parameters values and store them in scaling 1
(n)
! for matrix !Get (Row in matrix, col in matrix) = returns the row value from the matrix
scaling2=Paralim.Get(1,ele); ! get the min parameters values and store them in scaling 2
(n)
```

```
scaling=scaling1-scaling2; ! get the diff between max and min limits of the parameters
values
vartemp=fRand(0); !for vector ! returns the element stored at given position in container
! store all the n random values from x_norm in x_n
!Get(position in the container). return value is the element stored at that position.
vartemp1=vartemp*scaling+scaling2; ! denormalize all the values (n)
positions.Set(ele1,ele,vartemp1);
}
}

convergence_curve.Resize(max_iteration);
flag4ub.Resize(dim);
flag4lb.Resize(dim);

t=0; !contador de bucles
contador=0;
flag4ub.Resize(dim);
flag4lb.Resize(dim);

X_rand.Resize(dim);

while(t < max_iteration){
for(i=1;i<=searchagents_no;i+=1){
for(j=1;j<=dim;j+=1){
!Devolver los agentes de búsqueda que van más allá de los límites del espacio de
búsqueda.
vartemp = positions.Get(i,j);
vartemp1 = positions.Get(i,j);
scaling1=Paralim.Get(2,j);
scaling2=Paralim.Get(1,j);
vartemp = vartemp > scaling1;
vartemp1 = vartemp1 < scaling2;
flag4ub.Set(j,vartemp);
flag4lb.Set(j,vartemp1);
vartemp1 = positions.Get(i,j);
a = flag4ub.Get(j);
b = flag4lb.Get(j);
vartemp2 = .not.(a+b);
vartemp3 = scaling1*a+scaling2*b;
vartemp = vartemp1*vartemp2+vartemp3;
positions.Set(i,j,vartemp);
}
n=0;
for(ele=1;ele<=dim;ele+=1){
vartemp = positions.Get(i,ele);
if(vartemp < 0){
n+=1;
}
!printf('vartemp = %f',vartemp);
v_vartemp.Set(ele,vartemp);
}
!!!!!!
contador=contador+1;
!printf('ObjF = %d',contador);
```

```
!!!!!!
!!!! Llamar a la funcion objetivo
Obj_function_2.Execute(ffx);
!!!! Resultado de la funcion objetivo
fitness=ffx;
if(n=0){
!Actualizar al lider
if(fitness < leader_score){
leader_score = fitness; !Actualizar alfa
printf('ls hasta el momento = %g y t = %g',leader_score,t);
ffx_conv=leader_score;
ffx_eval=t;
Convergence.WriteDraw(); ! Internally save convergence results (for plotting)
for(ele1=1;ele1<=dim;ele1+=1){
vartemp = v_vartemp.Get(ele1);
leader_pos.Set(ele1,vartemp);
}
!ffx_conv=leader_score;
!ffx_eval=z;
!z=z+1;
m=0;
if(leader_score < 50e-6){
m=1;
break;
}
}
}
if(m=1){
break;
}
}

a1 = 2 - t*((2)/max_iteration); !a disminuye linealmente de 2 a 0 en la ecuación. (2,3)
a2 = -1 + t*((-1)/max_iteration); !a2 disminuye linealmente de -1 a -2 para calcular t en la
ecuación. (3,12)
! Actualizar la posición de los agentes de búsqueda
for(i=1;i<=searchagents_no;i+=1){
r1 = fRand(0); ! r1 es un número aleatorio en [0,1]
r2 = fRand(0); ! r2 es un número aleatorio en [0,1]
A = 2*a1*r1-a1; ! Eq. (2.3) en el papel
C = 2*r2; ! Eq. (2.4) in the paper
b = 1; ! parameters in Eq. (2.5)
vartemp = fRand(0);
l = (a2-1)*vartemp+1; ! parameters in Eq. (2.5)
p = fRand(0); ! p in Eq. (2.6)
!p = 0.5;
for(j=1;j<=dim;j+=1){
if(p < 0.5){
if(abs(A) >= 1){
vartemp2 = fRand(0);
rand_leader_index = floor(searchagents_no*vartemp2+1);
for(ele1=1;ele1<=dim;ele1+=1){
vartemp4 = positions.Get(rand_leader_index,ele1);
X_rand.Set(ele1,vartemp4);
```

```
}
vartemp = X_rand.Get(j);
vartemp1 = positions.Get(i,j);
D_X_rand = abs(C*vartemp-vartemp1); ! Eq. (2.7)
vartemp2 = vartemp-A*D_X_rand;
positions.Set(i,j,vartemp2); ! Eq. (2.8)
}else if(abs(A) < 1){
vartemp = leader_pos.Get(j);
vartemp2 = positions.Get(i,j);
D_Leader = abs(C*vartemp-vartemp2); ! Eq. (2.1)
vartemp3 = vartemp - A*D_Leader;
positions.Set(i,j,vartemp3); ! Eq. (2.2)
}
}else if(p >= 0.5){
vartemp = leader_pos.Get(j);
vartemp2 = positions.Get(i,j);
distance2Leader = abs(vartemp - vartemp2);
vartemp3 = distance2Leader*exp(b*l)*cos(l*2*pi())+vartemp; ! Eq. (2.5)
positions.Set(i,j,vartemp3);
}
}
}

t=t+1;
vartemp = leader_score;
convergence_curve.Set(t,vartemp);
printf('t = %d y leader_score = %g',t,leader_score);
!if(leader_score < 0.001){
! break;
!}
if(m=1){
break;
}
}


! Creating convergence plot
!GrB = GetGraphBoard();
!if (GrB=NULL) { output('No Graphics Board open'); exit(); }

!Plot = NULL;
!ViPg = GrB.GetPage(this:loc_name,1);
!if (ViPg) {
! ViPg.SetResults(Convergence);
! Plot=ViPg.GetVI('qot','VisPlot',1);
! if (Plot) {
! Plot:use_x=1;
! Plot.SetXVar(this,'b:ffx_eval');
! Plot.AddVars(this,'b:ffx_conv');
! }
!}
!if (ViPg=NULL.or.Plot=NULL) {
! Warn('Failed to create plot, continue calculation ...');
!}
```

```
!************************************

!i=0;
!j=0;
!for(i=1;i<=dim;i+=1){
! vartemp = leader_pos.Get(i);
! vartemp = abs(vartemp);
! leader_pos.Set(i,vartemp);
!}


!-----Load Best Param

nparam=v_vartemp.Size();
for(ele=1;ele<=nparam/5;ele+=1){
if (ele=1){
PSS1:Kpss=leader_pos.Get(1);
var1=leader_pos.Get(2);
var2=leader_pos.Get(3);
PSS1:T1=var2;
PSS1:T2=var2/var1;
var1=leader_pos.Get(4);
var2=leader_pos.Get(5);
PSS1:T3=var2;
PSS1:T4=var2/var1;
} else if (ele=2){
PSS2:Kpss=leader_pos.Get(6);
var1=leader_pos.Get(7);
var2=leader_pos.Get(8);
PSS2:T1=var2;
PSS2:T2=var2/var1;
var1=leader_pos.Get(9);
var2=leader_pos.Get(10);
PSS2:T3=var2;
PSS2:T4=var2/var1;
} else if (ele=3){
PSS3:Kpss=leader_pos.Get(11);
var1=leader_pos.Get(12);
var2=leader_pos.Get(13);
PSS3:T1=var2;
PSS3:T2=var2/var1;
var1=leader_pos.Get(14);
var2=leader_pos.Get(15);
PSS3:T3=var2;
PSS3:T4=var2/var1;
} else if (ele=4){
PSS4:Kpss=leader_pos.Get(16);
var1=leader_pos.Get(17);
var2=leader_pos.Get(18);
PSS4:T1=var2;
PSS4:T2=var2/var1;
var1=leader_pos.Get(19);
var2=leader_pos.Get(20);
PSS4:T3=var2;
```

```
PSS4:T4=var2/var1;
}
}


EchoOn(); ! Re-activating the user interface
Time_f=GetTime(4); ! Defining the final processor time
Dpl_time=Time_f-Time_o; ! Computing the elapsed time
printf('%s: %g','Processor Time',Dpl_time); ! Printing the elapsed time
```

ANEXO III. Código DPL Función Objetivo para la sintonización

```
! Objective Function Variable Declaration
int ele,nparam,cases,ele1,Neig;
double param,var1,var2,ele2;
double damplim, eigreal,eigimag,mindamp,minfreq,freq,damp;

! Defining the damping threshold
damplim = 0.051; ! Damping threshold equal to 15%

!!!! OBJECTIVE FUNCTION: Minimizing {abs(damplim-mindamp)}
nparam=x_denorm.Size();

for(ele=1;ele<=nparam/5;ele+=1){
if (ele=1){
PSS1:Kpss=x_denorm.Get(1);
var1=x_denorm.Get(2);
var2=x_denorm.Get(3);
PSS1:T1=var2;
PSS1:T2=var2/var1;
var1=x_denorm.Get(4);
var2=x_denorm.Get(5);
PSS1:T3=var2;
PSS1:T4=var2/var1;
} else if (ele=2){
PSS2:Kpss=x_denorm.Get(6);
var1=x_denorm.Get(7);
var2=x_denorm.Get(8);
PSS2:T1=var2;
PSS2:T2=var2/var1;
var1=x_denorm.Get(9);
var2=x_denorm.Get(10);
PSS2:T3=var2;
PSS2:T4=var2/var1;
} else if (ele=3){
PSS3:Kpss=x_denorm.Get(11);
var1=x_denorm.Get(12);
var2=x_denorm.Get(13);
PSS3:T1=var2;
PSS3:T2=var2/var1;
var1=x_denorm.Get(14);
var2=x_denorm.Get(15);
PSS3:T3=var2;
PSS3:T4=var2/var1;
} else if (ele=4){
PSS4:Kpss=x_denorm.Get(16);
var1=x_denorm.Get(17);
var2=x_denorm.Get(18);
PSS4:T1=var2;
PSS4:T2=var2/var1;
var1=x_denorm.Get(19);
var2=x_denorm.Get(20);
PSS4:T3=var2;
```

```
PSS4:T4=var2/var1;
}
}

! Computing eigenanalysis
mindamp=1;

Initialjc.Execute();
Modal.Execute();
LoadResData(Results);
Neig = ResNval(Results,0);
for(ele=0;ele<=Neig-1;ele+=1){
GetResData(eigreal,Results,ele,0);
GetResData(eigimag,Results,ele,1);
if ({abs(eigimag)>0}){
freq=abs(eigimag)/(2*pi());
damp=-eigreal/sqrt(sqr(eigreal)+sqr(eigimag));
mindamp=min(mindamp,damp);
}
}

! Computing Objective Function
ffx = abs(damplim-mindamp);

!printf('%f',ffx);
```

ANEXO IV. Código Python Ajuste con el Residuo Estimado

```
import sys #Provee variables y funcionalidades
import powerfactory as pf #Cuando se tiene esta libreria y el sys se tiene acceso a las
funciones del Power Factory
import numpy as np
#numpy Da soporte para crear vectores y matrices grandes multidimensionales, y
tambien contiene funciones matematicas de alto nivel
import pandas as pd #librería especializada en el manejo y analisis de estructuras de
datos.
import cmath #libreria de c de matematicas
import math

app = pf.GetApplication() #Da acceso a la aplicacion
app.ResetCalculation() #Resetea los calculos

def exportCSV(ValoresMVMO):
NOMBRES_PSS=np.array(['PSS 5', 'PSS 6','PSS 9','PSS 10'])
KPSS=np.array([ValoresMVMO[0],ValoresMVMO[5],ValoresMVMO[10],ValoresMVMO[15]
])
T1PSS=np.array([ValoresMVMO[1],ValoresMVMO[6],ValoresMVMO[11],ValoresMVMO[1
6]])
T2PSS=np.array([ValoresMVMO[2],ValoresMVMO[7],ValoresMVMO[12],ValoresMVMO[1
7]])
T3PSS=np.array([ValoresMVMO[3],ValoresMVMO[8],ValoresMVMO[13],ValoresMVMO[1
8]])
T4PSS=np.array([ValoresMVMO[4],ValoresMVMO[9],ValoresMVMO[14],ValoresMVMO[1
9]])

DAT= np.column_stack((NOMBRES_PSS, KPSS, T1PSS, T2PSS, T3PSS, T4PSS))
return DAT

def RIPart():
comMA = app.GetFromStudyCase('ComMod') #Obtengo del caso de estudio el objeto
ComMod (Ventana cuando corre la simulacion Modal)
comMA.dirMatl=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England AM'
comMA.Execute()
comRes = app.GetFromStudyCase('ComModres')
#comRes.Execute()
resd = app.GetFromStudyCase("Eigenvalues.ElmRes")
comres = app.GetFromStudyCase('ComRes')
comres.pResult=resd
comres.f_name=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\ComprobarAmort.csv'
##
comres.iopt_csel=0
comres.Execute()
df = pd.read_csv(r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\ComprobarAmort.csv', header=[0,1],delimiter=';')
g = df.columns
valores1 =df[df.columns].values
#app.PrintPlain('Numero de columnas')
#app.PrintPlain('valores1')
```

```
#app.PrintPlain(valores1)
#app.PrintPlain('columnasvalores')
columnasvalores1=len(valores1[1])
#app.PrintPlain(columnasvalores1)
re =df[df.columns[1]].values
im =df[df.columns[2]].values
freq=abs(im)/(2*math.pi)
filas=len(re)
damp=np.zeros(filas)
indice_mindamp=[]
for x in range(0,filas):
damp[x]=-re[x]/(math.sqrt(re[x]**2+im[x]**2))
if damp[x] < 0.05:
indice_mindamp.append(x)
#num_val_indice=len(indice_mindamp)
#app.PrintPlain('posicionmindamping:')
#app.PrintPlain(fivedamp)
#app.PrintPlain('indice_mindamp:')
#app.PrintPlain(indice_mindamp)
# Poner indices de modos con menor amortiguamiento que tengan parte imaginaria
mayor o igual a 0
filas_indice_mindamp=len(indice_mindamp)
vartemp=0
indice_mindamp_bien=[]
for j in range(0,filas_indice_mindamp):
vartemp=indice_mindamp[j]
if (im[vartemp] >= 0):
indice_mindamp_bien.append(vartemp)
filas_indice_mindamp_bien=len(indice_mindamp_bien)
fivedamp=[]
for x in range(0,filas_indice_mindamp_bien):
vartemp = damp[indice_mindamp_bien[x]]
fivedamp.append(vartemp)
# Obtengo el modo con menor amortiguamiento
mindamp = np.nanmin(np.where(damp == 0, np.nan, damp))
for y in range(0,filas):
if damp[y] == mindamp:
break
#################################################################
#Inicializo la matriz para los factores de participacion, observ y controla
FactoresPC=np.zeros((filas,columnasvalores1-3))
#Lazo for para poner todos los factores P C O
for k in range(0,filas):
contador1=0
for z in range(3,columnasvalores1):
FactoresPC[k,contador1]=valores1[k,z]
contador1=contador1+1
#Columnas de P C O
columnaspartipacion=(columnasvalores1-3)/6
columnaspartipacion=int(columnaspartipacion)
#Lazo para poner la Magnitud Participacion
MagnitudParticipacion=np.zeros((filas,columnaspartipacion))
for k in range(0,filas):
contador1=4
```

```
for z in range(0,columnaspartipacion):
MagnitudParticipacion[k,z]=FactoresPC[k,contador1]
contador1=contador1+6
#Lazo para poner la Angulo Participacion
AnguloParticipacion=np.zeros((filas,columnaspartipacion))
for k in range(0,filas):
contador1=5
for z in range(0,columnaspartipacion):
AnguloParticipacion[k,z]=FactoresPC[k,contador1]
contador1=contador1+6
#TXT variables de estado
datos = pd.read_csv(r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\VariableToIdx_Amat.txt',header=0, delim_whitespace=True)
#Pone en vectores al txt
datos1=np.asarray(datos)
filasdatos=len(datos1)
columnasdatos=len(datos1[0])
# Pone todas las variables de estado en un vector
Elm='.Elm'
state_variable=[]
for k in range(0,filasdatos):
for z in range(0,columnasdatos):
a = datos1[k,z]
a=str(a)
if (Elm in a):
l=datos1[k,z+1]
#app.PrintPlain(l)
state_variable.append(l)
# Posiciones de speed en state_variable
sp='speed'
indices_speed=[]
for i in range(0,filasdatos):
if (sp in state_variable[i]):
indices_speed.append(i)
# Nombre Generadores G1speed G2speed ... G10speed
NGen=[]
for i in range(0,10):
G='G'
a=i+1
a=str(a)
a=G+a+sp
NGen.append(a)
# Ver la magnitud de Participacion de los modos menor a 5% amort con im >= 0
#Encerara matrices para guardar Parti Magnitud y angulo de los
#modos con menor amortiguamiento
Modos_Mag_Participacion=np.zeros((filas_indice_mindamp_bien,10)) # 10 por N
Generadores
Modos_Ang_Participacion=np.zeros((filas_indice_mindamp_bien,10))
for j in range(0,filas_indice_mindamp_bien):
for i in range(0,10):
Modos_Mag_Participacion[j,i]=(MagnitudParticipacion[indice_mindamp[j],indices_speed[i]]
)
Modos_Ang_Participacion[j,i]=(AnguloParticipacion[indice_mindamp[j],indices_speed[i]])
```

```
# Magnitud y Angulo del modo menor que no cumple con el amortiguamiento
Modomin_Mag_Participacion=[]
Modomin_Ang_Participacion=[]
Real_Modomin_Participacion=[]
for i in range(0,10):
vartemp=MagnitudParticipacion[y,indices_speed[i]]
Modomin_Mag_Participacion.append(vartemp)
vartemp1=AnguloParticipacion[y,indices_speed[i]]
Modomin_Ang_Participacion.append(vartemp1)
vartemp1=vartemp1*(math.pi/180)
vartemp2=vartemp*math.cos(vartemp1)
Real_Modomin_Participacion.append(vartemp2)
# Poner datos solo para los 4 PSS que se tiene
NGen4PSS=[]
Modomin_Mag_Participacion4PSS=[]
Modomin_Ang_Participacion4PSS=[]
Real_Modomin_Participacion4PSS=[]
for i in range(0,10):
if (i == 4 or i == 5 or i == 8 or i == 9):
vartemp=NGen[i]
NGen4PSS.append(vartemp)
vartemp=Modomin_Mag_Participacion[i]
Modomin_Mag_Participacion4PSS.append(vartemp)
vartemp=Modomin_Ang_Participacion[i]
Modomin_Ang_Participacion4PSS.append(vartemp)
vartemp=Real_Modomin_Participacion[i]
Real_Modomin_Participacion4PSS.append(vartemp)
# Obtener el Generador que tiene mayor participacion de entre los 4 PSS
Real_Modomin_Participacion4PSSabs=np.abs(Real_Modomin_Participacion4PSS)
max_magnitud=np.max(Real_Modomin_Participacion4PSSabs)
for z in range(0,4):
if Real_Modomin_Participacion4PSSabs[z] == max_magnitud:
break
NGen4PSSmax=(NGen4PSS[z])
#Poniendo ya el nombre del PSS que mas participa
diez='10'
nueve='9'
seis='6'
cinco='5'
if cinco in NGen4PSSmax:
parti='PSS 5.ElmDsl'
elif seis in NGen4PSSmax:
parti='PSS 6.ElmDsl'
elif nueve in NGen4PSSmax:
parti='PSS 9.ElmDsl'
elif diez in NGen4PSSmax:
parti='PSS 10.ElmDsl'
#################################################################
return re[y], im[y], freq[y], damp[y], parti


def MINDAMP():
comMA = app.GetFromStudyCase('ComMod') #Obtengo del caso de estudio el objeto
ComMod (Ventana cuando corre la simulacion Modal)
```

```
comMA.dirMatl=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England AM'
comMA.Execute()
comRes = app.GetFromStudyCase('ComModres')
#comRes.Execute()
resd = app.GetFromStudyCase("Eigenvalues.ElmRes")
comres = app.GetFromStudyCase('ComRes')
comres.pResult=resd
comres.f_name=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\ComprobarAmort.csv'
##
comres.iopt_csel=0
comres.Execute()
df = pd.read_csv(r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\ComprobarAmort.csv', header=[0,1],delimiter=';')
g = df.columns
valores1 =df[df.columns].values
#app.PrintPlain('Numero de columnas')
#app.PrintPlain('valores1')
#app.PrintPlain(valores1)
#app.PrintPlain('columnasvalores')
columnasvalores1=len(valores1[1])
#app.PrintPlain(columnasvalores1)
re =df[df.columns[1]].values
im =df[df.columns[2]].values
filas=len(re)
damp=np.zeros(filas)
indice_mindamp=[]
for x in range(0,filas):
damp[x]=-re[x]/(math.sqrt(re[x]**2+im[x]**2))
# Obtengo el modo con menor amortiguamiento
mindamp = np.nanmin(np.where(damp == 0, np.nan, damp))
return mindamp


def ObtContPSS(a):
pss = app.GetCalcRelevantObjects(a)
Kpss = pss[0].GetAttribute('e:params:Kpss')
T1 = pss[0].GetAttribute('e:params:T1')
T2 = pss[0].GetAttribute('e:params:T2')
T3 = pss[0].GetAttribute('e:params:T3')
T4 = pss[0].GetAttribute('e:params:T4')
return Kpss, T1, T2, T3, T4

def SetearK(a,b):
pss = app.GetCalcRelevantObjects(a)
pss[0].SetAttribute('e:params:Kpss',b)


def AjusteResiduofunc():
min_damp=MINDAMP()

if min_damp < 0.05:
app.PrintPlain('El minimo amortiguamiento es menor a 5% y es:')
app.PrintPlain(min_damp)
```

```
app.PrintPlain('Es necesario realizar un ajuste.')
app.PrintPlain(' ')
else:
app.PrintPlain('El minimo amortiguamiento es mayor a 5% y es:')
app.PrintPlain(min_damp)
app.PrintPlain('No es necesario realizar un ajuste.')
app.PrintPlain(' ')
K1=9999
Kpssfinal=9999
return K1,Kpssfinal
while (min_damp < 0.05):
Tw=10
a1,b1,f1,amot1,p1=RlPart()
#app.PrintPlain('p1')
#app.PrintPlain(p1)
#app.PrintPlain(' ')
K1,T1,T2,T3,T4=ObtContPSS(p1)
#app.PrintPlain('Kpss')
#app.PrintPlain(K1)
inidamp=MINDAMP()
#app.PrintPlain(inidamp)
K2_1=K1*0.99
SetearK(p1,K2_1)
damp1=MINDAMP()
#app.PrintPlain(damp1)
K2_2=K1*1.01
SetearK(p1,K2_2)
damp2=MINDAMP()
#app.PrintPlain(damp2)
if damp1 > damp2:
condicion=1
K2=K2_1
#app.PrintPlain('Signo Negativo para la K')
else:
condicion=2
K2=K2_2
#app.PrintPlain('Signo Positivo para la K')
SetearK(p1,K2)
K2,T1,T2,T3,T4=ObtContPSS(p1)
#app.PrintPlain('Kpss')
#app.PrintPlain(K2)
a2,b2,f2,amort2,p2=RlPart()
Kpss=abs(K1-K2)
HPSS=((Tw*a1)/(1+a1*Tw))*((1+a1*T1)/(1+a1*T2))*((1+a1*T3)/(1+a1*T4))
Distancia=abs(a1-a2)
Residuo=(Distancia)/(HPSS*Kpss)
# Obtener K para el amortiguamiento del 5%
Amort_des=5.08/100
R_des=-(Amort_des*b2)/(math.sqrt(1-Amort_des**2))
Distancia2=abs(a2-R_des)
Kpss2=(Distancia2)/(HPSS*Residuo)
### MAS O MENOS
if condicion==1:
Kpssfinal=K2-Kpss2
```

```
else:
Kpssfinal=K2+Kpss2
app.PrintPlain('Kfinal')
app.PrintPlain(Kpssfinal)
SetearK(p1,Kpssfinal)
min_damp=MINDAMP()
if min_damp > 0.05:
app.PrintPlain('El minimo amortiguamiento ya es mayor al 5% y es:')
app.PrintPlain(min_damp)
return K1, Kpssfinal

def tomarValores():
vectorpss = []
pss = app.GetCalcRelevantObjects('PSS 5.ElmDsl')
vectorpss.append(pss[0].GetAttribute('e:params:Kpss'))
vectorpss.append(pss[0].GetAttribute('e:params:T1'))
vectorpss.append(pss[0].GetAttribute('e:params:T2'))
vectorpss.append(pss[0].GetAttribute('e:params:T3'))
vectorpss.append(pss[0].GetAttribute('e:params:T4'))
pss = app.GetCalcRelevantObjects('PSS 6.ElmDsl')
vectorpss.append(pss[0].GetAttribute('e:params:Kpss'))
vectorpss.append(pss[0].GetAttribute('e:params:T1'))
vectorpss.append(pss[0].GetAttribute('e:params:T2'))
vectorpss.append(pss[0].GetAttribute('e:params:T3'))
vectorpss.append(pss[0].GetAttribute('e:params:T4'))
pss = app.GetCalcRelevantObjects('PSS 9.ElmDsl')
vectorpss.append(pss[0].GetAttribute('e:params:Kpss'))
vectorpss.append(pss[0].GetAttribute('e:params:T1'))
vectorpss.append(pss[0].GetAttribute('e:params:T2'))
vectorpss.append(pss[0].GetAttribute('e:params:T3'))
vectorpss.append(pss[0].GetAttribute('e:params:T4'))
pss = app.GetCalcRelevantObjects('PSS 10.ElmDsl')
vectorpss.append(pss[0].GetAttribute('e:params:Kpss'))
vectorpss.append(pss[0].GetAttribute('e:params:T1'))
vectorpss.append(pss[0].GetAttribute('e:params:T2'))
vectorpss.append(pss[0].GetAttribute('e:params:T3'))
vectorpss.append(pss[0].GetAttribute('e:params:T4'))
return vectorpss

## Definición de funciones se accede a objetos para modificarlos
def setupSimulation(comInc, comSim, tiempoSIM): #Configuracion de la simulacion con
funciones
comInc.iopt_sim = 'rms' #Sirve con simulaciones rms
comInc.iopt_show = 0
comInc.iopt_adapt = 0
#comInc.dtemt = 0.0001
comInc.tstart = 0 #Desde donde calcula condiciones iniciales
comInc.dtgrd = 0.01666666
comSim.tstop = tiempoSIM
def runSimulation(comInc,comSim,direccion):
app.EchoOff()#Todas las aplicaciones de power factory tienen la palabra app, congela la
interfaz del usuario
comInc.Execute()
```

```
app.EchoOn() #Reactiva la interfaz del usuario , esto se pone para que no de problemas
al inicio de la simulacion
comSim.Execute()
#Exportar los datos
resd = app.GetFromStudyCase("CON PSSMVMO.ElmRes")
comres = app.GetFromStudyCase('ComRes')
comres.pResult=resd
comres.f_name=direccion
##
comres.iopt_csel=1
comres.Execute()

def outservicePSS():
pss = app.GetCalcRelevantObjects('PSS 5.ElmDsl')
pss[0].outserv = 1
pss = app.GetCalcRelevantObjects('PSS 6.ElmDsl')
pss[0].outserv = 1
pss = app.GetCalcRelevantObjects('PSS 9.ElmDsl')
pss[0].outserv = 1
pss = app.GetCalcRelevantObjects('PSS 10.ElmDsl')
pss[0].outserv = 1

def inservicePSS():
pss = app.GetCalcRelevantObjects('PSS 5.ElmDsl')
pss[0].outserv = 0
pss = app.GetCalcRelevantObjects('PSS 6.ElmDsl')
pss[0].outserv = 0
pss = app.GetCalcRelevantObjects('PSS 9.ElmDsl')
pss[0].outserv = 0
pss = app.GetCalcRelevantObjects('PSS 10.ElmDsl')
pss[0].outserv = 0
##

app.ClearOutputWindow()

# Sin PSS Exportar datos del PMU sin PSS

outservicePSS()

tiempoSIM = 10

comInc = app.GetFromStudyCase('ComInc')
comSim = app.GetFromStudyCase('ComSim')

setupSimulation(comInc, comSim, tiempoSIM)
direccionPMU=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\SINPSSPMUotro.csv'
runSimulation(comInc, comSim, direccionPMU)

inservicePSS()

# Correr el Dpl que esta en la misma carpeta

PythonScript=app.GetCurrentScript()
```

```
#-----------if the ComDpl is saved on the same location as ComPython----------
FolderOfPythonScript=PythonScript.GetParent()

Comprobar_MVMO=
FolderOfPythonScript.GetContents('Comprobar_MVMO.ComDpl',1)[0]
app.PrintPlain(' ')
app.PrintPlain('Ejecucion del Comprobar_MVMO')
app.PrintPlain(' ')
Comprobar_MVMO.Execute()

### Exportar datos del PMU con los parametros MVMOseteado

tiempoSIM = 10

comInc = app.GetFromStudyCase('ComInc')
comSim = app.GetFromStudyCase('ComSim')

setupSimulation(comInc, comSim, tiempoSIM)
direccionPMU=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\MVMOPMUseteado.csv'
runSimulation(comInc, comSim, direccionPMU)

app.PrintPlain(' ')
app.PrintPlain('Ajuste Residuo MVMO')
app.PrintPlain(' ')
K1MVMO, K2MVMO= AjusteResiduofunc()


### Exportar datos del PMU con los parametros MVMOajustado

setupSimulation(comInc, comSim, tiempoSIM)
direccionPMU=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\MVMOPMUajustado.csv'
runSimulation(comInc, comSim, direccionPMU)


# Tomar los valores de PSS MVMO

ValoresMVMO=tomarValores()
app.PrintPlain(ValoresMVMO)
# Exportar en un csv los valores que se obtuvieron del MVMO
DATMVMO=exportCSV(ValoresMVMO)
np.savetxt(r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\ValoresMVMOajustados.csv', DATMVMO, delimiter =";", fmt="%s")
Comprobar_WOA= FolderOfPythonScript.GetContents('Comprobar_WOA.ComDpl',1)[0]
app.PrintPlain(' ')
app.PrintPlain('Ejecucion del Comprobar_WOA')
app.PrintPlain(' ')
Comprobar_WOA.Execute()

### Exportar datos del PMU con los parametros WOAseteado

setupSimulation(comInc, comSim, tiempoSIM)
```

```
direccionPMU=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\WOAPMUseteado.csv'
runSimulation(comInc, comSim, direccionPMU)

app.PrintPlain(' ')
app.PrintPlain('Ajuste Residuo WOA')
app.PrintPlain(' ')
K1WOA, K2WOA= AjusteResiduofunc()


### Exportar datos del PMU con los parametros WOAajustado

setupSimulation(comInc, comSim, tiempoSIM)
direccionPMU=r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\WOAPMUajustado.csv'
runSimulation(comInc, comSim, direccionPMU)

# Tomar los valores de PSS WOA

ValoresWOA=tomarValores()
app.PrintPlain(ValoresWOA)
# Exportar en un csv los valores que se obtuvieron del MVMO
DATWOA=exportCSV(ValoresWOA)
np.savetxt(r'C:\Users\LENOVO\Desktop\Proyecto Final\New England
AM\ValoresWOAajustados.csv', DATWOA, delimiter =";", fmt="%s")
DistMVMO=abs(K1MVMO-K2MVMO)
DistWOA=abs(K1WOA-K2WOA)
if (K2MVMO >=0 and K2WOA >= 0):
if (DistMVMO < DistWOA):
app.PrintPlain(' ')
app.PrintPlain('Ajuste MVMO es mejor')
app.PrintPlain(' ')
else:
app.PrintPlain(' ')
app.PrintPlain('Ajuste WOA es mejor')
app.PrintPlain(' ')
elif (K2MVMO < 0 and K2WOA < 0):
app.PrintPlain('No se puede realizar el ajuste pruebe corriendo otra vez los algoritmos')
elif (K2MVMO >= 0 and K2WOA < 0):
app.PrintPlain(' ')
app.PrintPlain('Ajuste MVMO es mejor')
app.PrintPlain(' ')
elif (K2MVMO < 0 and K2WOA >= 0):
app.PrintPlain(' ')
app.PrintPlain('Ajuste WOA es mejor')
app.PrintPlain(' ')
```

ANEXO V. Código en Matlab para cargar una base de datos al software “ringdown”

```
%% Crear base de datos para el ringdown
clc, clear all, close all
%% LEER DATOS EXCEL

%SINPSSPMUinicial
%MVMOPMUinicial
%WOAPMUinicial

% Carga datos desde excel
[file,filePath] = uigetfile('WOAPMUajustado.csv'); %devuelve el nombre del archivo
%y la ruta al archivo cuando el usuario hace clic en Abrir.
%Si el usuario hace clic en Cancelar o en el botón de cerrar la ventana (X),
%entonces uigetfile devuelve 0 para ambos argumentos de salida.
dname = [filePath file];

[num,txt,raw]=xlsread(dname); %devuelve además los campos de texto del array de
celdas txt,
%y los datos numéricos y de texto del array de celdas raw,
%utilizando cualquiera de los argumentos de entrada de las sintaxis anteriores.

index=strfind(file,'.'); %k = strfind(str,pattern) busca str para las apariciones de pattern.
%La salida, k, indica el índice de inicio de cada ocurrencia de pattern en str.

file(index+1:end)=[] %borra todo lo que esta despues del punto del nombre del archivo
file=[file 'mat'];%le agrega mat a todo lo que se borro después del punto
% filePath=uigetdir;

file=['Database_' file];
dname = [filePath '\' file ]; %Direccion del Database

%% CREAR BASE DE DATOS
clc, clearvars  -except raw txt num file dname filePath

[d nrow]=size(raw); %[m,n] = size(A) devuelve el número de filas y columnas cuando A es
una matriz.

signals=txt(2,:); %Nombre de las señales del excel en forma de celda
SignalsDS={'Positive-Sequence, Active Power in MW' 'Speed in p.u.' 'Electrical Frequency
in Hz' 'Deviation of the El. Frequency in Hz' 'firot in deg' 'u, Magnitude in p.u.' 'Active
Power/Terminal i in MW'};
Signal_PMUs={'Stot_Re' 'sp' 'f' 'df' 'firot' 'V_Am' 'P_LT'};

for i=1:7
   for ii=1:nrow
      indice(ii)=~isempty(strfind(signals{ii},SignalsDS{i})); %TF = isempty(A) devuelve 1
lógico (verdadero) si A está vacío y 0 lógico (falso) en caso contrario.
                                                %Una matriz, tabla o calendario vacío tiene al
menos una dimensión con longitud 0, como 0 por 0 o 0 por 5.
                                                %Al momento que encuentre igual pondra 1
```

```
    end
    Name_Gen=txt(1,indice);
    T=raw(3:end,indice);
    T=cell2table(T);
%     if strcmp(SignalsDS{i},'fe')
%         SignalsDS{i}
%         T=T{:,:}*60;
%         T=array2table(T);
%     end

    T.Properties.VariableNames=Name_Gen; %Poner nombre en la tabla

    assignin('base',Signal_PMUs{i},T); %assignin(ws,var,val) asigna el valor val a la
variable var en el espacio de trabajo ws.
                                %Por ejemplo, assignin('base','x',42) asigna el valor 42 a la
variable x en el espacio de trabajo base de MATLAB®.
    try
    save(dname,Signal_PMUs{i},'-append');
    catch ME
        if ~isempty(strfind(ME.message,'No such file or directory.'))
            save(dname,Signal_PMUs{i});
        end
    end

end

NSignal_PMUs=Nombres_Senales(Signal_PMUs);

Tiempo=[0:1/60:(d-1)/60];
save(dname,'Signal_PMUs','NSignal_PMUs','Tiempo','-append');

%Signal_PMUs={'Stot_Re' 'sp' 'f' 'df' 'firot' 'V_Am' 'P_LT'}; Para saber la senal

load(file)
%load('Database_MVMO.mat')
NamePMU=Stot_Re.Properties.VariableNames;
Stot_Re=table2array(Stot_Re);
[row,col]=size(Stot_Re);
%name=num2str([1:col].','G%02d');
%Si es para las lineas de transmision cerca del G1 se pone el otro name
name=num2str([1:col].','LT%02d');
Tiempo=[0:1/60:(row-1)/60];
M=[Tiempo' Stot_Re];

ringdown;
movefile 'AnalysisProny.mat' 'P9_WOAPMUajustado_prony.mat'
clc, clear all;
```