2018

IMPLEMENTACIÓN DE CONTROLADOR INTELIGENTE PARA EL CONTROL AUTOMÁTICO DE LA GENERACIÓN Y SU APLICACIÓN AL SISTEMA SING-SADI

ACUÑA GONZÁLEZ, FELIPE IGNACIO

Universidad Técnica Federico Santa María
Departamento de Ingeniería Eléctrica
Valparaíso, Chile

Implementación de controlador inteligente para el control automático de la generación y su aplicación al sistema SING-SADI

Felipe Acuña González

2018

Requisito parcial para obtener el título de:
Ingeniero Electricista

Profesor Guía:
Dr. Victor Hinojosa (UTFSM)
Dr. Ignacio Calle (UTFSM)

Valparaíso, 13 de Julio del 2018.

Universidad Técnica Federico Santa María
Departamento de Ingeniería Eléctrica
Valparaíso, Chile

Implementación de controlador inteligente para el control automático de la generación y su aplicación al sistema SING-SADI

Felipe Acuña González

2018

”La inteligencia es la capacidad de adaptarse al cambio.”
— Stephen Hawking

Índice general

Resumen

Actualmente en Chile ya se ha implementado el uso de control automático de generación (AGC por sus siglas en inglés) para hacer regulación secundaria de frecuencia. Pero hasta hace no mucho este proceso se hacía de forma manual, lo que derivaba en no cumplimientos de las exigencias de la NTSyCS en cuanto a los rangos deseables de frecuencia. La automatización de este proceso permitió mayores velocidades de respuesta, ya que no sería necesaria la intervención humana, lo cual desencadena en mayor calidad de suministro de energía hacia los clientes.

La mayoría de los modelos de AGC contemplan controladores PID, ya que se ha demostrado a lo largo de la historia que cumplen el objetivo de controlar el sistema sin estudios exagerados sobre sus parámetros. Sin embargo, estos controladores están sujetos a los puntos de operación en los que se encuentran, por lo que sería necesario cambiar los parámetros dependiendo de la situación.

Durante los últimos años, gracias a la mejora de procesamiento de los computadores, además de un gran aumento en el interés sobre estructuras de control no convencionales, han aparecido nuevas herramientas y controladores basados en inteligencia artificial.

Precisamente este trabajo pretende demostrar que los métodos de inteligencia artificial pueden ser aplicados al control de generación automática, y determinar si se puede obtener un mejor desempeño al ser utilizados en el modelo de un sistema real como lo es el sistema interconectado. Para ello se contempla primero un desarrollo de un marco teórico explicado en el capítulo 2, que permite entender el funcionamiento del AGC, la lógica difusa (fuzzy logic), las redes neuronales y sus métodos de aprendizaje, para finalmente describir el funcionamiento de las estructuras ANFIS. Seguidamente, en el capítulo 3, se tiene una breve descripción de las implementaciones del AGC alrededor del mundo, además de usos de la inteligencia artificial dentro del área eléctrica. Posteriormente, en el capítulo 4, se parte implementando un modelo mínimo funcional en Simulink, para luego pasar al diseño en un caso IEEE de 14 barras en Digsilent que contemple optimizaciones, ajustes pertinentes, y modificaciones finas, que finalmente puedan ser replicados sin mayores problemas en el modelo SING-SADI, que claramente es el que ocupará más poder de procesamiento. Finalmente en los capítulos 5 y 6 se tienen los resultados del sistema real, tanto para el SING, como para el sistema interconectado.

Obtenidos los controladores correctos y ajustados en el caso de SING-SADI, se hicieron comparaciones pertinentes con una configuración teórica del controlador PI a ser usado, donde finalmente se llegó a la conclusión que los controladores inteligentes funcionan de buena forma, logran controlar el sistema, pero hay que tener varias consideraciones presentes a la hora de utilizar cada uno de ellos. Durante este ensayo se pudo notar que solo uno de los controladores -ANFIS- mejoró la situación actual, debido a sus grandes condiciones de adaptabilidad, que le permiten reformular sus rangos operativos de buena manera.

Abstract

Nowadays, AGC has already been implemented to do secondary frequency control. However, not too long ago, this process was made manually, so this used to mean that the frequency was not meeting the requirements regarding to the desirable operative ranges. Automatization in this matter allowed better response speeds, because human intervention wouldn't be necessary anymore. This also resulted in better power supply quality towards the client.

Most AGC models use PID controllers, because they have shown all over its existence that they are successful controlling systems without any exaggerated study beforehand over its parameters. However, this good behavior is restricted to an operation range, so it would be preferable to change its parameter depending on the situation. During the last few years, thanks to greater computing power and the constantly growing interest in nonstandard control schemes, new tools and AI controller have shown up.

Actually, this research pretends to show that AI methods can be applied to AGC, and determine if a better performance can be achieved when being used in a real model like SING-SADI connection. For that, this work pretends to explain everything to someone not familiar with this kind of logic, starting in chapter 2 where it explains how AGC works, fuzzy logic, neural networks and its learning techniques, and ANFIS logic and how it works (available at the appendix). After this, in chapter 3, there's a brief description about AGC in USA, and then, short explanations about how AI is being used in Power System Industry. Later, in chapter 4, this research starts showing implementation using a minimalistic model in Simulink so it can be used later in 14-bus IEEE Digsilent's model, so it can also apply optimizations and custom tunning to it, that can be also used in SING-SADI model, that is going to be the one with the most processing power required. Finally in chapters 5 and 6, real system results are shown, considering both cases, for SING alone, and SING-SADI interconnection.

Once correct controllers are obtained and tunned for the SING-SADI case, comparissons with theorical tunning for PI controller are made, where it concluded that smart controllers work in proper way, achieve the goal of controlling the system, but severals issues must be adressed before using any of them. During this test it can be noticed that only one of the controllers -ANFIS- improved the current situation, thanks to its great adapting qualities, that let it change their operating ranges in proper way.

Glosario

ACE	Area Control Error.
AGC	Automatic Generation Control.
AI	Artificial Intelligence.
ANFIS	Adaptive Neuro Fuzzy Inference system .
ANN	Artificial Neural Networks.
API	Application Programming Interface.
BALTSO	Baltic Transmission System Operators.
CEN	Coordinador Eléctrico Nacional.
CPF	Control Primario de Frecuencia.
CSF	Control Secundario de Frecuencia.
DSL	Digsilent Simulation Language.
DPL	Digsilent Programming Language.
FACTS	Flexible AC transmission Systems.
FIS	Fuzzy Inference System .
ISE	Integrated Square Error .
NERC	North American Electric Reliability Corporation.
NTSyCS	Norma Técnica de Seguridad y Calidad de Servicio.
PID	Proportional Integrative Derivative
PSO	Particle Swarm Optimization
PSS	Power System Stabilizer.
SCADA	Supervisory Control And Data Acquisition.
SIC	Sistema Interconectado Central.
SING	Sistema Interconectado del Norte Grande.
SADI	Sistema Argentino de Interconexión.

1

Introducción

La frecuencia del sistema está dada por el balance de potencia generada y demandada en cierto instante. Si la demanda es mayor que la generación se produce una baja en la frecuencia, caso contrario, si la generación es mayor que la demanda, se produce una subida en la frecuencia. Además tomando en cuenta la naturaleza de la demanda, de carácter dinámico, y a veces con movimientos no previstos, se tienen fluctuaciones a lo largo del día en lo que respecta al valor de la frecuencia.

Para poder sopesar de alguna forma estos vaivenes se tienen métodos de control sobre la potencia generada, de modo de hacer un buen balance entre demanda y generación. Por un lado lo primero en actuar ante una perturbación, es la respuesta inherente del sistema, o sea la respuesta inercial más el amortiguamiento. Dando paso luego, a la primera actuación de regulación de frecuencia, llamado control primario de frecuencia (CPF), el cual permite hacer una primera corrección sobre las unidades generadoras, de modo de detener los cambios abruptos de potencia en el sistema. Luego se tiene un segundo paso de control, que corresponde al control secundario de frecuencia (CSF), el cual tiene por objetivo intentar llevar al sistema a un balance correcto de potencia, y de ese modo reducir la desviación de frecuencia. También se tiene un tercer paso de control, que es llamado control terciario de frecuencia, que intenta llevar las potencias generadas a valores económicos en lo que respecta a despachos eficientes.

Actualmente, se está llevando una transición en lo que respecta al control secundario de frecuencia en Chile. Hasta el 2017, este proceso se llevaba a cabo de forma manual, esto significaba que no se lograba un ajuste fino en el balance mencionado anteriormente, lo cual resultaba en que no se cumplían los estándares fijados por la norma chilena en cuanto a calidad y servicio. Actualmente diversos estudios están creando camino para lograr tener un sistema automatizado del CSF, en el cual dicho proceso tiene el nombre de AGC por sus siglas en inglés. Esto permitiría mejorar la calidad de servicio y lograr encauzar el problema mencionado anteriormente.

Sin embargo para lograr tener un buen sistema automatizado de generación es necesario tener en cuenta muchos factores, como lo son: la cantidad de reserva en MW de las máquinas generadoras, la toma de carga de las mismas, interconexiones entre áreas, entre otros. Todos estos factores deben ser analizados de forma de crear un diseño de control que permita un control seguro y eficiente de las reservas de las máquinas.

Como se ha mencionado, este es un problema de control de generación. Con lo que se tiene que el esquema de automatización verá un error (llamado ACE en este caso como se verá posteriormente), y con ello hará decisiones sobre modificaciones en las potencias generadas.

Este problema se ha abordado generalmente en los sistemas de control con esquemas de controladores *clásicos* como lo son los P, PI, PID, los cuales ofrecen un gran rango de control y por lo general un buen funcionamiento. Sin embargo, deben ser optimizados para cada caso, y recurrentemente tienen problemas de robustez, o sea, no operan de forma correcta al tener el sistema en un punto de operación diferente. Para resolver este problema se han usado diferentes fórmulas como lo son: control robusto, cambiar los parámetros del controlador según la zona de operación, etc.

Este trabajo de memoria pretende dar una visión diferente utilizando herramientas de inteligencia artificial, concretamente: control difuso, control con redes neuronales, y control neuro-difuso, y con ello determinar primero si su funcionamiento tiene validez y funciona, y segundo, determinar si funciona de mejor forma que un controlador clásico.

Como controlador clásico se tiene la parametrización utilizada recientemente por el CDEC para sus estudios de funcionamiento de un AGC teórico. Detalles de esta comparación pueden variar dependiendo de como varíen en la aplicación final y real del sistema AGC que implementarán finalmente.

Los datos y modelos de Digsilent han sido extraídos de estudios del CDEC. Los eventos utilizados para este trabajo también fueron extraídos del mismo, debido a que representan casos reales que son de interés para el comportamiento del sistema.

En este trabajo se seguirá una estructura simple y de menor a mayor. En la primera sección se tiene el marco teórico que dará conceptos básicos para entender el funcionamiento del AGC en un sistema de potencia, luego se dará una explicación de los detalles de cada uno de los controladores a implementar, además de consideraciones con ejemplos de modo de hacer más fácil entender estas herramientas que pueden no estar internalizadas. Posterior a ello, en el capítulo 3, se abordará la experiencia internacional, tanto del uso de otros países de esquemas de control automático de generación como de métodos de inteligencia artificial utilizados en sistemas de potencia. Luego en el capítulo 4, se tiene la implementación y consideraciones preliminares, donde se explica como se abordará el problema y con que herramientas de simulación, además de entregar un modelo funcional en las programas señalados antes de aplicarlo a un sistema real. En el capítulo 5 se tiene la implementación de los controladores diseñados en el sistema SING-SADI, se hablará de sus resultados y comparaciones descritas anteriormente. Finalmente en el capítulo 6 se harán conclusiones explicando que se logró con este trabajo, y si tiene sentido aplicar un esquema de control como el que se propone en un sistema real, además de señalar las ventajas funcionales con respecto a los controladores clásicos.

2

Marco Teórico

2.1. Funcionamiento del AGC y sus componentes

En los sistemas eléctricos de potencia habitualmente se habla de dos tipos de control:

- Control de Potencia Reactiva-Tensión
- Control de Potencia Activa-Frecuencia

Ambos siendo indispensables para el sistema, tanto para que sea estable, como para ofrecer cierta calidad del suministro de energía. Por lo que, mantener la tensión y frecuencia en rangos aceptables de operación es vital para las entidades que se encargan de controlar dichas variables.

Se considera que el control "Q-V", o control Potencia reactiva-tensión, es de carácter local, ya que el cambio de potencia reactiva en la barra sólo afecta la tensión de dicha barra y las cercanas a ella. Dicho esto, y agregando además que el sistema tiene un comportamiento previsible y fácil de modelar, este tipo de control no presenta mayores problemas. Las subestaciones que deseen implementar este tipo de control, pueden hacerlo mediante diferentes opciones como lo son: banco de condensadores, *Static VAR Compesator* (SVC), esquemas STATCOM, cambiadores automáticos de taps, control sobre la coriente de excitación del generador, etc.

Por otro lado, el control "P-f", o control Potencia Activa-frecuencia, es de carácter global, ya que es necesario llevar a cabo un balance de generación-consumo de todo el sistema.

Por lo dicho anteriormente, el control correcto sobre la frecuencia sólo puede ser llevado a cabo por un ente con acceso a información de potencia generada y consumida de todo el sistema. A su vez, el modelamiento y las expresiones que gobiernan este sistema son mucho más complicadas que las del control "Q-V".

Adicionalmente existe un acoplamiento entre estos dos controles, ya que los efectos del control dinámico sobre la tensión se ven reflejados posteriormente en el control sobre la frecuencia. Sin embargo, el control Q-V es lo suficientemente rápido comparado con el P-f, tal que no se afectan el uno del otro, por lo que se pueden considerar como controles separados.

Debido a todas estas consideraciones, un control sobre la frecuencia puede ser muy complicado. Para hacer un buen diseño del controlador es necesario tener un buen conocimiento del funcionamiento de la *planta* a controlar, por lo que en las siguientes secciones se pretende presentar los fundamentos del comportamiento de la frecuencia en los sistemas de potencia. Para ello, todos los modelamientos serán ilustrados adicionalmente con simulaciones en Simulink.

2.1.1. Modelamiento del generador

Un generador básicamente es una máquina giratoria que permite la conversión electromecánica. Esto gracias a un torque que se aplique en su eje giratorio.

Por otro lado, la circulación de corriente por los bornes de la máquina genera un torque contrario al que está impulsando el generador, por lo que considerando la expresión de equilibrio de torque sobre el eje, se tendría entonces la ecuación (2.1):

$$T_m - T_e = \frac{d\omega_r}{dt} J \tag{2.1}$$

Donde:

$$T_m: \text{ Torque mecánico.}$$

$$T_e: \text{ Torque eléctrico.}$$

$$\omega_r: \text{ Velocidad angular.}$$

$$J: \text{ Inercia angular.}$$

La cual, utilizando las variables:

$$M: \text{ Momento angular de la máquina en p.u.}$$

Puede ser reescrita como se visualiza en la ecuación (2.2):

$$T_m - T_e = M\frac{\omega_r}{dt} \tag{2.2}$$

Si se recuerda que la potencia está ligada al torque de la forma:

$$P = \omega_r T \tag{2.3}$$

Se puede trabajar en la expresión que se obtiene como respuesta del generador ante un cambio de carga:

$$P_o + \Delta P = (\omega_0 + \Delta\omega_r)(T_0 + \Delta T) \tag{2.4}$$

$$\Delta P_m - \Delta P_e = \omega_0(\Delta T_m - \Delta T_e) + (T_{m0} - T_{e0})\Delta\omega_r \tag{2.5}$$

Considerando que el punto inicial es un punto estable, se tiene que:

$$T_{m0} = T_{e0} \tag{2.6}$$

$$\omega_0 = 1 \tag{2.7}$$

Con lo que la expresión (2.5) quedaría:

$$\Delta P_m - \Delta P_e = \Delta T_m - \Delta T_e \tag{2.8}$$

$$\Delta P_m - \Delta P_e = Ms\Delta\omega \tag{2.9}$$

Donde:

$$\Delta P_m: \text{ Variación de potencia mecánica.}$$

$$\Delta P_e: \text{ Variación de potencia eléctrica.}$$

Siendo la ecuación (2.9), la más utilizada en los recursos literarios, y que queda representada en diagrama de bloques como se observa en la Figura 2.1.

Con ello, se puede decir que al tener igualdad entre potencia mecánica y eléctrica, no hay cambios en la velocidad a la que gira el eje. Mientras que al haber diferencias se produciría una aceleración tal que aceleraría o desaceleraría el sistema dependiendo del signo, o balance de torques resultantes.

De este sistema, se puede concluir que una perturbación, o sea, un cambio de la potencia mecánica o potencia eléctrica, generaría una variación acumulativa de la frecuencia constante durante el tiempo, con lo que nunca llegaría a un estado estable.

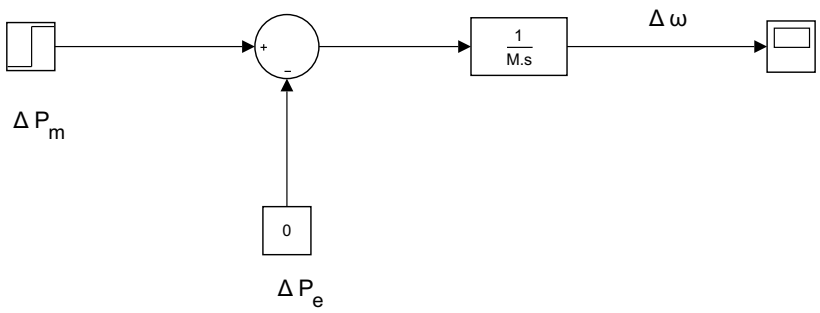


Figura 2.1: Modelo del generador utilizado en Simulink.

2.1.2. Modelamiento de la carga

Hasta ahora se ha considerado a la carga como inmutable ante cambios de frecuencia, lo cual ocurre con cargas puramente resistivas. Si se considera un modelo de la carga mas cercano a la realidad, se tiene que parte de la carga cambia ante variaciones de frecuencia.

Para modelar dicha variación, se designa un parámetro D que determina el cociente entre un cambio de demanda y la desviación de frecuencia provocada por dicho cambio.

$$D = \frac{\Delta P_L}{\Delta f}[p.u.] \tag{2.10}$$

Por ello, la expresión para la demanda se reescribe para incorporar la variación de la carga ante la frecuencia:

$$\Delta P_e = \Delta P_L + D\Delta f \tag{2.11}$$

Que siendo reemplazada en la expresión 2.9, da por consiguiente:

$$\Delta P_m - (\Delta P_L + D\Delta\omega) = Ms\Delta\omega \tag{2.12}$$

$$\Delta P_m - \Delta P_L = Ms\Delta\omega + D\Delta\omega \tag{2.13}$$

$$\Delta\omega = \frac{\Delta P_m - \Delta P_L}{Ms + D} \tag{2.14}$$

Siendo (2.14) la ecuación más utilizada en los estudios técnicos en cuanto a respuesta dinámica de los generadores. Ésta es representada mediante diagramas de bloques en Simulink en la Figura 2.2, de modo de obtener su respuesta ante un escalón de potencia de un 10%, y de D con un valor de 2% (un valor común de D [1]), la cual puede ser visualizada en la Figura 2.3. En ella se puede observar que no hay un cambio permanente como ocurría en el caso anterior, sino que tiende a un valor final, que puede ser calculado de forma simple usando el teorema del valor final:

$$\lim_{t\to\infty} \Delta\omega(t) = \lim_{s\to 0} s\Delta\omega(s) = \lim_{s\to 0} s \cdot \frac{(\Delta P_m - \Delta P_L)}{s}\frac{1}{Ms + D} = \frac{(\Delta P_m - \Delta P_L)}{D} \tag{2.15}$$

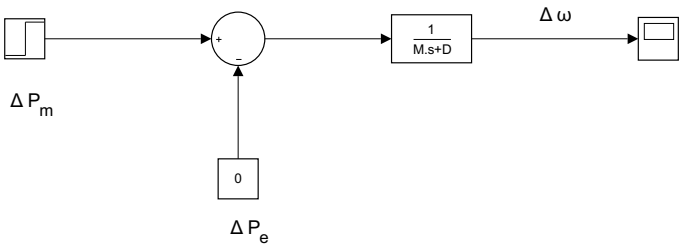


Figura 2.2: Modelo del generador utilizado en Simulink.

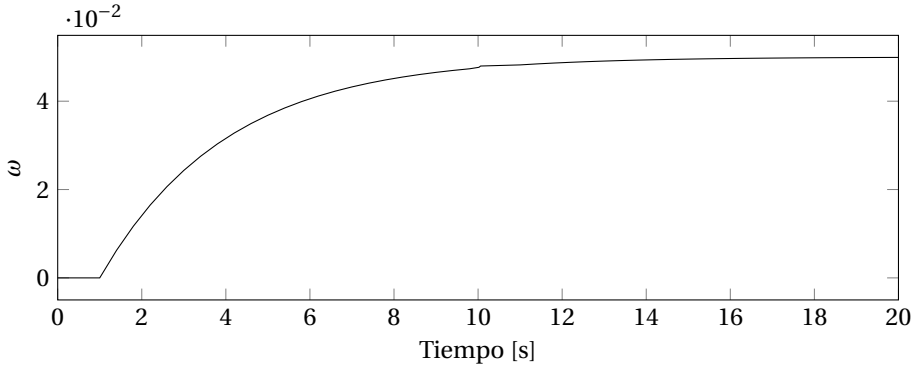


Figura 2.3: Respuesta a escalón, de un sistema de generador simple con amortiguamiento.

Como se pudo visualizar en la Figura 2.3, aunque la desviación de frecuencia se detiene, y tiende a un valor final, ésta es tan grande que no puede ser considerado permisivo en ningún sistema eléctrico (la frecuencia varió en un 5 %).

2.1.3. Control primario de frecuencia (CPF)

Debido a lo comentado anteriormente, un esquema de control que actúe sobre la potencia generada de una central es necesario. Para ello, dentro de las centrales es usado un regulador que cambia los parámetros de generación dependiendo de la variación de frecuencia. Por lo tanto dicho esquema vería una variación de frecuencia y definiría un cambio en la potencia mecánica que debería ser inyectada en el eje (ya sea abriendo las compuertas de agua o vapor, o entregando mayor combustible a la turbina). Dicho esquema puede ser visualizado en la Figura 2.4.

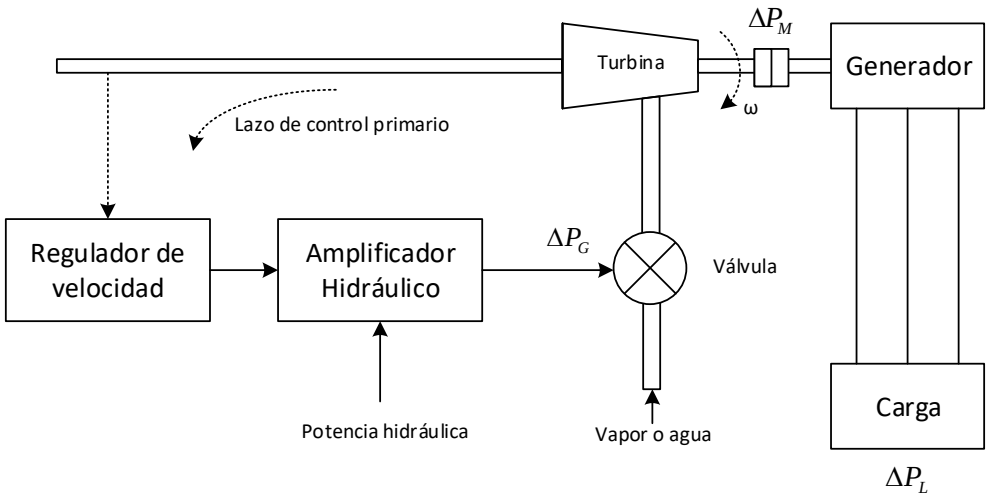


Figura 2.4: Esquema de generación con el uso de reguladores de velocidad.

Dichos modelos por lo general consideran retrasos, ya que no funcionan de forma inmediata en la realidad debido a los movimientos mecánicos necesarios para llevar a cabo dichos cambios en los posicionamientos de las válvulas. Luego, la posición de la válvula será dependiente de la desviación de frecuencia, lo cual llevado a diagrama de bloques puede ser representado en la Figura 2.5.

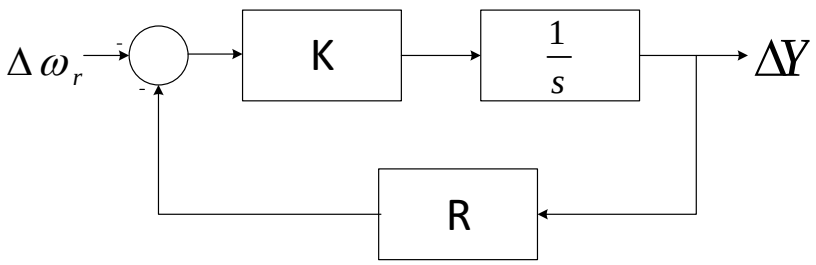


Figura 2.5: Diagrama de bloques de la posición de la válvula, ahora dependiente de la desviación de frecuencia.

Posteriormente, haciendo operaciones de diagramas de bloque, se puede representar de forma reducida como se muestra en la Figura 2.6.

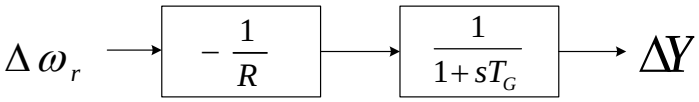


Figura 2.6: Diagrama de bloques de la posición de la válvula, ahora dependiente de la desviación de frecuencia, de forma reducida.

Donde T_G representa la constante de tiempo de la turbina, que está definido como:

$$T_G = \frac{1}{KR}$$

Si se aplica el teorema del valor final para determinar el valor en estado estacionario al ocupar este regulador se tiene que la posición de la válvula cambia de la forma:

$$\lim_{t\to\infty} \Delta Y(t) = \lim_{s\to 0} \Delta Y(s) = -\frac{1}{R}\Delta\omega_r \tag{2.16}$$

Donde a R se le define formalmente como estatismo, y es representativo de cada planta generadora. Con este valor se representa cuanto cambia la potencia generada ante un cambio en la desviación de frecuencia.

Dicho cambio, en estado estacionario, está definido formalmente como:

$$R = \frac{\Delta f\,\%}{\Delta P\,\%} \cdot 100 = \frac{\Delta f}{f_n} \cdot \frac{P_n}{\Delta P} \cdot 100 \tag{2.17}$$

Si se gráfica el cambio de frecuencia en relación a la potencia de salida de un generador, se tendría un gráfico como el de la Figura 2.7, donde el estatismo R es la pendiente de dicha gráfica. En ella se observa que al haber un cambio en la frecuencia, desde f_0 a f_1, la unidad generadora aumenta su generación, pasando de P_0 a P_1.

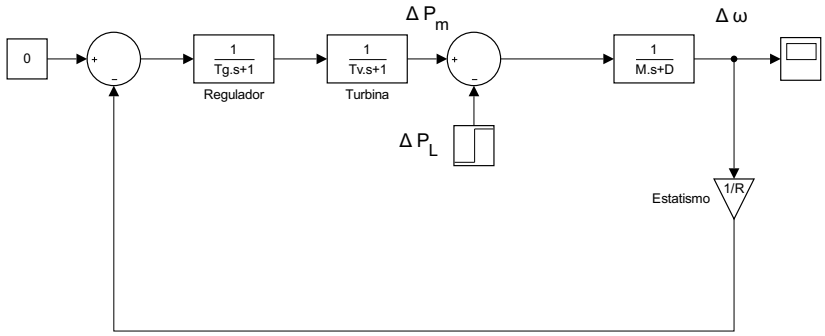


Figura 2.8: Diagrama de bloques de una planta con control primario de frecuencia.

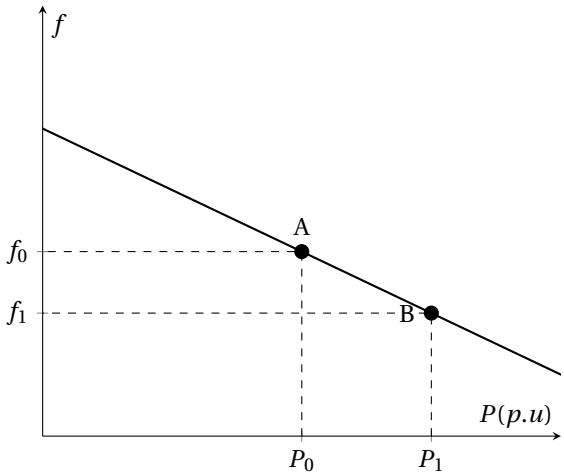


Figura 2.7: Curva característica de la máquina generadora, considerando diferentes valores de potencia de referencia.

Considerando todos los cambios que se le ha hecho al modelo, se podría generar un nuevo diagrama de bloque que considere el comportamiento de una planta generadora. Para ello, se tiene por ejemplo la Figura 2.8, que representa el modelo de una planta generadora con máquinas térmicas sin recalentamiento. En ella, se adicionada el bloque de la turbina, que representa la respuesta del sistema ante la variación de la posición de la válvula, y que tiene relación con el tipo de planta que se tenga, y con sus constantes mecánicas correspondientes.

Se debe tener en consideración que hay variados modelos de reguladores y turbinas dependiendo del tipo de planta. No es objetivo de este estudio entrar en detalle de dichos modelos, pero cabe destacar que todas las plantas se comportan diferente a la hora de variar sus parámetros de generación.

Dicho esto, la respuesta ante un escalón de balance de potencia de dicho sistema podría visualizarse en la figura 2.9, donde se observa que la frecuencia en estado estable (al final del tiempo de simulación) es de:

$$f = 49{,}7935[Hz] \tag{2.18}$$

Para dicha simulación se tomaron en cuenta las siguientes constantes, que son valores típicos [1]:

T_g	T_v	R	ΔP_L	D
0.25	0.25	0.045	0.1	2

Tabla 2.1: Datos usados para la simulación del caso de planta generadora térmica sin recalentamiento.

Con lo que se tiene una desviación de frecuencia en estado estable menor que en el caso anterior, pero sigue siendo necesario llevarlo a su valor nominal.

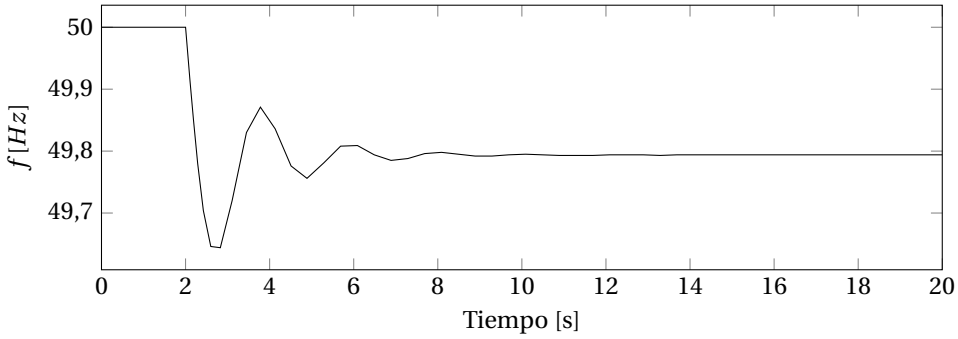


Figura 2.9: Respuesta a escalón, de una planta generadora térmica sin recalentamiento con control primario de frecuencia.

2.1.4. Potencia de referencia

Debido a lo comentado anteriormente, el control primario de frecuencia tiene como finalidad hacer estables las desviaciones de frecuencia. Como resultado de esto, se tiene una desviación de frecuencia permanente que necesita ser corregida.

Para ello es necesario definir nuevos niveles de generación en las máquinas generadoras, de modo de equilibrar el desbalance de potencias. Esto se realiza enviando diferentes consignas de potencia a las centrales generadoras, que les indican precisamente que cambien su generación. En algunos países ese proceso se realiza de forma manual, mientras en otros se ocupa un ente central que realiza las correcciones de forma automática, gracias a un controlador que corrige el error.

Para hacer dicha corrección en la generación de la central, se puede hacer una modificación al diagrama de bloques, que puede ser visualizado en la Figura 2.10 de modo de agregar una variable de control que se llama potencia de referencia o consigna de potencia.

Dicho valor, en p.u., le indica a la unidad generadora en cuanto debe cambiar su actual nivel de generación.

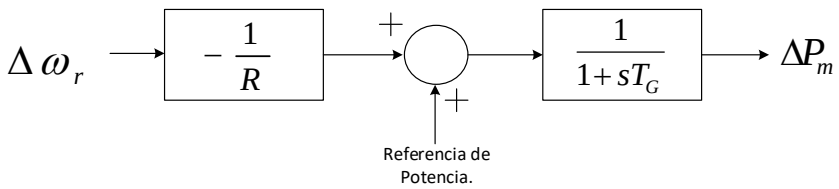


Figura 2.10: Diagrama de bloques modificado de modo de considerar la potencia de referencia, o consigna de potencia.

Con ello, el punto de la curva hacia donde se desplazaba por efecto del estatismo propio de la máquina en estado estable, puede ser manipulado de modo de corregir el estado permanente en el que ha quedado. Si se considera por ejemplo la Figura 2.11, en un punto de equilibrio inicial A con una potencia P_0, donde el sistema está con frecuencia nominal f_0, al ocurrir un cambio en el balance tal que la potencia se desplaza hacia el punto P_1, ocurrirá que en estado estable la frecuencia bajará, situándose como nuevo punto de equilibrio el punto B. Al incrementar el valor de la consigna de potencia, se puede desplazar el punto de forma vertical, de modo de llegar a la situación descrita en el punto C, donde a una potencia diferente se tiene nuevamente frecuencia nominal. Las curvas del gráfico dibujadas con líneas punteadas y continua, representan la misma unidad generadora, con el mismo estatismo, pero con diferente valor de potencia de referencia o consigna de potencia.

2.1.5. Control Secundario de frecuencia (CSP)

Como se ha comentado anteriormente, al ocurrir una diferencia en el balance de potencia, se produce un cambio de frecuencia en estado estable, que luego debe ser corregido modificando la consigna de potencia. Para hacer este proceso de forma automática es necesario modificar el valor conforme a cierto error, el cual viene condicionado claramente con la diferencia de frecuencia con respecto a la nominal. Por lo tanto, si se considera un controlador simple de modo integrador, donde se integra el error de frecuencia se podría

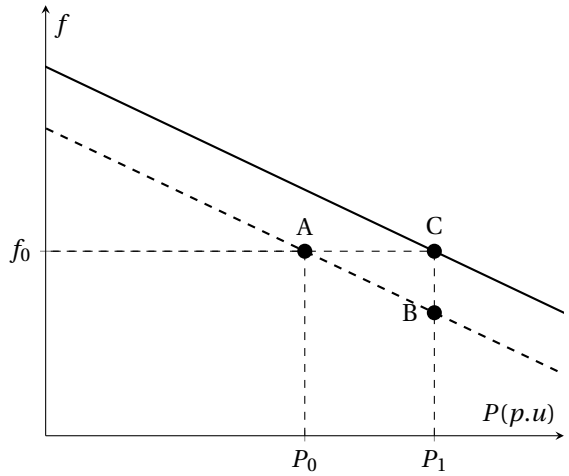


Figura 2.11: Curva característica de la máquina generadora, considerando diferentes valores de potencia de referencia.

tener un esquema como el de la Figura 2.12. En ella se visualizan dos tipos de unidades, una con regulación primaria y secundaria, y otra solamente con primaria. Además se observa un valor negativo al integrador, esto es debido a la referencia tomada: un balance negativo de potencia (demanda mayor a la potencia generada), requiere que se suba la potencia generada (signo contrario o negativo).

2.1.6. Control Automático de generación en un sistema con dos áreas

Si se tienen dos áreas conectadas por una línea dada, se pueden considerar los efectos condensados sobre dos áreas de modo de tener potencias mecánicas y demandas eléctricas P_{m1} y P_{m2}, P_{l1} P_{l2} equivalentes de las mismas. Adicionalmente se tiene una potencia que circula entre las dos áreas mencionadas P_{12}, donde el subíndice indica que la potencia circula desde el área 1 hacia el área 2.

Por lo tanto al ocurrir un desbalance de potencias en el área 1 (debido a un cambio en la demanda) se tendrá que el efecto producido en el área 1 se expresa:

$$\Delta P_{m1} - \Delta P_{12} - \Delta P_{l1} = D_1 \Delta\omega \tag{2.19}$$

Mientras que en el área 2 se tendría:

$$\Delta P_{m2} + \Delta P_{12} = D_2 \Delta\omega \tag{2.20}$$

Pero como las variaciones de potencia mecánica están controladas por el efecto del estatismo, pueden ser expresadas en función de los mismos, por lo tanto las expresiones anteriores pueden ser desarrolladas como se presentan en las ecuaciones (2.21) y (2.22).

$$\Delta\omega(\frac{1}{R_1} + D_1) = -\Delta P_{12} - \Delta P_{l1} \tag{2.21}$$

$$\Delta\omega(\frac{1}{R_2} + D_2) = \Delta P_{12} \tag{2.22}$$

Si se desarrolla para la potencia transferida entre áreas se tiene:

$$\Delta P_{12} = \frac{-\Delta P_{l1}(\frac{1}{R_2} + D_2)}{\frac{1}{R_1} + D_1 + \frac{1}{R_2} + D_2} \tag{2.23}$$

Donde se pueden hacer las consideraciones:

$$\beta_1 = \frac{1}{R_1} + D_1 \tag{2.24}$$

y

$$\beta_2 = \frac{1}{R_2} + D_2 \tag{2.25}$$

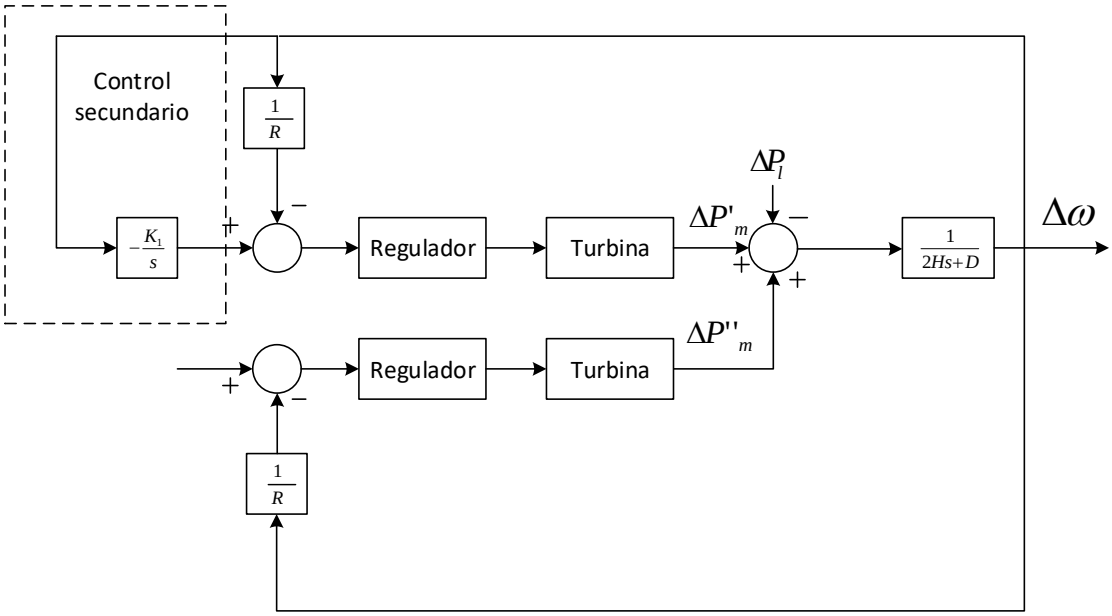


Figura 2.12: Esquema básico de un sistema con unidades con regulación secundaria de frecuencia, y unidades con regulación primaria únicamente.

Con esto se puede concluir que un cambio en la demanda del área 1, implica un cambio en la potencia transferida desde el área 1 hacia el área 2, lo cual significa que los cambios de demanda no son controlados completamente por la generación del sector propio donde se originó, sino que parte de esa generación se produce en otro sistema aledaño. Esta situación generalmente es indeseable, ya que podría estar en valores para los que no fue concebida dicha línea, o por acercarse demasiado a puntos de operación peligrosos para el sistema (sobrecargar la línea de transmisión), por lo que es necesario hacer una corrección sobre dicho valor.

Para ello, se hace un cambio en la concepción del error que se tenía antes. Ahora, además de considerar la desviación de frecuencia, se adiciona la desviación de potencia de intercambio al error. Dichas expresiones reciben el nombre de ACE (Area Control Error), y se podrían definir como se observa en las ecuaciones 2.26 y 2.27, que son las expresiones utilizadas para el área 1 y el área 2 correspondientemente:

$$ACE_1 = \Delta P_{12} + B_1 \Delta\omega \tag{2.26}$$

$$ACE_2 = -\Delta P_{12} + B_2 \Delta\omega \tag{2.27}$$

Donde B es una constante llamada factor bias de frecuencia. En teoría, cualquier valor de B sirve para el control secundario que se hará a continuación, ya que el objetivo de dicho control es llevar a cero la expresión de error en estado estable, independiente de que valor se haya escogido para la constante. Aún así, una correcta elección de B permite un mejor control en la dinámica del sistema, ya que se tiene una mejor percepción de que ocurre en el sistema (si se elige un valor muy bajo de B, el control inicialmente se empeña mayormente en controlar la potencia de intercambio, así como ocurre a la inversa). Una forma de poder determinar un valor correcto- de B es analizar la respuesta natural del sistema ante disturbios reales ocurridos. También en diversos recursos en la literatura ante la falta del anális anterior, recomiendan un valor típico de B, que dependa de las variables internas del sistema como su estatismo o constante de amortiguamiento:

$$B = \frac{1}{R} + D = \beta \tag{2.28}$$

Si se toman todas las consideraciones y expresiones mencionadas anteriormente, y son llevadas a un esquema de bloques, pueden ser resumidas en la Figura 2.13. Donde se puede observar en un primer plano

el sistema solamente, igual al visto anteriormente, pero con la diferencia que al balance de potencia se le adiciona la potencia de intercambio. Luego, dentro del cuadro punteado, se puede visualizar que se adiciona el sistema de control en las potencias de referencia de las dos áreas, de modo que cada una pueda controlar el error de área propio (ACE).

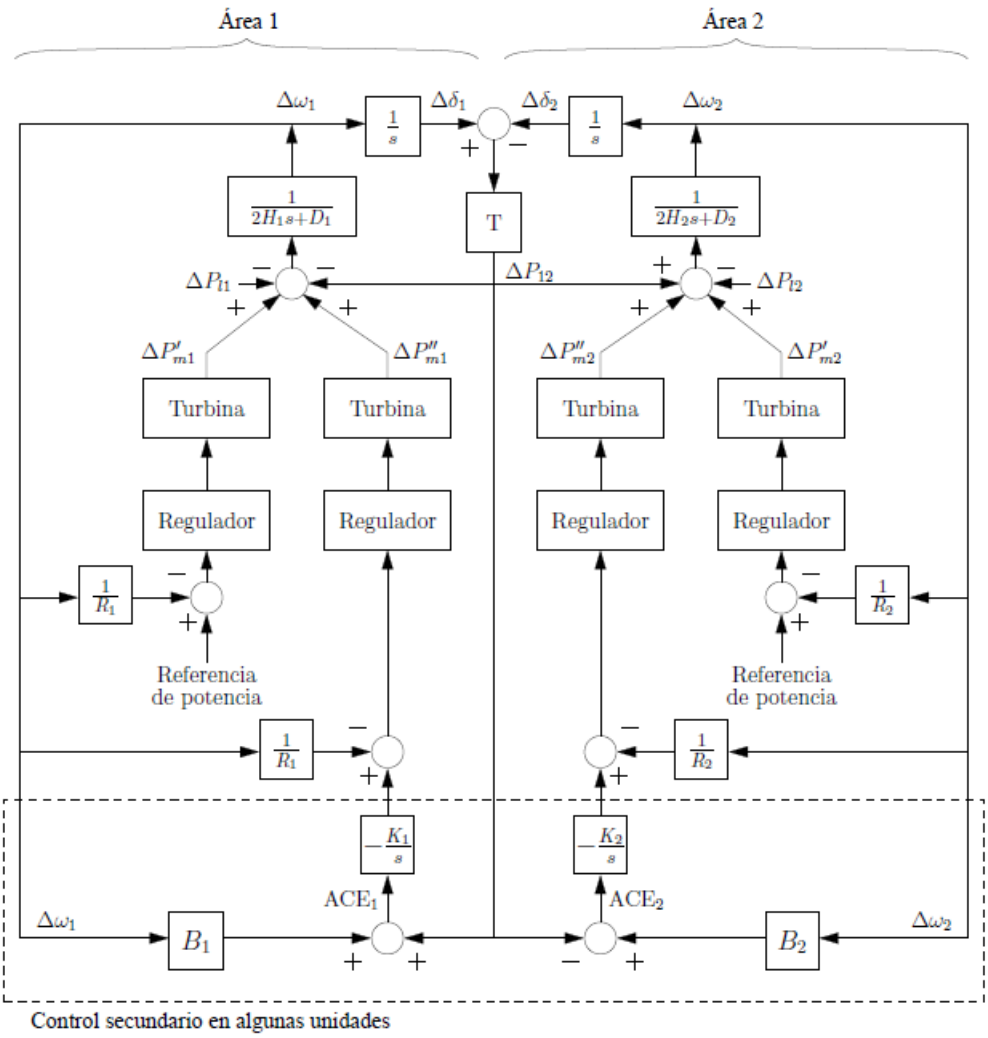


Figura 2.13: Esquema de un sistema eléctrico con dos áreas, adicionando el sistema de control secundario.

2.1.7. Factor de participación

Los factores de participación representan en porcentaje, cuanta de la generación adicional o reducción de generación, debe aportar cada una de las centrales que participen en el control secundario de frecuencia. Por lo mismo se considera entonces, que la suma de todos los factores de participación es igual a 1:

$$\sum_{k=1}^{n} \alpha_k = 1 \tag{2.29}$$

Siendo:

α_k : Factor de participación de la unidad k.

En un ambiente competitivo, los factores de participación se calculan en tiempo real, considerando sus precios de generación, disponibilidad, consideraciones en cuanto a congestión, límites de generación, entre otros.

2.1.8. Índice de calidad de control

De forma de poder determinar si el control está siendo efectivo o no, es necesario condensar su información dentro de algún valor medible. Para ello se han creado variados mecanismos de medición de la calidad del control.

Según la norma técnica, los factores de índices de calidad usados son:

Factor FECF:

Definido en el artículo 5-67 de la NTSyCS, es un factor que se determina de forma horaria y está definido como:

$$FECF(k) = 1 - ||\frac{\Delta F^*_{Max}(k)}{\Delta F_{Max}}|| \tag{2.30}$$

Donde $\Delta F^*_{Max}(k)$ es la desviación máxima instantánea del valor de frecuencia, mientras que ΔF_{Max} es la desviación máxima de frecuencia en estado permanente, una vez que se agota toda la reserva asociada al control primario de frecuencia.

Límites aceptables de desviación de frecuencia

De acuerdo al artículo 5-30 de la NTSyCS, la frecuencia del sistema debe estar contenida dentro de ciertos límites. Estos en el SING son: 49.8[Hz] y 50.2[Hz] el 97% del tiempo. Estas mediciones deben ser hechas durante 7 días seguidos, con una extracción de información de la frecuencia cada 10 segundos.

Integral del ACE

Es un indicador que ha sido usado en bastantes estudios [2], ya que es fácil de implementar a las optimizaciones. Está basado en la integración del error al cuadrado (ISE). Con lo que está definido formalmente:

$$I_{ACE} = \int_0^T (ACE)^2 dt \tag{2.31}$$

Que de forma discreta se puede representar de la siguiente forma:

$$I_{ACE} = \Delta t \sum_{i=1}^{N} ACE_i \tag{2.32}$$

En este estudio se utilizará una modificación de este parámetro, de modo de resumir la efectividad del controlador y compararlos. La diferencia radica en que se utiliza la integral del ACE^2, de modo de eliminar el efecto del signo, y en vez de ser multiplicado por el paso de tiempo, se divide por el número total de muestras N en la simulación:

$$I_{ACE} = \frac{\sum_{i=1}^{N} ACE_i^2}{N} \tag{2.33}$$

3

Experiencia internacional

El propósito de este capítulo es relatar las principales experiencias al rededor del mundo al utilizar las tecnologías descritas en este trabajo.

En una primera parte se hará una descripción de cómo se realiza la regulación de frecuencia en un sistema grande como lo es Norteamérica, de modo de entender como han enfrentado otras naciones el problema.

En la segunda parte se hará una revisión de la implementación de métodos de inteligencia artificial. Debido a que lo propuesto específicamente en este trabajo no ha sido aplicado en ningún sistema real (controladores inteligentes en regulación de frecuencia), se comentará sobre el uso de inteligencia artificial dentro de los sistemas de potencias, donde se destacan fuertemente en los casos que tengan que ver con optimización.

3.1. Control secundario de frecuencia en Norteamérica

En norteamérica (USA y parte de Canadá), el sistema está conformado por 4 grandes subsistemas: *Western Interconnection, Texas Interconnection, Eastern Interconnection y Quebec Interconnection* (su posición geográfica y las demandas energéticas de los sistemas de estados unidos pueden ser visualizados en la Figura 3.1).

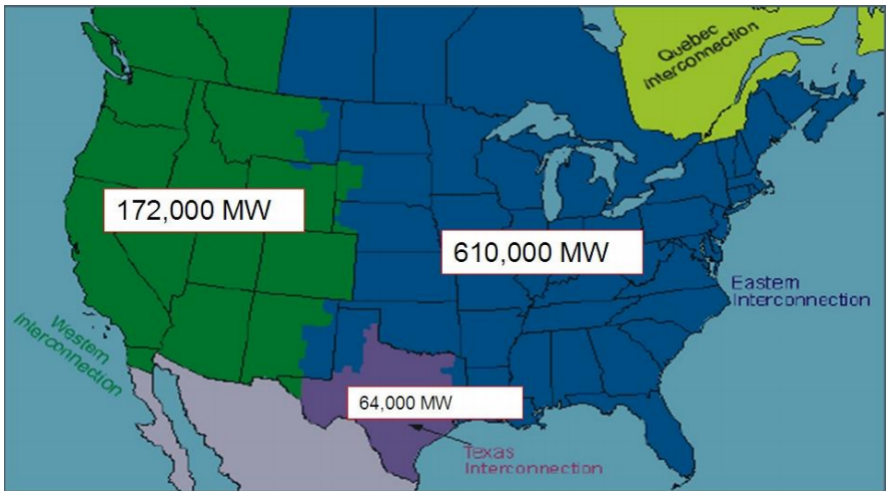


Figura 3.1: Imagen donde se observa la posición geográfica de cada uno de los sistemas en norteamérica.

El balance entre generación y carga dentro de estas interconexiones es manejado por entidades llamadas *Balancing Authorities*. Estas entidades manejan la potencia de despacho de los generadores, de modo de cumplir con las necesidades de la zona que tienen a cargo. Algunas de éstas también tienen control sobre la carga.

En la Figura 3.2 se muestra por ejemplo, la distribución geográfica de las Balancing Authorities en la interconexión Oeste de Estados Unidos.

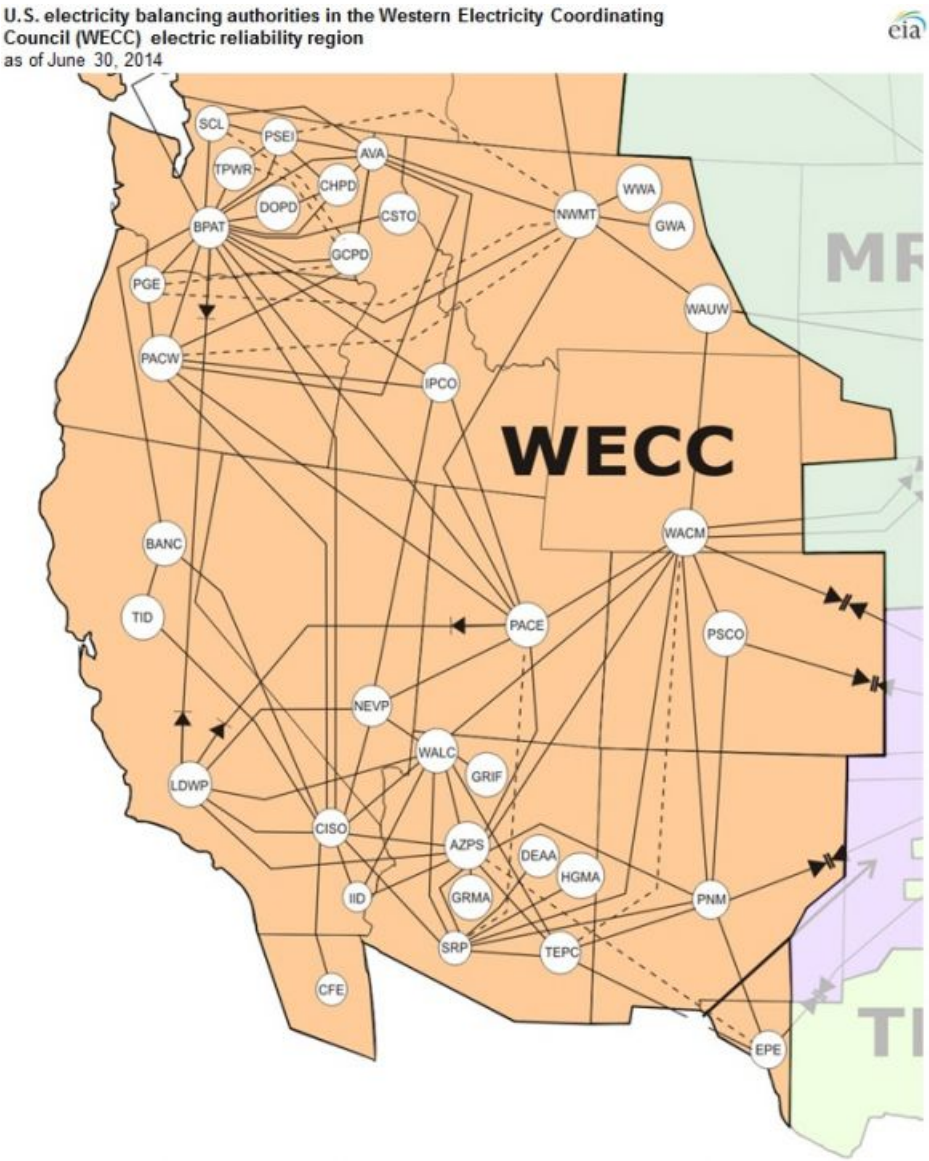


Figura 3.2: Mapa de las Balancing Authorities de Western interconnection

Hay más de 100 Balancing Authorities en Norteamérica interactuando con su par más cercano todo el tiempo.

Vigilando las Balancing Authorities existen operadores que reciben el nombre de *Reliability Coordinators*.

Cada Balancing Authority usa medidores en las interconexiones con sus vecinos, de modo de poder tener control y contabilidad de la energía que se traspasa entre subsistemas. Gracias a esto, los operadores pueden comprar o vender energía entre dichos subsistemas de modo de poder balancear de forma correcta la generación con la demanda.

Los operadores o despachadores de cada una de las Balancing Authorities, tienen como responsabilidad monitorear el ACE y mantenerlo dentro de los límites considerados para ese subsistema. Este balance se logra gracias a la combinación del ajuste del AGC, llamadas de teléfono a las plantas de generación, y si es necesario, también usando acciones de emergencia como lo son los esquemas de desconexión de carga.

3.2. Implementación de métodos de inteligencia artificial en SEP

En los últimos años se ha destinado una mayor atención a los métodos que se basan en inteligencia artificial. Han surgido numerosos papers o artículos, que son mencionados en algunos libros [3] que sugieren sus aplicaciones a diversas áreas de los sistemas de potencia.

Aunque la mayoría de los métodos aplicados aún están en una etapa de experimentación, hay países tales como Japón que ya aplican esquemas basados en inteligencia artificial, en varias de sus áreas.

Los esquemas tradicionales, tanto de operación y control de sistemas de potencia, planificación, y otros, están basados en conocimiento humano y modelos matemáticos. El principal problema con estos esquemas tradicionales, es que en la práctica, los sistemas de potencia contienen muchas variables que agregan incertidumbre al modelo. Para responder a dichas incertidumbres es necesario hacer ciertas consideraciones, lo cual aleja al modelo de la realidad, y lo limita fuertemente dependiendo de la consideración.

La implementación de inteligencia artificial para resolver estos problemas utilizando, por ejemplo, sistemas difusos, redes neuronales artificiales, algoritmos genéticos, búsqueda Tabú, etc, han demostrado dar buenos resultados.

El propósito de esta sección es señalar qué temas y tecnologías han sido implementadas en la realidad, demostrando que pudieron entregar un funcionamiento mejor que con el uso de esquemas tradicionales.

La tabla 3.1 da un pequeño resumen, sobre algunos de los aspectos eléctricos donde son un aporte fundamental.

Área de aplicación	**Método de inteligencia artificial**
Planificación de generación	Sistemas expertos Algoritmos genéticos Búsqueda Tabú
Predicción de demanda	Redes Neuronales Sistemas de inferencia difusa
Unit Commitment	Búsqueda Tabú
Diagnóstico	Sistemas expertos Redes Neuronales
Controladores PSS	Controladores con lógica difusa

Tabla 3.1: Aplicaciones

3.2.1. Planificación de expansión de sistemas de distribución

La implementación en este tipo de problemas va relacionado con la optimización. Para ello diversas técnicas de inteligencia artificial son ocupadas para resolver distintos escenarios. Las principales ventajas de estos modelos son:

- Expresar incertidumbres.
- Lograr planificación robusta y flexible.
- Métodos de optimización multi-objetivo.

Una tabla con los escenarios más ocupados de este tipo (resueltos con métodos inteligentes), es presentada en la tabla 3.2

Problema	**Técnicas de Inteligencia Artificial**
Optimización de tensión y corriente	Búsqueda Tabú
Minimización de pérdidas de distribución	Búsqueda Tabú
Optimización de la ubicación de equipos	Algoritmo Genético
Simulación de recuperación del sistema	Búsqueda tabú

Tabla 3.2: Tabla con las técnicas de inteligencia artificial usadas para los problemas de expansión en sistemas de distribución.

3.2.2. Predicción de la demanda

Métodos de inteligencia artificial para la predicción de la demanda han sido ocupados, por ejemplo, en Japón. Para ello se han ocupado redes neuronales artificiales, donde para entrenarlos, se ha ocupado infor-

mación tanto como datos de demanda máxima, demanda de potencia horaria, como datos metereológicos, considerando además la estación del año donde se encuentre.

El sistema en concreto funciona de la siguiente manera:

- La demanda máxima es predicha por el sistema con un día de antelación basado en la predicción del clima, incluyendo la temperatura máxima y mínima. La demanda máxima de potencia de un día y de una semana atras también es utilizada para estimar la demanda máxima de potencia del dia siguiente.
- Utilizando la predicción de la demanda máxima obtenida en el paso 1, se hace una estimación de la demanda horaria.

Los datos hasta ahora han sido bastante consistenes, y representan una buena estimación de como será la demanda para un día dado, logrando un error en promedio anual cercano al 1.5 %, por lo que la precisión de este modelamiento es relativamente alta.

3.2.3. Unit Commitment

En Japón, el problema de unit commitment con solución basada en inteligencia artificial, ha sido utilizado para un grupo de generadores hidroeléctricos. La tecnología en sí ha ocupado sistemas de reglas específicas y esquemas de búsqueda inteligentes.

El problema de Unit Commitment para estos casos es resuelto determinando los niveles óptimos de agua en las represas, el flujo estimado de agua, etc. La operación esperada sería ocupar estos volúmenes de agua cuando los costos de generación son demasiado altos.

La ventaja del uso de algoritmos inteligentes como el de búsqueda tabú, redes neuronales, enfriamiento simulado -*simulated annealing* - u otros, es que permiten la optimización para casos más complejos como los de planificación de mantenimiento con más de 4000 casos al año. Además de tener en consideración que el problema de Unit Commitment es no lineal y con variables binarias o discretas, lo cual hace que el uso de métodos mas tradicionales sean insuficientes para optimizar.

3.2.4. Diagnóstico de fallas

Siendo una tecnología también implementada en Japón, se usa para ayudar a los operadores humanos.

El sistema se compone de dos partes: una utiliza un sistema experto de reglas para la clasificación de los tipos de fallas, y otro que usa redes neuronales para asociar esas fallas con ciertas probabilidades de ocurrencia.

En la tabla 3.3 se puede observar la precisión de dicho método. A simple vista parece que para ciertas fallas su precisión es prácticamente perfecta, mientras que para otros no es muy bueno. Por lo tanto, hasta ahora, no es viable dejar este sistema de forma autónoma, sino que sea una herramienta de soporte para los operadores humanos.

Tipo de falla	**Identificación Correcta**	**Identificación Incorrecta**	Precisión (%)
Rayos	65	0	100
Maquinaria de construcción	3	2	60
Ramas de árboles	15	0	100
Animales pequeños	10	3	77

Tabla 3.3: Tabla la precisión de identificación de fallas usando el método descrito.

3.2.5. Control de estabilización

Durante largo tiempo, muchos métodos de optimización, técnicas robustas, además de sistemas expertos e inteligentes han sido utilizados en los estabilizadores de potencia para mejorar el rendimiento y funcionalidades de un sistema de potencia dado.

Entre todas las metodologías utilizadas, los sistemas basados en lógica difusa han sido los que han llamado más la atención, debido a que funcionan en un rango de operación mucho más robustas. Como prototipo, en junio de 1997 se ocupó por primera vez en una unidad hidroeléctrica para la estabilización del sistema de potencia (PSS) utilizando un computador personal. Luego en mayo de 1999, este controlador fue reemplazado y mejorado. Dicho equipo ha funcionado muy bien por más de 10 años, incluyendo la etapa de prototipo.

Se han propuesto diversas aplicaciones donde podría funcionar un controlador basado en lógica difusa, pero no han sido implementadas. Dichas aplicaciones son:

- Control de oscilaciones.
- Control de dispositivos FACTS.
- Control de voltaje y reactivos.

Además del control automático de generación discutido en esta tesis.

3.3. Controladores Inteligentes

El análisis de funcionamiento del control automático de generación ha sido estudiado de forma profunda a través del tiempo, y la literatura es más bien voluminosa en cuanto a este tema. Por lo mismo, los esquemas de AGC han ido evolucionando con el tiempo, además de enfrentarse a nuevos desafíos que impone el crecimiento de los sistemas eléctricos, restringido claramente, a cada región según como sea su matriz energética. Además de tener en consideración, que han sido incorporados sistemas que presentan incertidumbre en su funcionamiento tales como generación eólica, celdas fotovoltáicas, sistemas de almacenamiento de energía, etc. Las mas recientes visiones en cuanto a como enfrentar estos desafíos consideran el uso de control inteligente, utilizando por ejemplo redes neuronales, lógica difusa, algoritmos genéticos, técnicas heurísticas de optimización, etc.[3]

3.3.1. Lógica difusa con AGC

La principal ventaja de la lógica difusa es que no se basa en el modelamiento matemático complejo de un sistema para funcionar, sino que usa el conocimiento experto de un operador para determinar su lógica de funcionamiento.

Esto en términos de esfuerzo para comprender el sistema representa un logro gigantesco, ya que se ahorra el paso de tener que modelar un sistema, o enfrentarse a las no linealidades del mismo. Por lo mismo, han sido introducidos en el control de una gran variedad de sistemas.

Muchos sistemas de control basados en lógica difusa han sido presentados aplicados al AGC. Donde varían mucho en la estructura, cantidad de funciones de pertenencia, motor de inferencia, y métodos de defusificación.

3.3.2. Control con redes neuronales y Neuro-Difuso en el AGC

Las redes neuronales propiamente tal, funcionan como un estimador universal. Para ello, se le dice que comportamiento es el correcto y cual no, de modo de poder ajustar sus parámetros de forma que genere una mejor respuesta y con ello se tenga un mejor comportamiento del sistema controlado.

Actualmente es muy utilizado en sistemas de reconocimiento, predicción, etc. Pero en sistemas de control también tiene aplicaciones, ya que tiene la gran ventaja de poder variar sus parámetros según reglas especificadas de modo de lograr ciertos objetivos.

Por otro lado, de modo de utilizar estas ventajas en el ajuste de parámetros de lógica difusa, se desarrollaron métodos de control neuro-difuso. En él se condensan muchas de las ventajas de ambos sistemas.

Sin embargo, al igual que en el caso de la lógica difusa, estos esquemas de control no han sido aplicados a sistemas reales, y solo han sido tema de estudio.

4

Implementación y consideraciones preliminares

Este capítulo presenta los ajustes y consideraciones necesarias para llevar a cabo la correcta aplicación de los controladores en el programa Matlab Simulink y Digsilent PowerFactory. Para asegurar la funcionalidad de los modelos, se hará una primera implementación de los distintos tipos de controladores en el entorno de Matlab Simulink, en el cual, gracias a que vienen incluídas todas las librerías necesarias para hacer los análisis, no son necesarias modificaciones al entorno de simulación.

El sistema de potencia de dos áreas que se pretende controlar en esta primera parte, está representado por el diagrama de bloques de la figura 4.1, donde el bloque del subsistema *Controlador*, contiene el modelo a considerar de cada uno de los controladores. Como se observa en dicho modelo, la potencia de referencia está ajustada en 0 en uno de los generadores de cada área. Esto debido a que solo se realizará control de frecuencia usando uno de los generadores.

Luego de que el modelo del controlador funcione, tal que regule frecuencia correctamente, se puede proceder a implementar los controladores en el entorno de Digsilent PowerFactory en el modelo de 14 barras IEEE a modo de ejemplo, antes de ser ocupado en el modelo real de la conexión SING-SADI.

Debido a que los módulos para utilizar este tipo de controladores no están implementados por defecto, es necesario compilar una librería dinámica *digexfun.dll*, que siendo implementada en el programa permite ocupar funciones programadas por el usuario en el entorno DSL (*Digsilent Simulation Language*).

Además, considerando que hay un proceso de optimización de los parámetros para el caso del controlador Fuzzy, es indispensable también tener un sistema de automatización del programa, de modo de obtener un rendimiento para una simulación dada, y luego editar los parámetros mencionados. Esto puede ser llevado a cabo de tres diferentes formas gracias a las opciones que ofrece Digsilent:

- DPL: El lenguaje *Digsilent Programming Language* permite un acceso directo al programa, ya que el código es ejecutado dentro de él. Esto permite utilizar las variables del programa de forma simple y sin problemas, además de poder automatizar los procesos de simulación. Sin embargo, no ofrece mucho acceso a librerías externas, lo cual puede ser resuelto compilando un archivo *digdplfunmso.dll* donde se pueden agregar funciones programadas por el usuario.

- API en C++/C#: Mediante el uso del API proveída por Digsilent, es posible compilar un achivo ejecutable *.exe*, que es capaz de llamar a las librerías del programa. Gracias a esto se puede combinar con otras librerías de C++ para otros usos.

- API en Python: Es posible un acceso al programa Digsilent mediante el lenguaje Python de forma similar al de API en C++. Tiene la ventaja de ser un lenguaje simple de entender, y poder agregar librerías de forma sencilla y amigable.

Con las consideraciones anteriores se concluye que lo conveniente es usar un acceso a través del API. La elección del lenguaje será Python, debido a que es mas simple y amigable, y que además tiene una gran cantidad de librerías libres interesantes que ayudan a los cálculos y optimizaciones.

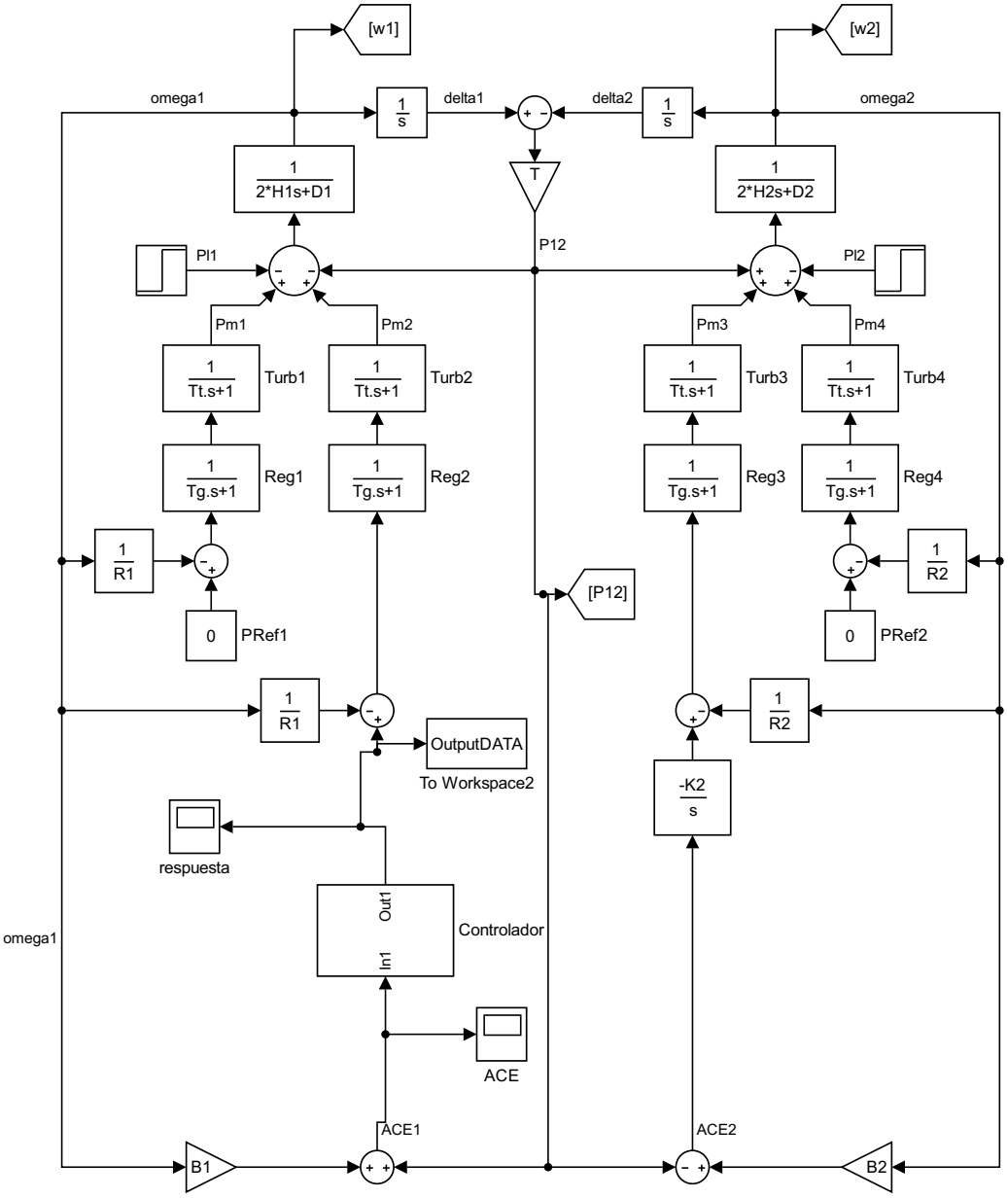


Figura 4.1: Diagrama de bloques del sistema de potencia.

4.1. Modelo del controlador con lógica difusa

Para el diseño del controlador de lógica difusa, se deben hacer las siguientes consideraciones:

- Cantidad de entradas: Se deben considerar como entradas, información importante del sistema, que permita tomar buenas decisiones. En este caso se tomará como entradas: el error de control de área y las variaciones de éste: ACE y ΔACE respectivamente.

- Cantidad de salidas: Se sabe que se debe modificar las consignas de potencia de los generadores, por lo que se toma la salida como una única gran consigna que luego es manejada a través de los factores de participación.

- Variables lingüísticas: Se deben considerar clasificaciones lingüistícas de las variables (tanto de entrada como de salida) de modo de catalogar los puntos en los universos de discurso. Para ello se han considerado: Muy positivo (MP), Positivo (P), Cero (Z), Negativo (N), Muy Negativo (MN). Estas clasificaciones están tanto para las entradas como para la salida.

- Se ocupan funciones triangulares debido a que es más fácil parametrizar las funciones de membresía en torno a una sola variable.

En el diseño del controlador será considerando lo siguiente: el sistema de inferencia Fuzzy sólo decidirá si es necesario más o menos generación y en cuanto, de parte de las unidades generadoras. Para lograr esto, es necesario adicionar a la salida un bloque integrador, de modo de adicionar o sustraerle

El diagrama de bloques del controlador completo puede ser visualizado de mejor forma en la Figura 4.2. En éste se tiene al ACE como entrada, y mediante el bloque de retraso se puede obtener el ΔACE, y además finalmente, se tiene a la salida del sistema de inferencia FIS un bloque integrador.

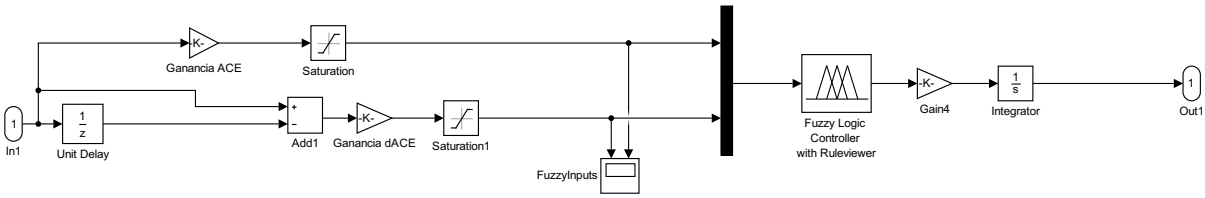


Figura 4.2: Diagrama de bloques del controlador difuso (FLC).

Luego de definidos los puntos anteriores, se deben hacer consideraciones o reglas lingüísticas que relacionen las entradas con la salida, lo cual es simple de realizar. Por ejemplo, si el valor de ACE es muy negativo significa que hay una subida de frecuencia (debido a como se modeló el ACE en este modelo, pero posteriormente se puede arreglar cambiando el signo de la salida), por lo que si a eso, le agregamos una caída grande de ACE (ΔACE), sería necesario que la generación baje abruptamente. Resumido en una línea se tiene:

Si ACE es MN y ΔACE es MN, entonces la salida es MN.

Luego se diseña la tabla 4.1 que contiene todas las combinaciones dadas.

	ΔACE				
ACE	MN	N	Z	P	MP
MN	MN	MN	N	N	Z
N	MN	N	N	Z	P
Z	N	N	Z	P	P
P	N	Z	P	P	MP
MP	Z	P	P	MP	MP

Tabla 4.1: Tabla con las reglas lingüísticas del controlador.

4.1.1. Resultados simulaciones en Simulink

Con el modelo de sistema y el controlador mencionado anteriormente se pueden hacer simulaciones dinámicas en Simulink, de modo de determinar si funciona correctamente. Para este caso no se ocupó ningún tipo de optimización, por lo que se utilizaron valores arbitrarios iguales para definir las funciones de pertenencia de las entradas y las salidas.

Los controladores ocupados fueron obtenidos directamente de la librería de Simulink, con lo que se simplifica el trabajo de tener que diseñar unos propios.

Debido a que se diseñó el FIS con un universo de discurso entre -1 y 1, se debe hacer un escalamiento de las entradas, de modo de que siempre estén en ese rango. Para ello se realizó una simulación con un controlador PI cualquiera, de modo de ver el máximo valor que toma el ACE y ΔACE, y dividir las entradas por ese factor. Luego el factor usado para el escalamiento del ACE es utilizado para re-escalar la salida, de modo que el ajuste de la salida tenga un impacto de la misma medida que el error.

La simulación fue realizada con éxito y es mostrada en las Figuras 4.3, 4.4 y 4.5 donde se puede observar que el sistema recupera su frecuencia nominal (50 Hz) sin mayores problemas, y sin oscilaciones muy pronunciadas. Además se puede notar en el gráfico de la Figura 4.5 que el controlador toma decisiones muy mesuradas en cuanto a la salida de generación, ya que incluso cuando el ACE que ve está oscilando en gran medida, no hace variaciones significativas en relación a cuanto generar, o sea, aunque el ACE oscile de forma considerable no se ve reflejado en oscilaciones de la potencia de referencia elegida por el controlador. Lo cual sería una característica interesante en los sistemas reales, ya que las generadoras recibirían un valor de referencia de generación no tan variable, lo cual se traduciría en un menor desgaste de las válvulas.

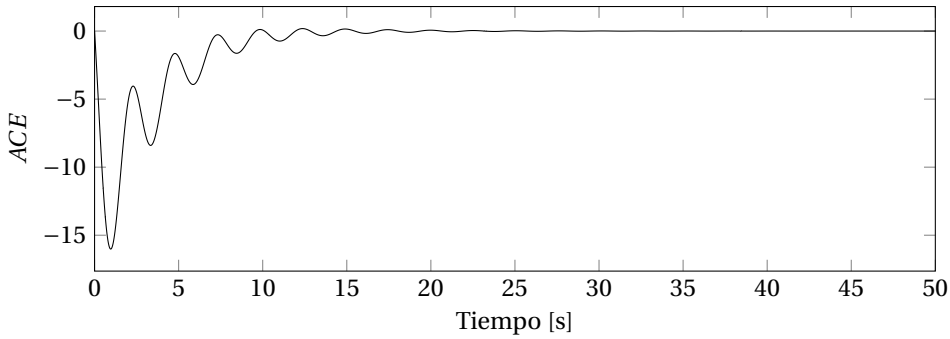


Figura 4.3: ACE en el tiempo, de un sistema con controlador difuso ante una subida de 10 p.u. de demanda en forma escalón.

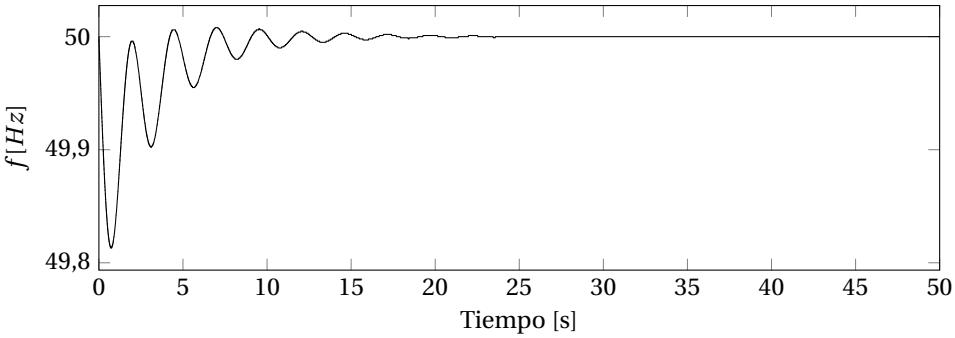


Figura 4.4: Frecuencia en el tiempo, de un sistema con controlador difuso ante una subida de 10 p.u. de demanda en forma escalón.

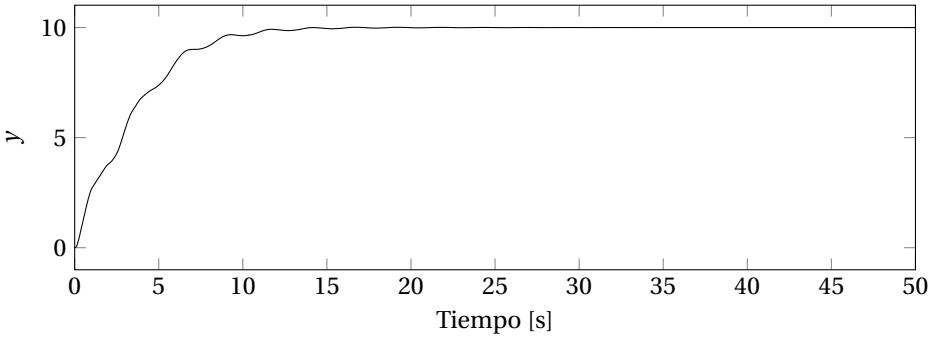


Figura 4.5: Salida del controlador en el tiempo, de un sistema con controlador difuso ante una subida de 10 p.u. de demanda en forma escalón.

4.1.2. Resultados simulaciones en Digsilent

Para el caso de las simulaciones en Digsilent se considera un modelo de funciones de pertenencia para la entradas: ACE y ΔACE, y para la salida: y, como las que se muestran en las Figuras 4.6, 4.7 y 4.8, definidas por los valores b_{ACE}, $b_{\Delta ACE}$, y b_{out}.

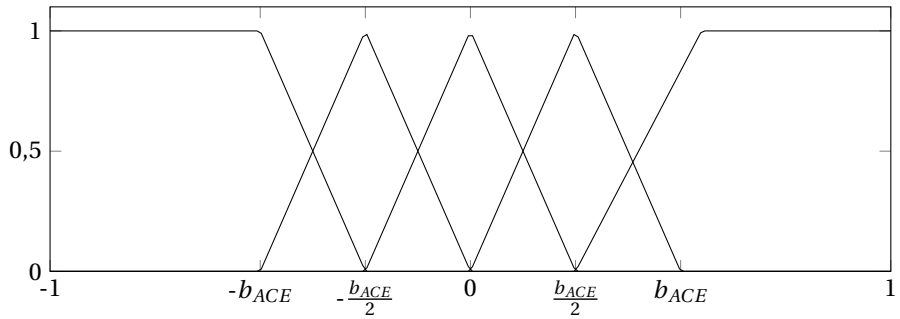


Figura 4.6: Funciones de pertencia para la entrada ACE.

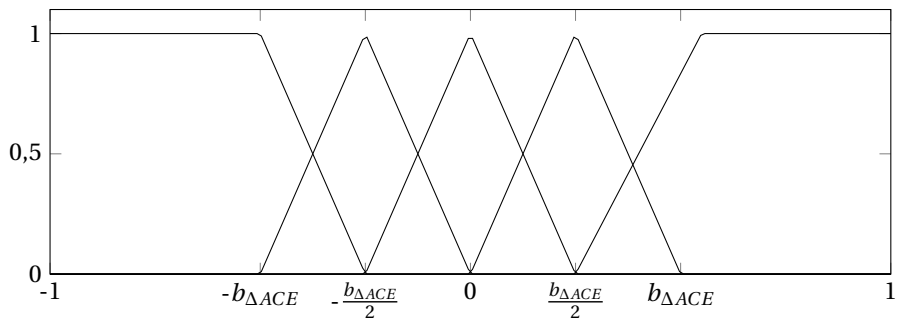


Figura 4.7: Funciones de pertencia para la entrada Δ ACE.

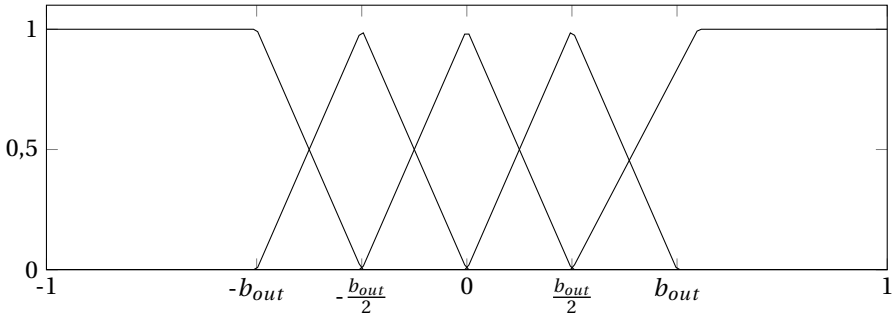


Figura 4.8: Funciones de pertencia para la salida.

Luego, se debe definir una expresión que determine el rendimiento del controlador para ciertos valores b_{ACE}, $b_{\Delta ACE}$, y b_{out}. Una buena medida de cómo se comporta el mismo, es calcular el área bajo la curva del ACE en el tiempo:

$$Rendimiento = \int_0^T (ACE)^2 dt \tag{4.1}$$

Por lo tanto, lo que se busca con el controlador es minimizar la expresión 4.1, lo cual puede lograrse por ejemplo, usando el algoritmo de optimización por cúmulo de partículas (PSO), que es el que será utilizado en este estudio.

Para ello se tiene primeramente que Python (el código está adjunto en el anexo) controla una instancia de Digsilent en modo ENGINE, que luego desde su función programada en *digexfun.dll* lee un archivo *.fis* con las configuraciones de su sistema de inferencia FIS, con lo que se hace una simulación de un escenario dado, con un controlador difuso definido. Luego se obtiene el ACE en el tiempo de dicha respuesta, y se exporta a un archivo *.txt*, el cual posteriormente es leído por Python nuevamente, donde esta vez hace el cálculo de desempeño mencionado anteriormente, con lo que hace una modificación de los parámetros b_{ACE}, $b_{\Delta ACE}$, y b_{out} en el archivo *.fis*, completando el proceso iterativo.

Con lo anterior en consideración, se tendría que el esquema de conexión entre programas y entornos sería como el de la Figura 4.9. Como se puede observar Python tiene acceso a Digsilent, donde sólo lo puede abrir (que a su vez carga el *.dll* que lee el archivo con parámetros) y simular cierto escenario, pero Python a su vez también tiene acceso al archivo con parámetros que está leyendo Digsilent.

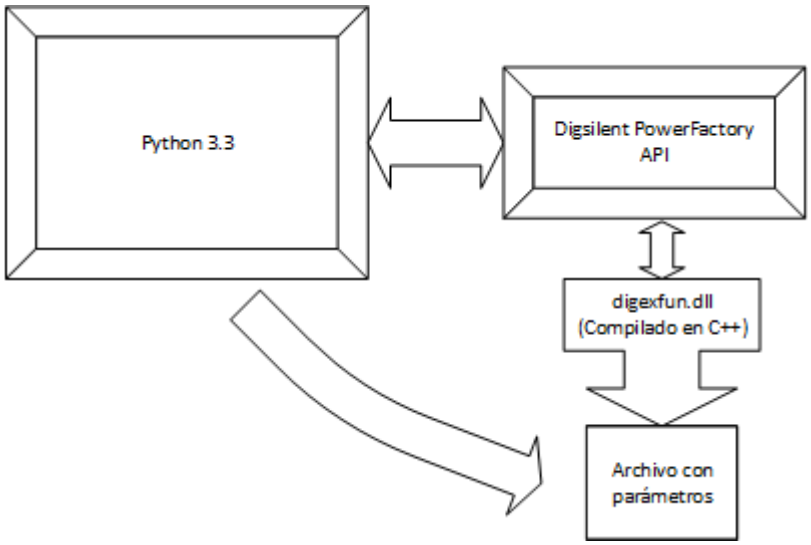


Figura 4.9: Esquema de conexión entre los entornos a utilizar.

Como esta sección es a modo de ejemplo, se elige un caso de forma arbitraria de modo ilustrativo. Para las optimizaciones se ocupará el caso de la pérdida de uno de los generadores que participan en la regulación de frecuencia (G21).

Finalmente, luego de 15 iteraciones con una muestra de 30 partículas, se obtuvieron los parámetros, que optimizan el caso de pérdida del generador 21:

b_{ACE}	$b_{\Delta ACE}$	b_{out}	$\int ACEdt$
0.5862434	0.9283341	0.95	2845.1537

Tabla 4.2: Valores óptimos obtenidos que definen las funciones de pertenencia.

Donde el comportamiento del controlador se evidencia en los gráficos de las Figuras 4.10, 4.11 y 4.12. En ellos se puede observar que el controlador cumple correctamente su tarea de reducir el ACE, y devolver la frecuencia a su valor nominal.

Además se puede notar que ocurre lo que se había observado en el caso de Simulink, ya que en el gráfico 4.12 queda a la vista que no hay mayores oscilaciones en la salida del controlador.

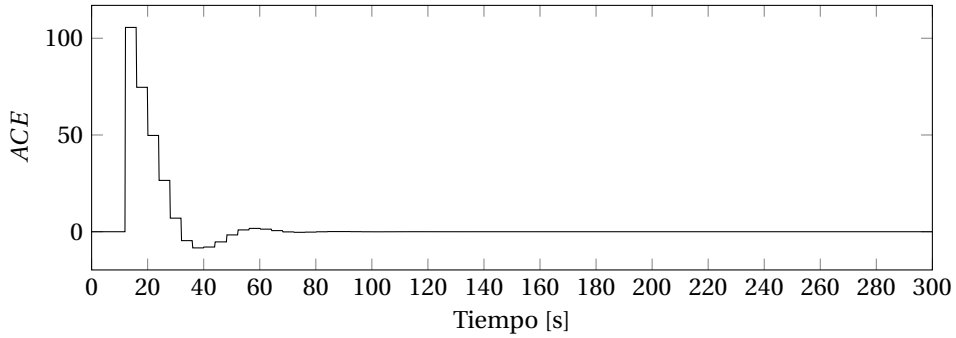


Figura 4.10: ACE en el tiempo, de un sistema con controlador difuso ante una caída de generación (G21)

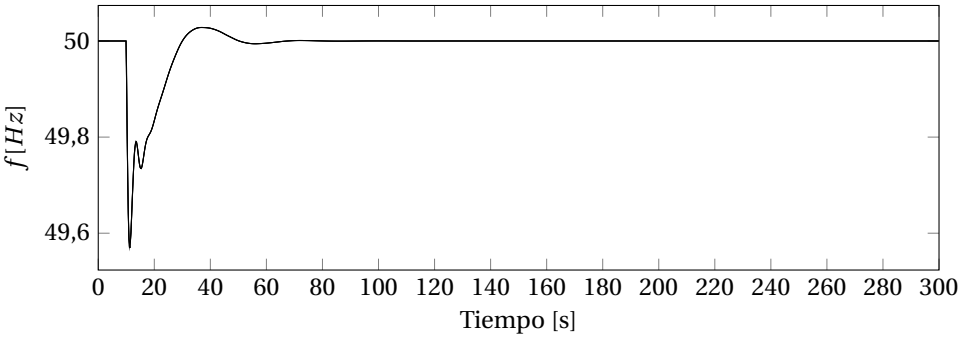


Figura 4.11: Frecuencia en el tiempo, de un sistema con controlador difuso ante una caída de generación (G21)

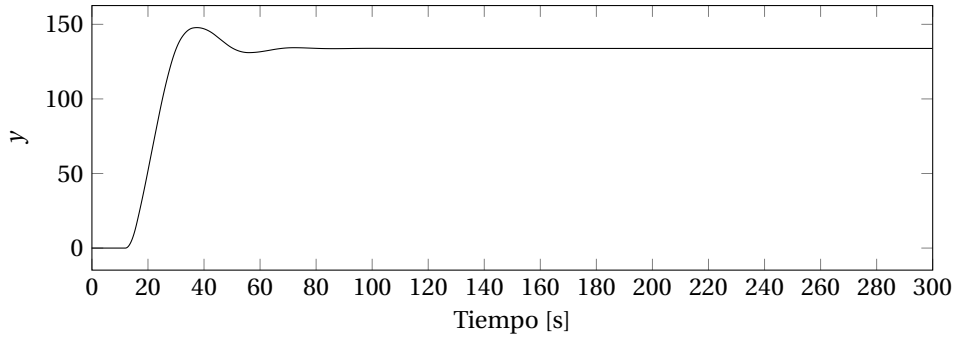


Figura 4.12: Salida del controlador en el tiempo, de un sistema con controlador difuso ante una caída de generación (G21)

4.2. Modelo con Redes Neuronales Artificiales

Para el controlador basado en redes neuronales se ocupará un modelo de una capa oculta, esto debido a que el comportamiento en sí del sistema y el controlador no requieren capas adicionales, además de tener en consideración que utilizar aprendizaje profundo requiere pasos adicionales para configurar correctamente las capas interiores, lo cual complica la creación de este controlador innecesariamente.

Para determinar el esquema a usar de la red neuronal se deben tener en cuenta los siguientes aspectos:

- Cantidad de capas ocultas: el uso de redes neuronales con más de una capa oculta se justifica cuando hay necesidad de especializar la red neuronal de modo de hacer operaciones complejas, por ejemplo extraer cualidades y hacer discriminaciones (el aprendizaje profundo, como es llamado, es usado ampliamente por ejemplo en el reconocimiento de imágenes). En este estudio, el uso de una capa es suficiente y simplifica el modelamiento, por lo que en definitiva se ocupará este tipo de esquemas.

- Cantidad de entradas a tener en consideración: Igual que en el caso del controlador difuso, es necesario para el correcto funcionamiento del controlador con redes neuronales, el poder determinar que parámetros debe tener en cuenta para hacer buenas decisiones. En este estudio se ocupará el ACE y ΔACE, ya que son buenos indicadores de lo que está ocurriendo en el sistema.

- Cantidad de neuronas por capa: Esto es complicado de determinar, ya que en la literatura no hay una decisión en común de como configurar la red neuronal, además de ser completamente dependiente del modelo y del caso a tener en cuenta. Para este estudio arbitrariamente se elegirán cuatro neuronas en la capa oculta.

- Funciones de activación de las capas: Para poder aplicar el método de propagación inversa es necesario que las funciones de activación sean derivables y continuas. Para el caso de la capa oculta es necesario algún tipo de función que permita realizar discriminaciones de modo de determinar que entradas son más importantes, por lo tanto podrían ser ocupadas funciones de tangente hiperbólica o sigmoide, siendo este último el elegido finalmente (cualquiera de las dos funciona bien). Mientras que para la capa de salida será usada una función lineal, esto debido a que el escalamiento de salida se hace innecesario, y el controlador por si mismo elige el factor de la salida, lo cual representa una gran ventaja.

Por lo tanto se tiene un esquema como el de la Figura 4.13, con dos entradas: ACE, y ΔACE, y una salida y, con una estructura de una capa oculta de cuatro neuronas. La cantidad de neuronas en esta capa no cambiarán demasiado las simulaciones, por lo que arbitrariamente se elige un número mayor a la cantidad de entradas, pero no muy alejado de ello.

Para entrenar redes neuronales se pueden considerar tres modalidades de entre las existentes:

- Tomar simulaciones de otros controladores (por ejemplo uno difuso), y obtener sus respuestas ante variados casos. Con ello se puede entregar la información de la entrada y la salida de modo que la red neuronal desarrolle un comportamiento similar.

- Dejar un escenario o un conjunto de escenarios a simular, y variar los parámetros de la red neuronal utilizando algún algoritmo como algoritmo genético.

- Desarrollar un mecanismo de entrenamiento *on-line*, lo cual significa que se le da un objetivo (en este caso disminuir el ACE), y a través del algoritmo de propagación inversa determinar los valores de los pesos que permitan llevar a cabo dicho objetivo.

El método 2 es sumamente ineficiente, ya que debido a la gran cantidad de combinaciones de pesos existentes, se necesitaría una cantidad muy grande de iteraciones en algoritmo genético para llegar a valores de pesos que permitan un buen comportamiento del controlador, y las iteraciones (simulaciones) a concretar son muy exigentes en recursos y toman mucho tiempo en realizarse.

Por lo general lo que se utiliza en estos casos es una combinación de los dos primeros casos, o sea, primero se llega a pesos *semi-óptimos* entrenando al controlador según el comportamiento de otros controladores que actúen correctamente, y luego utilizar ese punto como punto inicial para el uso de algoritmo genético.

Con lo mencionado anteriormente, el método 3 parece mucho más atractivo, además que supone que el controlador sería capaz de adaptarse en el tiempo, lo cual le da una plusvalía especial.

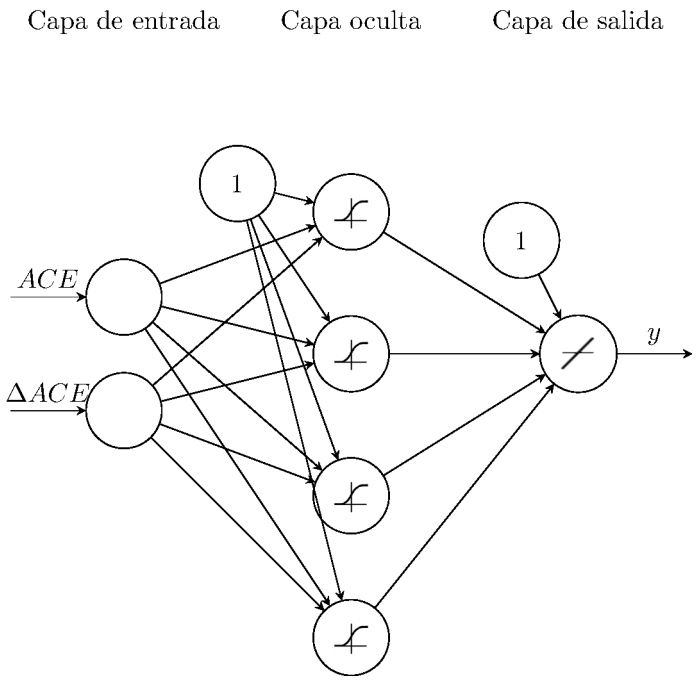


Figura 4.13: Esquema de red neuronal de una capa oculta.

Para la propagación inversa se tiene que el error es traspasado hacia atrás cambiando los pesos entre capas. Para ello, el error sería la diferencia entre la salida del controlador y la salida esperada, sin embargo, es imposible saber a priori la salida esperada.

Pero lo que ocurre realmente con la propagación inversa, es el intento llevar un valor al que llamamos error, a cero al propagarlo hacia atrás, cambiando los pesos según su influencia en dicho valor. Por lo tanto, si consideramos al error de la forma:

$$\epsilon = (ACE_{deseado} - ACE_{actual}) \tag{4.2}$$

Y consideramos que el $ACE_{deseado}$ debe anularse:

$$\epsilon = (0 - ACE_{actual}) = -ACE_{actual} \tag{4.3}$$

Por lo tanto, en resumen, en un primer tiempo la red neuronal lleva a cabo un cálculo de la salida del controlador, que causa un cambio en la generación del sistema, y luego, en un segundo tiempo obtiene de vuelta el ACE_{actual}, y lo usa para ajustar sus pesos de modo de llevarlo a cero.

4.2.1. Resultados en Simulink

Para las simulaciones en Simulink se tuvo que desarrollar un controlador propio usando las librerías existentes. Esto debido a que sólo existen librerías de redes neuronales ya entrenadas (o sea con pesos ya calculados anteriormente), lo cual no es lo que se va a utilizar sino una red neuronal que se adapte mientras simule.

Por lo tanto, se desarrolló un controlador propio que se puede visualizar en la Figura 4.14. En él se pueden distinguir dos procesos: propagación hacia adelante, y propagación inversa.

La propagación hacia adelante da el valor de salida del controlador según los pesos actuales del mismo. Luego, se le entrega el valor del ACE actual, el cual es propagado como error hacia atrás en la propagación inversa, esto debido a que el objetivo del controlador es llevar el ACE a cero.

Como se describió anteriormente, estos dos procesos ocurren en distintos tiempos (primero propagación hacia adelante y luego propagación inversa), por lo que es necesario agregar bloques de retrasos a los procesos, de modo de no tener *loops* algebraicos. También es necesario agregar bloques IC (initial condition) de modo de iniciar los pesos.

Utilizando el controlador mencionado anteriormente en la simulación, se obtuvieron los resultados de las Figuras 4.15, 4.16 y 4.17. En dichos resultados se aprecia claramente que el controlador regula la frecuencia correctamente, con lo que se puede concluir que fue correctamente implementado.

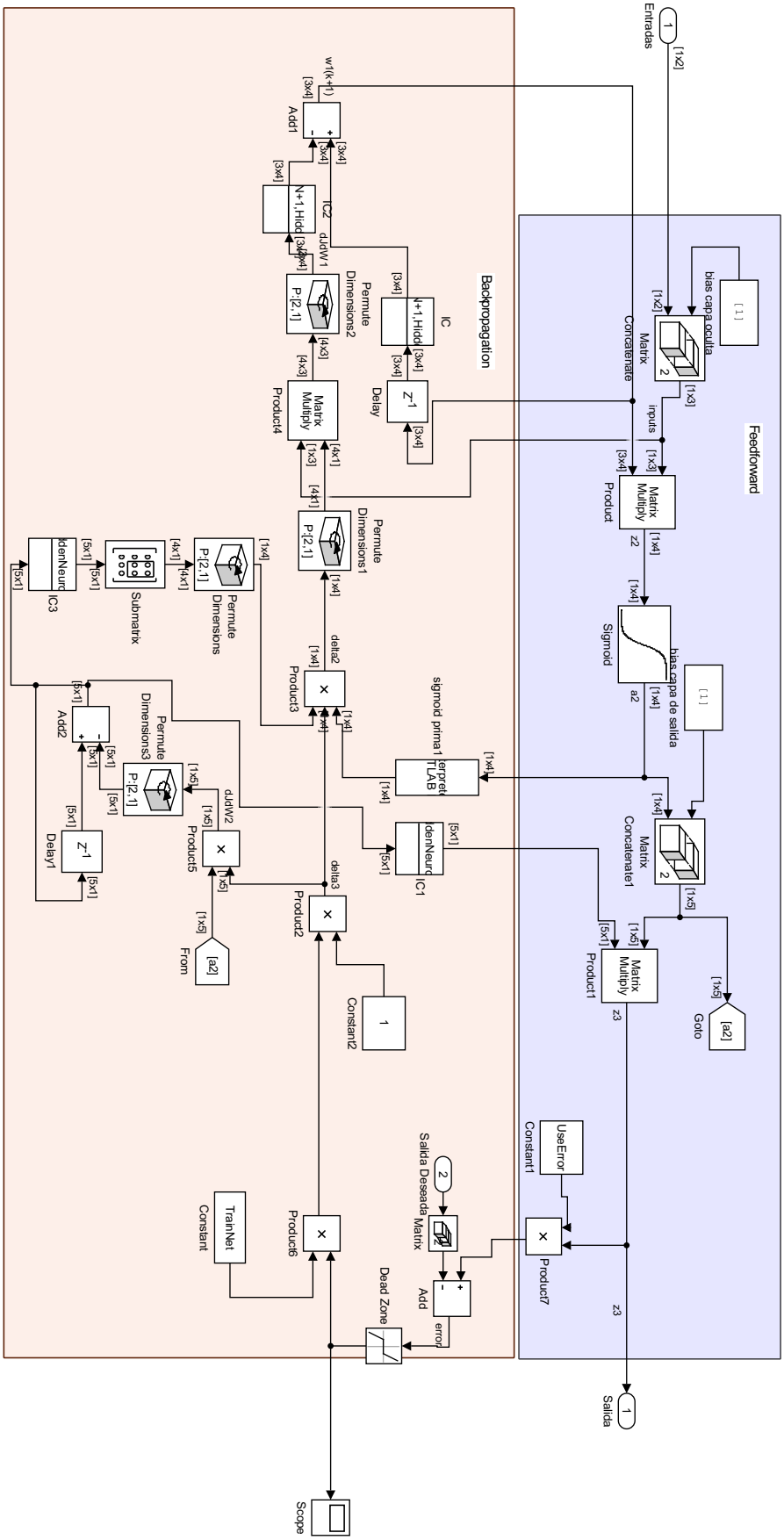


Figura 4.14: Controlador ANN utilizado en el modelo de Simulink.

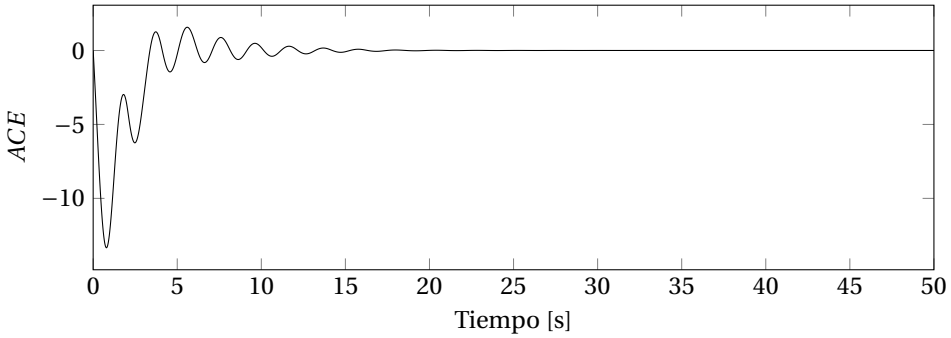


Figura 4.15: ACE en el tiempo, de un sistema con controlador con redes neuronales ante una subida de 10 p.u. de demanda en forma escalón.

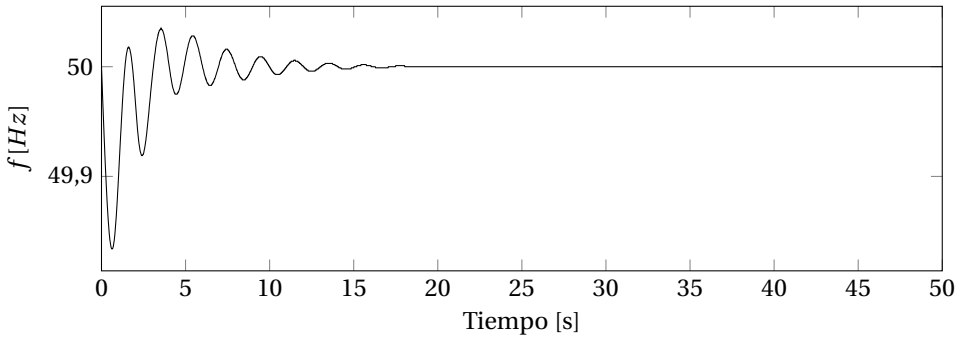


Figura 4.16: Frecuencia en el tiempo, de un sistema con controlador con redes neuronales ante una subida de 10 p.u. de demanda en forma escalón.

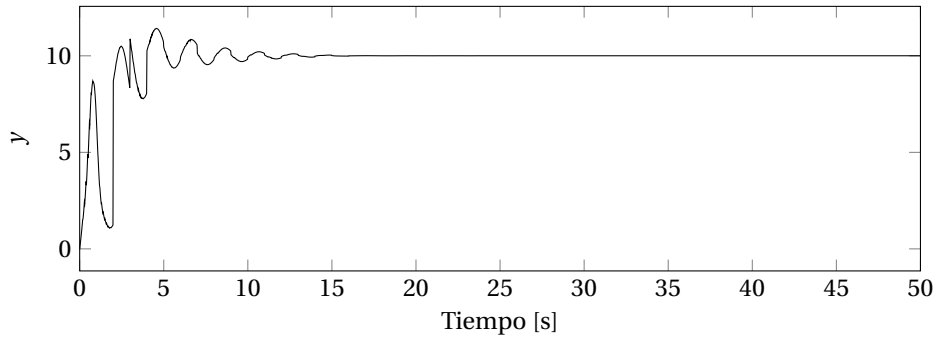


Figura 4.17: Salida del controlador en el tiempo, de un sistema con redes neuronales ante una subida de 10 p.u. de demanda en forma escalón.

4.2.2. Resultados en Digsilent

Para las simulaciones en Digsilent se tuvo que desarrollar un código propio y adicionarse a la librería de DSL. En él se aplicaron las mismas condiciones que las mencionadas anteriormente, salvo la excepción que fue necesario agregar un parámetro a la función en DSL, que indica cada cuantos pasos de tiempo se realiza una propagación inversa o cambio en los pesos. Esto fue necesario debido a que en el modelo real se ocupa el muestreo de las señales, por lo que el algoritmo trata de propagar ese error sin ver cambios, lo cual es desventajoso para el correcto funcionamiento del controlador.

Lo mencionado anteriormente es análogo a decir que realice propagación inversa cada t segundos, donde t debe ser directamente proporcional al tiempo de muestreo.

Adicionalmente, para evitar el funcionamiento durante los tiempos donde ocurre la regulación primaria, es útil también utilizar algún tipo de filtro de alta frecuencia. Para ello, el valor que se propaga como error es

el promedio de los valores en el intervalo descrito anteriormente, con ello se logra una señal que disminuye un poco los efectos *peak* de los primeros instantes al ocurrir la falla.

Finalmente se obtienen las simulaciones en Digsilent, y son mostradas en las Figuras 4.18, 4.19 y 4.20. En ellas se observa que el control no fue tan bueno como en el caso del controlador difuso, pero hay que tener en consideración que para este caso el controlador inició sin configuración alguna (pesos aleatorios) y terminó controlando correctamente la frecuencia.

El no haber sido necesario utilizar algún tipo de optimización del tipo PSO, sino que un algoritmo que actualiza el parámetro de forma *on-line* representa una gran ventaja, debido a que al realizar una optimización del primer tipo se tiene que tiene el mejor funcionamiento para ese caso, pero no se asegura un funcionamiento óptimo para otro escenario, mientras que en el controlador con redes neuronales se adapta naturalmente. Además de considerar que no es necesario utilizar recursos computacionales, ni diseños adicionales para realizar la optimización.

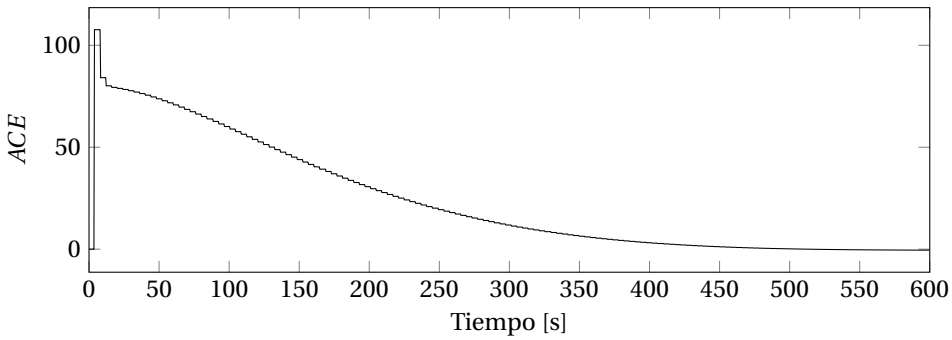


Figura 4.18: ACE en el tiempo, de un sistema con controlador con redes neuronales ante la caída de un generador (G21).

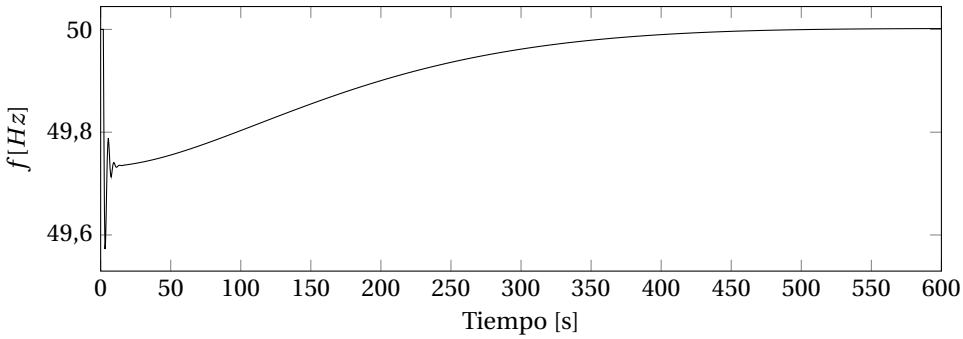


Figura 4.19: Frecuencia en el tiempo, de un sistema con controlador con redes neuronales ante la caída de un generador (G21).

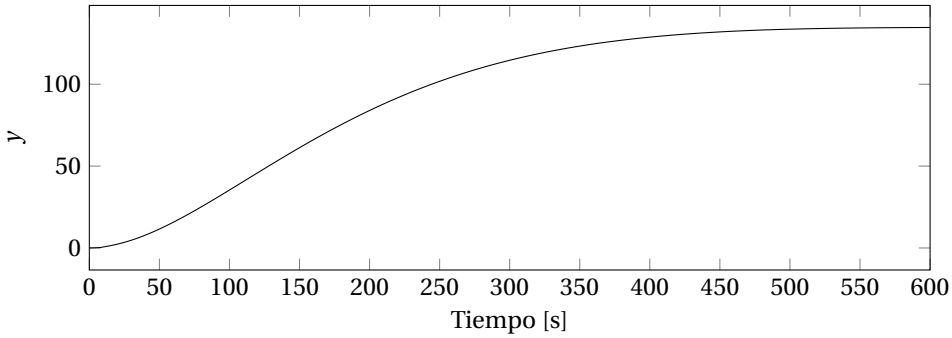


Figura 4.20: Salida del controlador en el tiempo, de un sistema con controlador con redes neuronales ante la caída de un generador (G21).

4.3. Modelo con sistema híbrido ANFIS

El modelo con sistema híbrido mantiene las características de los dos modelos presentados anteriormente. La idea es ajustar los parámetros de las funciones de pertenencia de forma automatíca, de modo de no ser necesario un conocimiento previo del sistema para desarrollar el controlador mismo.

4.3.1. Resultados en Simulink

Para el desarrollo del modelo en Simulink fue necesario crear el modelo desde cero, ya que las librerías existentes sólo permitian incorporar modelos ya entrenados, además de que no se logra comprender totalmente la estructura interna utilizada por los desarrolladores originales.

Para ello se tiene un modelo de propagación hacia adelante como el de la Figura 4.21, que hace una evaluación inicial de salida (dado que tiene valores iniciales aleatorios).

Con ello, luego, se tiene un error que se debe corregir, el cual es el ACE. Éste se propaga hacia atrás actualizando los parámetros que permitieron la evaluación anteriormente descrita. Esto se desarrolla también en Simulink como se muestra en la Figura 4.22.

Se debe notar que en el modelo solo se tienen dos reglas, esto acota mucho las posibilidades y respuestas del controlador. Esto fue hecho así ya que la única finalidad del modelo en Simulink es demostrar su funcionalidad, en la etapa posterior habrá un refinamiento en esta materia.

Aún así, los resultados - que pueden ser visualizados en las Figuras 4.23, 4.24 y 4.25 - demuestran que el sistema funciona sin mayores problemas y da buenos resultados.

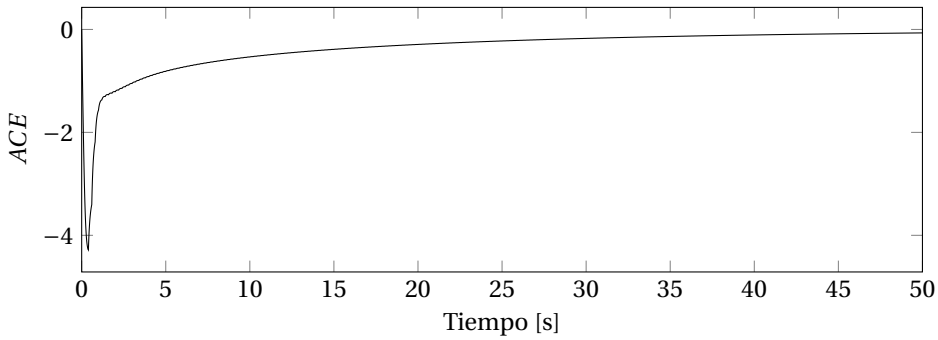


Figura 4.23: ACE en el tiempo, de un sistema con controlador ANFIS ante una subida de 10 p.u. de demanda en forma escalón.

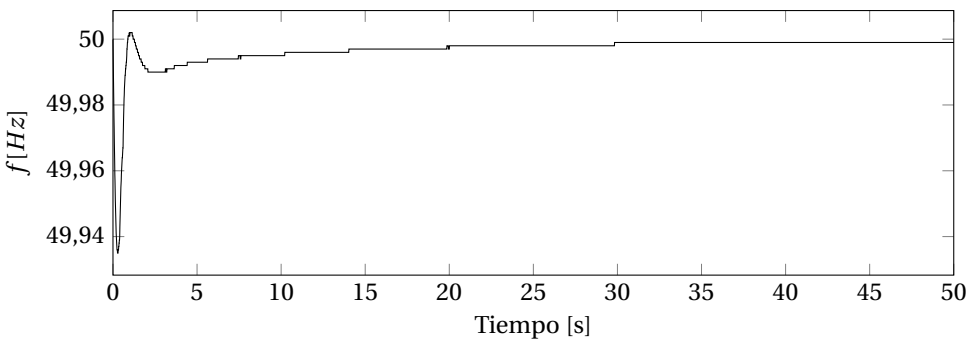


Figura 4.24: Frecuencia en el tiempo, de un sistema con controlador ANFIS ante una subida de 10 p.u. de demanda en forma escalón.

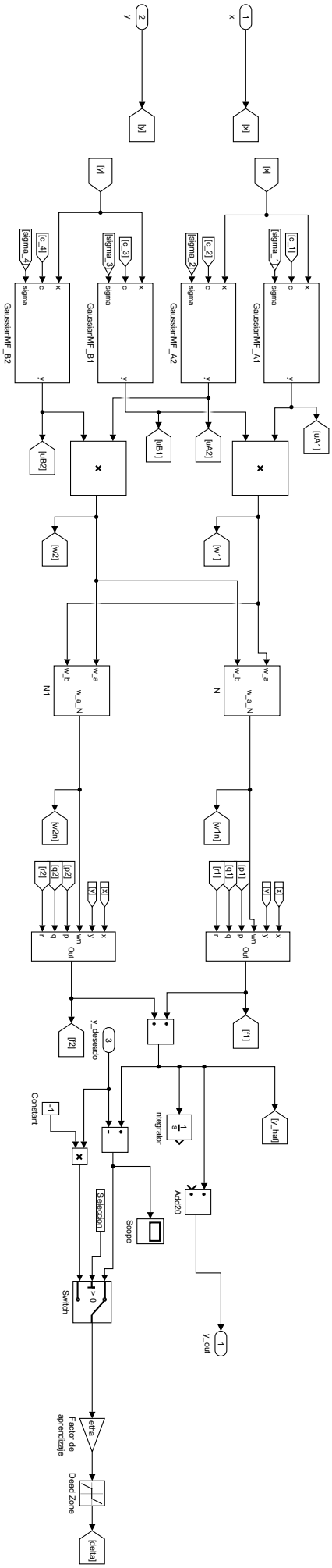


Figura 4.21: Propagación hacia adelante de un controlador ANFIS con dos reglas.

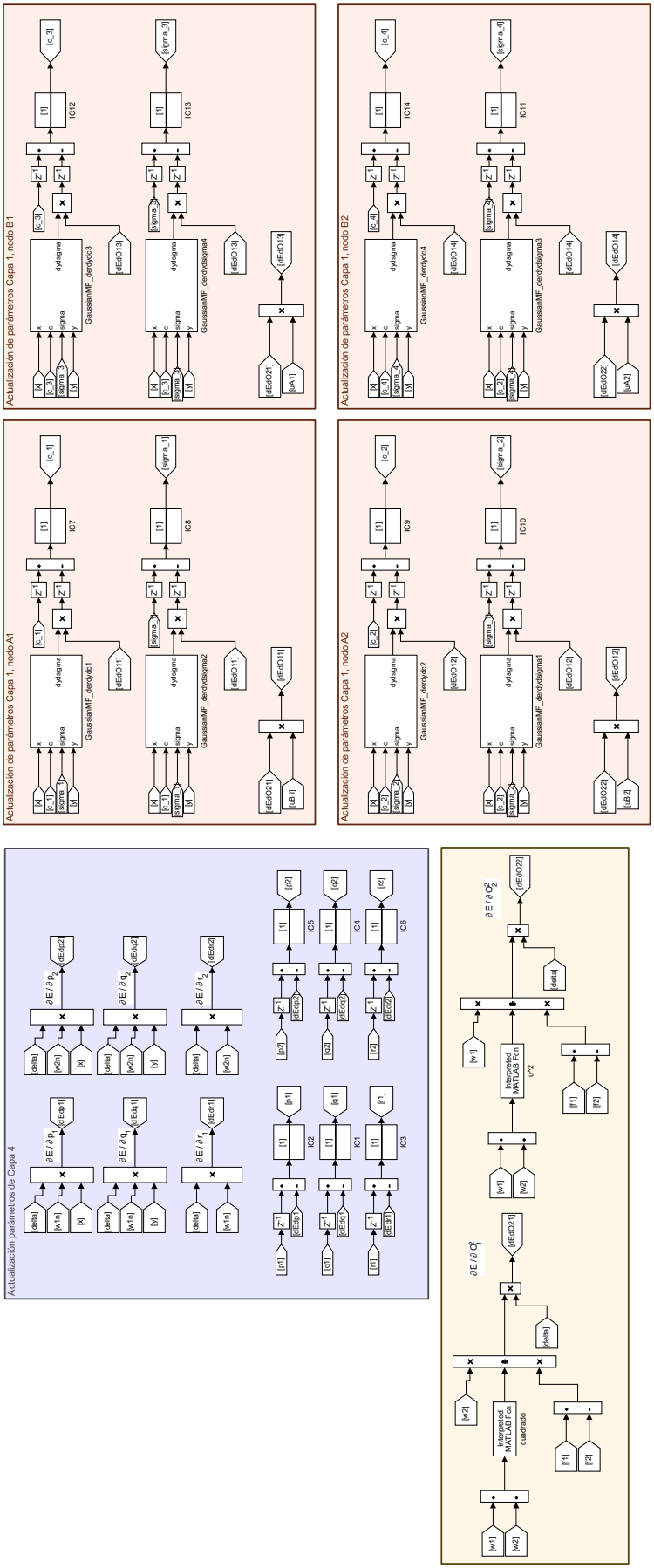


Figura 4.22: Propagación inversa de un controlador ANFIS con dos reglas.

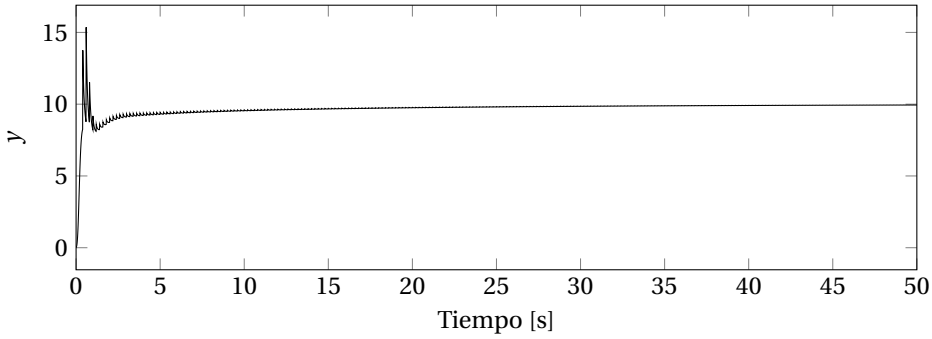


Figura 4.25: Salida del controlador en el tiempo, de un sistema con controlador ANFIS ante una subida de 10 p.u. de demanda en forma escalón.

4.3.2. Resultados en Digsilent

Para este caso se deben hacer consideraciones adicionales, ya que es el modelo que sera incorporado al sistema final. Por ello la cantidad de reglas aplicadas será mayor de modo de obtener una mejor respuesta del sistema.

Si se consideran tres funciones de pertenencia por entrada, se deberían tener 9 reglas, ya que la mejor representación sería la combinatoria entre todas las salidas de las funciones de pertenencia.

Un diagrama que permita visualizar de mejor manera el controlador es mostrado en la Figura 4.26.

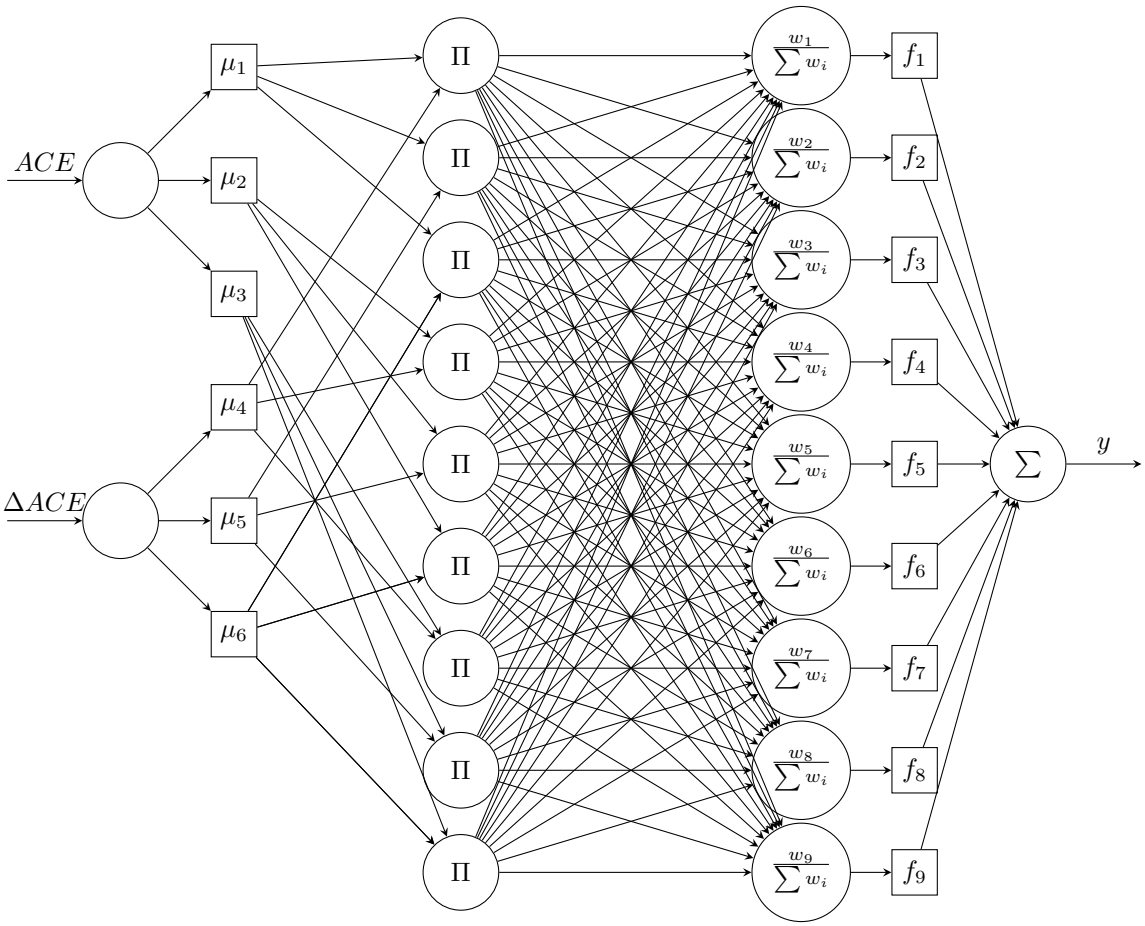


Figura 4.26: Esquema de red neuronal de una capa oculta.

Donde μ_i es una función del tipo Gaussiana, que depende de sus parámetros adaptables σ, y c:

$$\mu_i(x,\sigma,c) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{4.4}$$

Y f_i es una función adaptable, dependiente de p y q, del tipo:

$$f_i = w_i \cdot (x_1 \cdot p_i + x_2 \cdot q_i + r_i) \tag{4.5}$$

El modelo fue desarrollado en C++, creando un archivo *digexfun.dll*, que permite utilizar dentro del lenguaje DSL, la función *ANFISFeedforward* en Digsilent, que hace la evaluación por propagación hacia adelante y luego hace el ajuste de los parámetros por propagación inversa del mismo modo que el modelo con redes neuronales.

Los resultados de las simulaciones se pueden observar en las Figuras 4.27, 4.28 y 4.29.

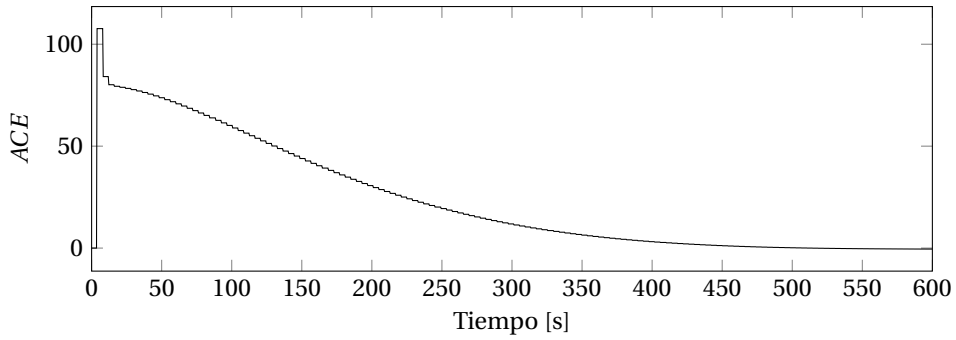


Figura 4.27: ACE en el tiempo, de un sistema con controlador con redes neuronales ante la caída de un generador (G21).

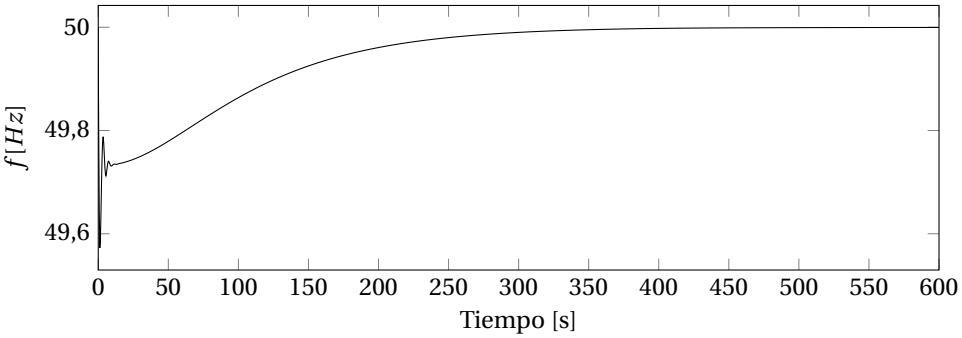


Figura 4.28: Frecuencia en el tiempo, de un sistema con controlador con redes neuronales ante la caída de un generador (G21).

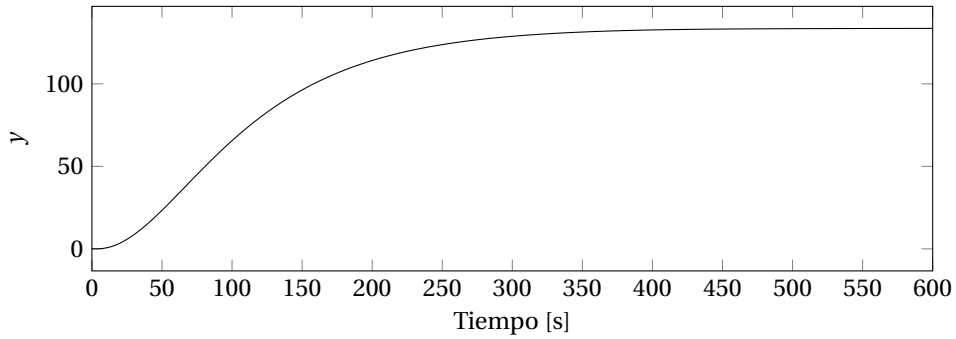


Figura 4.29: Salida del controlador en el tiempo, de un sistema con controlador con redes neuronales ante la caída de un generador (G21).

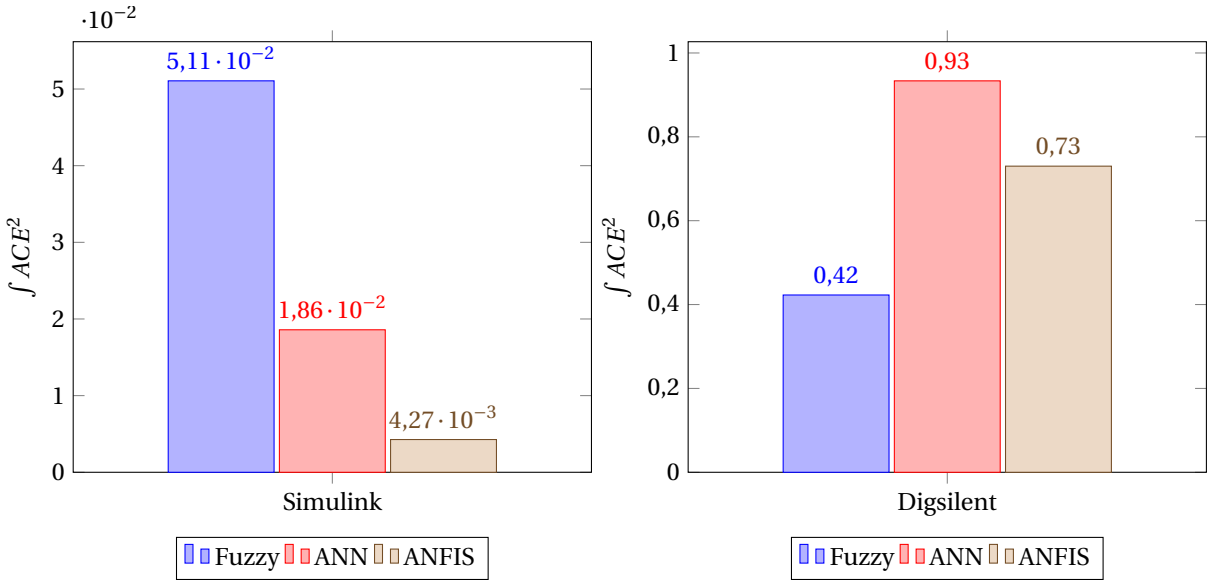
4.4. Conclusiones preliminares sobre los resultados

De modo de hacer una comparación preliminar, se obtienen los rendimientos utilizando la ecuación (2.33), tanto para las simulaciones en Simulink como para Digsilent. Dichos resultados están mostrados en las figuras 4.30a y 4.30b.

Para las simulaciones hechas en Simulink, se observa que el controlador Fuzzy fue el peor de los tres. Esto es debido a que en esta sección sus parámetros no fueron optimizados.

Para la simulación en Digsilent, contrario al caso anterior, el controlador Fuzzy fue el mejor. Lo cual se debe principalmente a que el controlador fue optimizado para este caso específico, y no se le han hecho cambios significativos o revisiones importantes a los demás controladores.

El principal motivo de este capítulo no es determinar que controlador es mejor, sino que lograr demostrar que todos los controladores regulan frecuencia correctamente, con lo que están modelados de buena forma.



(a) Resultados obtenidos en las simulaciones en Simulink. (b) Resultados obtenidos en las simulaciones de Digsilent.

Figura 4.30

5

Implementación en el modelo real del SING y resultados

Como se mencionó anteriormente, lo primero en el desarrollo de los controladores es determinar un escalamiento óptimo de modo que los casos queden acotados en un rango bien definido. Al hacer referencia al estudio realizado por el AC3E al CDEC con respecto al AGC [4], se tiene que el caso con mayor caída de frecuencia es el escenario correspondiente a la salida de la unidad generadora U16. Al observar los gráficos de las Figuras 5.1, y 5.2, se tiene el ACE y ΔACE de dicho escenario, con lo que se puede determinar que un factor de escalamiento apropiado (que llamaremos α_{ACE} y $\alpha_{\Delta ACE}$) podría ser:

$$\alpha_{ACE} = 230 \quad \alpha_{\Delta ACE} = 200 \tag{5.1}$$

Dicho escalamiento de entrada será utilizado para todos los controladores. Para la salida se tiene un reescalamiento que será en este caso el mismo que fue usado para el ACE, al cual se le llamará β_y:

$$\beta_y = 230 \tag{5.2}$$

Cabe destacar que estas suposiciones son arbitrarias, y no representan el óptimo escalamiento que debería ser ocupado. Esto por que se toma un caso particular, con lo que los demás escenarios al no comportarse del mismo modo, puede que no estén contenidos de la misma forma dentro de los rangos deseados.

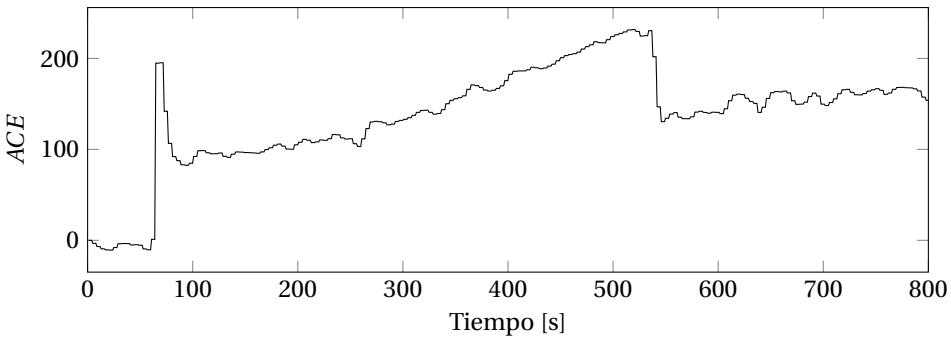


Figura 5.1: ACE del caso base (salida de generador U16).

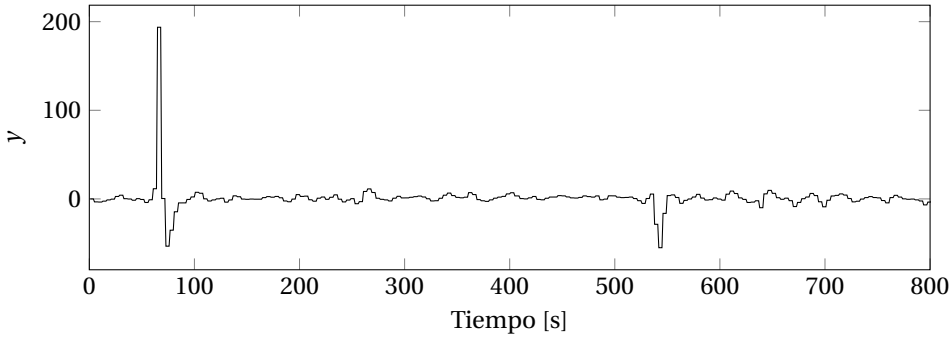


Figura 5.2: ΔACE del caso base.

Para lograr una forma de valorar el desempeño en el tiempo de los controladores se debe definir algún tipo de cálculo sobre alguna de las variables. Una forma apropiada de designar dicho cálculo podría ser:

$$\sum_{n=1}^{N} \frac{1}{N} ACE_n^2$$

Donde N es el número de muestras. Se divide por el tamaño de muestras tomada para evitar errores sobre si se tomó la simulación con tiempos más pequeños (y por lo tanto con más muestras).

5.1. Implementación

Para este trabajo se ha considerado el mismo esquema de control utilizado en el trabajo para el CDEC.

La Figura 5.3 muestra el esquema general de control del sistema en Digsilent, se observa en *1* un bloque que tiene de entradas las mediciones de frecuencia (se utiliza la medición en la barra Crucero 220kV). Dicho bloque puede ser visualizado con mayor detalle en la Figura 5.4, donde se observa que la frecuencia es pasada desde pu a Hz, posteriormente por un filtro pasabajo, y finalmente por un factor β mencionado anteriormente. Con ello finalmente se genera un valor de ACE continuo, que pasa por un bloque del tipo Sample and Hold, que permite simular el muestreo de datos discreto (cada 4 segundos).

Siguiendo con la descripción del diagrama general (Figura 5.3), en la parte señalada como *2* se muestra un bloque que corresponde al controlador AGC. El diagrama del controlador puede ser visualizado en la figura 5.5. En dicho esquema se observa que tiene como entradas el ACE (que es la salida del diagrama mostrado anteriormente), y las mediciones de potencia de todas las unidades generadoras pasando por un bloque del tipo *Sample and Hold*. Esto permite determinar un valor de ACE y salida del controlador y mínimo y máximo.

Finalmente en la parte *3* del diagrama general se tiene el generador de consignas del AGC, que puede ser visualizado con más detalle en la Figura 5.6. El próposito de este bloque es determinar los factores de participación de cada una de las unidades, teniendo en cuenta el nivel de potencia generada actualmente en cada central, y además, de su rampa de toma de carga (que es un valor fijo dado por la naturaleza de la central).

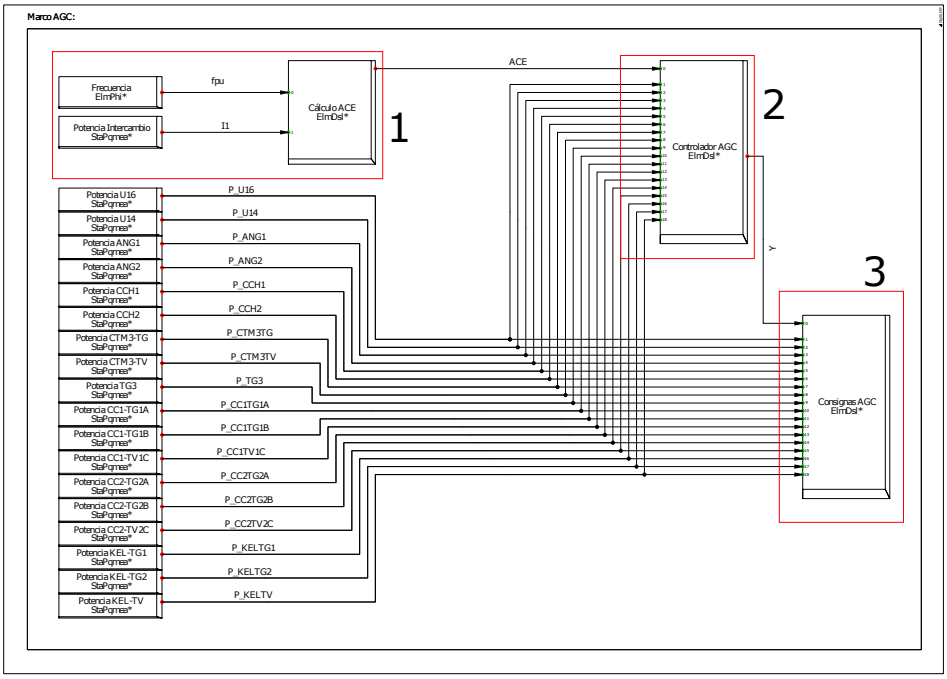


Figura 5.3: Diagrama general de control del sistema en Digsilent.

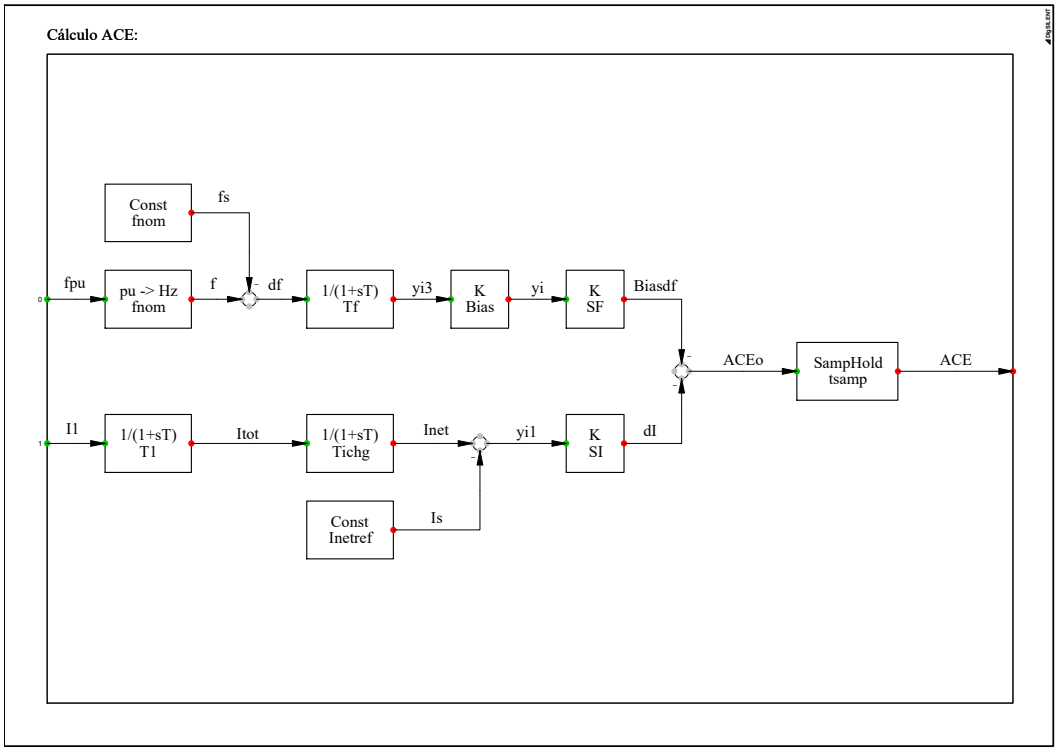


Figura 5.4: Diagrama del Cálculo del ACE.

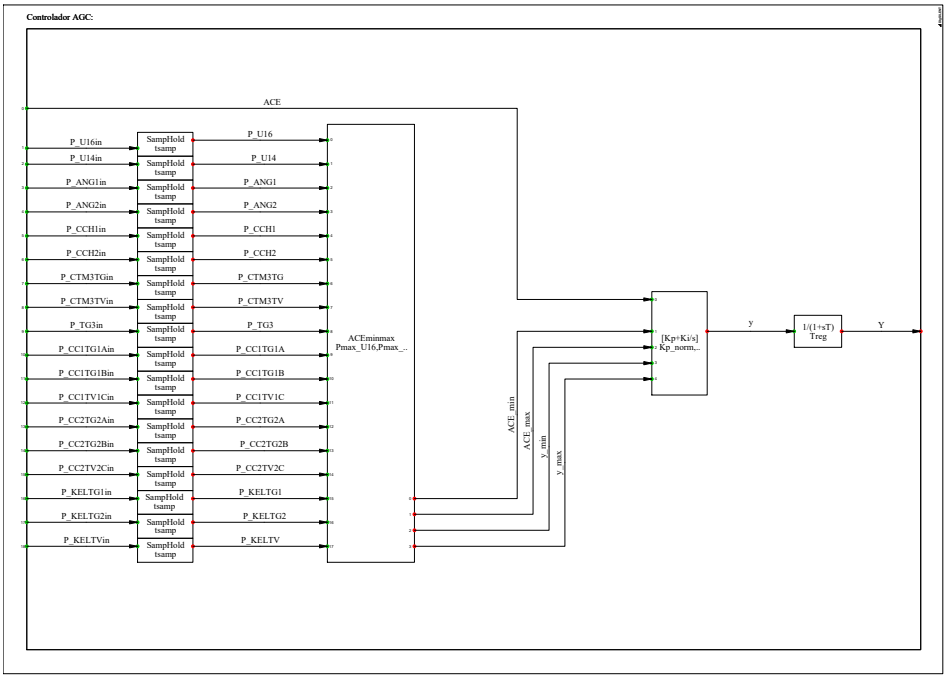


Figura 5.5: Diagrama del controlador.

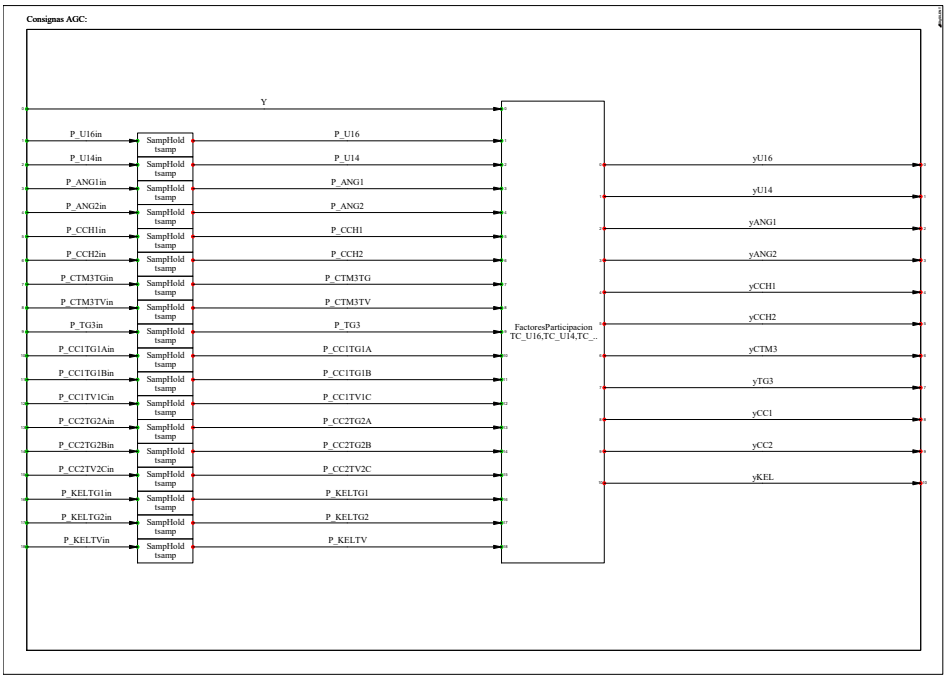


Figura 5.6: Diagrama del generador de consignas para AGC.

5.2. Descripción de los casos

Los casos elegidos se desprenden de un estudio realizado por el AC3E para el CDEC [4]. La idea de este trabajo es determinar el desempeño de los controladores contrastado con el controlador utilizado en dicho estudio, por lo tanto se dejará de lado la reacción de las centrales por sí solas, y se harán comparaciones con los valores de ACE, salida del controlador y la frecuencia del sistema.

Los perfiles de demanda y de generación ERNC en el tiempo pueden ser visualizados de forma detallada en los apéndices C y D, una breve descripción de los casos a considerar sería:

- Caso 1.1 (Alta penetración de ERNC): Todas las ERNC generando.
- Caso 2.1 (Sin ERNC): No se considera potencia generada de parte de las centrales ERNC.
- Caso 3.1 (Entrada ERNC): Centrales aumentan su generación de forma continua.
- Caso 4.1 (Salida ERNC): Centrales disminuyen su generación de forma continua.

Dicho esto, los casos elegidos para el estudio pueden ser visualizados en la tabla 5.1, donde se destacan que unidades participan en la regulación secundaria de frecuencia, y que unidades son elegidas como el slack (éstas también son consideradas como participantes en la regulación de frecuencia).

Unidad	Escenario 1.1	Escenario 2.1	Escenario 3.1	Escenario 4.1
	Despacho [MW]	Despacho [MW]	Despacho [MW]	Despacho [MW]
Cerro Pabellón	32	32	32	32
PAM	17	17	17	17
ANG1	268	268	268	226
ANG2	245	272	245	237
CCH2	222	257	222	222
CTA	165	165	165	165
CCH1	222	251	222	257
CTH	162	162	162	162
U16	137	183.4	200	235
CTM2	154	154	154	165
U15	116	116	116	120
U14	122	83	122	122
CTM1	149	149	149	149
U12	50	80	80	50
U13	X	80	80	50
CTM3	X	X	X	X
GAG TG+0.5TV	X	110	X	X
Kelar TG1	X	X	78	X
NTO1	132	132	132	100
CTTAR	100	140	140	100
NTO2	132	132	132	130
TG3	X	10	X	X
ERNC	368	X	77	224
Despacho Total	2793	2793.4	2793	2793

X = Fuera de Servicio
Unidad en AGC
Unidad Slack

Tabla 5.1: Casos para el sistema sin la consideración del SING-SADI

Se analizarán los resultados considerando las respuestas en frecuencia, el ACE y la salida Y del controlador. Para ello se ha considerado además un filtro pasabajo a la salida del controlador, con un tiempo de:

$$T_{reg} = 45[s]$$

Esto para evitar la acción del controlador ante variaciones grandes de frecuencia, y también evitar la incorrecta o excesiva retroalimentación de los controladores para los casos de ANN y ANFIS.

5.3. Controlador Fuzzy

Para el caso del controlador Fuzzy se tiene que el algoritmo utilizado para optimizar los valores es el mismo que el mencionado en el capítulo 4, o sea, se utiliza el método PSO para optimizar valores claves que definen las funciones de pertenencia.

Dicho esto, además, es necesario determinar con que caso de simulación se hará dicha optimización. Lo ideal en este caso sería hacer un barrido de todos los casos, e ir modificando lentamente los parámetros de modo de llegar a un óptimo. En la práctica esto implicaría un tiempo de simulación significativamente alto, debido a que el modelo a simular toma demasiado tiempo. Por lo tanto, se hace una simplificación, y se toma un caso que vendría a ser el más significativo.

La elección del caso más significativo se explica de forma breve: es en donde la salida correcta del controlador tenga mayor importancia. La explicación es simple, por ejemplo si por el contrario hubiera una caída muy grande y constante de frecuencia, tal que la respuesta del controlador sea simplemente llevar su valor Y a su tope no sería un caso significativo, ya que cualquier controlador que haga lo mismo sería una respuesta óptima para dicha optimización, lo cual llevaría al algoritmo a cualquier valor.

Por lo tanto, usando la misma construcción de funciones de pertenencia vistas en el capítulo 4, o sea las Figuras 4.6, 4.7 y 4.8, y dependientes de los parámetros b_{ACE}, $b_{\Delta ACE}$ y b_{out} que son los valores a optimizar, se tiene que con 15 iteraciones el óptimo se encuentra con los valores mostrados en la tabla 5.2:

b_{ACE}	$b_{\Delta ACE}$	b_out
0.518	0.296	0.252

Tabla 5.2: Valores de los parámetros que definen las funciones de pertenencia del controlador difuso.

El comportamiento de dicho controlador puede ser visualizado de mejor forma analizando el mapa de calor, que se presenta en la Figura 5.7. En dicha figura se pueden observar las decisiones que realiza el controlador ante cierto valor de ACE (eje horizontal) y de ΔACE, donde el color amarillo representa valores negativos, y el color rojo valores positivos.

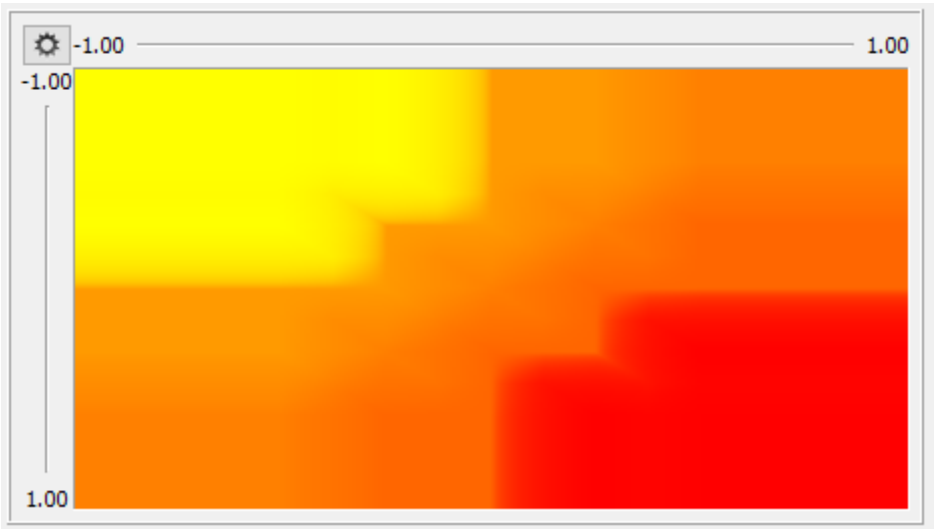


Figura 5.7: Mapa de calor del controlador difuso optimizado.

5.3.1. Caso 1.1: Alta penetración de ERNC y Salida ANG1

Para este caso las unidades en AGC son: CCH1, CCH2 y ANG2, y corresponden a un caso con alta penetración de ERNC y la salida de ANG1.

Las simulaciones se presentan a continuación en las Figuras 5.8, 5.9 y 5.10.

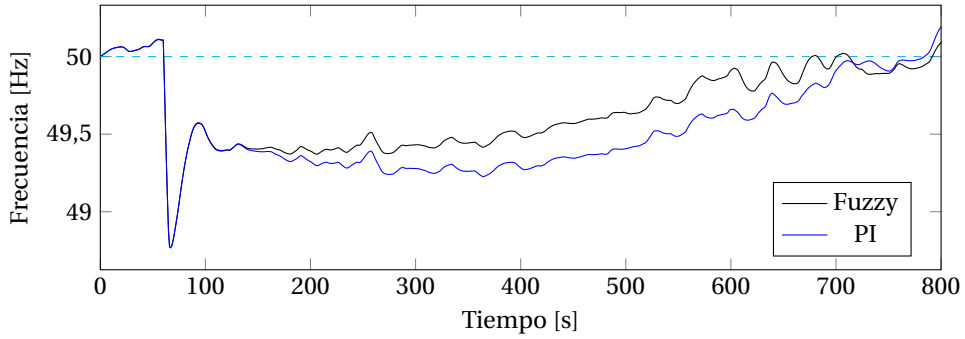


Figura 5.8: Frecuencia en Hz del caso 1.1.

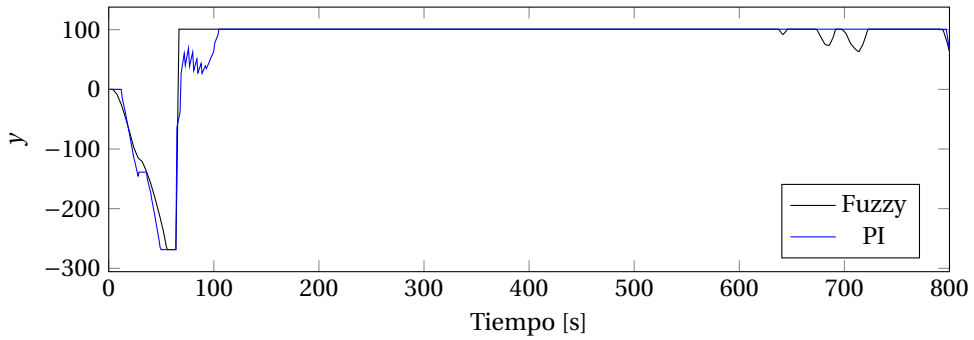


Figura 5.9: y del caso 1.1.

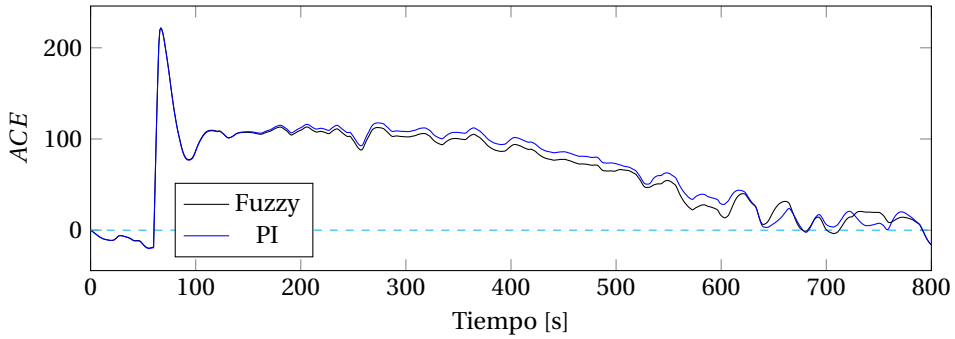
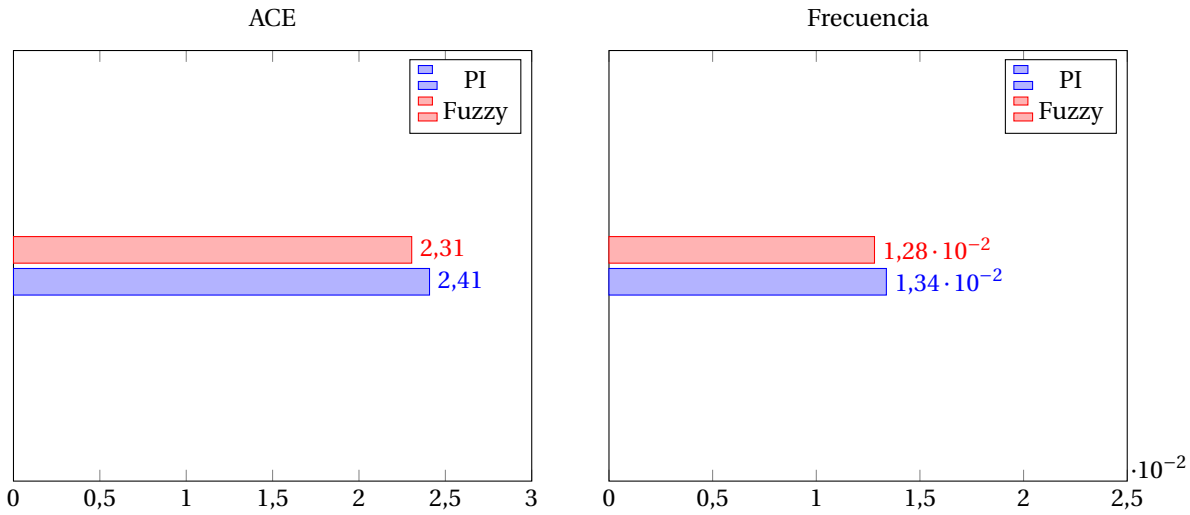


Figura 5.10: ACE del caso 1.1.

Se puede observar en este primer caso en la Figura 5.10 que hay una caída del ACE en un primer lugar. Esto debido a la subida de frecuencia provocada por la generación intermitente de las ERNC. Dicha frecuencia puede ser visualizada en la Figura 5.8.

Los valores del ACE previos a la caída del generador ANG1 provocan que los controladores no respondan rápidamente, debido a su naturaleza integradora y los filtros de alta frecuencia dispuestos a su salida. Sin embargo, el controlador difuso logra remontar dicho cambio de forma más oportuna, llevando el valor de salida del controlador hasta su tope segundos antes que su contraparte. Más específicamente se observa que el controlador PI empieza a subir su salida cercano a los 90 segundos, mientras que el controlador Fuzzy lo hace cercano a los 60 segundos, lo cual en términos generales genera una diferencia sustancial en la regulación pasados los 120 segundos.

Al cuantificar el desempeño de ambos controladores se obtienen las Figuras 5.11a y 5.11b. La Figura 5.11a permite determinar que efectivamente a lo largo del tiempo de simulación, el controlador difuso logró mejores resultados. Tuvo un desempeño 4.32 % superior para el caso del ACE.



(a) Resultados respectivos al ACE del caso 1.1 con la salida de ANG1.

(b) Resultados respectivos a la frecuencia del caso 1.1 con la salida de ANG1.

Figura 5.11

5.3.2. Caso 1.1: Alta penetración de ERNC y Salida U15

Considerando el mismo caso anterior, pero ahora con la salida del generador U15, se obtuvieron los resultados mostrados en las Figuras 5.12, 5.13 y 5.14.

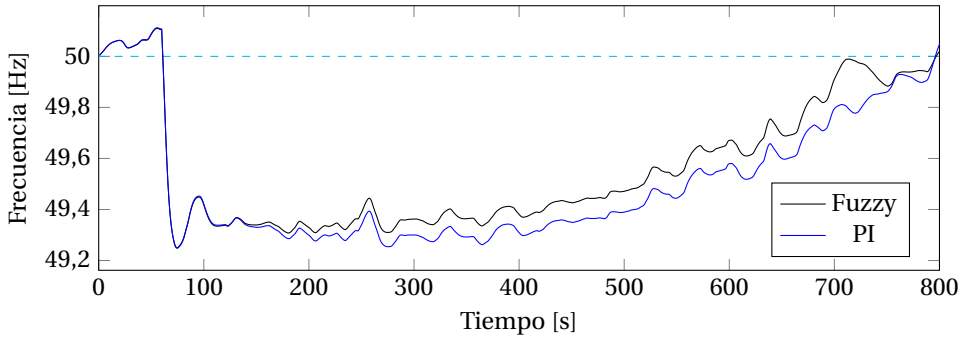


Figura 5.12: Frecuencia en Hz del caso 1.1.

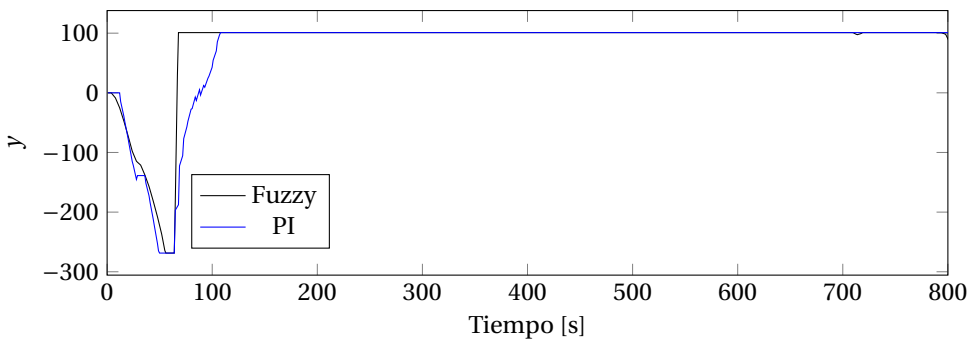


Figura 5.13: y del caso 1.1.

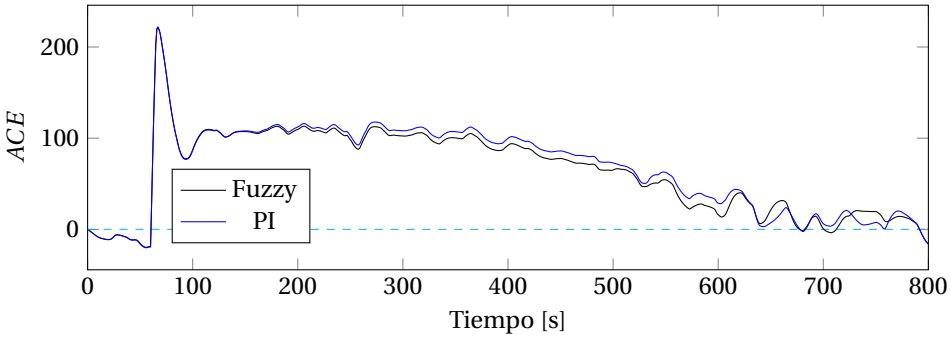
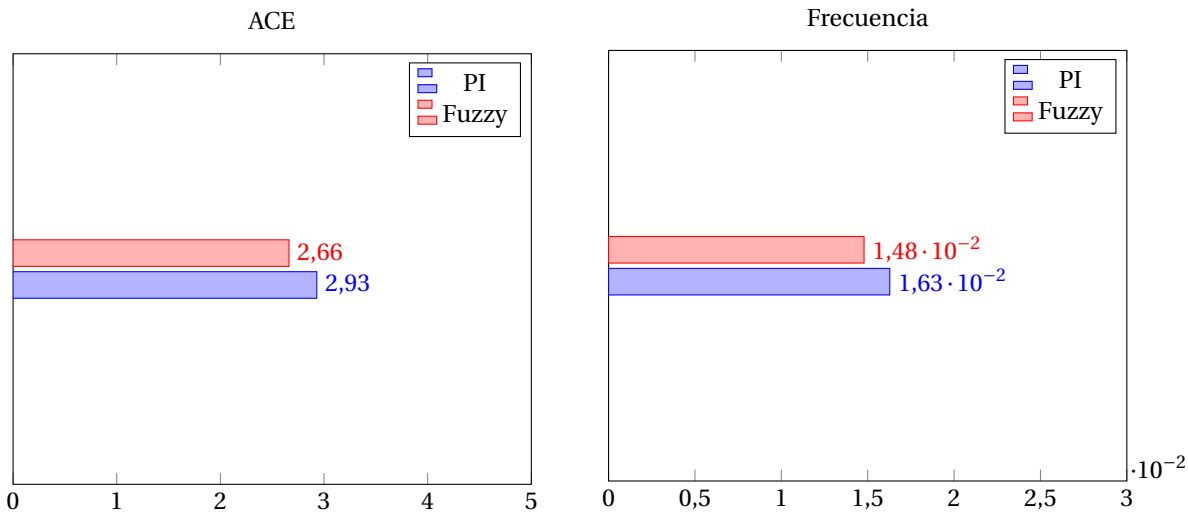


Figura 5.14: *ACE* del caso 1.1.

Se tiene algo similar a lo analizado anteriormente. El controlador Fuzzy sube inmediatamente ante la falla, lo cual, al ser más oportuno, permite al sistema recuperar la frecuencia de modo más rápido.

Cuantificando los efectos de ambos controladores, se tiene que para el caso del ACE el controlador Fuzzy fue 33.45 % mejor que su contraparte PI.



(a) Resultados respectivos al ACE del caso 1.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 1.1 con la salida de U15.

Figura 5.15

5.3.3. Caso 2.1: Sin ERNC y salida ANG2

Igual que en la simulación anterior, se tiene que las unidades en AGC son: CCH1, CCH2, U16 y GAG TG+0.5TV, y presenta la salida de la unidad generadora ANG2. Las simulaciones para dicho caso se presentan en las Figuras 5.16, 5.17, y 5.18.

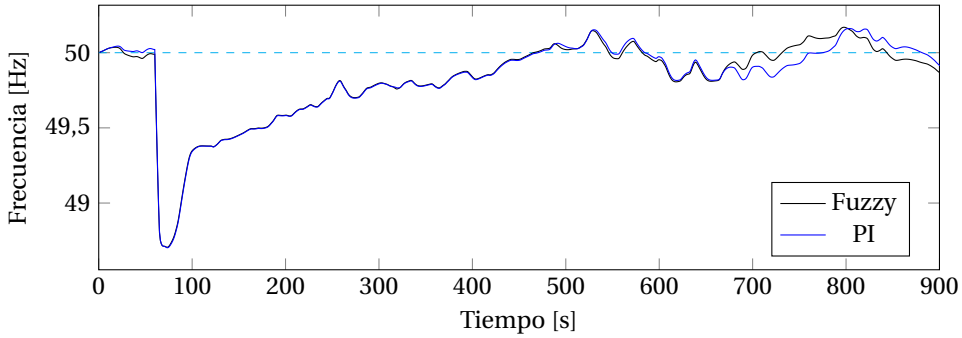


Figura 5.16: Frecuencia en Hz del caso 2.1.

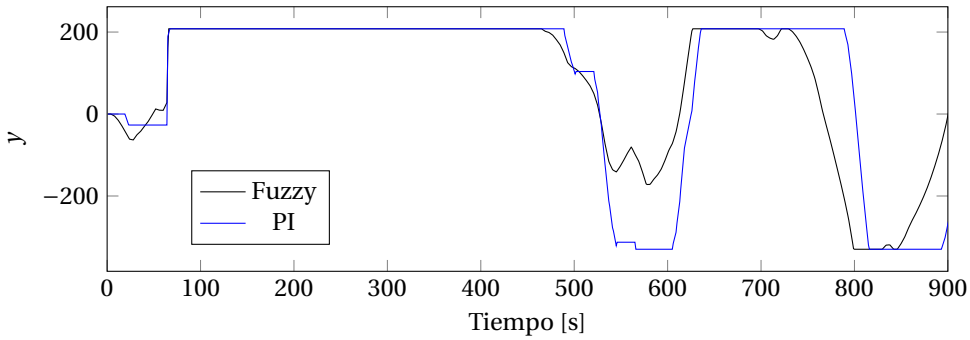


Figura 5.17: y del caso 2.1.

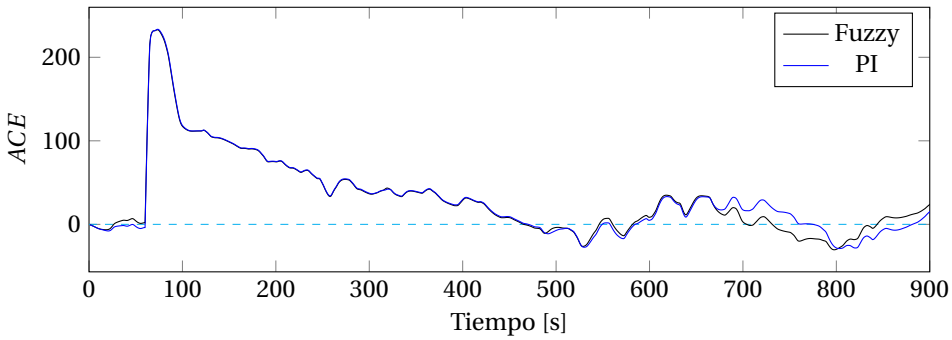
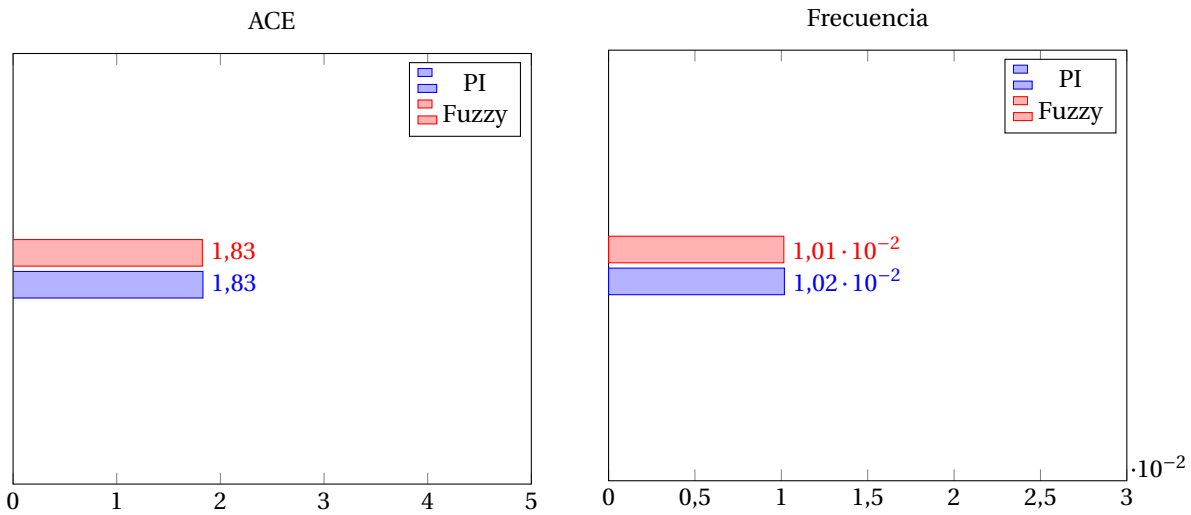


Figura 5.18: ACE del caso 2.1.

Para este caso se puede observar que la frecuencia en ambos casos se comporta de forma similar (Figura 5.16), dado que ambos controladores suben y bajan potencia de forma parecida.

Entre los 600 y 700 segundos se notan algunas diferencias: el controlador difuso baja la consigna de potencia en menor proporción que como lo hace su contraparte, el controlador PI. Esto provoca que en el momento en que el ACE cambia de signo, el controlador difuso puede subir de forma más rápida al valor máximo, sobrepasando la frecuencia nominal, movimiento el cual también se produce en el controlador PI, pero algunos segundos más tarde. No obstante, ambos controladores oscilan en torno a estos valores con lo que se tienen resultados similares.

Cuantificando el rendimiento de ambos controladores a través de la integral del ACE^2 y de la desviación de frecuencia, representados graficamente en las Figuras 5.43a y 5.43b respectivamente, se puede observar que ambos tuvieron un rendimiento idéntico (aunque los gráficos hayan sido ligeramente diferentes).



(a) Resultados respectivos al ACE del caso 2.1 con la salida de ANG2.

(b) Resultados respectivos a la frecuencia del caso 2.1 con la salida de ANG2.

Figura 5.19

5.3.4. Caso 2.1: Sin ERNC y salida U15

Para este caso, como fue presentado en la tabla 5.1, se tiene que las unidades en AGC son: CCH1, CCH2, U16 y GAG TG+0.5TV, y presenta la salida de la unidad generadora U15. Las simulaciones para dicho caso se presentan en las Figuras 5.20, 5.21, y 5.22.

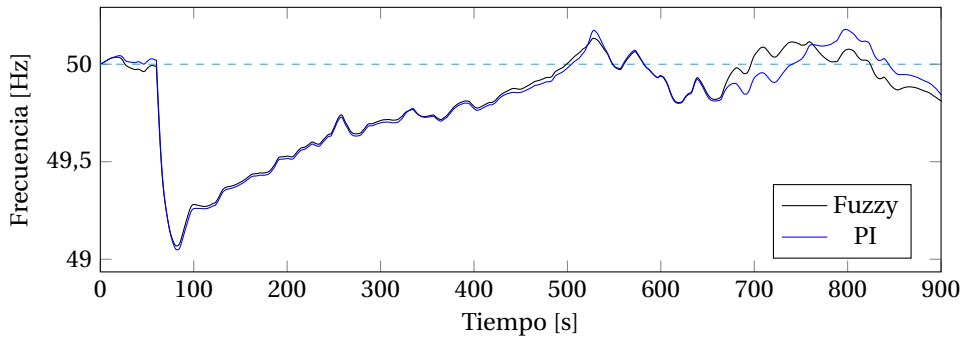


Figura 5.20: Frecuencia en Hz del caso 2.1.

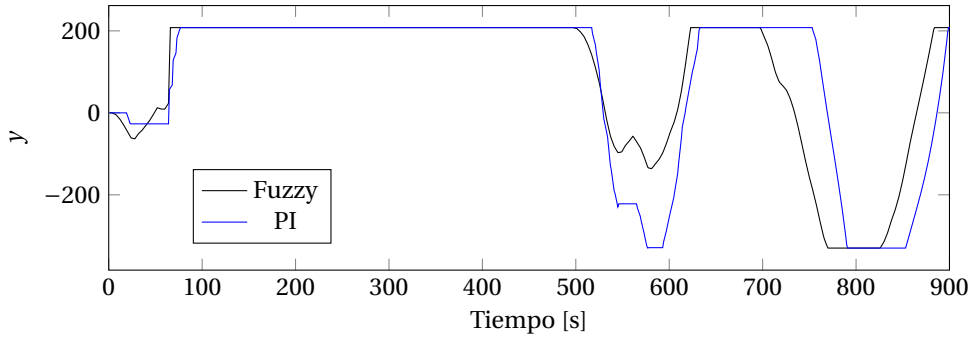


Figura 5.21: y del caso 2.1.

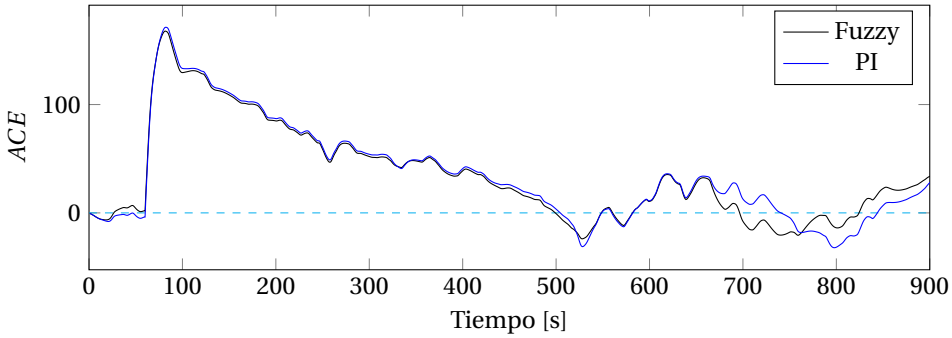
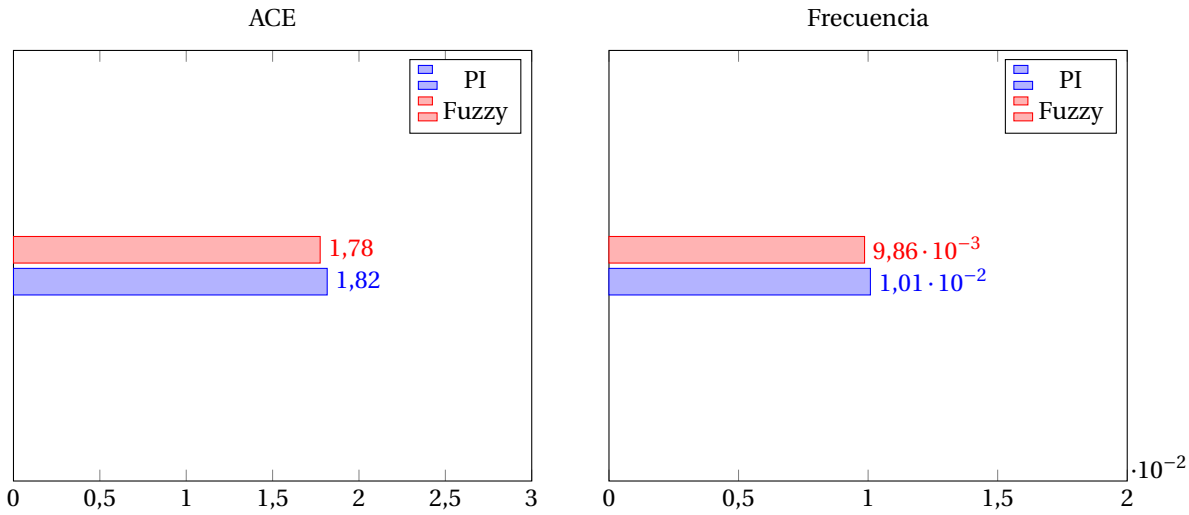


Figura 5.22: *ACE* del caso 2.1.

Se puede observar que antes de la salida del generador U15, hay una regulación de frecuencia un poco superior de parte del controlador Fuzzy, debido a una mayor reacción del controlador en ese período (una mayor caída de y). Luego se visualiza un comportamiento similar hasta el segundo 650, donde el controlador PI baja dramáticamente el valor de y debido al cambio de sus constantes (K_p y K_i), claramente afectado por el paso por 0 del ACE. Mientras que el controlador Fuzzy altera su comportamiento (baja), pero no de forma tan súbita, lo que permite que desde ese instante hasta cercano al segundo 650, los comportamientos de ambos controladores sean diferentes, siendo el controlador Fuzzy superior manteniendo la frecuencia en un valor más cercano al nominal.

Cuantificando el rendimiento de ambos controladores, se tiene que finalmente el controlador Fuzzy fue 2.24 % superior que el PI.



(a) Resultados respectivos al ACE del caso 2.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 2.1 con la salida de U15.

Figura 5.23

5.3.5. Caso 3.1: Entrada de ERNC y salida U15

En este caso las unidades que están en AGC son: CCH1, CCH2 y ANG2, y corresponde a la entrada de las ERNC.

Las simulaciones correspondientes están presentadas en las Figuras 5.24, 5.25, 5.26.

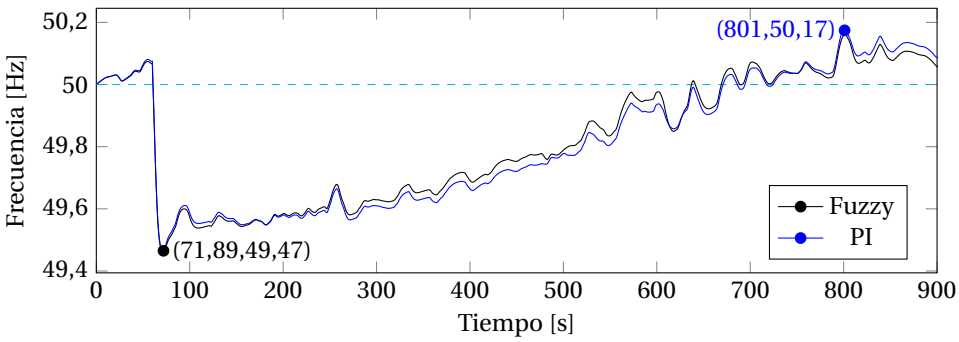


Figura 5.24: Frecuencia en Hz del caso 3.1.

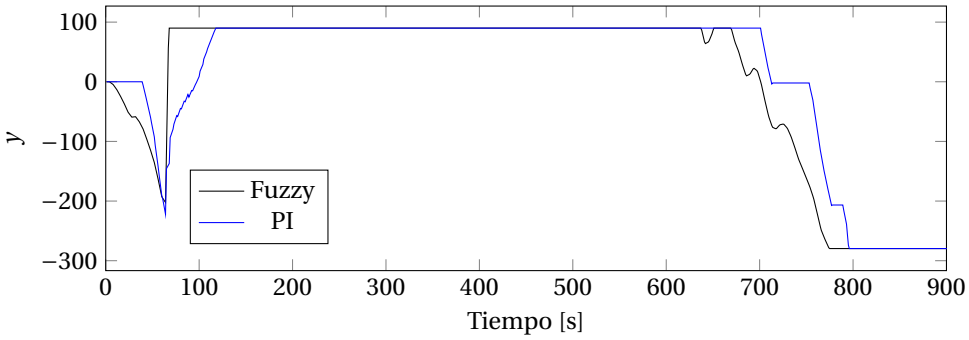


Figura 5.25: y del caso 3.1.

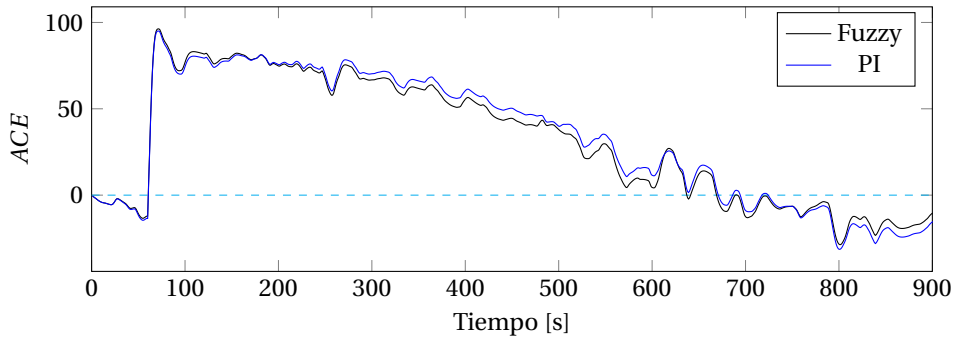


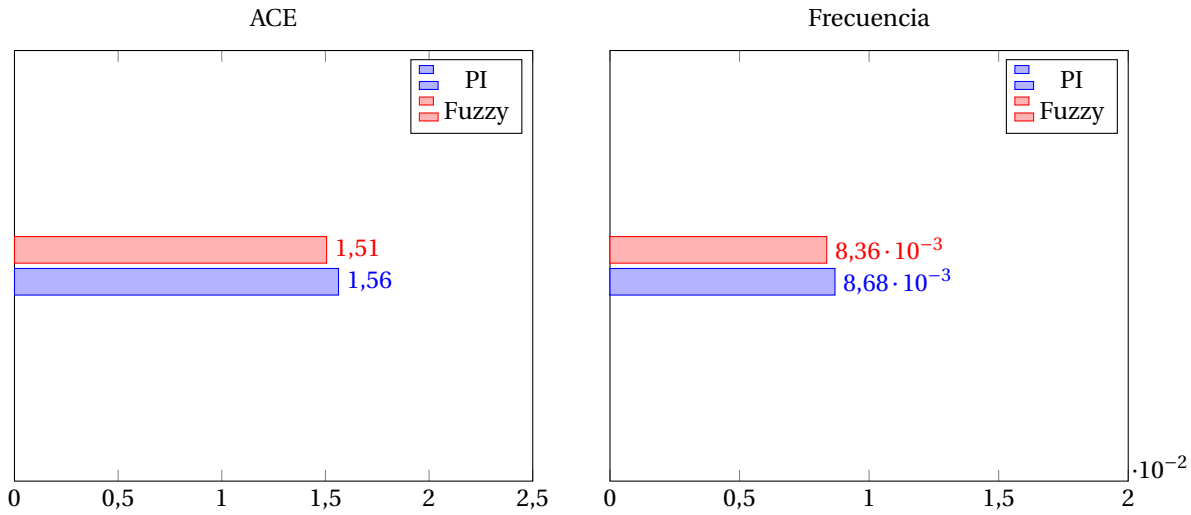
Figura 5.26: ACE del caso 3.1.

Se puede observar en el gráfico de frecuencia (Figura 5.24) que se llega a valores mínimos de frecuencia (49.47 [Hz]) en cualquiera de los controladores, lo cual es normal, debido a que el efecto de la salida de la central es demasiado rápido como para actuar en ese momento (además que no es la intención tampoco de la regulación secundaria de frecuencia).

Se nota también en el gráfico de salida del controlador (Figura 5.25) que el controlador Fuzzy responde de manera sumamente rápida ante el cambio de signo del ACE producido cercano a los 50 segundos, que es cuando cae el generador U15. Esto mientras el controlador PI se demora en reponer su salida un poco más, lo que genera resultados levemente mejores de parte del controlador Fuzzy.

Además de esto, se puede visualizar que después de los 650 segundos aproximadamente hay una caída más temprana en la salida del controlador Fuzzy, lo cual ayuda a obtener resultados un poco mejores dentro de ese rango.

Finalmente, se puede cuantificar el efecto de ambos controladores al ver el gráfico de la integral del ACE^2 en la Figura 5.27a. Esto señala que el controlador Fuzzy fue 3.31 % superior a su contraparte, el controlador PI.



(a) Resultados respectivos al ACE del caso 3.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 3.1 con la salida de U15.

Figura 5.27

5.3.6. Caso 4.1: Salida de ERNC y salida ANG1

Para este caso las unidades en AGC son: ANG1, ANG2 y CCH2, y considera la salida de ERNC además de la caída de ANG1 como central generadora.

Las simulaciones realizadas se pueden visualizar en las Figuras 5.28, 5.29 y 5.30.

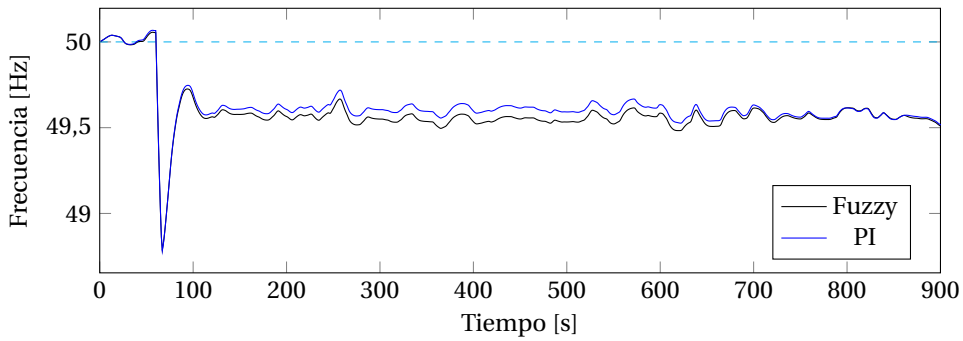


Figura 5.28: Frecuencia en Hz del caso 4.1.

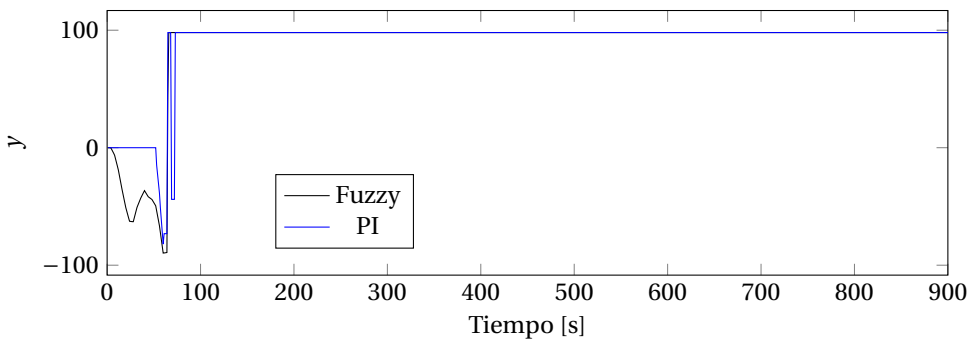


Figura 5.29: y del caso 4.1.

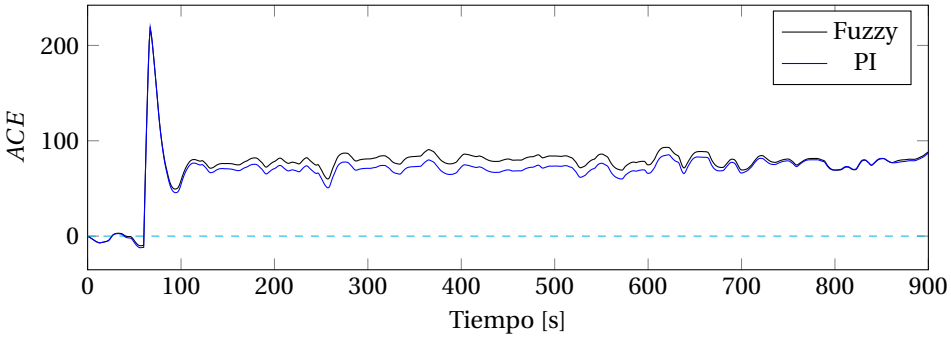
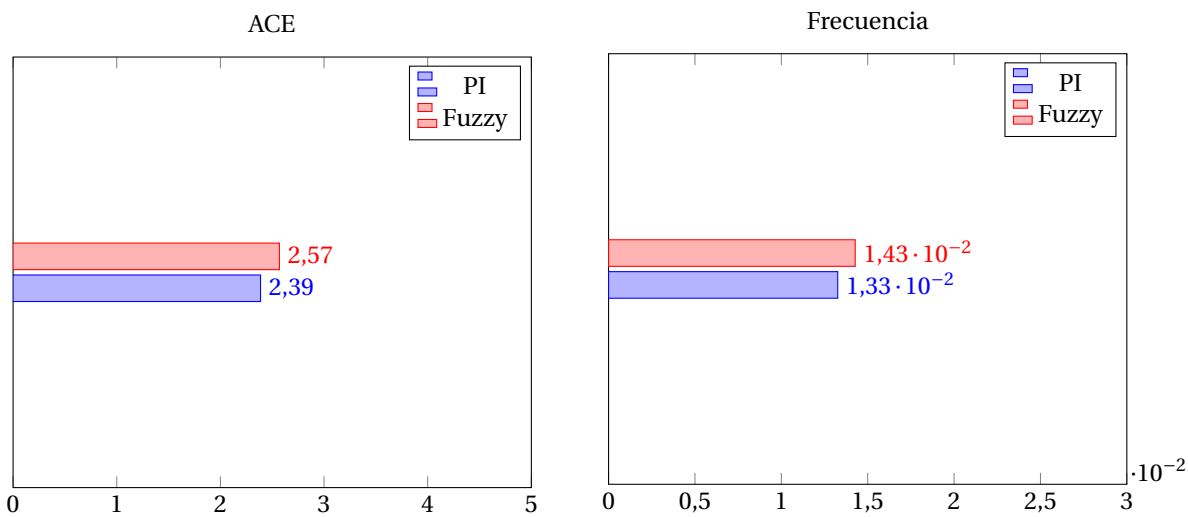


Figura 5.30: *ACE* del caso 4.1.

Observando los resultados se tiene que la frecuencia llega a su máximo debido al efecto de la demanda y la salida de ERNC, ya que ocurre antes de la salida de ANG1.

Este escenario no presenta mucho interés debido a que la respuesta del controlador es llevar a los generadores a su potencia máxima para abastecer la falta de generación. Por lo tanto, la mejor respuesta será del controlador que haga dicho paso de forma más rápida, que en este caso es el controlador PI.

Al cuantificar los resultados obteniendo la integral del ACE^2 y de la desviación de frecuencia mostrados en las Figuras 5.31a y 5.31b respectivamente, se tiene que el controlador difuso tiene un rendimiento 7.53 % menor que su contraparte el controlador PI.



(a) Resultados respectivos al ACE del caso 4.1 con la salida de ANG1.

(b) Resultados respectivos a la frecuencia del caso 4.1 con la salida de ANG1.

Figura 5.31

5.4. Controlador ANN

En esta sección se mostrarán los mismos casos anteriores, pero esta vez utilizando el controlador basado en redes neuronales descrito en los capítulos anteriores.

5.4.1. Caso 1.1: Alta penetración de ERNC y salida ANG1

Las simulaciones se presentan a continuación en las Figuras 5.32, 5.33 y 5.34.

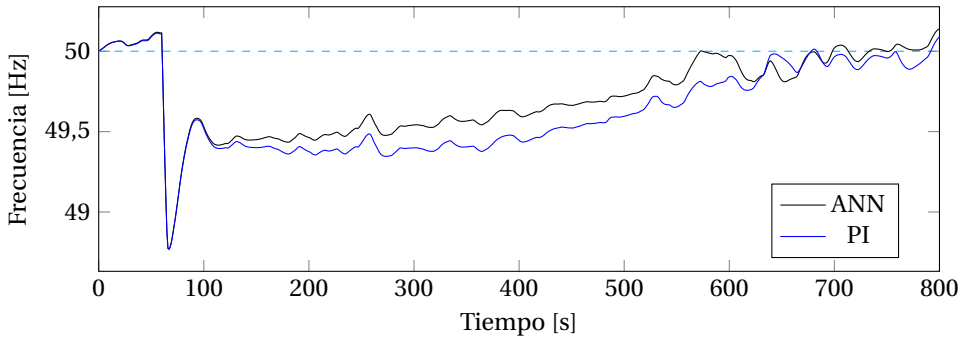


Figura 5.32: Frecuencia en Hz del caso 1.1.

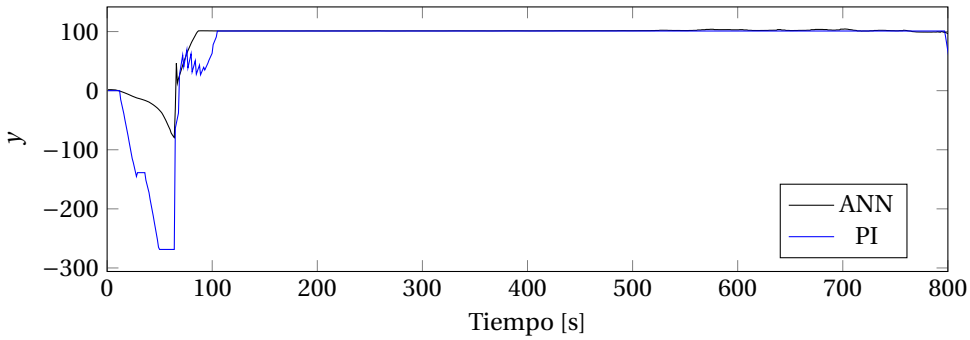


Figura 5.33: y del caso 1.1.

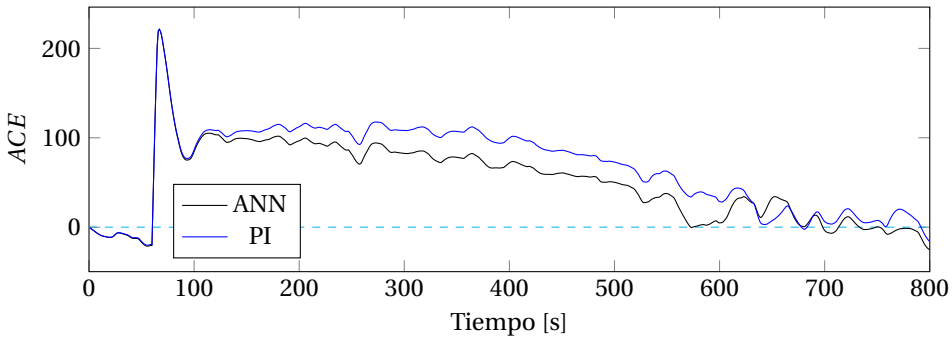
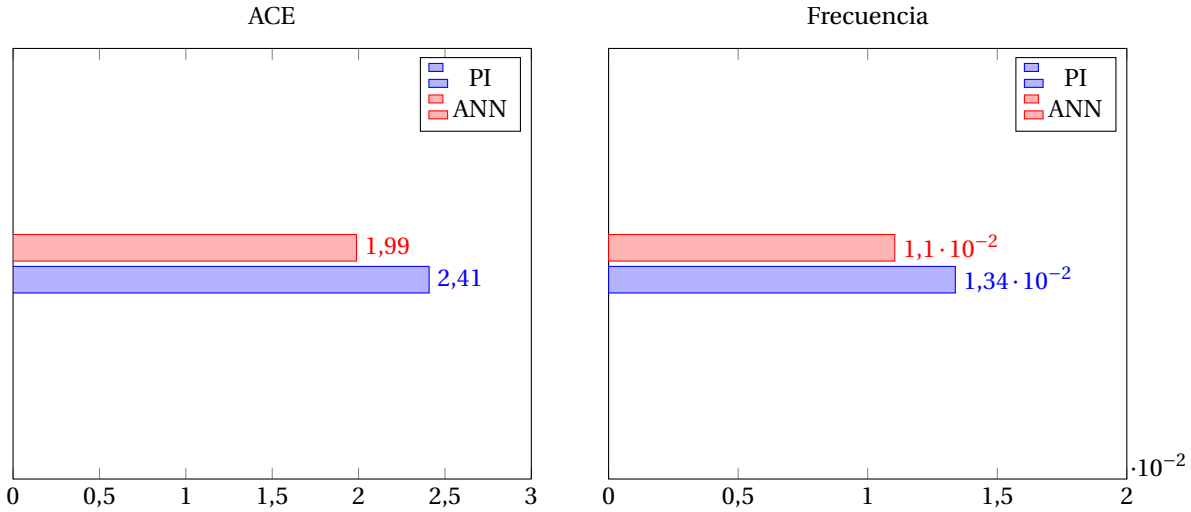


Figura 5.34: *ACE* del caso 1.1.

Se puede observar en el gráfico de la frecuencia (Figura 5.32) y de la salida del controlador (Figura 5.33), que existe regulación de frecuencia de parte de los dos controladores ante la variabilidad de la misma instantes antes de la caída de la central ANG1. Pero luego de dicho evento, el ACE cambia inmediatamente de signo, y la frecuencia cae a su mínimo, lo cual hace funcionar los EDAC que de forma casi instantánea recuperan parte de la caída. Sin embargo, el controlador ANN no reaccionó de forma exagerada previo a la caída, lo que le permitió recuperar su salida de forma más oportuna que su contraparte, el controlador PI. Esto a grandes

rasgos, fue clave para tener un mejor rendimiento de ahí en adelante, donde se puede ver claramente que en el controlador ANN el valor de frecuencia estuvo mucho más cerca del nominal.

Con este controlador, la acción de no sobrerreaccionar antes de la caída le permitió al sistema tener un mejor rendimiento. Al cuantificar dicho rendimiento se tiene que el controlador ANN fue 21.10 % superior a su contraparte PI, lo cual se puede ver claramente en los gráficos de los resultados, las Figuras 5.35a y 5.35b.



(a) Resultados respectivos al ACE del caso 1.1 con la salida de ANG1.

(b) Resultados respectivos a la frecuencia del caso 1.1 con la salida de ANG1.

Figura 5.35

5.4.2. Caso 1.1: Alta penetración de ERNC y Salida U15

Visualizando ahora la caída del generador U15 con el mismo caso anterior, se tienen los valores de frecuencia y salida del controlador en las Figuras 5.36 y 5.37.

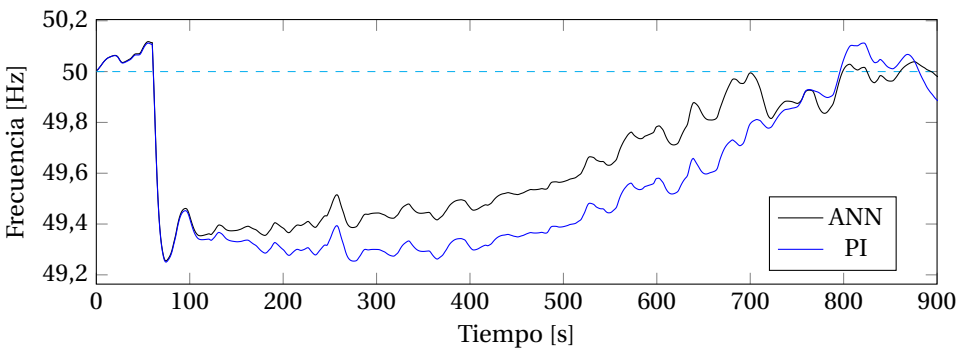


Figura 5.36: Frecuencia en Hz del caso 1.1.

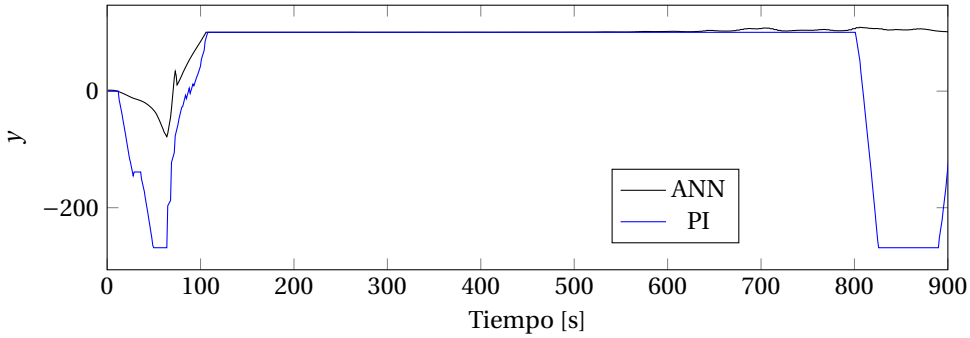


Figura 5.37: y del caso 1.1.

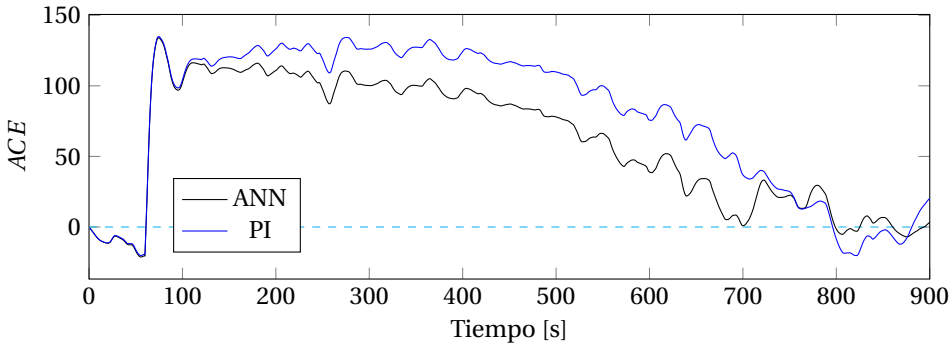
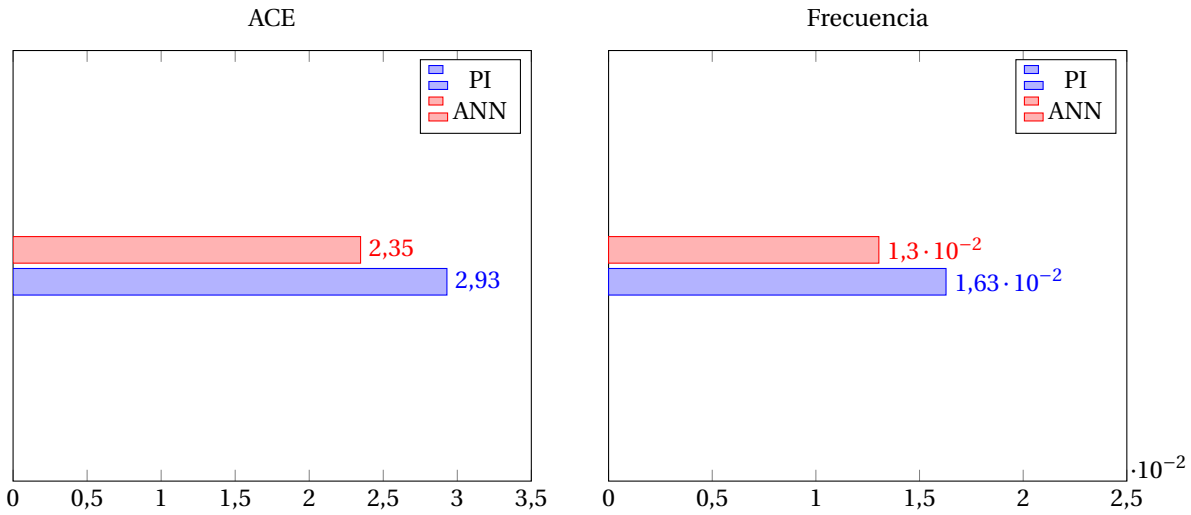


Figura 5.38: ACE del caso 1.1.

Para este caso, se presentan condiciones similares al visto anteriormente. Se tiene que previo a la caída del generador el controlador PI sobrerreaciona y lleva la salida al valor mínimo de generación, lo que provoca que al salir la unidad U15 demore más tiempo en subir, lo cual le da una clara ventaja al controlador ANN.

Se observa además, que cercano a los 800 segundos la frecuencia sobrepasa los 50 [Hz], lo que en el controlador PI hace bajar su generación de forma súbita, mientras que el controlador ANN rebaja su generación, pero de forma menos abrupta, lo que le permite mantener niveles de frecuencia más estables.

Estos efectos acumulados en el tiempo, permitieron al controlador ANN obtener un rendimiento 24.68 % superior al de su contraparte el controlador PI.



(a) Resultados respectivos al ACE del caso 1.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 1.1 con la salida de U15.

Figura 5.39

5.4.3. Caso 2.1: Sin ERNC y salida ANG2

Las simulaciones para dicho caso se presentan en las Figuras 5.40, 5.41, y 5.42.

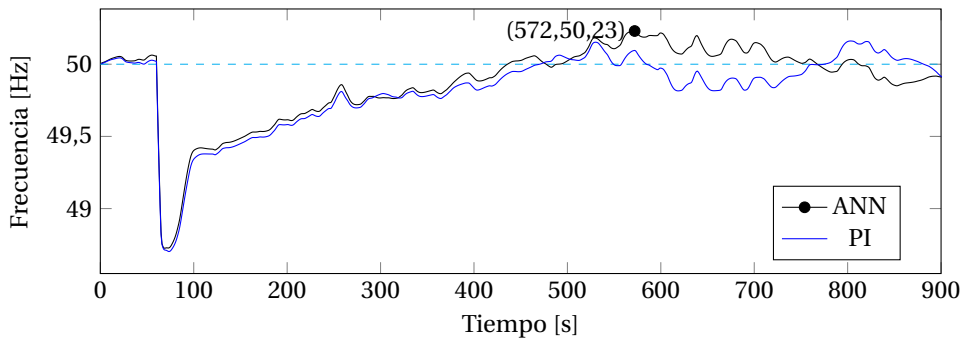


Figura 5.40: Frecuencia en Hz del caso 2.1.

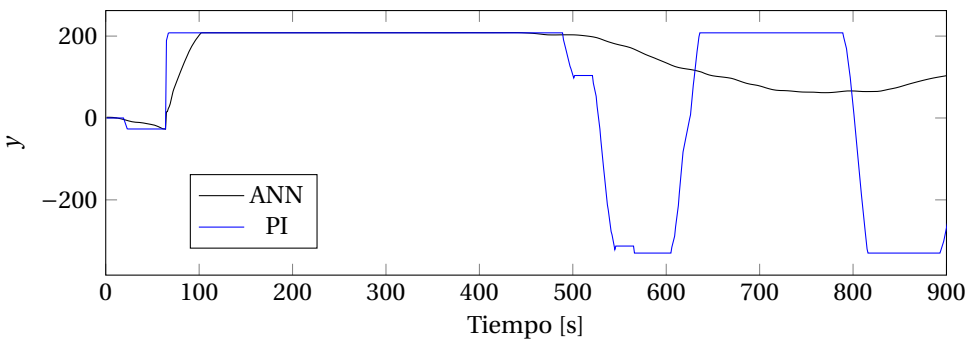


Figura 5.41: y del caso 2.1.

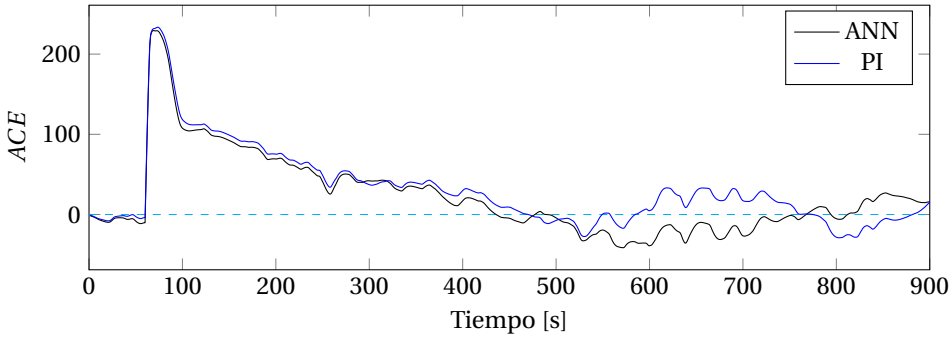
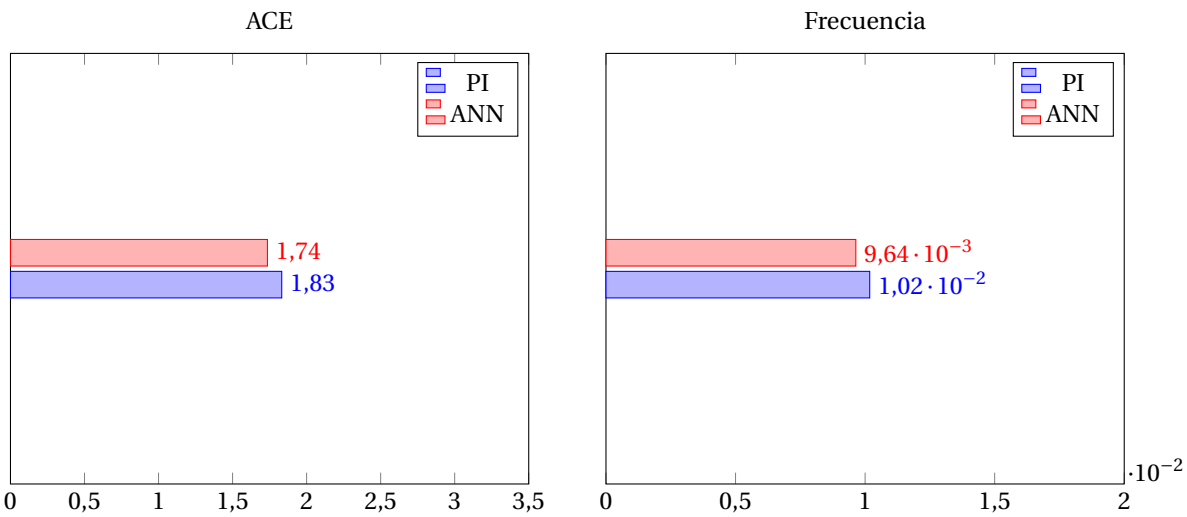


Figura 5.42: *ACE* del caso 2.1.

Para este caso se observa en el gráfico de frecuencia (Figura 5.40) que el valor máximo se presenta para el caso del controlador ANN, siendo de 50.23[Hz] a los 572 segundos. Sin embargo, ambos controladores rebasan el valor nominal haciendo que el ACE cambie de signo. Seguido de esto la salida del controlador PI cae significativamente, mientras que el controlador ANN lo hace de menor forma, lo cual le permite a este último tener un mejor rendimiento al considerar toda la simulación, además de no ser afectado por este funcionamiento oscilatorio.

Al cuantificar los rendimientos obteniendo la integral del ACE^2 y de la desviación se frecuencia, se tienen los resultados mostrados en las Figuras 5.43a y 5.43a correspondientemente. De dichas figuras se puede determinar que el controlador ANN fue 5.17 % superior ante su contraparte el controlador PI.



(a) Resultados respectivos al ACE del caso 2.1 con la salida de ANG2.

(b) Resultados respectivos a la frecuencia del caso 2.1 con la salida de ANG2.

Figura 5.43

5.4.4. Caso 2.1: Sin ERNC y salida U15

Las simulaciones para dicho caso se presentan en las Figuras 5.16, 5.41, y 5.42.

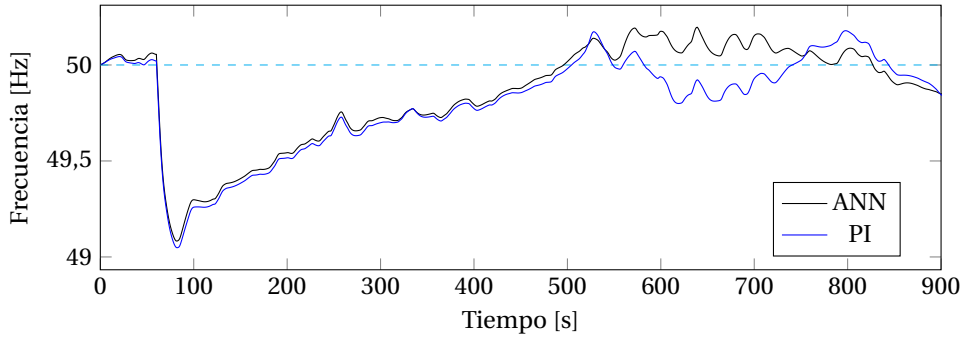


Figura 5.44: Frecuencia en Hz del caso 2.1.

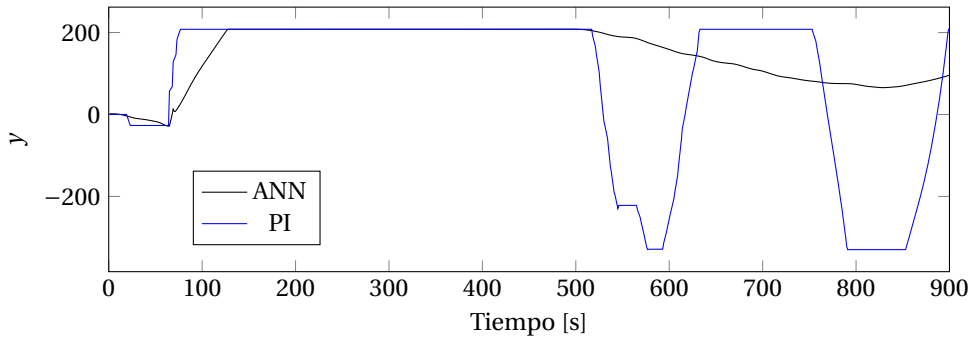


Figura 5.45: y del caso 2.1.

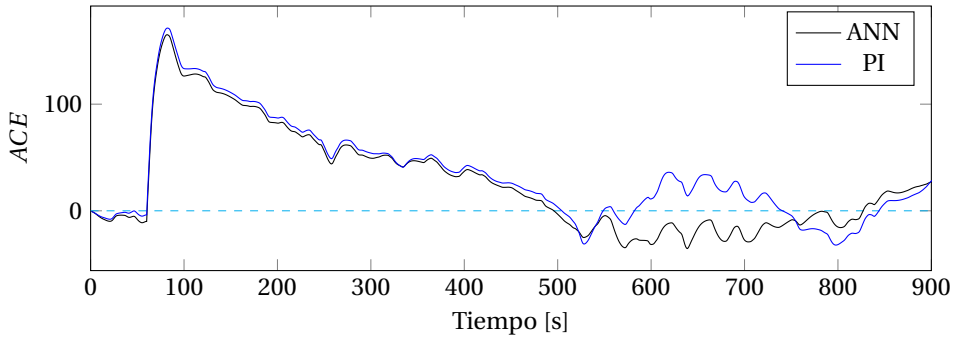
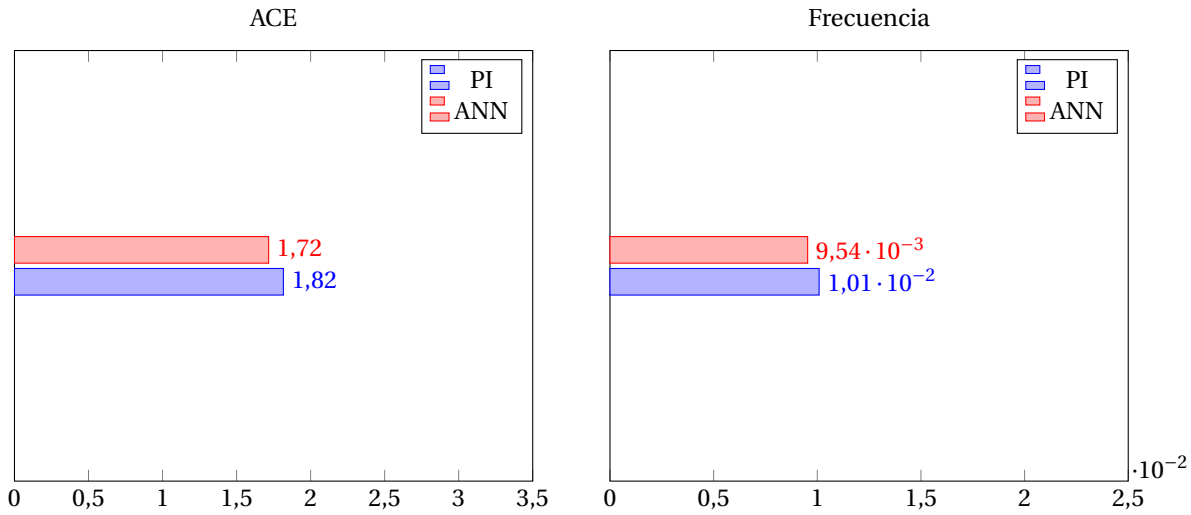


Figura 5.46: ACE del caso 2.1.

Se observa del escenario que la regulación de frecuencia previo a la caída de la generadora es mínima y casi nula debido al ACE tan pequeño (está en la banda muerta de hecho por unos instantes, con lo cual si es nula).

Ya al caer la central U15, se ve que ambos llevan la generación a su máximo de forma inmediata, lo cual produce que las simulaciones den valores similares en ambos casos hasta cercanos los 550 segundos, donde la frecuencia excede los 50[Hz], haciendo que el ACE cambie de signo y por lo tanto los controladores decidan disminuir generación. Sin embargo, el controlador PI lo hace de forma abrupta, mientras que el ANN hace una caída suave, lo cual evita el comportamiento oscilatorio ocurrido por el caso PI entre los 500 y 900 segundos.

Evitar ese comportamiento oscilatorio gracias a no sobrerreaccionar, le permitió al controlador ANN obtener un rendimiento 5.81 % superior comparado con su contraparte, el controlador PI. Dicho rendimiento puede observarse mediante el valor de la integral de ACE^2 y de la diferencia de frecuencia en las Figuras 5.47a y 5.47b correspondientemente.



(a) Resultados respectivos al ACE del caso 1.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 1.1 con la salida de U15.

Figura 5.47

5.4.5. Caso 3.1: Entrada de ERNC y salida U15

Las simulaciones correspondientes a este escenario están presentadas a continuación en las Figuras 5.48, 5.49 y 5.50.

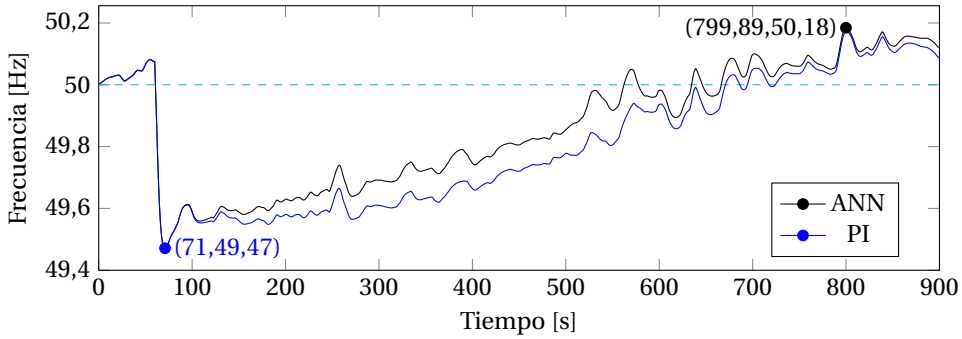


Figura 5.48: Frecuencia en Hz del caso 3.1.

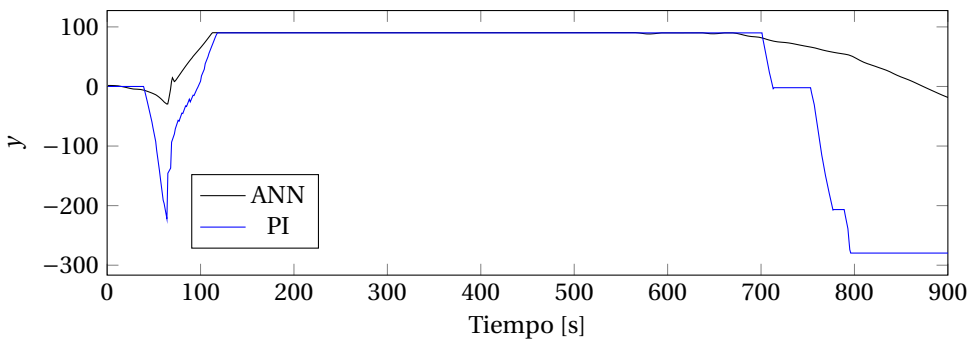


Figura 5.49: y del caso 3.1.

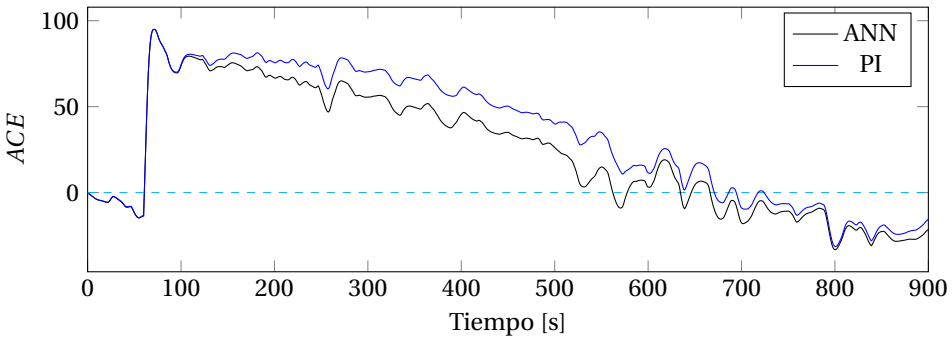


Figura 5.50: *ACE* del caso 3.1.

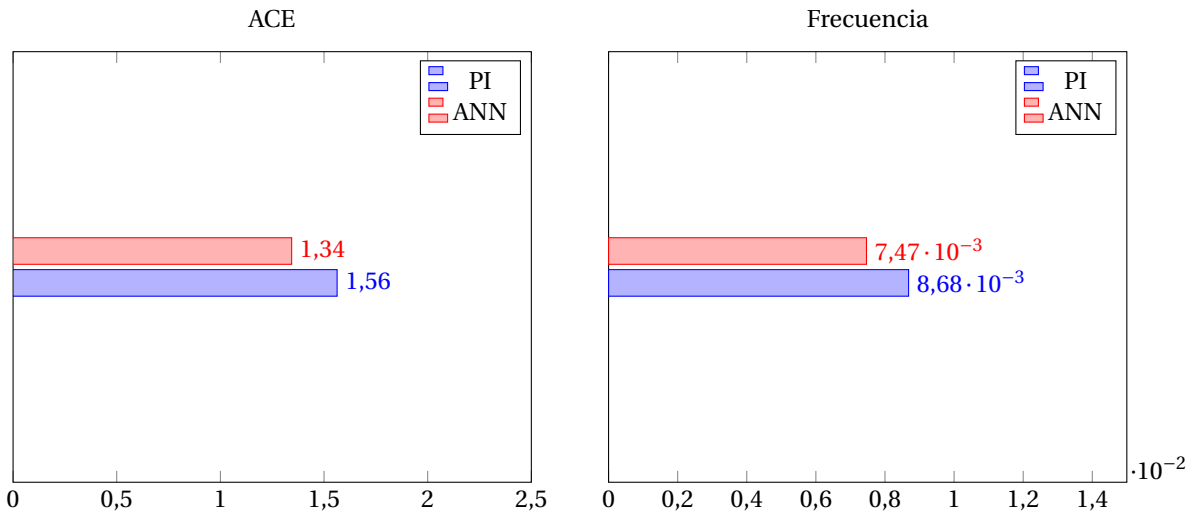
De la figura 5.48 se observa que antes de la caída del generador U15 hay variaciones de frecuencia debidos a la variabilidad de la demanda y la entrada de ERNC.

Debido a esto la salida del controlador PI cae súbitamente, mientras que el controlador ANN no ajusta demasiado su salida. Al salir la unidad generadora U15 se pierde generación, lo cual hace que la frecuencia caiga hasta su punto más bajo (49.47 [Hz]), lo que produce a su vez que el ACE cambie de signo y suba a niveles altos, provocando que los controladores suban las consignas de potencia. Sin embargo, debido a la salida existente previo a la falla, el controlador PI sube desde un punto más bajo, dándole ventaja al controlador ANN, que logra imponer una consigna de la máxima potencia en un tiempo inferior.

Estos cambios finalmente permiten al controlador ANN obtener un mejor rendimiento hasta el segundo 800, donde se llega al valor máximo de frecuencia (50.18[Hz]) para ambos controladores.

También se observa del ACE en la figura 5.50, que desde el segundo 700 cambia de signo (pasa a ser negativo), con lo que los controladores deben bajar sus consignas. Analizando la figura de las salidas de los controladores (Figura 5.49), se nota una diferencia abrupta de parte del caso PI, lo cual le permite mejores resultados pasado ese punto, pero no lo suficiente como para compararse con el controlador ANN.

Cuantificando los resultados y obteniendo las integrales del ACE^2 y de la desviación de frecuencia se tienen las Figuras 5.51a y 5.51b. Lo cual finalmente indica que el controlador ANN tuvo un rendimiento 16.42 % superior al de su contraparte, el controlador PI.



(a) Resultados respectivos al ACE del caso 3.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 3.1 con la salida de U15.

Figura 5.51

5.4.6. Caso 4.1: Salida de ERNC y Salida ANG1

Las simulaciones correspondientes a este escenario están presentadas a continuación en las Figuras 5.52, 5.53 y 5.54.

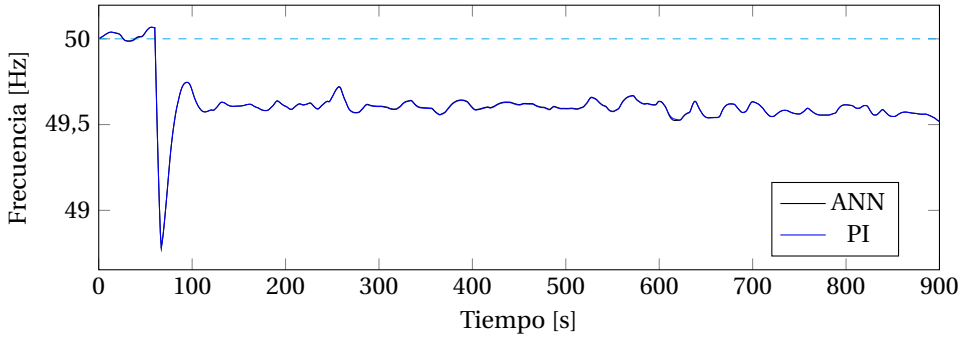


Figura 5.52: Frecuencia en Hz del caso 4.1.

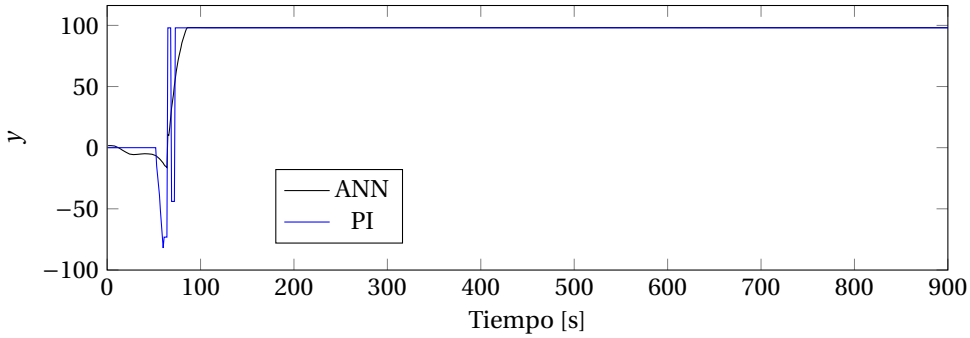


Figura 5.53: y del caso 4.1.

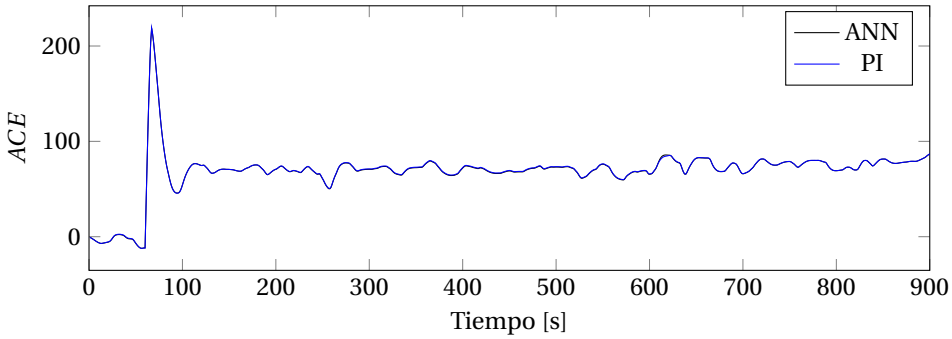
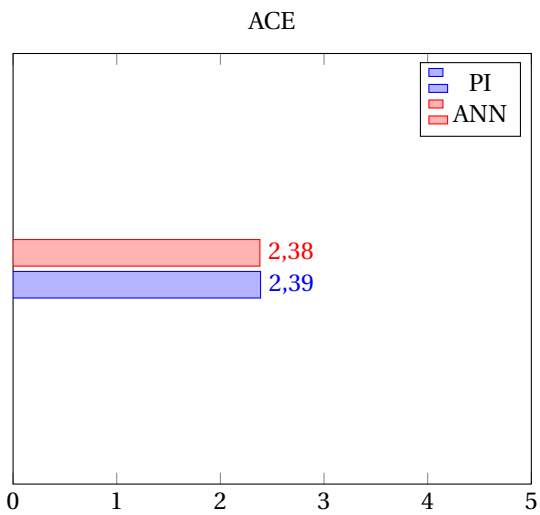


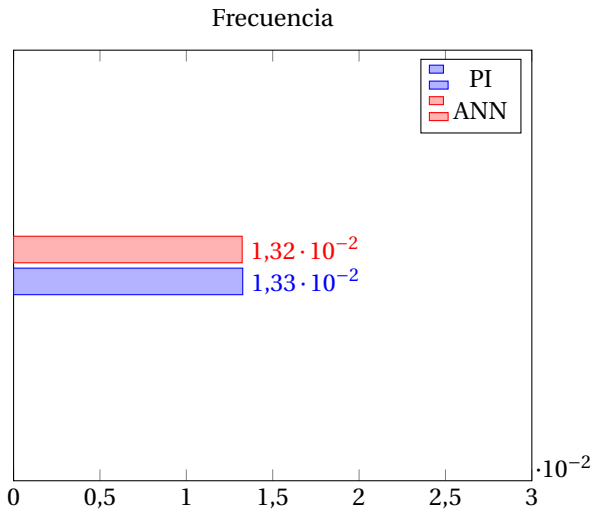
Figura 5.54: ACE del caso 4.1.

Este caso, al igual que lo ocurrido anteriormente, es de poco interés. Esto debido a que la única respuesta del controlador es llevar al máximo la generación, donde ambos controladores lo realizan.

Se tienen así respuestas idénticas tanto para el controlador PI como para el controlador, que pueden ser cuantificados mediante la integral del ACE^2 y la desviación de frecuencia, y que son mostrados en las Figuras 5.55a y 5.55b respectivamente.



(a) Resultados respectivos al ACE del caso 4.1 con la salida de ANG1.



(b) Resultados respectivos a la frecuencia del caso 4.1 con la salida de ANG1.

Figura 5.55

5.5. Controlador ANFIS

En esta sección se presentarán los mismos escenarios, pero esta vez utilizando el controlador tipo ANFIS descrito en los capítulos anteriores.

5.5.1. Caso 1.1: Alta penetración de ERNC y salida ANG1

Las simulaciones se presentan a continuación en las Figuras 5.56, 5.57 y 5.58.

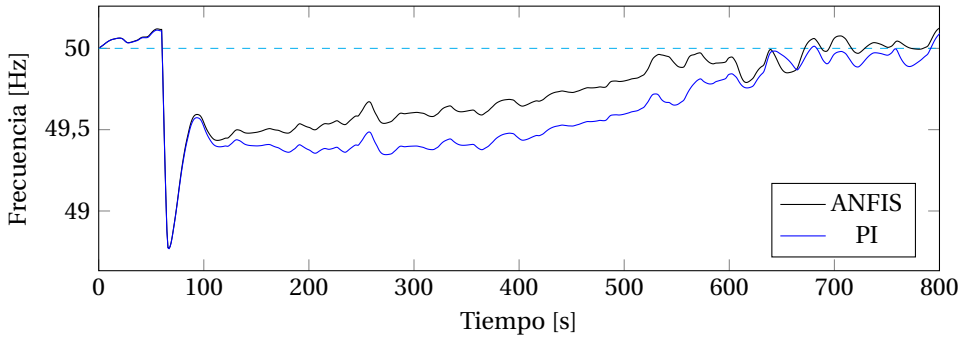


Figura 5.56: Frecuencia en Hz del caso 1.1.

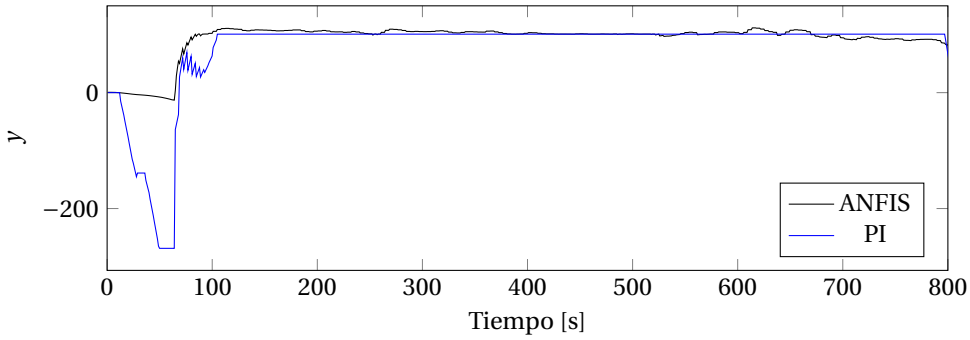


Figura 5.57: y del caso 1.1.

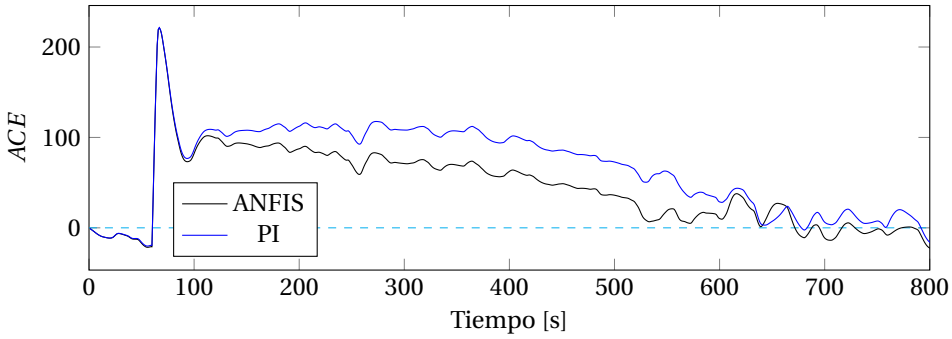


Figura 5.58: ACE del caso 1.1.

En el gráfico de frecuencia (Figura 5.56) se observa que hasta cercano a los 100 segundos la frecuencia para ambos casos es similar. Esto es debido a que la variabilidad y generación de ERNC, son demasiado vólatiles en ese tiempo con lo que la respuesta del controlador y de los generadores en sí no pueden hacer mucho.

Sin embargo, pasada la regulación primaria de frecuencia se empiezan a observar las primeras diferencias. Estas se explican visualizando el gráfico de la salida del controlador (Figura 5.57), donde se nota que debido a no sobrerreaccionar ante las variaciones previas a la caída del generador, permite al controlador

subir la consigna de potencia desde un punto más alto, lo que finalmente se traduce en un mejor comportamiento en el tiempo que le da un mayor rendimiento general.

Al cuantificar el efecto de ambos controladores usando la integral del ACE^2 (Figura 5.59), se observa que el controlador ANFIS fue 33.15 % superior que su contraparte, el controlador PI.

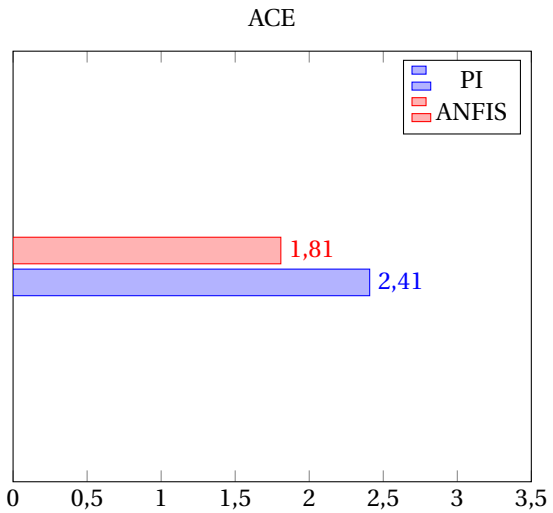


Figura 5.59: Resultados respectivos al ACE del caso 1.1 con la salida de ANG1.

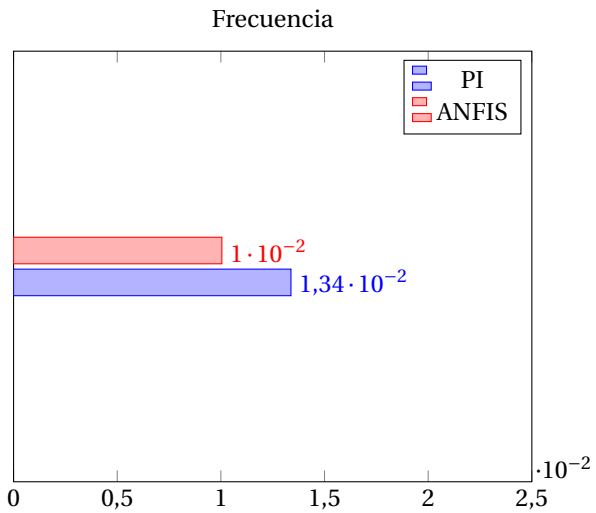


Figura 5.60: Resultados respectivos a la frecuencia del caso 1.1 con la salida de ANG1.

5.5.2. Caso 1.1: Alta penetración de ERNC y salida U15

Este caso también corresponde al escenario 1.1 donde las unidades participantes del AGC son: CCH1 y ANG2, pero en este caso se tiene la salida de U15.

Dichas simulaciones se muestran en 5.61, 5.62 y 5.63.

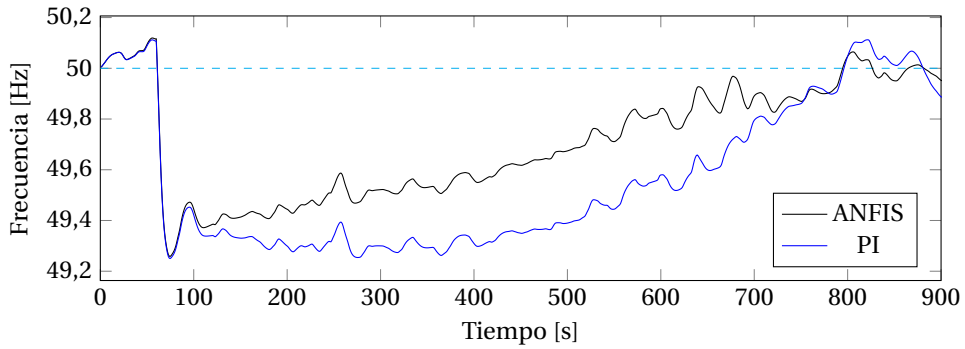


Figura 5.61: Frecuencia en Hz del caso 1.1.

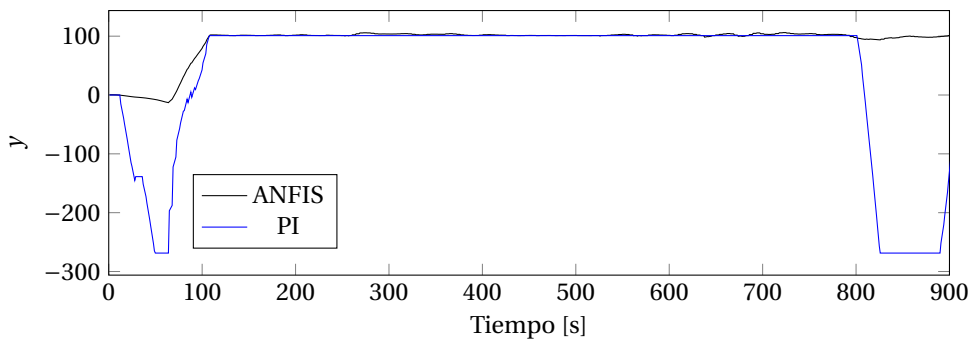


Figura 5.62: y del caso 1.1.

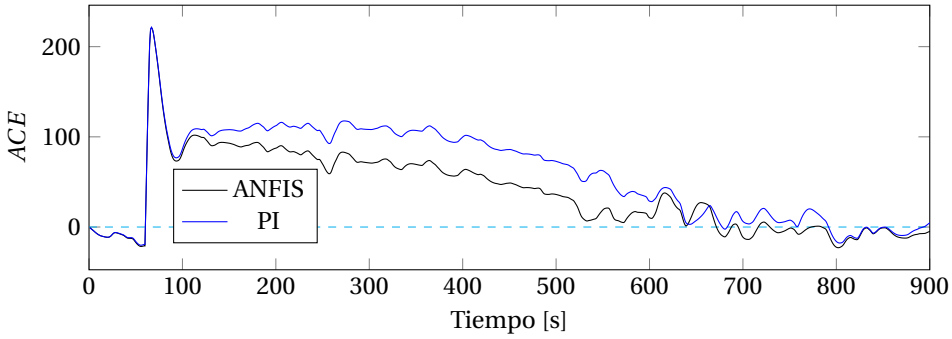


Figura 5.63: *ACE* del caso 1.1.

Se tiene que hay un comportamiento similar en cuanto al ACE y frecuencia previo a la salida de la unidad generadora, y actuación del controlador. Sin embargo, al terminar la etapa de regulación primaria de frecuencia, la respuesta del controlador cobra mayor relevancia.

Considerando que el controlador no sobreactuó ante las variaciones previas a la caída de U15, se encontraba en un punto más alto al iniciar la regulación secundaria de frecuencia, lo que le permitió al controlador ANFIS obtener un resultador más favorable.

Al cuantificar los efectos de ambos controladores de modo de compararlos, se tienen las Figuras 5.64 y 5.65, que son la integración del ACE^2 y desviación de frecuencia correspondientemente. De ello se concluye que el rendimiento del controlador ANFIS fue 42.23 % superior al de su contraparte el controlador PI.

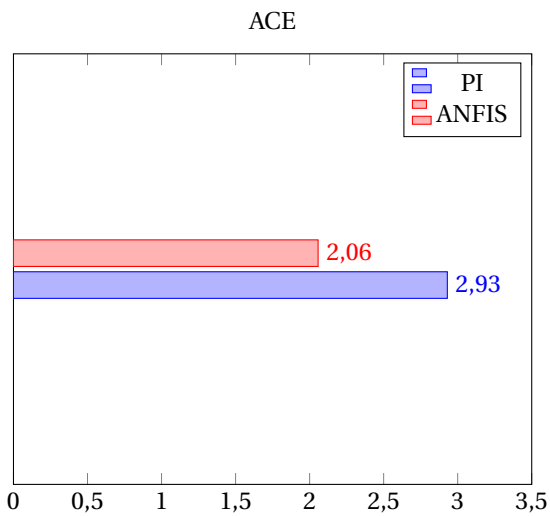


Figura 5.64: Resultados respectivos al ACE del caso 1.1 con la salida de U15.

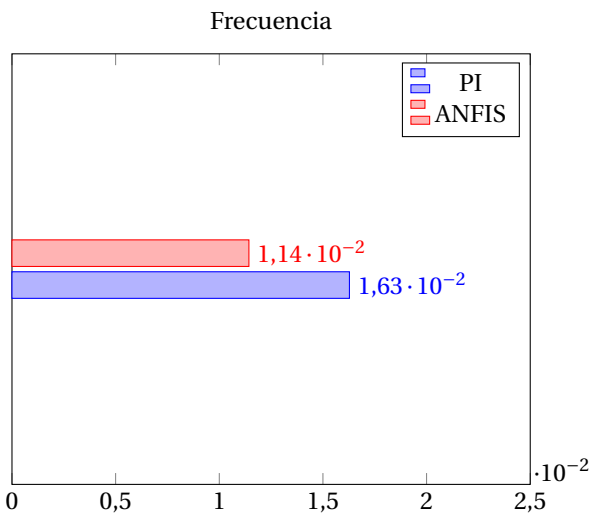


Figura 5.65: Resultados respectivos a la frecuencia del caso 1.1 con la salida de U15.

5.5.3. Caso 2.1: Sin ERNC y salida ANG2

Los resultados de las simulaciones son mostrados en las Figuras 5.66 y 5.68

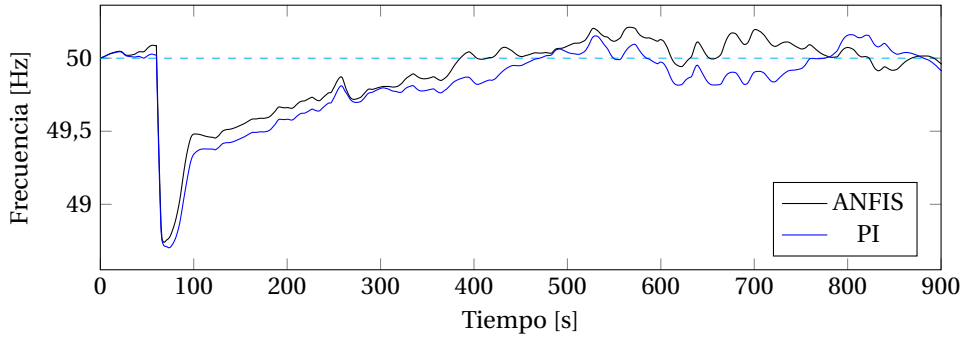


Figura 5.66: Frecuencia en Hz del caso 2.1.

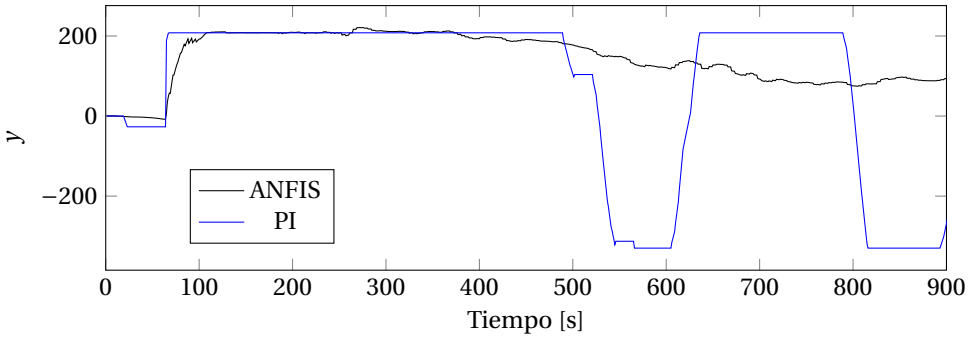


Figura 5.67: y del caso 2.1.

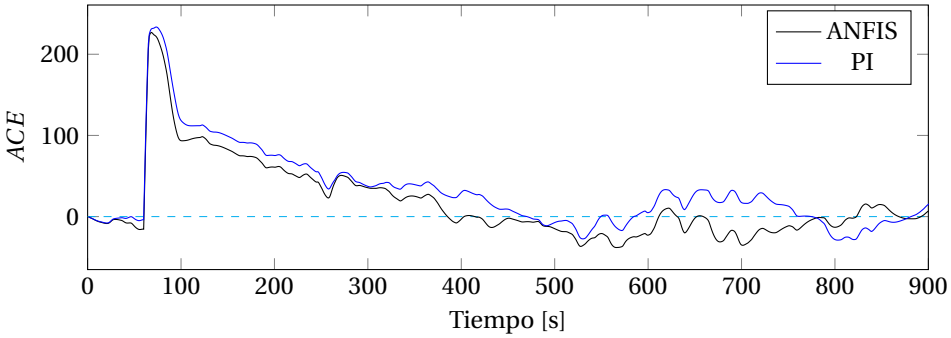
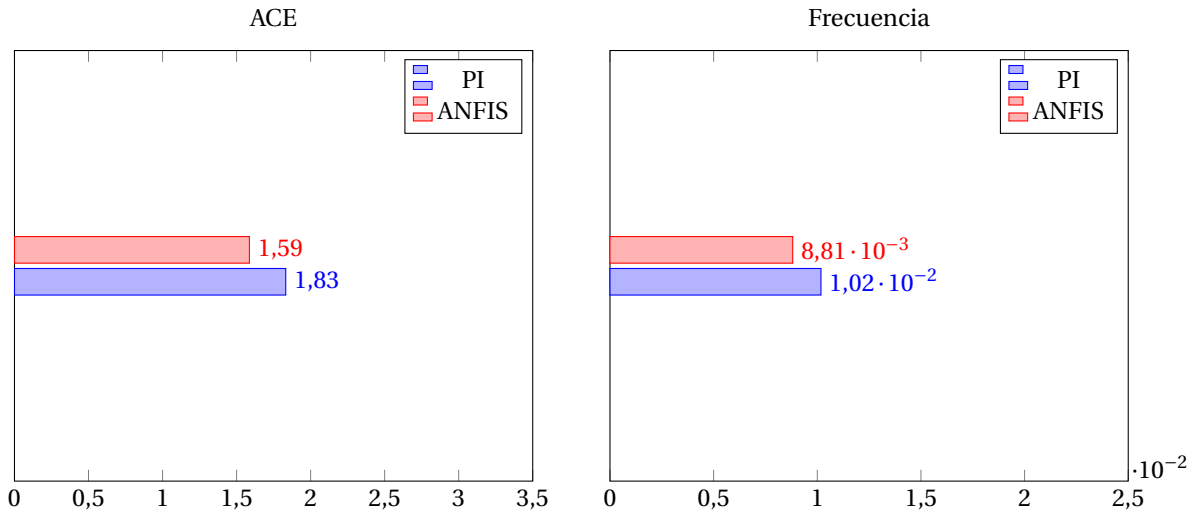


Figura 5.68: ACE del caso 2.1.

Para este caso se vuelve a evidenciar, que ante el cambio de signo del ACE el controlador PI reacciona de forma exagerada, llevando su salida al mínimo, e incluso mandando consignas de bajar la generación actual. Mientras que, en la simulación con controlador ANFIS, baja el valor de generación de forma más gradual, sin cambiar de signo. Lo anterior se traduce en un mejor comportamiento de parte del controlador ANFIS, que le permite obtener un mejor rendimiento.

Al cuantificar el rendimiento de ambos controladores, utilizando la integración de ACE^2 e integración de la desviación de frecuencia, se tienen las Figuras 5.69a y 5.69b correspondientemente. De ello se concluye que el rendimiento del controlador ANFIS fue 15.09 % superior al de su contraparte, el controlador PI.



(a) Resultados respectivos al ACE del caso 2.1 con la salida de ANG2.

(b) Resultados respectivos a la frecuencia del caso 2.1 con la salida de ANG2.

Figura 5.69

5.5.4. Caso 2.1: Sin ERNC y salida U15

En este escenario las unidades que participan del AGC son: CCH1, CCH2, U16 y GAG TG+0.5TV, y se considera la salida de U15.

Los resultados de las simulaciones son mostrados en las Figuras 5.70, 5.71 y 5.72.

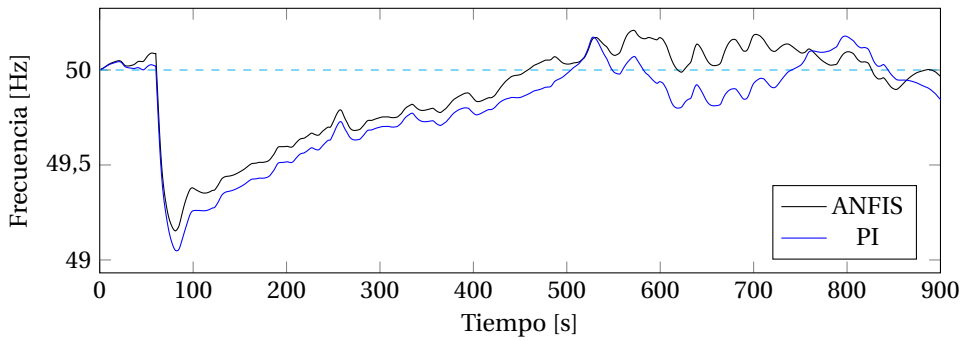


Figura 5.70: Frecuencia en Hz del caso 2.1.

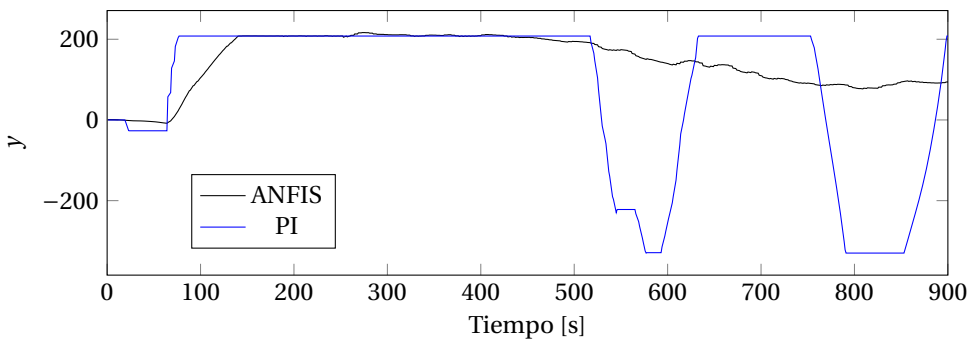


Figura 5.71: y del caso 2.1.

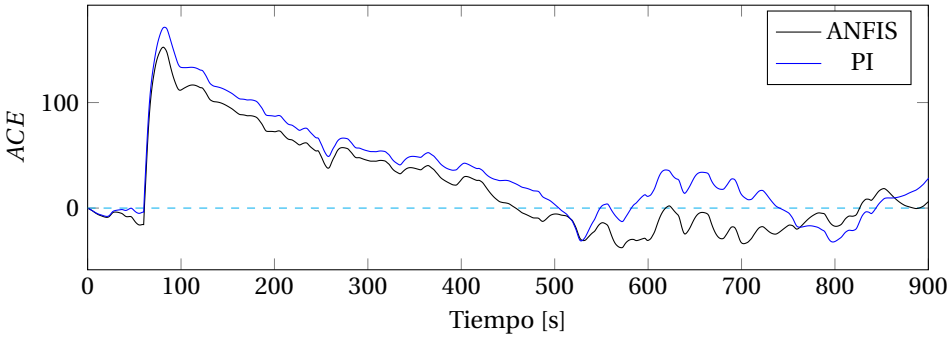


Figura 5.72: *ACE* del caso 2.1.

Se puede observar del gráfico de frecuencia (Figura 5.70) una pequeña diferencia al inicio de la simulación, donde previo a la caída del generador U15, existe control de parte del controlador PI, pero no de parte del controlador ANFIS. Esto debido a que el ACE llega a un valor suficientemente grande como para hacer un cambio súbito en el valor de la constante proporcional del controlador, mientras que en el controlador ANFIS se ajusta tan lentamente a bajar la potencia (debido a un bajo ACE), que no alcanza a controlar frecuencia de buena manera en ese instante.

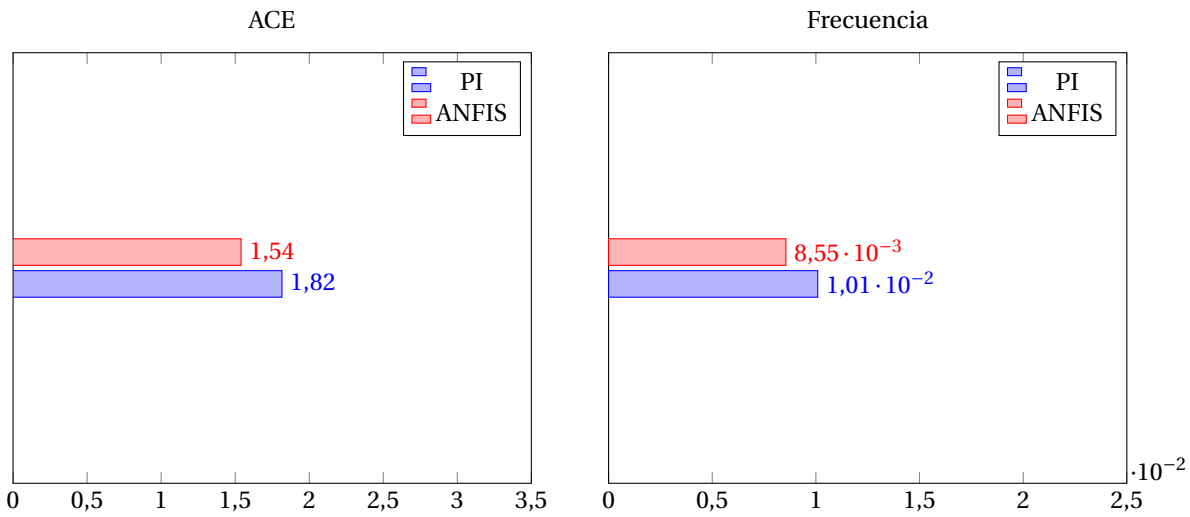
Este pequeño ajuste hecho previamente hace que el sistema esté en una frecuencia menor antes de la caída del generador, con lo cual al caer la unidad generadora, se llega a un valor aún más bajo.

Todo este proceso induce a que entre aproximadamente los 100 y 500 segundos el controlador ANFIS funcione de mejor forma, incluso cuando el controlador PI sube la consigna de potencia de forma más oportuna.

Pasado los 500 segundos se nota en el gráfico de la frecuencia (Figura 5.70), que baja y sobrepasa continuamente los 50 [Hz]. Esto se explica en la Figura 5.71, donde se observa que la salida del controlador empieza a oscilar. Contrario a este comportamiento, el controlador ANFIS no oscila, sino, que hace una bajada suave desde ese punto en adelante.

Cabe señalar que el efecto inicial de controlar la frecuencia previo a la salida del generador, que posteriormente deterioró los resultados dando ventaja al controlador ANFIS se podría considerar una coincidencia. Esto por que no es parte del controlador haberse anticipado a que la frecuencia caería hasta ese punto. Sin embargo, la no oscilación en la parte final de la simulación demuestra una clara ventaja del controlador ANFIS.

Finalmente se puede cuantificar el rendimiento de ambos controladores mediante la integración del ACE^2 y de la desviación de frecuencia, que puede ser visualizado en las Figuras 5.73a y 5.73b. Se puede constatar que el controlador ANFIS fue superior a su contraparte, el controlador PI, por un 18.18 %.



(a) Resultados respectivos al ACE del caso 2.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 2.1 con la salida de U15.

Figura 5.73

5.5.5. Caso 3.1 Entrada de ERNC y Salida U15

En este caso las unidades que están en AGC son: CCH1, CCH2 y ANG2 y corresponde a la entrada de ERNC. Las simulaciones correspondientes a este escenario están presentadas a continuación en las Figuras 5.74, 5.75 y 5.76.

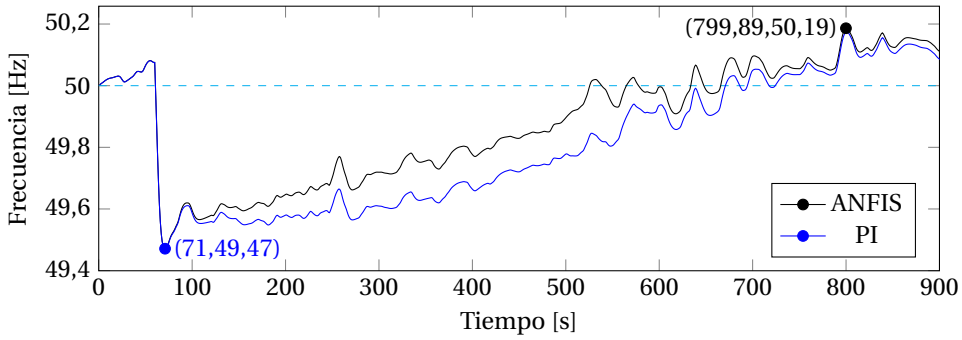


Figura 5.74: Frecuencia en Hz del caso 3.1.

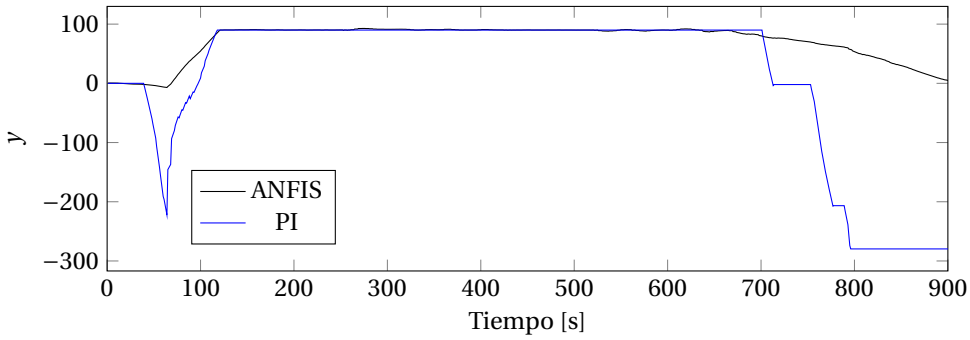


Figura 5.75: *y* del caso 3.1.

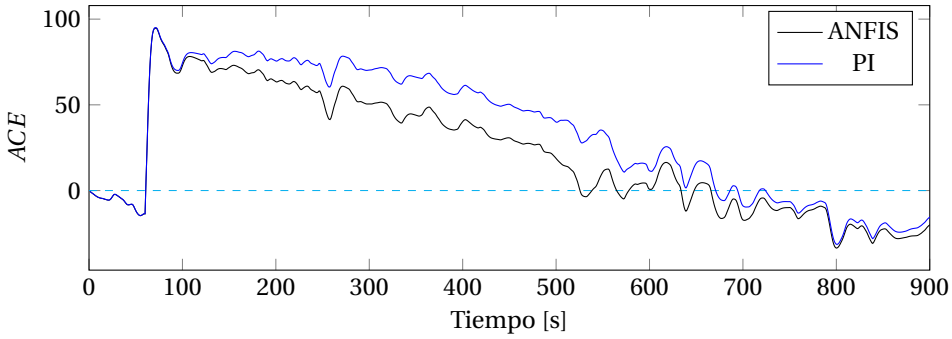


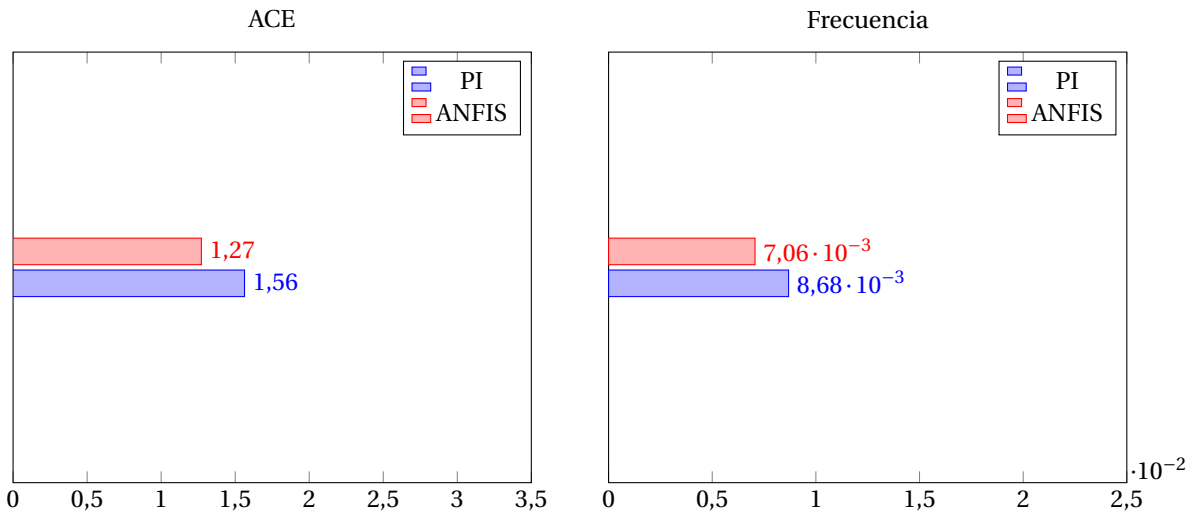
Figura 5.76: *ACE* del caso 3.1.

Se puede observar en el gráfico de frecuencia (Figura 5.74) que para ambos casos es la misma previa a la caída del generador. Esto se puede explicar viendo la salida del controlador (figura 5.75), donde se observa que el controlador PI reacciona solo un poco antes de la salida de U15, lo cual de todas formas deja su valor de salida bajo, mientras que el controlador ANFIS disminuye su salida, pero en menor proporción.

Luego de la caída de la unidad U15, la salida del controlador PI sube rápidamente, pero debido a que anteriormente se encontraba en un punto más bajo, le tomó más tiempo llegar al valor máximo.

Esto en términos generales provoca que el controlador ANFIS considere una ventaja, desde ese punto hasta el final de la simulación, como se puede ver en el gráfico de frecuencia y de ACE, donde se observan formas similares pero desplazadas por una proporción que se desvanece lentamente, lograda por el efecto explicado anteriormente.

Al cuantificar los resultados de ambos controladores mediante la integración del ACE^2 y la desviación de frecuencia, se tienen las Figuras 5.77a y 5.77b. De donde se puede concluir que el controlador ANFIS fue 22.83 % superior a su contraparte el controlador PI.



(a) Resultados respectivos al ACE del caso 3.1 con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 3.1 con la salida de U15.

Figura 5.77

5.5.6. Caso 4.1: Salida de ERNC y salida ANG1

En este caso las unidades que están en AGC son: ANG1, ANG2 y CCH2. Las simulaciones correspondientes a este escenario están presentadas a continuación en las Figuras 5.78, 5.79 y 5.80.

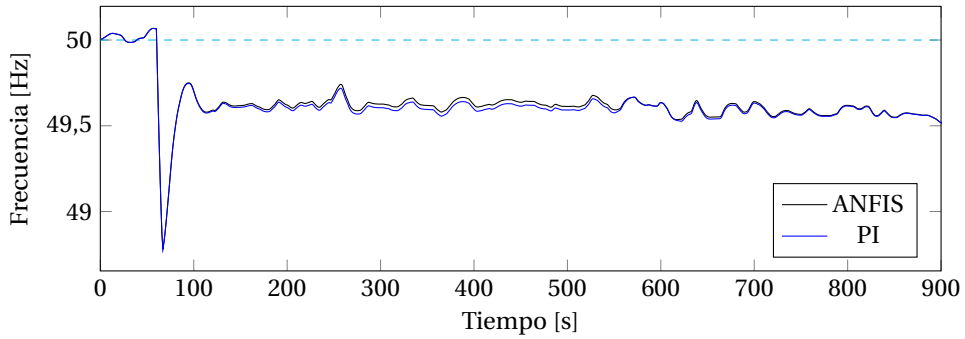


Figura 5.78: Frecuencia en Hz del caso 4.1.

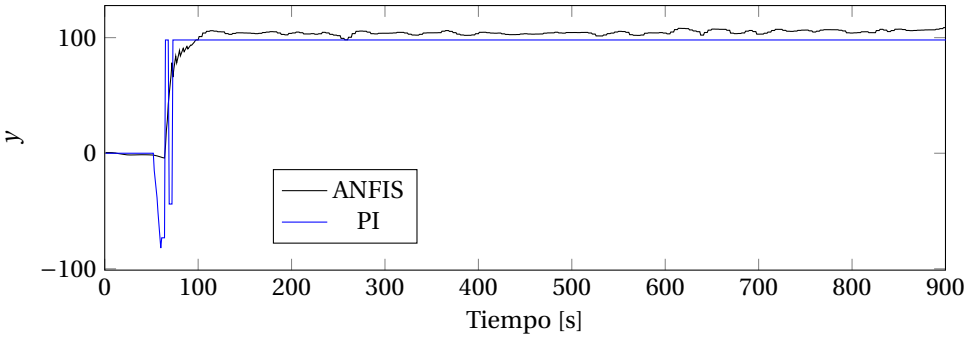


Figura 5.79: y del caso 4.1.

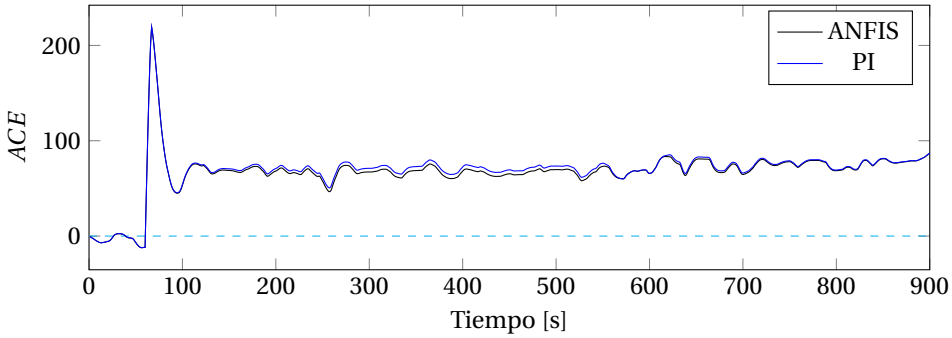
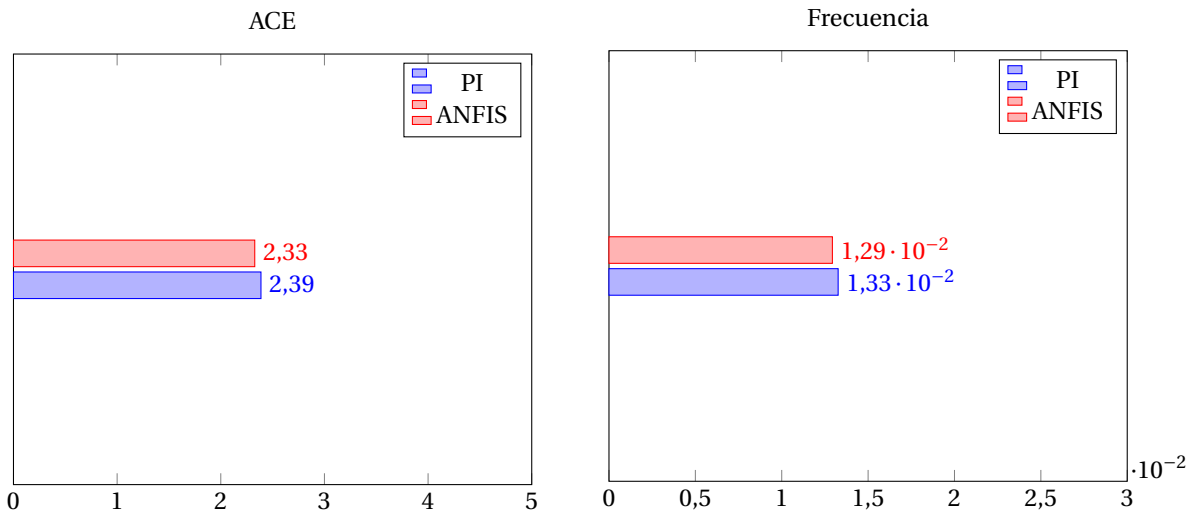


Figura 5.80: *ACE* del caso 4.1.

En este caso se puede observar que se tiene un comportamiento similar a lo visto anteriormente: la caída es tan fuerte y con tan poca potencia en reserva, que aunque ambos controladores llevan al máximo la generación, no hay un mayor efecto en la restauración de la desviación de frecuencia.

Por esto mismo, se tiene que la única respuesta necesaria fue una subida súbita, la cual fue más rápida en el controlador ANFIS, con lo cual tuvo una leve ventaja frente al controlador PI.

Cuantificando dichos efectos mediante la integral del ACE^2 y la desviación de frecuencia, representados en las Figuras 5.81a y 5.81b respectivamente, se tiene que el controlador ANFIS en este caso fue superior en un 2.575 %.



(a) Resultados respectivos al ACE del caso 4.1 con la salida de ANG1.

(b) Resultados respectivos a la frecuencia del caso 4.1 con la salida de ANG1.

Figura 5.81

5.6. Sumario de resultados

De modo de hacer más fácil la visualización del comportamiento de cada uno de los controladores se realizó un cuadro resumen que se muestra en la Figura 5.82. En dicha figura se puede observar que a excepción del caso 4.1, los controladores inteligentes utilizados en este trabajo tuvieron un rendimiento mucho mejor en comparación al uso del controlador PI, siendo el controlador ANFIS el de mejores resultados.

Que el controlador ANFIS sea el mejor se explica por su gran nivel de adaptabilidad (se ajustan curvas de pertenencias, y funciones de primer orden, en cambio en ANN solo se ajustan funciones de tipo sigmoide o arcotangete).

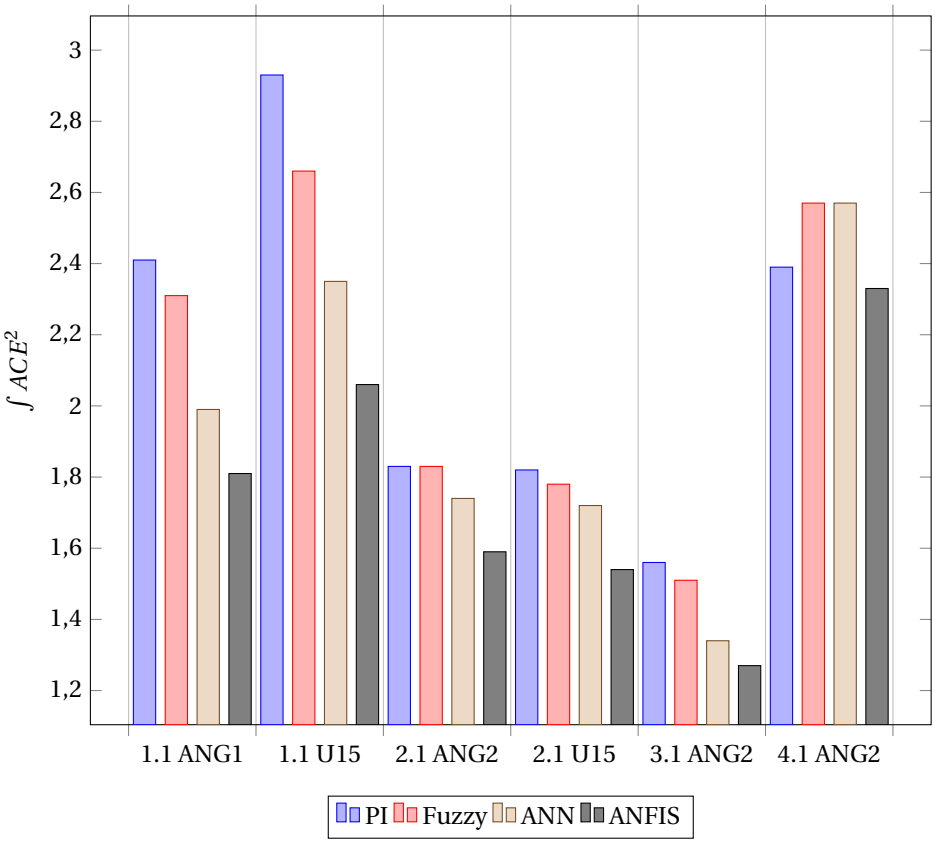


Figura 5.82: Resumen de resultados de todos los casos, considerando los rendimientos de todos los controladores.

6

Resultados para el sistema considerando interconexión SING-SADI

Para hacer las simulaciones correspondientes al sistema considerando la interconexión SING-SADI se tuvieron los siguientes casos, también obtenidos del estudio del AC3E:

Unidad	Escenario 1.1 Despacho [MW]	Escenario 1.2 Despacho [MW]
Cerro Pabellón	32	32
PAM	17	17
ANG1	268	253
ANG2	245	272
CCH2	222	150
CTA	165	165
CCH1	222	225
CTH	162	162
U16	137	288.9
CTM2	154	148
U15	116	75
U14	122	75
CTM1	149	149
U12	50	50
U13	X	50
CTM3	X	X
GAG TG+0.5TV	155.38	X
Kelar TG1	X	X
NTO1	132	132
CTTAR	100	100
NTO2	132	132
TG3	X	X
ERNC	368	368
Despacho Total	2793	2793.9

X = Fuera de Servicio
Unidad en AGC
Unidad Slack
Unidad Exportadora SING-SADI

Tabla 6.1: Tabla con los valores de despacho de las centrales para los escenarios 1.1 y 1.2 (Alta penetración de ERNC)

Debido a que para esta ocasión el controlador realizará corrección sobre la potencia de intercambio entre ambos sistemas, se debe ajustar el modelo para que lo incorpore al ACE. Para ello, utilizando el mismo modelo presentado en el capítulo 4 (Figura 5.4), basta con cambiar el factor K_{SI} de 0 a 1.

Además, se debe configurar la consigna de Potencia de intercambio, la cual se considerará como el valor de potencia transmitida en el enlace antes de que se produzca el evento.

Considerando lo anterior, el valor de potencia de intercambio para el escenario 1.1 es:

$$P_{int} = 148{,}106[MW]$$

Y para el caso 1.2:

$$P_{int} = 49{,}542[MW]$$

6.1. Modelamiento de la interconexión y del sistema SADI.

El modelo en Digsilent considera la interconexión mediante una línea entre las barras Andes y Salta, como se muestra en la Figura 6.1:

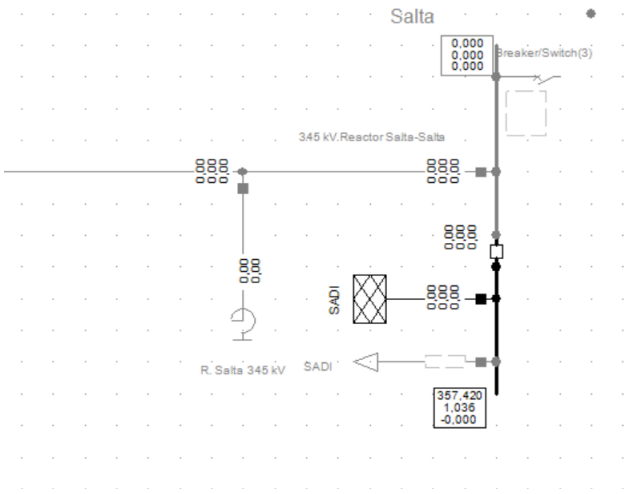


Figura 6.1: Interconexión con la barra Salta.

Además, el sistema SADI está modelado con todas sus unidades en una *Grid* adicional, lo que aumenta considerablemente el tiempo que toman las simulaciones.

6.2. Controlador Fuzzy

Para esta sección se realizarán las simulaciones correspondientes al uso del controlador tipo difuso. Para determinar de buena forma las funciones de pertenencia de la lógica difusa se debería realizar el mismo proceso que en el capítulo 4 (optimización mediante el algoritmo PSO). Sin embargo, debido al tamaño del sistema, se nota que las simulaciones tardan cerca de 6 horas cada una, por lo que utilizar nuevamente el algoritmo de optimización es poco práctico.

Para resolver este problema se debe pensar lo siguiente: el sistema se comportará de forma similar, en el sentido de las respuestas que debería tener el controlador ante ciertos eventos, pero con un sistema más grande. Tiene sentido entonces, considerar solo la normalización de las variables, o sea, cambiar los factores de escalamiento definidos en el capítulo 4, y dejar las funciones de pertenencia sin cambios.

Al observar las simulaciones con el controlador PI, se tiene que en el caso 1.1 con la salida de U15 (peor caso de los considerados en el estudio para la interconexión SING-SADI), el ACE se comporta como se muestra en la Figura 6.2. En dicha figura se observa que el valor máximo se alcanza a los 232 segundos, con un valor de 82.31.

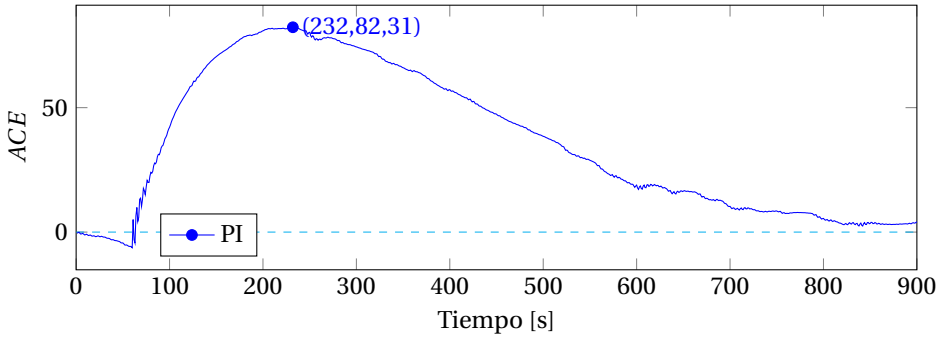


Figura 6.2: *ACE* del caso 1.1

Por lo tanto, se pueden definir como valores de escalamiento:

$$\alpha_{ACE} = 100 \quad \alpha_{\Delta ACE} = 100 \tag{6.1}$$

6.2.1. Caso 1.1: Alta penetración de ERNC y salida U15

Para este caso se tiene que las unidades participantes en el AGC son: CCH1, CCH2 y ANG2. Se realizan las simulaciones correspondientes obteniendo los gráficos de frecuencia, respuesta del controlador, ACE y potencia de intercambio en las figuras 6.3, 6.4, 6.5 y 6.6, correspondientemente.

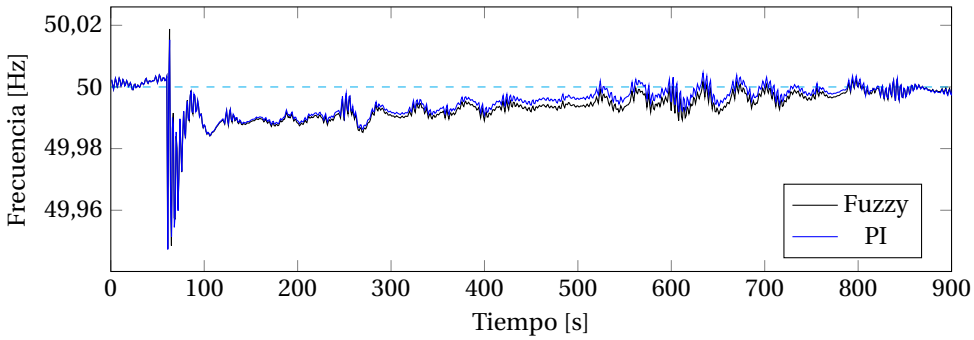


Figura 6.3: Frecuencia en Hz del caso 1.1.

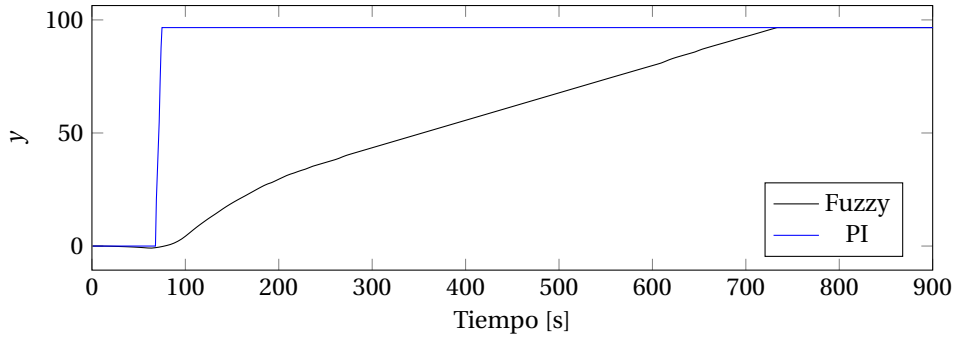


Figura 6.4: *y* del caso 1.1.

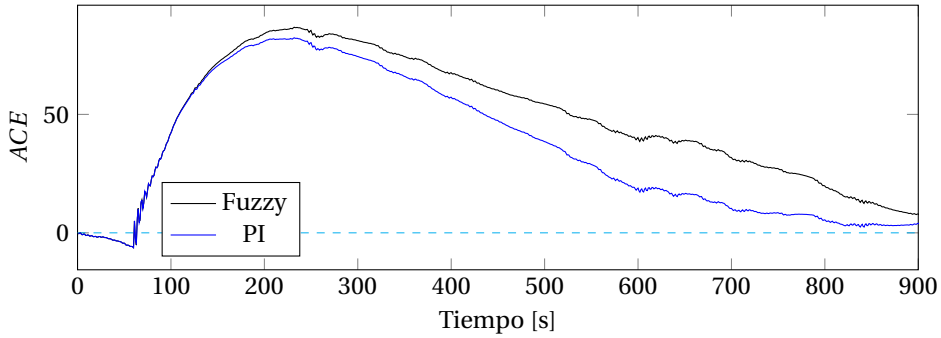


Figura 6.5: *ACE* del caso 1.1.

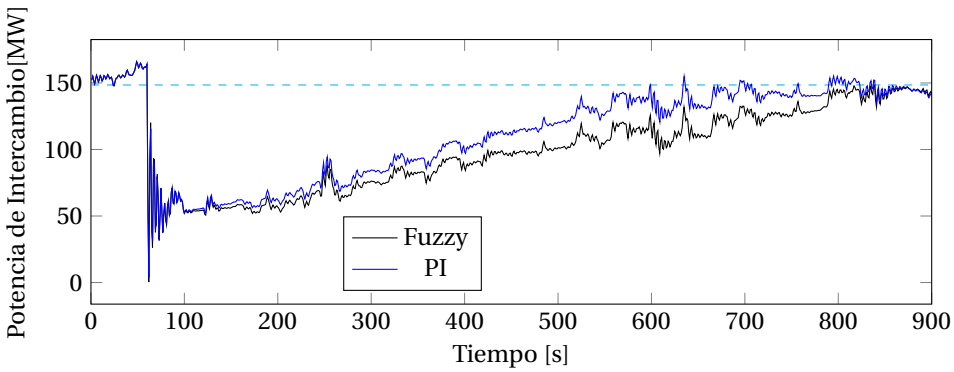


Figura 6.6: $intercambio$ del caso 1.1.

Se puede observar en el gráfico de la frecuencia que alcanza valores máximos en el segundo 62.89, con un valor de 50.02[Hz], también se aprecia que el mínimo se sitúa en 49.95[Hz], situaciones que se producen tanto por la variación de la demanda, variación ERNC como por la salida del generador U15.

De la Figura 6.4 se puede apreciar que ambos controladores llevan al tope su generación, siendo el controlador PI mucho más rápido para reaccionar.

Considerando que la frecuencia en todo momento está por debajo del valor nominal, llevar el valor a tope le da ventaja al controlador PI, dándole un finalmente un mayor rendimiento.

Al cuantificar el rendimiento de ambos controladores mediante la integral del ACE^2 y de la desviación de frecuencia representados en las Figuras 6.7 y 6.8 respectivamente, se puede concluir que el controlador difuso se comportó peor que su contraparte en un 16.66 %, debido principalmente a lo mencionado anteriormente.

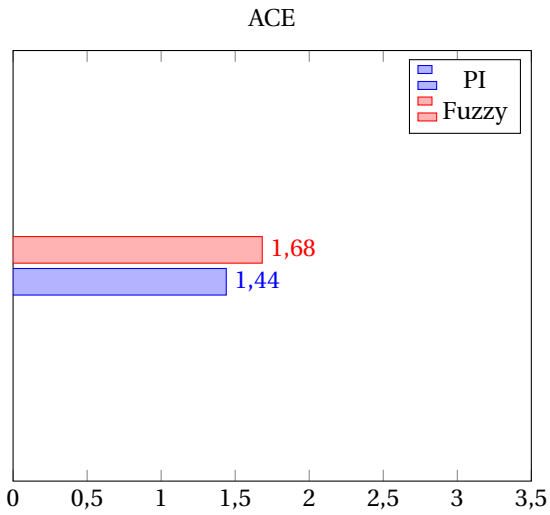


Figura 6.7: Resultados respectivos al ACE del caso 1.1 con interconexión al SADI, y con la salida de U15.

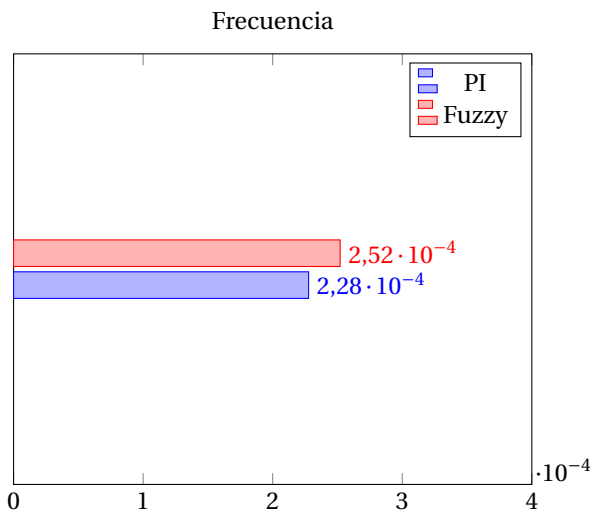


Figura 6.8: Resultados respectivos a la frecuencia del caso 1.1 con interconexión al SADI, y con la salida de U15.

6.2.2. Caso 1.2: Alta penetración de ERNC y salida U15

Para este caso se tiene que las unidades participantes en el AGC son: CCH1 y U16. Se realizan las simulaciones correspondientes obteniendo los gráficos de frecuencia, respuesta del controlador, ACE y potencia de intercambio en las Figuras 6.9, 6.10, 6.11 y 6.12, correspondientemente.

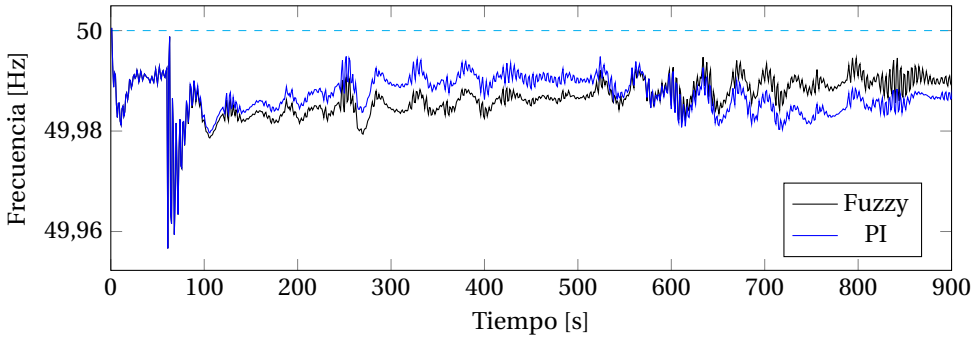


Figura 6.9: Frecuencia en Hz del caso 1.2.

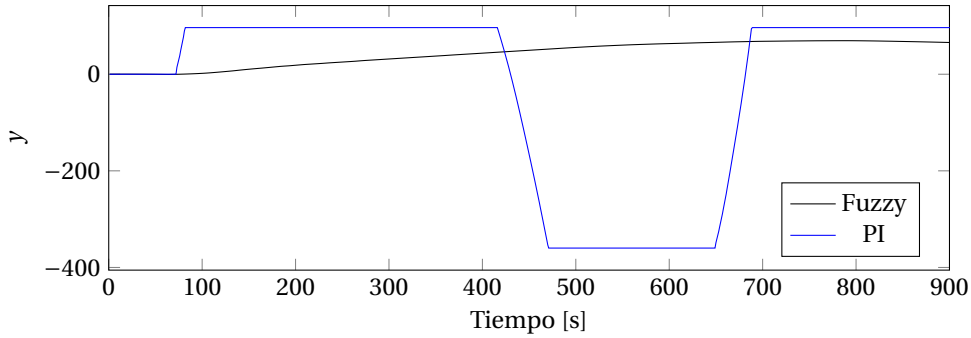


Figura 6.10: y del caso 1.2.

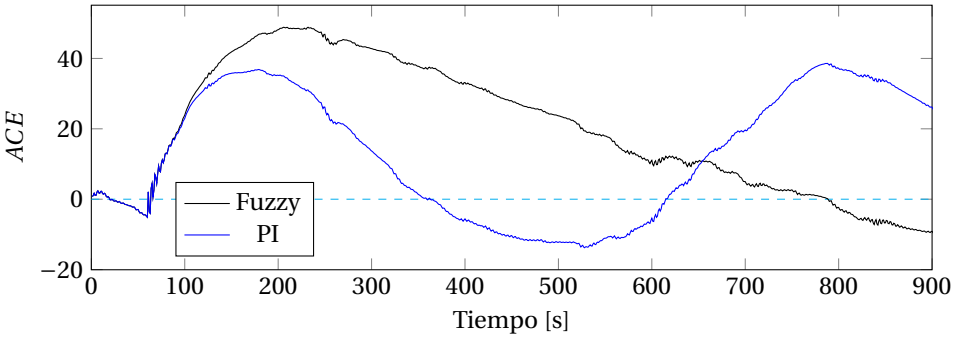


Figura 6.11: *ACE* del caso 1.2

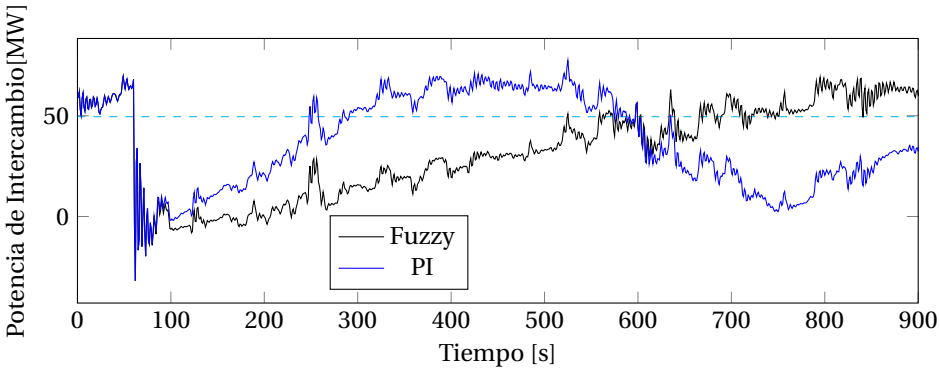


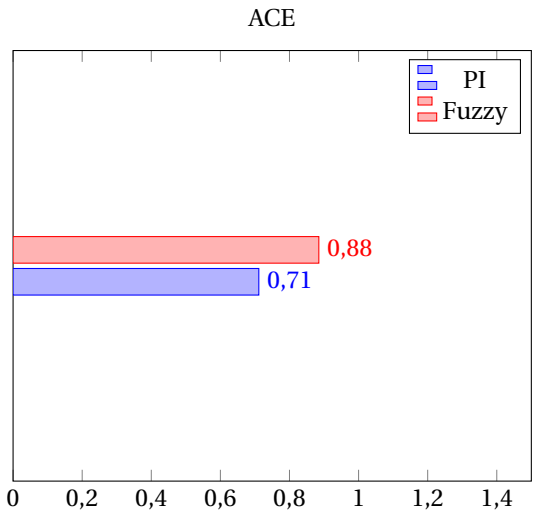
Figura 6.12: Potencia de intercambio del caso 1.2 con salida de U15

Se puede observar en la Figura 6.9 que la salida del generador U15. Esta desviación de frecuencia no es suficiente como para activar los esquemas automáticos de deslastre de carga, pero si lo suficientemente grandes como para hacer operar el AGC en los segundos posteriores.

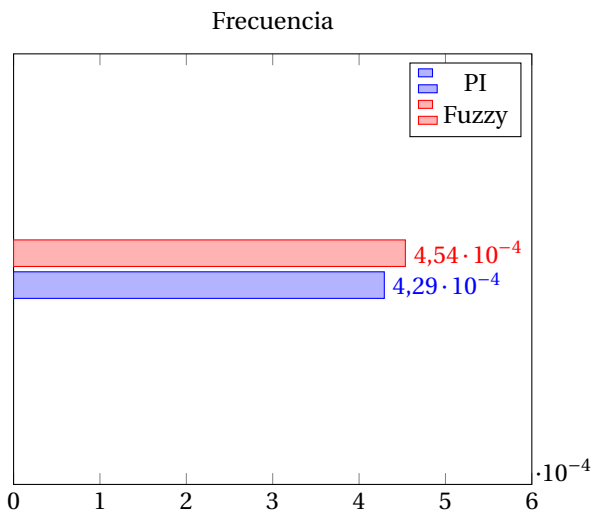
En la Figura 6.10, se nota que pasado el evento de la salida del generador, ambos controladores suben su valor de salida, siendo el controlador PI el que lleva dicho valor a tope de forma casi instantánea, mientras que el controlador difuso sube de forma lenta. Esto le permite al controlador PI subir su valor de frecuencia a valores nominales de forma mucho más rápida. Sin embargo, luego de ocurrido esto, se puede observar que el controlador difuso no oscila de la misma forma como lo hace el controlador PI, de hecho, el controlador difuso incluso al descender a valores negativos no hace cambios bruscos en su salida. Esto debido a que el ACE para el caso Fuzzy no cambió de signo.

Al cuantificar los efectos y control de ambos controladores utilizando los valores de la integral del ACE^2 y desviación de frecuencia en el tiempo se tienen las Figuras 6.13a y 6.13b. De ellas se puede notar que el controlador difuso tuvo un rendimiento 23.94 % menor a su contraparte.

Sin embargo, para este caso se podría considerar que el cuantificador de rendimiento no es completamente justo en determinar que controlador fue mejor, o por lo menos en el tiempo de simulación. Esto se explica observando que los valores finales de potencia de intercambio y frecuencia son mucho mejores en el caso del controlador Fuzzy, con lo que se podría pensar, que en un tiempo mayor de simulación el valor de rendimiento sería mejor.



(a) Resultados respectivos al ACE del caso 1.2 con interconexión al SADI, y con la salida de U15.



(b) Resultados respectivos a la frecuencia del caso 1.2 con interconexión al SADI, y con la salida de U15.

Figura 6.13

6.3. Controlador ANN

En esta sección se revisarán los casos vistos anteriormente, pero usando el controlador ANN.

Para demostrar la adaptabilidad y escalabilidad del controlador, no se corregirá el factor de escalamiento, y se usarán los ajustes tal cual fueron utilizados en el sistema SING sin interconexión.

6.3.1. Caso 1.1: Alta penetración de ERNC y salida U15

Para este caso se tiene que las unidades participantes en el AGC son: CCH1, CCH2 y ANG2. Se realizan las simulaciones correspondientes obteniendo los gráficos de frecuencia, respuesta del controlador, ACE y potencia de intercambio en las Figuras 6.14, 6.15, 6.16 y 6.17, correspondientemente.

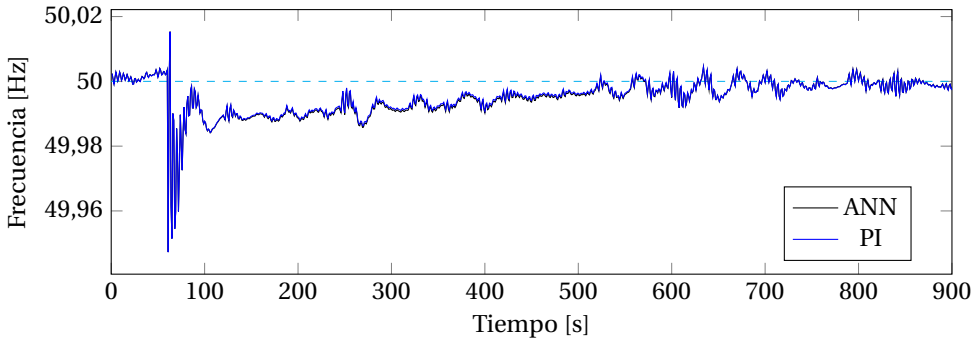


Figura 6.14: Frecuencia en Hz del caso 1.1.

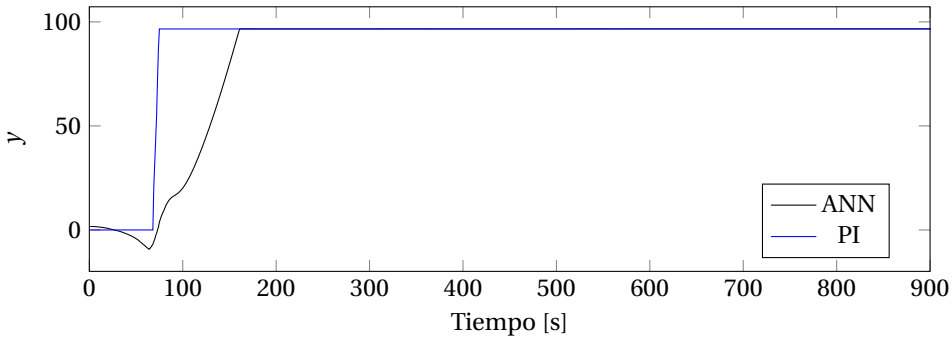


Figura 6.15: y del caso 1.1.

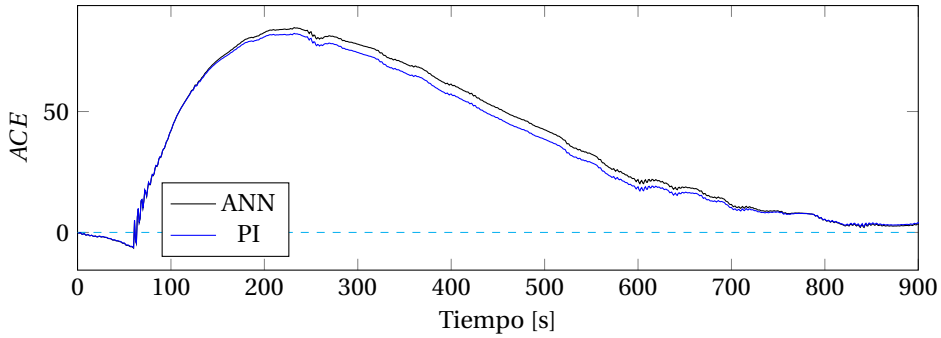


Figura 6.16: *ACE* del caso 1.1.

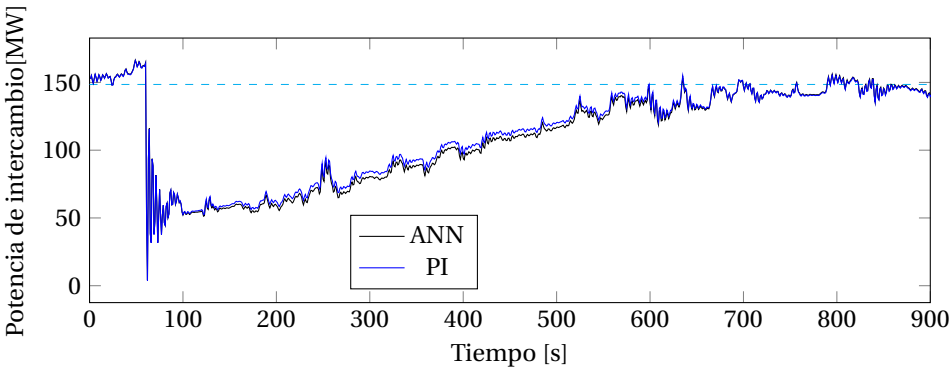
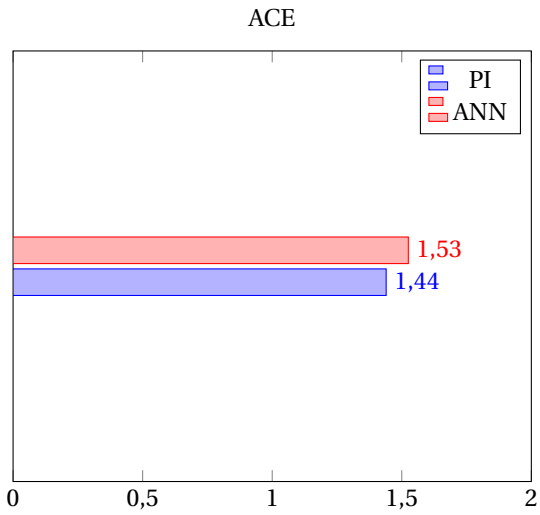


Figura 6.17: *ACE* del caso 1.1.

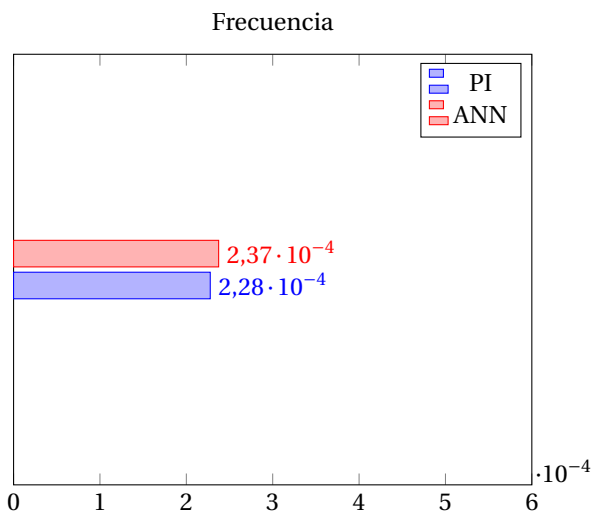
Se puede observar en el gráfico de la frecuencia que alcanza valores máximos en el segundo 62.89, con un valor de 50.02[Hz], también se aprecia que el mínimo se sitúa en 49.95[Hz], situaciones que se producen tanto por la variación de la demanda, variación ERNC como por la salida del generador U15.

De la Figura 6.15 se puede apreciar que hay un pequeño efecto previo al evento que produce que el controlador ANN baje su generación. Esto produce que dicho controlador sea más lento en reaccionar posterior a la salida del generador U15, con lo que demora un tiempo más en llevar a las demás generadoras pertenecientes al grupo de regulación de frecuencia a subir la consigna de potencia.

Esta pequeña diferencia le da ventaja al controlador PI, pero no demasiado. Al cuantificar los rendimientos considerando la integral del ACE^2 y de la desviación de frecuencia se tienen las Figuras 6.18a y 6.18a respectivamente, donde se tiene que el controlador PI fue mejor que el ANN en 6.25 %



(a) Resultados respectivos al ACE del caso 1.1 con interconexión al SADI, y con la salida de U15.



(b) Resultados respectivos a la frecuencia del caso 1.1 con interconexión al SADI, y con la salida de U15.

Figura 6.18

6.3.2. Caso 1.2: Alta penetración de ERNC y salida U15

Para este caso se tiene que las unidades participantes en el AGC son: CCH1 y U16. Se realizan las simulaciones correspondientes obteniendo los gráficos de frecuencia, respuesta del controlador, ACE y potencia de intercambio en las Figuras 6.19, 6.10, 6.21 y 6.22, correspondientemente.

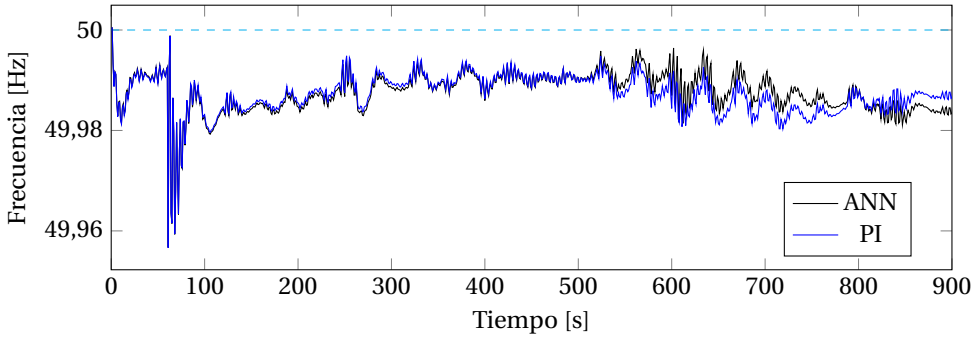


Figura 6.19: Frecuencia en Hz del caso 1.2.

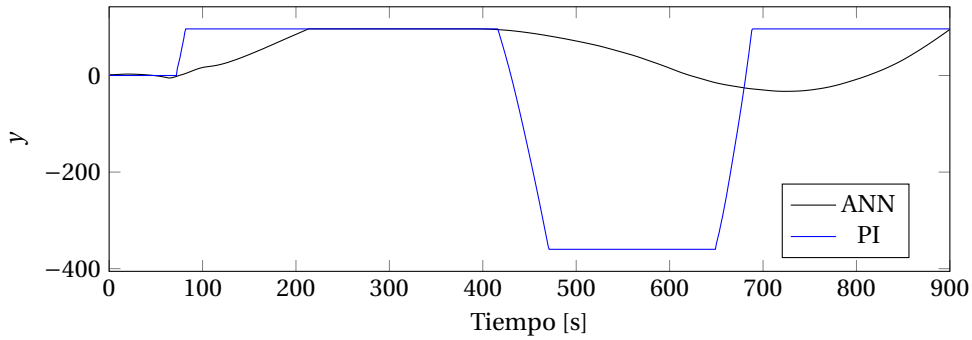


Figura 6.20: y del caso 1.2.

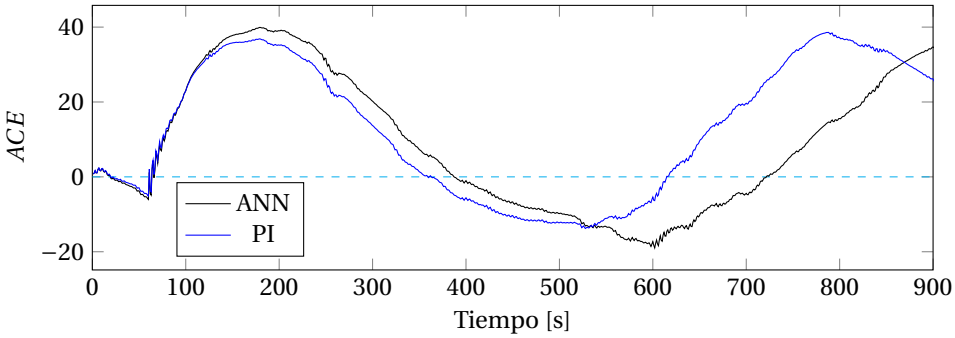


Figura 6.21: ACE del caso 1.2.

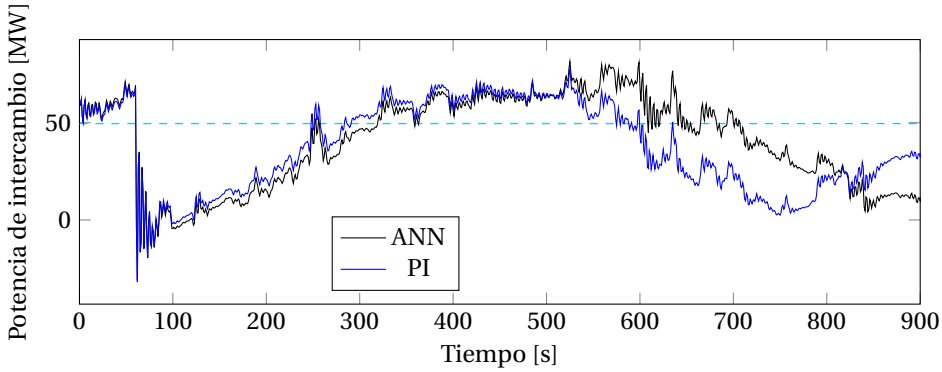
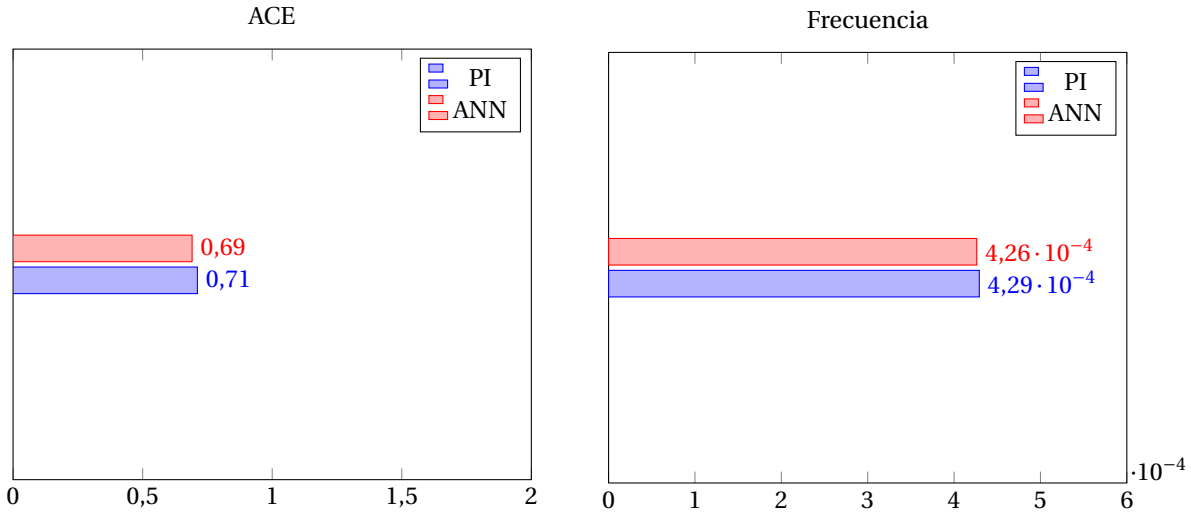


Figura 6.22: Potencia de intercambio del caso 1.2

Al igual que en los casos anteriores se puede observar en el gráfico del ACE (Figura 6.21) que se produce una oscilación. Esto es debido al comportamiento de la potencia de intercambio donde se puede apreciar en la Figura 6.22 , que ésta cruza varias veces el set point fijado a 49.512 [MW].

Al observar la salida de los controladores (Figura 6.20) se tiene que dichas oscilaciones son tomadas de forma abrupta por el controlador, haciendo que los valores cambien desde $y_{máx}$ a $y_{mín}$ en pocos segundos. Evento que no ocurre de la misma forma en controlador ANN, que tiene bajadas mucho más suaves, lo cual le permite a dicho controlador obtener un rendimiento superior al PI.

Al cuantificar los resultados mediante la integral del ACE^2 y de la desviación de frecuencia se tienen las Figuras 6.23a y 6.23a respectivamente. De dichas figuras se desprende que el controlador ANN fue superior al PI por un 2.89 %.



(a) Resultados respectivos al ACE del caso 1.2 con interconexión al SADI, y con la salida de U15.

(b) Resultados respectivos a la frecuencia del caso 1.2 con interconexión al SADI, y con la salida de U15.

Figura 6.23

6.4. Controlador ANFIS

6.4.1. Caso 1.1: Alta penetración de ERNC y salida U15

Para este caso se tiene que las unidades participantes en el AGC son: CCH1, CCH2 y ANG2. Se realizan las simulaciones correspondientes obteniendo los gráficos de frecuencia, respuesta del controlador, ACE y potencia de intercambio en las Figuras 6.24, 6.25, 6.26 y 6.27, correspondientemente.

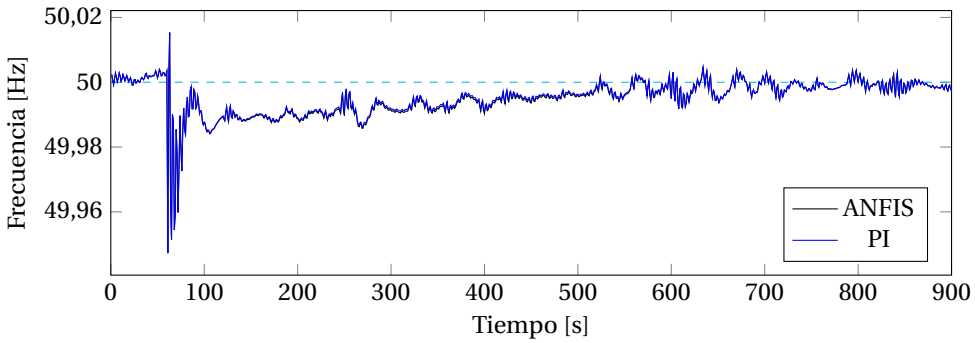


Figura 6.24: Frecuencia en Hz del caso 1.1.

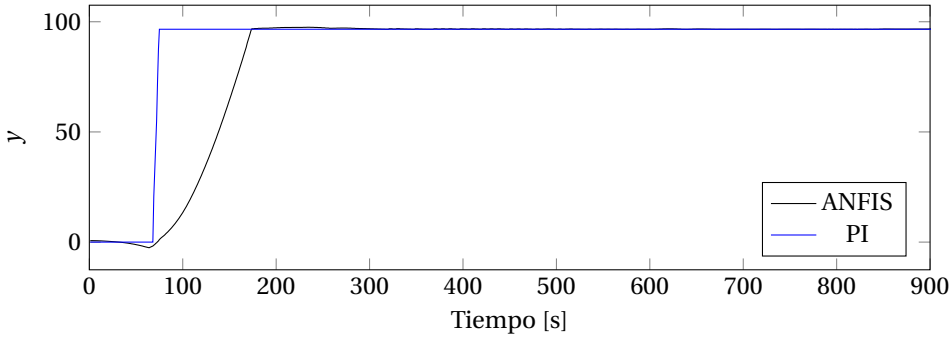


Figura 6.25: y del caso 1.1.

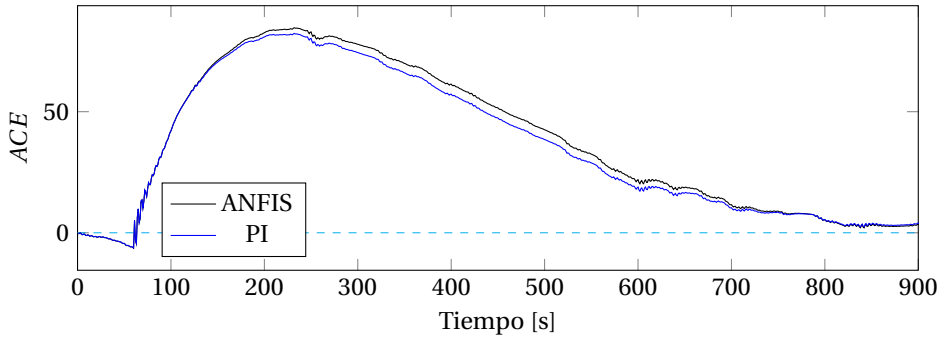


Figura 6.26: ACE del caso 1.1.

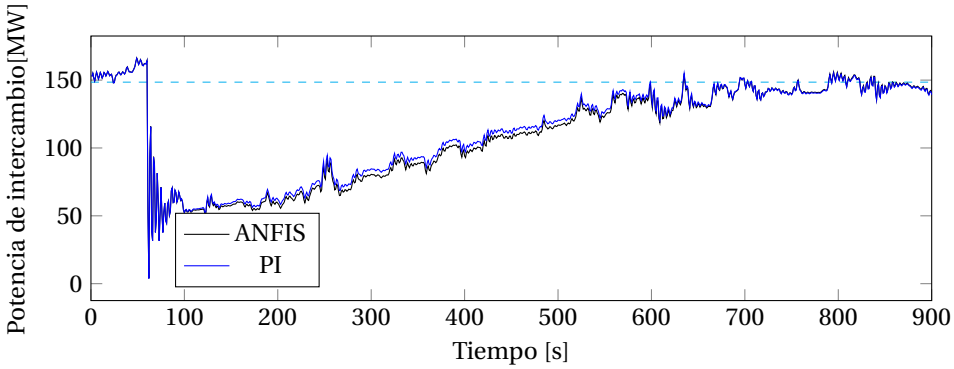
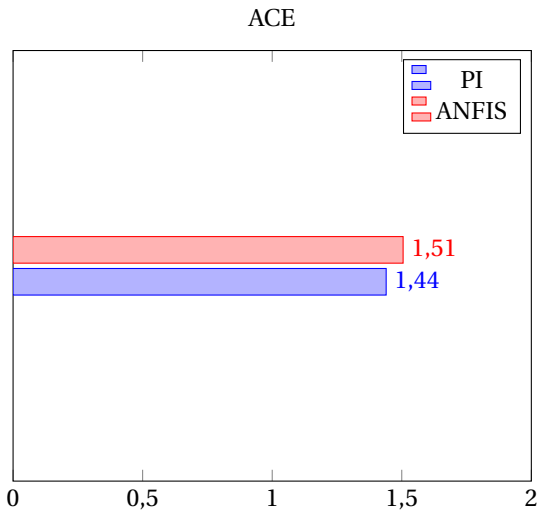


Figura 6.27: ACE del caso 1.1.

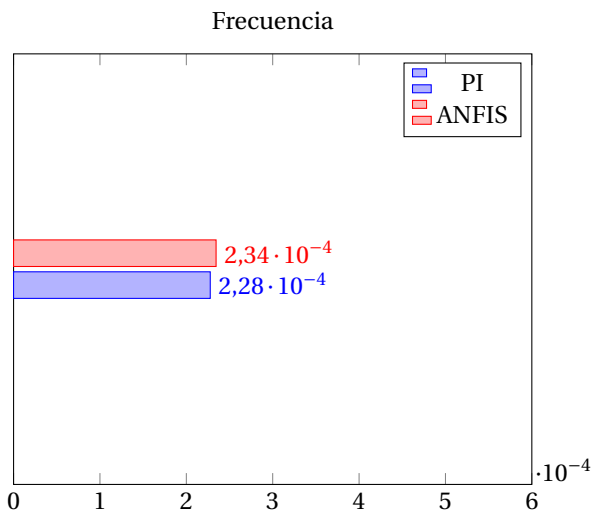
En esta simulación se puede observar nuevamente que lo necesario para lograr el control de frecuencia es llevar la salida del controlador a tope. Por ello, el controlador que lo haga de forma más rápida, será el cual tenga mejor rendimiento.

Por lo mencionado anteriormente, este caso no tiene mayor interés, ya que no se pueden visualizar todas las virtudes del controlador ANFIS. De todos modos, se tiene que dicho controlador es capaz de controlar la frecuencia de forma similar al controlador PI, con lo que se puede decir que no tiene problemas para adaptarse a este caso.

Al cuantificar los resultados utilizando la integral del ACE^2 y de la desviación de frecuencia, se tienen las representaciones gráficas de estos mismos en las Figuras 6.33a y 6.33a respectivamente. En ellas se puede visualizar que el controlador PI fue superior al controlador ANFIS en un 4.86 %.



(a) Resultados respectivos al ACE del caso 1.2 con interconexión al SADI, y con la salida de U15.



(b) Resultados respectivos a la frecuencia del caso 1.2 con interconexión al SADI, y con la salida de U15.

Figura 6.28

6.4.2. Caso 1.2: Alta penetración de ERNC y salida U15

Para este caso se tiene que las unidades participantes en el AGC son: CCH1 y U16. Se realizan las simulaciones correspondientes obteniendo los gráficos de frecuencia, respuesta del controlador, ACE y potencia de intercambio en las Figuras 6.29, 6.30, 6.31 y 6.32, correspondientemente.

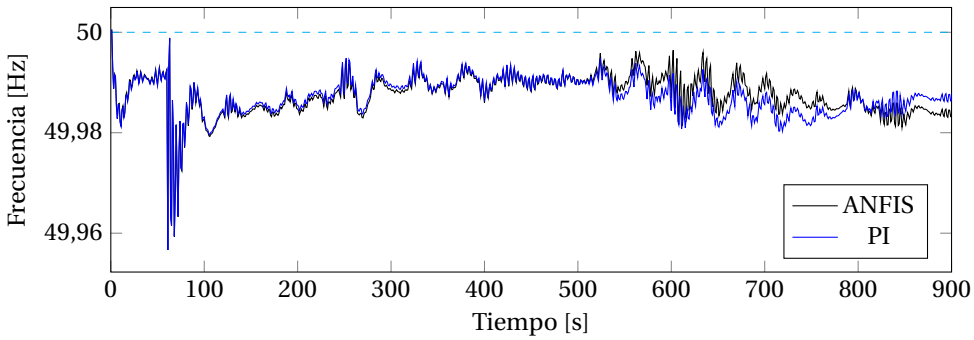


Figura 6.29: Frecuencia en Hz del caso 1.2.

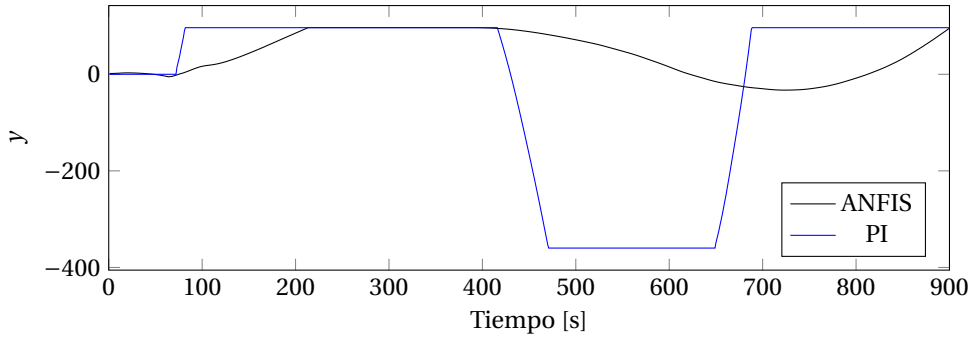


Figura 6.30: y del caso 1.2.

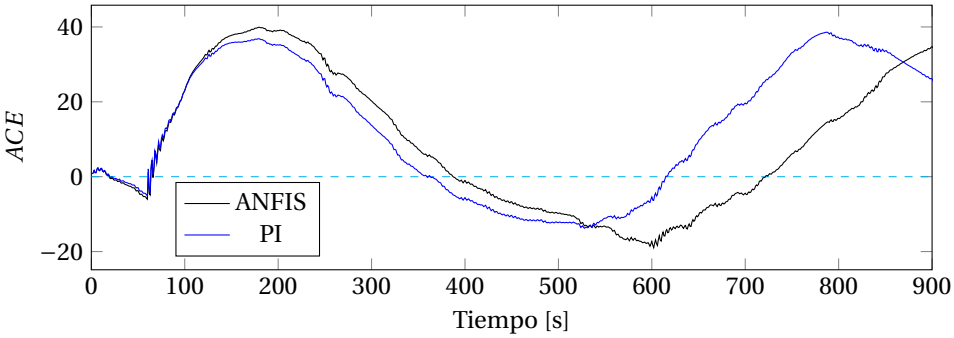


Figura 6.31: *ACE* del caso 1.2.

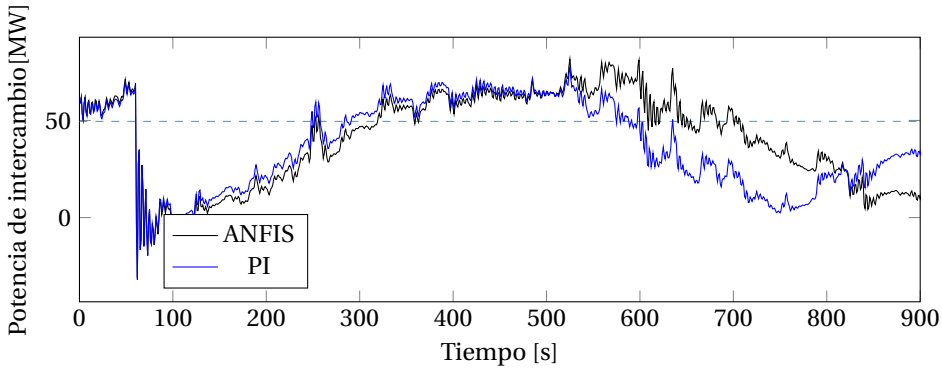
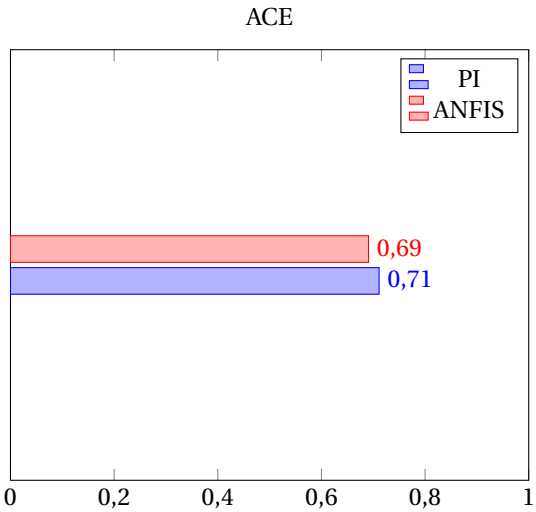


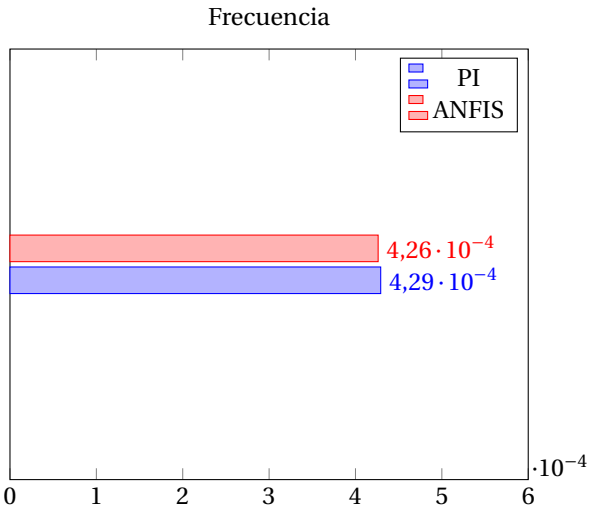
Figura 6.32: *ACE* del caso 1.2.

Para este caso se puede observar que la salida del controlador ANFIS tiene un comportamiento muy similar a lo visto anteriormente con el controlador ANN. Esto es debido a sus características similares en cuanto a adaptabilidad y que comparten el mismo escalamiento.

Al obtener la integral sobre el ACE^2 y la desviación de frecuencia para determinar el rendimiento, se tienen las Figuras 6.33a y 6.33b respectivamente. En ellas se puede apreciar que el controlador ANFIS fue superior al controlador PI por un 2.89 %, el mismo valor que fue presentado al utilizar el controlador ANN.



(a) Resultados respectivos al ACE del caso 1.2 con interconexión al SADI, y con la salida de U15.



(b) Resultados respectivos a la frecuencia del caso 1.2 con interconexión al SADI, y con la salida de U15.

Figura 6.33

6.5. Sumario de resultados

De modo de hacer más simple la comparación general de todos los controladores, se tomaron todos los rendimientos y se condensaron en un solo gráfico mostrado en la figura 6.34. En este gráfico, se puede notar que todos los controladores funcionaron de peor forma que el controlador PI, excepto por el caso 1.2 con salida de U15, donde se tiene que los basados en ANN y ANFIS fueron superiores.

Se veía que en el capítulo 5, los resultados eran favorables en su mayoría para los controladores inteligentes, siendo que en este caso no se ve lo mismo. Una respuesta a lo que ocurre es que no hay escalabilidad de algunos de ellos.

Para el caso del controlador difuso sería necesaria una nueva optimización y escalamiento, de modo de poder superar en rendimiento al controlador PI, sin embargo, tener dos versiones del controlador es poco práctico y no parece tener mucho sentido.

El controlador ANN, al tener características de adaptabilidad, su optimización se da caso a caso, por lo que el rendimiento de dicho controlador es superior al del difuso, sin embargo, por las características dadas de las neuronas de la capa de entrada, le resulta imposible escalar dichas variables. Por ello, en la literatura se ofrece una solución [3], que consiste en el uso de redes neuronales flexibles, que básicamente son esquemas de neuronas que no solo adaptan los pesos de las interconexiones sino también la escala de la salida de la neurona (las ocupadas en este trabajo tienen una salida fija entre -1 y 1, mientras que en las flexibles ese valor varía y es optimizado junto a los pesos). No fue objetivo de este trabajo implementar dicha red neuronal, debido a que el controlador ANFIS ofrece dicha virtud. Es decir, las funciones de pertenencia de la entrada se adaptan de tal forma que se pueden escalar al rango de los valores de entrada. Por lo mismo, se puede notar que fue el controlador con mejor respuesta en todos los casos, siendo solo superado por el PI en los casos sin mucho interés, los cuales fueron donde el controlador solo debía llevar la salida a tope.

Con todo lo mencionado anteriormente, se puede destacar de gran forma el funcionamiento del controlador ANFIS, que ofrece control de buena forma de la frecuencia, adaptabilidad y robustez.

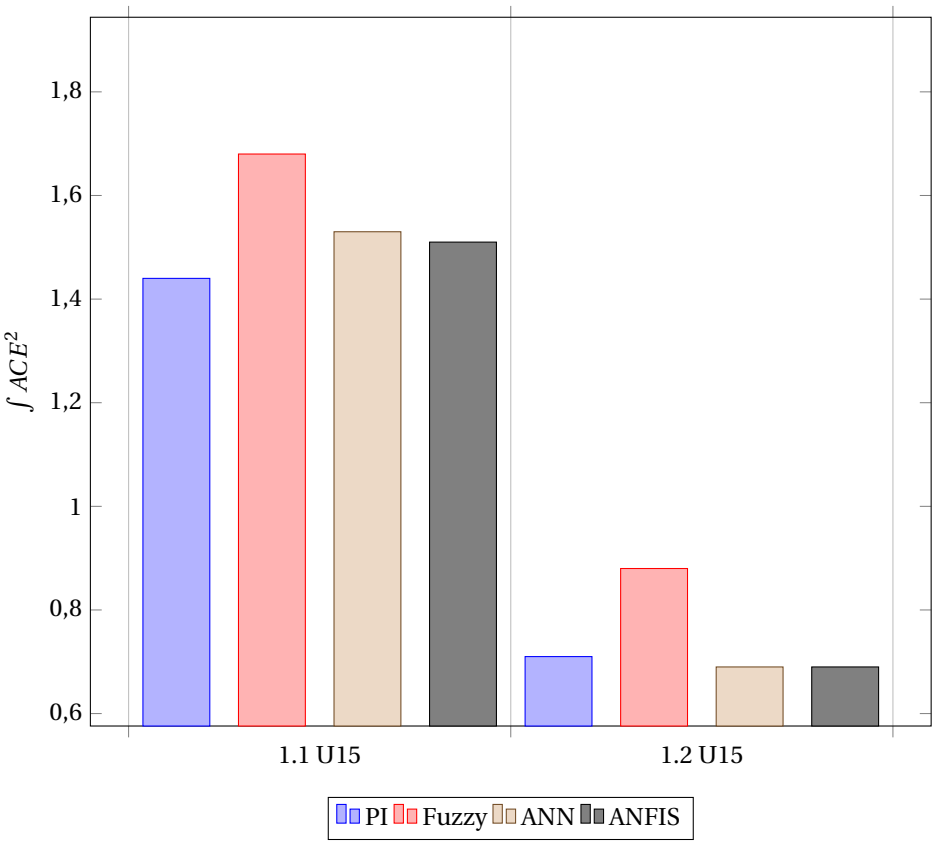


Figura 6.34: Resumen de resultados de todos los casos, considerando los rendimientos de todos los controladores.

7

Conclusiones

El objetivo principal de esta memoria fue dar las directrices para implementar de forma exitosa los controladores de tipo difuso, con redes neuronales y neuro-difusos, a un sistema de control complejo como lo es la regulación de frecuencia de un sistema de potencia.

Teniendo esto en cuenta, se considera que los resultados demuestran que es posible implementarlo en un entorno simulado utilizando ciertas herramientas que ofrecen cada uno de los software, así como también en el caso real donde se pudieron obtener algunas ventajas de estos controladores comparados con el controlador PI.

En el capítulo 5 se evidenciaron claras ventajas del controlador difuso, el cual obtuvo mejores prestaciones en todos los casos excepto por el caso 4.1. Esto debido a que la única respuesta necesaria de parte del controlador era llevar la generación al máximo posible, lo cual deja al caso como poco interesante debido a que como se mencionó, solo fue necesaria una acción rápida que si fue entregada de parte del controlador PI.

Una desventaja se pudo evidenciar en el capítulo 6, donde fue necesario el cambio del escalamiento para que el controlador hiciera un control correcto sobre un sistema más grande. Esto quita la ventaja de escalabilidad del controlador además de quitar también robustez de control, debido a que requirió análisis manual posterior para hacerlo funcionar de mejor manera (no fue automático).

Una forma de poder mejorar esto sería realizar la optimización nuevamente considerando la función objetivo como la integral del ACE^2, pero de ambos casos (SING sin interconexión y SING-SADI), con lo cual se llegaría a funciones de pertenencia que optimizan ambos casos.

Para el controlador ANN se pudo comprobar que es posible implementar un control a base de redes neuronales. Esto permitió desarrollar un esquema que se adapta al caso de forma automática, ya que realiza su optimización mientras está funcionando.

Al igual que en el uso de controlador difuso, se pudo notar que los resultados al ser reutilizado en el otro sistema considerando el mismo escalamiento, no fueron superiores al controlador PI. Por lo tanto, se tuvo un controlador que es mejor que el controlador clásico PI en el caso que se hizo el dimensionamiento (capítulo 4, casos sin interconexión), pero al ser llevado a otro sistema no responde de mejor manera (capítulo 6, casos con interconexión).

De todas maneras, el controlador logró el objetivo de controlar la frecuencia y la potencia de intercambio.

Finalmente, con el controlador ANFIS se tuvo el escenario ideal: fue el mejor, tanto para los casos sin interconexión como para los que sí se consideró. Además, no fue necesario hacer un escalamiento al pasar de un sistema a otro, con lo que se tiene un controlador funcional, superior al PI, y robusto.

El buen funcionamiento del controlador ANFIS se puede explicar gracias a que, como se explicó en el capítulo 2, consta de dos tipos de funciones adaptables, primero una capa de funciones de pertenencia y luego una función lineal multivariable.

Adaptar la capa de funciones de pertenencia permite al controlador ser robusto ante varios escenarios, ya que modificar dichas funciones permite determinar de buena manera la escala o tamaño del sistema (o variables de entradas) de forma automática, para luego en la segunda capa, generar una salida correspondiente para controlar la frecuencia.

Por lo tanto, se puede concluir que el controlador ANFIS no solo funciona, sino que es recomendable para este sistema, por todas las características mencionadas, además de tener mejor rendimiento que el contro-

lador PI. Además de esto, solo fue necesario el escalamiento inicial (básicamente solo para darle una noción de en que rangos se encuentran las variables de entrada), con lo que en teoría, al utilizar este controlador, ya no sería necesario ningún tipo de estudio inicial para determinar los parámetros del controlador PI, lo cual es una gran ventaja.

Sin embargo, se entiende que por restricciones técnicas, como la implementación de este sistema a los equipos de control, no sea la alternativa favorita.

7.1. Trabajo a futuro

Como trabajo futuro se pueden sugerir muchas cosas a implementar. Se pueden intentar esquemas de control diferentes, como el modelamiento inverso utilizando los métodos inteligentes mostrados en este trabajo, para luego utilizarlo como controlador del sistema. De la misma forma, se habló en el capítulo 6 de la implementación de redes neuronales flexibles, que permitirían mejorar la escalabilidad del controlador ANN dejándolo al mismo nivel que el controlador ANFIS. Pero la mayor ventaja de estos controladores radica en su inteligencia y versatilidad, podría ser ocupado para predecir por ejemplo, la variabilidad del sistema dada ciertas variables meteorológicas o eléctricas, de modo de adelantarse al evento y producir señales que mejoren las condiciones del sistema de forma de contrarrestar los efectos posteriores de dicho evento. Para ello serían necesarias nuevas consideraciones en cuanto a la estructura de control, como se optimiza, etc.

Finalmente cabe señalar, que esto se trabaja y mejora día a día, donde nuevos métodos y aplicaciones surgen de forma periódica, ya que ha ganado un gran interés de parte de la industria y de la comunidad académica. Vale la pena entonces, tener en cuenta estas herramientas para nuevos problemas que el futuro entregue.

A

Optimización por cúmulo de partículas

El algoritmo de optimización por cúmulo de partículas se basa en fundamentos similares a los de otros métodos evolutivos, como lo es el algoritmo genético por ejemplo.

Está inspirado en el comportamiento social de los pájaros, que se mueven al rededor de un punto buscando comida. Moverse al rededor de forma aleatoria no representa una buena estrategia, pero seguir al que esté mas cerca de la comida si lo es. Por lo tanto se tiene un grupo o cúmulo de partículas, que son evaluadas en cada iteración, de modo de determinar que tan cerca están del óptimo. Esta evaluación genera un valor de desempeño en cada una de las partículas, con lo que se puede determinar cúal es la que logró los mejores resultados.

Gracias a lo anterior se puede hacer una modificación a los patrones de *movimiento* que tienen el resto de elementos del cúmulo, con lo que todos se repartirían a zonas donde hay probables óptimos de la función a optimizar.

Entonces, como se dijo anteriormente, las partículas siguen patrones de *movimiento*, por lo que tienen posición y velocidad, donde la posición es el vector de solución mismo, y la velocidad es usada para actualizar dicha posición. Concretamente la posición y velocidad de la partícula i en la iteración k están sujetos a las expresiones A.1 y A.2 correspondientemente:

$$v_i^{k+1} = \omega \cdot v_i^k + c_1\phi_1 \cdot (p_i^k - x_i^k) + c_2 \cdot \phi_2 \cdot (p_{gi}^k - x_i^k) \tag{A.1}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{A.2}$$

Donde:

v_i^{k+1} : velocidad de la particula.

ω : Valor arbitrario de inercia.

ϕ : Valor aleatorio entre 0 y 1.

p_i^k : Mejor partícula.

p_{gi}^k : Mejor partícula del cúmulo.

Cabe destacar que este algoritmo también cae en las fallas del algoritmo genético, como lo es llegar a óptimos locales.

Todo el proceso termina luego de que se alcanza un óptimo, de modo que la mejora entre iteraciones es menor a un ϵ dado por defecto. El diagrama de flujo del algoritmo puede ser visualizado en la figura A.1 .

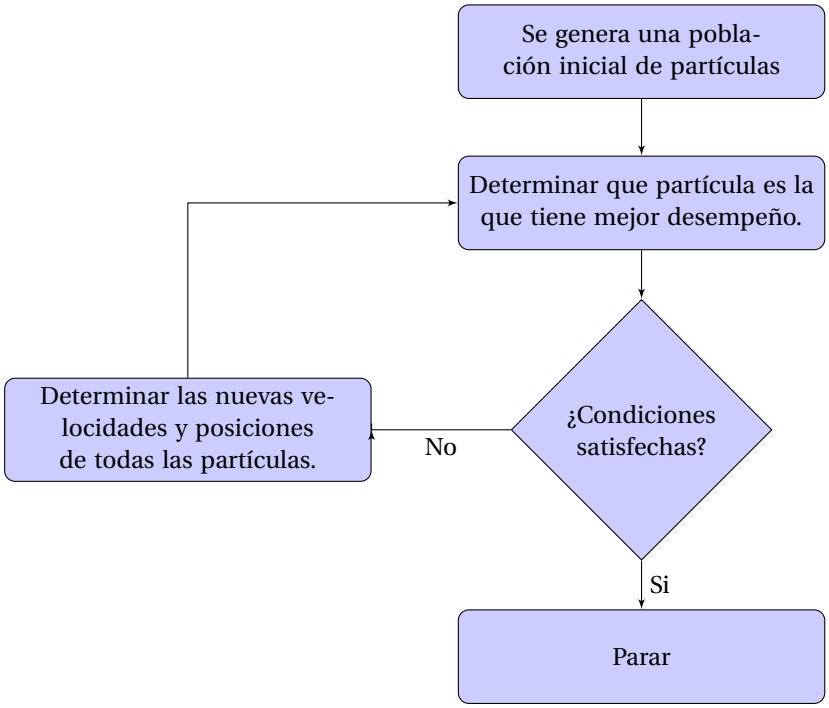


Figura A.1: Diagrama de flujo para el algoritmo PSO.

B

Controlador basado en lógica difusa

El objetivo de un controlador es obtener una salida correcta dada una entrada, que permita ser llevada al actuador, y de ese modo compensar o corregir el funcionamiento del sistema. Por ello, tiene sentido preguntarse que tipo de inferencia se está haciendo con los valores de la entrada.

En esta sección se explica como funciona la lógica difusa y fusificación, el sistema de reglas, y el proceso de defusificación.

B.1. Lógica difusa y fusificación

En la lógica tradicional los conjuntos pueden ser representados bajo variables binarias (pertenecen o no pertenecen a un conjunto). Esto cambia en la lógica difusa, donde la variable puede pertenecer hasta *cierto punto* a más de alguno de los conjuntos. Si se considera la variable μ como el *grado de pertenencia a un conjunto*, dado por una función de pertenencia dada, se tendría que en la lógica tradicional toma valores binarios: 0 o 1, mientras que en la lógica difusa se tendrían valores entre ellos.

Dichas funciones de pertenencia pueden ser descritas en base a parámetros que definan su forma, dependiendo del tipo de curva que sean.

B.1.1. Función de pertenencia Triangular

Una función de pertenencia triangular puede ser definida mediante los parámetros a,b,c como se presenta en la ecuación B.1:

$$f(x) = \begin{cases} 0 & si \quad x \leq a \\ \frac{x-a}{b-a} & si \quad a \leq x \leq b \\ \frac{c-x}{c-b} & si \quad b \leq x \leq c \\ 0 & si \quad c \leq x \end{cases} \tag{B.1}$$

donde b es el vértice superior de la función de pertenencia triangular, y a y c representan los vértices de la base.

B.1.2. Función de pertenencia Trapezoidal

La función de pertenencia trapezoidal puede ser definida según los parámetros a,b,c,d siguiendo la expresión B.2:

$$f(x) = \begin{cases} 0 & si \quad x \leq a \\ \frac{x-a}{b-a} & si \quad a \leq x \leq b \\ 1 & si \quad b \leq x \leq c \\ \frac{d-x}{d-c} & si \quad b \leq x \leq c \\ 0 & si \quad c \leq x \end{cases} \tag{B.2}$$

donde a,b,c,d representan las cuatro esquinas del trapezoide.

B.1.3. Función de pertenencia Gaussiana

La función de pertenencia Gaussiana puede ser definida por los parámetros c y σ según la expresión B.3:

$$f(x) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{B.3}$$

donde la función dependiente de dichos parámetros pueden ser visualizados de mejor forma en la figura B.1:

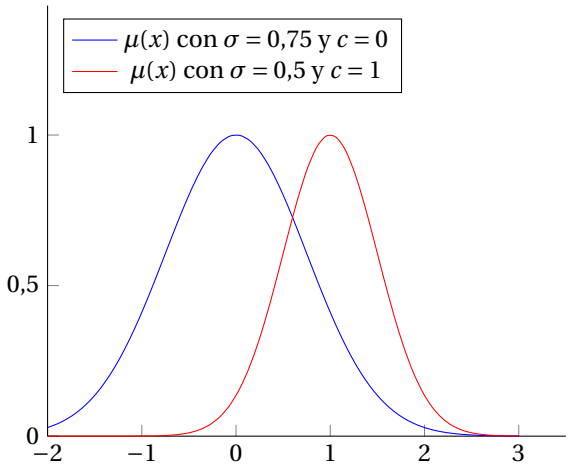


Figura B.1: Función de pertenencia del tipo Gaussiana, con diferentes valores de σ y c

La ventaja principal de la lógica difusa está en que permite utilizar cuantificadores lingüísticos, los cuales hacen que las consideraciones sean más simples de entender y sean más cercanos a los modelos reales.

Por ejemplo, si se quisiera hablar de un valor *alto* o *bajo*, primeramente sería necesaria una definición de dichos conjuntos. En la lógica tradicional la pertenencia a estos conjuntos sería abrupta, ya que el valor sólo puede pertenecer a una de las definiciones. Si se considerara que existen estos dos conjuntos, y se definen de forma tal que los valores mayores a 2 son altos y los menores son bajos se definirían funciones de membresía como las de la figura B.2. A la hora de definir las acciones a realizar, se tendría que la acción a realizar cuando el valor es menor e infinitamente cercano a 2 es totalmente diferente a la que se realizaría si el valor es 2.

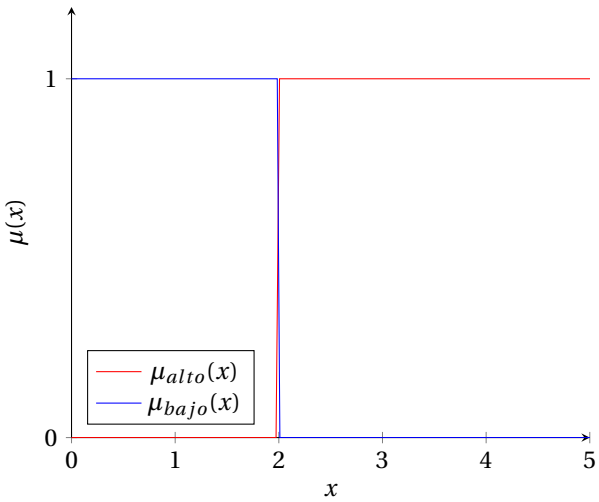


Figura B.2: Grado de pertenencia al conjunto alto y bajo con lógica tradicional.

Si se deseara obtener el grado de pertenencia de $x = 1{,}5$ se tendría:

$$\mu_{bajo}(1{,}5) = 1 \quad \mu_{alto}(1{,}5) = 0 \tag{B.4}$$

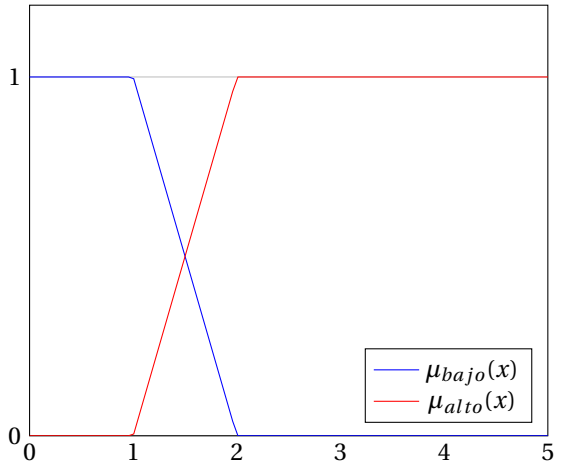


Figura B.3: Grados de pertenencia de los conjuntos altos y bajos en lógica difusa.

Al contrario de lo anterior, en la lógica difusa se pueden tener conjuntos más reales como el de la figura B.3. Esto debido a la concepción de que los valores no siempre pertencen totalmente a un solo conjunto, sino a varios en un distinto grado de pertenencia.

Si ahora nuevamente se definieran conjuntos, pero esta vez las funciones de pertenencia fueran como los de la figura B.3, haciendo la misma evaluación para $x = 1{,}5$ se tendría que el grado de pertenencia a ambos es:

$$\mu_{bajo}(1{,}5) = 0{,}5 \quad \mu_{alto}(1{,}5) = 0{,}5 \tag{B.5}$$

Con lo que se puede concluir, que utilizando esta lógica se puede llevar a cabo una acción de control mucho más fina, debido a una representación mas precisa de la variable de entrada.

Las variables de entrada que son conocidas y exactas toman el nombre de valores *certeros*, están contenidas en un *universo de discurso* y su evaluación en las funciones de membresía son valores *difusos*. Por ello este proceso es llamado fusificación.

B.2. Reglas Difusas

Luego de obtenidos los grados de pertenencia de las variables de entrada, es necesario imponer un *sistema de reglas* que permita tomar decisiones. El tipo de reglas utilizado en sistemas difusos son llamadas reglas *SI-ENTONCES*, y son definidas de la siguiente forma:

$$Si\ x_1\ es\ A\ y/o\ x_2\ es\ B,\ y\ es\ C$$

Esto luego es tomado por el sistema de inferencia, que genera las relaciones entre entradas y salidas difusas. Para ello las operaciones AND son tomadas como el mínimo o la multiplicación, y la operación OR como el máximo de los valores de las entradas. Por ejemplo, la operación para valores de entrada $x_1 = 0$ y $x_2 = 0{,}6$ y con la regla *si* x_1 *es* A_1 *y* x_2 *es* B_2, en las funciones de membresía mostradas en la figura B.4, daría como salida:

$$\mu_{A_1}(0) = 0{,}5 \quad AND \quad \mu_{B_2}(0{,}6) = 1 \quad = 0{,}5 \tag{B.6}$$

Este proceso debería hacerse tantas veces como reglas definidas en el sistema.

B.3. Generación de la salida

Luego de obtenidas las evaluaciones en todas las reglas, se procede a evaluar la salida. Para ello se debe tomar el valor arrojado por la operación anterior y ser evaluada en la función de membresía de la salida correspondiente, y con ello cortar el gráfico, de modo tal de obtener un conjunto de valores.

Siguiendo el ejemplo anterior, pero agregando a la regla el comportamiento de la salida: *si* x_1 *es* A_1 *y* x_2 *es* B_2, *y es* C_2, se puede observar que se entra con el valor a la función de membresía de la salida, y con ella se obtiene un conjunto.

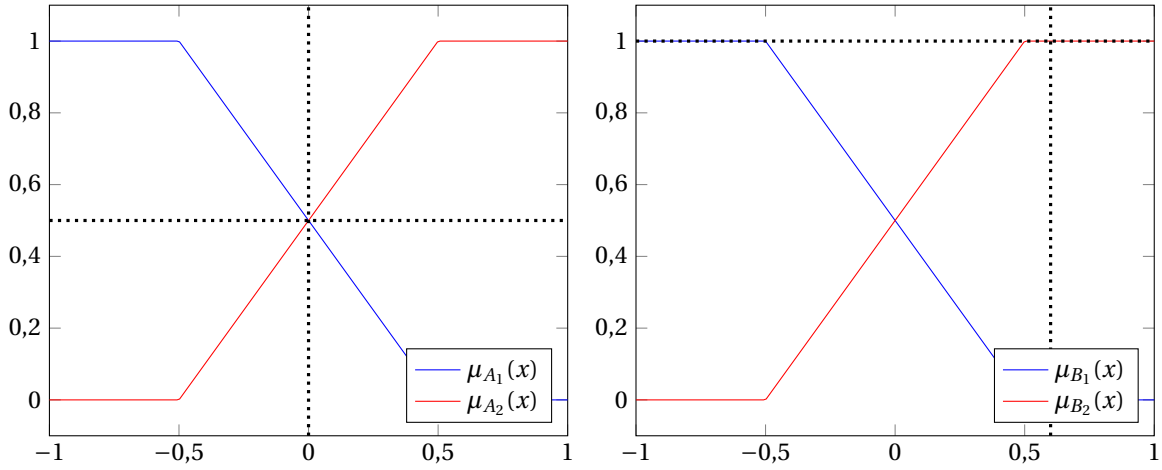


Figura B.4: Grados de membresía para $x_1 = 0$ y $x_2 = 0{,}6$.

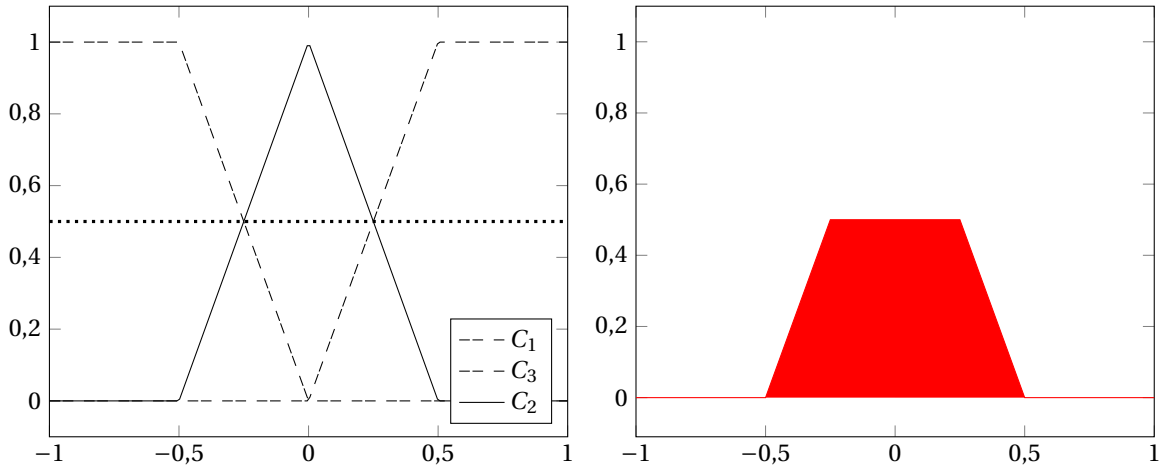


Figura B.5: Evaluación de la regla en la función de membresía de la salida para $x_1 = 0$ y $x_2 = 0{,}6$, implica un corte sobre la función misma C_1

Como se mencionó anteriormente, esto se hace para cada regla, con lo que el conjunto resultante es la unión de todos ellos.

Debido a que lo que se necesita a la salida es un valor certero o exacto, y no un conjunto, se realiza una operación a éste para obtenerlo (defusificación) como el método de centroide, que es el mayormente ocupado.

B.4. Ejemplo de aplicación

A modo de ejemplo se va a explicar como se hace todo el proceso de generación de una salida certera, a partir de valores certeros de entrada. Para ello se ocupará el ejemplo que aparece en la página de MATLAB [5].

Para ello, considérese el caso en que se trata de obtener la propina que se va a entregar dependiendo de si el servicio es: bueno (good), malo (poor), o excelente (excellent), y de si la comida es: rancia (rancid), o deliciosa (delicious).

Ahora, si se considera entonces un sistema difuso completo con las siguientes reglas:

- Si el servicio es **malo** o la comida es **rancia** entonces la propina es **poca**.
- Si el servicio es **bueno** entonces la propina es **promedio**.
- Si el servicio es **excelente** o la comida es **deliciosa** entonces la propina es **generosa**.

La tercera regla se evaluaría como se muestra en la figura B.6: si el valor del servicio fuese 3 y la comida 8, se obtiene que los grados de pertenencia son 0 y 0.7 correspondientemente. Al aplicar el operador OR (que sería el máximo de ambos), se tiene que la salida es 0.7.

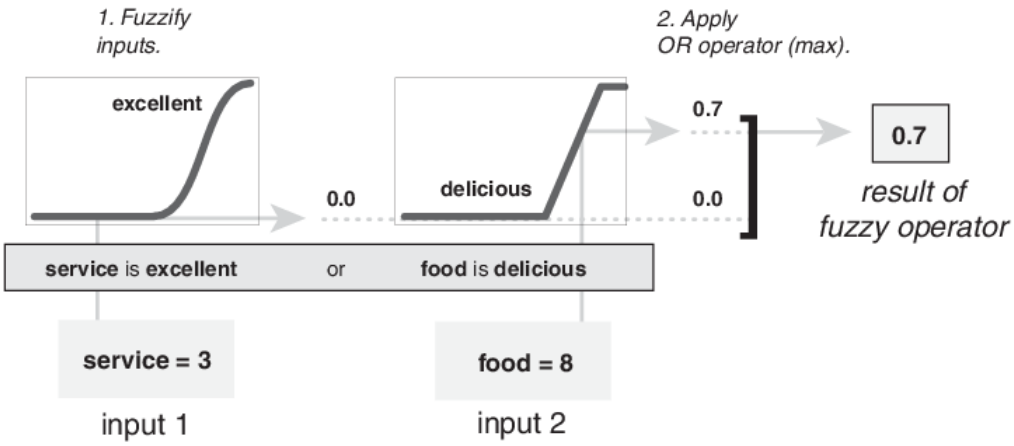


Figura B.6: Ejemplo de evaluación de una regla en lógica difusa.

Lo que sigue es evaluar en el resto de las reglas. Por lo tanto si se observa la figura B.7, se ve que se evalúa cada una de las reglas (con el valor de servicio en 3 y la comida en 8), y el resultado corta de forma horizontal la función de pertenencia de la salida correspondiente según la regla. Con ello, se tienen 3 funciones recortadas que por método de adición se juntan, generando una salida correspondiente.

Debido a que finalmente lo necesario es tener un valor certero de dicho resultado, es necesario aplicar una operación sobre dicho valor, que en este caso será el centroide, dando un valor final a la salida como puede observarse en la figura B.8

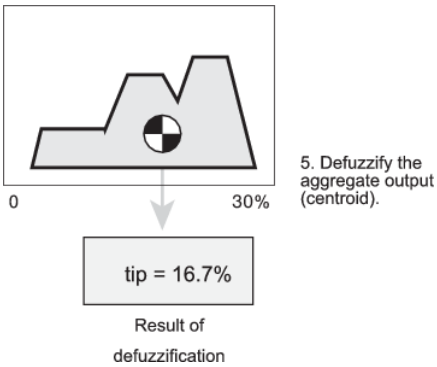


Figura B.8: Aplicación del método de centroide a la figura obtenida como resultado.

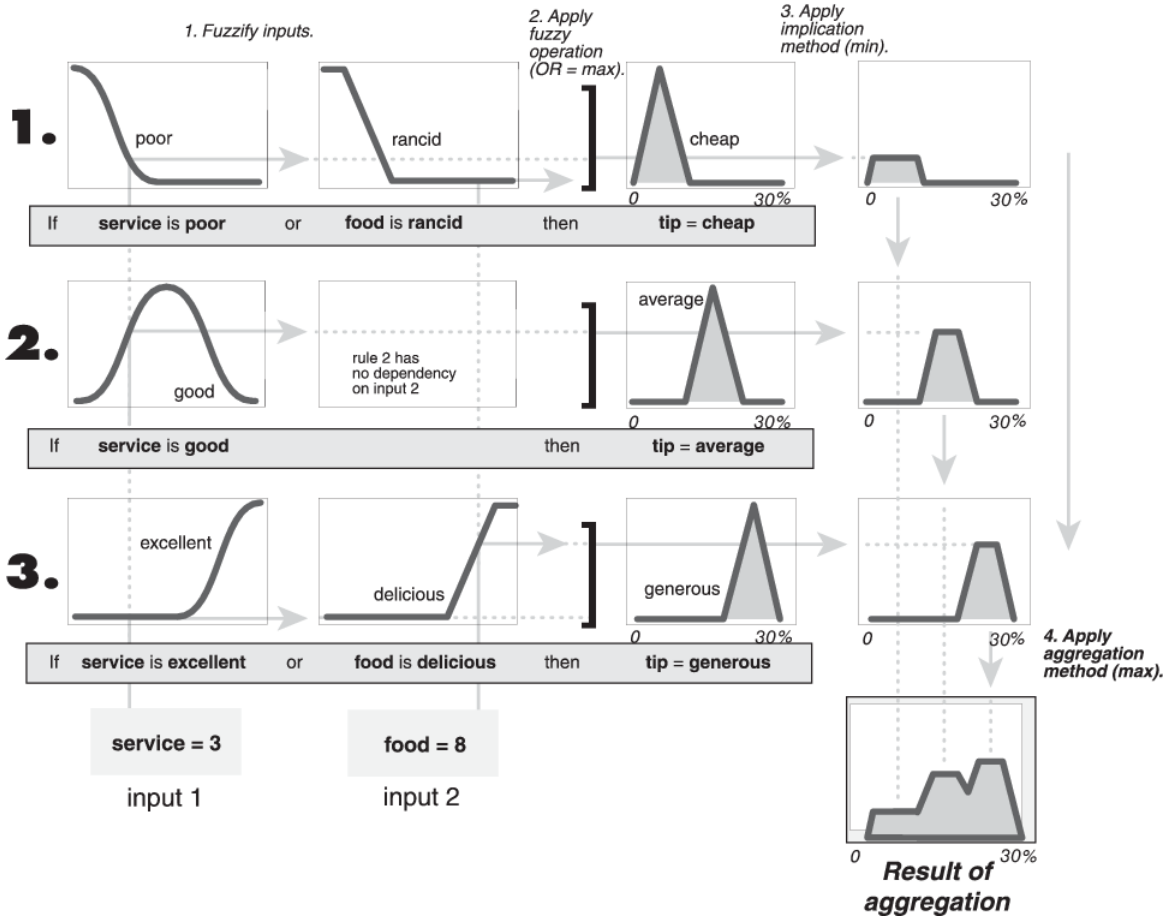


Figura B.7: Ejemplo de aplicación para la lógica difusa.

C

Redes Neuronales Artificiales

Las redes neuronales artificiales nacieron en los años 40, cuando un grupo de cientificos intentó adaptar modelos matemáticos de modo de asemejar el funcionamiento de aprendizaje de las redes neuronales reales [6]. Uno de los problemas clave de estos sistemas fue reconocido en los años 60s, donde se hacía referencia a que era necesario un gran poder de procesamiento para utilizarlos de forma útil [7]. Con ello su desarrollo se desaceleró y no despertó mayor interés en la comunidad científica, hasta entrado los años 80s, donde se tuvieron computadoras de mayor potencia y además del desarrollo del algoritmo de propagación inversa [8] (backpropagation), que suponía una forma eficiente y rápida de entrenar el sistema de redes neuronales.

Desde entonces las redes neuronales se han aplicado a una variedad gigante de problemas, en especial en aquellos donde la programación tradicional, donde se debía programar que debía hacer un sistema exactamente (se le llama *hard computing* en inglés) tenía problemas, ya que no era posible diseñar un único algoritmo que pudiera funcionar para todos los casos puntuales, como lo serían por ejemplo: predicciones, reconocimiento de patrones, etc.

Por estas características de aprendizaje y funcionamiento, resulta atractivo poder ocuparlo en sistemas de control, como el de este estudio.

C.1. Modelo de perceptron o neurona

La unidad más básica de un modelo de redes neuronales es el perceptron o neurona. Su comportamiento trata de imitar el de una neurona normal. Dicha comparación se puede clarificar observando la imagen de la figura C.1.

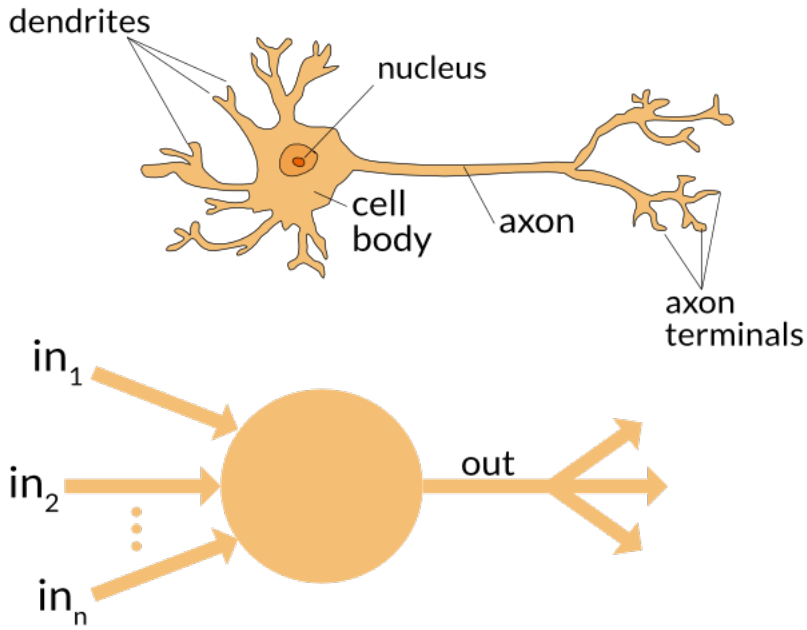


Figura C.1: Comparación entre una neurona biológica y una artificial.

Se tiene que las neuronas biológicas están conectadas a otras neuronas a partir de las dendritas y axones, donde su equivalente en el modelo artificial serían pesos o *weights* en inglés. Luego de que se tiene cierta información de entrada ésta es procesada (evaluada en una función de activación) para luego dar una salida, que luego se transmite a otras neuronas.

Cuando se desea que cierta neurona no actúe ante cierta señal, se debe **inhibir** el canal que ocupa para llegar a ser evaluada, por ello se debe cambiar su peso, de modo de que tenga mayor o menor importancia ante su evaluación posterior. De ese modo, algunas neuronas desarrollan funcionalidades específicas para el sistema que se está aplicando.

Otro aspecto importante del diseño de las redes neuronales es la elección de las funciones de activación de las neuronas. Por lo general el uso de funciones derivables como *funciones sigmoides* o tangentes hiperbólicas son suficientes para este proceso.

Considerando lo anterior, una forma completa de representar un perceptron puede ser visualizada en la figura C.2, donde se ve de izquierda a derecha: una capa de entrada, que adicionalmente considera un valor 1 (escrito como b) en la primera fila que representa el *bias*. Luego se hace una multiplicación de las entradas con los pesos que son sumados conformando un valor que se representa con la letra z, posteriormente es evaluado en una función de activación, que en el caso de que no llegue a otros perceptrones se llama salida o y, o en caso contrario, que si vaya a otros perceptrones, se le llama a este valor: a.

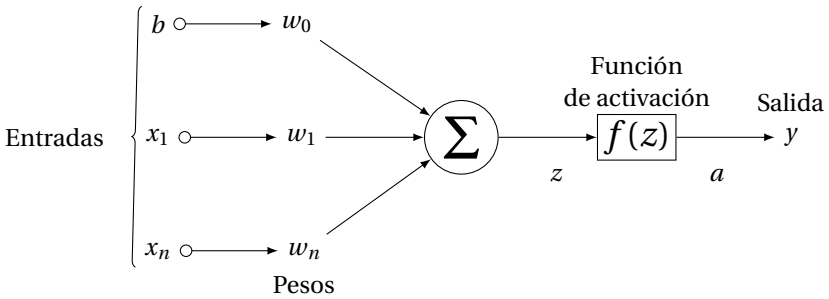


Figura C.2: Modelo perceptrón.

La inclusión del bias se hace debido a que su incorporación permite controlar de mejor forma el valor ingresado a la función de activación. Análogo a tratar de representar una recta: hacer un ajuste lineal a una serie de puntos es mucho más simple utilizando una función del tipo $f(x) = ax+b$ que sólo usando $f(x) = ax$.

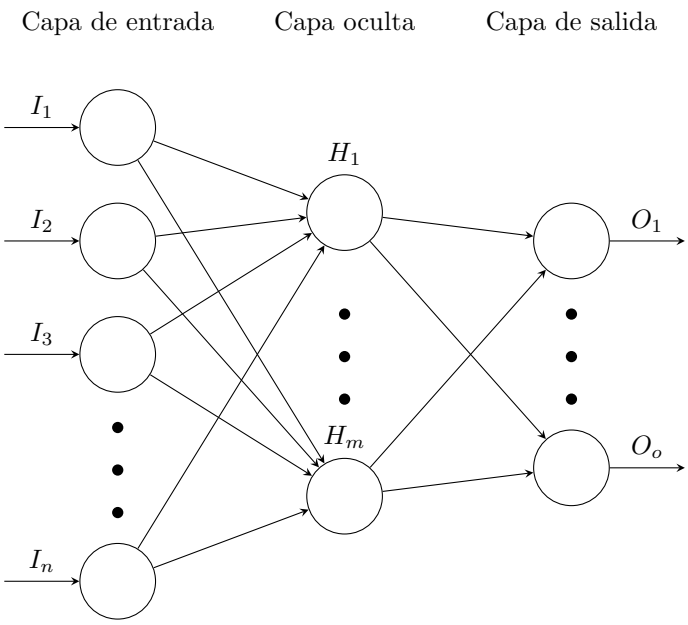


Figura C.3: Esquema de red neuronal de una capa oculta.

Además como se verá posteriormente, agregar el bias no implica mayores recursos computacionales en su optimización.

C.2. Capas de neuronas

Una columna de neuronas se llama capa, y la utilización de las redes neuronales implica el apilamiento horizontal de estas capas, de modo que las evaluaciones de las capas se propaguen hacia adelante en un proceso llamado propagación hacia adelante o *feedforward* en inglés. Siempre se tiene una capa de entrada y una de salida, además de una capa entremedio que recibe el nombre de capa oculta. Si una red neuronal contiene más de una capa oculta, el proceso se categoriza como aprendizaje profundo o *deep learning* en inglés. En este estudio se utilizarán redes de una capa oculta por lo que no es necesario hacer consideraciones adicionales correspondientes a los procesos de entrenamiento de aprendizaje profundo.

Una red neuronal con una capa oculta, con i entradas, j neuronas en la capa oculta, y k salidas, puede ser representada por un esquema como el de la figura C.3.

La salida de los perceptrones de la primera capa corresponde a un vector $a^1 = (1, a_1^1, a_2^1, ..., a_n^1)$, que en los casos de este estudio se cumplirá que son iguales a las entradas, por lo que en esta capa:

$$f(x) = x \Rightarrow a_i^1 = I_i \tag{C.1}$$

Siguiendo con la propagación hacia adelante, se tiene que las entradas a cada uno de los perceptrones de la segunda capa, representados en el vector $z^2 = (1, z_1^2, z_2^2, ..., z_k^2)$, se obtienen de sumar las entradas que llegan a ese perceptron multiplicados por sus respectivos pesos:

$$z_1^2 = w_{01}^1 \cdot 1 + w_{11}^1 \cdot a_1^1 + w_{21}^1 \cdot a_2^1 + ... + w_{n1}^1 \cdot a_n^1$$

$$z_2^2 = w_{02}^1 \cdot 1 + w_{12}^1 \cdot a_1^1 + w_{22}^1 \cdot a_2^1 + ... + w_{n2}^1 \cdot a_n^1$$

$$\vdots$$

$$z_m^2 = w_{0m}^1 \cdot 1 + w_{1m}^1 \cdot a_1^1 + w_{2m}^1 \cdot a_2^1 + ... + w_{nm}^1 \cdot a_n^1 \tag{C.2}$$

Lo cual de forma matricial se escribe:

$$z^2 = [1, a_1^1, a_2^1, ..., a_i^1] \cdot \begin{bmatrix} w_{01}^1 & w_{02}^1 & w_{03}^1 & ... & w_{0m}^1 \\ w_{11}^1 & w_{12}^1 & w_{13}^1 & ... & w_{1m}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 & ... & w_{2m}^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{n1}^1 & w_{n2}^1 & w_{n3}^1 & ... & w_{nm}^1 \end{bmatrix} = a^1 \cdot W^1 \tag{C.3}$$

Evaluando el vector z^2 punto a punto en la función de activación de la capa oculta, se tendría el vector de salida de dicha capa a^2. Siguiendo la propagación, ahora para obtener las salidas, se aplicaría nuevamente lo mismo de forma matricial:

$$\hat{y} = z^3 = a^2 \cdot W^2 \tag{C.4}$$

C.3. Descenso por gradiente (Gradient Descend)

El método de descenso por gradiente es un algoritmo de minimización. Es muy popular en el uso de machine learning, debido a que da paso a otros algoritmos que permiten el ajuste de parámetros de forma eficiente.

Su principio se basa en evaluar cuanto varía el error al cambiar sus parámetros de modo de determinar hacia que dirección deberían hacerse modificaciones.

Si se considera por ejemplo una función de costos J como la de la figura C.4, que depende de una variable θ, se tiene que al situarse en un punto dado de dicha gráfica, la derivada de dicha función indica si el costo o error sube o baja al incrementar θ. En el punto A la derivada es negativa, por lo tanto los cambios en el parámetro deberían ser incrementando su valor, todo lo contrario se tiene en B donde si θ sigue aumentando, su error subiría.

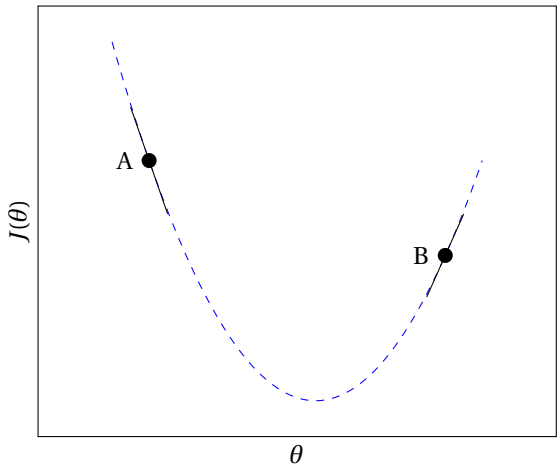


Figura C.4: Función de error demostrando que el algoritmo se aproxima hacia puntos que disminuyan su error.

Considerando la explicación hecha previamente, una modificación a cada elemento del vector de parámetros que disminuya el error estaría dada por la ecuación (C.5):

$$\theta_i^{k+1} = \theta^k - \eta \frac{\partial J(\theta_i)}{\partial \theta_i} \tag{C.5}$$

Donde η es un factor que permite hacer mayor o menor las modificaciones entre iteraciones. Este factor en algunos textos es llamado tasa de aprendizaje o *learning rate*, y su ajuste depende del caso a optimizar.

Ejemplo de aplicación Dado un conjunto de puntos, se espera obtener una regresión lineal tal que haga una buena representación de dichos puntos. Para ello se hará una demostración en Matlab de cada parte para demostrar como funciona el algoritmo.

Para este caso primeramente se van a generar 10 puntos que representen la curva:

$$y = \theta_1 x + \theta_0 = 2x + 5 \tag{C.6}$$

Por lo que se desearía encontrar valores del vector θ que minimicen el error dado por la diferencia entre la predicción actual del modelo (representado por $\hat{y}_i$) con el valor correcto (y_i), para un número n de muestras (en el caso de ejemplo serían 10):

$$J(\theta) = \sum_{i=1}^{n} \frac{1}{2N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \frac{1}{2N}(y_i - f_i(\theta))^2 \tag{C.7}$$

Se divide en dos solo para hacer mas simple la expresión de la derivada. Con lo que las derivadas parciales de la función de error, tanto para θ_1 como θ_2 serían:

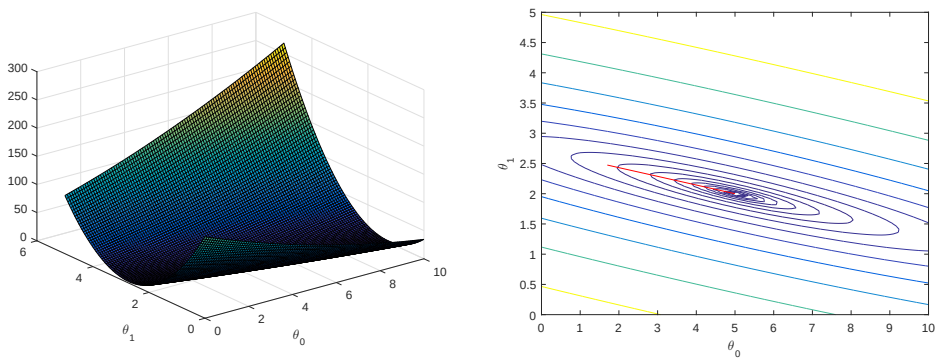
$$\frac{\partial J(\theta)}{\partial \theta_1} = \sum_{i=1}^{n} \frac{1}{N}(y_i - \hat{y}_i) \cdot x_i \tag{C.8}$$

$$\frac{\partial J(\theta)}{\partial \theta_2} = \sum_{i=1}^{n} \frac{1}{N}(y_i - \hat{y}_i) \cdot 1 \tag{C.9}$$

Utilizando las expresiones anteriores de forma iterativa hasta llegar al vector θ_{opt} tal que J se minimiza, se tiene el vector óptimo, que debería ser cercano a los puntos que se definieron en un principio a modo de ejemplo:

$$\theta_{opt} = [4{,}9990, 2{,}0001] \tag{C.10}$$

Si se visualiza el error en función de los parámetros θ, se tienen las figuras C.5a y C.5b, en ella se evidencia la intención del algoritmo, que es acercarse al punto más bajo del gráfico, teniendo como información hacia que dirección la bajada es mas pronunciada (gracias a la gradiente). Por lo anterior, algo sumamente importante para este algoritmo es la continuidad y diferenciabilidad de la función de error. Otra cosa que se puede observar es que este algoritmo puede caer en mínimos locales, ya que sigue la depresión más cercana al punto donde se encuentre. Estos problemas se pueden arreglar utilizando métodos de gradientes estócasticas, pero estos métodos requieren realizar la optimización más de una vez.



(a) Gráfico en 3D del error en función de los parámetros.

(b) Curva de nivel del error en función de los parámetros.

Figura C.5

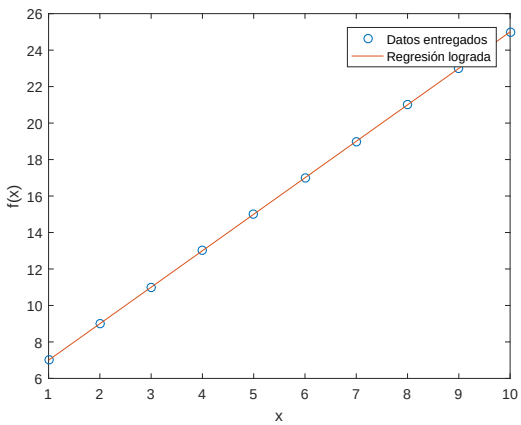


Figura C.6: Regresión lineal lograda con el algoritmo y los valores reales entregados.

Finalmente en la figura C.6 se puede apreciar la regresión lograda gracias al algoritmo. Con ello se concluye que mientras la función sea diferenciable y continua, el algoritmo de bajada por gradiente es muy poderoso y útil para la determinación de parámetros.

C.4. Método de propagación inversa (backpropagation)

El método de propagación inversa o backpropagation en inglés, es el método más utilizado para entrenar redes neuronales. Básicamente es la aplicación del algoritmo de descenso por gradiente presentado anteriormente, aplicado a la evaluación en cadena de las redes neuronales.

Si se considera por ejemplo una red neuronal de una capa oculta con sus dos matrices de pesos correspondientes W^1 y W^2, con un vector de entrada $\vec{x}$, y una salida $\hat{y}$. Gracias a lo explicado anteriormente en la propagación hacia adelante, se podría determinar la salida en una sola línea:

$$\hat{y} = f_3(f_2(a^1 \cdot W^1) \cdot W^2) = f_3(a^2 \cdot W^2) = f(z^3) \tag{C.11}$$

Esta salida puede variar del valor que se desea realmente con dichas entradas. A esta variación se le llamará error, y estará definida de la siguiente forma:

$$Error = J(W^1, W^2) = \frac{1}{2}(y - \hat{y})^2 \tag{C.12}$$

Por lo tanto, para determinar las matrices de pesos W^1 y W^2, tal que minimizan el error entre lo que entrega la red neuronal y lo que realmente se desea, utilizando descenso por gradiente, es necesario determinar las derivadas parciales del error con respecto a cada uno de los pesos:

$$\frac{\partial J}{\partial W^1} = \begin{bmatrix} \frac{\partial J}{\partial w^1_{01}} & \frac{\partial J}{\partial w^1_{02}} & \cdots \\ \frac{\partial J}{\partial w^1_{11}} & \frac{\partial J}{\partial w^1_{12}} & \cdots \\ \frac{\partial J}{\partial w^1_{21}} & \frac{\partial J}{\partial w^1_{22}} & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{C.13}$$

$$\frac{\partial J}{\partial W^1} = \begin{bmatrix} \frac{\partial J}{\partial w^1_{01}} & \frac{\partial J}{\partial w^1_{02}} & \cdots \\ \frac{\partial J}{\partial w^1_{11}} & \frac{\partial J}{\partial w^1_{12}} & \cdots \\ \frac{\partial J}{\partial w^1_{21}} & \frac{\partial J}{\partial w^1_{22}} & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{C.14}$$

$$\frac{\partial J}{\partial W^2} = \begin{bmatrix} \frac{\partial J}{\partial w^1_{01}} \\ \frac{\partial J}{\partial w^1_{11}} \\ \frac{\partial J}{\partial w^1_{21}} \\ \vdots \end{bmatrix} \tag{C.15}$$

Para desarrollar entonces las derivadas parciales, primero es necesario un valor de $\hat{y}$, por lo que es necesario hacer una evaluación de la red neuronal con las entradas.

Desarrollando entonces la expresión primeramente para W^2 para un j cualquiera:

$$\frac{\partial J}{\partial W^2_{j1}} = \frac{1}{2}\frac{\partial}{\partial W^2_{jk}}(y - f_3(a^2 \cdot W^2))^2 = (y - f_3(a^2 \cdot W^2)) \cdot -f_3'(a^2 \cdot W^2) \cdot \frac{\partial(1 \cdot W_{01} + a^2_1 \cdot W_{11} + a^2_2 \cdot W_{21} + ... + a^2_j \cdot W_{j1})}{\partial W^2_{j1}} \tag{C.16}$$

Se tiene que la expresión de la izquierda es constante, por lo que se podría representar como δ^3, con ello cada una de las ecuaciones que determinan las derivadas parciales podrían ser desarrolladas con facilidad:

$$\begin{aligned} \frac{\partial J}{\partial W^2_{01}} &= \delta^3 \cdot 1 \\ \frac{\partial J}{\partial W^2_{11}} &= \delta^3 \cdot a^2_1 \\ &\vdots \\ \frac{\partial J}{\partial W^2_{j1}} &= \delta^3 \cdot a^2_j \end{aligned} \tag{C.17}$$

Que podría ser representado de forma matricial, como lo muestra la ecuación (C.18):

$$\left[\frac{\partial J}{\partial W^2}\right] = \delta^3 \cdot f'(z^3) \cdot \begin{bmatrix} 1 \\ a^2_1 \\ a^2_2 \\ \vdots \end{bmatrix} = \delta^3 \cdot (a^2)^T \tag{C.18}$$

Para obtener las derivadas parciales con respecto a W^1 se puede partir de las expresiones anteriores, teniendo en cuenta que los valores a^2 dependen de W^1 de la siguiente forma:

$$a^2 = f_2(z^2) = f_2(a^1 \cdot W^1) = f_2(x \cdot W^1) \tag{C.19}$$

Por lo que se deberían seguir haciendo las operaciones de regla de la cadena. Para resumir esto se tiene entonces para valores i, j cualquiera se tiene la ecuación (C.20):

$$\frac{\partial J}{\partial W^1_{ij}} = \delta^3 \frac{\partial(a^2_j \cdot W^2_{j1})}{\partial W^1_{ij}} = \delta^3 \cdot f_2'(z^2_j) \cdot W^2_{j1} \frac{\partial(1 \cdot W^1_{0j} + x_1 \cdot W^1_{1j} + ... + x_n 1 \cdot W^1_{nj})}{\partial W^1_{ij}} = \delta^3 \cdot f_2'(z^2_j) \cdot W^2_{j1} x_j \tag{C.20}$$

Recordando que:

$$x_0 = 1$$

Para dejar la ecuación C.20 de forma matricial, primeramente se conforma un δ^2, que queda desarrollado de la forma:

$$\begin{aligned} \delta^2_1 &= \delta^3 \cdot f_2'(z^2_1) \cdot W^2_{11} \\ \delta^2_2 &= \delta^3 \cdot f_2'(z^2_2) \cdot W^2_{21} \end{aligned}$$

$$\vdots$$

$$\delta^2_m = \delta^3 \cdot f'_2(z^2_m) \cdot W^2_{m1} \tag{C.21}$$

Por lo tanto, δ^2 queda definido como un vector. También se puede observar que tiene una dimensión de m elementos, sin contar el bias, esto por que el peso que llega al elemento 0 no existe (ver figura de red neuronal), por ello la definición matricial de δ^2 queda:

$$\delta^2 = \delta^3 \cdot f'_2(z^2).W^2_{sin\ primer\ elemento} \tag{C.22}$$

Donde el punto se refiere a una multiplicación de Hadamard, o *element-wise multiplication* en inglés. Finalmente la estructura de la matriz de derivadas parciales con respecto a W^1 queda de la forma:

$$\frac{\partial J}{\partial W^1} = \begin{bmatrix} \delta^2_1 \cdot 1 & \delta^2_1 \cdot 1 & ... & \delta^2_1 \cdot 1 \\ \delta^2_1 \cdot x_1 & \delta^2_1 \cdot x_2 & ... & \delta^n_1 \cdot x_n \\ \vdots & \vdots & \vdots & \vdots \\ \delta^2_m \cdot 1 & \delta^2_m \cdot x_2 & ... & \delta^2_m \cdot x_n \end{bmatrix} = (\delta^2 \cdot x)^T \tag{C.23}$$

D

Controlador Híbrido ANFIS

Se observa que hay virtudes y defectos en ambas estructuras, por un lado se tiene -lógica difusa- que el poder para hacer decisiones usando definiciones lingüisticas es algo muy interesante de lograr, ya que con solo definir los grupos, el sistema puede tomar decisiones acertadas, ya que la inferencia sobre la respuesta se hace mediante una matemática dada. Sin embargo es necesario un ajuste fino con respecto a la creación de dichos grupos, lo cual puede acarrear un problema adicional al diseñar un controlador o una lógica deseada.

Por otro lado, se tiene un sistema que aprende de forma autónoma como debe ser su salida, sin mayores ajustes sobre los parámetros o sobre su estructura como tal. Pero carece de claridad sobre la generación de la salida.

La tabla comparativa D.1 puede resumir los temas mencionados anteriormente de forma simple:

Redes neuronales	**Lógica difusa**
✓La adaptación de parámetros puede ser hecha desde cero.	Es necesaria cierta información a priori.
✓Existen muchos algoritmos de aprendizaje/optimización de parametros.	El aprendizaje no es posible con la lógica difusa.
El comportamiento no es comprensible (el sistema en sí actúa como una caja negra).	✓Comportamiento comprensible.

Tabla D.1: Tabla comparativa entre las redes neuronales y la lógica difusa.

Por lo comentado anteriormente, Jang, a principio de los 1990s [9], postuló un sistema híbrido que contemplaba la unión de las dos virtudes mencionadas anteriormente, al cual le llamó ANFIS (*Adaptive Network-based Fuzzy Inference System*). Dicho sistema contempla la incorporación de la lógica difusa, pero con parámetros que son ajustables mediante conexiones neuronales como los mencionados anteriormente, en la descripción de la lógica de las redes neuronales.

D.1. Arquitectura básica de una red adaptable

Como se mencionó anteriormente, una red adaptable está hecha a base de nodos interconectados, donde cada uno de los nodos genera una única salida a partir de los valores de entrada que llegan a dicho nodo. Los nodos realizan entonces, una evaluación dependiendo de la función que tenga asociado dicho nodo, y que además puede tener parámetros que pueden ser fijos o modificables.

Una red típica de una configuración de red adaptable puede ser visualizada en la figura D.1 , en ella se puede observar la composición de varios nodos, anidados en capas que propagan un valor hacia adelante.

En la literatura, y en el desarrollo de este trabajo, los nodos circulares representan nodos que no son adaptables, por lo tanto su valor de salida sólo depende de los valores de entrada de dicho nodo. Por otro lado, el cuadrado simboliza un nodo adaptable, con lo que los parámetros de su función de evaluación pueden ser modificados.

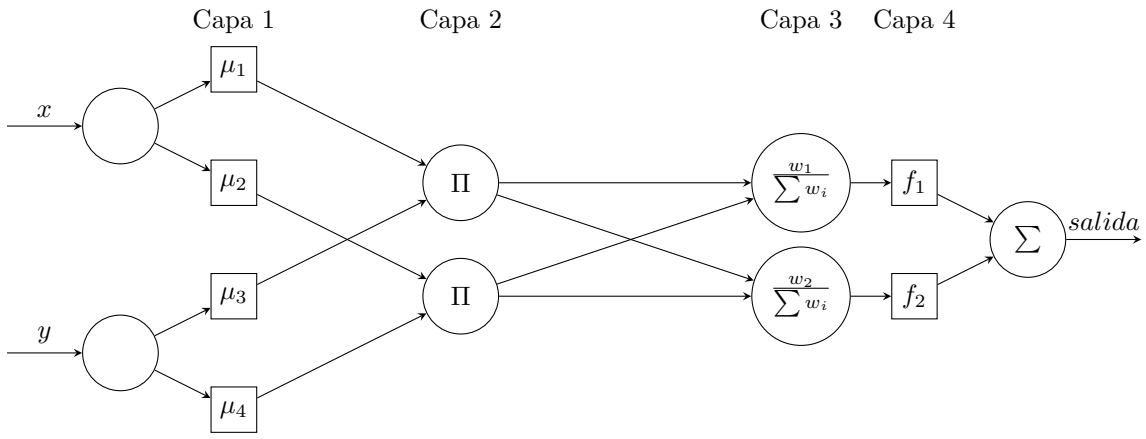


Figura D.1: Esquema de red neuro-difusa.

D.2. Estructura de un modelo ANFIS

Como se mencionó anteriormente el desarrollo de un modelo ANFIS está compuesto por capas, donde cada una de ellas presenta una funcionalidad diferente.

En la capa 1, cada nodo i es del tipo adaptable, con una salida que representa una función de pertenencia, que está dado por:

$$O_i^1 = \mu_{Ai}(x) \tag{D.1}$$

donde x es la entrada al nodo i, y A_i es la definición o variable lingüistica asociada a dicho nodo. Explicado de otra manera la salida O_i^1 representa el grado de pertenencia a dicha definición lingüistica. En éste trabajo se usarán funciones del tipo Gaussiano, ya que son diferenciables, continuas, y sin cambios abruptos, lo cual la hace una mejor candidata para consideraciones posteriores, en cuanto al algoritmo de aprendizaje u optimización.

En la capa 2, se presentan funciones del tipo multiplicatorias, ya que en sí representan las reglas AND, tal como se habían desarrollado en los sistemas de inferencia explicados anteriormente. La salida se toma como un peso w, que es representación de la fuerza de disparo de dicha regla (o sea que tanto se cumple).

$$O_i^2 = \Pi\, \mu_i \tag{D.2}$$

En la capa 3, se normalizan los valores de pesos con respecto a la sumatoria de todos los nodos. En sí la salida de un nodo de la capa 3 sería entonces:

$$O_i^3 = \frac{w_i}{\sum w_i} \tag{D.3}$$

En la capa 4, cada nodo i es una función lineal dada por las entradas iniciales (de la primera capa), y por los parámetros adaptables p, q, r, que además va multiplicado por el peso normalizado de la salida de la capa anterior, por lo tanto, simplificado en una expresión quedaría:

$$O_i^4 = w_n(p_i x + q_i y + r_i) \tag{D.4}$$

En la capa 5, simplemente se tiene que la salida es la sumatoria de las entradas.

$$\sum_{i=1}^{k} O_i^4 \tag{D.5}$$

Definida la estructura, se tiene la arquitectura necesaria para realizar una evaluación de modo de propagar los valores hacia adelante sin problemas. Sin embargo, es necesario un algoritmo de modo de adaptar los valores modificables, de modo que resuelvan minimizar el error en que están incurriendo al hacer dicha evaluación.

D.3. Propagación hacia atrás (Backpropagation), para la optimización de los parámetros.

Si se tiene una red adaptable de L capas, donde la k-ésima capa está configurada con #(k) nodos, donde $\#(\cdot)$ representa la cantidad de nodos de dicha capa. Se puede escribir la salida de un nodo i, en una capa k de la forma:

$$O_i^k = O_i^k(O_i^{k-1}, ..., O_{\#(k-1)}^{k-1}, a, b, c, ...) \tag{D.6}$$

Lo cual denota que O_i^k es una función que depende de las salidas de las capas anteriores ($O_i^{k-1}, ..., O_{\#(k-1)}^{k-1}$), y -si corresponde a un nodo adaptable- de sus propios parámetros modificables a, b, c, etc. Considerando entonces el error de la salida, como la diferencia entre el valor deseado (designado por T) y el valor obtenido por la evaluación de la red completa, o dicho de otro modo, la salida del nodo final (O^L), se tendría entonces:

$$E = \sum_{m=1}^{\#(k+1)} \frac{1}{2}(T_i - O_i^L)^2 \tag{D.7}$$

donde en el caso de una salida #(L) corresponde a 1, ya que la cantidad de nodos en la capa L (la última capa), sería 1.

Como se desarrolló anteriormente para los casos de redes neuronales, el método de aplicación de la propagación hacia atrás, consiste en obtener las derivadas del error con respecto a los parámetros que se desean modificar. Para ello en primera instancia se tiene que la tasa de error (o derivada parcial del error) con respecto a la salida final (o capa L) es:

$$\frac{\partial E}{\partial O_i^L} = -(T_i - O_i^L) \tag{D.8}$$

Las derivadas parciales del error con respecto a la salida de los nodos internos i, de una capa k cualquiera, están dadas por la suma de las derivadas parciales de la capa que le sucede:

$$\frac{\partial E}{\partial O_i^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E}{\partial O_m^{k+1}} \frac{\partial O_m^{k+1}}{\partial O_i^k} \tag{D.9}$$

Con lo que sólo quedaría continuar la regla de la cadena, ahora para un parámetro modificable cualquiera α. Entonces finalmente se tiene que la derivada del error con respecto a dicho parámetro es:

$$\frac{\partial E}{\partial \alpha} = \sum_{O*} \frac{\partial E}{\partial O*} \frac{\partial O*}{\partial \alpha} \tag{D.10}$$

E

Códigos para la optimización usando PSO

A continuación se presenta el código del script en Python *RunSimulationFLC.py*:

```
import os


from sys import executable


# Add powerfactory.pyd path to python path.
# This is an example for 32 bit PowerFactory architecture.
from sys import path

path.append("C:\\Program Files\\DIgSILENT\\PowerFactory 15.2\\Python\\3.3\\")
#import PowerFactory module
import powerfactory as pf

app = pf.GetApplication()


#app.Show()
project = app.ActivateProject("20170127 AGC Escs SING - FUZZY")
prj = app.GetActiveProject()

ini = app.GetFromStudyCase('ComInc')
sim = app.GetFromStudyCase('ComSim')

elmres = app.GetFromStudyCase('OptimizationResults.ElmRes')



print("iniciando simulacion")
ini.Execute()
sim.Execute()

comres = app.GetFromStudyCase('ComRes');

comres.f_name = r'C:\Memoria\resultados.txt'

comres.Execute()
print("simulacion terminada")
```

El propósito de dicho script es realizar las siguientes tareas:

- Abir una instancia de Digsilent, donde se carga al inicio el archivo *.fis*.
- Se realiza una simulación de un caso dado.
- Se guardan los resultados de la simulación (ACE) en un archivo, que en este caso estaría ubicado en *C:/Memoria/resultados.txt*.

- Se cierra dicha instancia.

Recordar que Digsilent carga el archivo .fis solo al inicio del programa, por eso es necesario que para cada simulación se tenga que abrir de nuevo.

Luego, se tiene el código del script en Python *PyControllerOptimization.py*:

```
import os
import string
import numpy as np
import timeit
import time
import imp

import subprocess

from sys import executable
from pyswarm import pso
from subprocess import Popen, CREATE_NEW_CONSOLE

import math
# Add powerfactory.pyd path to python path.
# This is an example for 32 bit PowerFactory architecture.
from sys import path


path.append("C:\\Program Files\\DIgSILENT\\PowerFactory 15.2\\Python\\3.3")
#import PowerFactory module
import powerfactory as pf

def getPI(x):
    if len(x)==3:
        bACE=x[0]
        bdACE=x[1]
        bout=x[2]
        writeFIS(generateStringFiveMF(bACE,bdACE,bout))

        SW_HIDE = 0
        info = subprocess.STARTUPINFO()
        info.dwFlags = subprocess.STARTF_USESHOWWINDOW
        info.wShowWindow = SW_HIDE


        p1= subprocess.Popen([executable, 'RunSimulationFLC.py'], creationflags=CREATE_NEW_CONSOLE,
     startupinfo=info)
        p1.wait()
        csv = np.genfromtxt ('C:\\Memoria\\resultados.txt', delimiter=',')
        csv = np.delete(csv, (0), axis=0)
        ACE = csv[:,1]
        PI= np.linalg.norm(ACE)

    return PI;

def writeCSV(stringToWrite):
    with open('C:\\Memoria\\IteracionesFLC\\iteraciones.csv', 'a') as file:
        file.write(stringToWrite)



def writeFIS(fileString):

    # Write the file out again
    with open('C:\\Program Files\\DIgSILENT\\PowerFactory 15.2\\fisfile.fis', 'w') as file:
        file.write(fileString)

def FiveMFInputs(b):
    header = "Range=[-1 1]\nNumMFs=5\n"
    MF1_line = "MF1='MN':'trapmf',["+str(-1.569)+ " " + str(-1.138) + " " + str(-b) + " " + str(-b/2) + "
     ]\n"
    MF2_line = "MF2='N':'trimf',["+str(-b)+ " " + str(-b/2) + " " + str(0) + "]\n"
    MF3_line = "MF3='Z':'trimf',["+str(-b/2)+ " " + str(0) + " " + str(b/2) +  "]\n"
    MF4_line = "MF4='P':'trimf',["+str(0)+ " " + str(b/2) + " " + str(b) +  "]\n"
```

```
    MF5_line = "MF5='MP':'trapmf',["+str(b/2) + " " + str(b) + " " + str(1.4) + " " + str(1.5) + "]\n"
    return header+MF1_line+MF2_line+MF3_line+MF4_line+MF5_line

def FiveMFOutput(b):
    header = "[Output1]\nName='salida'\nRange=[-1 1]\nNumMFs=5\n"
    MF1_line = "MF1='MN':'trapmf',["+str(-1.569)+ " " + str(-1.138) + " " + str(-b) + " " + str(-b/2) + "
     ]\n"
    MF2_line = "MF2='N':'trimf',["+str(-b)+ " " + str(-b/2) + " " + str(0) + "]\n"
    MF3_line = "MF3='Z':'trimf',["+str(-b/2)+ " " + str(0) + " " + str(b/2) +  "]\n"
    MF4_line = "MF4='P':'trimf',["+str(0)+ " " + str(b/2) + " " + str(b) +  "]\n"
    MF5_line = "MF5='MP':'trapmf',["+str(b/2) + " " + str(b) + " " + str(1.4) + " " + str(1.5) + "]\n"
    return header+MF1_line+MF2_line+MF3_line+MF4_line+MF5_line

def generateStringFiveMF(bACE,bdACE,bout):
    fileheader="[System]\nName='fislibromod'\nType='mamdani'\nVersion=2.0\nNumInputs=2\nNumOutputs=1\
     nNumRules=25\nAndMethod='prod'\nOrMethod='probor'\nImpMethod='min'\nAggMethod='max'\nDefuzzMethod='
     centroid'\n\n"
    Input1="[Input1]\nName='ACE'\n"+FiveMFInputs(bACE)+"\n"
    Input2="[Input2]\nName='dACE'\n"+FiveMFInputs(bdACE)+"\n"
    Output=FiveMFOutput(bout)+"\n"
    rules="[Rules]\n1.000 1.000 , 1.000 (1.000) : 1\n2.000 1.000 , 1.000 (1.000) : 1\n3.000 1.000 , 2.000
     (1.000) : 1\n4.000 1.000 , 2.000 (1.000) : 1\n5.000 1.000 , 3.000 (1.000) : 1\n1.000 2.000 , 1.000
     (1.000) : 1\n2.000 2.000 , 2.000 (1.000) : 1\n3.000 2.000 , 2.000 (1.000) : 1\n4.000 2.000 , 3.000
     (1.000) : 1\n5.000 2.000 , 4.000 (1.000) : 1\n1.000 3.000 , 2.000 (1.000) : 1\n2.000 3.000 , 2.000
     (1.000) : 1\n3.000 3.000 , 3.000 (1.000) : 1\n4.000 3.000 , 4.000 (1.000) : 1\n5.000 3.000 , 4.000
     (1.000) : 1\n1.000 4.000 , 2.000 (1.000) : 1\n2.000 4.000 , 3.000 (1.000) : 1\n3.000 4.000 , 4.000
     (1.000) : 1\n4.000 4.000 , 4.000 (1.000) : 1\n5.000 4.000 , 5.000 (1.000) : 1\n1.000 5.000 , 3.000
     (1.000) : 1\n2.000 5.000 , 4.000 (1.000) : 1\n3.000 5.000 , 4.000 (1.000) : 1\n4.000 5.000 , 5.000
     (1.000) : 1\n5.000 5.000 , 5.000 (1.000) : 1"
    fileonString=fileheader+Input1+Input2+Output+rules
    return fileonString



def singleSimulation(filePath):
    p1= Popen([executable, 'RunSimulation.py'], creationflags=CREATE_NEW_CONSOLE)
    p1.wait()
    csv = np.genfromtxt (filePath, delimiter=';')

def con(x):
    bACE=x[0]
    bdACE=x[1]
    bout=x[2]
    return [bACE-aACE,bdACE-adACE]





def optimizeFLCFiveMFInputs(con):
    lb=[0.1,0.1,0.1] #bACE,bdACE,bout
    ub=[0.95,0.95,0.95] #bACE,bdACE,bout
    xopt, fopt = pso(getPI, lb, ub, ieqcons=[], f_ieqcons=None, args=(), kwargs={},swarmsize=30, omega
     =0.5, phip=0.5, phig=0.5, maxiter=15, minstep=1e-4,minfunc=1e-4, debug=True, excel_progress=True)

optimizeFLCFiveMFInputs(con)
```

Este script contiene varias funciones, la función getPI() obtiene el rendimiento de una configuración de controlador difuso (dados b_{ACE},$b_{\Delta ACE}$ y b_{out}). Para ello realiza una ejecución del script presentado anteriormente, y del archivo de resultados obtenidos se calcula la norma del vector de valores.

Las funciones *FiveMFInputs, FiveMFOutput, generateStringFiveMF* generan los textos que representan al archivo fis que luego es guardado dentro de la carpeta de Digsilent mediante la función *writeFIS*. Todo esto para que en la siguiente iteración se cargue una nueva configuración del controlador difuso, con nuevos parámetros.

F

Códigos de digexfun.dll para cargar nuevas funciones a Digsilent

Debido a que claramente los controladores Fuzzy, ANN y ANFIS no vienen incluidos en Digsilent, fue necesario que fueran programados a mano. A continuación se presenta el código contenido en el archivo *userfun.cpp*, que al ser compilado en un dll, puede ser integrado a Digsilent. No se muestra la lógica interna de las evaluaciones ya que no sería práctico, por lo que se presentan la inicialización, además de los sistemas de evaluación y de propagación inversa en general.

```


#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#include "digexfun.hpp"

#include <fl\Headers.h>

#include <fstream>
#include <iostream>
#include "ANN.h"
#include "ANFIS.h"


using namespace std;
using namespace fl;


fl::Engine* engine;
fl::InputVariable* inputVariable1;
fl::InputVariable* inputVariable2;
fl::OutputVariable* outputVariable;

ANFIS anfis;


double ACE_int;
double ACE_int2;
int contador;

//pointer to print function (set on Init() call when DLL is loaded)
FP_PRINTMSG pPrintFnc(NULL);
FP_STOPSIM pStopSimFnc(NULL);


// ************************* NN ***************************
Net NN;
```

```
int i = 0;


// ************************* fin NN *************************




// this method is called when loading dll into powerfactory
void __stdcall Init(FP_PRINTMSG _pPrintFnc, FP_STOPSIM _pStopSimFnc){
  pPrintFnc = _pPrintFnc;
  pStopSimFnc = _pStopSimFnc;
  print_pf("DLL CARGADA", 0);

  // se inicializa la carga del controlador Fuzzy
  engine = fl::FisImporter().fromFile("fisfile.fis");
  inputVariable1 = engine->getInputVariable(0);
  inputVariable2 = engine->getInputVariable(1);
  outputVariable = engine->getOutputVariable(0);


  // se inicializa la carga del controlador ANN
  std::vector<int> typeOfLayers;

  typeOfLayers.push_back(TYPE_SIGMOID);
  typeOfLayers.push_back(TYPE_SIGMOID);
  typeOfLayers.push_back(TYPE_SIGMOID);

  contador = 0;
  ACE_int = 0;
  ACE_int2 = 0;
  NN.Initialize(2, 5, typeOfLayers);

  // se inicializa la carga del controlador ANFIS
  anfis.initialize(3, 3, MF_GAUSS);

  anfis.setLearningRate(0.75);


}


int __stdcall RegisterFunctions(int ifun, char* cnam, int* iargc)
{ // register userdefined functions number ifun:
  //   return name (cnam[50]) and number of arguments (iargc)
  // return != 0, if ifun exceeds the number of available functions
  if (!cnam || !iargc) return 1;

  LoadDigsiLibrary();


  if (ifun == 0) {
    strcpy(cnam, "FuzzyController");
    *iargc = 2;

  }
  else if (ifun == 1) {
    strcpy(cnam, "Feedforward");
    *iargc = 3;
  }
  else if (ifun == 2) {
    strcpy(cnam, "Backpropagate");
    *iargc = 1;
  }
  else if (ifun == 3) {
    strcpy(cnam, "ANFISFeedforward");
    *iargc = 3;
  }

  else { // unknown function
    return 1;
```

```
  }

  return 0;
}


void __stdcall FuzzyController() {
  double ACE = pop(); // 1. argument
  double dACE = pop();      // 2. argument

  if (ACE > 1) {
    inputVariable1->setInputValue(1);
  }
  else if (ACE< -1) {
    inputVariable1->setInputValue(-1);
  }
  else {
    inputVariable1->setInputValue(ACE);
  }

  if (dACE > 1) {
    inputVariable2->setInputValue(1);
  }
  else if (dACE< -1) {
    inputVariable2->setInputValue(-1);
  }
  else {
    inputVariable2->setInputValue(dACE);
  }




  engine->process();

  scalar outputscalar = outputVariable->getOutputValue();
  push((double)outputscalar);

}

void __stdcall Feedforward() {

  double ACE = pop(); // 1. argument
  double dACE = pop();      // 2. argument
  int t_sampling = pop();
  vector<double> inputVals, resultVals;
  double salida;
  arma::mat inputs;



  inputs << ACE << dACE << arma::endr;

  NN.setInputs(inputs);

  //NN.inputs.print();

  salida = NN.feedForwardResult();

  contador = contador + 1;
  ACE_int2 = ACE_int2 + ACE;
  if(contador % t_sampling == 0 ){
    ACE_int2 = ACE_int2 / t_sampling;
    NN.BackpropagateError(ACE_int2);
    ACE_int2 = 0;
  }



  push(salida);

```

```
184 }
185
186 void __stdcall Backpropagate() {
187
188   double ACE = pop(); // 1. argument
189
190
191
192   push(1);
193
194 }
195
196 void __stdcall ANFISFeedforward() {
197
198   double ACE = pop(); // 1. argument
199   double dACE = pop();      // 2. argument
200   int t_sampling = pop();
201   double salida;
202
203
204   anfis.setInputs(ACE, dACE);
205
206   salida = anfis.feedForward();
207   contador = contador + 1;
208   ACE_int = ACE_int + ACE;
209   if (contador %t_sampling == 0) {
210     ACE_int = ACE_int / t_sampling;
211     anfis.backpropagate(-ACE_int);
212     ACE_int = 0;
213   }
214
215
216
217   push(salida);
218
219 }
```

G

Variabilidad de la demanda

La variabilidad de la demanda y la generación ERNC están configuradas utilizando un bloque como los que se presentan a continuación:

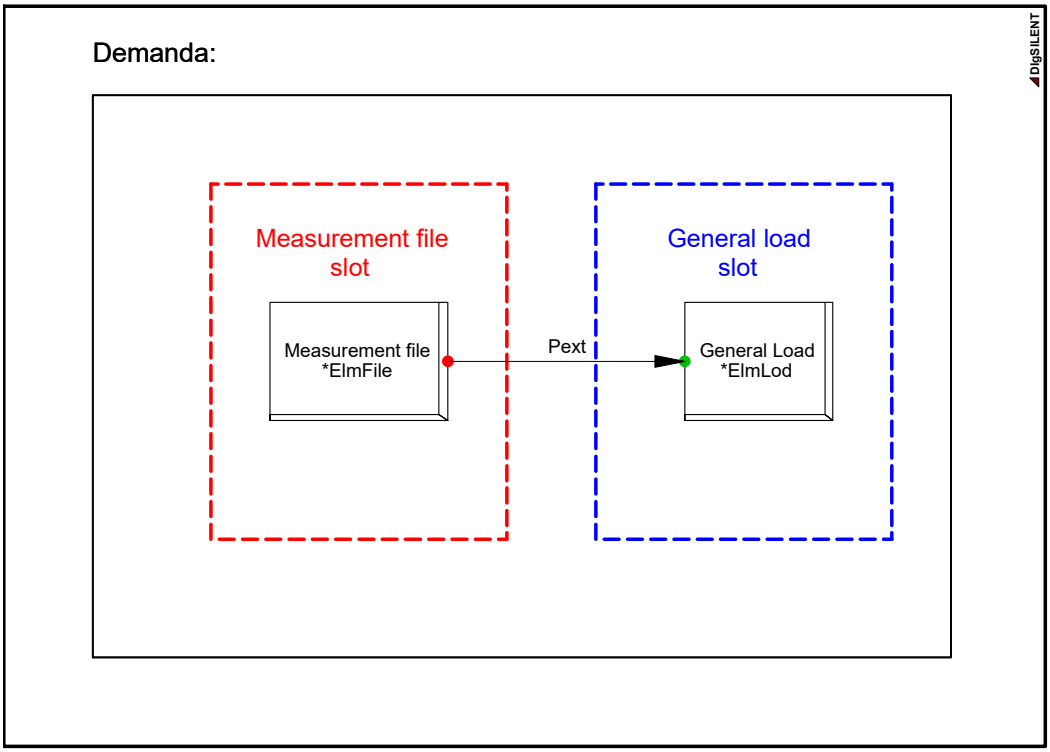


Figura G.1: Bloque de Digsilent que permite la lectura de demanda en el tiempo desde un archivo.

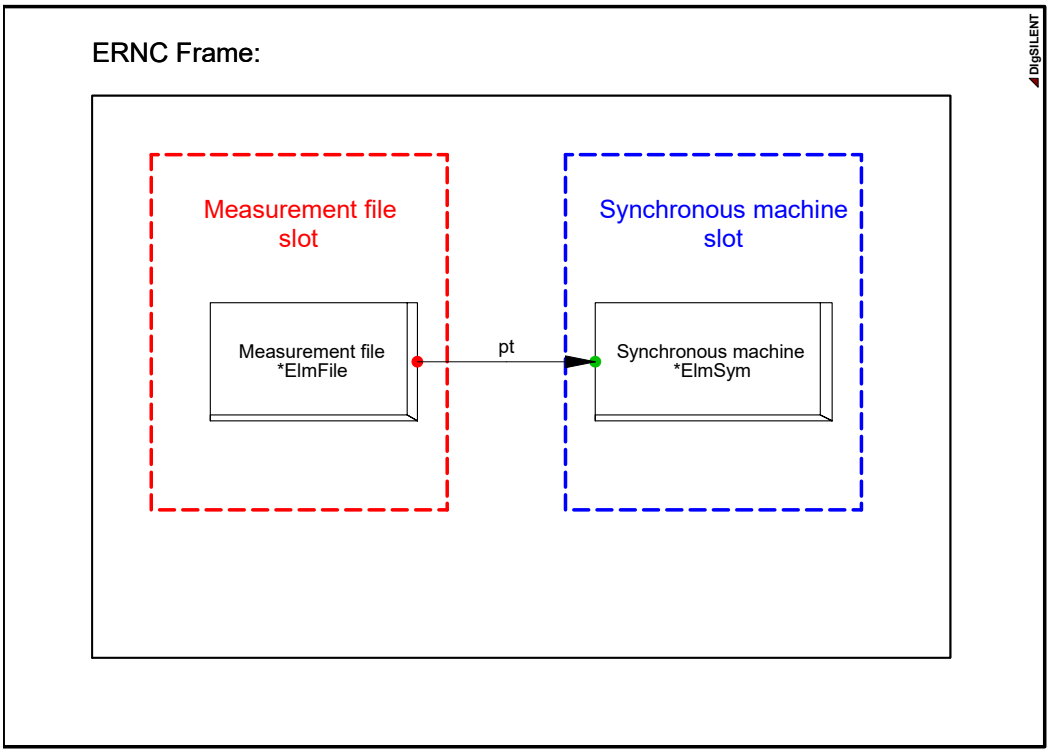


Figura G.2: Bloque de Digsilent que permite la lectura de la generación ERNC desde un archivo.

Las variaciones de demandas son leídas desde un archivo txt ubicado en *C:/AGC/Demanda.txt*, y la generación ERNC leída desde *C:/AGC/Escenarios*. La variación de demanda graficada en el tiempo puede visualizarse en la figura G.3:

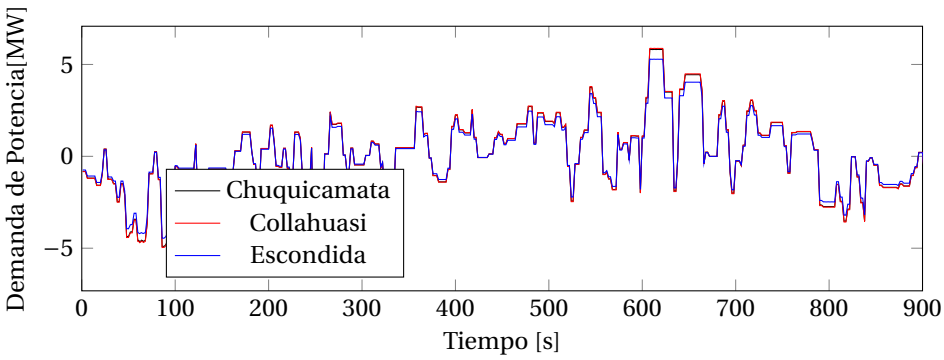


Figura G.3: Demandas variables en [MW], de las mineras Chuquicamata, Collahuasi y Escondida.

H

Perfiles de generación de ERNC

Los casos mostrados en este trabajo correspondientes a los escenarios 1.1, 3.1 y 4.1 consideran generación de parte de las centrales ERNC. De modo de hacer esto más ilustrativo, los perfiles de generación en el tiempo de dichas centrales se presentan a continuación (imágenes obtenidas de parte del estudio del AC3E).

H.1. 1.1 (Alta penetración de ERNC):

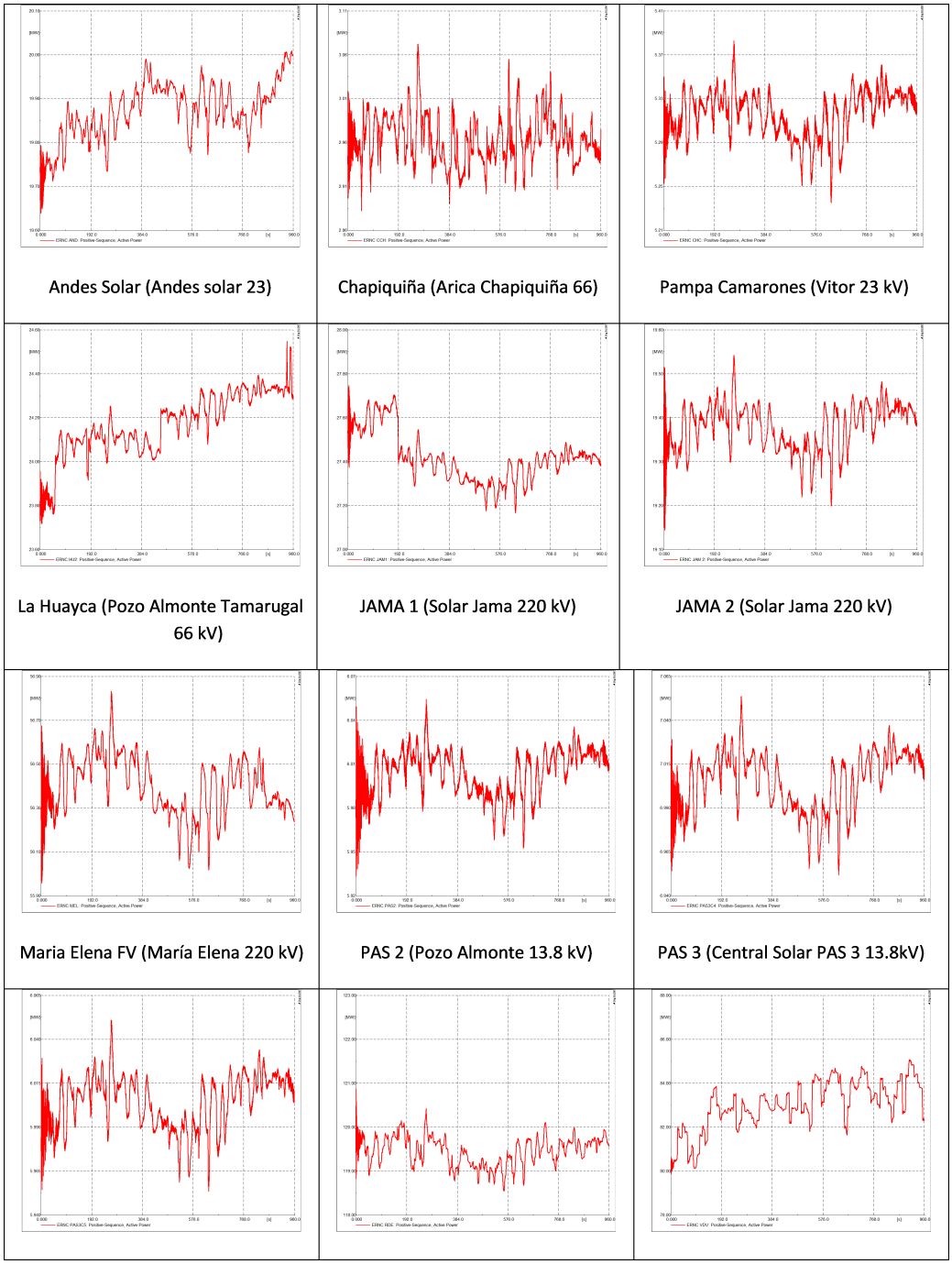
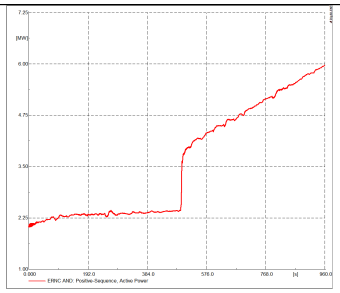
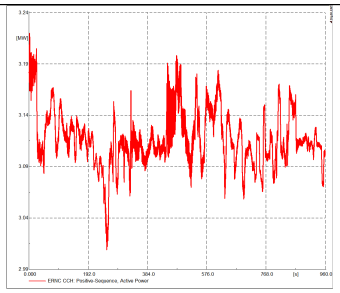


Figura H.1: Perfil de generación de las centrales para el caso con alta penetración de ERNC.

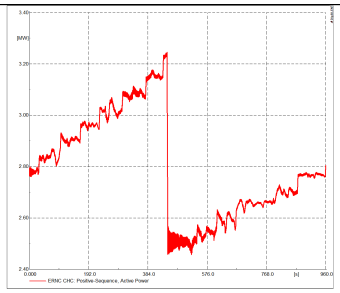
H.2. 3.1 (Entrada de ERNC):



Andes Solar (Andes solar 23)

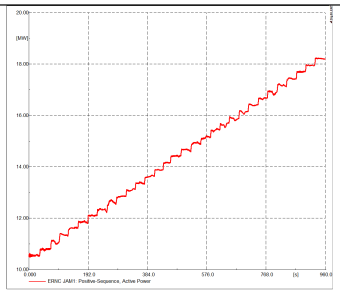


Chapiquiña (Arica Chapiquiña 66)

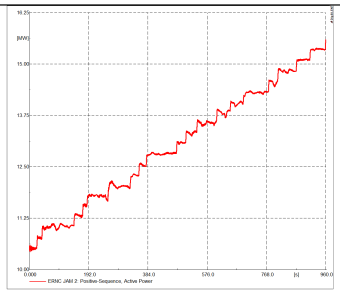


Pampa Camarones (Vitor 23 kV)

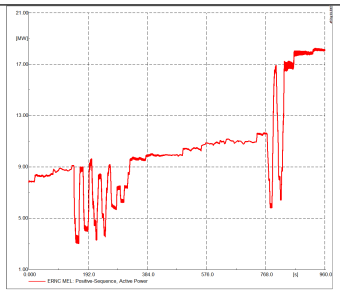
La Huayca (Pozo Almonte Tamarugal 66 kV)



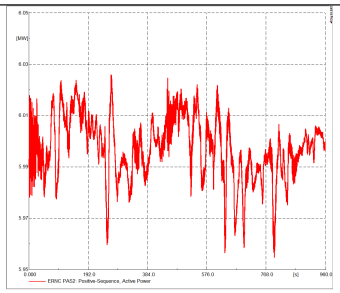
JAMA 1 (Solar Jama 220 kV)



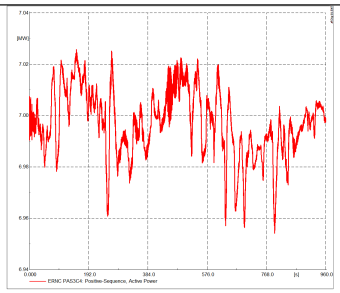
JAMA 2 (Solar Jama 220 kV)



Maria Elena FV (María Elena 220 kV)



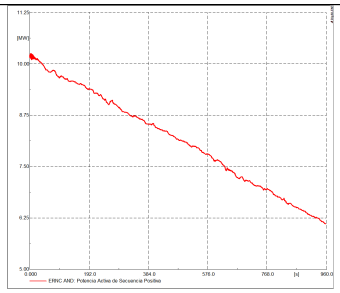
PAS 2 (Pozo Almonte 13.8 kV)



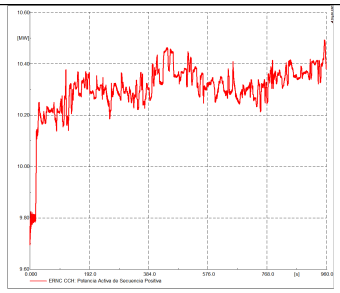
PAS 3 (Central Solar PAS 3 13.8kV)

Figura H.2: Perfil de generación de las centrales para el caso con entrada de ERNC.

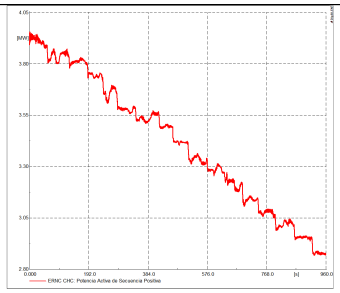
H.3. 4.1 (Salida de ERNC):



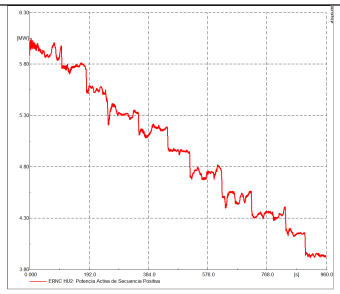
Andes Solar (Andes Solar 23)



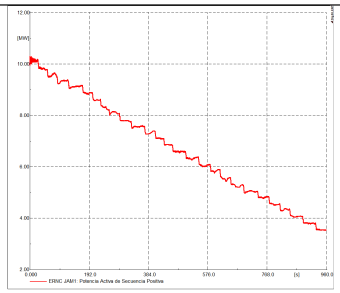
Chapiquiña (Arica Chapiquiña 66)



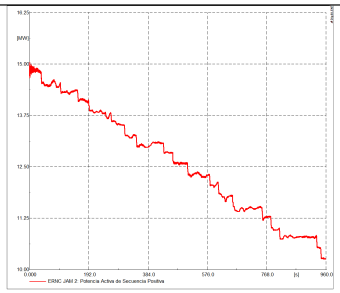
Pampa Camarones (Vitor 23 kV)



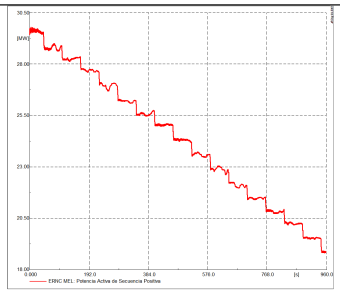
La Huayca (Pozo Almonte Tamarugal 66 kV)



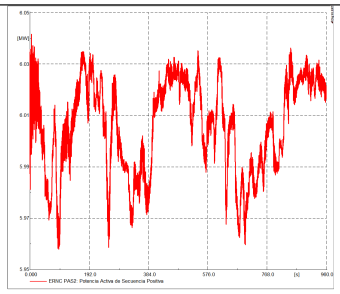
JAMA 1 (Solar Jama 220 kV)



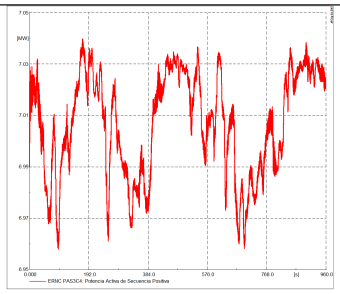
JAMA 2 (Solar Jama 220 kV)



Maria Elena FV (María Elena 220 kV)



PAS 2 (Pozo Almonte 13.8 kV)



PAS 3 (Central Solar PAS 3 13.8kV)

Figura H.3: Perfil de generación de las centrales para el caso con salida de ERNC.

Bibliografía

[1] W. Almeida, "Metodología para la determinación de parámetros para la sintonización del AGC en sistemas multi-área," 2004.

[2] J. venkatachalam y S. Rajalaxmi, "Automatic generation control of two area interconnected power system using particle swarm optimization," 2013.

[3] H. Bevrani y T. Hiyama, "Intelligent automatic generation control," 2011.

[4] J. Palacios, J. Velásquez, V. Hinojosa, y I. Calle, "Desarrollo de una herramienta para sintonización permanente del agc del sing, propuestas de parametrización y evaluación de las definiciones operativas para su implementación," 2017.

[5] MATLAB, "Matlab documentation, fuzzy inference process," 2017. [Online]. Available: https://www.mathworks.com/help/fuzzy/fuzzy-inference-process.html

[6] M. Warren y W. Pitts, "A logical calculus of ideas immanent in nervous activity bulletin of mathematical biophysics. 5 (4): 115–133," 1943.

[7] M. Marvin y P. Seymour, "Perceptrons: An introduction to computational geometry," 1969.

[8] Werbos. P.J, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1975.

[9] Jang. J, "Fuzzy modeling using generalized neural networks and kalman filter algorithm," 1991.