

ESCUELA POLITÉCNICA NACIONAL

**FACULTAD DE INGENIERÍA ELÉCTRICA Y
ELECTRÓNICA**

DESPACHO HIDROTÉRMICO DE CORTO PLAZO.

**HERRAMIENTA COMPUTACIONAL, EN LENGUAJE PYTHON,
PARA LA SOLUCIÓN DEL DESPACHO HIDROTÉRMICO DE
CORTO PLAZO CONSIDERANDO LA RED ELÉCTRICA.**

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO
REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO.**

KEVIN DAVID PARRAGA PICO

kevin.parraga@epn.edu.ec

DIRECTOR: Dr.-Ing. NELSON VICTORIANO GRANDA GUTIÉRREZ

nelson.granda@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

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

Kevin David Parraga Pico

Certifico que el presente trabajo de integración curricular fue desarrollado por Kevin David Parraga Pico, bajo mi supervisión.

Nelson Victoriano Granda Gutiérrez
DIRECTOR

DECLARACIÓN DE AUTORÍA

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

Kevin David Parraga Pico

Nelson Victoriano Granda Gutiérrez

DEDICATORIA

Este gran logro le dedico primordialmente a Dios y a la Virgen del Cisne por siempre guiarme por el camino del bien y bendecirme con la salud cada día de mi vida.

Con mucho amor a mis padres, Jorge y Amparito, que con su gran cariño y amor incondicional siempre me guiaron hacia adelante en busca de mis objetivos y me enseñaron a no rendirme para alcanzar este gran logro académico, gracias a su gran apoyo incondicional todo fue posible.

A mis hermanas, Leslie y Estefanía, gracias por compartir su gran sabiduría mediante consejos para mejorar cada día en mi etapa académica y por su gran preocupación brindándome motivación cada día a pesar de la distancia lo que ha sido un gran pilar para convertirme en una extraordinaria persona llena de valores y principios.

A toda mi familia, especialmente a mis abuelitos, tíos, primos les agradezco por siempre estar pendiente de mi persona y agradezco por su apoyo incondicional en esta gran etapa educativa de mi vida.

A mis grandes amigos de la universidad, que siempre me han apoyado en todas las decisiones y que han formado parte de esta gran trayectoria, siendo amigos incondicionales que han estado a mi lado en los momentos buenos y malos de mi vida a través de los años.

AGRADECIMIENTO

Quiero expresar mi más grande agradecimiento a mis padres que siempre me apoyaron desde el primer día de esta gran etapa de mi vida, a mis amigos que me acompañaron en este difícil camino lleno de buenas y malas experiencias en el transcurso de todos estos años en la universidad, a la prestigiosa Facultad de Ingeniería Eléctrica y Electrónica EPN, que me ha brindado los mejores docentes que han sido un pilar muy importante en la adquisición de nuevos conocimientos y en el aprendizaje continuo para mi formación personal y profesional.

Un agradecimiento especial al Dr. Ing. Nelson Granda por su tiempo, su guía, su experimentada orientación y sus consejos para ayudar a desarrollar este trabajo.

ÍNDICE DE CONTENIDO

CERTIFICACIONES	I
DECLARACIÓN DE AUTORÍA	II
DEDICATORIA	III
AGRADECIMIENTO	IV
ÍNDICE DE CONTENIDO.....	V
RESUMEN.....	VII
ABSTRACT.....	VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO	1
1.1 Objetivo general.....	1
1.2 Objetivos específicos	1
1.3 Marco teórico	2
1.3.1 Despacho Hidrotérmico de Corto Plazo	2
1.3.2 Flujo Óptimo de Potencia	3
1.3.3 Formulación matemática del DHT -CP con OPF -AC.....	3
1.3.3.1 Función Objetivo	3
1.3.3.2 Restricciones	5
1.3.3.2.1 Balance de potencia activa y reactiva	5
1.3.3.2.2 Límites de potencia activa generada.....	6
1.3.3.2.3 Límites de potencia reactiva generada.....	7
1.3.3.2.4 Restricciones de rampa de arranque y subida	7
1.3.3.2.5 Restricciones de rampa de parada y bajada	8
1.3.3.2.6 Lógica binaria de operación	8
1.3.3.2.7 Volumen máximo y mínimo de embalses	9
1.3.3.2.8 Balance hidráulico.....	9
1.3.3.2.9 Caudal turbinado.....	10
1.3.3.2.10 Límites de flujo de potencia en red de transmisión.....	12
1.3.3.2.11 Límites de magnitud de voltajes en barras	12
1.3.3.2.12 Límites de ángulos de voltaje en las barras.....	12
1.3.3.3 Herramientas de Software.....	13
1.3.3.3.1 Python.....	13
1.3.3.3.2 Pyomo.....	13
1.3.3.3.3 Ipopt.....	14
2 METODOLOGÍA.....	15

2.1 Descripción de la herramienta computacional.....	15
3 RESULTADOS	41
3.1 Casos de estudio	41
3.1.1 Aplicación al Sistema IEEE de 14 barras	41
3.1.2 Aplicación al Sistema Nacional Interconectado (SNI) ecuatoriano	58
4 CONCLUSIONES Y RECOMENDACIONES	70
4.1 Conclusiones	70
4.2 Recomendaciones	71
5 REFERENCIAS BIBLIOGRÁFICAS.....	72
6 ANEXOS	74

RESUMEN

En el presente trabajo de integración curricular se desarrolla una herramienta computacional, en lenguaje Python, para la solución del problema del Despacho Hidrotérmico de Corto Plazo (DHT-CP) considerando la red eléctrica en sistemas eléctricos de potencia. Para el cálculo de las pérdidas de potencia activa en la red eléctrica se usa flujo de potencia AC. Se incluye la formulación matemática de las restricciones técnicas y no técnicas del problema. En la herramienta computacional se implementan dos métodos para el cálculo de la potencia generada por las centrales hidroeléctricas.

La herramienta computacional se aplica a los sistemas IEEE de 14 barras y Sistema Nacional Interconectado Ecuatoriano (SNI); para el SNI se estudian dos casos de estudio según su hidrología: la temporada seca y la temporada lluviosa.

La herramienta entrega resultados óptimos para la potencia generada por cada central en función del tiempo, donde como objetivo se tiene que minimizar los costos de generación de las centrales térmicas; en el programa se pueden activar o desactivar las diferentes restricciones del problema y los resultados se muestran gráficamente y se reportan en tablas en formato Excel.

PALABRAS CLAVE: Despacho Hidrotérmico de Corto Plazo, flujo de potencia AC, Sistema Nacional Interconectado Ecuatoriano (SNI), sistema IEEE 14 barras, Pyomo, Python.

ABSTRACT

In this curricular integration work a computational tool is developed, in Python language, for the solution of the Short-Term Hydrothermal Dispatch problem (DHT-CP) considering the electrical network in power systems.

For the calculation of active power losses in the electrical network, AC power flow is used. The mathematical formulation of the technical and non-technical constraints of the problem is included. Two methods are implemented in the computational tool for the calculation of the power generated by hydroelectric power plants.

The computational tool is applied to the IEEE 14-bus system and the Ecuadorian National Interconnected System (SNI); for the SNI two study cases are studied according to its hydrology: dry season and rainy season.

The tool provides optimal results for the power generated by each power plant as a function of time, where the objective is to minimize the generation costs of the thermal power plants; in the program the different restrictions of the problem can be activated or deactivated and the results are shown graphically and reported in tables in Excel format.

KEYWORDS: Short term hydrothermal dispatch, AC power flow, Ecuadorian National Interconnected System (SNI), IEEE 14-Bus System, Pyomo, Python.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El Despacho Hidrotérmico de Corto Plazo (*DHT-CP*) es muy importante para la entrega diaria de energía en un sistema eléctrico, se trata de un problema de optimización complejo, de gran dimensión. En el caso de Ecuador SNI se tiene muchas centrales de generación hidroeléctricas y térmicas, además al considerar la red eléctrica se vuelve más complejo ya que se tienen pérdidas de flujo de potencia en los elementos de transmisión. En algunas formulaciones, el DHT-CP debe minimizar los costos de producción y minimizar el impacto ecológico, minimizando el uso las centrales térmicas y privilegiar la producción de energía por las centrales hidroeléctricas, cuyo recurso principal, el agua, para corto plazo no tiene ningún costo y es renovable.

En el presente trabajo se desarrolla una herramienta computacional, en lenguaje Python, para la modelación, parametrización y resolución del DHT-CP. En este modelo se incluyen las pérdidas en la red eléctrica, las restricciones técnicas y operativas del Sistema Eléctrico de Potencia (*SEP*) en un horizonte de tiempo de 24 horas; se tienen como datos de entrada los costos de combustible para centrales térmicas, los caudales hídricos, las demandas de potencia y las características técnicas del SEP, posteriormente, se determinan los costos por hora de la energía eléctrica, así como la producción de cada generador.

Como una aplicación importante se estudia el sistema eléctrico ecuatoriano, que cuenta con centrales hidroeléctricas, las cuales tienen el mayor aporte en la generación, y centrales térmicas. Debido a que el recurso hídrico proviene, en su gran mayoría, de las cuencas hídricas amazónicas se presentan dos escenarios según la abundancia de este recurso: la época seca, donde entran en operación varias centrales térmicas para suplir la demanda, con un incremento del costo de operación operativo, y la época lluviosa, donde se tiene una mayor operación de las centrales hidroeléctricas.

1.1 Objetivo general

Implementar una herramienta computacional, en lenguaje Python, para la solución del Despacho Hidrotérmico de Corto Plazo, considerando la red eléctrica.

1.2 Objetivos específicos

1. Realizar una revisión bibliográfica relacionada a la modelación matemática y diferentes métodos de solución del despacho hidrotérmico de corto plazo, flujo

óptimo de potencia de corriente alterna (OPF-AC) y linealización de la función de producción de energía eléctrica de las centrales hidroeléctricas.

2. Implementar una herramienta computacional para la solución del DHT-CP considerando la red eléctrica mediante flujo óptimo de potencia de corriente alterna, utilizando la librería PYOMO.
3. Implementar una herramienta computacional para la solución del DHT-CP considerando diferentes alternativas para la modelación de la función de potencia eléctrica generada por unidades hidroeléctricas, con el uso de la librería PYOMO.
4. Aplicar la herramienta computacional desarrollada al sistema de prueba IEEE 14 barras y al Sistema Nacional Interconectado Ecuatoriano.

1.3 Marco teórico

1.3.1 Despacho Hidrotérmico de Corto Plazo

El Despacho Hidrotérmico de Corto Plazo (*DHT-CP*) determina el estado de operación y la potencia generada por cada central de generación para suministrar una demanda dada, dentro de un horizonte de tiempo determinado, que para corto plazo usualmente se considera de 24 horas [1].

La mayoría de los sistemas eléctricos tienen una combinación entre centrales hidroeléctricas y centrales térmicas para suministrar su demanda, donde las centrales hidroeléctricas se usan para suministrar la carga base ya que sus costos no son elevados, en comparación a los costos elevados de las centrales térmicas que usan derivados del petróleo. Por otro lado, las centrales térmicas se usan para abastecer la demanda pico, ya que ocurre en periodos de tiempos pequeños, generalmente, cuando se usa más la energía eléctrica en las cargas que abastece la distribuidora eléctrica. La coordinación sistemática de la operación de un sistema con unidades de generación hidroeléctricas y termoeléctricas es más compleja que la programación del despacho de energía de un sistema totalmente térmico. Se deben tomar en cuenta las condiciones operativas como modelos de los embalses, la capacidad de almacenamiento de las centrales, la generación a partir del recurso hidráulico, etc [2].

Para la solución del DHT-CP se tienen que minimizar los costos totales de generación para abastecer la demanda del sistema en un período de tiempo determinado. El costo de la generación de energía eléctrica con el uso de centrales térmicas se determina

generalmente por el costo del combustible y tiempos de funcionamiento del encendido (arranque) y apagado (parada) de los generadores térmicos en un determinado tiempo, cada central térmica tiene diferentes costos según su tecnología y dependiendo que tipo de combustible se use para generar la energía eléctrica [2].

1.3.2 Flujo Óptimo de Potencia

El Flujo Óptimo de Potencia (**OPF**) combina el cálculo de despacho económico con el cálculo de flujos de potencia, de manera que se resuelvan de forma simultánea. Las pérdidas totales que se presentan en el sistema de transmisión forman parte del cálculo de flujo de potencia en corriente alterna y se reflejan en potencia de generación en la barra de referencia, por lo que no es necesario calcular las pérdidas. El despacho económico se restringe para cumplir con los límites técnicos del sistema de transmisión, de los generadores, y, los límites operativos de voltaje en las barras. Como consecuencia, se tiene un despacho de generación que representa el mínimo costo total de generación en [\$/MWh] [1], [3].

1.3.3 Formulación matemática del DHT -CP con OPF -AC

La forma general de un problema de optimización se expresa de la siguiente manera:

$$\text{Sujeto} \quad \min_x \quad f(X, I) \quad (1)$$

$$G(X, I) \leq 0 \quad (2)$$

$$H(X, I) = 0 \quad (3)$$

Donde f es la función objetivo, G y H son el conjunto de restricciones, I es la variable de entrada para el problema de optimización, X es el conjunto de variables de decisión para satisfacer las restricciones G y H y optimizar el valor de f [4].

1.3.3.1 Función Objetivo

El objetivo de la coordinación hidrotérmica es minimizar los costos totales de generación que pueden ser variables y/o fijos, costos del arranque y parada de las centrales térmicas que se asocian a la operación del Sistema Eléctrico de Potencia (SEP). Cada central termoeléctrica se asocia a una curva de costos propia, donde la función objetivo viene dada como [5]:

$$FO = \min \sum_{t=1}^T \sum_{g=1}^G (F_g(P_g^t) + C_g^A * Y_g^t + C_g^P * W_g^t) \quad (4)$$

Donde:

g	Es la g -ésima central térmica; donde $g = 1,2,3, \dots, G$.
t	Es el t -ésimo periodo de análisis; donde $t = 1,2,3, \dots, T$.
$F_g(P_g^t)$	Es la función de costo total de la g -ésima central térmica [\$/ h].
C_g^A	Son los costos fijos de arranque de la g -ésima central térmica [\$/].
C_g^P	Son los costos fijos de parada de la g -ésima central térmica [\$/].
Y_g^t	Es la variable binaria que se asocia al arranque de la g -ésima central térmica; donde 1 (arranca) y 0 (no arranca).
W_g^t	Es la variable binaria que se asocia a la parada de la g -ésima central térmica; donde 1 (parada) y 0 (no está parada).

La función del costo de la central térmica g , generalmente se presenta como una función polinomial (no lineal) o como una función lineal por partes. Se expresa como [5]:

$$F_g(P_g^t) = a_g * U_g^t + b_g * P_g^t + c_g * P_g^{t^2} \quad (5)$$

Donde:

a_g, b_g, c_g	Son los coeficientes del costo de combustible de la g -ésima central térmica.
P_g^t	Es la potencia generada por la g -ésima central térmica en un periodo t [MW].
c_g	Es el coeficiente de costos cuadrática de la g -ésima central térmica [\$/ MW^2].
b_g	Es el coeficiente de costos lineal de la g -ésima central térmica [\$/ MW].
a_g	Es el coeficiente de costos constante de la g -ésima central térmica [\$/].

Además, una función no lineal se puede aproximar mediante una linealización por partes o una linealización simple que permite lograr una reducción del uso de recursos computacionales. Si se calcula la derivada de la función de costos, se obtiene la función lineal de costo marginal de la central térmica g [1].

En la linealización por partes se tienen tres segmentos del generador g que se representan como g_1 , g_2 , y g_3 donde la variable P_g se reemplaza por tres nuevas variables P_{g1} , P_{g2} y P_{g3} donde cada segmento tiene una pendiente asignada s_{g1} , s_{g2} y s_{g3} . Entonces la función de costos se representa como la suma del costo P_g^{min} más la suma del costo lineal [1].

$$F_g(P_g^t) = F_g(P_g^{min}) + s_{g1}P_g + s_{g2}P_g + s_{g3}P_g \quad (6)$$

Mediante la derivada de la función de costos cuadrática se tiene una función lineal de los costos marginales de combustible de la central térmica g como [1]:

$$\frac{\delta F_g(P_g^t)}{\delta P_g^t} = b_g * U_g^t + c_g * P_g^t \quad (7)$$

Donde:

U_g^t Es la variable binaria que se asocia al estado de operación de la g -ésima central térmica; donde 1 (está en servicio) y 0 (está fuera de servicio).

1.3.3.2 Restricciones

Se tiene un conjunto de restricciones: i) De igualdad, que se relacionan con el balance de la potencia en cada una de las barras del SEP y el balance hídrico del embalse de las centrales hidroeléctricas, y, ii) De desigualdad, que se relacionan con los límites técnicos del sistema de transmisión, de los generadores, límites operativos y de seguridad, etc [6].

1.3.3.2.1 Balance de potencia activa y reactiva

Considerando el flujo de potencia AC, la restricción de balance de potencia activa se expresa como [5]:

$$\begin{aligned} \sum_{g=1}^G (P_g^t) + \sum_{h=1}^H (P_h^t) - \sum_{d=1}^D (P_{Dem}^t) &= \sum_{n,m=1}^N (P_{nm}) \\ &= \text{Real} \left\{ V_n \left(\sum_{m=1}^N (Y_{nm} * V_m) \right)^* \right\}; \quad \forall t \end{aligned} \quad (8)$$

Donde:

P_{Dem}^t Es la potencia activa de demanda en la hora t [MW].
 P_g^t Es la potencia activa generada por la g -ésima central térmica en un periodo t [MW].
 P_h^t Es la potencia activa generada por la h -ésima central hidroeléctrica en un periodo t [MW].
 P_{nm} Es la potencia activa transmitida desde la barra n hasta la barra m [MW].
 V_n Es la magnitud de voltaje en la barra n [V].
 V_m Es la magnitud de voltaje en la barra m [V].
 Y_{nm} Es la admitancia entre la barra n y la barra m [Ω^{-1}].
 N Número de barras.

Considerando el flujo de potencia AC, la restricción de balance de potencia reactiva se expresa como [5]:

$$\sum_{g=1}^G (Q_g^t) + \sum_{h=1}^H (Q_h^t) - \sum_{d=1}^D (Q_{Dem}^t) = \sum_{n,m=1}^N (Q_{nm}) \quad (9)$$

$$= \text{Imag} \left\{ V_n \left(\sum_{m=1}^N (Y_{nm} * V_m) \right)^* \right\}; \quad \forall t$$

Donde:

Q_{Dem}^t	Es la potencia reactiva de demanda en la hora t [MVar].
Q_g^t	Es la potencia reactiva generada por la g -ésima central térmica en un periodo t [MVar].
Q_h^t	Es la potencia reactiva generada por la h -ésima central hidroeléctrica en un periodo t [MVar].
Q_{nm}	Es la potencia reactiva transmitida desde la barra n hasta la barra m [MVar].

Los elementos de compensación reactiva se toman en cuenta en la construcción de la admitancia de barra ya que para su cálculo se suma la admitancia shunt que es $Y/2$.

1.3.3.2.2 Límites de potencia activa generada

La potencia activa generada por cada unidad no debe sobrepasar los límites máximo y mínimo definidos por el fabricante, ya que así se garantiza la operación segura del generador [5].

$$P_{gmin} * U_g^t \leq P_g^t \leq P_{gmax} * U_g^t; \quad \forall t \quad (10)$$

$$P_{hmin} \leq P_h^t \leq P_{hmax}; \quad \forall t \quad (11)$$

Donde:

P_{gmin}	Es la potencia activa mínima de la g -ésima central térmica [MW].
P_{gmax}	Es la potencia activa máxima de la g -ésima central térmica [MW].
P_{hmin}	Es la potencia activa mínima de la h -ésima central hidroeléctrica [MW].
P_{hmax}	Es la potencia activa máxima de la h -ésima central hidroeléctrica [MW].

1.3.3.2.3 Límites de potencia reactiva generada

La potencia reactiva generada por cada unidad no debe sobrepasar sus límites técnicos máximo y mínimo, ya que así se garantiza la operación segura del generador. Para facilitar los cálculos las restricciones se trabajan en p.u. [5].

$$Q_{gmin} * U_g^t \leq Q_g^t \leq Q_{gmax} * U_g^t; \quad \forall t \quad (12)$$

$$Q_{hmin} \leq Q_h^t \leq Q_{hmax}; \quad \forall t \quad (13)$$

Donde:

Q_{gmin}	Es la potencia reactiva mínima de la g-ésima central térmica [MVar].
Q_{gmax}	Es la potencia reactiva máxima de la g-ésima central térmica [MVar].
Q_{hmin}	Es la potencia reactiva mínima de la h-ésima central hidroeléctrica [MVar].
Q_{hmax}	Es la potencia reactiva máxima de la h-ésima central hidroeléctrica [MVar].

1.3.3.2.4 Restricciones de rampa de arranque y subida

Las rampas de arranque y subida de potencia (toma de carga) se relacionan con el límite máximo de potencia que se puede añadir desde un periodo t a un periodo $t + 1$, donde la rampa de arranque indica que la central se encuentra desacoplada en un periodo t [5].

$$P_g^t - P_g^0 \leq R_g^S(U_g^0) + R_g^A(Y_g^t); \quad t = 1 \quad (14)$$

$$P_g^t - P_g^{t-1} \leq R_g^S(U_g^{t-1}) + R_g^A(Y_g^t); \quad t = 2, \dots, T \quad (15)$$

$$P_h^t - P_h^0 \leq R_h^S; \quad t = 1 \quad (16)$$

$$P_h^t - P_h^{t-1} \leq R_h^S; \quad t = 2, \dots, T \quad (17)$$

Donde:

R_g^S	Es la rampa de subida de la g-ésima central térmica [MW].
R_g^A	Es la rampa de arranque de la g-ésima central térmica [MW].
R_h^S	Es la rampa de subida de la h-ésima central hidroeléctrica [MW].
<i>Superíndice 1</i>	Arranca.
<i>Superíndice 0</i>	No arranca.

El resultado es que la potencia generada por un generador g en un periodo t , menos la potencia generada en un periodo anterior $t - 1$, debe ser igual o menor a la suma de la potencia máxima de subida si el generador está en línea donde $U_g^{t-1} = 1$ en el periodo $t - 1$, más la potencia máxima de arranque si el generador arranca donde $Y_g^t = 1$ en el periodo t [5].

1.3.3.2.5 Restricciones de rampa de parada y bajada

Las rampas de parada y bajada de potencia se relacionan con el límite máximo de potencia que se puede disminuir desde un periodo t a un periodo $t - 1$ o detener la generación en un periodo t para la variable binaria U_g^t [5].

$$P_g^0 - P_g^t \leq R_g^B(U_g^t) + R_g^P(W_g^t); \quad t = 1 \quad (18)$$

$$P_g^{t-1} - P_g^t \leq R_g^B(U_g^t) + R_g^P(W_g^t); \quad t = 2, \dots, T \quad (19)$$

$$P_h^0 - P_h^t \leq R_h^B; \quad t = 1 \quad (20)$$

$$P_h^{t-1} - P_h^t \leq R_h^B; \quad t = 2, \dots, T \quad (21)$$

Donde:

R_g^B	Es la rampa de bajada de la g-ésima central térmica [MW].
R_g^P	Es la rampa de parada de la g-ésima central térmica [MW].
R_h^B	Es la rampa de bajada de la h-ésima central hidroeléctrica [MW].
Superíndice 1	Está parada.
Superíndice 0	No está parada.

El resultado es que la potencia generada por un generador g en un periodo $t - 1$, menos la potencia generada en un periodo anterior t , debe ser igual o menor a la suma de la potencia máxima de bajada si el generador está en línea donde $U_g^t = 1$ en el periodo t , más la potencia máxima de parada si el generador se encuentra parado donde $W_g^t = 1$ en el periodo t [5].

1.3.3.2.6 Lógica binaria de operación

Este conjunto de restricciones representa el estado de la unidad generadora como acoplamiento, arranque y parada. [5]

$$U_g^t - U_g^0 = Y_g^t - W_g^t; \quad t = 1 \quad (22)$$

$$U_g^t - U_g^{t-1} = Y_g^t - W_g^t; \quad t = 2, \dots, T \quad (23)$$

$$Y_g^t + W_g^t \leq 1 \quad (24)$$

Donde:

U_g^t Es la variable binaria que se asocia al estado de operación de la g-ésima central térmica; donde 1 (está en servicio) y 0 (está fuera de servicio).

Y_g^t Es la variable binaria que se asocia al arranque de la g-ésima central térmica; donde 1 (arranca) y 0 (no arranca).

W_g^t Es la variable binaria que se asocia a la parada de la g-ésima central térmica; donde 1 (parada) y 0 (no está parada).

Se tiene cuatro posibles estados operativos de la central:

- Si se tiene en t que esta acoplada y en $t - 1$ está desacoplada, entonces la central arranca en t .
- Si se tiene en t que esta acoplada y en $t - 1$ está acoplada, entonces la central ya estaba operando en $t - 1$ y no puede arrancar ni realizar la parada en un tiempo t .
- Si se tiene en t que esta desacoplada y en $t - 1$ está acoplada, entonces la central se paró en t .
- Si se tiene en t que esta desacoplada y en $t - 1$ está desacoplada, entonces la central no estaba operando en $t - 1$ y no puede arrancar ni realizar la parada en un tiempo t [5].

1.3.3.2.7 Volumen máximo y mínimo de embalses

Dependiendo de la planificación de operación se define una cantidad de energía a suministrar, donde el embalse debe mantener un mínimo de agua que garantice la continuidad de operación de la central. Además, se tiene un límite máximo del embalse, que evite el desperdicio del recurso hídrico [5].

$$V_{hmin} \leq V_h^t \leq V_{hmax}; \quad \forall t \quad (25)$$

Donde:

- V_{hmin} Es el volumen mínimo del embalse de la h-ésima central hidroeléctrica [Hm^3].
- V_{hmax} Es el volumen máximo del embalse de la h-ésima central hidroeléctrica [Hm^3].
- V_h^t Es el volumen del agua del embalse de la h-ésima central hidroeléctrica en un tiempo t [Hm^3].

1.3.3.2.8 Balance hidráulico

Embalse independiente

Para asegurar un uso eficaz del embalse se necesita controlar el nivel del agua en cada periodo t . La expresión matemática que relaciona el volumen de agua en el embalse con la topología de la red hidráulica, cuando se tiene embalses independientes está dada por [5]:

$$V_h^t = V_h^{t-1} + 3600(r_h^t) - q_h^t - S_h^t \quad (26)$$

Donde:

r_h^t	Es el influjo natural del embalse de la h-ésima central hidroeléctrica en un periodo t [m^3/s].
q_h^t	Es el caudal turbinado de la h-ésima central hidroeléctrica en un periodo t [m^3/h].
S_h^t	Son los vertimientos del embalse de la h-ésima central hidroeléctrica en un periodo t [m^3/h].

Embalses dependientes en cascada

Cuando la topología de la red hidráulica tiene centrales hidroeléctricas en cascada o con acoplamiento hídrico, es decir varios embalses en un mismo afluyente, se debe incluir un tiempo de desfase (t_{DT}) entre las centrales, ya que el caudal de una central aguas arriba se demora un tiempo en llegar a la central aguas abajo [5].

$$V_h^t = V_h^{t-1} + 3600(r_h^t) - q_h^t - S_h^t + \sum (q_{ha}^{t-t_{DT}} + S_{ha}^{t-t_{DT}}); \quad \forall t \quad (27)$$

Donde:

q_{ha}^t	Es el caudal turbinado de la central hidroeléctrica aguas arriba en un periodo t [m^3/h].
S_{ha}^t	Son los vertimientos del embalse de la central hidroeléctrica aguas arriba en un periodo t [m^3/h].
t_{DT}	Es el desfase temporal entre la unidad aguas arriba y la h-ésima central hidroeléctrica [m^3/h].

1.3.3.2.9 Caudal turbinado

La determinación del caudal turbinado considera varios parámetros: tipo de turbina, canales de flujo o tipo de tuberías, coeficientes de pérdidas por fricción, volumen y forma del embalse, etc. Pero, cuando se va a realizar un análisis para corto plazo estos parámetros tienen poca influencia. Si adicionalmente, la altura del embalse se mantiene casi constante durante el periodo de despacho, el caudal turbinado se relaciona con la potencia generada por la central hidroeléctrica mediante el coeficiente de producción de cada central. La expresión matemática para cálculo del caudal se expresa como [7]:

$$q_h^t = \frac{P_h^t}{\rho_h}; \quad \forall t \quad (28)$$

Donde:

ρ_h Es el coeficiente de producción de la h-ésima central hidroeléctrica [MWh/m³].

En general, la potencia eléctrica generada por una central hidroeléctrica se determina por sus características técnicas como la altura de caída del agua, el caudal del agua, la eficiencia de la turbina y se expresa mediante la ecuación (29) [8]:

$$q = \frac{P_h^t}{g * h * n_t * \rho} \quad (29)$$

Donde:

P_h^t Es la potencia generada por la h-ésima unidad hidroeléctrica en un periodo t [MW].
 q Es el caudal turbinado [m³/h].
 g Es la aceleración gravitacional [m/h²].
 h Es el salto hídrico [m].
 n_t Es la eficiencia de la turbina.
 ρ Es la densidad del agua [kg/m³].

En la referencia [9] se propone otro método para el cálculo del caudal turbinado, donde se usa una ecuación cuadrática que toma como dato de entrada la potencia eléctrica generada, y mediante los coeficientes de descarga de la central, que dependen de las características constructivas de la tubería de descarga, la altura de la caída del agua, los materiales de construcción y el diámetro de la tubería, se calcula el caudal turbinado mediante [9]:

$$q = x * P_h^2 + y * P_h + z \quad (30)$$

Donde:

P_h Es la potencia generada por la h-ésima unidad hidroeléctrica [MW].
 q Es el caudal turbinado [m³/h].
 x Es el coeficiente de descarga $x \left[\frac{m^3}{MW^2 * h} \right]$.
 y Es el coeficiente de descarga $y \left[\frac{m^3}{MW * h} \right]$.
 z Es el coeficiente de descarga $z \left[\frac{m^3}{h} \right]$.

Los coeficientes de descarga vienen dados en tablas nemotécnicas y son propios de cada central hidroeléctrica [9].

1.3.3.2.10 Límites de flujo de potencia en red de transmisión

$$-P_{nm}^{max} \leq P_{nm} = V_n^2 \cdot G_{nm} - V_n \cdot V_m (G_{nm} \cos \theta_{nm} + B_{nm} \sin \theta_{nm}) \leq P_{nm}^{max}; \forall n, m = 1, 2, \dots, N \quad (31)$$

Donde:

P_{nm}^{max}	Es la potencia activa máxima que se transmite desde la barra n hasta la barra m [MW].
P_{nm}	Es la potencia activa transmitida desde la barra n hasta la barra m [MVA].
G_{nm}	Es la conductancia desde la barra n hasta la barra m [Ω^{-1}].
B_{nm}	Es la susceptancia desde la barra n hasta la barra m [Ω^{-1}].
V_n	Es la magnitud de voltaje en la barra n [V].
V_m	Es la magnitud de voltaje en la barra m [V].
θ_{nm}	Ángulo de voltaje desde la barra n hasta la barra m [rad].

$$S_{nm} = P_{nm} + jQ_{nm} = \sqrt{P_{nm}^2 + Q_{nm}^2} \leq S_{nm}^{max}; \forall n, m = 1, 2, \dots, N \quad (32)$$

$$-Q_{nm}^{max} \leq Q_{nm} = -V_n^2 \cdot B_{nm} - V_n \cdot V_m (G_{nm} \sin \theta_{nm} - B_{nm} \cos \theta_{nm}) \leq Q_{nm}^{max}; \forall n, m = 1, 2, \dots, N \quad (33)$$

Donde:

S_{nm}^{max}	Es la potencia aparente máxima que se transmite desde la barra n hasta la barra m [MVA].
S_{nm}	Es la potencia aparente transmitida desde la barra n hasta la barra m [MVA].

1.3.3.2.11 Límites de magnitud de voltajes en barras

$$V_n^{min} \leq V_n \leq V_n^{max}; \forall n = 1, 2, 3, \dots, N \quad (34)$$

Donde:

V_n^{min}	Es la magnitud de voltaje mínimo en la barra n .
V_n^{max}	Es la magnitud de voltaje máximo en la barra n . [10]

1.3.3.2.12 Límites de ángulos de voltaje en las barras

$$-\theta_n^{min} \leq \theta_n \leq \theta_n^{max}; \forall n = 1,2,3, \dots, N \quad (35)$$

$$-180 \leq \theta \leq 180 \quad (36)$$

Donde:

θ	Es el ángulo de voltaje en la barra.
θ_n^{min}	Es el ángulo de voltaje mínimo en la barra n .
θ_n^{max}	Es el ángulo de voltaje máximo en la barra n .

1.3.3.3 Herramientas de Software

1.3.3.3.1 Python

Python es un lenguaje de programación avanzado, de alto nivel, interpretado, gratuito, multiplataforma, dinámico, el cual da la posibilidad de realizar un código para formular un problema de optimización. Cuenta con gran variedad de librerías libres y gratuitas disponibles, además de diferentes módulos para realizar el procesamiento de datos para una convergencia rápida del problema de optimización. Se tiene un entorno virtual de trabajo de Python muy amigable con el usuario para que la codificación sea eficaz y en un lenguaje muy sencillo. Una gran ventaja es que es muy versátil ya que usa bastantes librerías de terceros además de la gran rapidez en la lectura de la información donde se tienen una gran cantidad de formatos para el almacenamiento de datos [11].

Para realizar la programación del despacho hidrotérmico a corto plazo con el lenguaje Python se usa varias librerías propias de este lenguaje, se incluye módulos de formulación matemática y de optimización que son descargados previamente en el entorno de trabajo de Python [11].

1.3.3.3.2 Pyomo

Pyomo es una librería de optimización orientada a objetos de acceso libre y gratuita. Estos objetos de modelado de Pyomo para la modelación del problema se encuentran internamente en el lenguaje Python y su librería se descarga previamente en el entorno de trabajo. Pyomo es usado para la formulación, análisis y resolución de modelos matemáticos para problemas de optimización avanzados donde se puede visualizar los datos y resultados. Este módulo admite funciones algebraicas para resolver diferentes problemas de programación matemática con enteros mixtos [12].

Se tiene varias ventajas de Pyomo como:

- Soporta el modelado de problemas de optimización de manera estructurada con lenguajes de modelación AML como AMPL, AIMMS, GAMS.

- Sirve para realizar una predicción sobre el estado de un sistema.

-Usar diferentes solucionadores comerciales de código abierto para encontrar la solución óptima del problema.

-Tiene una gran variedad de librerías de apoyo de terceros.

-Cuenta con una gran documentación de apoyo [12].

1.3.3.3 Ipopt

Ipopt es un solver o solucionador de problemas de optimización no lineal, de gran tamaño, y de código abierto. Para la optimización se usa el método de filtro de búsqueda de línea de punto interior para encontrar una solución al problema matemático [13].

2 METODOLOGÍA

La herramienta computacional se desarrolla en el ambiente de desarrollo Spyder, el cual cuenta con funciones para edición y depuración del código, librerías propias y un entorno de código numérico.

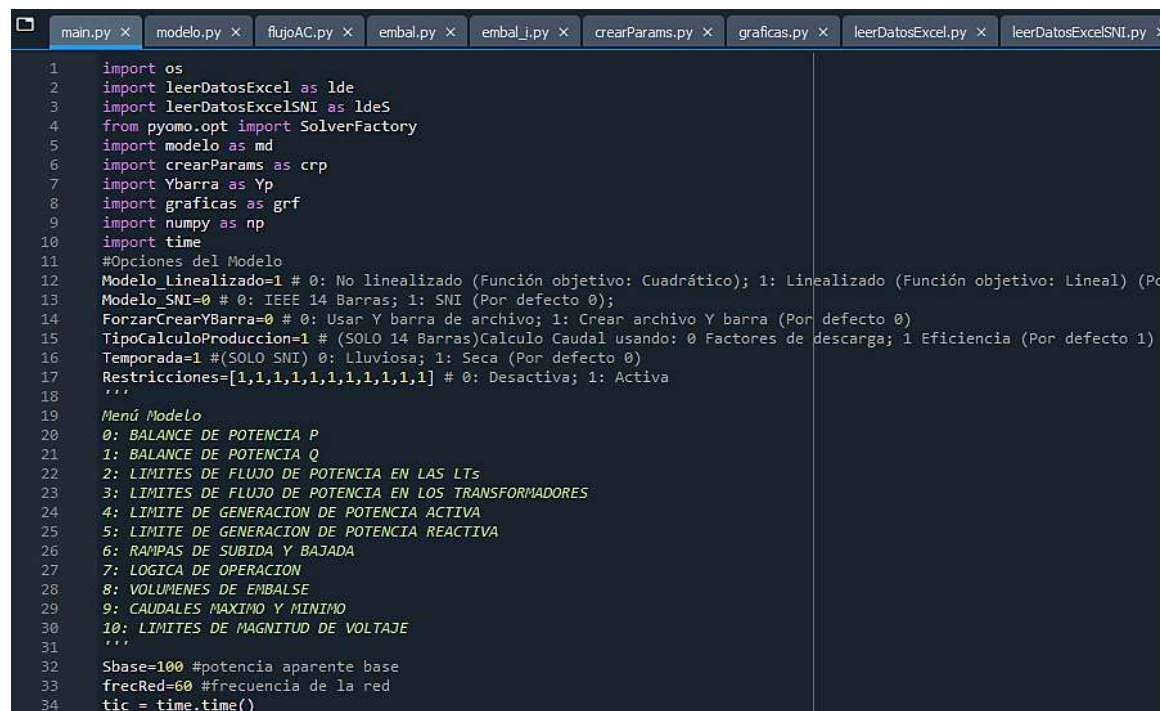
2.1 Descripción de la herramienta computacional

A continuación, se presenta una descripción del código desarrollado y la implementación de la formulación matemática del DHT-CP. Se han desarrollado diferentes “scripts” o funciones de Python para la solución del problema.

Inicialización

Al inicio de la función “modelo.py” se crean los modelos en Pyomo que se pueden definir como una inicialización de un conjunto a la clase “Set”, los parámetros con la clase “Param”, los contadores con la clase “RangeSet”, las variables con la clase “Var”.

En las variables de control se tiene la potencia activa y reactiva generada de las centrales térmicas o hidroeléctricas; y en las variables de estado se tiene los ángulos y magnitud de voltaje en las barras, las variables binarias, y los flujos de potencia de las líneas de transmisión y transformadores.



```
1 import os
2 import leerDatosExcel as lde
3 import leerDatosExcelSNI as ldeS
4 from pyomo.opt import SolverFactory
5 import modelo as md
6 import crearParams as crp
7 import Ybarra as Yp
8 import graficas as grf
9 import numpy as np
10 import time
11 #Opciones del Modelo
12 Modelo_linealizado=1 # 0: No linealizado (Función objetivo: Cuadrático); 1: Linealizado (Función objetivo: Lineal) (Por defecto 1)
13 Modelo_SNI=0 # 0: IEEE 14 Barras; 1: SNI (Por defecto 0);
14 ForzarCrearYbarra=0 # 0: Usar Y barra de archivo; 1: Crear archivo Y barra (Por defecto 0)
15 TipoCalculoProduccion=1 # (SOLO 14 Barras)Calculo Caudal usando: 0 Factores de descarga; 1 Eficiencia (Por defecto 1)
16 Temporada=1 #(SOLO SNI) 0: Lluviosa; 1: Seca (Por defecto 0)
17 Restricciones=[1,1,1,1,1,1,1,1,1,1] # 0: Desactiva; 1: Activa
18 '''
19 Menú Modelo
20 0: BALANCE DE POTENCIA P
21 1: BALANCE DE POTENCIA Q
22 2: LIMITES DE FLUJO DE POTENCIA EN LAS LTs
23 3: LIMITES DE FLUJO DE POTENCIA EN LOS TRANSFORMADORES
24 4: LIMITE DE GENERACION DE POTENCIA ACTIVA
25 5: LIMITE DE GENERACION DE POTENCIA REACTIVA
26 6: RAMPAS DE SUBIDA Y BAJADA
27 7: LOGICA DE OPERACION
28 8: VOLUMENES DE EMBALSE
29 9: CAUDALES MAXIMO Y MINIMO
30 10: LIMITES DE MAGNITUD DE VOLTAJE
31 '''
32 Sbase=100 #potencia aparente base
33 frecRed=60 #frecuencia de la red
34 tic = time.time()
```

Figura 2.1.1. Función “main.py”

Como se muestra en la Figura 2.1.1, se encuentra la función “main.py” que se encarga de dar inicio al programa y llama a las diferentes funciones que se crearon para resolver el problema de optimización de despacho de energía eléctrica, las demás funciones deben estar en la misma carpeta que se encuentra guardada la función “main.py”.

Además, se crea un menú de las restricciones del problema de optimización en la línea 17 como una función booleana llamada “Restricciones” donde en el caso que se coloque 0 se desactiva la restricción y cuando se coloca 1 se activa la restricción, además desde la línea 18 hasta la línea 31 está la descripción de las restricciones en el script “main.py”.

```

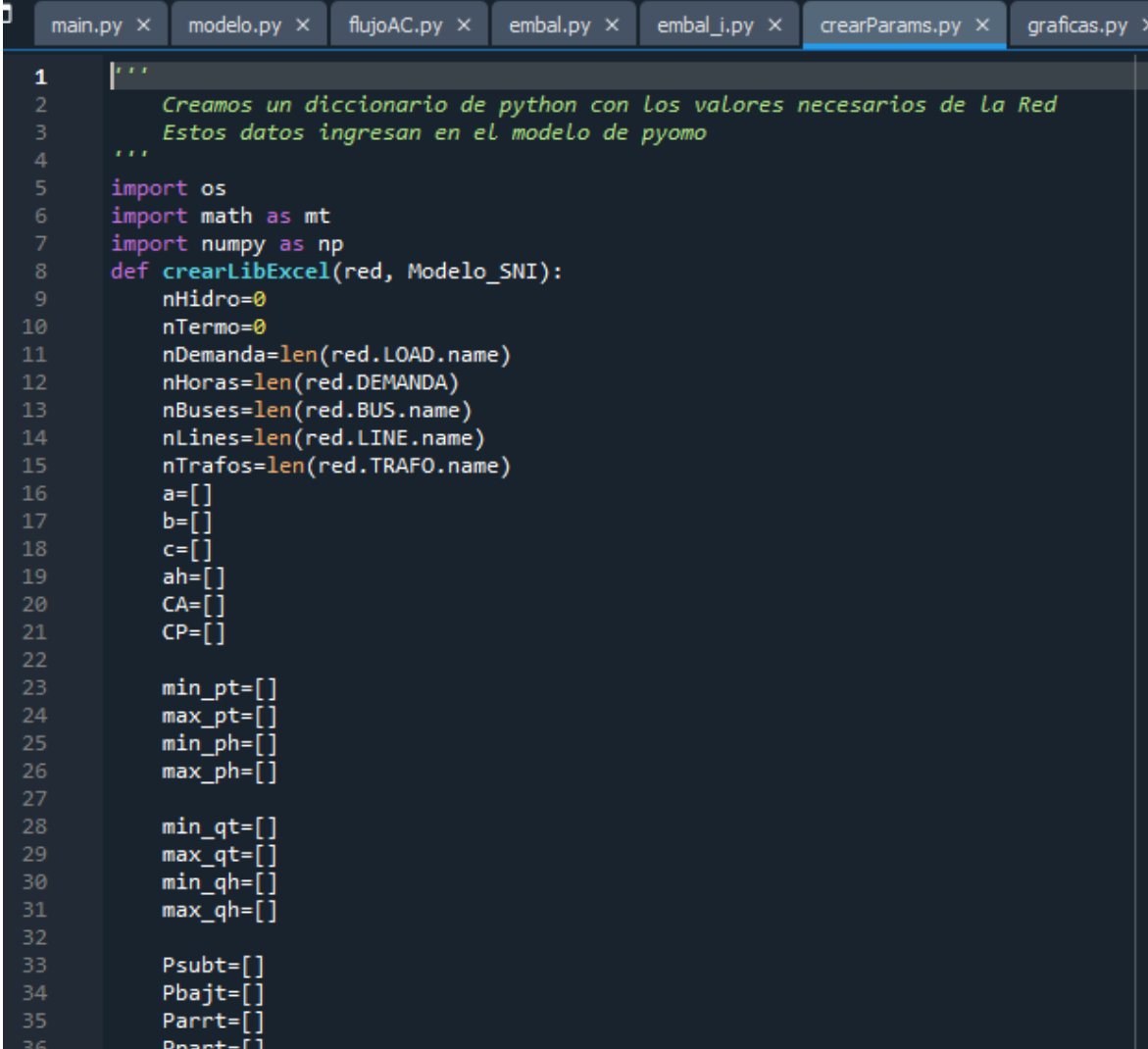
59     if Temporada==0:
60         # Temporada Lluviosa
61         archivo_Ybarra_1='Excel/ISNI_2_V1_08TempLluviosa_Ybarra1_Nuevo'
62         archivo_excel='Excel/SNI_2_V1_08TempLluviosa.xlsx'
63     else:
64         # Temporada Seca
65         archivo_Ybarra_1='Excel/ISNI_2_V1_08TempSeca_Ybarra1_Nuevo'
66         archivo_excel='Excel/SNI_2_V1_08TempSeca.xlsx'
67         # Cargamos los datos de los Excel en la variable red
68         red=ldeS.leerexcel(archivo_excel)
69     print('Terminado')
70     print('Obteniendo Ybarra:')
71     if ForzarCrearYbarra==1 or os.path.exists(archivo_Ybarra_1 + '.npy')==False :
72         # Comprobamos si en las opciones esta activa la opcion de calcular la Y de
73         # Barra o si nos falta el archivo de la misma y calculamos la Y de Barra
74         Ybarra1=Yp.Ybarra(red, Sbase, frecRed, archivo_Ybarra_1)
75     else:
76         # Si existe el archivo lo cargamos en la variable correspondiente
77         Ybarra1=np.load(archivo_Ybarra_1 + '.npy')
78     print('Terminado')
79     #Obtengo el Modelo
80     print('Obteniendo modelo:')
81     # Creamos el Modelo de Pyomo
82     model=md.crearModelo(red, Modelo_Linealizado,Sbase,frecRed, Modelo_SNI, Restricciones,Ybarra1,TipoCalculoProduccion)
83     print('Terminado')
84     print('Obteniendo Datos:')
85     # Obtenemos los datos necesarios de la Red y lo almacenamos en un diccionario
86     data=crp.crearLibExcel(red, Modelo_SNI)
87     print('Terminado')
88     print('Creando modelo:')
89     # insertamos los datos de el diccionario en el modelo de Pyomo
90     i=model.create_instance(data)
91     print('Terminado')
92     # Elegimos el solver que sera usado
93     solver=SolverFactory('ipopt')
94     print('Resolviendo modelo:')
95     # Resolvemos el Modelo
96     results = solver.solve(i)
97     print(results) # Imprime el resultado del modelo
98     # Creamos las Graficas y exportamos los resultados a Excel
99     r=grf.graficarPotencias(i, red)
100    r=grf.graficarVariablesBinariasU(i, red)

```

Figura 2.1.2. Función “main.py”

Como se muestra en la Figura 2.1.2, se encuentra la función “main.py” donde se tiene la potencia aparente base en 100 [MVA], la frecuencia de la red en 60 [Hz], la impresión del menú del modelo del problema de optimización, se obtiene los datos del sistema de los archivos Excel para los casos de aplicación del sistema IEEE de 14 barras y el SNI, se carga el archivo “.npy” o se obtiene el archivo de la Y de barra, se obtiene y se crea el

modelo del problema en Pyomo, se elige el solucionador “IPOPT”, se resuelve y se imprime los resultados del problema de optimización en donde se visualiza si se llegó a una solución óptima y el tiempo de ejecución del programa que se realiza con el comando “toc”. Finalmente, se grafica los resultados en función del tiempo de algunas variables y se exporta los resultados de estas variables en archivos Excel.



```
1  '''
2      Creamos un diccionario de python con los valores necesarios de la Red
3      Estos datos ingresan en el modelo de pyomo
4  '''
5  import os
6  import math as mt
7  import numpy as np
8  def crearLibExcel(red, Modelo_SNI):
9      nHidro=0
10     nTermo=0
11     nDemanda=len(red.LOAD.name)
12     nHoras=len(red.DEMANDA)
13     nBuses=len(red.BUS.name)
14     nLines=len(red.LINE.name)
15     nTrafos=len(red.TRAFO.name)
16     a=[]
17     b=[]
18     c=[]
19     ah=[]
20     CA=[]
21     CP=[]
22
23     min_pt=[]
24     max_pt=[]
25     min_ph=[]
26     max_ph=[]
27
28     min_qt=[]
29     max_qt=[]
30     min_qh=[]
31     max_qh=[]
32
33     Psubt=[]
34     Pbajt=[]
35     Parrt=[]
36     Ppart=[]
```

Figura 2.1.3. Función “crearParams.py”

Como se muestra en la Figura 2.1.3, se encuentra la función “crearParams.py” que sirve para crear los parámetros en el formato de Pyomo, crear un diccionario o lista de Python de las variables de los datos de la red y estos datos ingresan en el modelo de Pyomo. Se crea diccionarios para los parámetros que son los datos de entrada del problema como los costos de las centrales térmicas, las potencias activas y reactivas máximas, las potencias activas y reactivas mínimas, demanda, influjos, entre otros.

```
420     'thetamin':thetamin_dict,
421     'Voltmax':Voltmax_dict,
422     'Voltmin':Voltmin_dict,
423
424     'Voltimax':Voltimax_dict,
425     'Voltimin':Voltimin_dict,
426
427     'Pijmax':Pijmax_dict,
428     'PijTmax':PijTmax_dict,
429     'Vini': Vini_dict,
430     'Vmax': Vmax_dict,
431     'Vmin': Vmin_dict,
432     'qmin': qmin_dict,
433     'qmax': qmax_dict,
434     'rh': rh_dict,
435     'eh': eh_dict,
436
437     'x':x_dict,
438     'y':y_dict,
439     'z':z_dict,
440
441     'upstream': upstream_dict,
442     'embalse': embalse_dict,
443     'altura': altura_dict,
444     'largo': largo_dict,
445     'volument': volument_dict,
446     'unidades': unidades_dict,
447     'dt': dt_dict,
448     'dtq': dtq_dict,
449     'rht': rht_dict,
450
451     'lineas_posicion': lineas_posicion_dict,
452     'trafos_posicion': trafos_posicion_dict,
453 }}
454 return data
```

Figura 2.1.4. Función “crearParams.py”

Como se muestra en la Figura 2.1.4, siguiendo la función “crearParams.py” se guarda los diccionarios de los diferentes parámetros del modelo en una variable “data”, al final en la línea 454 se devuelve la variable “data” con el cual se pasa a la función “modelo” por lo que ya se tiene estructurado los parámetros en el formato de Pyomo.

Ingreso de datos

```
maniz.py x  modelo.py x  fljooAC.py x  embal.py x  embal_1.py x  creaParams.py x  grafCal.py x  leerDatosExcel.py x  leerDatosExcelSNI.py x
16     self.INFLUJO = INFLUJO
17 <class BUS:
18     def __init__(self, number, name, vn_kv, etype, zone, in_service, min_vm_pu, max_vm_pu, vn_i, Tipo):
19         self.number = number
20         self.name = name
21         self.vn_kv = vn_kv
22         self.etype = etype
23         self.zone = zone
24         self.in_service = in_service
25         self.min_vm_pu = min_vm_pu
26         self.max_vm_pu = max_vm_pu
27         self.vn_i = vn_i
28         self.Tipo = Tipo
29 <class LOAD:
30     def __init__(self, number, name, bus, p_mw, q_mvar, const_z_percent, const_i_percent, sn_mva, scaling, in_service, etype,
31         self.number = number
32         self.name = name
33         self.bus = bus
34         self.p_mw = p_mw
35         self.q_mvar = q_mvar
36         self.const_z_percent = const_z_percent
37         self.const_i_percent = const_i_percent
38         self.sn_mva = sn_mva
39         self.scaling = scaling
40         self.in_service = in_service
41         self.etype = etype
42         self.controllable = controllable
43         self.cos_phi = cos_phi
44         self.mode = mode
45 <class GEN:
46     def __init__(self, number, name, bus, p_mw, vn_pu, sn_mva, min_q_mvar, max_q_mvar, scaling, slack, in_service, etype, con
47         self.number = number
48         self.name = name
49         self.bus = bus
```

Figura 2.1.5. Función “leerDatosExcel.py”

Como se muestra en la Figura 2.1.5, se encuentra la función “leerDatosExcel.py” que sirve para leer los datos del sistema IEEE de 14 barras desde un archivo Excel. Esta función realiza la lectura cada una de las pestañas del archivo Excel y realiza la tabulación de los datos para los diferentes datos de cada uno de los elementos del sistema. Se define como clase “class” a cada una de las pestañas del Excel y se extrae los datos de las variables del sistema de cada una de las columnas de las diferentes pestañas con el comando “self” para guardarlo en su correspondiente clase. Finalmente, en la línea 193 se devuelve la variable “datos_sistema” donde se guardan todos los datos del sistema como los datos de las barras, las cargas, los generadores, las líneas de transmisión, los transformadores, demandas, entre otros.

```

154 class GTERM:
155     def __init__(self, number, Referencia, PotenciaS, PotenciaB, PotenciaA, PotenciaP, CostoA, CostoP):
156         self.number = number
157         self.Referencia = Referencia
158         self.PotenciaS = PotenciaS
159         self.PotenciaB = PotenciaB
160         self.PotenciaA = PotenciaA
161         self.PotenciaP = PotenciaP
162         self.CostoA = CostoA
163         self.CostoP = CostoP
164     def leerexcel(nombre):
165         Aux = pd.read_excel(nombre, sheet_name='BUS', skiprows=1, header=None)
166         bbus=BUS(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7],Aux[10],Aux[11])
167         Aux = pd.read_excel(nombre, sheet_name='LOAD', skiprows=1, header=None)
168         lload=LOAD(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7],Aux[8],Aux[9],Aux[10],Aux[11],Aux[12])
169         Aux = pd.read_excel(nombre, sheet_name='GEN', skiprows=1, header=None)
170         ggen=GEN(Aux[0],Aux[1],Aux[2],Aux[3],Aux[10],Aux[11],Aux[13],Aux[14],Aux[4],Aux[6],Aux[7])
171         Aux = pd.read_excel(nombre, sheet_name='LTWE', skiprows=1, header=None)
172         lline=LIME(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[8],Aux[7],Aux[8],Aux[9],Aux[10],Aux[11],Aux[12],Aux[13],Aux[14],Au
173         Aux = pd.read_excel(nombre, sheet_name='TRAFQ2', skiprows=1, header=None)
174         ttrafo=TRAFQ(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7],Aux[8],Aux[9],Aux[10],Aux[11],Aux[12],Aux[13],Aux[14],
175         Aux = pd.read_excel(nombre, sheet_name='SHUNT', skiprows=1, header=None)
176         sshunt=SHUNT(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7],Aux[8])
177         Aux = pd.read_excel(nombre, sheet_name='COST', skiprows=1, header=None)
178         ccost=COST(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7],Aux[8])
179         Aux = pd.read_excel(nombre, sheet_name='GHIDRO', skiprows=1, header=None)
180         gghidro=GHIDRO(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7],Aux[8],Aux[9],Aux[10],Aux[11],Aux[12],Aux[13],Aux[14]
181         Aux = pd.read_excel(nombre, sheet_name='GTerm', skiprows=1, header=None)
182         ggterm=GTERM(Aux[0],Aux[1],Aux[2],Aux[3],Aux[4],Aux[5],Aux[6],Aux[7])
183         Dem = pd.read_excel(nombre, sheet_name='Demanda', skiprows=1, header=None)
184         DemQ = pd.read_excel(nombre, sheet_name='DEMANDAQ', skiprows=1, header=None)
185         infl = pd.read_excel(nombre, sheet_name='Influjos', skiprows=1, header=None)
186         datos_sistema=Datos(bbus, lload, ggen, lline, ttrafo, sshunt, ccost, gghidro, ggterm, Dem, DemQ, infl)
187         return datos_sistema

```

Figura 2.1.6. Función “leerDatosExcelSNI.py”

Como se muestra en la Figura 2.1.6, se encuentra la función “leerDatosExcelSNI.py” que sirve para leer los datos del SNI desde un archivo Excel. Esta función realiza la lectura cada una de las pestañas del archivo Excel y realiza la tabulación de los datos para los diferentes datos de cada uno de los elementos del sistema. Se define como clase “class” a cada una de las pestañas del Excel y se extrae los datos de las variables del sistema de cada una de las columnas de las diferentes pestañas con el comando “self” para guardarlo en su correspondiente clase. Finalmente, en la línea 186 se devuelve la variable “datos_sistema” donde se guardan todos los datos del sistema como los datos de las barras, las cargas, los generadores, las líneas de transmisión, los transformadores, demandas, influjos, entre otros.

Función Objetivo

Como se muestra en la Figura 2.1.7, la ecuación (5) que se refiere a la función objetivo del problema de despacho hidrotérmico de energía se encuentra en el archivo “modelo.py” y es una función que se llama desde el script principal “main.py”.

```

main.py ×  modelo.py ×  flujoAC.py ×  embal.py ×  embal_i.py ×  crearParams.py ×  graficas.py
129     def funcion_objetivo(model):
130         variable=0
131         if Modelo_Linealizado==0: #FO Cuadrática
132             variable+=sum(model.a[j]*model.Pt[i,j]*model.Pt[i,j]*model.U[i,j]
133                           for i in model.I
134                           for j in model.J)
135             variable+=sum(model.b[j]*model.Pt[i,j]*model.U[i,j]
136                           for i in model.I
137                           for j in model.J)
138             variable+=sum(model.c[j]*model.U[i,j] #Ecuación 5
139                           for i in model.I
140                           for j in model.J)
141         else: #FO lineal
142             variable+=sum(model.a[j]*model.U[i,j]
143                           for i in model.I
144                           for j in model.J)
145             variable+=sum(model.b[j]*model.Pt[i,j] #Ecuación 7
146                           for i in model.I
147                           for j in model.J)
148             variable+=sum(model.CA[j]*model.Y[i,j]
149                           for i in model.I
150                           for j in model.J)
151             variable+=sum(model.CP[j]*model.W[i,j]
152                           for i in model.I
153                           for j in model.J)
154             return variable #Ecuación 4
155         model.fun_obj = pyo.Objective(rule=funcion_objetivo)
156

```

Figura 2.1.7. Función “modelo.py”

En la línea 129 se define la función “funcion_objetivo” mediante la función “def” y se define el modelo abstracto creado de Pyomo “model” de la línea 8 del script “modelo.py”, luego se inicializa la “variable” en cero en la línea 130 y se ingresa a un lazo “if-else” donde se crea la ecuación de la función objetivo; se tienen dos opciones, primera función de costo de las centrales térmicas cuadrática (no lineal) que corresponde a la ecuación (5), y, segunda la función de costos de las centrales térmicas linealizada mediante la ecuación (7).

Para el primer caso, la función objetivo del problema se forma mediante la sumatoria de la función de costos cuadrática que corresponde a la ecuación (5) que se forma desde la línea 132 hasta la línea 140 donde “model.I” es el modelo para el contador del tiempo en horas y “model.J” es el modelo para el contador de las centrales térmicas, primero se tiene desde la línea 132 hasta la línea 134 el parámetro de costos cuadráticos a para las centrales térmicas “model.a” por el producto de las potencias activas generadas de las centrales térmicas por hora “model.Pt” elevado al cuadrado por la variable binaria “model.U”, luego se realiza la sumatoria desde la línea 135 hasta la línea 137 el parámetro de costos

b para las centrales térmicas “model.b” por el producto de las potencias activas generadas de las centrales térmicas por hora “model.Pt” por la variable binaria “model.U[i,j]”, luego se realiza la sumatoria desde la línea 138 hasta la línea 140 el parámetro de costos c para las centrales térmicas “model.c” por el producto de la variable binaria “model.U[i,j]”, luego se realiza la sumatoria desde la línea 148 hasta la línea 150 de los costos fijos de arranque de las centrales térmicas “model.CA” por el producto de la variable binaria “model.Y”, más desde la línea 151 hasta la línea 153 los costos fijos de parada de las centrales térmicas “model.CP” por el producto de la variable binaria “model.W”. Finalmente, mediante esta sumatoria se devuelve la función objetivo total en la línea 154 con la función “return” en el argumento “variable”, y, en la línea 155 se declara la función objetivo como un objeto de Pyomo para la optimización.

Para el segundo caso, la función objetivo del problema se forma mediante la sumatoria de la función de costos lineal que corresponde a la ecuación (7) que se forma desde la línea 141 hasta la línea 147 donde “model.I” es el modelo para el contador del tiempo en horas y “model.J” es el modelo para el contador de las centrales térmicas, primero se tiene desde la línea 142 hasta la línea 144 el parámetro de costos a para las centrales térmicas “model.a” por el producto de la variable binaria “model.U”, luego se realiza la sumatoria desde la línea 145 hasta la línea 147 el parámetro de costos b para las centrales térmicas “model.b” por el producto de las potencias activas generadas de las centrales térmicas por hora “model.Pt”, luego se realiza la sumatoria desde la línea 148 hasta la línea 150 de los costos fijos de arranque de las centrales térmicas “model.CA” por el producto de la variable binaria “model.Y”, más desde la línea 151 hasta la línea 153 los costos fijos de parada de las centrales térmicas “model.CP” por el producto de la variable binaria “model.W”. Finalmente, mediante esta sumatoria se devuelve la función objetivo total en la línea 154 con la función “return” en el argumento “variable” y en la línea 155 la función objetivo se la declara como un objeto de Pyomo para la optimización del problema.

Restricciones

```

156
157
158     ## DEFINE RESTRICCIONES
159
160     # Restricción Balance de potencia:
161     def balance_potencia_p(model, i):# para cada uno de los elementos de horas se tiene que cumplir con la demanda P
162         print("Balance P Hora:",i)
163         # Sumatoria de Generadores
164         if Modelo_Linealizado==0:
165             Pt=sum((model.Pt[i,j])*model.U[i,j]
166                   for j in model.J)
167         else:
168             Pt=sum((model.Pt[i,j])
169                   for j in model.J)
170         Ph=sum(model.Ph[i,k]
171               for k in model.K)
172         # Sumatoria de Cargas
173         Pd=sum(model.dt[i,d]
174               for d in model.D)
175         # Calculamos Las Pérdidas de potencia activa
176         NUMEROBARRAS = len(red.BUS.name) #Obtiene el número de barras de la Red
177         Pnm=[] #Matriz Pnm de inicialización
178         for ii in range(NUMEROBARRAS):
179             YnmVm=[]
180             for jj in range(NUMEROBARRAS):
181                 YnmVm.append(np.real(Ybarra1[ii,jj])*model.VoltajeBarras[i,jj])
182                 Pnm.append(np.sum(YnmVm)*model.VoltajeBarras[i,ii])
183         Perdidas=np.sum(Pnm) #Ecuación 8
184         return Pt+Ph==Pd+Perdidas*Sbase
185

```

Figura 2.1.8. Función “modelo.py”

Como se muestra en la Figura 2.1.8 en la línea 161 se define la función “balance_potencia_p” para el balance de potencia activa mediante la función “def”, se define el modelo abstracto creado de Pyomo “model” de la línea 8 del script “modelo.py” y se define índice i que corresponde al periodo de tiempo en horas, donde para cada uno de los elementos en cada hora en un periodo de tiempo se tiene que cumplir con la demanda de potencia activa. Primero, se realiza la sumatoria de potencias activas generadas por las centrales térmicas P_t en un periodo de tiempo desde la línea 164 hasta la línea 169, luego se tiene la sumatoria de las potencias activas generadas por las centrales hidroeléctricas P_h en un periodo de tiempo desde la línea 170 hasta la línea 171, después se tiene la sumatoria de las potencias activas de la demanda o las cargas P_d en un periodo de tiempo desde la línea 172 hasta la línea 174. Después, se calcula las pérdidas de potencia activa desde la barra n hasta la barra m “Perdidas” desde la línea 175 hasta la línea 182, donde a través de dos lazos “for” se forma la sumatoria entre la parte real de las admitancias entre la barra n hasta la barra m “Ybarra1” que se obtienen de la función “Ybarra” por el producto de la magnitud de voltaje en la barra m en la línea 181 “model.VoltajeBarras[i,jj]” que se obtiene de la función “flujoAC.py” y luego en la línea 182 para obtener las pérdidas de potencia activa “Perdidas” se le multiplica por la magnitud de voltaje en la barra n “model.VoltajeBarras[i,ii]” que se obtiene de la función “flujoAC.py”. El resultado de las pérdidas de potencia activa en p.u. se almacena en la variable “Perdidas” en la línea 183 de la función “modelo.py”. Finalmente, en la línea 184 se tiene la ecuación (8) expresada

en [MW] donde las potencias activas generadas por las centrales térmicas P_t más las potencias activas generadas por las centrales hidroeléctricas P_h es igual a las potencias activas de las cargas P_d más las pérdidas de potencias activas en p.u. “Perdidas” por la potencia aparente base “Sbase” que es igual a 100 [MVA] y se encuentra en la función “main.py”.

```

185
186 def balance_potencia_q(model, i):# para cada uno de los elementos de horas se tiene que cumplir con la demanda Q
187     print("Balance Q Hora:",i)
188     # Sumatoria de Generadores
189     if Modelo_Linealizado==0:
190         Qt=sum((model.Qt[i,j])*model.U[i,j]
191               for j in model.J)
192     else:
193         Qt=sum(model.Qt[i,j]
194               for j in model.J)
195         Qh=sum(model.Qh[i,k]
196               for k in model.K)
197         # Sumatoria de Cargas
198         Qd=sum(model.dtc[i,d]
199               for d in model.D)
200         # Calculamos Las Perdidas
201         NUMEROBARRAS = len(red.BUS.name)
202         Qnm=[]
203         for ii in range(NUMEROBARRAS):
204             YnmVm=[]
205             for jj in range(NUMEROBARRAS):
206                 YnmVm.append(-1*np.imag(Ybarra1[ii,jj])*model.VoltajeBarras[i,jj])
207             Qnm.append(np.sum(YnmVm)*model.VoltajeBarras[i,ii])
208         PerdidasQ=np.sum(Qnm) #Ecuación 9
209         return Qt+Qh==Qd+PerdidasQ*Sbase
210
211 if Restricciones[0]==1:
212     model.bal_pot_p = pyo.Constraint(model.I, rule=balance_potencia_p)
213 if Restricciones[1]==1:
214     model.bal_pot_q = pyo.Constraint(model.I, rule=balance_potencia_q)
215

```

Figura 2.1.9. Función “modelo.py”

Como se muestra en la Figura 2.1.9 en la línea 186 se define la función “balance_potencia_q” para el balance de potencia reactiva mediante la función “def”, se define el modelo abstracto creado de Pyomo “model” de la línea 8 del script “modelo.py” y se define i que corresponde al periodo de tiempo en horas, donde para cada uno de los elementos en cada hora en un periodo de tiempo se tiene que cumplir con la demanda de potencia reactiva. Primero, se realiza la sumatoria de potencias reactivas generadas por las centrales térmicas Q_t en un periodo de tiempo desde la línea 188 hasta la línea 194, luego se tiene la sumatoria de las potencias reactivas generadas por las centrales hidroeléctricas Q_h en un periodo de tiempo desde la línea 195 hasta la línea 196, después se tiene la sumatoria de las potencias reactivas de la demanda o las cargas Q_d en un periodo de tiempo desde la línea 197 hasta la línea 199. Después, se calcula las pérdidas de potencia reactiva desde la barra n hasta la barra m “PerdidasQ” desde la línea 200 hasta la línea 209 donde a través de dos lazos “for” donde se forma la sumatoria entre la parte

imaginaria de la admitancia entre la barra n hasta la barra m “Ybarra1” que se obtienen de la función “Ybarra” por el producto de la magnitud de voltaje en la barra m en la línea 206 “model.VoltajeBarras[i,jj]” que se obtiene de la función “flujoAC.py” y luego en la línea 207 para obtener las pérdidas de potencia reactiva “PerdidasQ” se le multiplica por la magnitud de voltaje en la barra n “model.VoltajeBarras[j,ii]” que se obtiene de la función “flujoAC.py”. El resultado de las pérdidas de potencia reactiva p.u. se almacena en la variable “PerdidasQ” en la línea 208 de la función “modelo.py”. Finalmente, en la línea 209 se tiene la ecuación (9) expresada en [MVA r] donde las potencias reactivas generadas por las centrales térmicas Q_t más las potencias reactivas generadas por las centrales hidroeléctricas Q_h es igual a las potencias reactivas de las cargas Q_d más las pérdidas de potencias reactivas en p.u. “PerdidasQ” por la potencia aparente base “Sbase” que es igual a 100 [MVA] y se encuentra en la función “main.py”. Además, desde la línea 211 hasta la línea 214 se define las restricciones mediante el comando de Pyomo “pyo.Constraint” como la primera restricción “Restricciones[0]” al balance de potencia activa y a la segunda restricción “Restricciones[1]” al balance de potencia reactiva

```

299
300 -----Restricción Límite de generación:-----
301 def limite_generaciont_max(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
302     return model.Pt[i,j]<=model.Pmaxt[j]*model.U[i,j] #Ecuación 10
303
304 def limite_generaciont_min(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
305     return model.Pt[i,j]>=model.Pmint[j]*model.U[i,j] #Ecuación 10
306
307 def limite_generacionh_max(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
308     return model.Ph[i,j]<=model.Pmaxh[j] #Ecuación 11
309
310 def limite_generacionh_min(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
311     return model.Ph[i,j]>=model.Pminh[j] #Ecuación 11
312
313 if Restricciones[4]==1:
314     model.lim_genh_max = pyo.Constraint(model.I, model.K, rule=limite_generacionh_max)
315     model.lim_gent_max = pyo.Constraint(model.I, model.J, rule=limite_generaciont_max)
316     model.lim_gent_min = pyo.Constraint(model.I, model.J, rule=limite_generaciont_min)
317     model.lim_genh_min = pyo.Constraint(model.I, model.K, rule=limite_generacionh_min)

```

Figura 2.1.10. Función “modelo.py”

Como se muestra en la Figura 2.1.10 en la línea 301 se define la función “limite_generaciont_max” para el límite de potencia activa máxima generada por las centrales térmicas, en la línea 304 se define la función “limite_generaciont_min” para el límite de potencia activa mínima generada por las centrales térmicas, en la línea 307 se define la función “limite_generacionh_max” para el límite de potencia activa máxima generada por las centrales hidroeléctricas, en la línea 310 se define la función “limite_generacionh_min” para el límite de potencia activa mínima generada por las centrales hidroeléctricas, mediante la función “def”.

Primero, en la línea 302 y en la línea 305 se tiene la ecuación (10) donde en la línea 302 la potencia activa generada por las centrales térmicas “model.Pt” debe ser menor o igual a la potencia activa máxima generada por las centrales térmicas “model.Pmaxt” multiplicada por la variable binaria “model.U” y en la línea 305 la potencia activa generada por las centrales térmicas “model.Pt” debe ser mayor o igual a la potencia activa mínima generada por las centrales térmicas “model.Pmint” multiplicada por la variable binaria “model.U”.

Además, en la línea 308 y en la línea 311 se tiene la ecuación (11) donde en la línea 308 la potencia activa generada por las centrales hidroeléctricas “model.Ph” debe ser menor o igual a la potencia activa máxima generada por las centrales hidroeléctricas “model.Pmaxh” y en la línea 311 la potencia activa generada por las centrales hidroeléctricas “model.Ph” debe ser mayor o igual a la potencia activa mínima generada por las centrales hidroeléctricas “model.Pminh”. Finalmente, desde la línea 313 hasta la línea 317 se definen las restricciones del límite de potencias activas generadas de las centrales térmicas e hidroeléctricas mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la quinta restricción “Restricciones[4]”.

```

318
319 - def limite_generaciont_max_q(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
320     return model.Qt[i,j]<=model.Qmaxt[j]*model.U[i,j] #Ecuación 12
321 - def limite_generaciont_min_q(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
322     return model.Qt[i,j]>=model.Qmint[j]*model.U[i,j] #Ecuación 12
323 - def limite_generacionh_max_q(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
324     return model.Qh[i,j]<=model.Qmaxh[j] #Ecuación 13
325 - def limite_generacionh_min_q(model, i, j):# la generacion debe cumplir con el limite permitido en cada hora y
326     return model.Qh[i,j]>=model.Qminh[j] #Ecuación 13
327
328 - if Restricciones[5]==1:
329     model.lim_gent_max_q = pyo.Constraint(model.I, model.J, rule=limite_generaciont_max_q)
330     model.lim_gent_min_q = pyo.Constraint(model.I, model.J, rule=limite_generaciont_min_q)
331     model.lim_genh_max_q = pyo.Constraint(model.I, model.K, rule=limite_generacionh_max_q)
332     model.lim_genh_min_q = pyo.Constraint(model.I, model.K, rule=limite_generacionh_min_q)
333

```

Figura 2.1.11. Función “modelo.py”

Como se muestra en la Figura 2.1.11 en la línea 319 se define la función “limite_generaciont_max_q” para el límite de potencia reactiva máxima generada por las centrales térmicas, en la línea 321 se define la función “limite_generaciont_min_q” para el límite de potencia reactiva mínima generada por las centrales térmicas, en la línea 323 se define la función “limite_generacionh_max_q” para el límite de potencia reactiva máxima generada por las centrales hidroeléctricas, en la línea 325 se define la función “limite_generacionh_min_q” para el límite de potencia reactiva mínima generada por las centrales hidroeléctricas, mediante la función “def”.

Primero, en la línea 320 y en la línea 322 se tiene la ecuación (12) donde en la línea 320 la potencia reactiva generada por las centrales térmicas “model.Qt” debe ser menor o igual

a la potencia reactiva máxima generada por las centrales térmicas “model.Qmaxt” multiplicada por la variable binaria “model.U” y en la línea 322 la potencia reactiva generada por las centrales térmicas “model.Qt” debe ser mayor o igual a la potencia reactiva mínima generada por las centrales térmicas “model.Qmint” multiplicada por la variable binaria “model.U”.

Además, en la línea 324 y en la línea 326 se tiene la ecuación (13) donde en la línea 324 la potencia reactiva generada por las centrales hidroeléctricas “model.Qh” debe ser menor o igual a la potencia reactiva máxima generada por las centrales hidroeléctricas “model.Qmaxh” y en la línea 326 la potencia reactiva generada por las centrales hidroeléctricas “model.Qh” debe ser mayor o igual a la potencia reactiva mínima generada por las centrales hidroeléctricas “model.Qminh”. Finalmente, desde la línea 328 hasta la línea 332 se definen las restricciones del límite de potencias reactivas generadas de las centrales térmicas e hidroeléctricas mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la sexta restricción “Restricciones[5]”.

```

333
334 #-----Restricción RAMPAS DE SUBIDA Y BAJADA-----
335 def rampa_bajadat(model, i, j):# Condición de rampa
336     if i+1<=model.t-1:
337         return model.Pt[i,j]-model.Pt[i+1,j]<=model.Pbajt[j]*model.U[i+1,j]+model.Ppart[j]*model.W[i+1,j] #rampa de parada y bajada Ec.19
338     else:
339         return pyo.Constraint.Skip
340
341 def rampa_subidat(model, i, j):# Condición de rampa
342     if i+1<=model.t-1:
343         return model.Pt[i+1,j]-model.Pt[i,j]<=model.Psubt[j]*model.U[i+1,j]+model.Parrt[j]*model.Y[i+1,j] #rampa de arranque y subida Ec.15
344     else:
345         return pyo.Constraint.Skip
346
347 def rampa_subidah(model, i, j):# Condición de rampa subida hidrica
348     if i+1<=model.t-1:
349         return model.Ph[i+1,j]-model.Ph[i,j]<=model.Psubh[j] #Ecuación 17
350     else:
351         return pyo.Constraint.Skip
352
353 if Restricciones[6]==1:
354     model.ram_subt = pyo.Constraint(model.I, model.J, rule=rampa_subidat)
355     model.ram_subh = pyo.Constraint(model.I, model.K, rule=rampa_subidah)
356     model.ram_bajt = pyo.Constraint(model.I, model.J, rule=rampa_bajadat)

```

Figura 2.1.12. Función “modelo.py”

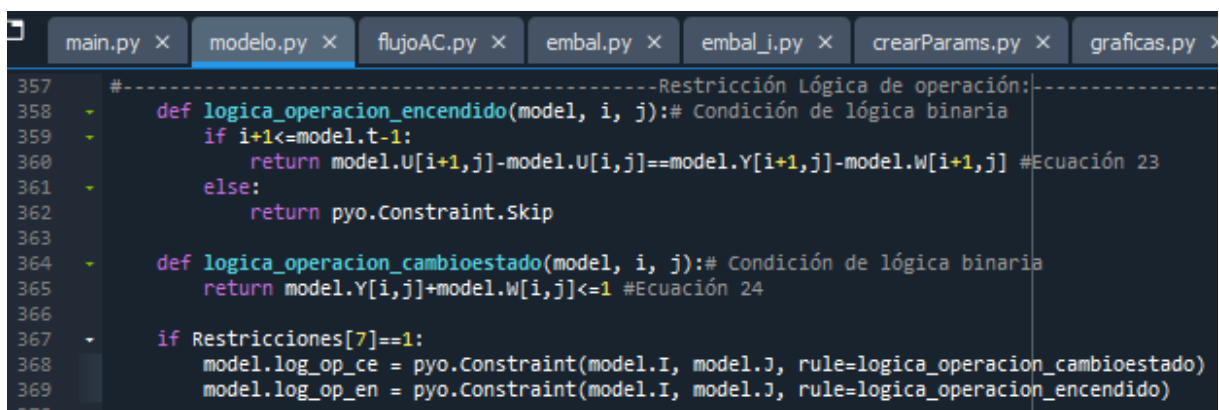
Como se muestra en la Figura 2.1.12 en la línea 335 se define la función “rampa_bajadat” para la rampa de parada y bajada de las centrales térmicas, en la línea 341 se define la función “rampa_subidat” para la rampa de arranque y toma de carga de las centrales térmicas, en la línea 347 se define la función “rampa_subidah” para la rampa de arranque y subida de las centrales hidroeléctricas, mediante la función “def”.

Primero, en la línea 337 se tiene la ecuación (19) correspondiente a la restricción de rampa de parada y bajada de las centrales térmicas donde la potencia activa generada por las centrales térmicas “model.Pt” en un tiempo $t - 1$ menos la potencia activa generada por las centrales térmicas “model.Pt” en un tiempo t debe ser menor o igual a la rampa de bajada de las centrales térmicas “model.Pbajt” multiplicada por la variable binaria “model.U”

más la rampa de parada de las centrales térmicas “model.Ppart” multiplicada por la variable binaria “model.W”.

Luego, en la línea 343 se tiene la ecuación (15) correspondiente a la restricción de rampa de arranque y subida de las centrales térmicas donde la potencia activa generada por las centrales térmicas “model.Pt” en un tiempo t menos la potencia activa generada por las centrales térmicas “model.Pt” en un tiempo $t - 1$ debe ser menor o igual a la rampa de subida de las centrales térmicas “model. Psubt” multiplicada por la variable binaria “model.U” en un tiempo $t - 1$ más la rampa de arranque de las centrales térmicas “model.Parrt” multiplicada por la variable binaria “model.Y”.

Después, en la línea 349 se tiene la ecuación (17) correspondiente a la restricción de rampa de arranque y subida de las centrales hidroeléctricas donde la potencia activa generada por las centrales hidroeléctricas “model.Ph” en un tiempo t menos la potencia activa generada por las centrales hidroeléctricas “model.Ph” en un tiempo $t - 1$ debe ser menor o igual a la rampa de subida de las centrales hidroeléctricas “model. Psubh”. Finalmente, desde la línea 353 hasta la línea 356 se definen las restricciones para la rampa de arranque y subida de las centrales hidroeléctricas y térmicas, además de las restricciones de parada y bajada de las centrales térmicas mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la séptima restricción “Restricciones[6]”.

The image shows a screenshot of a code editor with several tabs open: 'main.py', 'modelo.py', 'flujoAC.py', 'embal.py', 'embal_j.py', 'crearParams.py', and 'graficas.py'. The 'modelo.py' tab is active, displaying Python code. The code includes comments in Spanish and defines two functions: 'logica_operacion_encendido' and 'logica_operacion_cambioestado'. The first function is defined on line 358 and returns a Pyomo constraint based on a binary condition. The second function is defined on line 364 and returns a Pyomo constraint. There is also a conditional block starting on line 367 that applies these constraints based on the value of 'Restricciones[7]'.

```
357 #-----Restricción Lógica de operación:-----
358 - def logica_operacion_encendido(model, i, j):# Condición de lógica binaria
359 -     if i+1<=model.t-1:
360 -         return model.U[i+1,j]-model.U[i,j]==model.Y[i+1,j]-model.W[i+1,j] #Ecuación 23
361 -     else:
362 -         return pyo.Constraint.Skip
363
364 - def logica_operacion_cambioestado(model, i, j):# Condición de lógica binaria
365 -     return model.Y[i,j]+model.W[i,j]<=1 #Ecuación 24
366
367 - if Restricciones[7]==1:
368 -     model.log_op_ce = pyo.Constraint(model.I, model.J, rule=logica_operacion_cambioestado)
369 -     model.log_op_en = pyo.Constraint(model.I, model.J, rule=logica_operacion_encendido)
```

Figura 2.1.13. Función “modelo.py”

Como se muestra en la Figura 2.1.13 en la línea 358 se define la función “logica_operacion_encendido” que define la lógica binaria de operación, en la línea 364 se define la función “logica_operacion_cambioestado” que define la lógica binaria de operación.

Primero, en la línea 360 se tiene la ecuación (23) correspondiente a la lógica binaria de operación de las centrales térmicas donde la variable binaria “model.U” en un tiempo t menos la variable binaria “model.U” en un tiempo $t - 1$ debe ser igual a la variable binaria “model.Y” en un tiempo t menos la variable binaria “model.W” en un tiempo t . Luego, en la línea 364 se tiene la ecuación (24) correspondiente a la condición de la lógica binaria de operación de las centrales térmicas donde la variable binaria “model.Y” en un tiempo t más la variable binaria “model.W” en un tiempo t debe ser menor o igual a la unidad.

Finalmente, desde la línea 367 hasta la línea 369 se definen las restricciones para la lógica binaria de operación de las centrales térmicas, mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la octava restricción “Restricciones[7]”.

```

370
371 #-----Restricción Volumen del embalse inicial y final:-----
372 def embalse_min(model, i, j):# Volumen mínimo de embalse
373     if model.embalse[j]==1:
374         if TipoCalculoProduccion==0:
375             embalse, qA=em.embal(model, red,i, j, Modelo_SNI)# Cálculo Factores de descarga
376         else:
377             embalse, qA=emi.embal(model, red,i, j, Modelo_SNI)# Cálculo eficiencia
378         return embalse>=model.Vmin[j]*1000000 #Ecuación 25
379     else:
380         return pyo.Constraint.Skip
381
382 def embalse_max(model, i, j):# Volumen máximo de embalse
383     if model.embalse[j]==1:
384         if TipoCalculoProduccion==0:
385             embalse, qA=em.embal(model, red,i, j, Modelo_SNI)# Cálculo Factores de descarga
386         else:
387             embalse, qA=emi.embal(model, red,i, j, Modelo_SNI)# Cálculo eficiencia
388         return embalse<=model.Vmax[j]*1000000 #Ecuación 25
389     else:
390         return pyo.Constraint.Skip
391
392 if Restricciones[8]==1:
393     model.emb_max = pyo.Constraint(model.I, model.K, rule=embalse_max)
394     model.emb_min = pyo.Constraint(model.I, model.K, rule=embalse_min)
395

```

Figura 2.1.14. Función “modelo.py”

Como se muestra en la Figura 2.1.14 en la línea 372 se define la función “embalse_min” para el volumen mínimo del embalse de las centrales hidroeléctricas, en la línea 382 se define la función “embalse_max” para el volumen máximo del embalse de las centrales hidroeléctricas, mediante la función “def”.

Primero, en la línea 378 y en la línea 388 se tiene la ecuación (25) donde en la línea 378 el volumen de agua del embalse de las centrales hidroeléctricas “embalse” debe ser mayor o igual al volumen mínimo del embalse de las centrales hidroeléctricas “model.Vmin”

multiplicado por 1000000 para obtener en las unidades de $[Hm^3]$ y en la línea 388 el volumen de agua del embalse de las centrales hidroeléctricas “embalse” debe ser menor o igual al volumen máximo del embalse de las centrales hidroeléctricas “model.Vmax” multiplicado por 1000000 para obtener en las unidades de $[Hm^3]$.

Se implementan dos formas para determinar el caudal turbinado por las centrales hidroeléctricas usando la función “TipoCalculoProduccion” donde la opción “0” realiza el cálculo usando los coeficientes de descarga mediante la función “embal.py” y la opción “1” realiza el cálculo mediante la eficiencia de la central mediante la función “embal_i.py”.

Finalmente, desde la línea 392 hasta la línea 394 se define la restricción del volumen máximo y mínimo de embalses de las centrales hidroeléctricas mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la novena restricción “Restricciones[8]”.

```

1  """
2  Programa que calcula el Embalse y caudal del sistema usando factores de descarga
3  """
4  def embal(model,red, ii, jj, Modelo_SNI):
5      # Obtengo datos iniciales
6      PhMax=model.Pmaxh[jj]*1000*1000#W
7      qmax=model.qmax[jj]*m^3/h
8      eh=model.eh[jj]*m^3/h
9      Vini=model.Vini[jj]*1000000#m^3
10     upstream=model.upstream[jj]
11     DenAgua=1000#kg/m^3
12     if Modelo_SNI==1:
13         altura=model.altura[jj]# m
14         largo=model.largo[jj]# m
15         volument=model.volument[jj]# m
16         unidades=model.unidades[jj]# m
17     if upstream=='-': # verifico si la central tiene un upstream (una central río arriba)
18         embalseaux=[]
19         embalseA=Vini
20         embalse=0
21     #Calculamos el Caudal
22     for l in range(ii+1):
23         Phl=model.Ph[l, jj]*1000*1000#W
24         rhtl=3600*model.rht[l, jj] #Influjo m^3/h
25         #Caudal
26         if Modelo_SNI==1: # Para el SNI verifica si hay datos de altura para calcular el caudal con e
27             if altura==0:
28                 ql=Phl*3600/(eh*1000*1000)#m^3/h Ecuación 28
29             else:
30                 varFaltante=volument/(largo*altura)
31                 altural=embalseA/(largo*varFaltante)
32                 efil=(PhMax/unidades)/(9.8*DenAgua*altura*qmax)
33                 ql=(3600*Phl/unidades)/(9.8*DenAgua*altural*efil)#m^3/h Ecuación 29

```

Figura 2.1.15. Función “embal.py”

Como se muestra en la Figura 2.1.15 para la función “embal.py” en la línea 4 se define la función “embal” para calcular el embalse y el caudal turbinado usando los factores de descarga de las centrales hidroeléctricas.

Primero, para el caso del SNI, si no se tiene datos de altura del embalse se calcula el caudal turbinado ql usando la eficiencia en la línea 28 donde se implementa la ecuación (28) mediante la potencia activa generada de las centrales hidroeléctricas Phl por 3600 dividido para la eficiencia eh por 1000000, para obtener el caudal turbinado en $[m^3/h]$.

Luego, para el caso del SNI, si se verifica que existen datos de altura del embalse se calcula el caudal turbinado ql de las centrales hidroeléctricas en la línea 33 donde se implementa la ecuación (29) mediante la potencia activa generada por la central Phl por 3600 dividido para la aceleración gravitacional “9.8” multiplicado por el salto hídrico $altural$ multiplicado por la eficiencia de la turbina $efil$ multiplicado por la densidad del agua $DenAgua$, para obtener el caudal turbinado en $[m^3/h]$.

```

34     else:
35         # PARA 14 Barras
36         # calculo caudal usando factores de descarga X, Y, Z
37         # El caudal debe quedar en m3/h
38         x=red.GHIDRO.x[jj]
39         y=red.GHIDRO.y[jj]
40         z=red.GHIDRO.z[jj]
41         ql=x*(60*60)*(Phl/(1000*1000))*(Phl/(1000*1000))+y*(60*60)*(Phl/(1000*1000))+z*(60*60)#m3/h Ecuación 30
42         # Cálculo el embalse tomando en cuenta el embalse de la hora anterior
43         if l==0:
44             embalse=vini+rhtl-ql #Ecuación 26
45             embalseA=vini+rhtl-ql
46         else:
47             embalse=embalseaux[l-1]+rhtl-ql #Ecuación 26
48             embalseA=embalseaux[l-1]+rhtl-ql
49         embalseaux.append(embalse)

```

Figura 2.1.16. Función “embal.py”

Como se muestra en la Figura 2.1.16, para el sistema de estudio IEEE 14 barras se calcula el caudal turbinado ql de las usando los factores de descarga, en la línea 41 se implementa la ecuación (30) mediante el factor de descarga x por 3600 multiplicado por la potencia activa generada de las centrales hidroeléctricas $Phl/1000000$ elevado al cuadrado más el factor de descarga y por 3600 multiplicado por la potencia activa generada de las centrales hidroeléctricas $Phl/1000000$ más el factor de descarga z por 3600, para obtener el caudal turbinado en $[m^3/h]$.

Además, en las líneas 44 y 47 se tiene la ecuación (26) que se refiere al cálculo del embalse independiente donde el volumen de agua del embalse de las centrales hidroeléctricas “embalse” es igual al volumen de agua de embalse de la hora anterior $t - 1$ de las centrales hidroeléctricas $Vini$ más 3600 por el influjo natural de embalse de las centrales hidroeléctricas “model.rht” y menos el caudal turbinado ql .


```

main.py × modelo.py × flujoAC.py × embal.py × embal_i.py × crearParams.py × graficas.py × leerDatosExcel.py ×
50 else: # si existe upstream es importante calcular el caudal y embalse de las generadores que estan río arriba
51 genAnteriores=str(model.upstream[jj]).split(";")
52 embalseaux=[]
53 embalseA=Vini
54 embalse=0
55 genAntL=len(genAnteriores)
56 for l in range(ii+1):
57     caudalA=0
58     for i in range(genAntL):
59         genAnt=int(genAnteriores[i])
60         embalseA, qA=embal(model, red,l, genAnt, Modelo_SNI)
61         caudalA+=qA
62         Phl=model.Ph[l,jj]*1000*1000#W
63         rhtl=3600*model.rht[l,jj]+caudalA #Influjo m^3/h
64         #Caudal
65         if Modelo_SNI==1:
66             if altura==0:
67                 ql=Phl*3600/(eh*1000*1000)#m^3/h Ecuación 28
68             else:
69                 varFaltante=volumen/(largo*altura)
70                 altural=embalseA/(largo*varFaltante)
71                 efil=(PhMax/unidades)/(9.8*DenAgua*altura*qmax)
72                 ql=(3600*Phl/unidades)/(9.8*DenAgua*altural*efil)#m^3/h Ecuación 29
73         else:
74
75             x=red.GHIDRO.x[jj]
76             y=red.GHIDRO.y[jj]
77             z=red.GHIDRO.z[jj]
78
79
80             ql=x*(60*60)*(Phl/(1000*1000))*(Phl/(1000*1000))+y*(60*60)*(Phl/(1000*1000))+z*(60*60)#m^3/h Ecuación 30
81         if l==0:
82             embalse=Vini+rhtl-ql #Ecuación 27
83             embalseA=Vini+rhtl-ql
84         else:
85             embalse=embalseaux[l-1]+rhtl-ql #Ecuación 27
86             embalseA=embalseaux[l-1]+rhtl-ql
87             embalseaux.append(embalse)
88         return embalse, ql

```

Figura 2.1.17. Función “embal.py”

Como se muestra en la Figura 2.1.17, desde la línea 50 hasta la línea 88 para el caso de las centrales hidroeléctricas en cascada, se calcula el caudal y embalse de las centrales hidroeléctricas que están río arriba. Si no existen datos de altura del embalse se calcula el caudal turbinado ql usando la eficiencia de la central. En caso se verifiquen datos de altura, se calcula el caudal turbinado ql en la línea 72 mediante la ecuación (29)

Además, para el caso de aplicación IEEE de 14 barras se calcula el caudal turbinado ql de las centrales hidroeléctricas usando factores de descarga en la línea 80 donde se tiene la ecuación (30) mediante el factor de descarga x por 3600 multiplicado por la potencia activa generada de las centrales hidroeléctricas $Phl/1000000$ elevado al cuadrado más el factor de descarga y por 3600 multiplicado por la potencia activa generada de las centrales hidroeléctricas $Phl/1000000$ más el factor de descarga z por 3600, para obtener el caudal turbinado en $[m^3/h]$.

Además, en las líneas 82 y 85 se tiene la ecuación (27) que se refiere al cálculo del embalse dependiente donde el volumen de agua del embalse de las centrales hidroeléctricas “embalse” es igual al volumen de agua de embalse de la hora anterior $t - 1$ de las centrales

hidroeléctricas *Vini* más 3600 por el influjo natural de embalse de las centrales hidroeléctricas “model.rht” y menos el caudal turbinado ql .

```

main.py x modelo.py x flujoAC.py x embal.py x embal_i.py x crearParams.py x graficas.py x leerDatosExcel.py x
10 upstream=model.upstream[jj]
11 DenAgua=1000#kg/m^3
12 if Modelo_SNI==1:
13     altura=model.altura[jj]# m
14     largo=model.largo[jj]# m
15     volument=model.volument[jj]# m
16     unidades=model.unidades[jj]# m
17 if upstream=='-': # verifico si la central tiene un upstream (una central río arriba)
18     embalseaux=[]
19     embalseA=Vini
20     embalse=0
21     #Calculamos el Caudal
22     for l in range(ii+1):
23         Phl=model.Ph[l,jj]*1000*1000#W
24         rhtl=3600*model.rht[l,jj] #Influjo m^3/h
25         #Caudal
26         if Modelo_SNI==1: # Para el SNI verifica si hay datos de altura para calcular el caudal con ella o con la eficien
27             # Ecuación 29
28             if altura==0:
29                 ql=Phl*3600/(eh*1000*1000)#m^3/h Ecuación 28
30             else:
31                 varFaltante=volument/(largo*altura)
32                 altura1=embalseA/(largo*varFaltante)
33                 efil=(PHMax/unidades)/(9.8*DenAgua*altura*qmax)
34                 ql=(3600*Phl/unidades)/(9.8*DenAgua*altura1*efil)#m^3/h Ecuación 29
35         else:
36             # PARA 14 Barras
37             # Calculo caudal usando eficiencia
38             # El caudal debe quedar en m3/h
39             # Ecuación 28
40             ql=Phl/(eh*1000*1000)#m^3/h Ecuación 28
41         # Cálculo el embalse tomando en cuenta el embalse de la hora anterior
42         if l==0:
43             embalse=Vini+rhtl-ql #Ecuación 26
44             embalseA=Vini+rhtl-ql
45         else:
46             embalse=embalseaux[l-1]+rhtl-ql #Ecuación 26
47             embalseA=embalseaux[l-1]+rhtl-ql
48         embalseaux.append(embalse)

```

Figura 2.1.18. Función “embal_i.py”

Como se muestra en la Figura 2.1.18 para la función “embal_i.py” en la línea 4 se define la función “embal” para calcular el embalse y el caudal del sistema usando la eficiencia de las centrales hidroeléctricas, mediante la función “def”.

Primero, desde la línea 26 hasta la línea 34 para el caso de aplicación del SNI si no se tiene datos de altura se calcula el caudal turbinado ql de las centrales hidroeléctricas usando la eficiencia en la línea 29 donde se implementa la ecuación (28). Además, para el SNI si se verifica que se tiene datos de altura se calcula el caudal turbinado ql de las centrales hidroeléctricas en la línea 34 donde se tiene la ecuación (29).

Después, desde la línea 35 hasta la línea 40 para el caso de aplicación IEEE de 14 barras se calcula el caudal turbinado ql de las centrales hidroeléctricas usando factores de descarga en la línea 40 donde se tiene la ecuación (30) mediante la potencia activa generada de las centrales hidroeléctricas Phl dividido para la eficiencia eh por 1000000, para obtener el caudal turbinado en $[m^3/h]$.

Además, en las líneas 43 y 46 se tiene la ecuación (26) que se refiere al cálculo del embalse independiente donde el volumen de agua del embalse de las centrales hidroeléctricas

“embalse” es igual al volumen de agua de embalse de la hora anterior $t - 1$ de las centrales hidroeléctricas *Vini* más 3600 por el influjo natural de embalse de las centrales hidroeléctricas “model.rht” y menos el caudal turbinado ql .

```

main.py x modelo.py x flujoAC.py x embal.py x embal_i.py x crearParams.py x graficas.py x leerDa
48     embalseaux.append(embalse)
49     else: # Si existe upstream es importante calcular el caudal y embalse de las generad
50         genAnteriores=str(model.upstream[jj]).split(";")
51         embalseaux=[]
52         embalseA=Vini
53         embalse=0
54         genAntL=len(genAnteriores)
55         for l in range(ii+1):
56             caudalA=0
57             for i in range(genAntL):
58                 genAnt=int(genAnteriores[i])
59                 embalseA, qA=embal(model, red,l, genAnt, Modelo_SNI)
60                 caudalA+=qA
61                 Phl=model.Ph[l, jj]*1000*1000#w
62                 rhtl=3600*model.rht[l, jj]+caudalA #Influjo m^3/h
63                 #Caudal
64                 if Modelo_SNI==1:
65                     if altura==0:
66                         ql=Phl*3600/(eh*1000*1000)#m^3/h Ecuación 28
67                     else:
68                         varFaltante=volumen/(largo*altura)
69                         altural=embalseA/(largo*varFaltante)
70                         efil=(PhMax/unidades)/(9.8*DenAgua*altura*qmax)
71                         ql=(3600*Phl/unidades)/(9.8*DenAgua*altural*efil)#m^3/h Ecuación 29
72                 else:
73                     ql=Phl/(eh*1000*1000)#m^3/h Ecuación 28
74                 if l==0:
75                     embalse=Vini+rhtl-ql #Ecuación 27
76                     embalseA=Vini+rhtl-ql
77                 else:
78                     embalse=embalseaux[l-1]+rhtl-ql #Ecuación 27
79                     embalseA=embalseaux[l-1]+rhtl-ql
80                 embalseaux.append(embalse)
81     return embalse, ql

```

Figura 2.1.19. Función “embal_i.py”

Como se muestra en la Figura 2.1.19, desde la línea 49 hasta la línea 81 para el caso del embalse dependiente si existe “upstream” se calcula el caudal y embalse de las centrales hidroeléctricas que están río arriba.

```

main.py x modelo.py x flujoAC.py x embal.py x embal_i.py x crearParams.py x graficas.py x leerDatos
397 def limite_caudal_min(model, i, j):# Caudal mínimo turbinado
398     if TipoCalculoProduccion==0:
399         embalse, qA=em.embal(model, red,i, j, Modelo_SNI)# CálculoFactores de descarga
400     else:
401         embalse, qA=emi.embal(model, red,i, j, Modelo_SNI)# Cálculo eficiencia
402     if Modelo_SNI==1:
403         qA=qA/3600
404
405     return qA>=model.qmin[j] #Ecuación 25 para caudal
406
407 def limite_caudal_max(model, i, j):# Caudal máximo turbinado
408     if TipoCalculoProduccion==0:
409         embalse, qA=em.embal(model, red,i, j, Modelo_SNI)# Cálculo Factores de descarga
410     else:
411         embalse, qA=emi.embal(model, red,i, j, Modelo_SNI)# Cálculo eficiencia
412     if Modelo_SNI==1:
413         qA=qA/3600
414     upstream=model.upstream[j]
415     if upstream=='-': # Verifico si no existe una central río arriba
416         return qA<=model.qmax[j] #Ecuación 25 para caudal
417     else: # existe upstream (central río arriba)
418         if model.embalse[j]==1: # verifico si tiene embalse
419             return qA<=model.qmax[j]
420         else: # si no tiene embalse el influjo esta determinado por las centrales río arriba
421             genAnteriores=str(model.upstream[j]).split(";")
422             genAntL=len(genAnteriores)
423             caudalA=0
424             for ii in range(genAntL):
425                 if TipoCalculoProduccion==0:
426                     embalseA, qAA=em.embal(model, red,i, j, Modelo_SNI) # Cálculo Factores de descarga
427                 else:
428                     embalseA, qAA=emi.embal(model, red,i, j, Modelo_SNI) # Cálculo eficiencia
429                 caudalA+=qAA
430             if Modelo_SNI==1:
431                 caudalA=caudalA/3600
432             return qA<=caudalA
433
434     if Restricciones[9]==1:
435         model.lim_cau_max = pyo.Constraint(model.I, model.K, rule=limite_caudal_max)
436         model.lim_cau_min = pyo.Constraint(model.I, model.K, rule=limite_caudal_min)

```

Figura 2.1.20. Función “modelo.py”

Como se muestra en la Figura 2.1.20 en la línea 397 se define la función “limite_caudal_min” para el caudal mínimo turbinado de las centrales hidroeléctricas, en la línea 407 se define la función “limite_caudal_max” para el caudal máximo turbinado de las centrales hidroeléctricas, mediante la función “def”.

Se implementan dos métodos de cálculo que se definen en la función “TipoCalculoProduccion” donde la opción “0” realiza el cálculo con los coeficientes de descarga mediante la función “embal.py” y la opción “1” realiza el cálculo con la eficiencia mediante la función “embal_i.py”.

Además, en la línea 405 y en la línea 416 se tiene la ecuación donde en la línea 405 el caudal turbinado de las centrales hidroeléctricas qA debe ser mayor o igual al caudal mínimo turbinado de las centrales hidroeléctricas “model.qmin” en $[m^3/h]$.

Luego, en la línea 416 para el caso de embalse independiente y en la línea 419 para el caso de embalse dependiente con “upstream” y para ambos casos el caudal turbinado de las centrales hidroeléctricas qA debe ser menor o igual al caudal máximo turbinado de las centrales hidroeléctricas “model.qmax” en $[m^3/h]$.

Además, para el caso de centrales de pasada, el influjo está determinado por las centrales río arriba y en la línea 432 el caudal turbinado de las centrales hidroeléctricas q_A debe ser menor o igual al caudal máximo turbinado de las centrales hidroeléctricas $caudal_A$ en $[m^3/h]$.

Finalmente, desde la línea 434 hasta la línea 436 se define la restricción del caudal máximo y mínimo turbinado de las centrales hidroeléctricas mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la décima restricción “Restricciones[9]”.

```

216 #-----FLUJO DE POTENCIA POR LAS LÍNEAS-----
217 def Limite_Pline(model,i,l): # Límite de Carga P de la línea
218     print("Flujo P Línea:",l," Hora:",i)
219     return model.Piji[i,l]<=model.Pijmax[l]
220 def Limite_Sline(model,i,l): # Límite de Carga S de la línea
221     print("Flujo S Línea:",l," Hora:",i)
222     return model.Siji[i,l]<=model.Pijmax[l]*model.Pijmax[l]
223 def Calculo_Pij(model,i,l): # Cálculo de la potencia P de la línea
224     print("Flujo Valor P Línea:",l," Hora:",i)
225     Pos0=int(pyomo.value(model.lineas_posicion[0,l]))
226     Pos1=int(pyomo.value(model.lineas_posicion[1,l]))
227     Gnm=np.real(Ybarra1[Pos0,Pos1])
228     Bnm=np.imag(Ybarra1[Pos0,Pos1])
229     Vn=model.VoltajeBarras[i,Pos0]
230     Vm=model.VoltajeBarras[i,Pos1]
231     Thetanm=model.ThetaBarras[i,Pos0]-model.ThetaBarras[i,Pos1]
232     P=np.abs(Vn*Vn*Gnm-Vn*Vm*(Gnm*pyomo.cos(Thetanm)+Bnm*pyomo.sin(Thetanm))) #Ecuación 31
233     return model.Piji[i,l]==P*Sbase
234 def Calculo_Qij(model,i,l): # Cálculo de la potencia Q de la línea
235     print("Flujo Valor Q Línea:",l," Hora:",i)
236     Pos0=int(pyomo.value(model.lineas_posicion[0,l]))
237     Pos1=int(pyomo.value(model.lineas_posicion[1,l]))
238     Gnm=np.real(Ybarra1[Pos0,Pos1])
239     Bnm=np.imag(Ybarra1[Pos0,Pos1])
240     Vn=model.VoltajeBarras[i,Pos0]
241     Vm=model.VoltajeBarras[i,Pos1]
242     Thetanm=model.ThetaBarras[i,Pos0]-model.ThetaBarras[i,Pos1]
243     Q=np.abs(-Vn*Vn*Bnm-Vn*Vm*(Gnm*pyomo.sin(Thetanm)-Bnm*pyomo.cos(Thetanm))) #Ecuación 33
244     return model.Qiji[i,l]==Q*Sbase
245 def Calculo_Sij(model,i,l): # Cálculo de la potencia S de la línea
246     print("Flujo Valor S Línea:",l," Hora:",i)
247     S=model.Qiji[i,l]*model.Qiji[i,l]+model.Piji[i,l]*model.Piji[i,l] #Ecuación 32
248     return model.Siji[i,l]==S
249 if Restricciones[2]==1:
250     model.Limite_Pline = pyomo.Constraint(model.I,model.L, rule=Limite_Pline)
251     model.Limite_Sline = pyomo.Constraint(model.I,model.L, rule=Limite_Sline)
252     model.Calculo_Pij = pyomo.Constraint(model.I,model.L, rule=Calculo_Pij)
253     model.Calculo_Qij = pyomo.Constraint(model.I,model.L, rule=Calculo_Qij)
254     model.Calculo_Sij = pyomo.Constraint(model.I,model.L, rule=Calculo_Sij)

```

Figura 2.1.21. Función “modelo.py”

Como se muestra en la Figura 2.1.21 en la línea 217 se define la función “Limite_Pline” para el límite de flujo de potencia activa en líneas de transmisión, en la línea 220 se define

la función "Limite_Sline" para el límite de flujo de potencia aparente en líneas de transmisión, mediante la función "def".

Primero, en la línea 232 se tiene la ecuación (31) para el cálculo del flujo de potencia activa en líneas de transmisión P en coordenadas polares en p.u., en donde, primero se obtiene la posición de las líneas de transmisión $Pos0$ y $Pos1$ mediante las líneas 225 y 226 respectivamente, luego en las líneas 227 y 228 se calcula la conductancia G_{nm} y susceptancia B_{nm} entre la barra n y la barra m , respectivamente, usando la matriz de admitancia $Ybarra1$. En la línea 229 se calcula el voltaje en la barra n V_n y en la línea 230 se calcula el voltaje en la barra m V_m con la ecuación de la magnitud de voltaje en las barras "model.VoltajeBarras" que se obtiene del flujo de potencia AC de la función "flujoAC.py". Además, en la línea 231 se calcula el ángulo $Theta_{nm}$ entre la barra n y la barra m con la ecuación de ángulo de voltaje en las barras "model.ThetaBarras" que se obtiene del flujo de potencia AC de la función "flujoAC.py". Luego, en la línea 233 se calcula el flujo de potencia activa en línea de transmisión P entre la barra n y la barra m en [MW] ya que la variable P_{iji} debe ser igual a $P * Sbase$. Finalmente, en la línea 219 el límite de la variable del flujo de potencia activa en las líneas de transmisión P_{iji} entre la barra n y la barra m debe ser menor o igual al parámetro de la potencia activa máxima de la línea de transmisión P_{ijmax} entre la barra n y la barra m .

Luego, en la línea 243 se tiene la ecuación (33) para el cálculo del flujo de potencia reactiva en líneas de transmisión Q en coordenadas polares en p.u., en donde primero se obtiene la posición de las líneas de transmisión $Pos0$ y $Pos1$ mediante las líneas 236 y 237 respectivamente. En las líneas 238 y 239 se calcula la conductancia G_{nm} y susceptancia B_{nm} entre la barra n y la barra m , respectivamente, mediante la matriz de admitancia $Ybarra1$, luego en la línea 240 se calcula el voltaje en la barra n V_n y en la línea 241 se calcula el voltaje en la barra m V_m con la ecuación de la magnitud de voltaje en las barras "model.VoltajeBarras" que se obtiene del flujo de potencia AC de la función "flujoAC.py". Además, en la línea 242 se calcula el ángulo $Theta_{nm}$ entre la barra n y la barra m con la ecuación de ángulo de voltaje en las barras "model.ThetaBarras" que se obtiene del flujo de potencia AC de la función "flujoAC.py". Luego, en la línea 244 se tiene el flujo de potencia reactiva en líneas de transmisión Q entre la barra n y la barra m en [MVar] ya que la variable Q_{iji} debe ser igual a $Q * Sbase$. Finalmente, en la línea 247 se tiene la ecuación (32) del flujo de potencia aparente en líneas de transmisión S en coordenadas polares entre la barra n y la barra m en [MVA] ya que en la línea 248 la variable S_{iji} debe ser igual a S . Además, en la línea 222 el límite de la variable del flujo de

potencia aparente en las líneas de transmisión S_{ij} entre la barra n y la barra m debe ser menor o igual al parámetro de la potencia aparente máxima de la línea de transmisión S_{ijmax} entre la barra n y la barra m .

Finalmente, desde la línea 249 hasta la línea 254 se define la restricción del límite de flujo de potencia activa y aparente por las líneas de transmisión mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la tercera restricción “Restricciones[2]”.

```

259 #-----
260 def Limite_PTrafos(model,i,l): # Límite de Carga P del Transformador
261     print("Flujo P Trafo:",l," Hora:",i)
262     return model.PijIT[i,l]<=model.PijTmax[l]
263 def Limite_STrafos(model,i,l): # Límite de Carga S del Transformador
264     print("Flujo S Línea:",l," Hora:",i)
265     return model.SijIT[i,l]<=model.PijTmax[l]*model.PijTmax[l]
266 def Calculo_Pij_T(model,i,l): # Cálculo de la potencia P del Transformador
267     print("Flujo Valor P Trafo:",l," Hora:",i)
268     Pos0=int(pyomo.value(model.trafos_posicion[0,l]))
269     Pos1=int(pyomo.value(model.trafos_posicion[1,l]))
270     Gnm=np.real(Ybarra1[Pos0,Pos1])
271     Bnm=np.imag(Ybarra1[Pos0,Pos1])
272     Vn=model.VoltajeBarras[i,Pos0]
273     Vm=model.VoltajeBarras[i,Pos1]
274     Thetanm=model.ThetaBarras[i,Pos0]-model.ThetaBarras[i,Pos1]
275     P=np.abs((Vn*Vn*Gnm-Vn*Vm*(Gnm*pyomo.cos(Thetanm)+Bnm*pyomo.sin(Thetanm)))) #Ecuación 31
276     return model.PijIT[i,l]==P*Sbase
277 def Calculo_Qij_T(model,i,l): # Cálculo de la potencia Q del Transformador
278     print("Flujo Valor Q Línea:",l," Hora:",i)
279     Pos0=int(pyomo.value(model.trafos_posicion[0,l]))
280     Pos1=int(pyomo.value(model.trafos_posicion[1,l]))
281     Gnm=np.real(Ybarra1[Pos0,Pos1])
282     Bnm=np.imag(Ybarra1[Pos0,Pos1])
283     Vn=model.VoltajeBarras[i,Pos0]
284     Vm=model.VoltajeBarras[i,Pos1]
285     Thetanm=model.ThetaBarras[i,Pos0]-model.ThetaBarras[i,Pos1]
286     Q=np.abs((-Vn*Vn*Bnm-Vn*Vm*(Gnm*pyomo.sin(Thetanm)-Bnm*pyomo.cos(Thetanm)))) #Ecuación 33
287     return model.QijIT[i,l]==Q*Sbase
288 def Calculo_Sij_T(model,i,l): # Cálculo de la potencia S del Transformador
289     print("Flujo Valor S Línea:",l," Hora:",i)
290     S=model.QijIT[i,l]*model.QijIT[i,l]+model.PijIT[i,l]*model.PijIT[i,l] #Ecuación 32
291     return model.SijIT[i,l]==S
292 if Restricciones[3]==1:
293     model.Limite_PTrafos = pyomo.Constraint(model.I,model.TRANSF, rule=Limite_PTrafos)
294     model.Limite_STrafos = pyomo.Constraint(model.I,model.TRANSF, rule=Limite_STrafos)
295     model.Calculo_Pij_T = pyomo.Constraint(model.I,model.TRANSF, rule=Calculo_Pij_T)
296     model.Calculo_Qij_T = pyomo.Constraint(model.I,model.TRANSF, rule=Calculo_Qij_T)
297     model.Calculo_Sij_T = pyomo.Constraint(model.I,model.TRANSF, rule=Calculo_Sij_T)

```

Figura 2.1.22. Función “modelo.py”

Como se muestra en la Figura 2.1.22 en la línea 260 se define la función “Limite_PTrafos” para el límite de flujo de potencia activa en transformadores, en la línea 263 se define la función “Limite_STrafos” para el límite de flujo de potencia aparente en transformadores, mediante la función “def”. El proceso de implementación es similar al descrito para líneas de transmisión. Finalmente, desde la línea 292 hasta la línea 297 se define la restricción

del límite de flujo de potencia activa y aparente en los transformadores mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la cuarta restricción “Restricciones[3]”.

```

439
440 #-----Restricción Límite de magnitud de voltaje:--
441 def Voltaje_MAX(model, i, j):# Magnitud de voltaje máximo
442     print("Limite Max Voltaje Barra:",j," Hora:",i)
443     # Encontramos la posición de las barras de los generadores
444     NUMEROBARRAS = len(red.BUS.name)
445     NameBarra= red.BUS.name
446     Nbarra_Ph=[]
447     Nbarra_Pt=[]
448     for l in range (NUMEROBARRAS):
449         aux=NameBarra[l]
450         exist_count = list(red.GEN.bus).count(aux)
451         if exist_count > 0:
452             nameGen=red.GEN.name[list(red.GEN.bus).index(aux)]
453             exist_count = list(red.GHIDRO.Referencia).count(nameGen)
454             if exist_count > 0:
455                 Nbarra_Ph.append(l)
456             else:
457                 exist_count = list(red.GTERM.Referencia).count(nameGen)
458                 if exist_count > 0:
459                     Nbarra_Pt.append(l)
460         aux=0
461         for l in range(len(Nbarra_Ph)):
462             if Nbarra_Ph[l]==j:
463                 aux=1
464
465         for l in range(len(Nbarra_Pt)):
466             if Nbarra_Pt[l]==j:
467                 aux=1
468         if aux==0:
469             return model.VoltajeBarras[i,j]<=model.Voltmax[j] #Ecuación 34
470         else:
471             return pyo.Constraint.Skip # Saltar si es un voltaje fijo de generadoras
472 def Voltaje_MIN(model, i, j):# Magnitud de voltaje mínimo
473     print("Limite Min Voltaje Barra:",j," Hora:",i)
474     # Encontramos la posición de las barras de los generadores
475     NUMEROBARRAS = len(red.BUS.name)
476     NameBarra= red.BUS.name

```

Figura 2.1.23. Función “modelo.py”

Como se muestra en la Figura 2.1.23 en la línea 441 se define la función “Voltaje_MAX” para la restricción del límite máximo de magnitud de voltaje en las barras y en la línea 472 se define la función “Voltaje_MIN” para la restricción del límite mínimo de magnitud de voltaje en las barras.

Primero, se encuentra la posición de las barras de los generadores y luego en la línea 469 se tiene la ecuación (34) donde la magnitud de voltaje en las barras “VoltajeBarras[i,j]” se obtiene del flujo de potencia AC en la función “flujoAC.py” debe ser menor o igual a la magnitud de voltaje máximo en las barras “Voltmax[j]”.


```

487         else:
488             exist_count = list(red.GTERM.Referencia).count(nameGen)
489             if exist_count > 0:
490                 Nbarra_Pt.append(1)
491             aux=0
492             for l in range(len(Nbarra_Ph)):
493                 if Nbarra_Ph[l]==j:
494                     aux=1
495
496             for l in range(len(Nbarra_Pt)):
497                 if Nbarra_Pt[l]==j:
498                     aux=1
499             if aux==0:
500                 return model.VoltajeBarras[i,j]>=model.voltmin[j] #Ecuación 34
501             else:
502                 return pyo.Constraint.Skip # Saltar si es un voltaje fijo de generadoras
503     def calculo_Voltajes(model, i, j): # Calculamos ecuaciones de Voltaje con el flujo AC
504         print("Voltaje Valor Barra:",j," Hora:",i)
505         if j==0: # Calculamos solo para la j=0 ya que el resultado del flujo entrega el valor para todas las barras
506             [VolBarras, ThetaBarras]=fac.FlujoAC(red, model, Ybarra1, Sbase, frecRed, i, Modelo_SNI)
507             for n in range (len(red.BUS.name)):
508                 VoltajeBarrasFlujo[i,n]=VolBarras[n] # ingresamos los valores en una variable auxiliar con las ecuaciones de
509             return model.VoltajeBarras[i,j]==VoltajeBarrasFlujo[i,j] # Indicamos el valor para cada barra optimizando el tiempo de
510     def calculo_Theta(model, i, j): # Calculamos ecuaciones de Angulo con el flujo AC
511         print("Theta Valor Barra:",j," Hora:",i)
512         if j==0: # Calculamos solo para la j=0 ya que el resultado del flujo entrega el valor para todas las barras
513             [VolBarras, ThetaBarras]=fac.FlujoAC(red, model, Ybarra1, Sbase, frecRed, i, Modelo_SNI)
514             for n in range (len(red.BUS.name)):
515                 ThetaBarrasFlujo[i,n]=ThetaBarras[n] # ingresamos los valores en una variable auxiliar con las ecuaciones de t
516             return model.ThetaBarras[i,j]==ThetaBarrasFlujo[i,j] # Indicamos el valor para cada barra optimizando el tiempo de ej
517     if Restricciones[10]==1:
518         model.Voltaje_MAX = pyo.Constraint(model.I,model.N, rule=Voltaje_MAX)
519         model.Voltaje_MIN = pyo.Constraint(model.I,model.N, rule=Voltaje_MIN)
520         model.Calculo_Voltajes = pyo.Constraint(model.I,model.N, rule=calculo_Voltajes)
521         model.Calculo_Theta = pyo.Constraint(model.I,model.N, rule=calculo_Theta)
522
523     return model
524

```

Figura 2.1.24. Función “modelo.py”

Como se muestra en la Figura 2.1.24, se encuentra la posición de las barras de los generadores y luego en la línea 500 se tiene la ecuación (34) donde la magnitud de voltaje en las barras “VoltajeBarras[i,j]” se obtiene del flujo de potencia AC en la función “flujoAC.py” debe ser mayor o igual a la magnitud de voltaje mínimo en las barras “Voltmin[j]”.

Finalmente, desde la línea 517 hasta la línea 521 se define la restricción del límite de magnitud de voltajes en las barras mediante el comando de Pyomo “pyo.Constraint” y se almacena en el menú del script “main.py” como la onceava restricción “Restricciones[10]”.

3 RESULTADOS

3.1 Casos de estudio

En el presente capítulo se presentarán los resultados obtenidos de forma gráfica de la herramienta computacional desarrollada para la solución del problema de optimización de despacho hidrotérmico de corto plazo considerando la red eléctrica usando flujo óptimo de potencia AC para los casos de aplicación del sistema IEEE de 14 barras y para el SNI ecuatoriano.

3.1.1 Aplicación al Sistema IEEE de 14 barras

Para validar la herramienta computacional desarrollada se aplica al sistema IEEE de 14 barras y 5 generadores que se presenta en la Figura 3.1.1.1.

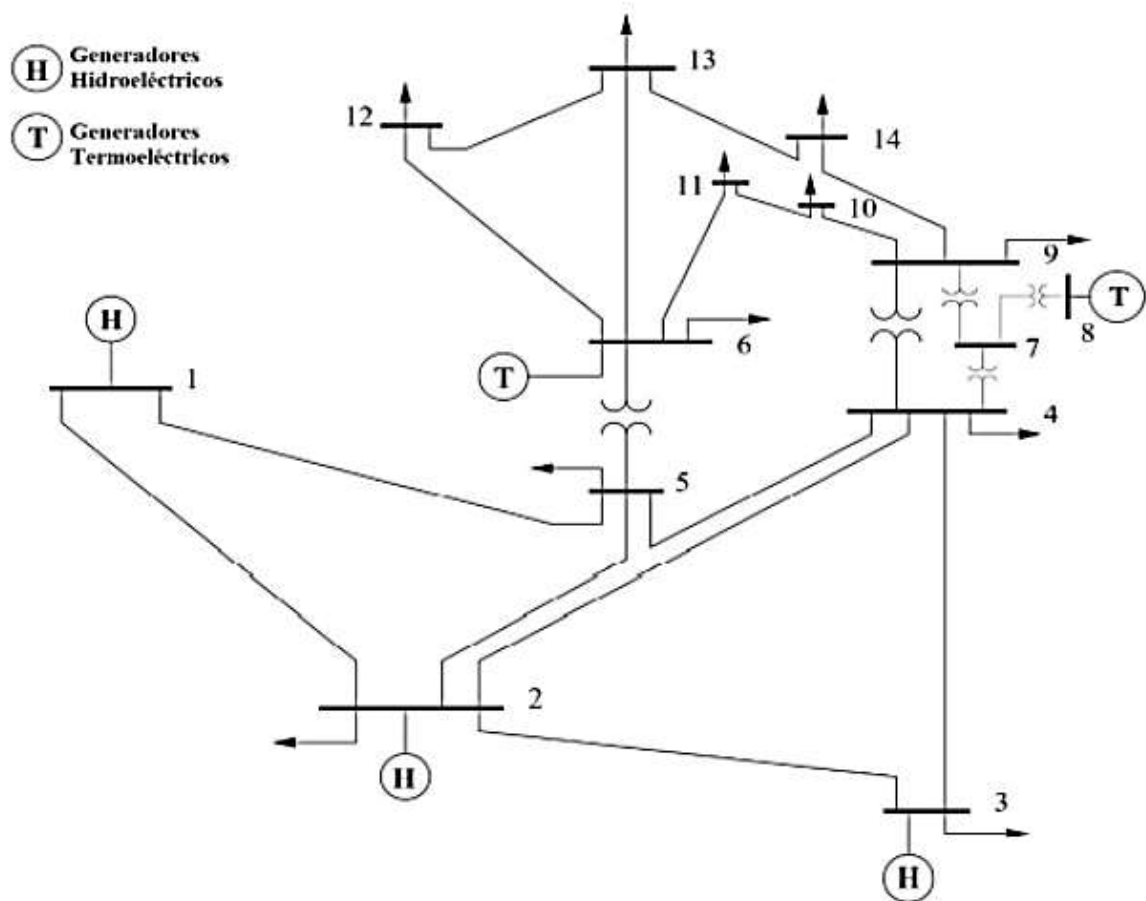


Figura 3.1.1.1. Sistema IEEE de 14 barras [14].

El sistema consta de tres centrales hidroeléctricas con embalse representadas con "H" y dos centrales térmicas representadas con "T", donde la CH que se encuentra en la barra 3

está ubicada aguas abajo de las dos CH restantes, por lo que se tiene concatenación hídrica “upstream” tal como se muestra en la Figura 3.1.1.2.

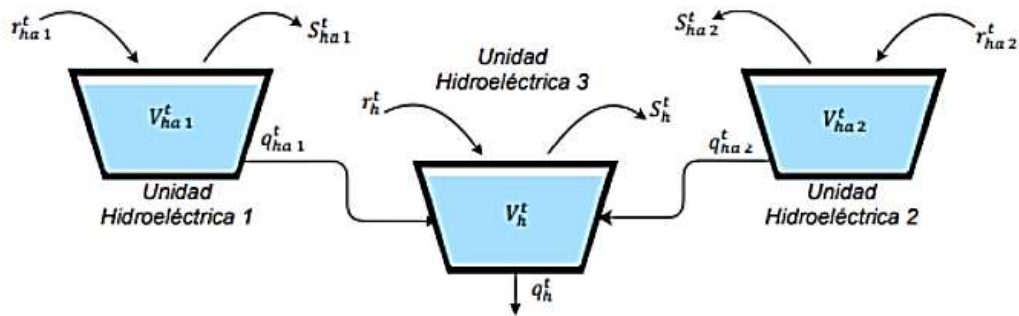


Figura 3.1.1.2. Concatenación hídrica del sistema IEEE de 14 barras [6].

- **Caso de estudio A**

Para este caso se considera la siguiente parametrización:

Tabla 1 Parametrización del modelo

Opción	Tipo
Función de costos de producción	Linealizada
Cálculo de función de producción	Con eficiencia
Solucionador	Ipopt
Cálculo con pérdidas en la red eléctrica	Con pérdidas

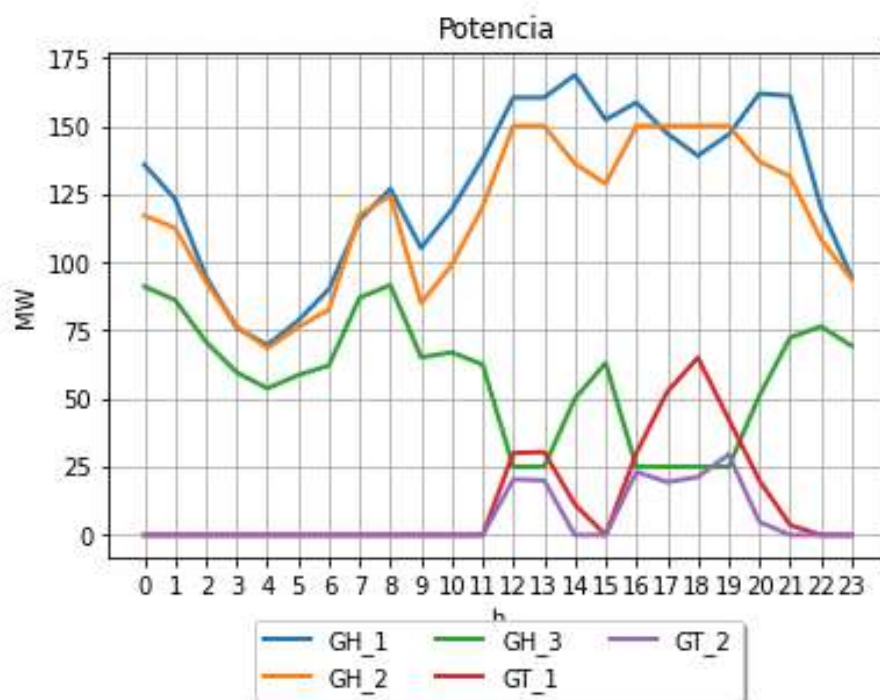


Figura 3.1.1.3. Potencia generada. Sistema IEEE de 14 barras.

En la Figura 3.1.1.3 se presenta la potencia generada por las centrales térmicas e hidroeléctricas para un período de 24 horas, donde se observa que la mayoría de la demanda es suministrada por las centrales hidroeléctricas ya que el costo de producción es más bajo en comparación a las centrales térmicas. En las horas pico, desde las 18 hasta las 21 horas, ambas centrales térmicas se encuentran en funcionamiento para suplir la demanda, posteriormente ambas salen de operación.

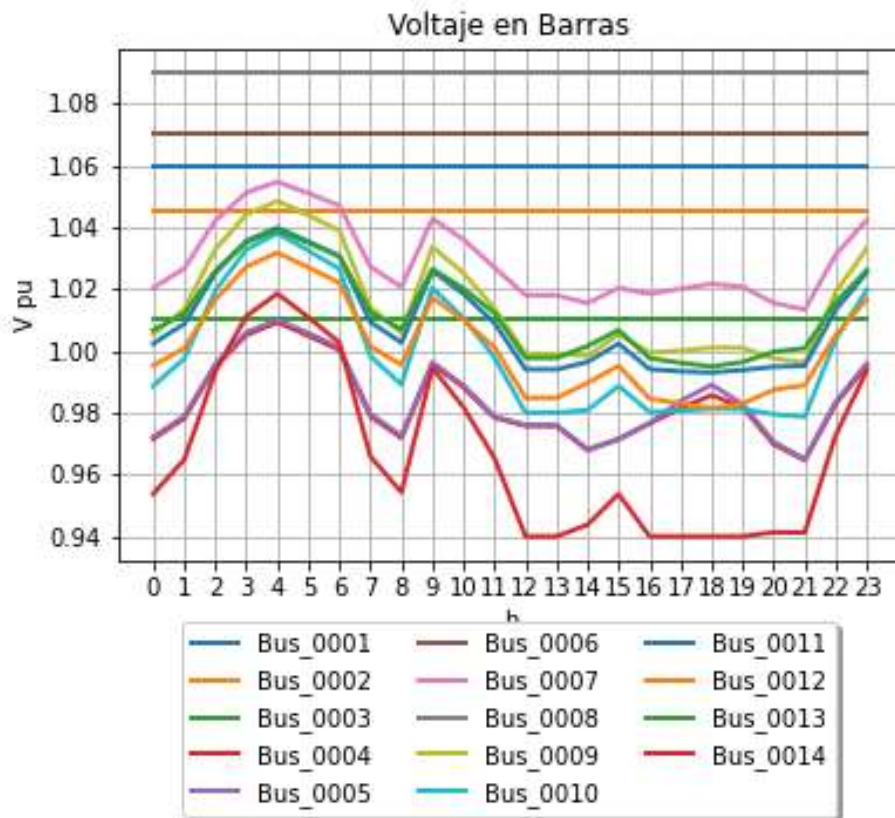


Figura 3.1.1.4. Perfil de voltaje en barras en p.u. Sistema IEEE de 14 barras

Como se visualiza en la Figura 3.1.1.4, la magnitud del voltaje en las barras cumple con los límites establecidos entre 0.94 p.u. y 1.09 p.u. La barra 14 es la que más bajo perfil de voltaje presenta en el tiempo, posiblemente se deba a que se encuentra eléctricamente alejada de los generadores. Por otro lado, la barra 8 presenta el perfil de voltaje más alto, debido a que tiene un generador conectado a ella, que realiza el control de voltaje.

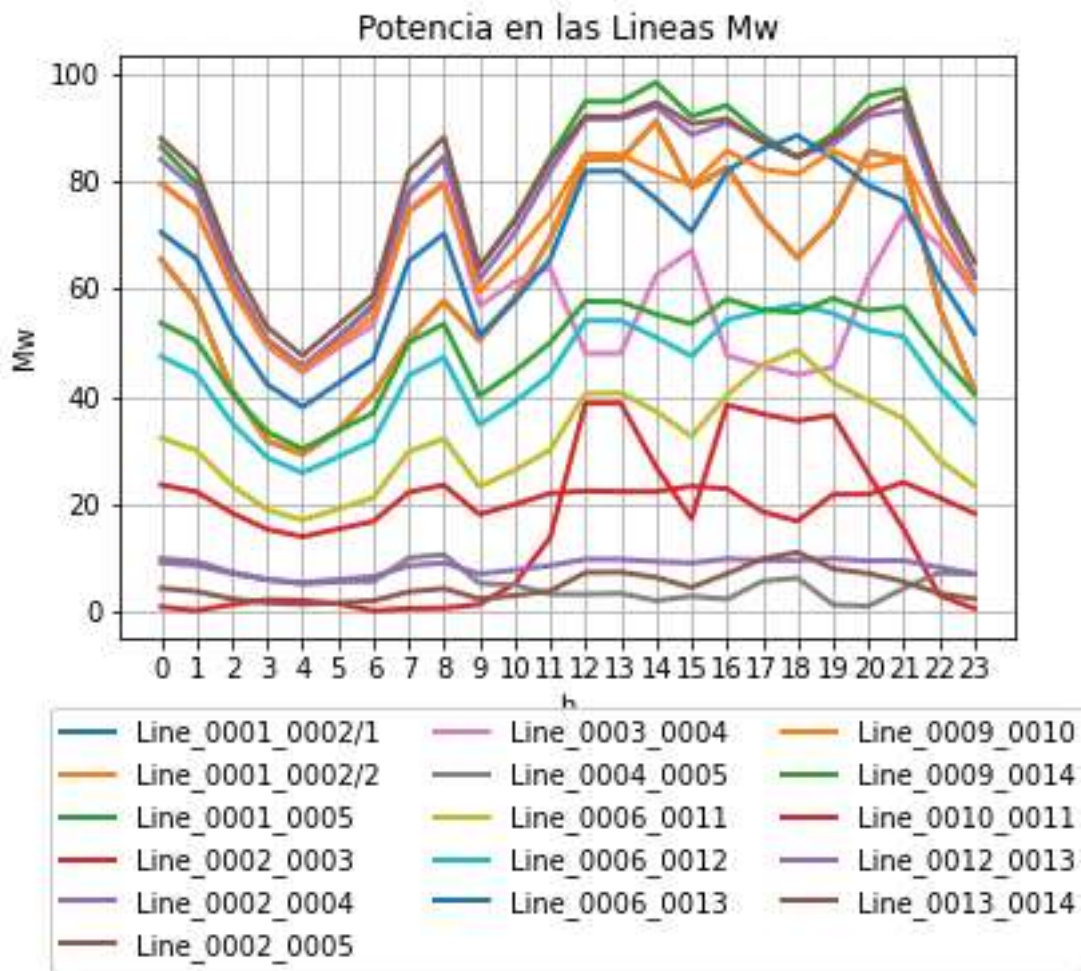


Figura 3.1.1.5. Flujo de potencia activa en líneas de transmisión. Sistema IEEE de 14 barras.

En la Figura 3.1.1.5 se muestra el flujo de potencia activa en las líneas de transmisión, en [MW], que en todos los casos cumple con los límites establecidos y la potencia en las líneas no supera los 100 [MW]. Se observa que la línea “Line_0001_0005” es la que está más sobrecargada ya que a la barra 1 se conecta el generador hidroeléctrico “GH_1” que entrega más potencia.

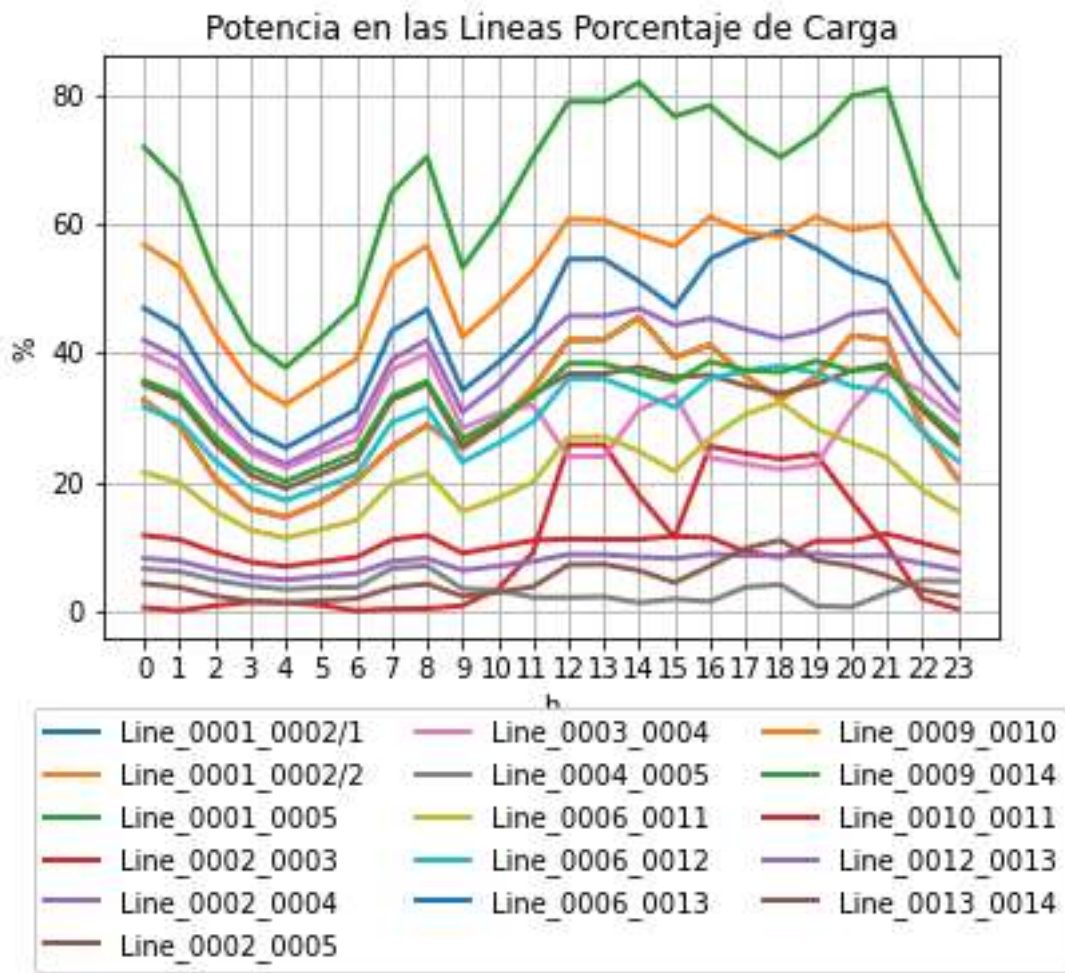


Figura 3.1.1.6. Flujo de potencia activa en las líneas de transmisión en porcentaje.
Sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.6 la línea con mayor porcentaje de cargabilidad es la línea "Line_0001_0005" y no supera el 85% de su capacidad máxima.



Figura 3.1.1.7. Flujo de potencia activa en transformadores. Sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.7 el flujo de potencia activa en transformadores, en [MW], cumple con los límites establecidos. El transformador “Trf_0005_0006” está más sobrecargado que el resto y se debe a que está conectado con el generador térmico “GT_1” en la barra 6.

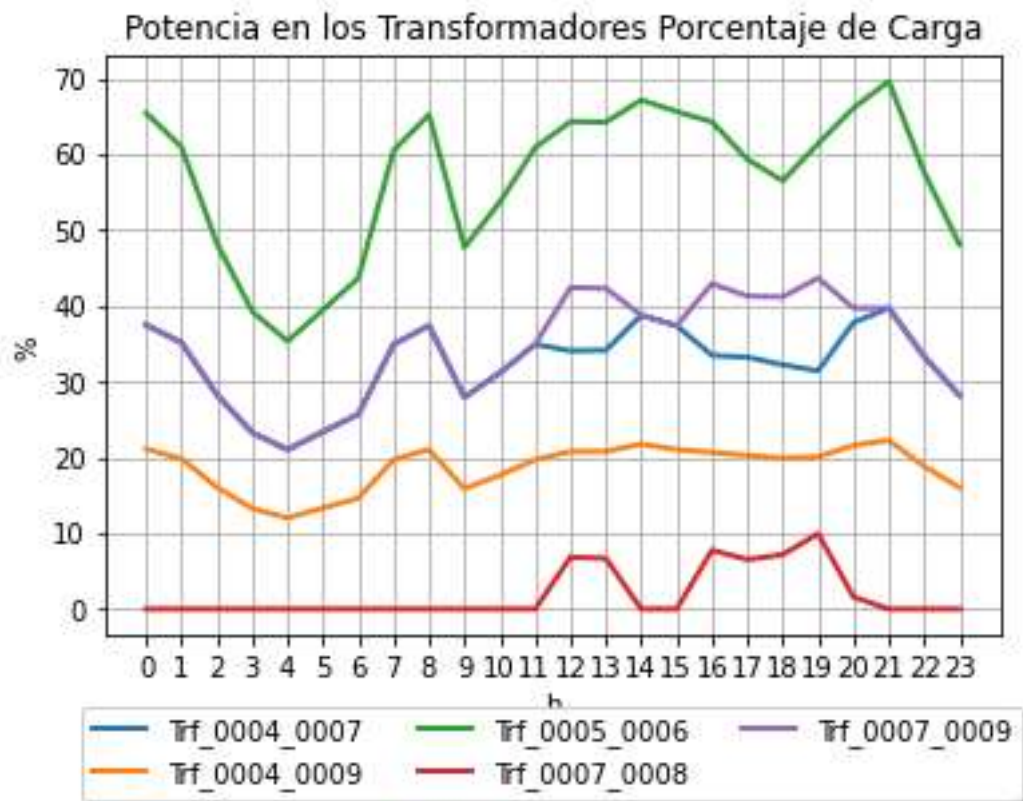


Figura 3.1.1.8. Curva del flujo de potencia activa en los transformadores en porcentaje de carga para el caso de estudio A usando el sistema IEEE de 14 barras.

En la Figura 3.1.1.8 se muestra el flujo de potencia activa en los transformadores en [%], observándose que no se supera la cargabilidad del 100 %, por tanto, se cumple con criterios de seguridad estática.

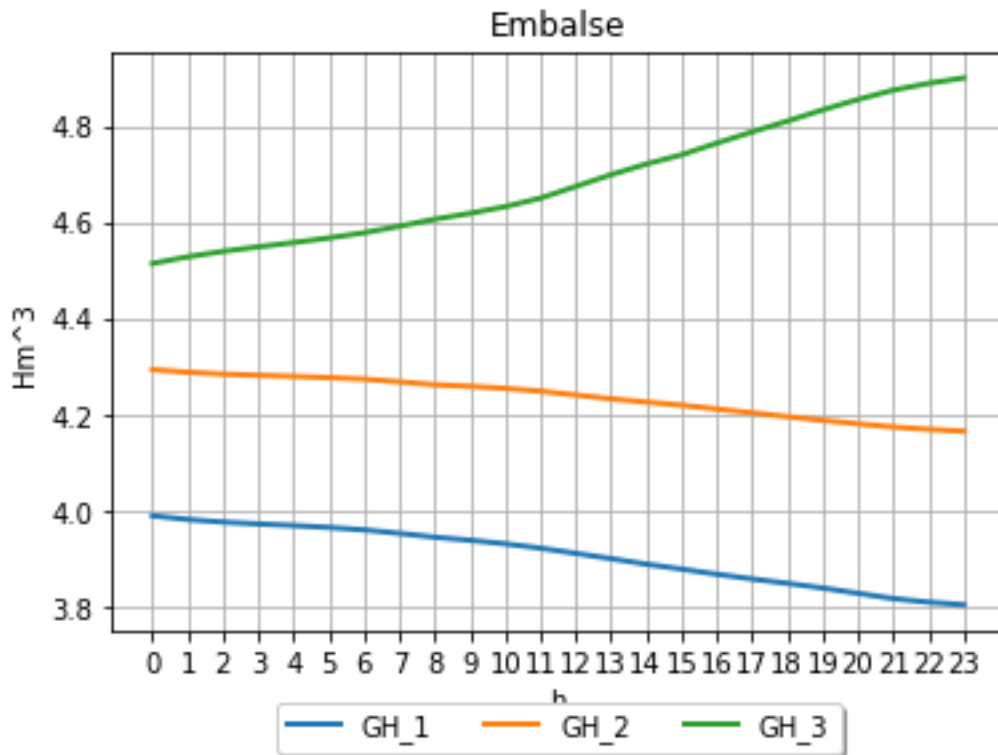


Figura 3.1.1.9. Evolución de los embalses. Sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.9 la central hidroeléctrica “GH_3”, que se encuentra en la barra 3, está ubicada aguas abajo de las dos centrales hidroeléctricas restantes “GH_1” y “GH_2”, por lo que se tiene concatenación hídrica “upstream” y su embalse aumenta en función del tiempo. La central hidroeléctrica “GH_3” almacena el agua en un embalse y tiene la capacidad de generar potencia de forma variable desde 25 [MW] hasta 110 [MW], por lo tanto, genera menos potencia que las centrales hidroeléctricas “GH_1” y “GH_2” las cuales cuentan con límites de potencia activa máxima generada con valores más elevados y por ende generan más potencia que “GH3”.

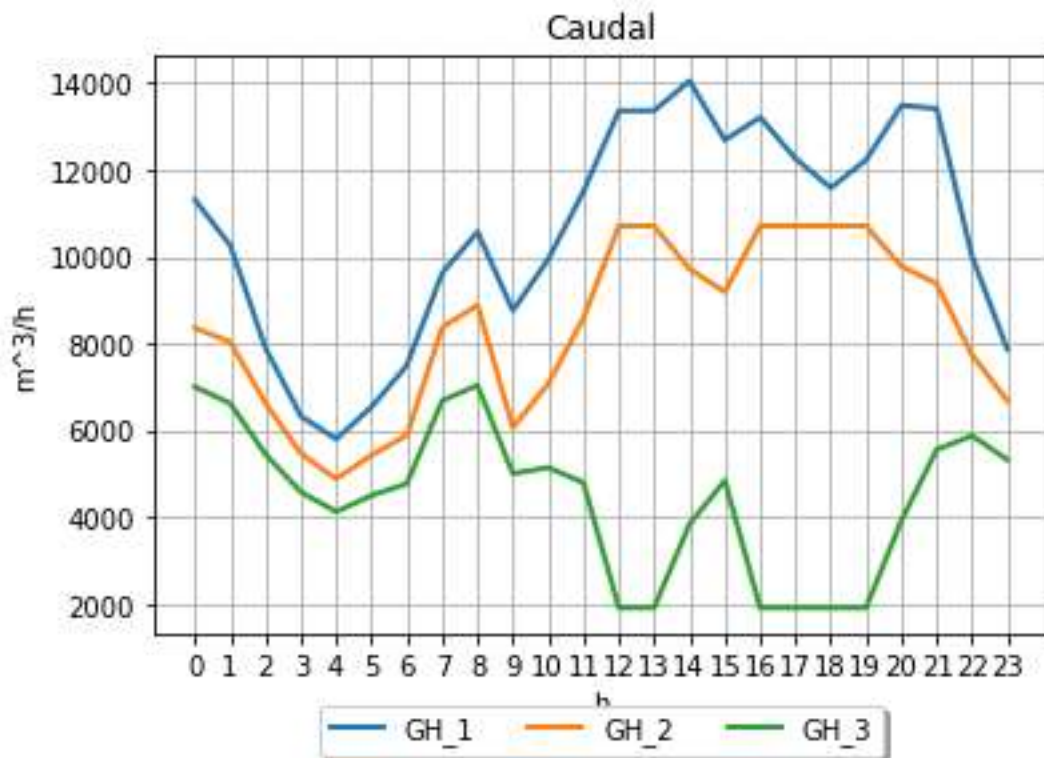


Figura 3.1.1.10. Curva del caudal para el caso de estudio A usando el sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.10 el caudal, sigue una tendencia similar a la curva de potencia generada por las centrales hidroeléctricas y se observa que el generador “GH_1” es el que más caudal turbina ya que entrega también más potencia generada.

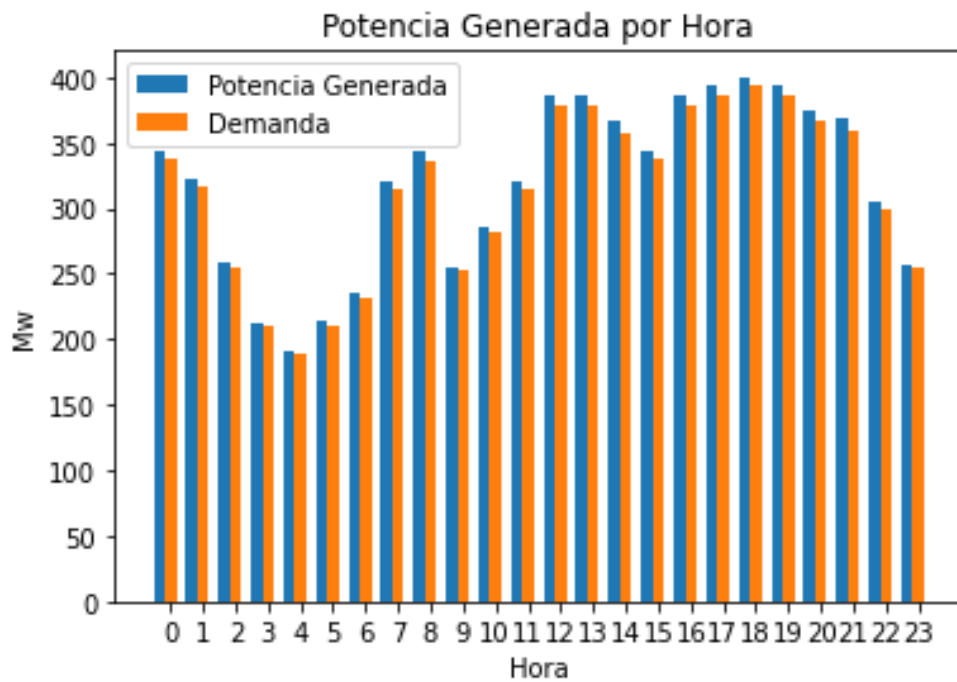


Figura 3.1.1.11. Potencia activa generada y demanda total. Sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.11, la curva de la potencia activa total generada y demanda total, al considerar pérdidas en el sistema, la potencia activa total generada es mayor a la demanda.

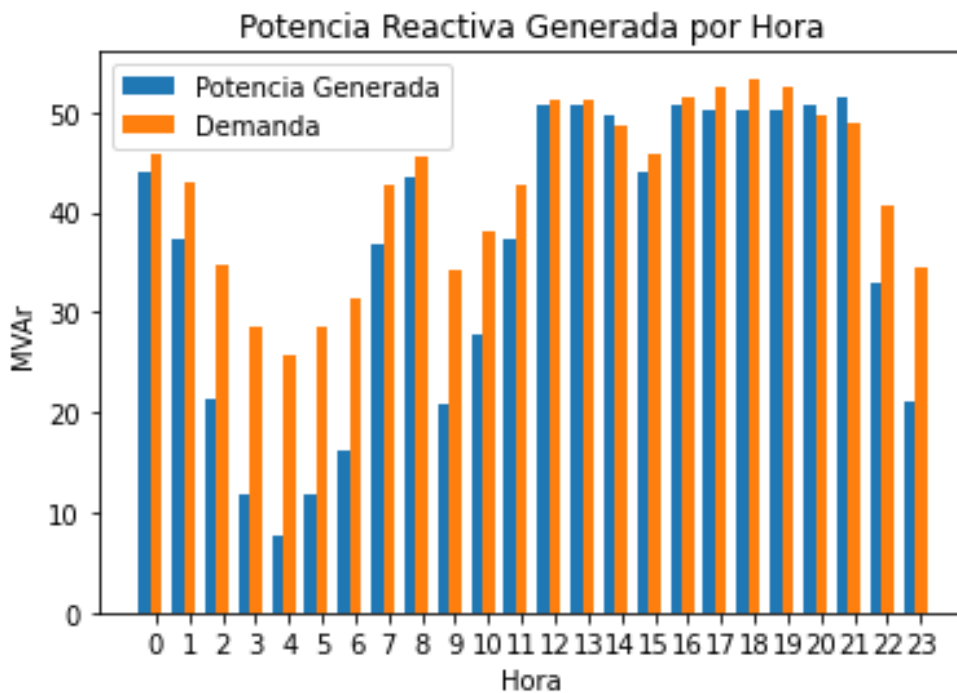


Figura 3.1.1.12. Potencia reactiva generada y demanda de potencia reactiva. Sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.12 para la curva de la potencia reactiva generada y demanda, al considerar pérdidas en el sistema la potencia reactiva generada puede ser mayor o menor a la demanda en función del tiempo, debido a que se tiene potencias reactivas capacitivas o inductivas.

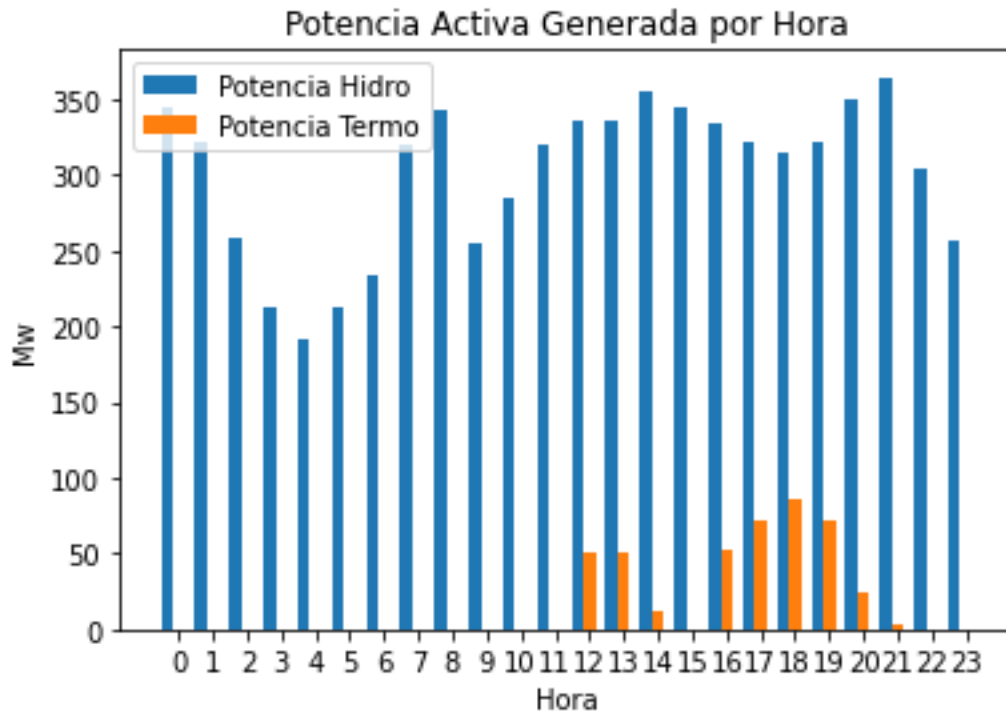


Figura 3.1.1.13. Potencia activa generada total por centrales térmicas e hidroeléctricas. Sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.13, la generación de potencia activa que entregan las centrales hidroeléctricas es mucho más grande en comparación con las centrales térmicas, que solo generan unas pocas horas, pues tiene un costo operativo mucho menor.

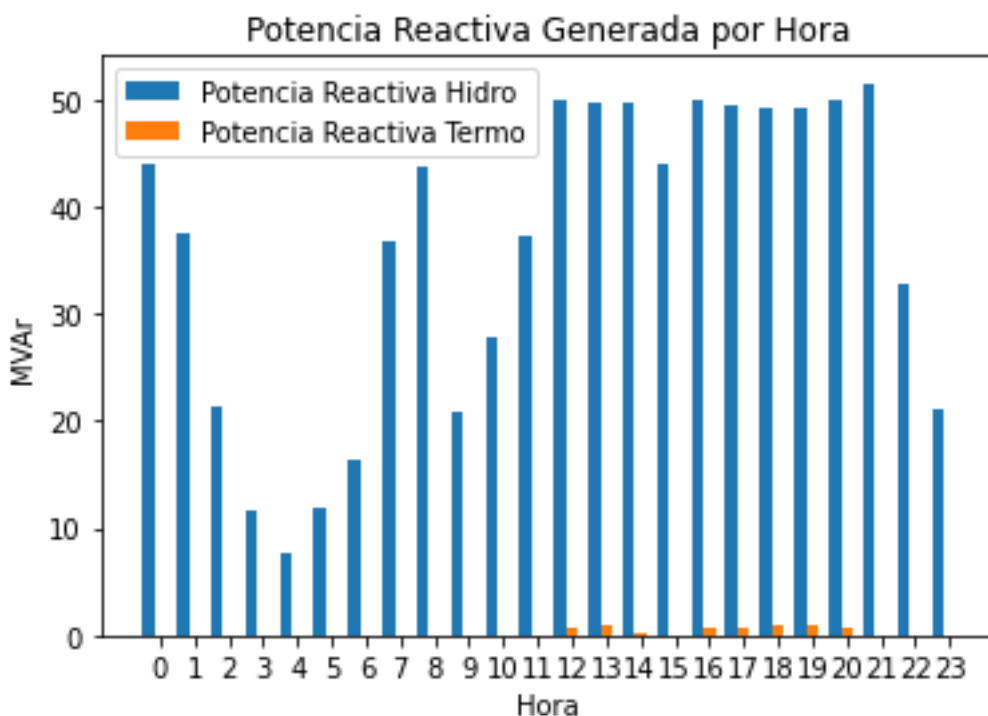


Figura 3.1.1.14. Curva de la potencia reactiva generada por las centrales térmicas e hidroeléctricas para el caso de estudio A usando el sistema IEEE de 14 barras.

Como se visualiza en la Figura 3.1.1.14, se analiza que la generación de potencia reactiva la realizan las centrales hidroeléctricas, siendo los pilares para el control de voltaje.

- **Caso de estudio B**

Tabla 2 Parametrización del modelo

Opción	Tipo
Función de costos de producción	No linealizada
Cálculo de función de producción	Con eficiencia
Solucionador	Ipopt
Cálculo con pérdidas en la red eléctrica	Con pérdidas

- **Caso de estudio C**

Tabla 3 Parametrización del modelo

Opción	Tipo
Función de costos de producción	Linealizada
Cálculo de función de producción	Con factores de descarga
Solucionador	Ipopt
Cálculo con pérdidas en la red eléctrica	Con pérdidas

- **Caso de estudio D**

Tabla 4 Parametrización del modelo

Opción	Tipo
Función de costos de producción	No linealizada
Cálculo de función de producción	Con factores de descarga
Solucionador	Ipopt
Cálculo con pérdidas en la red eléctrica	Con pérdidas

Tabla 5 Costos totales de operación para los diferentes casos de estudio

Costos totales de operación	
Caso de estudio A	\$13,406.50
Caso de estudio B	\$17,655.49
Caso de estudio C	\$13,531.85
Caso de estudio D	\$16,543.63

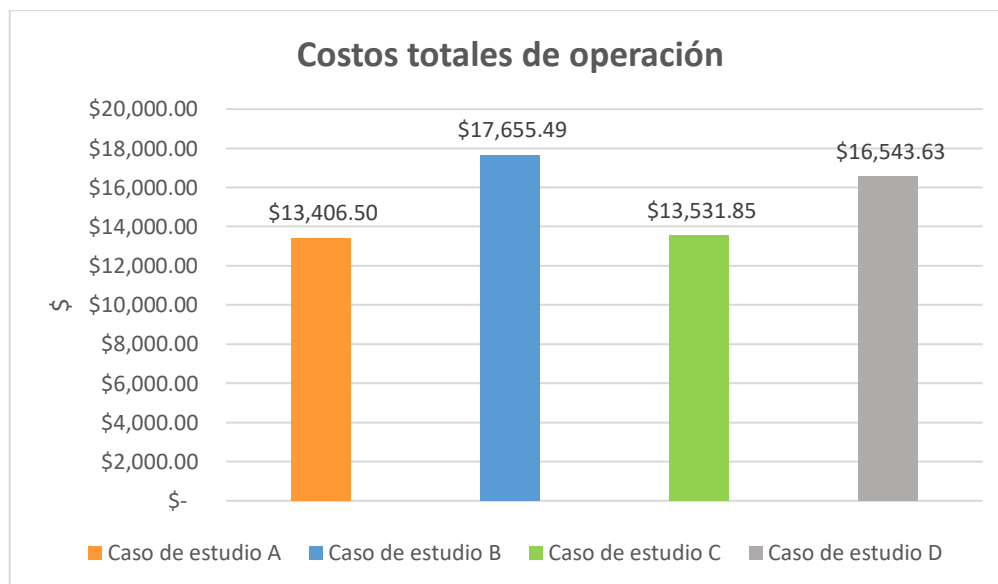


Figura 3.1.1.15. Costos totales de operación para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.1.15 y en la tabla 5, se analiza que para los cuatro casos se tienen costos totales similares de operación, por lo que la herramienta computacional funciona correctamente. Además, para los casos de estudio A y C se tienen valores muy cercanos ya que se usa una función de costos linealizada para la minimización del problema; mientras que para los casos de estudios B y D se tienen valores cercanos ya que se usa una función de costos no linealizada para la minimización del problema de optimización. Se analiza que el caso B, es el que mayor costo total de operación genera en comparación a los otros casos, debido a los valores de la eficiencia de las centrales hidroeléctricas en los datos de entrada del Excel y al estado de operación de las variables binarias de las centrales térmicas.

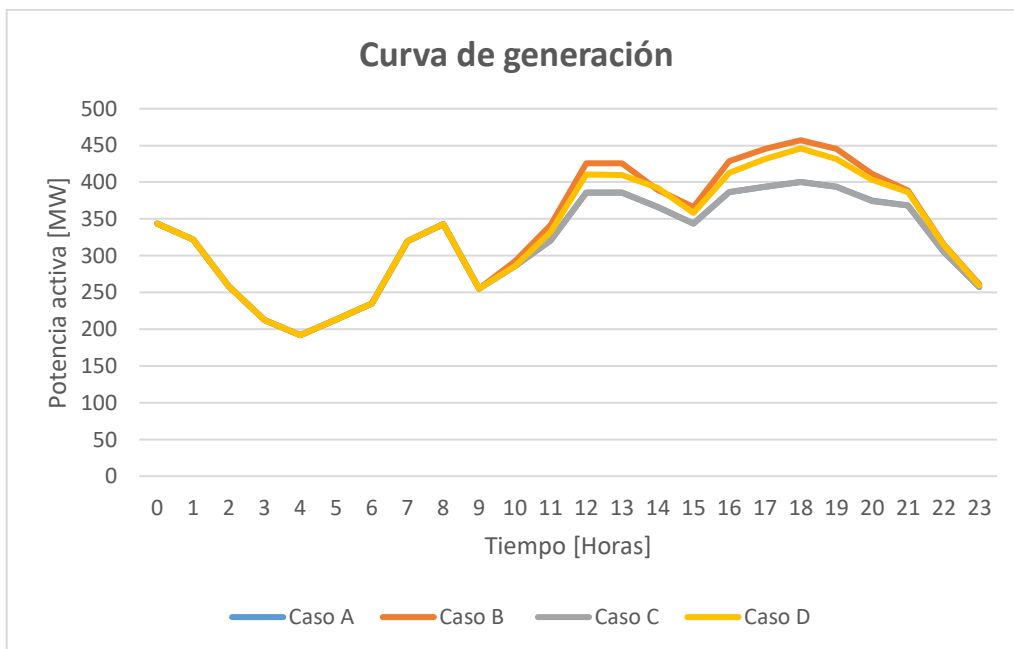


Figura 3.1.1.16. Curva de generación para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.1.16 y en el Anexo IV, se analiza que para los cuatro casos de estudio se tienen perfiles de generación muy similares en función del tiempo ya que la demanda para los cuatro casos es muy parecida, por lo que se comprueba que la herramienta computacional funciona de manera correcta. Se concluye que los perfiles de generación para todo el día del caso de estudio A con un valor de 7662.98 [MW] y del caso C con un valor de 7662.185 [MW] son casi iguales, por lo que se comprueba el uso de una función de costos linealizada para minimizar costos del problema considerando pérdidas de la red eléctrica, además de usar para el cálculo de la función de producción de las centrales hidroeléctricas mediante la eficiencia para el caso A y de factores de descarga para el caso C. De igual forma, se tiene perfiles de generación muy similares para el caso de estudio B con un valor de 8089.3019 [MW] y el caso D con un valor de 7968.5497 [MW], pero con el uso de una función de costos no linealizada para la minimización de costos del problema de optimización, además se analiza que en las horas que se tiene menor demanda son casi iguales. Se tiene mayor potencia activa generada en el caso B con un gran aporte al día de las centrales hidroeléctricas de 93.12 % y un aporte de las centrales térmicas de 6.88%, cuando se usa una función de costos no linealizada y con la eficiencia, pues se tiene un costo más alto ya que se usa más generación térmica en este caso. Finalmente, para los cuatro casos de estudio en la hora 18 se tiene la máxima generación ya que se debe suplir la demanda máxima en esa hora pico.

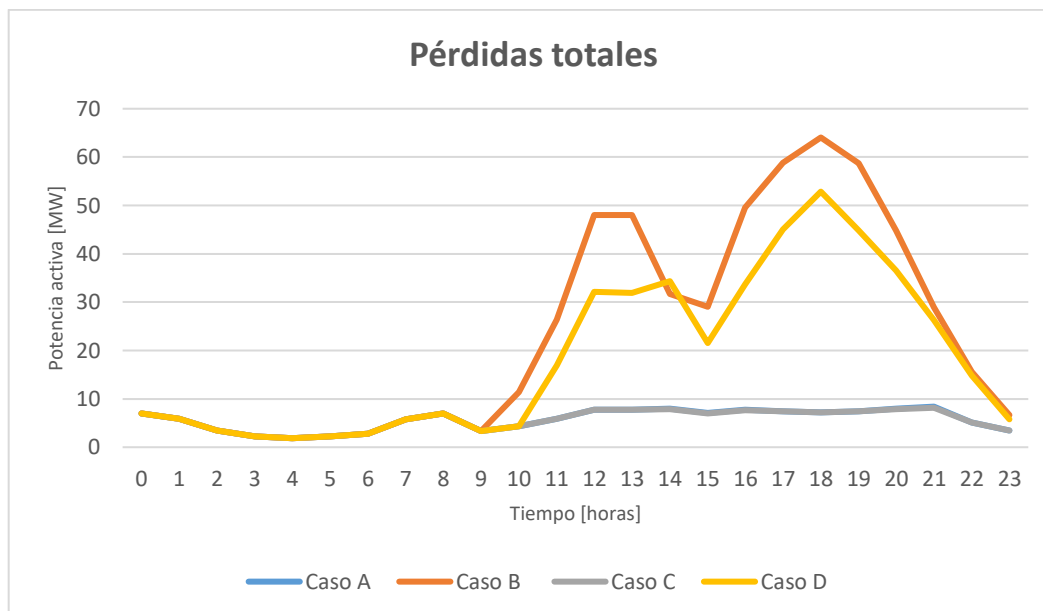


Figura 3.1.1.17. Pérdidas totales para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.1.17 y en el Anexo IV, se analiza que para el caso A con un valor de 137.143 [MW] y el caso C con un valor de 136.348 [MW] se tiene casi las mismas pérdidas cuando se usa la función de costos linealizada. Para el caso de estudios B con un valor de 563.465 [MW] y el caso D con un valor de 442.713 [MW] se tienen pérdidas en la red eléctrica cercanas ya que se usa una función de costos no linealizada para la minimización del problema de optimización y su diferencia es tolerable. Además, en las horas con baja demanda se obtiene pérdidas en el sistema eléctrico casi iguales para los cuatro casos, pero cuando se tiene mayor demanda varían su tendencia ya que se tiene un gran aumento de pérdidas en el sistema eléctrico. Para el caso B, se tiene las mayores pérdidas donde se usa la función de costos no linealizada, ya que se debe generar más potencia para cubrir la demanda y por ende se genera más pérdidas en la red, pues se comprueba que se tiene costos de producción totales más elevados en este caso debido a que se tiene más pérdidas.

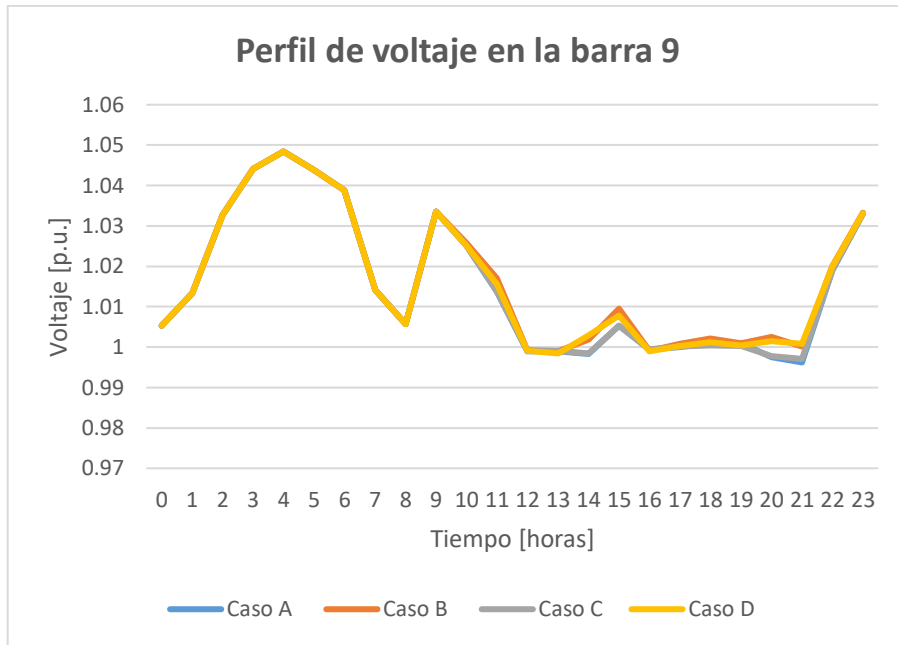


Figura 3.1.1.18. Perfil de voltaje para los diferentes casos de estudio en la barra 9.

Como se visualiza en la Figura 3.1.1.18 y en el Anexo IV, se analiza que los valores de los voltajes en p.u. en la barra 9 para el sistema IEEE de 14 barras son casi iguales en los cuatro casos con muy pequeñas diferencias en las horas pico de demanda desde la hora 18 hasta la hora 21 debido a que tiene dos transformadores y una carga conectada a ella, para realizar el control de voltaje. Para el caso A, en la hora 21 de demanda máxima se tiene un nivel de voltaje mínimo de 0.9962 p.u., pues al ser una barra de carga el perfil de voltaje cambia con su demanda.

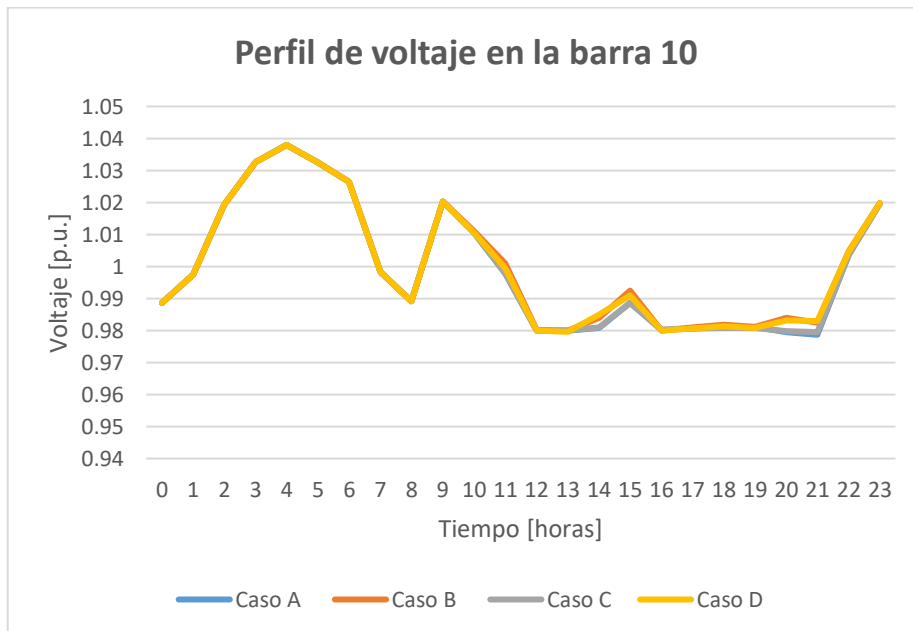


Figura 3.1.1.19. Perfil de voltaje para los diferentes casos de estudio en la barra 10.

Como se visualiza en la Figura 3.1.1.19 y en el Anexo IV, se analiza que los valores de los voltajes en p.u. en la barra 10 del sistema IEEE de 14 barras son muy similares para los cuatro casos de estudio, donde se tiene pequeñas diferencias entre las 13 y 15 horas, y entre las 19 y 21 horas debido a que tiene una carga grande conectada a ella, para realizar el control de voltaje. Se analiza que para las horas de demandas pico se tiene niveles de voltajes mínimos y para las horas de demandas bajas se tiene niveles de voltajes mayores, pues es una barra de carga. Finalmente, para el caso A, en la hora 21 de demanda máxima se tiene un nivel de voltaje mínimo de 0.9786 p.u.

Tabla 6 Tiempos de ejecución para los diferentes casos de estudio

Tiempos de ejecución [s]	
Caso de estudio A	27.25786
Caso de estudio B	18.79152
Caso de estudio C	21.29868
Caso de estudio D	20.25944

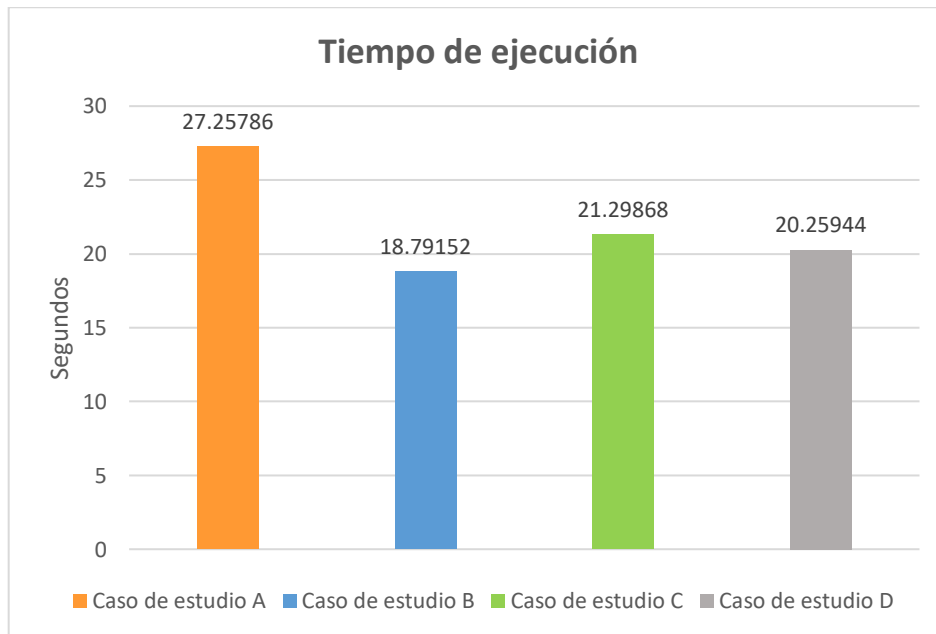


Figura 3.1.1.20. Tiempos de ejecución para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.1.20 y en la Tabla 6, se analiza que el tiempo de ejecución más alto es el caso A debido a que se tiene un mayor uso de memoria del computador al crear la Y de barra por primera vez para el caso del sistema IEEE de 14 barras ya que en los otros casos ya se carga el archivo Y de barra calculado previamente, además en este método se usa la función de costos linealizada para una mejor

aproximación del problema de optimización y el método de la eficiencia. Para los casos B, C y D se tienen tiempos más bajos y muy similares con una pequeña diferencia de centésimas de segundos, por lo que se comprueba que el programa está bien implementado. Finalmente, el caso B es el más rápido cuando se usa la función de costos no linealizada y el método de la eficiencia ya que se tiene menos aproximaciones en los cálculos internos del programa desarrollado.

3.1.2 Aplicación al Sistema Nacional Interconectado (SNI) ecuatoriano

La herramienta computacional se aplica al SNI ecuatoriano que cuenta con varias centrales hidroeléctricas que suministran la mayor parte de energía eléctrica. Se modelan 69 generadores, 355 barras, 89 cargas, 213 líneas de transmisión, 221 transformadores, etc. La gran cantidad de elementos eléctricos de la red dificulta la resolución del DHT-CP.

- **Caso de estudio A**

Tabla 7 Parametrización del modelo

Opción	Tipo
Función de costos de producción	Linealizada
Cálculo de función de producción	Con eficiencia
Temporada	Lluviosa
Solucionador	Ipopt
Cálculo con pérdidas en la red eléctrica	Con pérdidas

El caso de temporada lluviosa se caracteriza porque se cuenta con un mayor caudal de los ríos, por lo que los embalses tienen mayor volumen de agua y las unidades hidroeléctricas entregan una gran cantidad de energía; se cubre con la demanda sin ningún problema.

Los resultados que se obtienen son los siguientes:

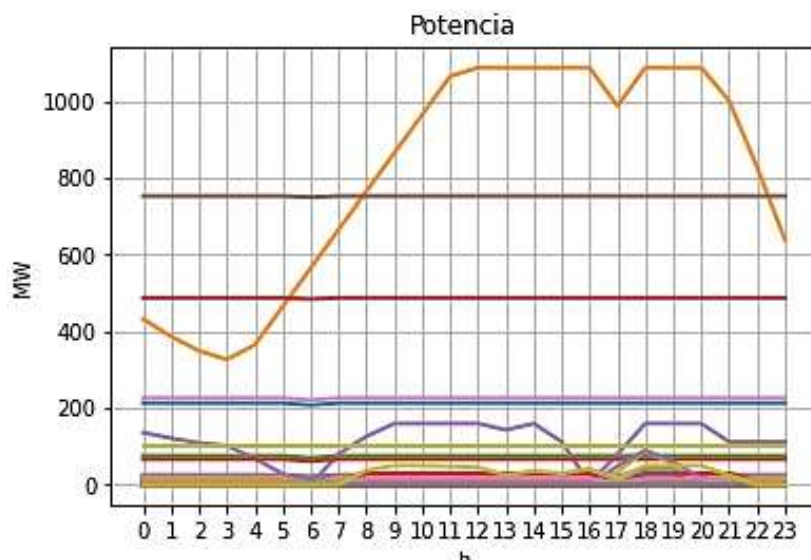


Figura 3.1.2.1. Potencia activa generada. SNI temporada lluviosa.

Como se visualiza en la Figura 3.1.2.1, la mayoría de la demanda es suministrada por las centrales hidroeléctricas Paute, Sopladora y Coca Codo Sinclair, además de las otras centrales hidroeléctricas de menor envergadura. La potencia que entrega la central hidroeléctrica Coca Codo Sinclair es casi constante en el tiempo de análisis ya que es una central de pasada que utiliza al máximo el caudal natural del río Coca. La central no genera su potencia máxima debido a que varias unidades se encuentran en mantenimiento programado, por lo que no se encuentran disponibles.

La central Paute, debido a que tiene embalse, pueden suministrar potencia de forma variable desde 326.69 [MW], en horas de valle, hasta llegar a 1087.59 [MW] cuando se tiene el pico de demanda.

Las centrales hidroeléctricas Paute, Sopladora y Mazar se encuentran en la misma cuenca del río Paute, por lo que se tiene concatenación hídrica y sus despachos se relacionan entre sí, por consiguiente, si la central hidroeléctrica Mazar genera más potencia, también lo harán las centrales hidroeléctricas Paute y Sopladora.

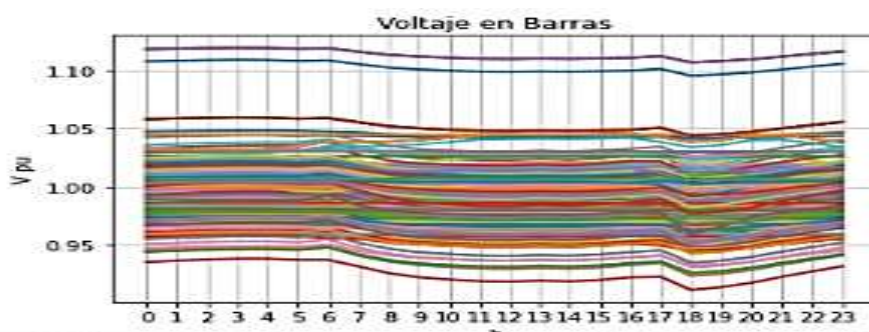


Figura 3.1.2.2. Perfil de voltaje en las barras en p.u. SNI temporada lluviosa.

Como se visualiza en la Figura 3.1.2.2, el perfil de voltaje, en p.u., se encuentra entre 0.90 p.u. y 1.10713 p.u. Las barras 151, 152 y 153 alcanzan el valor máximo de voltaje, en la zona Santa Rosa – Totoras, estas operan a un nivel de voltaje de 500 kV. Se analiza, que la barra 88 alcanza el valor más bajo de voltaje, en la zona de Pascuales.

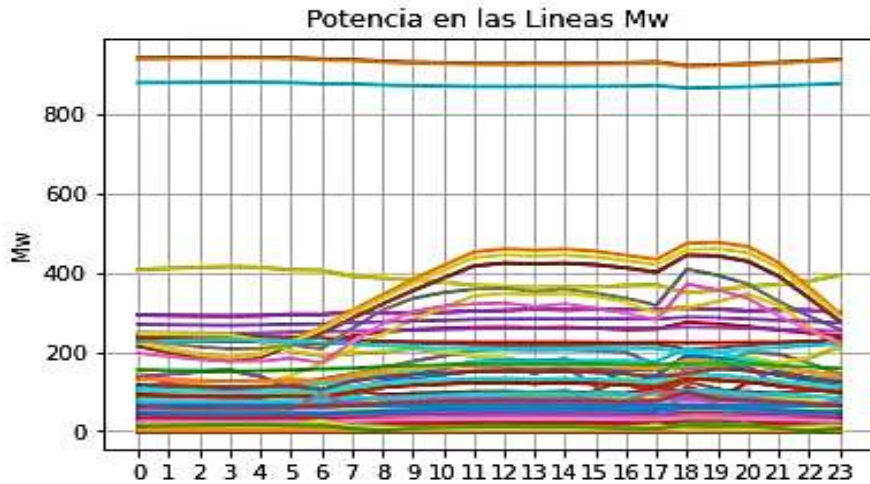


Figura 3.1.2.3. Flujo de potencia activa en líneas de transmisión. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.3, la línea “L_E005_SIDE_1_1” en la zona Molino – Milagro es la que mayor cargabilidad presenta, para evitar este inconveniente se debe redistribuir de mejor forma los flujos de potencia en esta zona.

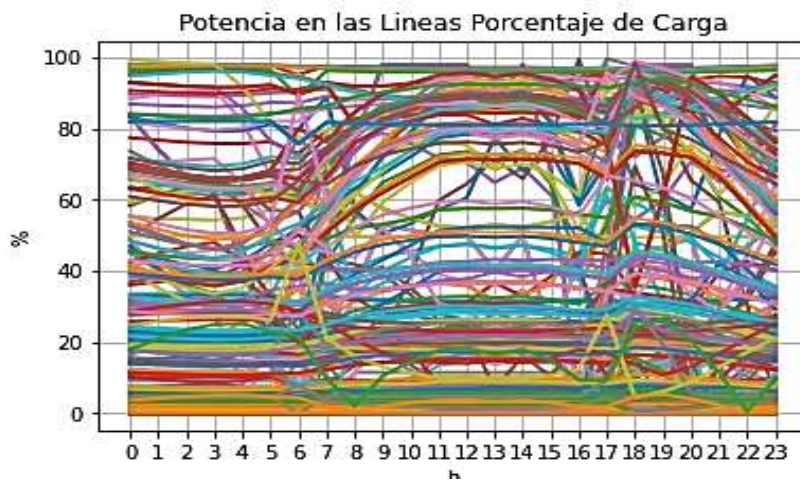


Figura 3.1.2.4. Flujo de potencia activa en líneas de transmisión, en porcentaje. SNI ecuatoriano.

Lo antes mencionado puede confirmarse en la Figura 3.1.2.4 donde se observa que la línea “L_E005_SIDE_1_1” en la zona Molino – Milagro, que está cerca del 100% de cargabilidad.

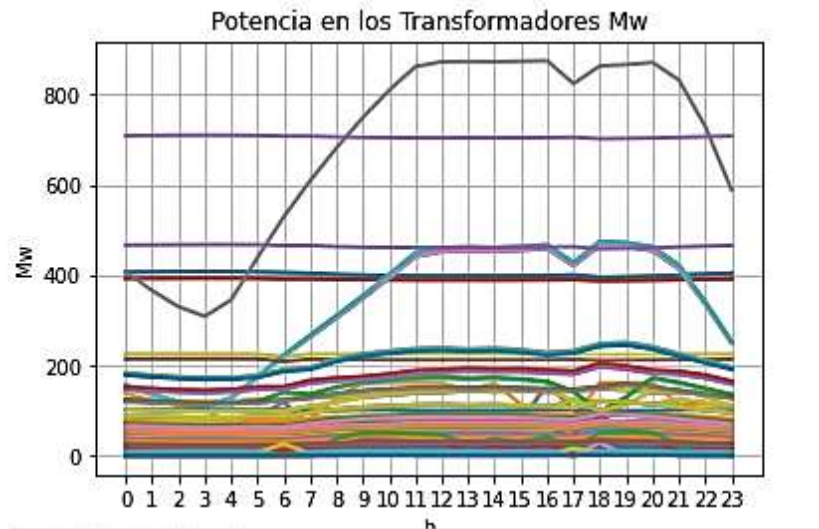


Figura 3.1.2.5. Flujo de potencia activa en transformadores. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.5 el transformador “T_INGA_ATJ_iv” está más sobrecargado que el resto en la zona de Quito, cumple con los límites establecidos.



Figura 3.1.2.6. Flujo de potencia activa en los transformadores, en porcentaje. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.6 se cumple con los límites de cargabilidad, además se observa que el transformador “T_INGA_ATJ_iv” es el más sobrecargado en la zona de Quito, pero no supera el 100 % cumpliendo con criterios de seguridad estática.

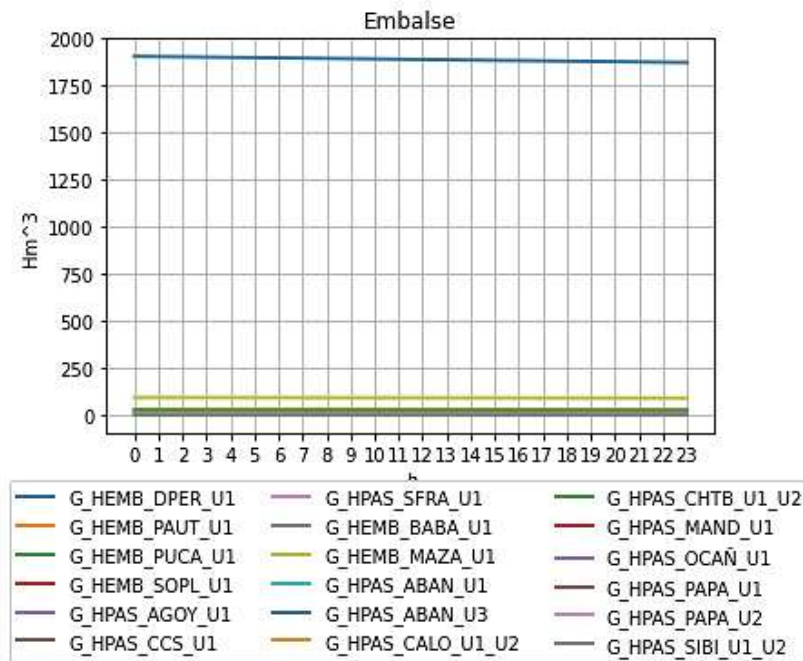


Figura 3.1.2.7. Evolución de embalses. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.7, el embalse de la central hidroeléctrica Daule - Peripa “G_HEMB_DPER_U1” tiene mayor volumen y se utiliza al máximo el caudal natural del río Daule para la zona de Guayaquil. Además, la central hidroeléctrica de pasada Agoyán “G_HPAS_AGOY_U1” tiene menor volumen.

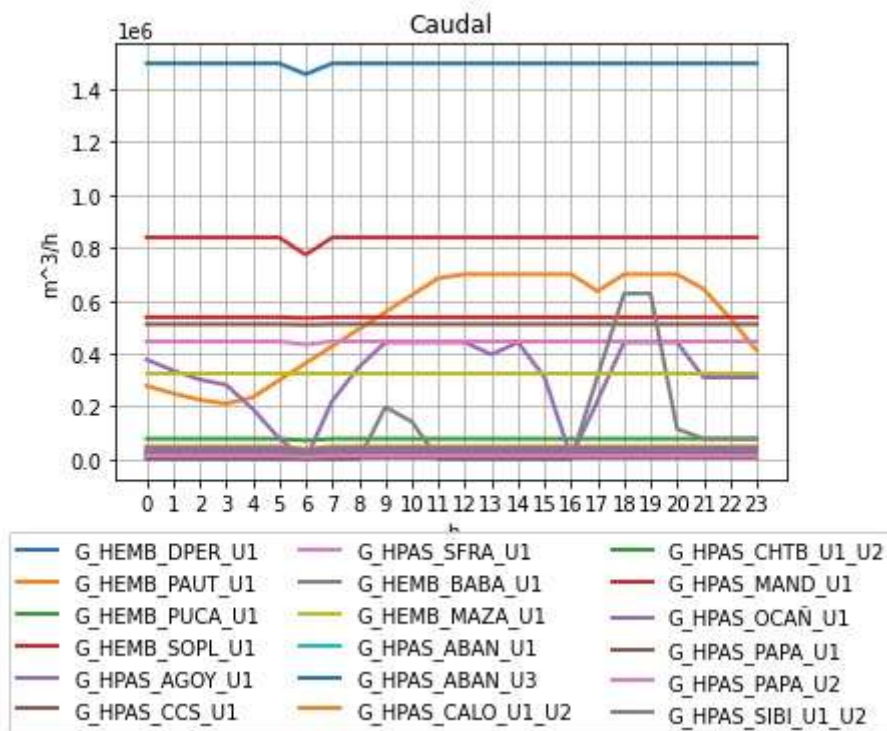


Figura 3.1.2.8. Curva del caudal. SNI ecuatoriano.

Lo antes mencionado puede confirmarse en la Figura 3.1.2.8 donde para la central hidroeléctrica Daule - Peripa “G_HEMB_DPER_U1” se utiliza el caudal máximo natural del río Daule con un valor aproximado de $1497717.964 \text{ [m}^3/h\text{]}$ para abastecer de energía eléctrica a la zona de Guayaquil.

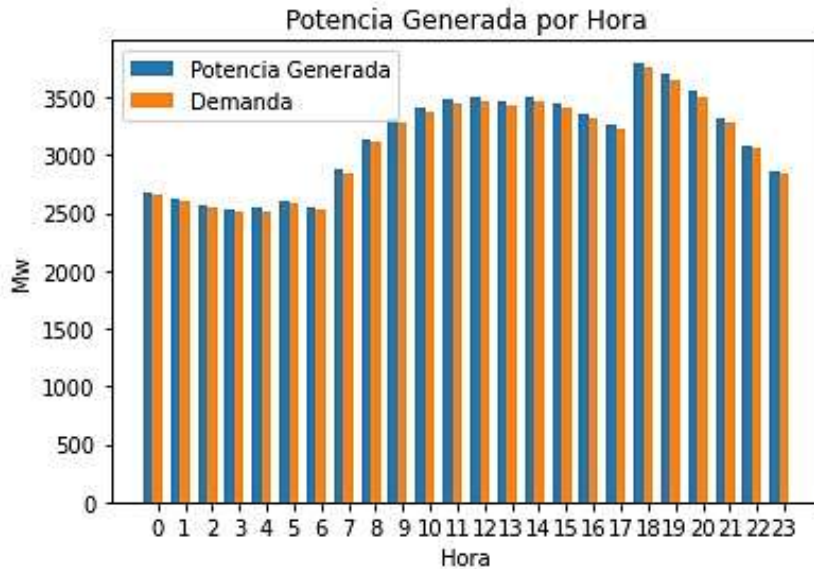


Figura 3.1.2.9. Potencia activa generada y demanda total. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.9 cuando se tiene pérdidas en la red eléctrica, la potencia activa generada es mayor a la demanda. Se tiene valores más elevados de potencia generada en las horas pico de demanda desde las 18 hasta las 20 horas, para suplir la demanda máxima de 3753.506 [MW] a la hora 18 en la generación hidráulica se tiene un valor de 3325.88 [MW] y en la generación térmica se tiene un valor de 475.775 [MW] .

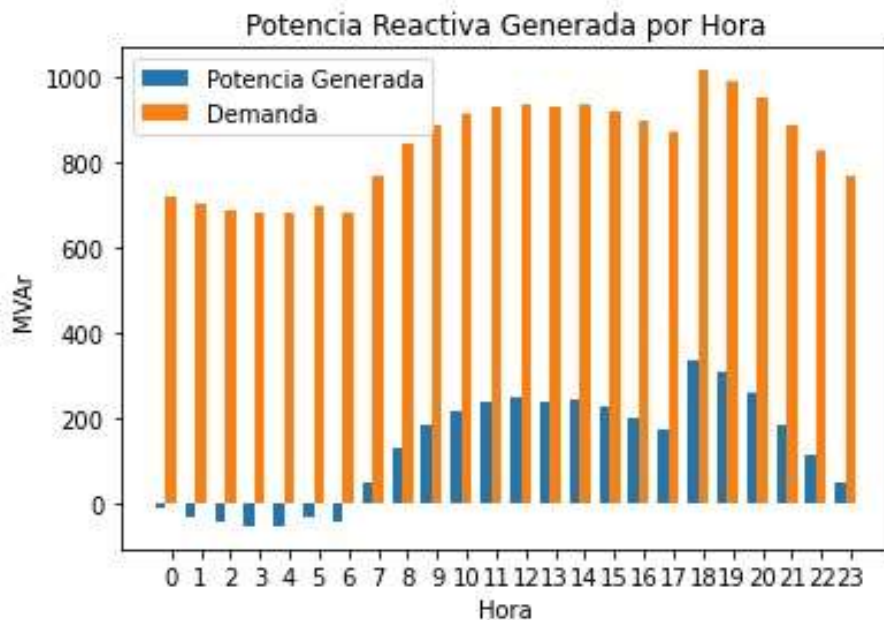


Figura 3.1.2.10. Potencia reactiva generada y demanda total. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.10 al considerar pérdidas en la red eléctrica, la potencia reactiva generada es mayor a la demanda. Se tiene valores más elevados de potencia reactiva generada en las horas pico de demanda desde las 18 hasta las 20 horas, con un valor máximo de generación de 336.33 [MVAR] en la hora 18. Se analiza que desde las 0 hasta las 6 horas se tiene valores negativos ya que se absorbe potencia reactiva.

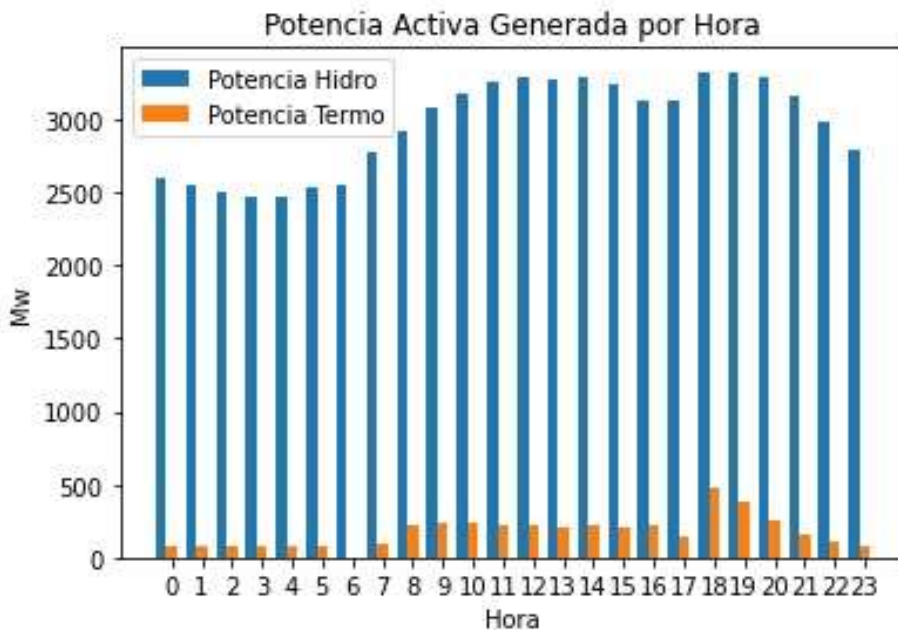


Figura 3.1.2.11. Potencia activa generada por centrales térmicas e hidroeléctricas. SNI ecuatoriano.

Como se visualiza en la Figura 3.1.2.11, la generación de potencia activa que entregan las centrales hidroeléctricas en el día es más grande representan el 94.528 % del total de su generación en comparación a las centrales térmicas que al día representan el 5.472 %, pues se tiene un costo operativo muy bajo para la temporada lluviosa.

- **Caso de estudio B**

Tabla 8 Parametrización del modelo

Opción	Tipo
Función de costos de producción	Linealizada
Cálculo de función de producción	Con eficiencia
Temporada	Seca
Solucionador	Ipopt
Cálculo con pérdidas en la red eléctrica	Con pérdidas

Tabla 9 Costos totales de operación para los diferentes casos de estudio

Costos totales de operación	
Caso de estudio A	\$10,213.22
Caso de estudio B	\$68,103.62

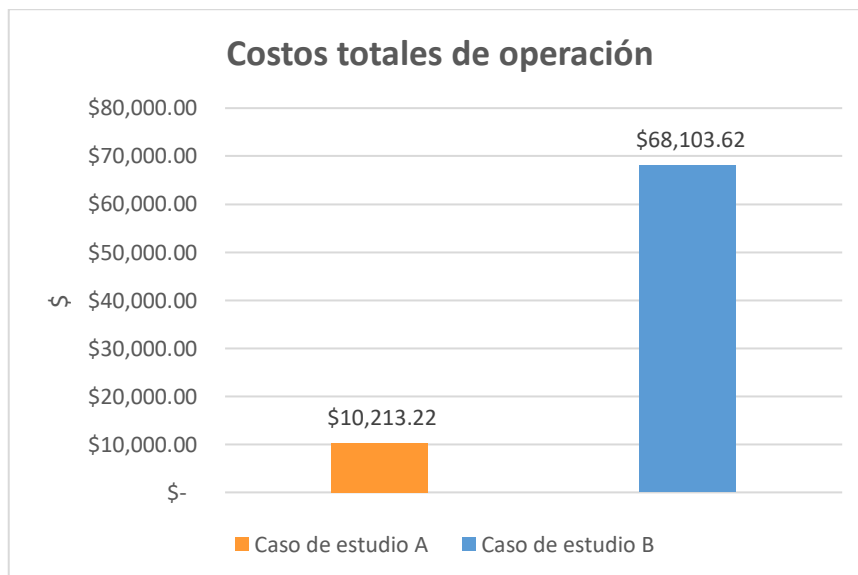


Figura 3.1.2.12. Costos totales de operación para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.2.12 y en la tabla 9, se analiza que para los dos casos no se tienen costos totales similares de operación para el SNI ecuatoriano, se concluye que para el caso A que corresponde a la temporada lluviosa se tiene un costo operativo más bajo ya que la mayoría de demanda se suple con generación hidráulica. Se analiza que el caso B que corresponde a la temporada seca, es mayor alrededor de 7 veces al

costo total de operación del caso A, debido a que se usa más generación térmica para suplir la demanda, pues se tiene un mayor costo total debido al cambio en las variables binarias que se asocian al estado de operación de las centrales térmicas.

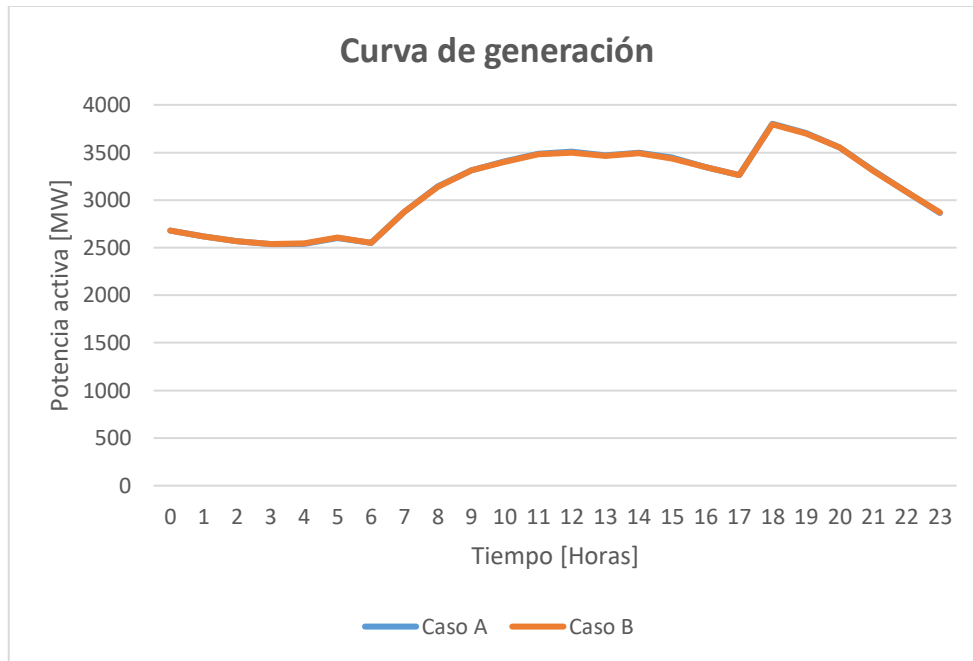


Figura 3.1.2.13. Curva de generación para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.2.13 y en el Anexo V, se analiza que para los dos casos de estudio se tienen perfiles de generación muy similares en función del tiempo ya que la demanda total es muy similar, para el caso A se tiene un valor de 74382.76 [MW] y para el caso B se tiene un valor de 74339.65 [MW], además se analiza que el caso A tiene mayor potencia activa generada por hora al día. Se analiza que para el caso A de la temporada lluviosa, la generación de potencia activa total al día es de 75178.90 [MW], el porcentaje total de cobertura al día de las centrales hidroeléctricas es del 94.528 % con un valor de 71065.09 [MW] y el aporte de las centrales térmicas es de 5.472 % con un valor de 4113.81 [MW], pues se tiene un costo operativo mucho menor. Se analiza que para el caso B de la temporada seca, la generación de potencia activa total al día es de 75131.90 [MW], el porcentaje total de cobertura al día de las centrales hidroeléctricas es del 81.327 % con un valor de 61102.73 [MW] y el aporte de las centrales térmicas es de 18.673 % con un valor de 14029.17 [MW], pues se tiene un costo operativo más grande. Finalmente, para los dos casos de estudio en la hora 18 se tiene la máxima generación ya que se debe suplir la demanda máxima en esa hora pico.

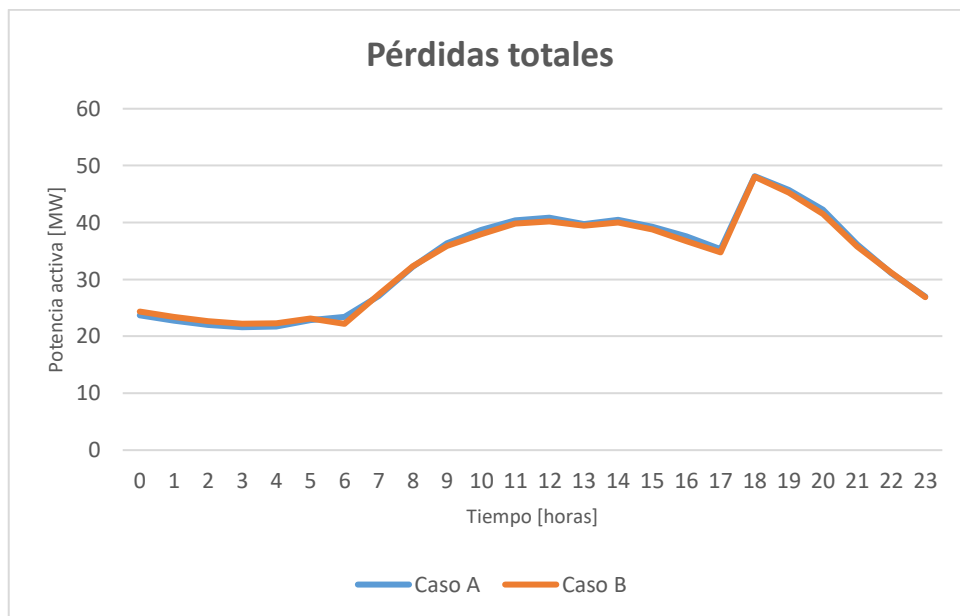


Figura 3.1.2.14. Pérdidas totales para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.2.14 y en el Anexo V, se analiza que para el caso A y B se tienen pérdidas similares cuando se usa la función de costos linealizada, en donde para el caso A se tiene un total de pérdidas al día de 796.14 [MW] y para el caso B se tiene un valor de 792.25 [MW], el perfil de pérdidas totales de ambos casos sigue una misma tendencia, entre las 18 y 21 horas se tiene las máximas pérdidas ya que se debe generar más potencia para suplir la demanda pico. Se concluye que para el caso A en la temporada lluviosa, se tiene mayores pérdidas ya que se tiene mayor generación de potencia activa para suplir una demanda un poco más grande en comparación al caso B.

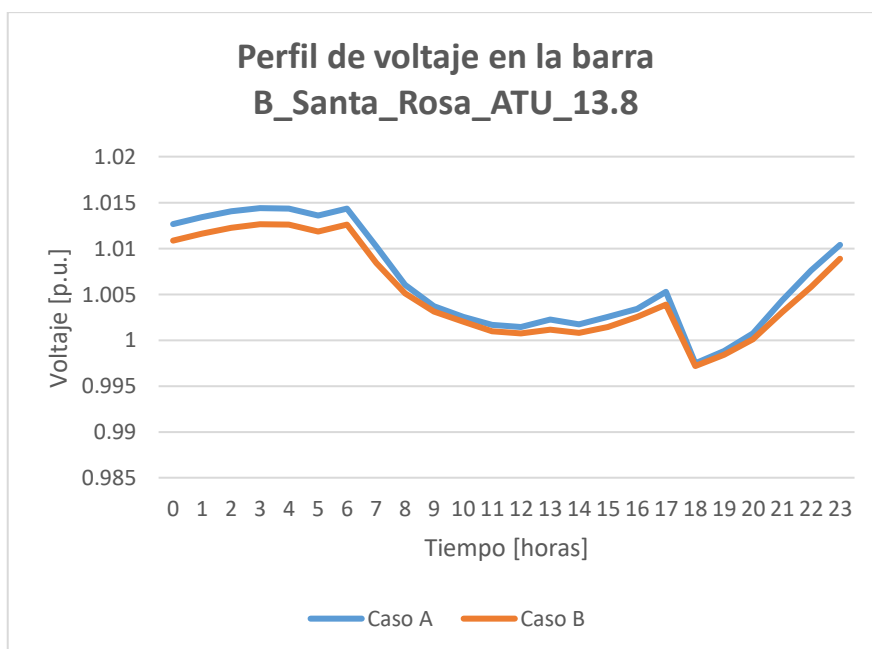


Figura 3.1.2.15. Perfil de voltaje para los diferentes casos de estudio en la barra “B_Santa_Rosa_ATU_13.8”.

Como se visualiza en la Figura 3.1.2.15 y en el Anexo V, se analiza que los valores de los voltajes en p.u. en la barra “B_Santa_Rosa_ATU_13.8” para el SNI ecuatoriano son muy similares en los dos casos con muy pequeñas diferencias en función del tiempo para la zona de Santa Rosa – Totoras, una carga y un transformador se conecta a esta barra para realizar el control de voltaje. Como es una barra de carga PQ el perfil de voltaje cambia con su demanda, para la hora 18 de demanda máxima se tiene un nivel de voltaje mínimo, donde para el caso A se tiene un valor de 0.9974681 p.u. y para el caso B un valor de 0.9971732 p.u., pues se tiene un valor más bajo en la temporada seca y se cumple con los límites establecidos para el límite de magnitud de voltaje.

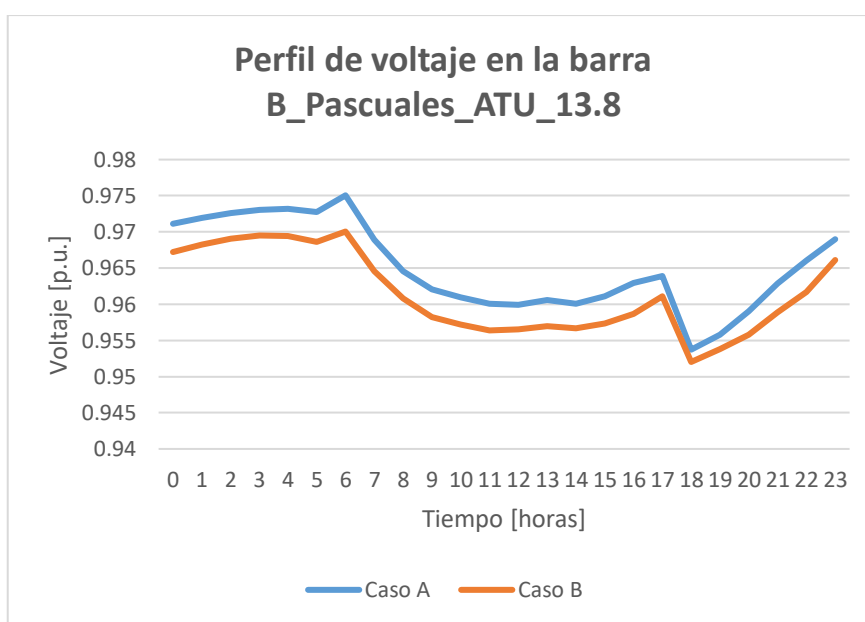


Figura 3.1.2.16. Perfil de voltaje para los diferentes casos de estudio en la barra “B_Pascuales_ATU_13.8”.

Como se visualiza en la Figura 3.1.2.16 y en el Anexo V, se analiza que los valores de los voltajes en p.u. en la barra “B_Pascuales_ATU_13.8” para el SNI ecuatoriano son similares en los dos casos con muy pequeñas diferencias en el análisis en función del tiempo para la zona de Pascuales, una carga y un transformador se conecta a esta barra para realizar el control de voltaje. Como es una barra de carga PQ el perfil de voltaje cambia con su demanda, para la hora 18 cuando se tiene demanda máxima se tiene un nivel de voltaje mínimo, donde para el caso A se tiene un valor de 0.9537214 p.u. y para el caso B un valor de 0.9520209 p.u., pues se tiene un valor más bajo en la temporada seca y se cumple con los límites establecidos para el límite de magnitud de voltaje.

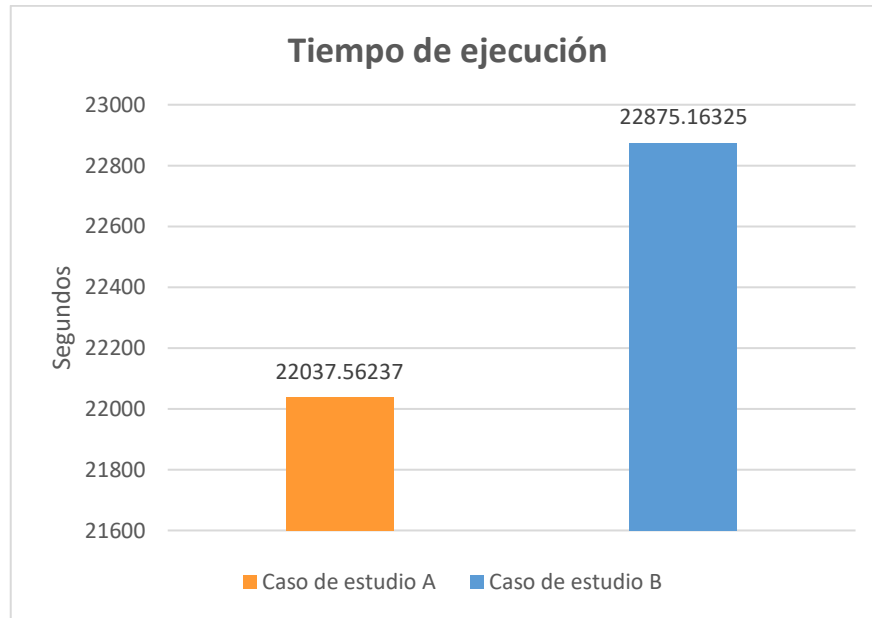


Figura 3.1.2.17. Tiempos de ejecución para los diferentes casos de estudio.

Como se visualiza en la Figura 3.1.2.17, se analiza que el tiempo de ejecución para los dos casos es muy elevado porque se tiene una gran cantidad de elementos eléctricos de la red lo que dificulta la resolución del problema de DHT-CP. Se tiene una mayor complejidad al incorporar los límites de flujo de potencia AC en las líneas y en los transformadores ya que se genera muchas ecuaciones para un gran número de barras del SNI ecuatoriano, pues el tiempo de ejecución es mayor. El caso B para la temporada seca su tiempo de ejecución es mayor en aproximadamente 14 minutos en comparación al caso A debido a que se tiene un mayor uso de memoria del computador al incorporar las variables binarias que se asocian al estado de operación de las centrales térmicas.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se desarrolló una herramienta computacional en lenguaje Python con la que se logró implementar un programa óptimo para el despacho hidrotérmico a corto plazo con diferentes alternativas para la modelación de la función de la potencia generada por las centrales hidroeléctricas, con el cual se obtiene una minimización de costos totales de producción de las centrales térmicas considerando flujo AC y pérdidas en el sistema eléctrico de potencia.
- Se analiza que para el caso de la temporada lluviosa debido a que se tiene mayor recurso hídrico en las centrales hidroeléctricas se tiene un menor costo de producción en la generación de potencia activa para suplir la demanda, pues la mayoría de generación se tiene por las centrales hidroeléctricas donde el porcentaje total de cobertura al día es de 94.528 % y el aporte de las centrales térmicas es de apenas el 5.472 %. En caso contrario, cuando se tiene el caso de la temporada seca se tiene una hidrología baja, se utiliza mucha más generación térmica para suplir la demanda dando como resultado costos totales de producción más elevados, donde el porcentaje total de cobertura de generación al día de las centrales hidroeléctricas es del 81.327 % y para las centrales térmicas son de 18.673 %.
- Se comprueba que la herramienta computacional implementada para el sistema IEEE de 14 barras da resultados similares en los costos totales de producción, pérdidas totales del sistema eléctrico de potencia, perfiles de generación y perfiles de voltaje para los diferentes casos de estudio cumpliendo todos los límites establecidos para las diferentes restricciones del problema, por lo que se demuestra la confiabilidad del software desarrollado y por consiguiente su aplicación a un sistema eléctrico más robusto como el SNI ecuatoriano, sin embargo el tiempo de ejecución es más elevado debido al gran número de elementos eléctricos de la red lo que dificulta la resolución del DHT-CP.
- El problema de despacho hidrotérmico a corto plazo se resuelve con el solucionador Ipopt el cual sirve para la resolución de problemas de optimización no lineales y es compatible con la librería Pyomo.
- Un gran problema al usar el algoritmo es el tiempo de ejecución para las redes de gran tamaño como el SNI ecuatoriano, ya que se demora mucho tiempo

construyendo las ecuaciones de magnitud y ángulo de voltaje para todas las barras mediante el flujo de potencia AC que posteriormente se utilizan en las restricciones de los límites de flujo de potencia de la red eléctrica y en las demás restricciones lo que dificulta la resolución del DHT-CP.

4.2 Recomendaciones

- Se analiza que los tiempos de cálculo para el sistema del SNI es extenso en comparación al sistema IEEE de 14 barras por lo que para mejorar el rendimiento del programa se recomienda implementar un modelo más optimizado en el flujo AC.
- Verificar los datos de entrada del Excel para una correcta implementación de las restricciones del problema de optimización.
- Usar valores adecuados en los datos de entrada de las diferentes restricciones del Excel para que el problema de DHT-CP converja.

5 REFERENCIAS BIBLIOGRÁFICAS

- [1] A. Wood, B. Wollenberg, y G. Sheblé, *Power Generation, Operation and Control*. New Jersey: John Wiley & Sons Inc, 2014.
- [2] G. S. Christensen y S. A. Soliman, *Optimal Long-Term Operation of Electric Power Systems*, Primera. New York, 1988.
- [3] M. E. El-Hawary y G. S. Christensen, *Optimal Economic Operation of Electric Power Systems*. Estados Unidos, 1979.
- [4] A. Soroudi, *Power System Optimization Modeling in GAMS*. Springer, 2017.
- [5] J. A. Henríquez Valencia y S. A. Montano Rivas, “Modelo de la programación de la operación de un sistema hidrotérmico con flujo óptimo de potencia utilizando herramientas convencionales de optimización”, Universidad de El Salvador, 2018.
- [6] C. A. Rueda Mayorga, “Implementación de una herramienta computacional en lenguaje Python para obtener las curvas de oferta de importación/ exportación de un sistema eléctrico de potencia”, Escuela Politécnica Nacional, 2021.
- [7] A. Harvey, A. Brown, P. Hettiarachi, y A. Inversin, *Micro-hydro design manual*. 1993.
- [8] S. Pinzón, “Despacho económico de generación Hidroeléctrica y Termoeléctrica usando programación dinámica”, Universidad Politécnica Salesiana Sede Quito, 2020.
- [9] A. Kumar Sharma, “Short term Hydrothermal scheduling using evolutionary programming”, Thapar University, 2009.
- [10] C. Coffrin y P. Van Hentenryck, “A Linear-Programming Approximation of AC Power Flows”, *INFORMS J. Comput.*, vol. 26, núm. 4, pp. 718–734, 2014, doi: 10.1287/ijoc.2014.0594.
- [11] S. Delgado, “Aprende Python”, *Aprende Python*, núm. python, pp. 1–488, 2022, [En línea]. Disponible en: https://aprendepython.es/_downloads/907b5202c1466977a8d6bd3a2641453f/aprendepython.pdf.
- [12] Hart, E. William, J. Watson, y D. L. Woodruff, *Pyomo – Optimization Modeling in Python*. Springer, 2017.
- [13] Y. Kawajir, “Ipopt Documentation”, *Ipopt*, 2005. <https://coin-or.github.io/Ipopt/>

(consultado jul. 01, 2023).

- [14] M. Moscoso, "HERRAMIENTA COMPUTACIONAL, EN LENGUAJE PYTHON, PARA LA SOLUCIÓN DEL PROBLEMA DE DESPACHO HIDROTÉRMICO DE CORTO PLAZO CONSIDERANDO LA RED ELÉCTRICA CON UN FLUJO ÓPTIMO DE POTENCIA DE CORRIENTE CONTINUA Y EL EFECTO DE LAS PÉRDIDAS", Escuela Politécnica Nacional, 2022.