TEC | Tecnológico de Costa Rica

Escuela de Ingeniería Electromecánica

Modelado de la Infraestructura Eléctrica en la Red de Distribución del ICE mediante software de simulación eléctrica en un Entorno GIS para Líneas de Baja y Media Tensión, Cargas y Transformadores

Informe de Trabajo Final de Graduación para optar por el Título de Ingeniero en Mantenimiento Industrial, Grado Licenciatura

Autor:

Keitlyn Valeria Montiel Arce

Cartago, Junio 2024

Carrera Acreditada por:

AAPIA

Agencia de Acreditación de programas de Ingeniería y de Arquitectura

Hoja de Datos

Información del Estudiante:

Nombre: Keitlyn Valeria Montiel Arce

Cédula: 2-0823-0840

Carné ITCR: 2019190037

Dirección de residencia en época lectiva: Cartago, Cartago, Oriental.

Teléfono: 8416-9282

Correo electrónico: keitlyn12354@gmail.com

Información del Proyecto:

Título: Modelado de la Infraestructura Eléctrica en la Red de Distribución del ICE mediante Software de Simulación Eléctrica en un Entorno GIS para Líneas de Baja y Media Tensión, Cargas y Transformadores.

Asesor Industrial: Mgtr. Gonzalo Mora Jiménez

Profesor Guía: Mgtr. Gonzalo Mora Jiménez.

Jurado Evaluador:

- M.Sc. Gustavo Gómez Ramírez.
- Mgtr. Luis Carlos Muñoz Chacón.

Información de la Empresa:

Nombre: Instituto Costarricense de Electricidad (ICE).

Zona: San José.

Dirección: Sabana Norte, edificio ICE.

Actividad principal: Desarrollo de fuentes productoras de energía eléctrica, incluyendo prestación de servicios de electricidad y el servicio de comunicaciones.

Contacto: Mgtr. Gonzalo Mora Jiménez.

Teléfono: 2000-5728.

Resumen

La creciente demanda de energía en Costa Rica y los cambios que buscan promover la generación distribuida en el país plantean desafíos para las empresas eléctricas, como el Instituto Costarricense de Electricidad (ICE). Uno de los principales desafíos radica en la necesidad de desarrollar modelos eléctricos precisos y actualizados que permitan visualizar el comportamiento de la red ante la inclusión de la generación distribuida.

Este proyecto tiene como objetivo desarrollar un modelo eléctrico detallado de las líneas de baja y media tensión, cargas y transformadores que conforman un circuito de la red de distribución eléctrica del ICE, preparando así la red, para futuros análisis que ayuden a la comprensión de su estado actual, facilitando la toma de decisiones frente a la incorporación de generación distribuida. Para ello, se utilizaron los softwares de simulación eléctrica OpenDSS y OpenDSS-G y se modeló el circuito en un entorno de Sistema de Información Geográfica (GIS).

Para llevar a cabo el modelado del circuito, se ejecutaron distintas fases. En la primera fase se implementaron estrategias de minería de datos utilizando Python para analizar los conjuntos de datos geográficos y eléctricos proporcionados por el ICE para garantizar su calidad. Luego, se desarrolló un programa en Python para convertir automáticamente estos datos al formato compatible con el software de simulación eléctrica. Una vez preparados los datos, se transfirieron al software para construir el modelo completo del circuito de la red. Finalmente, se realizaron simulaciones para evaluar el comportamiento del sistema eléctrico bajo diversas condiciones operativas.

Finalmente, la creación de programas en Python agilizó la preparación de los datos de los elementos del circuito, mejorando la eficiencia en la transferencia de información. Por otra parte, la minería de datos garantizó la calidad de los datos utilizados en el circuito. La automatización con Python facilitó la definición de elementos del circuito eléctrico. Por último, la validación con OpenDSS aseguró la precisióndel modelo para futuros análisis y simulaciones.

Palabras Claves: Red de distribución, Sistema de Información Geográfica (GIS), modelado, Python, OpenDSS, generación distribuida.

Abstract

The growing demand for energy in Costa Rica and the changes aimed at promoting distributed generation in the country pose challenges for electric companies, such as the Costa Rican Institute of Electricity (ICE). One of the main challenges lies in the need to develop accurate and up-to-date electrical models that allow visualizing the behavior of the grid in the face of the inclusion of distributed generation.

This project aims to develop a detailed electrical model of the low and medium voltage lines, loads, and transformers that make up a circuit of the Instituto Costarricense de Electricidad (ICE) electrical distribution network, thus preparing the grid for future analyses that help understand its current state and facilitate decision-making regarding the incorporation of distributed generation. For this purpose, the OpenDSS and OpenDSS-G electrical simulation software were used, and the circuit was modeled in a geographic information system (GIS) environment.

To carry out the circuit modeling, different phases were executed. In the first phase, data mining strategies were implemented using Python to analyze the geographic and electrical data sets provided by ICE to ensure their quality. Then, a Python program was developed to automatically convert this data to the format compatible with the electrical simulation software. Once the data was prepared, it was transferred to the software to build the complete model of the grid circuit. Finally, simulations were performed to evaluate the behavior of the electrical system under various operating conditions.

Ultimately, the creation of Python programs streamlined the preparation of the circuit element data, improving efficiency in information transfer. Furthermore, data mining ensured the quality of the data used in the circuit. Python automation facilitated the definition of electrical circuit elements. And finally, validation with OpenDSS ensured the accuracy of the model for future analyses and simulations.

Key Words: distribution network, geographic information system (GIS), modeling, Python, OpenDSS, Distributed generation.

Agradecimientos

Quisiera expresar mi más profundo agradecimiento a mi familia, en especial a mi papá y a mi mamá, cuyo amor incondicional, constante apoyo y sacrificios innumerables han sido la piedra angular de mi camino educativo. Su aliento y ejemplo han sido mi mayor inspiración y motivación para alcanzar mis metas.

Agradezco también a todas las personas que han sido parte de mi formación educativa, desde mis primeros maestros hasta mis compañeros de clase y amigos cercanos que han contribuido a mi crecimiento personal y académico.

En particular, deseo reconocer y agradecer al profesor Gonzalo Mora por su invaluable orientación y apoyo en el desarrollo de este proyecto. Sus conocimientos, experiencia y dedicación han sido fundamentales para el desarrollo de este trabajo.

A todos ustedes, mi más sincero agradecimiento por ser parte de este viaje. Sin su apoyo y aliento este logro no habría sido posible.

Dedicatoria

A mis queridos padres,

por su amor incondicional, su constante apoyo y su eterna inspiración.

Gracias por ser mi roca en los momentos difíciles y mi mayor alegría en los triunfos.

Con todo mi amor.

Miembros del Tribunal

Informe presentado a la Escuela de Electromecánica del Instituto Tecnológico de Costa Rica como requisito parcial para optar por el Título de Ingeniera en Mantenimiento Industrial con el grado de Licenciatura

GONZALO MORA JIMENEZ (FIRMA)
Firmado digitalmente por GONZALO MORA JIMENEZ (FIRMA)
Fecha: 2024.06.06 21:33:34 -06'00'

Magister Gonzalo Mora Jiménez
Supervisor

GUSTAVO ADOLFO GOMEZ RAMIREZ (FIRMA)
Firmado digitalmente por GUSTAVO ADOLFO GOMEZ RAMIREZ (FIRMA)
Fecha: 2024.06.05 16:53:26 -06'00'

M.Sc. Gustavo Gómez Ramírez
Lector 1

LUIS CARLOS MUÑOZ CHACON (FIRMA)
Firmado digitalmente por LUIS CARLOS MUÑOZ CHACON (FIRMA)
Fecha: 2024.06.06 12:15:22 -06'00'

Magister Luis Carlos Muñoz Chacón
Lector 2

GONZALO MORA JIMENEZ (FIRMA)
Firmado digitalmente por GONZALO MORA JIMENEZ (FIRMA)
Fecha: 2024.06.06 21:34:17 -06'00'

Magister Gonzalo Mora Jiménez
Asesor académico

Índice general

Índice de tablas

Índice de figuras

Nomenclatura

Δ	Configuración en Delta
A	Amperios
AAAC	Conductor de Aleación de Aluminio
AAC	Conductor de Aluminio
ACSR	Conductor de Aluminio Reforzado con Acero
CA	Corriente alterna
Cu	Copper (Cobre)
DSS	Distribution System Simulator
EMT	Transitorio Electromagnético
EPERlab	Electrical Power and Energy Research Laboratory
GIS	Sistema de Información Geográfica
Hz	Hercio
I	Corriente
ICE	Instituto Costarricense de Electricidad
IPSA	Análisis Integrado de Sistemas de Energía
kV	Kilo Voltio
kW	Kilo Vatios
LL	Línea-Línea

LL	Línea-Neutro
MW	Mega Vatios
OS	Organismo Supervisor
P	Potencia activa
PSS/E	Simulador de sistemas de energía para ingeniería
Q	Potencia reactiva
RF	Radio frecuencia
S	Potencia aparente
V	Tensión
Y	Configuración trifásica en estrella

Capítulo 1

Introducción

Actualmente, las empresas eléctricas de Costa Rica enfrentan un desafío debido al crecimiento en tamaño y complejidad de la red eléctrica. El aumento de la red y las nuevas tecnologías que dependen de esta, han introducido una creciente necesidad de planificación para futuras expansiones. Por otra parte, el costo asociado a las adiciones o modificaciones de la infraestructura eléctrica combinado con la necesidad de garantizar un suministro de energía eficiente han impulsado el interés en desarrollar tecnologías para la gestión eficiente de sistemas de energía eléctrica.

Entregar un servicio de energía de calidad a los consumidores finales de energía es esencial para las empresas distribuidoras de energía, es por esto que la confiabilidad del sistema eléctrico se ha convertido en un aspecto crucial en la gestión de los sistemas eléctricos.

Según Gómez-Ramírez (2016) la confiabilidad se define como la probabilidad de que un componente, subsistema o sistema desempeñe adecuadamente sus funciones durante un período bajo condiciones operativas específicas, esto implica no solo la continuidad del servicio sino también la buena regulación de tensión y control de frecuencia, factores esenciales para el correcto funcionamiento de la red eléctrica; por lo que evaluar y garantizar la confiabilidad es vital para minimizar interrupciones y asegurar un suministro continuo y de calidad a los usuarios finales.

Con el fin de garantizar la confiabilidad, los ingenieros deben realizar estudios detallados que tengan en cuenta tanto las condiciones operativas normales como los eventos transitorios, así como las variaciones en las cargas y las expectativas futuras de funcionamiento. Es por esto que los programas para el análisis y diseño de sistemas eléctricos se han convertido en una herramienta esencial para la planificación y manejo eficiente de la red. Estos programas abarcan una amplia gama de funciones, desde el cálculo del flujo de potencia hasta la evaluación de cortocircuitos y eventos transitorios. Además, son capaces de determinar las características del sistema en condiciones de operación estacionaria, permitiendo a los ingenieros tomar las decisiones

fundamentadas sobre ajustes en la red.

La evaluación de la confiabilidad mediante técnicas probabilísticas y determinísticas, como señala Gómez-Ramírez (2016), juega un papel fundamental en la planificación y expansión de la red eléctrica. Estas evaluaciones permiten identificar puntos críticos y predecir el comportamiento del sistema bajo diversas condiciones operativas, facilitando la implementación de mejoras y optimizaciones. La integración de estas técnicas en los programas de análisis no solo mejora la precisión de los modelos, sino que también contribuye a una gestión más eficiente y resiliente de la infraestructura eléctrica.

Dentro de los programas de análisis y diseño de sistemas eléctricos, se encuentran los sistemas de Información Geográfica (GIS). Estos se han revelado como una herramienta de suma importancia en la gestión de redes eléctricas de la industria energética moderna. Los GIS permiten una visualización detallada y precisa de la infraestructura eléctrica en un contexto espacial, facilitando la identificación y ubicación de componentes clave como líneas de transmisión, subestaciones, transformadores y puntos de conexión.

Esta capacidad de visualización espacial no solo mejora la comprensión de la red eléctrica, sino que también permite realizar análisis avanzados, como la optimización de rutas de transmisión, la planificación de mantenimiento preventivo basado en ubicación y la predicción de posibles áreas de fallo, lo que contribuye significativamente a la eficiencia operativa y la confiabilidad del suministro eléctrico.

Este proyecto se centra en el modelado de las líneas de un circuito de potencia utilizando un entorno de Sistema de Información Geográfica (GIS) en el programa OpenDss. El objetivo principal es desarrollar un modelo eléctrico de las líneas de baja, media tensión, cargas y transformadores, en la red de distribución eléctrica de ICE, utilizando un software de simulación eléctrica dentro de un entorno GIS; preparando así la red para futuros análisis que ayuden a la comprensión de su estado actual, facilitando la toma de decisiones frente a la incorporación de generación distribuida.

Además, para realizar este proyecto, se busca utilizar herramientas tecnológicas como Python que ayudarán en la implementación eficiente del modelado de las líneas de circuito de potencia en el entorno GIS. Python, con su amplia gama de bibliotecas especializadas, proporciona un entorno de desarrollo versátil que permitirá la manipulación, análisis y visualización de datos de manera efectiva. Con este proyecto se pretende brindar a la empresa ICE una herramienta valiosa para la planificación efectiva y la gestión óptima de la infraestructura eléctrica.

1.1. Reseña de la Empresa

1.1.1. Historia

El Instituto Costarricense de Electricidad (ICE) se estableció en 1949 tras décadas de esfuerzos para abordar la escasez de energía eléctrica en Costa Rica. Surgió de la iniciativa de ingenieros y líderes como Jorge Manuel Dengo Obregón y se destacó por su enfoque en la electrificación y el desarrollo sostenible (Instituto Costarricense de Electricidad, 2021). En sus inicios, solo el 14 % del país tenía acceso a la energía eléctrica, pero a lo largo del tiempo, el ICE expandió la cobertura, introdujo diversas fuentes renovables como hidroeléctrica, geotérmica, eólica y solar, y desafió los problemas de telecomunicaciones, masificando la telefonía fija, pública, móvil e internet. (Instituto Costarricense de Electricidad, 2021).

El ICE, a lo largo de su historia, ha sido fundamental en la modernización y expansión de la infraestructura eléctrica y de comunicaciones en Costa Rica. Desde su creación, ha evolucionado continuamente, promoviendo la adopción de tecnologías emergentes, como la electrificación de alta calidad, la diversificación de energías renovables y la introducción de servicios de comunicación avanzados (Instituto Costarricense de Electricidad, 2021).

Su impacto se refleja en la alta cobertura eléctrica del 99,7 % enel país, así como en su liderazgo en la descarbonización de la economía a través de iniciativas como la electromovilidad y el desarrollo de ciudades inteligentes. A través de sus empresas asociadas, el ICE ha contribuido significativamente al progreso tecnológico y a la conectividad global de Costa Rica (Instituto Costarricense de Electricidad, 2021).

1.1.2. Misión

"Brindar energía, conectividad y servicios digitales, seguros y sostenibles a los habitantes de Costa Rica". (Instituto Costarricense de Electricidad, 2023).

1.1.3. Visión

“El Grupo ICE liderará la electrificación renovable de la economía y proveerá al país de un ecosistema seguro de telecomunicaciones digitales de última generación". (Instituto Costarricense de Electricidad, 2023).

1.1.4. Valores de la empresa

Los valores fundamentales que guían el Grupo ICE se basan en: la integridad, siendo coherentes entre lo expresado y las acciones realizadas, promoviendo la confianza, transparencia, honestidad, rectitud y

respeto, en aras del crecimiento personal y colectivo; el compromiso, sintiéndose orgulloso de formar parte de esta organización, aportando desde cada rol al valor generado por la empresa, siendo conscientes del significativo servicio ofrecido al país; la excelencia, buscando de forma constante resultados sobresalientes que influyan en los objetivos del Grupo ICE, fomentando la innovación, mejora continua, colaboración y métodos ágiles para impulsar una cultura de responsabilidad y el desarrollo integral de su recurso humano. (Instituto Costarricense de Electricidad, 2023).

1.2. Planteamiento del Problema

Los Sistemas de Información Geográfica (GIS) se han posicionado como una herramienta sumamente importante para las empresas eléctricas, debido a que los GIS permiten abordar de manera efectiva las necesidades de mapeo y gestión de recursos. También, ofrecen una manera efectiva de manejar la información espacial al permitir la división de datos en capas temáticas almacenadas de forma independiente. Esto facilita un manejo ágil y simplificado de la información, mejorando la capacidad de las empresas eléctricas para analizar y visualizar datos geoespaciales críticos para sus operaciones. (Hormazábal Sanhuezaand Ramírez Estrada, 2014).

Además de las innovaciones tecnológicas como los Sistemas de Información Geográfica, los cambios en el marco regulatorio también están transformando el panorama energético global. Ejemplo de esto es la implementación de leyes como la Ley 10086 en Costa Rica, que busca promover y regular el uso de fuentes renovables para la generación de energía, permitiendo a los consumidores comerciales y residenciales producir electricidad a partir de fuentes renovables, a cambio de créditos en sus facturas eléctricas e inyectar los excedentes en la red eléctrica nacional. (La Gaceta, 2023).

Sin embargo, la integración de ciertas energías renovables presenta desafíos sustanciales para las empresas distribuidoras del sistema eléctrico costarricense. Uno de los principales interrogantes es cómo afectará esta transición a la calidad de la tensión entregada por la red y cuáles podrían ser las consecuencias derivadas. La variabilidad inherente de fuentes como la energía solar y eólica plantea desafíos técnicos, ya que pueden generar fluctuaciones en la frecuencia y estabilidad de la red, lo que podría afectar la confiabilidad del suministro eléctrico.

Además, la incorporación de generación distribuida con energías renovables requiere cambios operativos significativos por parte de las empresas distribuidoras. Estos cambios incluyen la implementación de nuevos programas de gestión de la red eléctrica y la necesidad de capacitar al personal. Asimismo, es posible que se requiera inversión en infraestructura para adaptar la red a las nuevas demandas y asegurar su estabilidad frente a los cambios en la matriz energética.

En respuesta a estos desafíos, la regulación en Costa Rica está adoptando medidas concretas para facilitar la integración de energías renovables en el sistema eléctrico. El artículo 5, inciso f) de la Ley N° 10086 incluye disposiciones que exigen a las empresas distribuidoras proporcionar al Organismo Supervisor (OS) datos detallados y modelos de circuitos en un formato GIS para recibir recomendaciones no vinculantes sobre la aplicación de este procedimiento, teniendo únicamente un plazo de 24 meses una vez que la ley entra en vigencia para entregar estos modelos. (La Gaceta, 2023).

La implementación de estos requisitos regulatorios presenta desafíos particulares para empresas como el Instituto Costarricense de Electricidad (ICE). En el caso de la empresa estatal ICE, se poseen los datos en un Sistema de Información Geográfica, pero lamentablemente no todos estos datos están integrados en un programa que posibilite la realización de estudios esenciales, pedidos por la ley. Esto hace que surja la necesidad de modelar el sistema eléctrico en un software de simulación eléctrica dentro del entorno GIS.

1.3. Objetivo General

Establecer un modelo eléctrico de las líneas de baja y media tensión, cargas y transformadores de un circuito de la red de distribución del ICE utilizando OpenDSS y OpenDSS-G en un entorno GIS facilitando análisis futuros y la toma de decisiones sobre generación distribuida.

1.4. Objetivos Específicos

1. Implementar estrategias de minería de datos utilizando Python para el análisis de los conjuntos de datos geográficos y eléctricos proporcionados por la empresa ICE, garantizando la calidad, integridad y consistencia de los datos, así como su utilidad en análisis y simulaciones futuras.

2. Desarrollar un programa en Python que convierta de manera automatizada los archivos de formato Excel del Sistema de Información Geográfica del ICE a archivos compatibles con la plataforma utilizada por la Instituto para la simulación de circuitos, optimizando el proceso de transferencia de información.

3. Transferir los archivos generados por Python al software de simulación eléctrica, con el fin de construir un modelo del circuito que permita el futuro análisis del comportamiento del sistema eléctrico.

4. Realizar con OpenDSS análisis de flujos de carga en el modelo del circuito eléctrico del ICE, proporcionando una base para futuros estudios y simulaciones.

1.5. Justificación

El crecimiento continuo de la demanda de energía y la expansión de las redes eléctricas ha provocado que las estas requieran una gestión eficiente para garantizar un suministro de energía confiable y de calidad. La representación geográfica de un sistema permite una mejor visualización de la infraestructura eléctrica, como el GIS está compuesto por una detallada base de datos geoespaciales de la red de distribución, facilita una mejor comprensión de la topología de la red, la ubicación de componentes clave y las características físicas del sistema. Esta información posibilita llevar a cabo análisis avanzados, simulaciones y una planificación efectiva de las operaciones y el mantenimiento, permitiendo a los ingenieros encargados de la planificación tener una visión integral de la red de distribución, promoviendo la gestión eficiente del sistema. (Nayeripour et al., 2010).

En contraste, la falta de una representación precisa de la red eléctrica en un entorno geográfico puede llevar a una planificación inadecuada, dificultades en la localización de problemas y retrasos en la respuesta a fallos, lo que resultaría en interrupciones del suministro eléctrico, pérdidas económicas y en última instancia, insatisfacción de los usuarios.

La solución propuesta se basa en la integración de datos geográficos y eléctricos en un entorno de GIS, utilizando el software de simulación eléctrica OpenDSS para el modelado del sistema. Esto permitirá una representación precisa de las líneas de transmisión y distribución, facilitando la planificación óptima y la toma de decisiones informadas. La visualización interactiva y las funcionalidades de análisis en el software de simulación eléctrica mejorarán la eficiencia operativa y la confiabilidad del sistema eléctrico.

La implementación de este proyecto implica una optimización de la operación, planificación efectiva de expansión, aumentando la competitividad de la empresa. Actualmente, la gestión de redes eléctricas carece de una representación integral en un entorno geográfico, lo que limita la eficiencia en la operación y planificación, así como la respuesta efectiva a incidencias. Las empresas distribuidoras necesitan tener un Sistema de Información Geográfica robusto e integrado que proporcione una representación precisa y completa de las líneas de circuito de distribución, para permitir una gestión óptima y una respuesta eficiente ante situaciones imprevistas en la red eléctrica.

1.6. Viabilidad

Para garantizar la viabilidad y éxito del proyecto de modelado de líneas de un circuito de potencia en un entorno de Sistema de Información Geográfica (GIS), se cuenta con las siguientes herramientas.

- Software de simulación eléctrica.

- Lenguajes de programación: Python.
- Información sobre la configuración de los elementos del sistema eléctrico.

1.7. Antecedentes del Proyecto

En la actualidad el desarrollo tecnológico continuo en Costa Rica está impulsando una revolución en la gestión y distribución de la electricidad, siendo este indispensable para sostener el estilo de vida moderno y asegurar una gestión de recursos sostenible. El crecimiento de las energías renovables, impulsado por la ley 10086, está transformando la forma en que se genera y se distribuye la electricidad. Este cambio hacia fuentes más sostenibles ha creado la necesidad de modernizar y optimizar los sistemas de distribución eléctrica para garantizar una red confiable y eficiente.

Como presenta Viganò et al. (2021), en su artículo la aparición de nuevas tecnologías que buscan reemplazar fuentes contaminantes, como los generadores fotovoltaicos residenciales y los vehículos eléctricos, implica que la mayoría de estos equipos se conecten a las redes de distribución eléctrica. Esto requiere que los encargados de la planeación del sistema de distribución refuercen la red o adopten soluciones de control avanzadas para evitar violaciones de corriente y tensión. Además, explica que el GIS es una herramienta indispensable que permite obtener un modelo preciso de la red eléctrica; facilitando así la planificación eficiente de la infraestructura eléctrica, incluyendo la ubicación óptima de subestaciones, líneas de transmisión y distribución, considerando variables como la demanda energética y los recursos renovables disponibles, posibilitando la simulación de distintos escenarios.

Se ha demostrado que los Sistemas de Información Geográfica (GIS) son herramientas sumamente efectivas para mejorar y prever diversos aspectos en una red eléctrica. Mediante la integración de datos geoespaciales con información detallada sobre infraestructura eléctrica, demanda de energía, condiciones climáticas y otros factores relevantes, el GIS permite realizar análisis avanzados y modelar diferentes escenarios.

El uso de Sistemas de Información Geográfica (GIS) ha sido fundamental en la optimización de redes eléctricas, debido a su capacidad para integrar y visualizar información estática y dinámica de manera simultánea. Derakhshan et al. (2013) en su artículo explica cómo se implementó una nueva metodología basada en GIS en Azerbaiyán Oriental, para optimizar las operaciones de distribución de energía, utilizando un modelo de programación que incorpora datos estáticos de la red (como ubicación de infraestructuras y condiciones geográficas) junto con datos dinámicos en tiempo real (como tensión y corriente de líneas), el sistema calcula de manera eficiente las cargas de los clientes y ajusta el equilibrio de carga en tiempo real, lo que demuestra la efectividad del GIS para mejorar la gestión y operación de las redes eléctricas.

Adicionalmente, Chen et al. (2017) menciona en su artículo que el GIS puede combinarse con una base de datos climática para analizar las condiciones climáticas previstas en un lugar particular. Mediante el uso de curvas de daño, se podría estimar el potencial daño y la vulnerabilidad de los componentes de la red eléctrica ante una amenaza climática específica, considerando también las condiciones pronosticadas en esa región, facilitando la evaluación de la capacidad de adaptación de la red eléctrica frente a eventos extremos como tormentas, terremotos o fluctuaciones en la demanda, lo que ayuda a implementar estrategias de mitigación y respuesta.

Por otra parte, Gómez-Ramírez et al. (2021) presenta un estudio de caso sobre la mejora de la respuesta a la demanda utilizando sistemas de almacenamiento de energía en Honduras. El artículo destaca cómo la implementación de sistemas de almacenamiento puede contribuir significativamente a mejorar la estabilidad y eficiencia del sistema eléctrico, especialmente en regiones con desafíos en la gestión de la demanda. Además, Gómez-Ramírez et al. (2022) proporciona ideas importantes sobre cómo el uso estratégico del almacenamiento puede abordar los desafíos de capacidad en sistemas eléctricos regionales.

Ambos artículos ofrecen diferentes perspectivas que evidencian los beneficios de modelar en un entorno geográfico. Entre estos beneficios se incluyen:

- Mejora en la planificación y optimización de la infraestructura eléctrica al tener en cuenta datos geoespaciales sobre la distribución de recursos energéticos y la demanda de energía.
- Facilita la identificación de ubicaciones óptimas para la implementación de sistemas de almacenamiento de energía al considerar factores geográficos y de demanda.
- Permite una mejor integración de las energías renovables al tomar en consideración la disponibilidad geográfica de recursos como la luz solar y la velocidad del viento.
- Facilita la gestión de eventos climáticos extremos al proporcionar información geoespacial sobre áreas vulnerables y posibles impactos en la infraestructura eléctrica.

1.8. Metodología

Este proyecto se dividirá en cuatro fases distintas:

Fase 1: Recolección y minería de Datos.

En esta etapa inicial, se realizará la recolección de datos del Departamento GIS de ICE. Se obtendrán conjuntos de datos geográficos y eléctricos necesarios para realizar el modelo del circuito eléctrico. Posteriormente, se llevará a cabo la depuración de datos para asegurar la integridad y consistencia de la información recopilada.

Fase 2: Preparación de Datos para Simulación.

Una vez depurados, los datos geográficos serán transformados al formato compatible con el software de simulación eléctrica (OpenDSS). Se desarrollará un programa en Python para automatizar esta transformación, para asegurar la compatibilidad de los datos con el programa.

Fase 3: Modelado del circuito.

En esta etapa, se procederá a construir el modelo eléctrico de la red de distribución utilizando el software OpenDSS. Se calcularán las matrices de impedancia de las líneas y se conectarán los componentes eléctricos (como transformadores, líneas de media y baja tensión y cargas) en el modelo. Se realizarán simulaciones de flujos de carga para evaluar el comportamiento del sistema eléctrico bajo diversas condiciones operativas.

Fase 4: Análisis y generación de gráficos.

Finalmente, se analizarán los resultados obtenidos de las simulaciones para extraer información relevante sobre el funcionamiento del sistema eléctrico.

Para una visualización clara y concisa de las fases del proyecto junto con sus objetivos y actividades asociadas, se presenta la tabla 1.1. Esta tabla proporciona un resumen estructurado de cada fase del proyecto, describiendo los objetivos específicos y las actividades detalladas que se llevarán a cabo en cada etapa. Por otra parte, en la figura 1.1 se observa un diagrama de flujo de la metodología.

Tabla 1.1

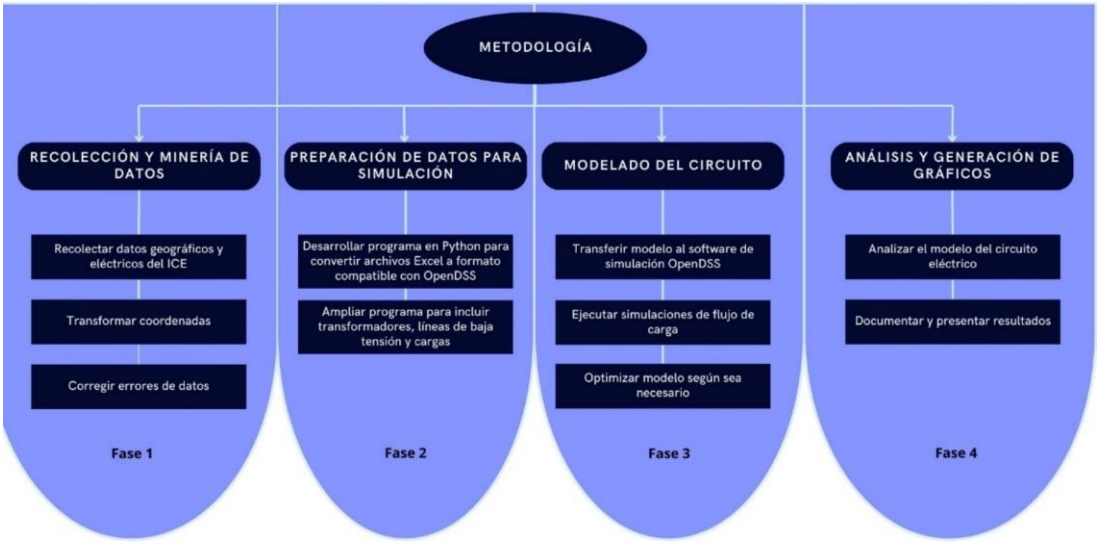
Cuadro Metodológico

Fase	**Objetivo asociado**	**Actividades**
Recolección y minería de datos	Implementar estrategias de minería de datos utilizando Python para el análisis de los conjuntos de datos geográficos y eléctricos proporcionados por la empresa ICE, garantizando la calidad, integridad y consistencia de los datos, así como su utilidad en análisis y simulaciones futuras.	1. Recolección de Datos: Colaborar con el Departamento de Información Geográfica (GIS) del ICE para obtener conjuntos de datos geográficos y eléctricos relevantes para la red de distribución. 2. Transformación de Coordenadas: Desarrollar un programa en Python para transformar las coordenadas geográficas al formato requerido por el software de simulación eléctrica (OpenDSS), asegurando la precisión y compatibilidad de los datos. 3. Corrección de Errores de Datos: Identificar y corregir posibles errores en los datos, como líneas desconectadas o inconsistencias en las especificaciones, mediante algoritmos de limpieza de datos en Python.
Preparación de Datos para Simulación	Desarrollar un programa en Python que convierta de manera automatizada los archivos de formato Excel del Sistema de Información Geográfica del ICE a archivos compatibles con la plataforma utilizada para la simulación de circuitos, optimizando el proceso de transferencia de información.	1. Desarrollo del generador de archivos automatizado: Implementar un programa en Python que lea archivos de Excel con especificaciones de los buses y líneas de media tensión y genere los archivos DSS compatibles con el programa para la simulación de circuitos. 2. Ampliación del generador de archivos automatizado: Ampliar el programa Python para generar archivos DSS que incluyan transformadores, líneas de baja tensión, cargas.
Modelado del circuito	Transferir los archivos generados por Pythonal software de simulación eléctrica, con el fin de construir un modelo del circuito que permita el futuro análisis del comportamiento del sistema eléctrico.	1. Simulación y Evaluación del Modelo: Transferir el modelo completo al software OpenDSS y ejecutar simulaciones de flujos de carga para evaluar el comportamiento del sistema eléctrico. 2. Optimización del Modelo: Realizar ajustes en el modelo de simulación según sea necesario para mejorar su precisión y utilidad en futuros análisis y estudios de la red eléctrica.
Análisis y generación de gráficos	Realizar con OpenDSS análisis de flujos de carga en el modelo del circuito eléctrico del ICE, proporcionando una base para futuros estudios y simulaciones.	1. Análisis del Modelo Eléctrico: Obtener gráficos y resultados que permitan determinar el comportamiento del modelo. 2. Documentación y Presentación de Resultados: Documentar el proceso de desarrollo y los resultados obtenidos en un informe final.

Fuente: Elaboración propia.

Figura 1.1

Diagrama de Flujo de la Metodología



Fuente: elaboración propia.

1.8.1. Alcance

El alcance de este proyecto incluye diversas actividades y tareas que se llevarán a cabo para lograr los objetivos planteados. A continuación, se detallan las acciones específicas que se realizarán:

- Recolección y minería de datos: se recopilarán datos geográficos y eléctricos relevantes del Departamento GIS del ICE. Esta información es fundamental para construir el modelo del circuito eléctrico, asegurando que todos los elementos y variables necesarios estén representados adecuadamente.
- Transformación y preparación de datos: se desarrollará un programa en Python para transformar los datos geográficos al formato requerido por el software de simulación OpenDSS. La transformación precisa de datos es crucial para garantizar la compatibilidad con el software de simulación, lo que permitirá una representación precisa de la red eléctrica.
- Modelado del circuito: utilizando OpenDSS, se construirá un modelo eléctrico de la red de distribución, incluyendo líneas de baja y media tensión, cargas y transformadores.
- Simulaciones y análisis de resultados: se ejecutarán simulaciones de flujo de carga para evaluar el comportamiento del sistema eléctrico.

- Generación de gráficos y documentación: Se generarán gráficos y se documentarán los resultados obtenidos de las simulaciones. La visualización de los datos y resultados facilita la interpretación y proporciona una base para futuros análisis.

Entregables

Los entregables del proyecto se enumeran a continuación:

1. Programa de generación de archivos de extensión DSS en Python: código principal que prepara todos los datos dados en Excel para convertirlos en el formato compatible con OpenDSS.
2. Programa en Python para la minería de datos de las líneas de media tensión.
3. Programa en Python para la minería de datos de las líneas de baja tensión.
4. Modelo Eléctrico del Circuito: Archivos del modelo del circuito eléctrico construido en OpenDSS, incluyendo todos los componentes y configuraciones necesarias.
5. Resultados de Simulaciones de Flujo de Carga: informes y gráficos que muestran los resultados de las simulaciones de flujo de carga.
6. Documentación y Presentación Final: documento final que integra todo el trabajo realizado, los métodos utilizados, los resultados obtenidos y las conclusiones del proyecto.

1.8.2. Limitaciones

Aunque el proyecto de modelado de líneas de circuito de potencia en un entorno GIS es de gran importancia para ICE, hay diferentes limitaciones debido al alcance y recursos disponibles. A continuación, se describirán algunos aspectos que no serán abordados en este proyecto, pero podrían ser explorados en futuros proyectos:

- No se realizarán estudios específicos de confiabilidad, estabilidad transitoria o flujos de potencia detallados en el modelo, debido a las restricciones de tiempo del proyecto.
- No se llevará a cabo una verificación exhaustiva del modelo mediante la comparación con datos reales del sistema eléctrico, ya que no se cuenta con la información necesaria de mediciones en campo.
- El alcance del proyecto se limita al modelado de las líneas de circuito de potencia y componentes principales, sin incluir un análisis profundo de la calidad de la energía o la implementación de estrategias de control avanzadas.

- Dependiendo de los módulos y la licencia del software de simulación eléctrica utilizado, es posible que existan limitaciones en cuanto al tamaño del sistema que se puede modelar o las funcionalidades disponibles para el análisis.

Estas limitaciones se deben principalmente a las restricciones de tiempo y recursos del proyecto, así como a la falta de disponibilidad de ciertos conjuntos de datos necesarios para realizar verificaciones y estudios más detallados. Sin embargo, el modelo desarrollado sentará las bases para futuros análisis y mejoras en el sistema eléctrico de ICE.

1.8.3. Cronograma

A continuación, en la figura 1.2 se muestra el cronograma del proyecto.

Figura 1.2

Cronograma del Proyecto

Tecnológico de Costa Rica
Escuela de Ingeniería Electromecánica
Cronograma de Trabajo Final de Graduación
Primer Semestre 2024

TEC | Tecnológico de Costa Rica

Fase	Actividad	1	2	3	4	5	6	7	Semana Santa	8	9	10	11	12	13	14	15	16	17
Recolección y minería de Datos	Recolección de Datos.																		
	Transformación de Coordenadas.																		
	Corrección de Errores de Datos.																		
Preparación de Datos para Simulación	Desarrollo del generador de archivos automatizado.																		
	Ampliación del generador de archivos automatizado.																		
Modelado del circuito	Simulación y Evaluación del Modelo.																		
	Optimización del Modelo.																		
Análisis y generación de gráficos	Validación del Modelo Eléctrico.																		
	Documentación y Presentación de Resultados.																		
Entrega de documento a profesores lectores																			
Revisión del documento por parte del profesor guía y lectores																			
Revisión del documento por parte del filólogo																			
Entrega final del documento																			
Exposiciones de práctica de especialidad ante jurado																			

Fuente: Elaboración propia

Capítulo 2

Marco Teórico

2.1. Sistemas de potencia

Un sistema de potencia se define como un conjunto de dispositivos que transforman energía primaria en energía eléctrica, utilizando corriente alterna para facilitar su transporte y distribución hacia los consumidores finales. Este sistema se compone principalmente de tres etapas clave: generación, transmisión y distribución. En la figura 2.1 se puede observar una descripción breve de estos procesos. (Matulic, 2003).

Figura 2.1:

Etapas de los Sistemas de Potencia: Generación, Transmisión y Distribución

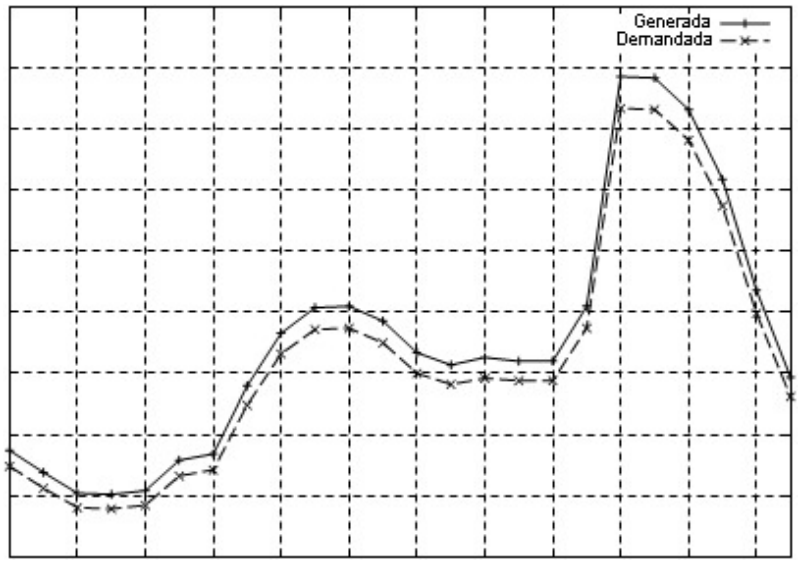
Fuente: Contraloría General de la República (2019).

Según Matulic (2003), el proceso de generación, transmisión y distribución de la energía eléctrica es esencialmente instantáneo, dado que la electricidad en forma de corriente alterna no se puede almacenar. Por lo tanto, el comportamiento de un sistema de potencia es dinámico, ya que el consumo de energía varía con el tiempo. En cada instante, la potencia generada debe ser exactamente igual a la consumida más la pérdida en los procesos de transmisión y distribución. De no cumplirse esta condición, los generadores del sistema, que funcionan a una velocidad constante para mantener estable la frecuencia eléctrica, acelerarán o desacelerarán dependiendo de si hay un exceso o déficit de generación.

Para mantener el equilibrio y la estabilidad en todo momento, existen mecanismos de control en las centrales que, ante un aumentoo disminución en el consumo eléctrico, automáticamente ajustan la potencia mecánica entregada por lasturbinas. En la figura 2.2 se muestra un ejemplo de una curva diaria de carga de un sistema de potencia.

Figura 2.2

Curva Diaria de Carga.



Fuente: Matulic (2003).

2.2. Generación

La etapa inicial de un sistema eléctrico de potencia es la generación eléctrica, que consiste en producir electricidad a partir de diversas fuentes de energía. Este proceso se lleva a cabo en centrales generadoras, las cuales están distribuidas en un territorio y funcionan en paralelo.

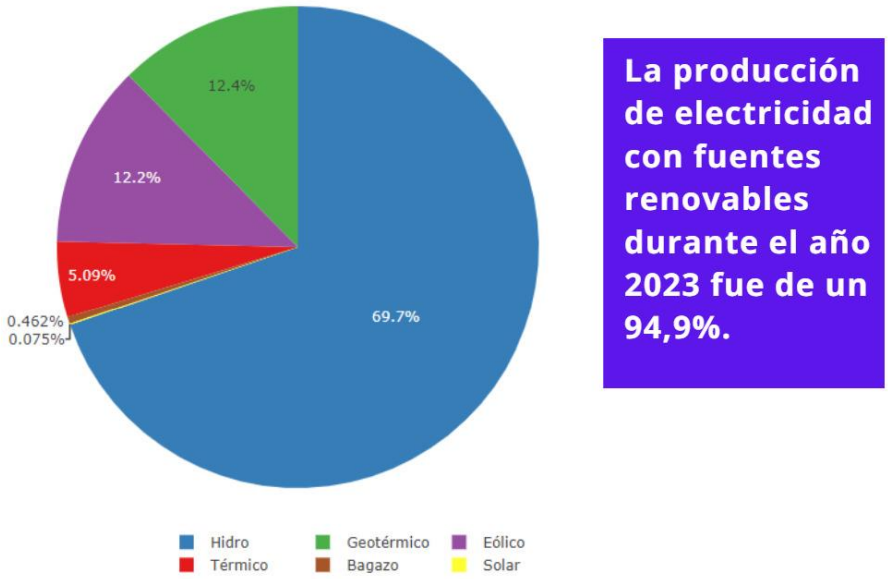
Existen varias fuentes de energía utilizadas para la generación eléctrica, que incluyen: energía hidroeléctrica, obtenida del agua y mediante represas; energía térmica, derivada de combustibles fósiles

como gas, petróleo y carbón; energía nuclear, generada a partir de uranio; energía eólica, obtenida del viento; energía geotérmica, proveniente del calor interno de la Tierra; y energía solar, derivada del sol. Además, la electricidad no solo se genera localmente, sino que también se transporta e importa desde países vecinos;un grupo de países con contratos comerciales a largo plazo y un flujo de energía y control de frecuencia común forman un consorcio de energía. (Schavemaker and van der Sluis, 2017).

Los recursos necesarios para obtener la energía eléctrica no están distribuidos de manera uniforme en todo el mundo; sin embargo, cada región cuenta con algún tipo de recurso de energía renovable. En Costa Rica, según el Instituto Costarricense de Electricidad (ICE) (2023), la generación de energía en 2023 fue 91,3 % renovable; en la figura 2.3 se muestra en detalle el porcentaje de generación según el tipo de energía que se utilizó.

Figura 2.3

Producción de Electricidad Según el Tipo de Fuente en Costa Rica



Fuente: Matulic (2003).

Costa Rica se destaca en la región centroamericana por su notable avance en la generación de electricidad a partir de fuentes renovables. Gómez-Ramírez et al. (2023a) explica que Costa Rica logró que más del 99% de su producción energética fuera renovable en el 2020. Esta situación es singular en comparación con otros países de Centroamérica, donde la dependencia de fuentes no renovables sigue siendo considerable. Por ejemplo, Nicaragua aún utiliza una proporción del 33.38% de energías no renovables para su abastecimiento energético, aunque también tiene un plan de expansión significativo para incrementar su capacidad de energías renovables.

Además, mientras Costa Rica ha logrado una cobertura eléctrica del 99.4% de su población, otros países como Honduras muestran rezagos con solo el 77.2 % de accesibilidad eléctrica.

2.3. Transmisión

La transmisión de energía eléctrica es un proceso fundamental. Esta etapa del sistema eléctrico de potencia es la que se encarga de realizar la transferencia de electricidad a largas distancias desde las plantas de generación hasta los puntos de distribución a través de una red de infraestructura que está diseñada para transportar grandes cantidades de energía eléctrica a tensiones elevadas, minimizando las pérdidas de energía durante el proceso de transmisión. (Lumbreras and Ramos, 2016).

El transporte de energía se realiza a través de un sistema organizado de líneas de alta tensión y subestaciones que permiten transportar grandes cantidades de energía eléctrica desde las centrales generadoras hasta las áreas de distribución. La configuración de esta red se determina en función de la distancia que debe recorrer la electricidad y la cantidad de energía que se necesita transportar.

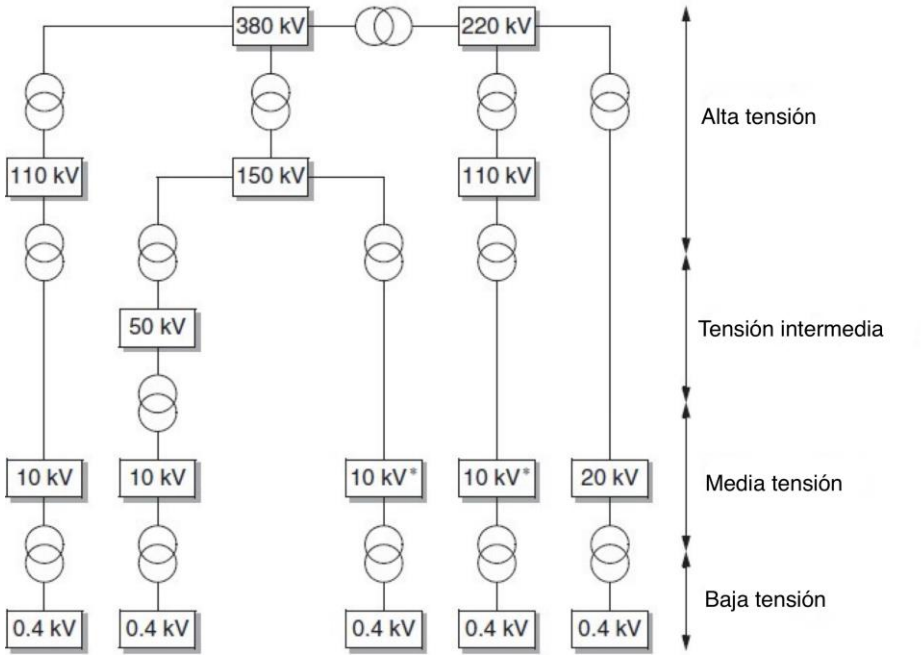
La red de transmisión, que opera a tensiones elevadas como 380 kV y 220 kV, conecta las centrales eléctricas principales con las subestaciones de transmisión. Estas subestaciones luego suministran energía a las redes de distribución que alimentan a las cargas finales, como hogares, industrias y comercios. En algunos casos, se establecen sistemas de subtransmisión para facilitar la transferencia de energía entre la red de transmisión principaly las redes de distribución, aunque a veces puede ser difícil definir claramente estos límites. (Schavemaker and van der Sluis, 2017).

El diseño y funcionamiento de una red de transmisión eléctrica evoluciona con el tiempo según las necesidades de energía y los avances tecnológicos. En general, las redes de transmisión de alta tensión como 380 kV, 220 kV y 180kV forman la columna vertebral del sistema, transportando la mayor parte de la energía generada. Otras redes, como las de 150 kV, 110 kV, 50 kV y 34.5kV cumplen funciones de transporte en niveles intermedios.

Los transformadores desempeñan un papel crucial al permitir la reducción de tensiones a niveles adecuados para la distribución a los consumidores finales. Las distancias entre subestaciones varían según el nivel de tensión, siendo más amplias para las estaciones de alta tensión y más cortas para las de distribución local. Este diseño modular y escalado garantiza la eficiencia y estabilidad del suministro eléctrico en todo el sistema de transmisión y distribución. En la figura 2.4 se presenta un ejemplo de niveles de tensión y pasos de transformación en un sistema eléctrico (específicamente sistema eléctrico holandés). (Schavemaker and van der Sluis, 2017).

Figura 2.4

Ejemplo de Niveles de Tensión y Pasos de Transformación en un Sistema Eléctrico (Específicamente Sistema Eléctrico Holandés)



Fuente: Schavemaker and van der Sluis (2017).

Por otra parte, como explica Gómez-Ramírez et al. (2023a), la transmisión de energía eléctrica en Centroamérica está marcada por la interconexión de los sistemas nacionales a través del Sistema de Interconexión Eléctrica de los Países de América Central (SIEPAC). Este sistema comprende una línea de transmisión de 1786 km que conecta a Guatemala, El Salvador, Honduras, Nicaragua, Costa Rica y Panamá, facilitando el intercambio de electricidad entre estos países. SIEPAC incluye 18 subestaciones y cuenta con una capacidad de transmisión de 300 MW en cada uno de sus dos circuitos principales. Esta infraestructura busca mejorar la estabilidad y seguridad del suministro eléctrico en la región, aunque enfrenta desafíos significativos como la variabilidad de las fuentes de energía renovable y la necesidad de mejoras en la gestión y almacenamiento de energía.

2.3.1. Transmisión en Costa Rica

La red de transporte de electricidad en Costa Rica desempeña un papel crucial en la distribución de energía desde las centrales de generación hacia los centros de carga. Esta red está diseñada para mantener la calidad y confiabilidad del suministro eléctrico, alineándose con los criterios económicos y ambientales establecidos en las políticas nacionales e institucionales de energía.

En Costa Rica la transmisión es gestionada por el Instituto Costarricense de Electricidad (ICE), que opera una red de transmisión extensa y robusta para distribuir electricidad a lo largo del país. Costa Rica se beneficia de una infraestructura bien desarrollada que facilita la integración de fuentes de energía renovable, permitiendo que más del 99 % de la electricidad del país provenga de estas fuentes. El sistema de transmisión incluye líneas de alta tensión que conectan las principales plantas generadoras, muchas de ellas hidroeléctricas, con los centros de consumo. (Gómez-Ramírez et al., 2023a).

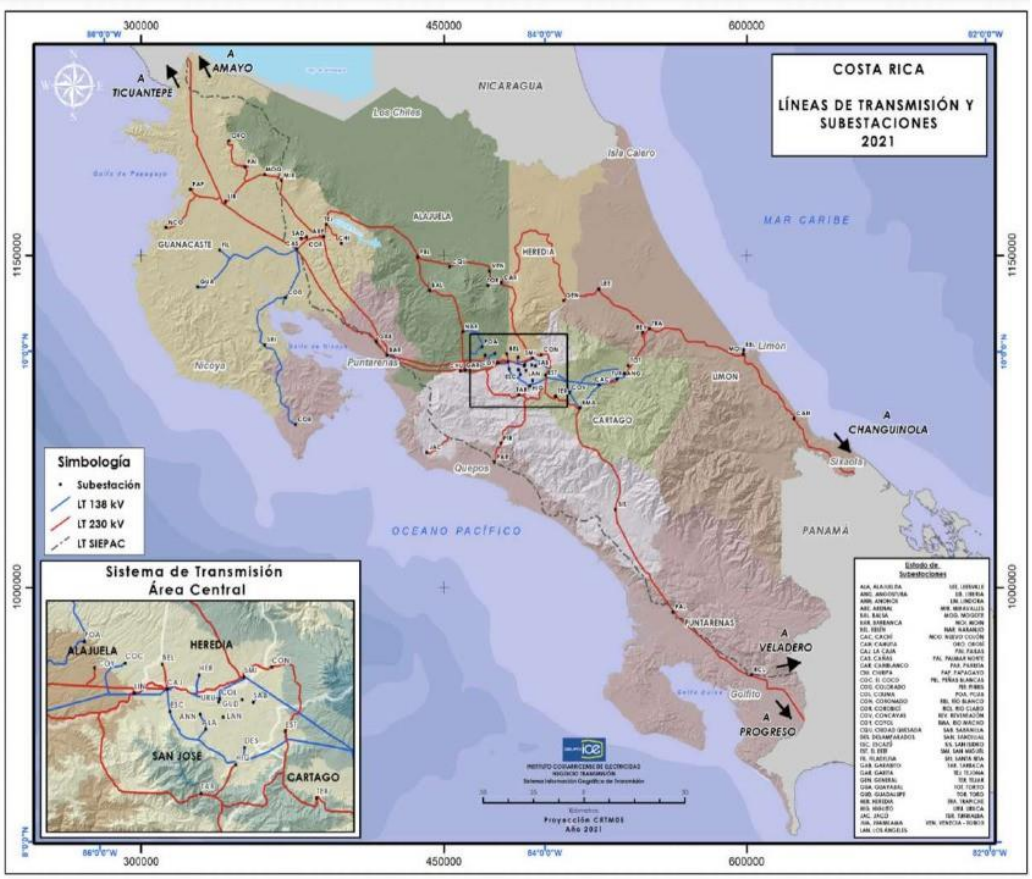
La red de transmisión costarricense, según el Instituto Costarricense de Electricidad, División Transmisión (2021), opera principalmente en dos niveles de tensión: 138kV y 230kV y tiene una cobertura que se extiende desde la Zona Norte hasta los centros de consumo en todo el país. La Zona Norte es clave, ya que concentra la mayoría de las centrales generadoras y transporta la energía al nivel de tensión de 230kV. Este nivel de tensión facilita el transporte eficiente de grandes cantidades de energía hacia los centros de consumo en otras partes del país. El transporte de energía desde la Zona Norte se lleva a cabo mediante cuatro corredores de líneas de 230 Kv y la energía se convierte a 138kV antes de alimentar las subestaciones principales.

La red eléctrica cubre geográficamente desde Peñas Blancas en el norte (frontera con Nicaragua) hasta Paso Canoas en el sur (frontera con Panamá), y desde Puerto Limón en el Caribe hasta Cóbano en la Península de Nicoya en el Pacífico. Esta extensa infraestructura atraviesa todas las provincias del país, desde zonas costeras hasta elevaciones de hasta 3500 metros sobre el nivel del mar, contando con aproximadamente 2986 km de líneas de transmisión en total, de las cuales el 82% de esta infraestructura pertenece al ICE, mientras que el 18 % restante está en manosde entes privados. (Instituto Costarricense de Electricidad, División Transmisión, 2021).

Además de abastecer las necesidades internas, la red de transmisión de Costa Rica facilita la interconexión a nivel regional a través de cinco interconexiones: dos con Nicaragua y tres con Panamá. Estas conexiones permiten la integración de nuevas fuentes de generación y la conexión de clientes de alta tensión, garantizando así el suministro eléctrico para las empresas de distribución que operan a tensiones nominales de 69 kV, 34.5 kV, 24.9 kV y 13.8 kV. En la figura 2.5 se observa la red de transmisión costarricense en el año 2021. (Instituto Costarricense de Electricidad, División Transmisión, 2021).

Figura 2.5

Red de Transmisión Año 2021



Fuente: Instituto Costarricense de Electricidad, División Transmisión (2021).

2.4. Distribución

Luego de la red de transmisión se tiene la red de distribución, la cual está diseñada para llevar la electricidad desde las subestaciones de distribución hasta los consumidores finales de manera eficiente. Este sistema se compone de diferentes elementos interconectados que trabajan en conjunto para asegurar la entrega confiable de la electricidad a través de varios niveles de tensión hasta alcanzar los usuarios. (Glover et al., 2015).

Las subestaciones de distribución son esenciales en este proceso, ya que reciben la electricidad de las subestaciones de subtransmisión y por medio de transformadores reductores disminuyen la tensión antes de distribuir la electricidad a través de los circuitos de distribución hacia los hogares, comercios e industrias; generalmente esta tensión se va a encontrar entre 2,2kV y 46 kV. Estos transformadores están conectados a través de disyuntores y sistemas de protección contra sobretensiones que aseguran un suministro eléctrico

estable y seguro a los consumidores. (Glover et al., 2015).

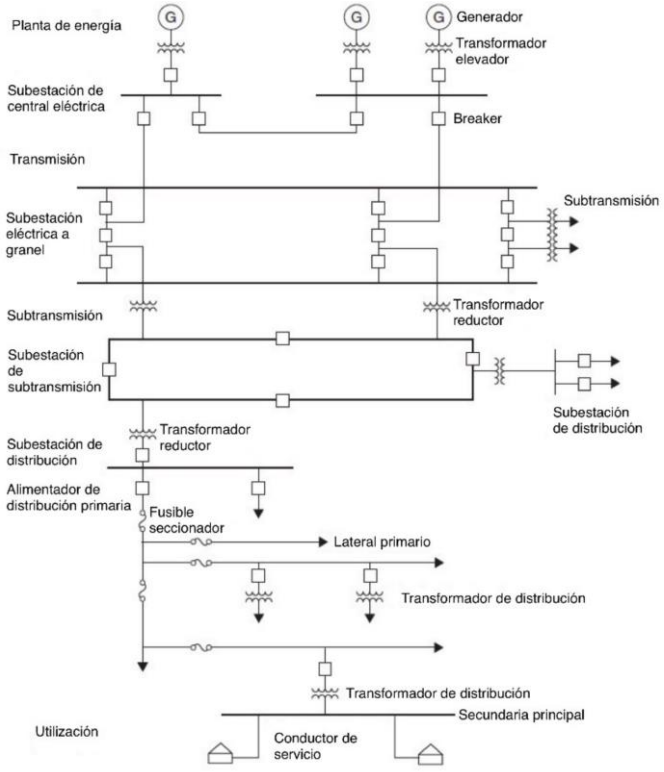
Desde las subestaciones de distribución, la electricidad se distribuye a través de líneas de distribución primaria y alimentadores hacia los clientes. Estas líneas pueden ser aéreas o subterráneas y su objetivo es transportar la electricidad desde las subestaciones hacia los transformadores de distribución ubicados en las proximidades de los usuarios finales. (Glover et al., 2015).

En los transformadores de distribución se reduce aún más la tensión para adecuarlo a las necesidades específicas de los consumidores, estas tensiones suelen estar entre los 120V a 480V. Estos transformadores, suelen tener capacidades que van desde 5 a 5000 kVA, se instalan en postes, plataformas o bóvedas y están equipados con sistemas de protección como fusibles y disyuntores para garantizar un suministro seguro de electricidad. (Glover et al., 2015).

A partir de los transformadores de distribución, la energía se entrega a los consumidores a través de líneas de servicio conectadas a medidores ubicados en las instalaciones de los clientes. Estos medidores registran el consumo de energía eléctrica para propósitos de facturación y control operativo, permitiendo así una gestión eficiente y transparente del suministro eléctrico a nivel local. (Glover et al., 2015). En la figura 2.6 se muestra un ejemplo gráfico de los componentes básicos de un eléctrico Sistema de Potencia.

Figura 2.6

Componentes Básicos de un Eléctrico Sistema de Potencia



Fuente: Glover et al. (2015).

2.4.1. Distribución Primaria

La distribución primaria es aquella que transporta tensiones entre 2.2kV a 34.5kV y existen 3 sistemas básicos: radial, de anillo y sistema de redes.

Sistema de Distribución Primario Radial

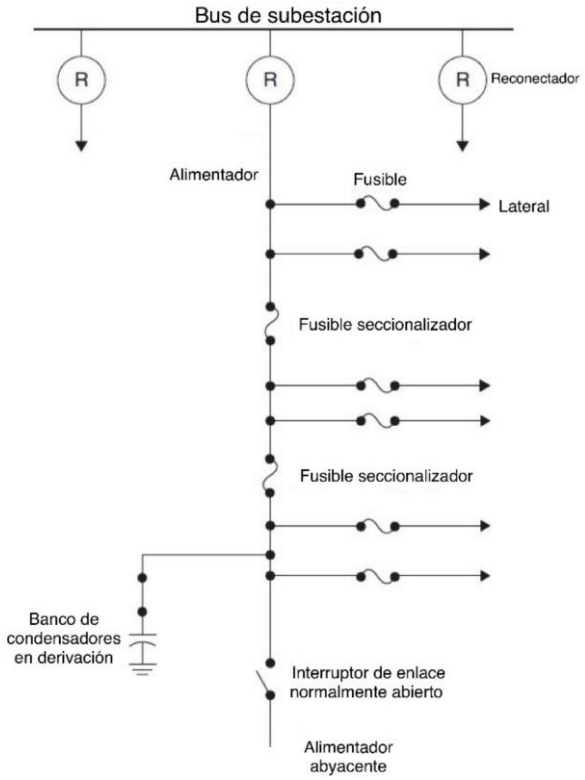
Un sistema de distribución primario radial es un método económico y comúnmente utilizado para la distribución de energía eléctrica en áreas de baja densidad de carga. En este sistema varios alimentadores principales trifásicos se extienden desde una subestación de distribución de manera radial, sirviendo cada uno a una zona geográfica específica. Por ejemplo, una subestación puede alimentar varios alimentadores que abastecen a diferentes áreas residenciales suburbanas. Estos alimentadores pueden tener longitudes que van desde una milla hasta varias decenas de millas y están diseñados con laterales monofásicos conectados a través de fusibles para permitir la desconexión de secciones individuales sin interrumpir el flujo de energía en todo el alimentador. (Glover et al., 2015).

Para minimizar las interrupciones, se utilizan dispositivos como reconectores automáticos ubicados estratégicamente a lo largo de los alimentadores. Estos dispositivos pueden identificar y corregir fallas temporales en la línea, como descargas eléctricas o contactos momentáneos, restableciendo automáticamente el flujo de energía después de una interrupción breve. Además, se emplean fusibles seccionadores y dispositivos de conexión automática entre alimentadores adyacentes para garantizar la continuidad del suministro durante emergencias, permitiendo el aislamiento y reparación rápidos de secciones con fallas. (Glover et al., 2015).

En los sistemas primarios también se utilizan bancos de condensadores para mejorar el factor de potencia y reducir las pérdidas de energía. Estos condensadores pueden ser controlados automáticamente para optimizar su funcionamiento según la carga eléctrica del área. Asimismo, se implementan programas informáticos para determinar la cantidad y ubicación óptimas de estos dispositivos, con el objetivo de mejorar la eficiencia y reducir costos operativos. Además, se pueden establecer alimentadores independientes para cargas críticas, como hospitales, asegurando un suministro continuo de energía en caso de interrupciones en el alimentador principal mediante sistemas de transferencia automática rápida y segura. (Glover et al., 2015). En la figura 2.7 se muestra un ejemplo de un sistema de distribución primario radial.

Figura 2.7

Sistema de Distribución Primario Radial



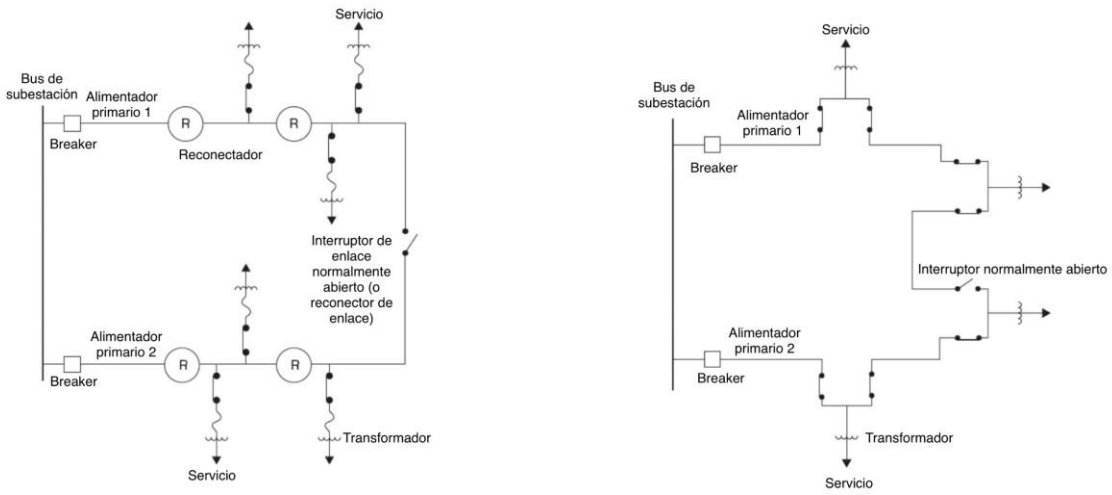
Fuente: Glover et al. (2015).

Sistema de Distribución Primario en Anillo

El sistema de distribución primario en anillo es un método utilizado especialmente en áreas donde la confiabilidad es crucial. En este sistema, un cable de alimentación rodea un área de carga y vuelve a la subestación de distribución, permitiendo la alimentación bidireccional desde la subestación. Los cables de alimentación mantienen un tamaño constante en todo el bucle para manejar la carga actual y futura. Se utilizan reconectadores y seccionadores para minimizar interrupciones en el suministro y aislar secciones del circuito con fallas. Cuando está en funcionamiento, un cliente recibe energía a través de una ruta específica desde la subestación, dependiendo del estado de los reconectadores. En circuitos subterráneos el tamaño del cable también se selecciona para soportar la carga actual y futura. Aunque las fallas en los alimentadores subterráneos son menos comunes que en los aéreos, suelen ser permanentes y requieren tiempo para localizar y reparar. (Glover et al., 2015). En la figura 2.8 (a) se muestra un sistema de distribución en anillo aéreo y en 2.8 (b) se muestra un sistema de distribución en anillo subterráneo.

Figura 2.8

Sistema de Distribución en Anillo (a) Aéreo y (b) Subterráneo.



(a) Sistema de distribución en anillo aéreo. (b) Sistema de distribución en anillo subterráneo.

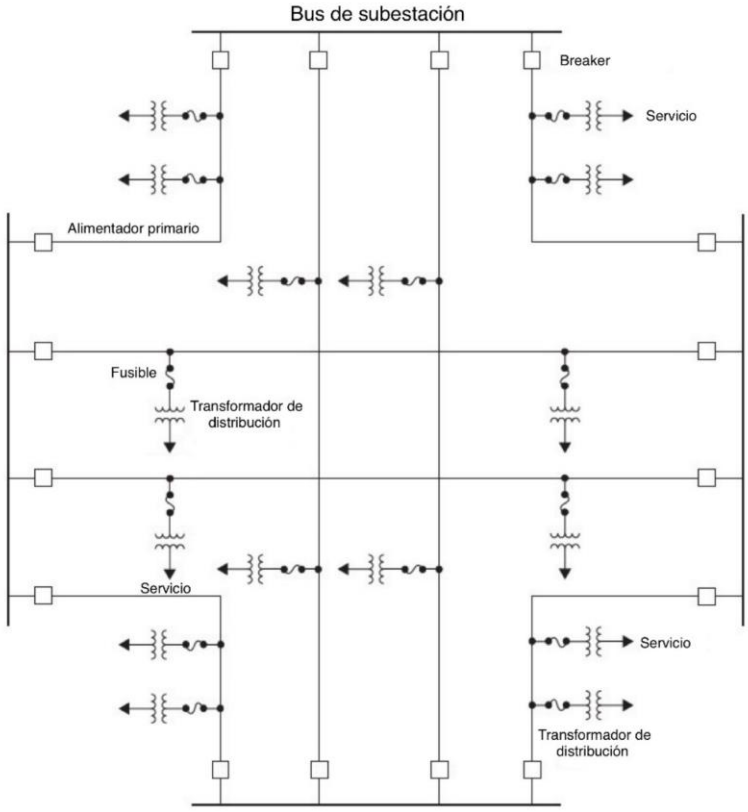
Fuente: Glover et al. (2015).

Sistema de Distribución Primario en Red

Estas redes se ubican generalmente en zonas centrales de grandes ciudades con alta densidad de consumo eléctrico. El sistema de distribución primario en red consiste en una red de alimentadores interconectados que reciben energía de varias subestaciones. Los reguladores de tensión en las subestaciones y transformadores cerca de los centros de carga principales tienden a mantener los niveles adecuados de tensión en los puntos de uso; sin embargo, mantener una tensión estable en toda la red bajo diversas condiciones es complicado; por esta razón actualmente dicha configuración es poco usada. En la figura 2.9 se muestra un ejemplo de esta configuración.

Figura 2.9

Sistema de Distribución Primario en Red



Fuente: Glover et al. (2015).

En Costa Rica, el sistema de distribución de energía está diseñado de forma radial, lo que significa que está configurado para dirigir el flujo de energía en una sola dirección. Sin embargo, con la implementación de la nueva Ley 10086 y el uso de generación distribuida se requieren flujos bidireccionales de energía. Por lo tanto, surge la necesidad de implementar sistemas de distribución enmallados o en anillo que puedan acomodar estos flujos en ambas direcciones de manera eficiente y segura. (Geo Ingeniería Ingenieros Consultores S.A., 2024).

2.4.2. Distribución Secundaria

La distribución eléctrica secundaria es la parte de la infraestructura eléctrica de potencia que se encarga de llevar la energía eléctrica desde los transformadores de distribución hasta los puntos de consumo, como hogares, negocios e industrias. En la tabla 2.1 se muestra una serie de tensiones típicas según Glover et al. (2015) en líneas de distribución secundarias y sus aplicaciones. En la tabla 2.2 se pueden observar los niveles típicos de tensión Costa Rica según Resolución RJD- 070-2015 (2015):

Tabla 2.1

Tensiones Típicas de Distribución Secundaria

Tensión	Número de fases	Número de cables	Aplicación
120/240V	Monofásico	Tres	Residencial
208Y/120V	Trifásico	Cuatro	Residencial/Comercial
480Y/277V	Trifásico	Cuatro	Comercial/Industrial

Fuente: Glover et al. (2015).

Tabla 2.2

Tensiones Típicas de Distribución Secundaria en Costa Rica

	Tensión	
Sistema	Entre líneas activas (V)	Entre líneas activas y neutro (V)
Monofásico bifilar[1].	-	120
Monofásico trifilar	240	120
Bifásico trifilar	208	120
Trifásico, 4 conductores[2]	208	120
Trifásico, 4 conductores[3]	480	277
Trifásico, 3 conductores[4]	240	-
Trifásico, 4 conductores[5]	240	120
Trifásico, 3 conductores[4]	480	-
Trifásico, 4 conductores[5]	480	240

Fuente: Resolución RJD-070-2015 (2015).

2.5. Componentes de una Red de Distribución

2.5.1. Líneas de distribución de aéreas

En los sistemas de distribución eléctrica, las líneas pueden clasificarse como monofásicas, bifásicas y trifásicas, estás líneas son no transpuestas y se utilizan para alimentar cargas desequilibradas. Al estas líneas ser no transpuestas es importante mantener la integridad de los términos de impedancia propia y mutua de los conductores y considerar la ruta de retorno a tierra para las corrientes desequilibradas para asegurar un funcionamiento eficiente y seguro del sistema, minimizando pérdidas y garantizando una distribución estable de la energía eléctrica. (Kersting, 2001).

Las ecuaciones de Carson son fundamentales para comprender el comportamiento de las líneas de distribución. Estas ecuaciones describen cómo las características de impedancia y capacitancia de los conductores afectan la distribución de la corriente y la tensión a lo largo de la línea. En líneas no transpuestas, estas ecuaciones son especialmente relevantes porque cada fase o conductor tiene su propia relación con respecto de la tierra y las interacciones entre los conductores afectan significativamente el desempeño global

del sistema. La no transposición de las líneas implica que las características eléctricas individuales de cada conductor tienen un impacto más directo en la distribución de la corriente y la tensión, lo que requiere un análisis más detallado de la impedancia y la capacitancia. (Kersting, 2001). A continuación, se presentan las ecuaciones modificadas de Carson.

La impedancia serie de un conductor se calcula con la siguiente ecuación 2.1:

$$Z_{ii} = r_{ii} + 0{,}059216 + j0{,}075397\left(\ln\left(\frac{0{,}3048}{GRM_i}\right) + 7{,}93402\right) \tag{2.1}$$

$$Z_{ii} = 0{,}059216 + j0{,}075397\left(\ln\left(\frac{0{,}3048}{D_{ij}}\right) + 7{,}93402\right) \tag{2.2}$$

Donde:

Z_{ii}: es la impedancia de línea en Ω/km.

r_{ii}: resistencia ca del conductor i en Ω/km.

GRM_i: radio medio geométrico en m.

D_{ij}: distancia entre el conductor i y el conductor j en m.

Kersting (2001), explica en su libro que a partir de esta ecuación se puede calcular las impedancias propias y mutuas de líneas aéreas para obtener una matriz de impedancias primitiva.

$$[\hat{Z}_{primitive}] = \left[\begin{array}{ccc|ccc} \hat{Z}_{aa} & \hat{Z}_{ab} & \hat{Z}_{ac} & \hat{Z}_{a\,n1} & \hat{Z}_{a\,n2} & \hat{Z}_{a\,nm} \\ \hat{Z}_{ba} & \hat{Z}_{bb} & \hat{Z}_{bc} & \hat{Z}_{b\,n1} & \hat{Z}_{b\,n2} & \hat{Z}_{b\,nm} \\ \hat{Z}_{ca} & \hat{Z}_{cb} & \hat{Z}_{cc} & \hat{Z}_{c\,n1} & \hat{Z}_{c\,n2} & \hat{Z}_{c\,nm} \\ \hline \hat{Z}_{n1a} & \hat{Z}_{n1b} & \hat{Z}_{n1c} & \hat{Z}_{n1\,n1} & \hat{Z}_{n1\,n2} & \hat{Z}_{n1\,nm} \\ \hat{Z}_{n2a} & \hat{Z}_{n2b} & \hat{Z}_{n2c} & \hat{Z}_{n2\,n1} & \hat{Z}_{n2\,n2} & \hat{Z}_{n2\,nm} \\ \hat{Z}_{nma} & \hat{Z}_{nmb} & \hat{Z}_{nmc} & \hat{Z}_{nm\,n1} & \hat{Z}_{nm\,n2} & \hat{Z}_{nm\,nm} \end{array}\right] \tag{2.3}$$

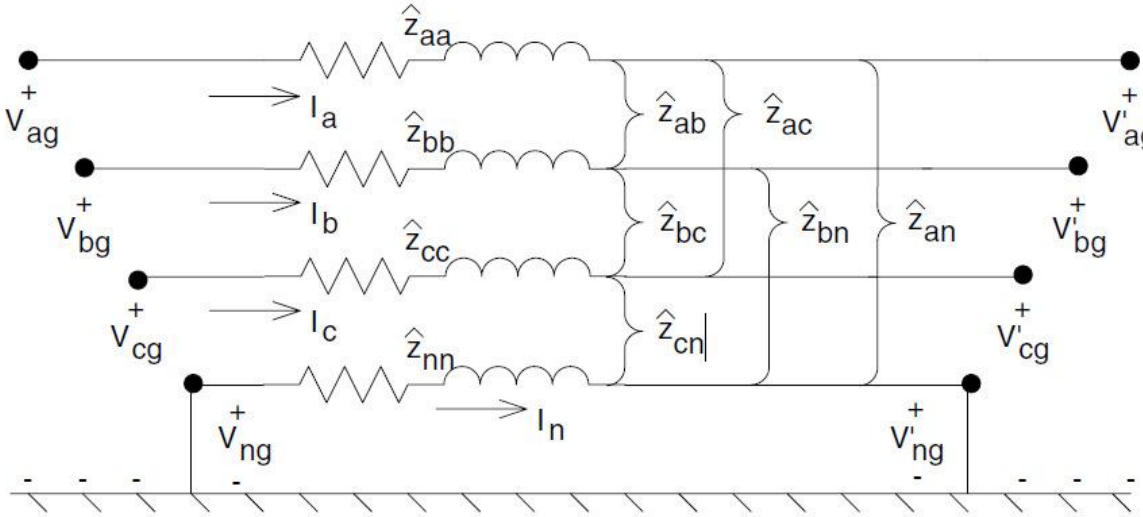
En su forma dividida es:

$$[\hat{Z}_{primitive}] = \begin{bmatrix} [\hat{Z}_{ll}] & [\hat{Z}_{ln}] \\ [\hat{Z}_{nl}] & [\hat{Z}_{nn}] \end{bmatrix} \tag{2.4}$$

En la mayoría de las aplicaciones, se necesita simplificar la matriz de impedancia original a una matriz de 3x3 que representa las impedancias equivalentes entre las tres fases; un método común de simplificación es la reducción de Kron. En la figura 2.10 se representa una línea con conexión a tierra utilizando cuatro cables, suponiendo que el neutro tiene una conexión a tierra múltiple y en la figura 2.11 se encuentra la misma línea luego de aplicar la simplificación de Kron.

Figura 2.10

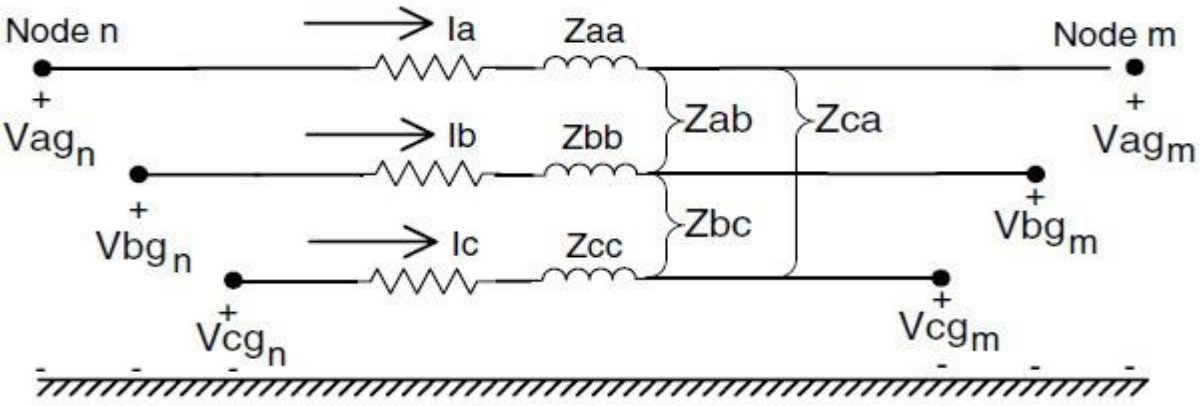
Línea con Conexión a Tierra Utilizando Cuatro Cables



Fuente: Kersting (2001).

Figura 2.11

Línea Luego de la Simplificación de Kron



Fuente: Kersting (2001).

Kersting (2001) explica que para realizar la reducción de Kron se debe aplicar la ley de tensión de Kirchhoff en el circuito. En la ecuación 2.5 se puede ver esta aplicación.

$$\begin{bmatrix} V_a \\ V_b \\ V_c \\ V_n \end{bmatrix} = \begin{bmatrix} V'_a \\ V'_b \\ V'_c \\ V'_n \end{bmatrix} + \begin{bmatrix} z_{aa} & z_{ab} & z_{ac} & z_{an} \\ z_{ba} & z_{bb} & z_{bc} & z_{bn} \\ z_{ca} & z_{cb} & z_{cc} & z_{cn} \\ z_{na} & z_{nb} & z_{nc} & z_{nn} \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \\ I_n \end{bmatrix} \tag{2.5}$$

En su forma simplificada la ecuación pasa a ser:

$$\begin{bmatrix} [V_{an}] \\ [V_{n1}] \end{bmatrix} = \begin{bmatrix} [V'_{an}] \\ [V'_{n1}] \end{bmatrix} + \begin{bmatrix} [z_{11}] & [z_{12}] \\ [z_{21}] & [z_{22}] \end{bmatrix} \begin{bmatrix} [I_{an}] \\ [I_{n1}] \end{bmatrix} \tag{2.6}$$

Debido a que el neutro está conectado a tierra, las tensiones V_n y $V_n{}'$ son iguales a cero; sustituyendo esos valores en la ecuación 2.6 y expandiendo se obtiene:

$$[V_{an}] = [V'_{an}] + [z_{11}] \cdot [I_{an}] + [z_{12}] \cdot [I_{n1}] \tag{2.7}$$

$$[0] = [0] + [z_{21}] \cdot [I_{an}] + [z_{22}] \cdot [I_{n1}] \tag{2.8}$$

Resolviendo la ecuación 2.8 para $[I_{n1}]$ se obtiene:

$$[I_{n1}] = -[z_{22}]^{-1} \cdot [z_{21}] \cdot [I_{an}] \tag{2.9}$$

Sustituyendo la ecuación 2.9 dentro de la ecuación 2.7, queda como resultado la ecuación 2.10:

$$\begin{aligned} [V_{an}] &= [V'_{an}] + ([z_{11}] - [z_{12}] \cdot [z_{22}]^{-1} \cdot [z_{21}]) \cdot [I_{an}] \\ &= [V'_{an}] + [z_{abc}] \cdot [I_{an}] \end{aligned} \tag{2.10}$$

Donde:

$$[z_{abc}] = [z_{11}] - [z_{12}] \cdot [z_{22}]^{-1} \cdot [z_{21}] \tag{2.11}$$

La ecuación 2.11 es la forma final de la reducción de Kron y la matriz de impedancia trifásica final se convierte en:

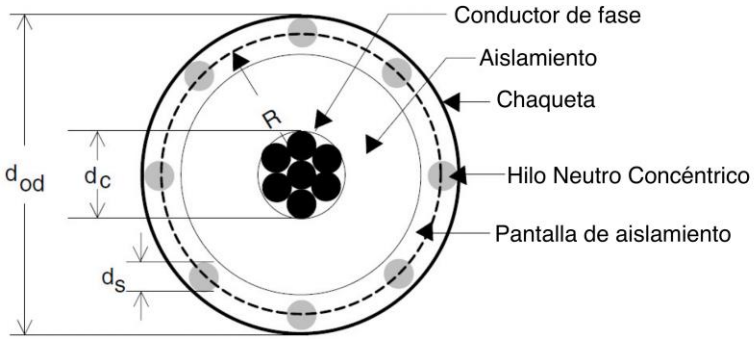
$$[z_{abc}] = \begin{bmatrix} z_{aa} & z_{ab} & z_{ac} \\ z_{ba} & z_{bb} & z_{bc} \\ z_{ca} & z_{cb} & z_{cc} \end{bmatrix} \Omega/\text{km} \tag{2.12}$$

2.5.2. Líneas de Distribución de Subterráneas

Para las líneas de distribución subterráneas se utilizan las mismas ecuaciones de Carson y la simplificación de Kron. Sin embargo, para poder utilizar estas ecuaciones se debe conocer la resistencia y el GMR (Radio Geométrico Medio) del conductor de fase y el equivalente del neutro. En las líneas subterráneas es común utilizar un cable neutro concéntrico; en la figura 2.12 se muestra un diagrama de este tipo de cable, que consiste en un conductor de fase central cubierto por una delgada capa de pantalla semiconductora no metálica, aislada por un material aislante y otra pantalla semiconductora aislante. Los hilos sólidos del neutro concéntrico están dispuestos en espiral alrededor de la pantalla semiconductora con un espacio uniforme entre ellos.

Figura 2.12

Diagrama de Cable Neutro Concéntrico



Fuente: Kersting (2001).

Kersting (2001) explica que para aplicar las ecuaciones de Carson a este cable es necesario extraer los siguientes datos de una tabla de cables subterráneos:

d_c = diámetro del conductor de fase (m).

d_{od} = diámetro nominal sobre los neutros concéntricos del cable (m).

d_s = diámetro de un hilo neutro concéntrico (m).

GMR_c = radio medio geométrico del conductor de fase (m).

GMR_s = radio medio geométrico de un hilo neutro (m).

r_c = resistencia del conductor de fase Ω/km.

r_n = resistencia de un hilo neutro sólido Ω/km.

k = número de hilos neutros concéntricos.

Luego el radio medio geométrico equivalente del neutro concéntrico se calcula utilizando la ecuación 2.13.

$$\mathrm{GMR}_{cn} = t\left(\mathrm{GMR}_c \cdot k^{\frac{1}{k}}\right)\ \mathrm{m} \tag{2.13}$$

donde:

t = radio de un círculo que pasa por el centro de los hilos neutros concéntricos

$$t = \frac{d_c + d_n}{24}\ \mathrm{m} \tag{2.14}$$

La resistencia equivalente del neutro concéntrico es:

$$r_{cn} = \frac{r_n}{k}\ \Omega/\mathrm{milla} \tag{2.15}$$

2.5.3. Transformadores

Los transformadores son dispositivos clave en la distribución eficiente de energía eléctrica, debido a que son los encargados de convertir las tensiones de un nivel a otro. Según Short (2004), un transformador típico consta de dos conjuntos de bobinas interconectadas a través de un campo magnético. Este campo magnético facilita la transferencia de energía eléctrica, teniendo pérdidas mínimas en el proceso de transferencia.

Los transformadores de distribución transforman la tensión primaria a un nivel más bajo apto para el consumo de los clientes, típicamente oscilando entre unos pocos kVA a varios MVA. La relación de vueltas del transformador determina la relación entre las tensiones de entrada y salida en un transformador ideal. La configuración de conexión del transformador también es crucial, debido a que define las tensiones disponibles para los clientes y la configuración de conexión a tierra, garantizando un suministro eléctrico seguro y eficiente. (Short, 2004). Los transformadores, dependiendo de las necesidades específicas de distribución y del entorno, pueden ser tanto monofásicos como trifásicos.

Transformadores Monofásicos

Los transformadores monofásicos, como su nombre lo indica funcionan en una sola fase, estos tienen un diseño básico que consiste en dos devanados, el primario y el secundario, que están acoplados magnéticamente a través de un núcleo de hierro y la relación de vueltas entre los devanados determina la relación de transformación, permitiendo la conversión de tensiones a niveles adecuados para distribución. (Short, 2004).

Los transformadores monofásicos pueden utilizarse de manera individual para alimentar cargas monofásicas o combinarse estratégicamente para suministrar energía trifásica al sistema. (Short, 2004). Por ejemplo, al usar dos transformadores monofásicos, se puede configurar una conexión delta (Δ) o estrella (Y) para lograr una configuración trifásica. También es común utilizar tres unidades monofásicas, conectadas apropiadamente, para lograr una configuración trifásica equilibrada que cumpla con las necesidades de cargas industriales y comerciales.

La nomenclatura y las designaciones de los transformadores monofásicos son cruciales para entender cómo conectar y operar estos dispositivos de manera segura y eficiente. Las tablas 2.3 y 2.4 presentadas muestran ejemplos de cómo se etiquetan y describen diferentes configuraciones de devanados para transformadores monofásicos según IEEE (2000), permitiendo una fácil identificación de las características de tensión, conexiones y capacidades de operación.

Tabla 2.3

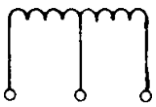
Designaciones de Devanados para Transformadores Monofásicos Primarios y Secundarios con un Devanado

Nomenclatura	Marcado de placa de identificación y diagrama de bobinado típico	Descripción
E	13800 	E indicará un devanado de E voltios que es adecuado para conexión Δ en un sistema de E voltios.
E/E_1Y	2400/4160Y 	E/E_1Y indicará un devanado de E voltios que es adecuado para conexión Δ en un sistema de E voltios o para conexión Y en un sistema de E_1 voltios.
E/E_1GrdY	7200/12470GrdY 	E/E,GrdY indicará un devanado de E voltios con aislamiento reducido que es adecuado para conexión Δ en un sistema de E voltios o conexión Y en un sistema de E_1 voltios, transformador, neutro efectivamente aterrizado.
E_1GrdY/E	12470GrdY/7200 480GrdY/277 	E_1GrdY/E indicará un devanado de E voltios con aislamiento reducido en el extremo neutro. El extremo neutro puede conectarse directamente al tanque para Y o para operación monofásica en un sistema de E_1 voltios, siempre que el extremo neutro del devanado esté efectivamente aterrizado.
$E_1 = \sqrt{3}E$		

Fuente: IEEE (2000).

Tabla 2.4

Designaciones de Transformadores de Dos Devanados para Primarios y Secundarios Monofásicos

Nomenclatura	Marcado de placa de identificación y diagrama de bobinado típico	Descripción
E/2E	120/240 	E/2E indicará un devanado, cuyas secciones pueden conectarse en paralelo para operación a E voltios, o pueden conectarse en serie para operación a 2E voltios o conectarse en serie con un terminal central para operación trifásica a 2E voltios entre los terminales extremos y E voltios entre el terminal central y cada uno de los terminales extremos.
2E/E	240/120 	2E/E indicará un devanado para 2E voltios, dos hilos kilovoltamperios completos entre terminales extremos o para servicio trifásico 2E/E voltios con 1/2 kVA disponible solamente, desde el punto medio a cada terminal extremo.
E × 2E	240 × 480 	E × 2E indicará un devanado solo para operación en paralelo o serie, pero no apto para servicio trifásico.

Fuente: IEEE (2000).

Transformadores Trifásicos

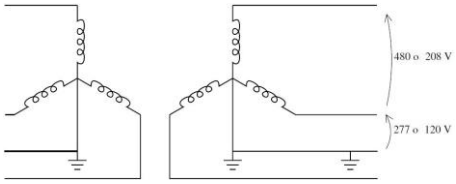
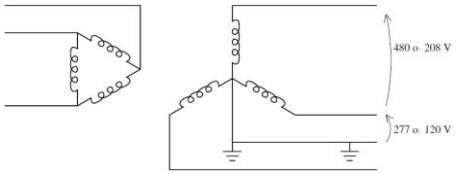
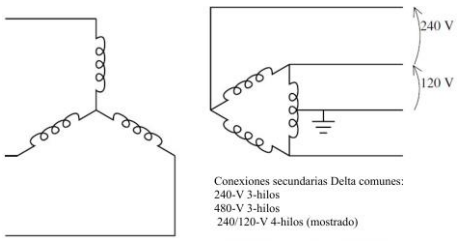
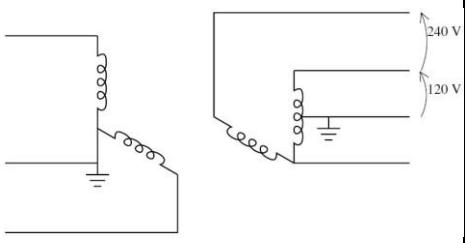
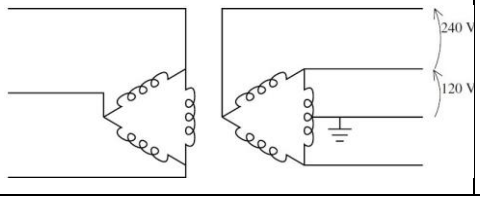
Los transformadores trifásicos pueden variar en diseño y configuración dependiendo de su aplicación específica: ya sea para servicio aéreo o subterráneo. Short (2004) explica que los transformadores aéreos trifásicos generalmente se construyen utilizando tres unidades monofásicas, mientras que los transformadores trifásicos destinados al servicio subterráneo suelen ser unidades individuales.

Existen diferentes configuraciones de conexión de devanados en transformadores trifásicos, como delta Δ, estrella (Y), delta aterrizada o estrella aterrizada. Estas configuraciones no solo afectan la distribución de la tensión, sino que también influyen en la forma en que el transformador opera y se conecta en el sistema eléctrico (Short, 2004). En la tabla 2.5 se presenta un listado general de diversas configuraciones de conexión de devanados en transformadores trifásicos, enfocándose en sus representaciones gráficas y características particulares, es importante recalcar que estas configuraciones describen la conexión de los devanados del transformador, no la configuración del sistema de suministro.

Por otra parte, en la tabla 2.6 se presentan varias designaciones comunes de transformadores trifásicos, junto con sus correspondientes marcados de placa y descripciones detalladas de su configuración y aplicaciones específicas. Estas designaciones permiten identificar rápidamente cómo están conectados internamente los devanados del transformador y para qué condiciones de tensión están diseñados

Tabla 2.5

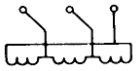
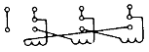
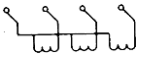
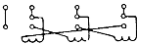
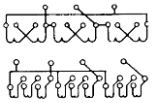
Tipos de Conexión de Devanado del Transformador

Tipo de Conexión	**Representación Gráfica**	**Características**
Estrella aterrizada-Estrella aterrizada		▪ Tipo de Conexión: esta conexión requiere un sistema de cuatro cables con conexión a tierra tanto en la entrada como en la salida. ▪ Aplicación: esta configuración está diseñada para suministrar servicios de tensión típicamente de 480Y/277 V o 208Y/120 V. No puede suministrar tensiones de 120 y 240 V.
Delta-Estrella aterrizada		▪ Tipo de Conexión: la conexión puede funcionar tanto en sistemas de tres como de cuatro cables. ▪ Aplicación: esta configuración está diseñada para suministrar servicios de tensión típicamente de 480Y/277 V o 208Y/120 V. No puede suministrar simultáneamente tensiones de 120 V y 240V.
Estrella flotante-Delta		▪ Configuración: Proporciona una salida de tensión en configuración delta en el lado secundario, permitiendo flexibilidad en la conexión de carga. ▪ Aplicación: Ideal para suministrar tensiones a clientes que requieren un servicio de tres o cuatro cables sin conexión a tierra, con tensiones de 120 y 240 V.
Estrella abierta-Delta abierta		▪ Tipo de Conexión: esta conexión puede suministrar carga trifásica desde un suministro de dos fases (pero el suministro debe tener un neutro). ▪ Aplicación: Puede suministrar servicio no conectado a tierra, así como servicio de cuatro cables con 240/120V con conexión a tierra en el centro.
Delta - Delta		▪ Tipo de Conexión: La configuración Delta – Delta es compatible con sistemas de tres o cuatro cables. ▪ Aplicación: Puede suministrar servicio no conectado a tierra, así como servicio de cuatro cables con 240/120 V.

Fuente: Short (2004).

Tabla 2.6

Designaciones de Transformadores Trifásicos

Nomenclatura	Marcado de placa de identificación y diagrama de bobinado típico	Descripción
E	2400 	E indica un devanado permanentemente conectado en Δ para operar en un sistema de tensión E.
E_1Y	4160Y 	E_1Y indica un devanado permanentemente conectado en estrella sin el neutro disponible (aislado) para operar en un sistema de tensión E_1.
E_1Y/E	4160Y/2400 	E_1Y/E indica un devanado permanentemente conectado en estrella sin el neutro disponible para operar en un sistema de tensión E_1, con tensión E disponible de línea a neutro.
E/E_1Y	2400/4160Y 	E/E_1Y indica un devanado que puede estar conectado en Δ para operar en un sistema de tensión E o puede estar conectado en estrella sin el neutro disponible (aislado) para operar en un sistema de tensión E_1.
$E/E_1Y/E$	2400/4160Y/2400 	$E/E_1Y/E$ indica un devanado que puede estar conectado en Δ para operar en un sistema de tensión E o puede estar conectado en estrella con un neutro completamente aislado para operar en un sistema de tensión E_1 con tensión E disponible de línea a neutro.
E_1GrdY/E	12470GrdY/7200 	E_1GrdY/E indica un devanado con aislamiento reducido y conectado permanentemente en estrella, con un neutro aislado y efectivamente a tierra para operar en un sistema de tensión E_1 con tensión E disponible de línea a neutro.
$E/E_1GrdY/E$	7200/12470GrdY/7200 	$E/E_1GrdY/E$ indica un devanado, con aislamiento reducido, que puede estar conectado en Δ para operar en un sistema de tensión E o puede estar conectado en estrella con el neutro aislado y efectivamente a tierra para que operaren un sistema de tensión E_1 con tensión E disponible de línea a neutro.
$V \times V_1$	7200 × 14400 	$V \times V_1$ indica un devanado, cuyas secciones pueden estar conectadas en paralelo para obtener una de las tensiones (definidas en a-g) de V o pueden estar conectadas en serie para obtener una de las tensiones (definidas en a-g) de V_1, o están permanentemente conectadas en Δ o Y.

Fuente: IEEE (2000).

2.5.4. Cargas

Una carga eléctrica en términos generales se puede definir como cualquier dispositivo que consume energía eléctrica del sistema y estos dispositivos pueden clasificarse en diferentes categorías según su función, por ejemplo, motores, equipos de calentamiento, equipos electrónicos y los equipos de iluminación. Para los

sistemas de potencia las cargas se definen en función a los usuarios, es por esto que se agrupan en tres grandes categorías: residenciales, comerciales e industriales. (Gallego Rendón et al., 2016).

En sistemas eléctricos de mayor escala, como los que operan a niveles de megavatios, las cargas presentan ciertas características distintivas, estas cargas suelen utilizar las tres fases del sistema eléctrico y son altamente predecibles en cuanto a su magnitud y variación horaria. Además, la variación entre periodos consecutivos es generalmente previsible y lenta en comparación con las constantes de tiempo del sistema eléctrico. (Gallego Rendón et al., 2016).

Las cargas típicas en sistemas de potencia consumen potencia reactiva debido a la presencia de máquinas eléctricas rotativas; existe una parte del consumo que depende del nivel de tensión, como es el caso de las cargas de calefacción, mientras que otra parte es independiente de este nivel, como en las cargas motorizadas. Asimismo, se asume que las cargas son simétricas y balanceadas entre las fases del sistema eléctrico, aunque la conexión de cargas monofásicas o asimétricas puede resultar en un sistema balanceado en términos de corrientes. (Gallego Rendón et al., 2016).

Para los estudios de sistemas de potencia se suelen utilizar varios modelos de carga fundamentales que permiten entender y simular el comportamiento de las cargas eléctricas en el sistema. Entre los modelos más importantes se encuentran el modelo de inyección de potencia constante, el modelo de corriente constante y el modelo de impedancia constante.

Modelo de Inyección de Potencia Constante

Este modelo de carga es ampliamente utilizado en análisis de flujos de potencia en sistemas eléctricos estacionarios. Representa consumos grandes desde subestaciones, donde la potencia activa (P) y reactiva (Q) de la carga se calculan como la suma de consumos de usuarios residenciales, comerciales e industriales conectados a una barra de subestación. Los valores de P y Q se consideran constantes durante el período de análisis y se derivan a partir de mediciones en la subestación y datos estadísticos de demanda. Esta es la forma más común de representar la carga en estudios de flujos de potencia. (Gallego Rendón et al., 2016).

Modelo de Corriente Constante

Aunque menos utilizado en cargas agregadas, este modelo es esencial para estudios de armónicos en sistemas eléctricos. La magnitud de la corriente se mantiene constante, lo que es útil para estudios específicos sobre distorsión armónica y comportamiento de la red eléctrica. (Gallego Rendón et al., 2016).

Modelo de Impedancia Constante

Este modelo es fundamental para estudios de estabilidad transitoria en redes de distribución de media y baja tensión. Para este modelo de carga, se supone que P y Q de la carga permanece constante. (Gallego Rendón et al., 2016). Estos modelos agregados respaldan la representación simplificada de la carga en sistemas eléctricos.

En conjunto, estos modelos estacionarios genéricos representan diferentes formas de caracterizar la carga eléctrica, considerando su comportamiento en relación con la tensión y la frecuencia de la red. La elección del modelo adecuado depende del tipo de análisis que se esté realizando y de la precisión requerida para la representación de la carga en el sistema eléctrico.

2.5.5. Reguladores de Tensión

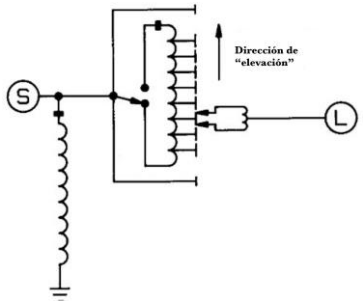
Un regulador de tensión en un sistema de distribución eléctrica es un dispositivo diseñado para mantener la tensión de salida dentro de un rango de funcionamiento predefinido, compensando las variaciones en la carga o en las condiciones de la red eléctrica. (Glover et al., 2015). Este dispositivo es fundamental para garantizar que los usuarios finales reciban un suministro eléctrico estable y dentro de los estándares de seguridad y calidad.

Matulic (2003) explica que el regulador de tensión funciona monitoreando constantemente la tensión de salida de la red eléctrica. Cuando se detectan variaciones en la tensión debido a cambios en la carga o en otras condiciones, el regulador ajusta automáticamente su configuración para mantener la tensión de salida en el nivel deseado. (Kersting, 2001).

Los reguladores de tensión utilizados en este proyecto son reguladores de tensión por pasos. Estos funcionan con un autotransformador y un mecanismo que permite cambiar las tomas de carga para modificarla tensión. Los reguladores estándar por pasos incluyen un interruptor reversible que permite un rango de regulación de ±10 %, usualmente en 32 pasos. Esto equivale a un cambio de la tensión de 0.75 V por paso en una base de 120 V (Kersting, 2001). Los reguladores por pasos pueden ser conectados según las especificaciones Tipo A o Tipo B de la norma ANSI/IEEE C57.15-1986. La configuración Tipo B, mostrada en la Figura 2.14, es la más común; sin embargo, en este proyecto se utilizaron regulares tipo A, representados en la figura 2.13.

Figura 2.13

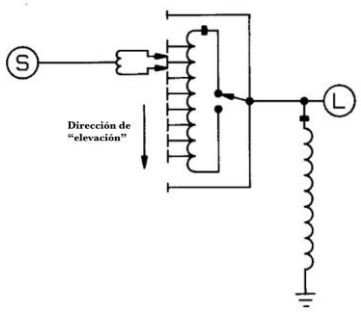
Regulador de Paso Tipo A



Fuente: IEEE Power Engineering Society (1988).

Figura 2.14

Regulador de Paso Tipo B



Fuente: IEEE Power Engineering Society (1988).

Algunos conceptos importantes, según Kersting (2001), de los autotransformadores son:

1. Nivel de tensión: la tensión deseada (sobre una base de 120 V) que debe mantenerse en el centro de carga, ya sea en la terminal de salida del regulador o en un nodo remoto del alimentador.

2. Ancho de Banda: la variación permitida de la tensión en el centro de carga con respecto del nivel establecido. La tensión se mantendrá dentro de ± la mitad del ancho de banda alrededor del nivel establecido. Por ejemplo, si el nivel se fija en 122 V y el ancho de banda es de 2 V, el regulador ajustará las tomas hasta que la tensión en el centro de carga esté entre 121 y 123 V.

3. Retardo de Tiempo: la duración entre la solicitud de un cambio de toma y su ejecución real, para evitar ajustes durante transitorios o cambios rápidos de corriente.

4. Compensador de Caída de Línea: ajuste para compensar la caída de tensión entre el regulador y el centro de carga, con configuraciones específicas de R y X en voltios que reflejan la impedancia entre ambos puntos.

2.6. Software de Simulación Eléctrica para Sistemas de Potencia

Un software de simulación eléctrica de potencia es una herramienta informática especializada, utilizada para analizar sistemas eléctricos de potencia (Hay and Ferguson, 2015). Algunos de estos programas incluyen DINIS, IPSA, PSS/E, ETAP y DIgSILENT, que permiten llevar a cabo una variedad de estudios, como análisis de estado estacionario, armónicos y fenómenos transitorios (EMT). Estos estudios abarcan análisis de flujo de carga, capacidad de potencia reactiva, evaluación de contingencias, niveles de falla, análisis armónico, estabilidad transitoria y respuesta a fallas. Tales análisis son fundamentales para la planificación de redes eléctricas y conexiones de generación, especialmente en las redes de distribución. En estas últimas, el estudio de flujos inversos de potencia es esencial para la gestión y protección de la tensión, estabilidad y coordinación.

El software de simulación ETAP (Electrical Transient Analyzer Program) es una de estas herramientas avanzadas de simulación y análisis de sistemas eléctricos de potencia. ETAP es destacado en la industria porque se utiliza para realizar estudios complejos que incluyen flujos de potencia, análisis de cortocircuitos, estabilidad dinámica y confiabilidad del sistema, ofreciendo precisión y capacidad de modelar detalladamente sistemas eléctricos extensos y complejos. (Gómez-Ramírez et al., 2023b).

Las ventajas de ETAP según (Gómez-Ramírez et al., 2023b) son varias; a continuación, se presenta una lista de estas ventajas:

- Su interfaz gráfica facilita la creación y modificación de modelos de sistemas eléctricos.
- ETAP incluye una amplia biblioteca de componentes y datos reales que mejoran la precisión de las simulaciones.
- Tiene capacidad para manejar grandes volúmenes de datos y realizar simulaciones en tiempo real, lo que es crucial para el análisis y la planificación de sistemas eléctricos complejos.
- ETAP también permite realizar análisis detallados de estabilidad y confiabilidad, lo que ayuda a identificar y mitigar posibles fallos en el sistema.

2.6.1. Estudio de Flujo de Potencia

El flujo de potencia es una herramienta fundamental y ampliamente utilizada en la planificación y operación de sistemas de energía eléctrica, abarcando tanto la transmisión como la distribución de electricidad. Este método implica el cálculo y análisis de la distribución de potencia en una red eléctrica con el objetivo principal de determinar las condiciones de operación en régimen permanente. A través del flujo de potencia, es posible evaluar y verificar diversos aspectos críticos del sistema, como la estabilidad de

tensión, la presencia de sobrecargas y las pérdidas de energía, entre otros. Estos análisis son esenciales tanto para sistemas eléctricos existentes, permitiendo abordar problemas de operación económica y reducción de pérdidas, como para la planificación de nuevos sistemas, donde se busca comprender el comportamiento de los componentes del sistema en diferentes escenarios de diseño y planificación. Por lo tanto, el flujo de potencia se erige como una herramienta clave en la gestión y desarrollo eficiente de infraestructuras eléctricas. (Gallego et al., 2009).

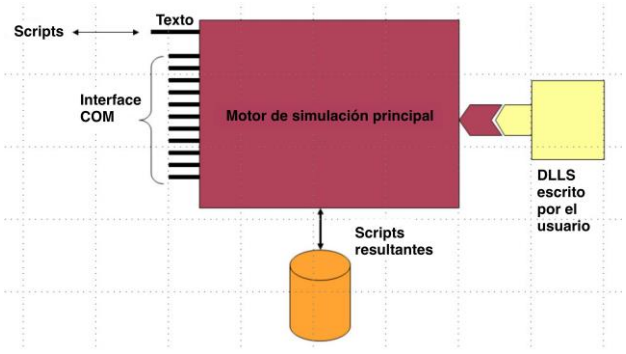
2.6.2. OpenDss

El Open Distribution System Simulator (OpenDSS) es una herramienta de simulación para sistemas de distribución eléctrica, diseñada para el análisis y planificación de redes eléctricas. OpenDSS proporciona una interfaz de usuario basada en texto para desarrollar scripts y visualizar soluciones, además de un servidor COM DLL para integrarse con otras plataformas de software. Este programa soporta una amplia gama de análisis en estado estacionario (en el dominio de la frecuencia) y está diseñado para adaptarse a las necesidades emergentes del sector eléctrico, como la integración de generación distribuida y la automatización de redes eléctricas inteligentes. (Dugan, 2013).

La estructura del OpenDSS se compone de varios modos de solución incorporados, como Snapshot Power Flow, Daily Power Flow, Yearly Power Flow, Harmonics, Dynamics, Faultstudy y Monte Carlo Faultstudy, entre otros. Estos modos fueron desarrollados para abordar necesidades específicas de análisis en proyectos particulares y están diseñados para ser fácilmente personalizables mediante la interfaz COM DLL, permitiendo a los usuarios diseñar y ejecutar modos de solución personalizados desde programas externos; en la figura 2.15 se muestra un diagrama de la estructura de OpenDSS. (Dugan, 2013).

Figura 2.15

Estructura de OpenDSS



Fuente: Dugan (2013).

Una característica destacada del OpenDSS es su capacidad de interoperabilidad con múltiples lenguajes de programación como MATLAB y Python, lo que permite a los usuarios aprovechar capacidades analíticas externas y visualizar resultados de manera efectiva. Además, el programa puede ser personalizado mediante la descarga del código fuente para adaptarse a necesidades específicas o desarrollando DLLs que se integren con sus capacidades de modelado de dispositivos. (Dugan, 2013).

OpenDss-G

El Simulador Gráfico de Sistemas de Distribución (OpenDSS-G) es una interfaz desarrollada sobre la base de OpenDSS, es un software de procesamiento paralelo a OpenDSS que surgió de la necesidad de contar con una interfaz más sofisticada para satisfacer las demandas futuras de estudios de planificación y operación de redes inteligentes. OpenDSS-G ofrece un entorno integral diseñado para aprovechar las capacidades avanzadas de OpenDSS a través de una interfaz gráfica amigable, similar a una sala de control equipada con herramientas intuitivas. Su diseño flexible y dinámico proporciona una amplia gama de herramientas gráficas que simplifican los entornos de simulación mientras abordan los requisitos de estudios completos de sistemas eléctricos. (Montenegro and Dugan, 2020).

2.6.3. OpenDSS-Viewer

OpenDSS-Viewer es un complemento de OpenDSS-G que está diseñado para mejorar la visualización de los resultados obtenidos a través del software OpenDSS-G. Su plataforma de visualización ofrece una amplia gama de funcionalidades, permitiendo la generación de gráficos para numerosas variables eléctricas de manera efectiva. Con campos personalizables y modos de interacción avanzados, permite analizar ampliamente los resultados de las simulaciones. (Hernández, 2018).

2.6.4. Sistema de Información Geográfica

Se refiere a un conjunto de software de computadora, hardware y periféricos diseñados para transformar datos referenciados geográficamente en información acerca de localizaciones, interacciones espaciales y relaciones geográficas de las entidades, tanto estáticas como dinámicas, que ocupan un espacio en entornos naturales o construidos. (Santos Preciado, 2020).

2.7. Python

Es un lenguaje de programación de alto nivel, interpretado, interactivo y orientado a objetos. Surgió en la década de 1980 bajo la dirección de Guido van Rossum. Ha ganado considerables niveles de popularidad

y adopción en la comunidad de desarrollo de software debido a su sintaxis clara y legible, lo que lo convierte en una elección frecuente para programadores con diferentes niveles de experiencia. (Moreno Muñoz and Cordobes Córcoles, 2019).

2.7.1. Pandas

Es una biblioteca de Python que facilita la manipulación de estructuras de datos y herramientas de análisis de datos. Como se explica en Ortega Candel (2022), Pandas permite manejar diversas fuentes de datos, como datos tabulados, series temporales, matrices de datos y datos estadísticos. Entre las estructuras de datos que ofrece se destacan:

- Series: Son arreglos unidimensionales con indexación, similares a los diccionarios y se pueden generar a partir de diccionarios o listas.
- DataFrame: Estructuras de datos similares a las tablas en bases de datos relacionales, como SQL.
- Panel, Panel4D y PanelND: Estructuras de datos que permiten trabajar con más de dos dimensiones.

2.7.2. GeoPandas

GeoPandas es un proyecto de código abierto que simplifica la manipulación de datos geoespaciales en Python al extender los tipos de datos de Pandas para incluir operaciones espaciales en geometrías. Haciendo uso de Shapely para operaciones geométricas y dependiendo de fiona para el acceso a archivos y matplotlib para visualizaciones, GeoPandas amalgama las capacidades de Pandas y Shapely, ofreciendo así una interfaz de alto nivel que posibilita realizar operaciones geoespaciales en múltiples geometrías de Shapely en Python, eliminando la necesidad de recurrir a una base de datos espaciales como PostGIS para realizar estas operaciones. (GeoPandas.org, 2013).

2.7.3. Shapely

Shapely es una biblioteca de Python con licencia BSD que se utiliza para la manipulación y el análisis de objetos geométricos planos. Su objetivo principal es proporcionar una interfaz para trabajar con geometrías simples (como puntos, líneas y polígonos) de manera eficiente y fácil de usar. Shapely envuelve las geometrías y operaciones de la biblioteca GEOS, que es un motor de geometría de código abierto ampliamente desplegado y utilizado en sistemas de información geográfica (GIS) como PostGIS, y es un puerto de JTS (Java Topology Suite). (PyPI, 2024).

Capítulo 3

Modelado del Circuito

Para llevar a cabo este proyecto, el ICE asignó un circuito eléctrico, cuya ubicación es confidencial, donde se inició el proceso de modelado utilizando OpenDSS y OpenDSS-G. En este capítulo se proporcionará una descripción detallada del proceso de modelado de dicho circuito. Se iniciará describiendo el proceso de definición de los buses y sus respectivas coordenadas. Posteriormente, se explicará el proceso de definición de las líneas de media tensión o primarias, además del procedimiento de minería de datos para limpiar la información del circuito, corregir errores en los buses y conectar las líneas de manera adecuada.

Continuando con los demás elementos del circuito, se demostrará cómo se realizó la incorporación de los transformadores de distribución, siendo estos los encargados de transformar la tensión para hacer el paso a las líneas secundarias o de baja tensión. Seguidamente, se procederá a mostrar la configuración de las líneas de baja tensión, las cuales son responsables de distribuir la energía hacia los consumidores finales. Una vez establecidos los componentes principales del circuito, se abordará la integración de las cargas al circuito de distribución y, por último, se agregarán dos reguladores de tensión para estabilizar la tensión del circuito.

Debido a la cantidad de elementos del circuito, se empleó Python como herramienta principal para generar los archivos que definen los elementos, agilizando el procesamiento de los datos. Se realizó un programa llamado “GeneradorArchivoDSS.py”, que genera los archivos de extensión dss para todos los elementos del circuito exceptuando los reguladores de tensión. Además, se desarrollaron programas específicos para corregir los errores en los datos dados.

3.1. Obtención de Datos

Primeramente, el ICE proporcionó un archivo de Excel llamado “Circuito_5” que incluye varias hojas. La primera, llamada ‘Busbar”, contiene información sobre 23879 barras. La segunda, llamada ‘LineAsym’, detalla datos sobre 23389 líneas, tanto primarias como secundarias. La tercera, ‘Trafo2WindingAsym’, presenta información sobre 808 transformadores monofásicos y 12 transformadores bifásicos. La cuarta, ‘Trafo2Winding’, proporciona detalles sobre 13 transformadores trifásicos. La quinta, ‘Load’, describe 7712 cargas. Finalmente, la sexta, ‘TrafoRegulator’, incluye información sobre 2 reguladores de tensión.

Tabla 3.1

Estructura de la Hoja Busbar del Archivo Circuito_5

	Name	AliasName	Un	Uset	Umax	Umin	Ir	Ipmax	Zone	Area	CoordX1	CoordY1	CoordX2	CoordY2
0	MT_AREA_434111_810263	MT_AREA_434111_810263	34,5								434111,8103	1101203,528	434111,8103	1101203,528
1	MT_AREA_434122_651601	MT_AREA_434122_651601	34,5								434122,6516	1101218,809	434122,6516	1101218,809
2	MT_AREA_434100_781346	MT_AREA_434100_781346	34,5								434100,7813	1101152,47	434100,7813	1101152,47
...	...	...	...	...							...			
23878	143164_T	143164_T	0,24								427504,9688	1070703,629	427504,9688	1070703,629

Fuente: ICE (2024).

Tabla 3.2

Estructura de la Hoja LineAsym del Archivo Circuito_5

Unnamed: 0	Switch1	Switch2	Node1	Node2	Name	AliasName1	LibraryType	LineType	Phase	Length	Un
0	VERDADERO	VERDADERO	BT_SUB_430175_221521	BT_SUB_430174_651095	15206__0	15206__0	BT_SUB_A: _250 COBRE RHH_250		4	0,001119997	0,24
1	VERDADERO	VERDADERO	BT_SUB_430174_651095	BT_SUB_430176_281846	15206__1	15206__1	BT_SUB_A: _250 COBRE RHH_250		4	0,003008579	0,24
2	VERDADERO	VERDADERO	BT_SUB_430176_281846	BT_SUB_430177_310055	15206__2	15206__2	BT_SUB_A: _250 COBRE RHH_250		4	0,001034381	0,24
...	...	...	...	...	...	...	...		...	...	...
23388	VERDADERO	VERDADERO	MT_AREA_429763_007282	MT_AREA_429763_688325	9030622__0	9030622__0	MT_B: 6 CO_SO_NT_NT_NT_NT_NE_N		2	0,001020001	34,5

Fuente: ICE (2024).

Tabla 3.3

Estructura de la hoja Trafo2WindingAsym del archivo Circuito_5

Unnamed: 0	Name	Node1	Node2	Switch1	Switch2	IsRegulated	Un1	Un2	Sr	LibraryType	CoordX1	CoordY1	SymbolSize	SymbolAngle
0	291178_T	MT_AREA_432812_157258	BT_AREA_432812_157258	VERDADERO	VERDADERO	FALSO	34,5	0,24	15	A_15.0_34.5_kV_.240_kV_1	432812,1573	1099781,302	1	0
1	291179_T	MT_AREA_434118_110706	BT_AREA_434118_110706	VERDADERO	VERDADERO	FALSO	34,5	0,24	50	C_50.0_34.5_kV_.240_kV_1	434118,1107	1099066,517	1	0
2	291180_T	MT_AREA_434685_31123	BT_AREA_434685_31123	VERDADERO	VERDADERO	FALSO	34,5	0,24	50	C_50.0_34.5_kV_.240_kV_1	434685,3112	1099210,755	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
819	2993267_T	MT_AREA_438869_405053	2993267_T	VERDADERO	VERDADERO	FALSO	34,5	0,24	25	B_25.0_34.5_kV_.240_kV_1	438869,4051	1096869,944	1	0

Fuente: ICE (2024).

Tabla 3.4

Estructura de la Hoja Trafo2Winding del Archivo Circuito_5

Unnamed: 0	Name	Node1	Node2	Switch1	Switch2	IsRegulated	Un1	Un2	Sr	LibraryType	CoordX1	CoordY1	SymbolSize	SymbolAngle
0	547035_T	MT_SUB_427710_448811	BT_SUB_427710_448811	VERDADERO	VERDADERO	FALSO	34,5	0,24	300	ABC_300.0_34.5_kV_.240_kV_2_Estrella_Fase_Partida	427710,4488	1072828,308	1	0
1	140156_T	MT_AREA_430225_648804	BT_AREA_430225_648804	VERDADERO	VERDADERO	FALSO	34,5	0,208	225	ABC_225.0_34.5_kV_.208_kV_1_Estrella_Estrella	430225,6488	1094622,586	1	0
2	140199_T	MT_AREA_431868_677033	BT_AREA_431868_677033	VERDADERO	VERDADERO	FALSO	34,5	0,24	150	ABC_150.0_34.5_kV_.240_kV_1_Estrella_Fase_Partida	431868,677	1095348,344	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12	306006_T	MT_AREA_428526_779389	BT_AREA_428526_779389	VERDADERO	VERDADERO	FALSO	34,5	0,48	225	ABC_225.0_34.5_kV_.480_kV_1_Estrella_Estrella	428526,7794	1094602,005	1	0

Fuente: ICE (2024).

Tabla 3.5

Estructura de la Hoja Load del Archivo Circuito_5

Unnamed: 0	Node1	Name	Phase	Switch1	Un	E	VelanderK1	LfType	Unit	CosPhi	CoordX1	CoordY1	Tipo
0	BT_AREA_431310_873973	87140__0	4	VERDADERO	0,24	948	0,001388889	EC	LV	0,95	431310,874	1074819,457	2
1	BT_AREA_427720_355503	87165__1	4	VERDADERO	0,24	3572	0,001388889	EC	LV	0,95	427720,3555	1072837,057	2
2	BT_SUB_427709_420947	91777__2	0	VERDADERO	0,24	15667	0,001388889	EC	LV	0,95	427709,4209	1072823,382	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7712	BT_AREA_430846_964866	BT_AREA_430846_964866	4	VERDADERO	0,24	621	0,001388889	EC	LV	0,95	430846,9649	1094909,424	1

Fuente: ICE (2024).

Tabla 3.6

Estructura de la Hoja TrafoRegulator del Archivo Circuito_5

	Name	Node1	Node2	Switch1	Switch2	Un1	Un2	Phase	LibraryType	X	Y
0	8011_R	MT_AREA_435110_859936	MT_AREA_435112_952647	True	True	34,5	34,5	0	ABC_667.0_3	435110,9	1087069
1	8012_R	MT_AREA_431474_646134	MT_AREA_431407_6761	True	True	34,5	34,5	0	ABC_667.0_3	431474,6	1078620

Fuente: ICE (2024).

Además de los datos del circuito, el ICE proporcionó otro archivo de Excel, llamado “Libreria_Lineas.xlsx”, que contiene las matrices de impedancia de algunas líneas de distribución primarias. Sin embargo, es importante destacar que dicho archivo no contiene la información de todas las líneas de distribución primarias del circuito, por lo que luego se tuvo que calcular las impedancias para las líneas faltantes.

Para el cálculo de las matrices de impedancia faltantes el ICE otorgó un archivo de Excel llamado “Copia de Impedancias de línea DDC” y otro archivo de PowerPoint nombrado como “Configuraciones-Conductores_V14” con las configuraciones usadas en las líneas (observar anexo A figuraA.1).

También el ICE proporcionó cuatro archivos DSS con las características de diferentes conductores. Estos archivos se denominan: “WireDataAAAC.ds”, “WireDataAAC.dss”, “WireDataACSR.dss” y “WireDataCU.dss”, y contienen información sobre los conductores AAAC, AAC, ACSR y Cobre, respectivamente.

Adicionalmente, el ICE proporcionó dos archivos DSS que detallan las características de las líneas secundarias. Uno de estos archivos nombrado como “LineCodeAcometidas”, contiene las matrices de impedancia para las líneas tríplex y cuádruplex, mientras que el otro llamado “LineCodeSubterraneas” se enfoca en las líneas de distribución secundarias subterráneas. Más adelante con OpenDss se hizo un archivo con las características de línea de las líneas de distribución secundarias faltantes.

Finalmente, se proporcionó un archivo de Python denominado ’trafoOperations.py’, el cual no es funcional; es decir, no ejecuta ningún programa. Este archivo contiene los parámetros comunes de los transformadores y fue creado por el Electrical Power and Energy Research Laboratory (EPERLab, 2017) de la Universidad de Costa Rica.

3.2. Definición de los Buses

Para la definición de los buses se añadió una sección llamada “Creación del archivo de buses” al código de Python nombrado como “GeneradorArchivoDSS.py” que cumple con un doble propósito. En primer lugar, este programa realiza una conversión de las coordenadas asignadas a los buses, del sistema EPSG:5367 al sistema EPSG:4326. Debido a que el ICE proporcionó las coordenadas de los buses en el sistema de coordenadas EPSG:5367. Este sistema expresa las coordenadas en metros y se dividen en coordenadas este (X) y norte (Y). Sin embargo, OpenDSS procesa las coordenadas en un formato de longitud y latitud,

donde la longitud varía entre -180° y 180° y la latitud entre -90° y 90°, conforme con el sistema EPSG:4326, haciendo esencial la transformación para permitir una adecuada integración de los datos geoespaciales en el modelado del circuito eléctrico. Una vez completada la conversión, el programa genera un archivo llamado BusCoords en formato DSS que contiene los datos de cada bus. El archivo resultado generado por este programa de Python sigue una estructura simple, cada línea del archivorepresenta un bus del circuito eléctrico y contiene tres elementos separados por una coma y un espacio: el nombre del bus, su coordenada de longitud y su coordenada de latitud, en la figura 3.1 se muestra parte del archivo generado.

Figura 3.1

Ejemplo Archivo de Salida de Coordenadas de los Buses BusCoords.dss

```
MT_AREA_434111_810263, -84.60092686924038, 9.95840316688248
MT_AREA_434122_651601, -84.60082824842144, 9.958541506570082
MT_AREA_434100_781346, -84.6010266085358, 9.957941351610943
MT_AREA_434076_319879, -84.60124873234811, 9.957413471166005
MT_AREA_434047_010128, -84.6015148196254, 9.95674778441674
MT_AREA_434014_622266, -84.60180895533117, 9.95606954007476
MT_AREA_433005_4634, -84.61100053430907, 9.949635094562774
MT_AREA_433006_5023, -84.61099100953636, 9.949608108015429
MT_AREA_433006_502296, -84.61099100957267, 9.949608107924949
MT_AREA_433000_759344, -84.6110432823619, 9.949553273935686
MT_AREA_432973_061798, -84.61129538730349, 9.949288815903353
MT_AREA_432963_065377, -84.61138664839739, 9.94934015065852
MT_AREA_434009_530893, -84.60185482410465, 9.955761441562075
MT_AREA_433982_150765, -84.60210352811897, 9.955214272094372
MT_AREA_433955_119809, -84.60234905281202, 9.954670323418991
MT_AREA_433914_370927, -84.60271987632952, 9.954231281897997
```

Fuente: Elaboración propia.

En el apéndice A en la figura A.1 se puede observar la sección del archivo de Python llamado “GeneradorArchivoDSS.py” que realiza la definición de los buses. Este comienza importando las bibliotecas necesarias, incluyendo pandas para el manejo de datos, geopandas para trabajar con datos geoespaciales, shapely para operaciones geométricas, subprocess para ejecutar procesos en el sistema operativo y csv para operaciones con archivos CSV (esta biblioteca no se utiliza en esta sección del código, pero se utiliza más adelante para definir otros elementos del circuito).

El siguiente paso consiste en especificar la ruta del archivo Excel que contiene la información de los buses del circuito. Este archivo se localiza en la ruta especificada por la variable ‘file_path’. Una vez definida la ruta del archivo, se procede a leer los datos del archivo Excel en un DataFrame de pandas. Específicamente, se lee la pestaña ‘Busbar’ que contiene la información de los buses. Este DataFrame se denomina ‘Busbar_df’.

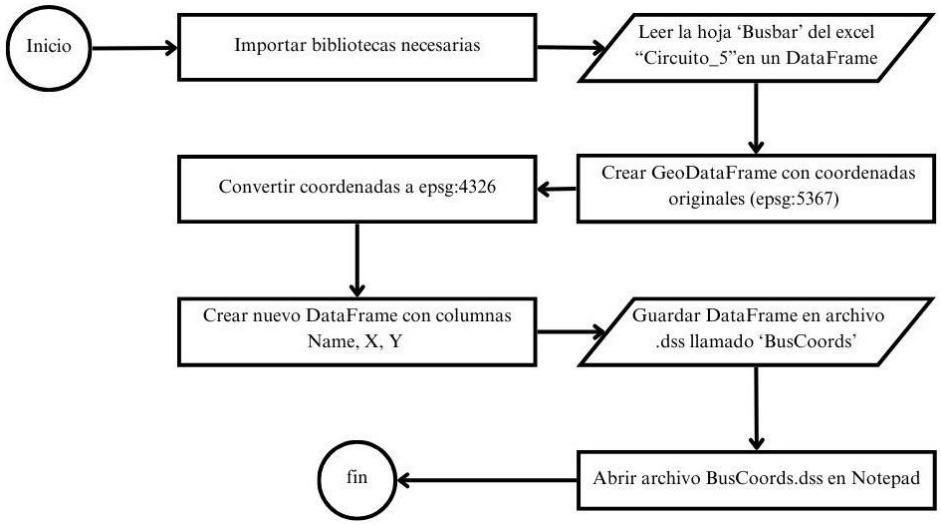
Con los datos cargados en el DataFrame, se procede a crear un GeoDataFrame utilizando las coordenadas originales de los buses, las cuales están representadas en el sistema de coordenadas EPSG:5367. Se utiliza

la biblioteca shapely para crear objetos de tipo Point que representan las coordenadas 'X' e 'Y' de cada bus. Estos objetos se agregan al GeoDataFrame 'gdf_orig'. Posteriormente, se convierten las coordenadas de EPSG:5367 a EPSG:4326 para adecuarlas al formato utilizado por OpenDSS. Esta conversiónse realiza mediante el método 'to_crs' de geopandas, obteniendo un nuevo GeoDataFrame denominado 'gdf_converted'.

Una vez convertidas las coordenadas, se crea un nuevo DataFrame llamado 'output_df' que contiene las columnas relevantes para la definición de los buses. Estas columnas incluyen el nombre del bus ('Name') y las coordenadas convertidas ('X' y 'Y'). El siguiente paso consiste en guardar este DataFrame en un archivo de texto con extensión '.dss' el cual contendrá los nombres y coordenadas de los buses. El nombre del archivo de salida se especifica como 'BusCoords.dss'. Finalmente, se abre automáticamente este archivo en el bloc de notas para revisarlo utilizando la función 'subprocess.run (['notepad.exe', BusCoords])'. En la figura 3.2 se muestra un diagrama de flujo que resume el funcionamiento de esta sección del código.

Figura 3.2

Diagrama de Flujo del Proceso de Generación del Archivo BusCoords.dss



Fuente: elaboración propia

3.3. Modelado de las Líneas de Distribución Primaria

3.3.1. Cálculo de las Matrices de Impedancia.

Para comenzar el modelado de las líneas de distribución primaria en OpenDSS, primero se llevaron a cabo los cálculos de las matrices de impedancia faltantes. Este proceso se realizó utilizando OpenDSS. Para calcular las matrices de impedancia en OpenDSS, se necesitan definir tres objetos clave: WireData

(características de los conductores), spacing (espaciado entre conductores) y Geometry (geometría de los conductores).

Para calcular las matrices de impedancia se hizo un código en OpenDSS, para cada una de las líneas sin matriz de impedancia, siguiendo la lógica que se muestra en el diagrama de flujo de la figura 3.3. Este código en primer lugar establece la configuración del circuito eléctrico, incluyendo la base de tensión, ángulo, frecuencia y otras características relevantes.

Luego, se detallan las propiedades de los conductores mediante la referencia a archivos que contienen información específica, como la resistencia en corriente alterna (Rac), el radio geométrico medio (GMR), el diámetro externo del conductor (Diam) y la corriente nominal del conductor (Normamps). Estos parámetros son esenciales para modelar adecuadamente los conductores en el sistema.

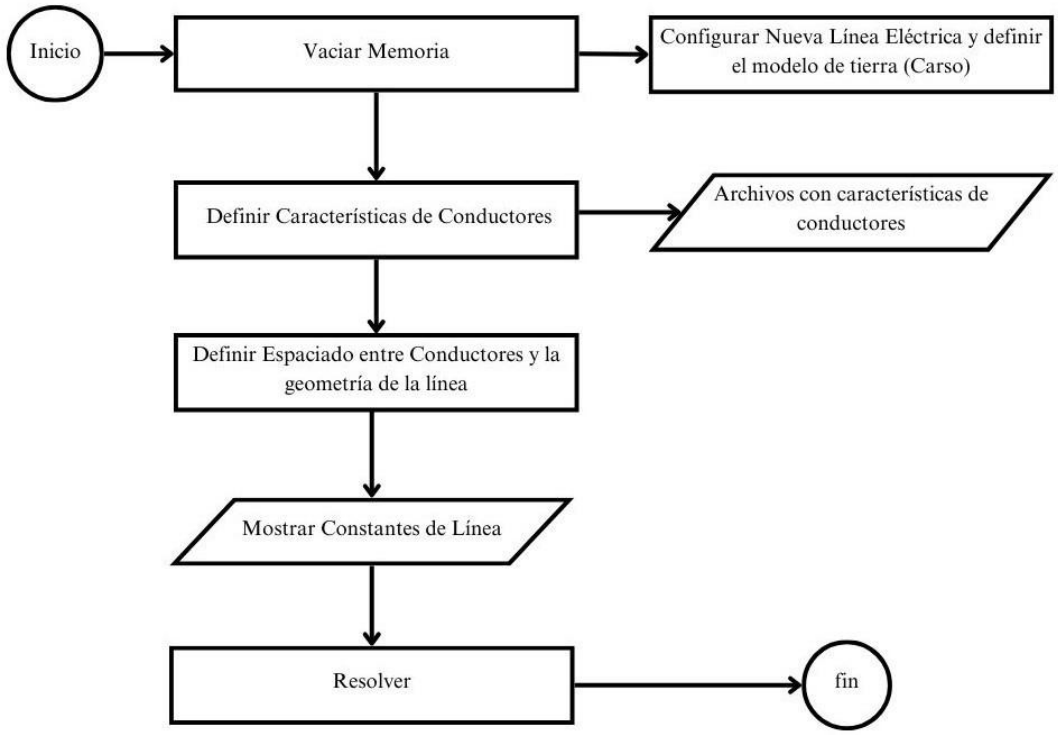
Posteriormente, se establece el espaciado entre conductores y se define la geometría de la línea.

Finalmente, se calculan las impedancias y capacitancias de la línea y se resuelve el sistema para obtener las matrices de impedancia en un archivo denominado LineCode. Este proceso se repitió para todas las líneas faltantes de matriz de impedancias. En el apéndice B en la figura B.1 se muestra un ejemplo de un código empleado para calcular la matriz de impedancia de una de las líneas faltantes.

Figura 3.3

Diagrama de Flujo del Código para Cálculo de Impedancias en OpenDSS.

Fuente: Elaboración propia.



Al correr el código en OpenDss se obtiene la matriz de impedancia que se muestra en la figura 3.4 y los resultados se transcribieron en el archivo de Excel denominado como librería_Lineas.xlsx.

Figura 3.4

Ejemplo Resultado de la Matriz Impedancia de una Línea Monofásica

```
!--- OpenDSS Linecodes file generated from Show LINECONSTANTS command
!--- Frequency = 60 Hz, Earth resistivity = 100 ohm-m
!--- Earth Model = Carson

New Linecode.1fmv1/0aaac_aaac_1/0_nt_nt_7_n nphases=1  Units=km
~ Rmatrix=[0.747134  ]
~ Xmatrix=[0.797625  ]
~ Cmatrix=[6.88231  ]
```

Fuente: Elaboración propia.

En la hoja “LineAsym” del archivo del circuito 5, hay una columna etiquetada como “LibraryType”. Esta columna contiene una designación que está relacionada con el archivo de “Libreria_Lineas.xlsx”, donde se pueden encontrar las matrices de impedancia correspondientes. Por otra parte, esta designación proporciona los datos necesarios para el cálculo de las matrices de impedancia para las líneas que no están en “Libreria_Lineas.xlsx”, por ejemplo.

“MT_A: 1/0 AAAC_AAAC_1/0_NT_NT_7_N” de la figura B.1 se desglosa de la siguiente manera:

- ‘MT’: indica que la línea es de media tensión.
- ‘_A’: indica que la línea está conectada a la fase A y que es monofásica.
- ‘1/0 AAAC’: significa que los conductores de la fase son del tipo 1/0 AAAC.
- ‘_AAAC_1/0’: indica que el neutro es del tipo 1/0 AAAC.
- ‘_7’: indica que la configuración de la línea es la número 7 (ver Anexos).

En la tabla 3.7 se mostrarán los significados del inicio de las designaciones en LibraryType:

Tabla 3.7

Nomenclatura de LibraryType

Designación	Número de Fases	Fase conectada	Tensión	Representación de nodo OpenDss
MT_A	1	Fase A	Media tensión	.1
MT_B	1	Fase B	Media tensión	.2
MT_C	1	Fase C	Media tensión	.3
MT_AB	2	Fase AB	Media tensión	.1.2
MT_BC	2	Fase BC	Media tensión	.2.3
MT_AC	2	Fase AC	Media tensión	.1.3
MT_ABC	3	Fase ABC	Media tensión	.1.2.3
BT_A	1	Fase A	Baja tensión	.1
BT_B	1	Fase B	Baja tensión	.2
BT_C	1	Fase C	Baja tensión	.3
BT_AB	2	Fase AB	Baja tensión	.1.2
BT_BC	2	Fase BC	Baja tensión	.2.3
BT_AC	2	Fase AC	Baja tensión	.1.3
BT_ABC	3	Fase ABC	Baja tensión	.1.2.3
MT_SUB_A	1	Fase A	Media tensión Subterránea	.1
MT_SUB_B	1	Fase B	Media tensión Subterránea	.2
MT_SUB_C	1	Fase C	Media tensión Subterránea	.3
MT_SUB_AB	2	Fase AB	Media tensión Subterránea	.1.2
MT_SUB_BC	2	Fase BC	Media tensión Subterránea	.2.3
MT_SUB_AC	2	Fase AC	Media tensión Subterránea	.1.3
MT_SUB_ABC	3	Fase ABC	Media tensión Subterránea	.1.2.3
BT_SUB_A	1	Fase A	Baja tensión Subterránea	.1
BT_SUB_B	1	Fase B	Baja tensión Subterránea	.2
BT_SUB_C	1	Fase C	Baja tensión Subterránea	.3
BT_SUB_AB	2	Fase AB	Baja tensión Subterránea	.1.2
BT_SUB_BC	2	Fase BC	Baja tensión Subterránea	.2.3
BT_SUB_AC	2	Fase AC	Baja tensión Subterránea	.1.3
BT_SUB_ABC	3	Fase ABC	Baja tensión Subterránea	.1.2.3

Fuente: Elaboración propia.

3.3.2. Programa en Python para la Definición de las Líneas Primarias

En la sección del código GeneradorArchivoDSS.py, encargada de crear el archivo para las líneas de media tensión, se genera como producto final un archivo que incluye los elementos necesarios para definir cada línea en OpenDSS. Una parte de este archivo se muestra en la figura 3.5. La definición de una líneaen OpenDSS

se realiza mediante el objeto ***'new line.'*** . Inmediatamente después del objeto, se deben incluir el nombre asignado en la hoja de Excel 'LineAsym' para las líneas, la cantidad de fases por línea (Phase), el nombre del bus de inicio con sus respectivos nodos (bus1), el nombre del bus de fin con sus respectivos nodos (bus2),

la longitud de la línea (Length), las unidades (Units) y la matriz de impedancia (definida previamente).

Figura 3.5

Ejemplo del Archivo dss para la Definición de las Líneas Primarias

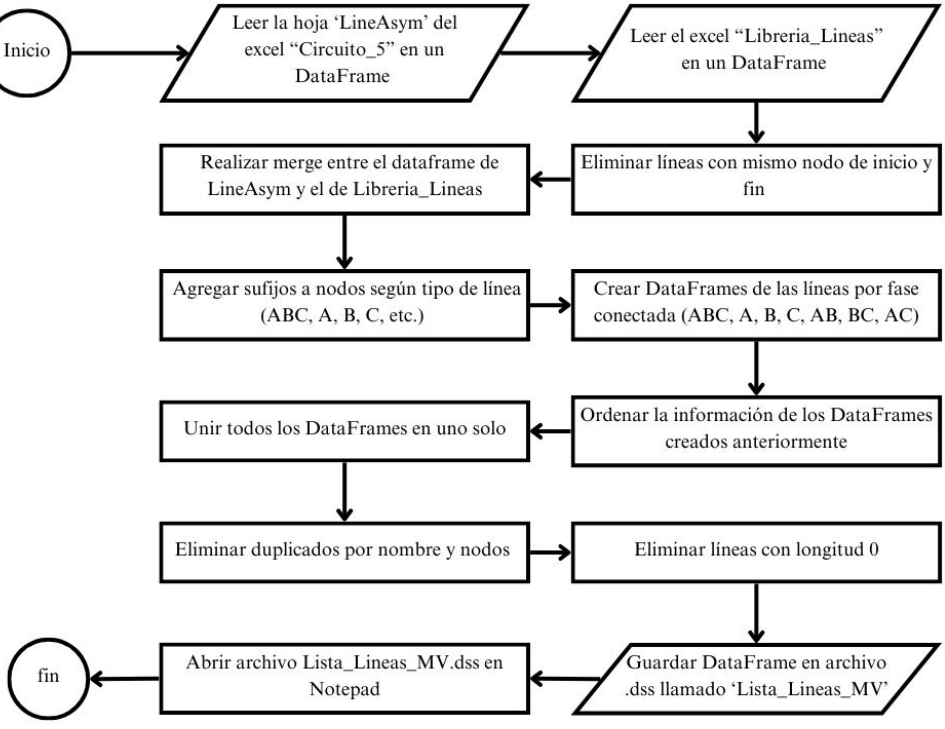
```
new line.23623_0 phases=3 bus1=MT_AREA_430189_9977.1.2.3 bus2=MT_SUB_430187_1536.1.2.3
length=0.00299999584331625 units=km rmatrix= [0.206462 0.0592176 0.0592176 0.206462  0.0592176
0.206462] xmatrix= [0.900531 0.703891  0.10025 0.900531  0.10025  0.900531] xmatrix= [14.9962 -8.87549
-5.132440000000000e-05 12.9962   -5.132440000000000e-05  9.74324]
new line.23623_1 phases=3 bus1=MT_SUB_430187_1536.1.2.3 bus2=MT_SUB_430189_11013.1.2.3
length=0.0772378532586265 units=km rmatrix= [0.206462 0.0592176 0.0592176 0.206462] xmatrix=
[0.900531 0.703891  0.10025 0.900531  0.10025  0.900531] xmatrix= [14.9962 -8.87549
-5.132440000000000e-05 12.9962   -5.132440000000000e-05  9.74324]
```

Fuente: Elaboración propia.

En la figura 3.6 se muestra un diagrama de flujo que representa la lógica utilizada en esta sección del código. Este diagrama facilita la comprensión del funcionamiento del código. Sin embargo, a continuación, se explicará detalladamente el código para brindar una comprensión completa.

Figura 3.6

Diagrama de Flujo del Código para la Definición de las Líneas Primarias



Fuente: Elaboración propia

El código comienza creando un DataFrame que contiene los datos de la hoja “LineAsym” del archivo Excel “Circuito_5” y otro DataFrame con los datos del archivo Excel “Libreria_Lineas”. Además, se eliminaron todas las líneas cuyos nodos de inicio y final son iguales, considerándolas como errores del GIS. Posteriormente, utilizando la función merge, se generó un nuevo DataFrame que fusiona los datos de “LineAsym” con las matrices de impedancias contenidas en la “Libreria_Lineas”.

Luego, se procedió a agregar una sección que modificara los nombres de los buses de las líneas, agregando sufijos que representan los nodos. Estos sufijos son utilizados en OpenDSS para especificar la fase a la cual está conectada la línea. En el caso de una línea trifásica, se conecta a los nodos (.1.2.3), donde (.1) representa la fase A, (.2) la fase B y (.3) la fase C. El nodo cero (.0) está aterrizado por definición. Además, el nodo (.4) se puede utilizar para declarar un neutro no aterrizado.

Es importante destacar que el número de nodos que puede tener un bus en OpenDSS es ilimitado, en la Tabla 3.7 se muestran los sufijos utilizados para las líneas dependiendo del LibraryType asignado y después se dividieron las líneas en distintos DataFrames dependiendo de la fase en la que estuvieran conectadas. Todo lo descrito anteriormente se puede observar en el apéndice A, figura A.2.

Posteriormente, se procedió a establecer la estructura para el documento donde se definirán las líneas de media tensión, añadiendo los elementos mencionados anteriormente, para que OpenDSS pueda definir una línea de manera adecuada. Esta estructura se realizó de manera personalizada para cada DataFrame, considerando la fase a la que está conectada la línea. Esto se debe a que el archivo “Libreria_Lineas” contiene diversas columnas que albergan datos de resistencia, inductancia y capacitancia por fase, para la estructura de las matrices de impedancia, teniendo las siguientes columnas (esta sección del código se puede observar en el apéndice A, figura A.3):

- Primero se tienen seis columnas para la resistencia, identificadas con la letra R, seguida de los subíndices RR, RS, RT, SS, ST y TT. Donde R representa la fase A, S la fase B y T la fase C.
- Seguidamente están otras seis columnas para la inductancia, designadas con la letra X seguida de los subíndices RR, RS, RT, SS, ST y TT. De igual manera, R denota la fase A, S la fase B y T la fase C.
- Por último, se encuentran otras seis columnas para la capacitancia, identificadas con la letra C y los subíndices RR, RS, RT, SS, ST y TT. Igualmente, R representa la fase A, S la fase B y T la fase C.

Para finalizar, los datos se organizaron en el orden adecuado y se fusionaron todos los DataFrames en uno solo. Luego, se guardó la información en un archivo DSS denominado “Lista_Lineas_MV.dss”, el cual se abre automáticamente. En el apéndice A, figura A.4 se muestra el final de esta sección del código.

3.3.3. Generación del Archivo Máster de OpenDSS y la Importación a OpenDSS-G

En la figura C.1 del apéndice C se puede observar el código de OpenDSS para la simulación del circuito, este archivo dss se nombró "Master.dss" y comienza limpiando cualquier configuración previa mediante el comando 'clear', asegurando así un entorno limpio para la ejecución del script. Luego, se crea un nuevo circuito llamado "circuito5" utilizando el comando 'new circuit.' y se establece la frecuencia base predeterminada en 60 Hz mediante 'Set DefaultBaseFrequency=60'. Posteriormente, se redirige la salida del script a un archivo llamado "V_source.dss" con el comando 'Redirect'; este archivo contiene la siguiente información:

Edit "Vsource.source" bus1=MT_AREA_427371_830453 basekv=34.5 pu=1.0 angle=0 frequency=60 phases=3 Mvasc3=3000

donde:

- "Vsource.source": indica que se está modificando una fuente de tensión llamada "source" en el circuito.
- bus1=MT_AREA_427371_830453: especifica el bus al que está conectada la fuente de tensión (en este caso, la fuente está conectada al bus denominado "MT_AREA_427371_830453").
- basekv=34.5: establece el nivel de tensión base de la fuente de tensión en kilovoltios (aquí se indica que la tensión base es de 34.5 kV).
- pu=1.0: define el valor de la tensión de la fuente de tensión en unidades porcentuales (pu) (un valor de 1.0 indica que la tensión es igual a su valor base).
- angle=0: especifica el ángulo de fase de la fuente de tensión en grados (en este caso, el ángulo de fase es cero grados).
- frequency=60: establece la frecuencia de la fuente de tensión en hertzios (aquí se indica que la frecuencia es de 60 Hz).
- phases=3: indica el número de fases de la fuente de tensión (en este caso, la fuente tiene tres fases).
- Mvasc3=3000: define la capacidad aparente de la fuente de tensión en megavoltiamperios (MVA) para cada fase (aquí se indica que la capacidad aparente es de 3000 MVA para cada fase).

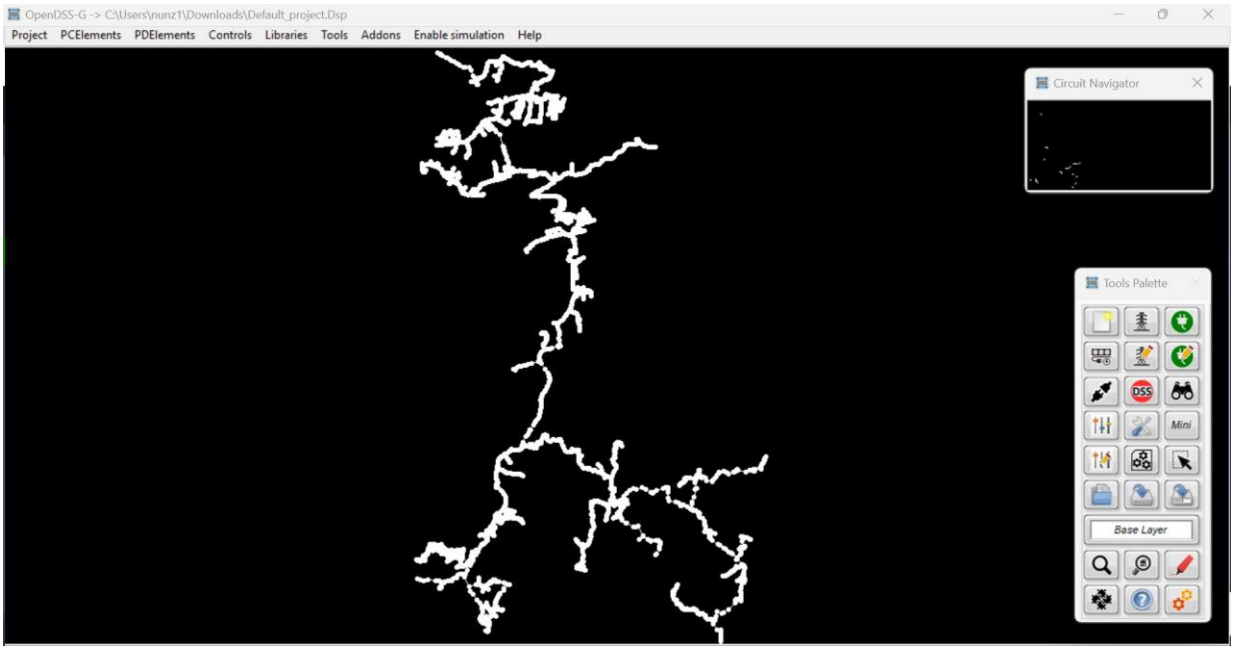
Luego, se redirige al archivo de líneas "Lista_Lineas_MV.dss", cuyo proceso de creación se describió anteriormente. Además, se crea una lista de buses en el sistema mediante el comando 'MakeBusList' y se importan las coordenadas de los buses desde el archivo creado anteriormente llamado "BusCoords.dss"

y se redirige la entrada del script a un archivo llamado “BusVoltageBases.dss” que contiene las tensiones bases del sistema, las cuales son 34.5kV, 0.480kV, 0.240kV y 0,208kV.

Por último, se resuelve el circuito con el comando ‘solve’ y se muestra la topología del circuito utilizando ‘show topology’. Por otra parte, se importó el archivo en OpenDSS-G para facilitar la visualización del circuito (observar la figura 3.7).

Figura 3.7

Circuito con las Líneas de Distribución Primarias



Fuente: OpenDSS (2024).

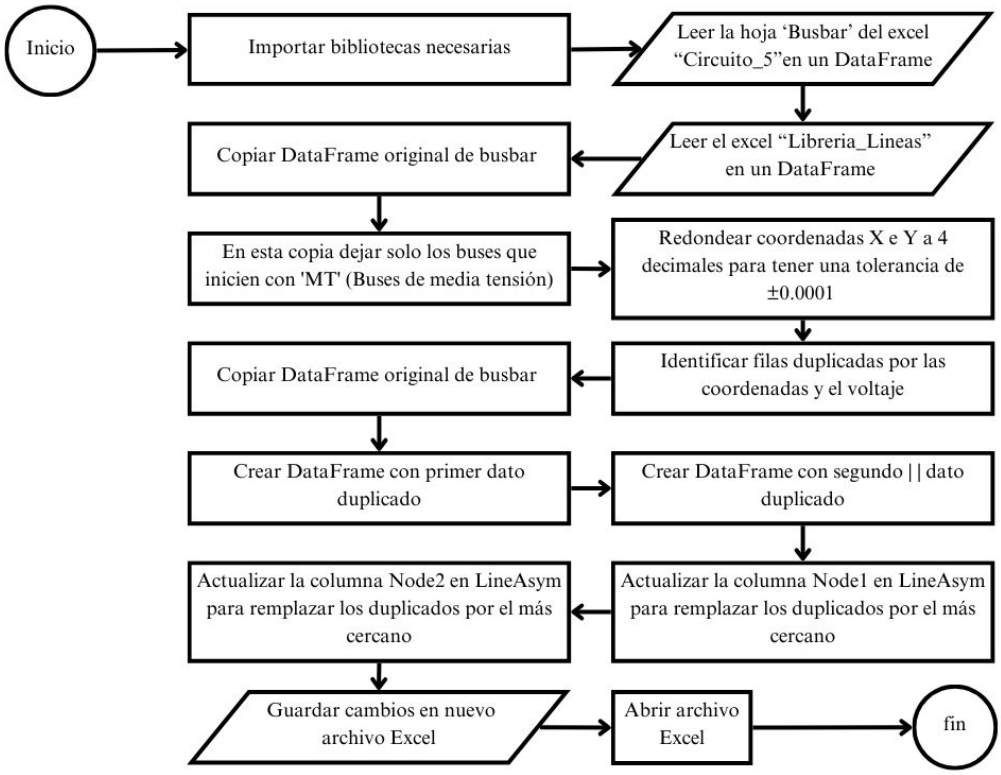
3.3.4. Programa en Python para la Unión de las Líneas Primarias

Al ejecutar el archivo “Master.dss” en OpenDSS y revisar resumen de la topología del circuito, se encontraron 5169 líneas desconectadas, como se puede observar en la figura 3.9. Por consiguiente, se debía encontrar la fuente del problema y una solución para unir las líneas. Al analizar la información, se observó que algunos buses compartían las mismas coordenadas o estaban tan próximos que podrían considerarse como un error de duplicación en el GIS.

Por lo tanto, se desarrolló un código en Python que elimina los buses con coordenadas repetidas o que están muy cercanos (utilizando una tolerancia de ±0,0001 para determinar la cercanía) y reemplaza automáticamente el bus duplicado en la hoja de LineAsym (para observar este código debe verse el apéndice D figura D.1). En la figura 3.8 se observa un diagrama de flujo que pretende facilitar la comprensión de este código.

Figura 3.8

Diagrama de Flujo del Código para la Unión de las Líneas Primarias



Fuente elaboración propia.

Este código comienza por importar la biblioteca Pandas para el manejo de datos y el módulo subprocess para ejecutar comandos del sistema. Luego, se carga el archivo de Excel "Circuito_5", utilizando la función 'pd.read_excel()' y se crean dos DataFrames, uno llamado 'busbaroriginal_df' y otro llamado 'lineas_asym_df', que contienen los datos de las hojas 'Busbar' y 'LineAsym' respectivamente.

Luego se procede a copiar el DataFrame original 'busbaroriginal_df' en un nuevo DataFrame llamado 'busbar_df' y se dejan solamente los buses en los que la columna 'Name' comienza con 'MT'; es decir, los buses de media tensión. Además, se redondean las coordenadas 'CoordX1' y 'CoordY1' a cuatro decimales asignando una tolerancia de ±0,0001. Para después identificar las filas duplicadas basadas en las columnas 'CoordX1' (Coordenada en X), 'CoordY1' (Coordenada en Y) y 'Un' (la tensión de la barra), almacenando los resultados en los objetos 'duplicados_primero' y 'duplicados_segundo'.

Se continúa creando dos DataFrames llamados 'primer_dato_repetido' y 'segundo_dato_repetido', que contienen las filas duplicadas de 'busbar_df', identificadas previamente en 'duplicados_primero' y 'duplicados_segundo' y a continuación, se crean otros dos DataFrames, 'first_df' y 'second_df', que contienen las filas de 'bus- bar_df' que no están presentes en 'segundo_dato_repetido' y 'primer_dato_repetido', respectivamente. Esto se hace utilizando la función pd.merge() para combinar los DataFrames y luego filtrando las filas usando el método query ().

Posteriormente, se crea una copia de ‘lineas_asym_df’ llamada ‘nuevo_lineas_asym_df’, para guardar los cambios que se hacen a continuación. Para cambiar de manera automática los buses repetidos en la hoja LineAsym hay dos bucles ‘for’ que iteran sobre las filas del DataFrame ‘second_df’, el primer ‘for’ para ‘Node1’ y el segundo para ‘Node2’. En cada iteración, se buscan coincidencias en ‘lineas_asym_df’ donde ‘Node1’ o ‘Node2’ coincidan con el nombre de nodo de la fila actual de ‘second_df’. Si se encuentra una coincidencia, se busca en ‘first_df’ la fila correspondiente que coincida con las coordenadas de la fila actual de ‘second_df’ y se cambia el valor de ‘Node1’ o ‘Node2’ en ‘nuevo_lineas_asym_df’ con el nombre de nodo correspondiente en ‘first_df’ (es decir el primer bus que tiene las coordenadas repetidas).

Finalmente, se guardan los cambios realizados en los DataFrames ‘primer_dato_repetido’ y ‘nuevo_lineas_asym_df’ en un nuevo archivo Excel llamado ‘Circuito5_actualizado.xlsx’, con las hojas ‘Busbar’ y ‘LineAsym’, respectivamente, el cual se abre automáticamente utilizando el módulo ‘subprocess’. Luego de ejecutar este programa todas las líneas quedan conectadas, como se observa en la figura 3.10.

Figura 3.9

Resumen de Topología del Circuito con los Datos sin Filtrar y Líneas Primarias

```
Topology analysis for switch control algorithms

950 Levels Deep
3 Loops
0 Parallel PD elements
0 Isolated PD components
0 Controlled Switches
```

Fuente: OpenDSS (2024).

Figura 3.10

Resumen de Topología del Circuito con los Datos Filtrados y Líneas Primarias

```
Topology analysis for switch control algorithms

34 Levels Deep
0 Loops
0 Parallel PD elements
5169 Isolated PD components
0 Controlled Switches
```

Fuente: OpenDSS (2024).

3.4. Modelado de los Transformadores

3.4.1. Programa en Python para la Creación de una Librería de Transformadores

En OpenDSS, la definición de transformadores se lleva a cabo mediante el objeto 'new transformer.'. Para definir este objeto, se requiere la siguiente información: el nombre único del transformador, el número de fases (phases), el número de devanados (windings), el porcentaje de reactancia entre los devanados de alta a baja (Xhl), entre alta y terciario (Xht), y entre baja y terciario (Xlt). También se necesita el porcentaje de resistencia de cada devanado, tanto en su componente resistiva como en la reactancia (Rs), las pérdidas sin carga (%noloadloss), el porcentaje de corriente de magnetización (%imag), un vector con el nombre y nodos de las barras de cada terminal, las tensiones nominales en cada devanado (kvs), la capacidad de cada devanado (kVAs) y la conexión de cada devanado (conns), indicando si es estrella (wye) o delta.

Todos los datos mencionados fueron suministrados por el ICE. Sin embargo, los valores correspondientes a Xhl, Xht, Xlt, Rs, %noloadloss y %imag fueron proporcionados en el archivo "trafoOperations.py". Debido a que este archivo de Python no era funcional, se tomó la decisión de realizar un código de Python que genera un archivo de Excel similar a "Libreria_Lineas" que contiene estos datos. Esto se hizo con el propósito de agilizar el proceso de definición de los transformadores.

En este código se realizaron tres funciones que contiene la información específica de los distintos tipos de librería que hay en las hojas 'Trafo2WindingAsym' y 'Trafo2Winding' del archivo "Circuito_5". La primera función que se observa en la figura E.1 del apéndice E contiene los datos para los transformadores monofásicos, la segunda función que se puede ver en la figura E.2 del apéndice E contiene la información para los transformadores trifásicos y la última función vista en la figura E.3 del apéndice E contiene los datos para los transformadores bifásicos. Estas tres funciones, tienen la misma estructura, toman un argumento library_type, que representa un tipo de biblioteca de valores para cierta clase de transformadores.

Luego, basado en el valor de 'library_type', la función devuelve un diccionario con valores asignados a diferentes columnas etiquetadas como 'Xhl', 'Xht', 'Xlt', ' %Rs', ' %noloadloss',' %imag', 'kVAs ' y 'kvs'. Cada tipo de 'library_type' tiene un conjunto específico de valores asociados que son devueltos en el diccionario. En caso de que 'library_type' no coincida con ninguno de los casos definidos, la función devuelve un diccionario vacío.

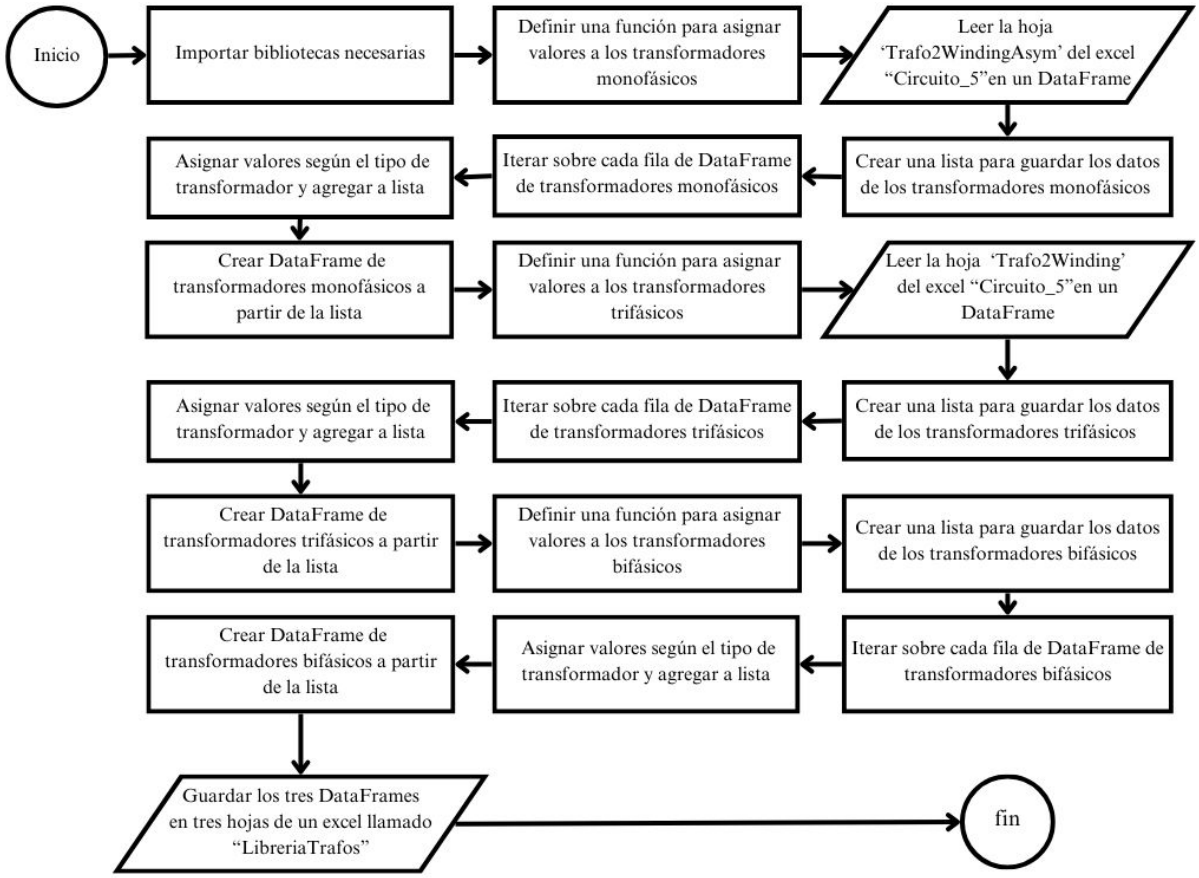
Luego de la función para asignar los parámetros a los transformadores monofásicos se carga un archivo de Excel desde una ruta especificada utilizando la biblioteca Pandas. Luego, se crea una lista vacía llamada 'new_rows' para almacenar las nuevas filas que se generarán. A continuación, itera sobre cada fila del DataFrame original ('df') utilizando el método 'iterrows ()'. Para cada fila obtiene el valor de la columna 'LibraryType' y lo asigna a la variable 'library_type'. Luego, utiliza la función 'asignar_valores (library_type)' para obtener un diccionario de valores correspondientes a ese tipo de biblioteca. Si

'Library_type' comienza con 'A_', 'B_', o 'C_', lo que se verifica utilizando 'startswith ()', se obtienen los valores correspondientes y se agregan a la lista 'new_rows' como un nuevo diccionario. Estos valores incluyen 'Xhl', 'Xht', 'Xlt', ' %Rs', ' %noloadloss', ' %imag', 'kvs' y 'kVAs'.

Finalmente, se crea un nuevo DataFrame llamado 'new_df' a partir de la lista de diccionarios 'new_rows', que contiene las filas filtradas y actualizadas con los valores correspondientes a los tipos de biblioteca que cumplen con las condiciones especificadas. Este mismo proceso se repite para los transformadores trifásicosy bifásicos. Por último, guarda los datos en un Excel llamado "LibreriaTrafos" con tres hojas nombradas como 'monofásicos', 'bifásicos' y 'trifásicos'. Para facilitar la comprensión de este código se realizó un diagrama de flujo que se puede observar en la figura 3.11.

Figura 3.11

Diagrama de Flujo del Código para la Creación del Archivo de Librería de Transformadores



Fuente: Elaboración propia.

La designación utilizada en libraryType para los transformadores da información sobre el transformador que se está modelando, por ejemplo: "A_15.0_34.5_kV_.240_kV_1" se desglosa de la siguiente manera:

- '_A': indica que la línea está conectada a la fase A y que es monofásico.
- 15: significa que la potencia de cada devanado es de 15kVA.
- 34,5kV: indica que el primer devanado tiene una tensión nominal de 34,5kV.
- .240kV: indica que el primer devanado tiene una tensión nominal de 0,240kV.

En la tabla 3.8 se mostrarán los significados de algunos de los códigos usados en LibraryType para los transformadores.

Tabla 3.8

LibraryType para los Transformadores

Designación	Número de Fases	Devanado primario	Representación de nodo en devanado primario	Representación de nodo en devanado secundario
A	1	Conexión a la Fase A	.1.0	Devanado 1: .1.0 Devanado 2: .0.2
B	1	Conexión a la Fase B	.2.0	Devanado 1: .1.0 Devanado 2: .0.2
C	1	Conexión a la Fase C	.3.0	Devanado 1: .1.0 Devanado 2: .0.2
AB Unidad A Unidad B	2 1 1	 Conexión a la fase A Conexión a la fase A	 .1.0 .2.0	 .2.1 .3.2
BC Unidad B Unidad C	2 1 1	 Conexión a la fase A Conexión a la fase A	 .2.0 .3.0	 .2.1 .3.2
AC Unidad B Unidad C	2 1 1	 Conexión a la fase A Conexión a la fase A	 .1.0 .3.0	 .2.1 .3.2
ABC	3	Conexión a las fases ABC	.1.2.3	.1.2.3

Fuente: Elaboración propia.

3.4.2. Programa en Python para la Definición de los Transformadores

Para definir los transformadores se creó un archivo llamado “trafos.dss”, el cual contiene la siguiente información para cada transformador: el nombre único del transformador, el número de fases (phases), el número de devanados (windings), el porcentaje de reactancia entre los devanados de alta a baja (Xhl), entre alta y terciario (Xht) y entre baja y terciario (Xlt). También se necesita el porcentaje de resistencia de cada devanado, tanto en su componente resistiva como en la reactancia (Rs), las pérdidas sin carga (%noloadloss), el porcentaje de corriente de magnetización (%imag), un vector con el nombre y nodos de las barras de cada terminal, las tensiones nominales en cada devanado (kvs), la capacidad de cada devanado (kVAs) y la conexión de cada devanado (conns), indicando si es estrella (wye) o delta. En la figura 3.12 se puede observar una parte de este archivo.

Figura 3.12:

Parte del Archivo de Transformadores trafos.dss

```
new transformer.291178_T phases=1 windings=3 Xhl=5.52 Xht=5.52 Xlt=3.68 %Rs=[0.92  1.83  1.83]
%noloadloss=0.47 %imag=1.0 Buses=[MT_AREA_432812_157258.1.0 BT_AREA_432812_157258.1.0
BT_AREA_432812_157258.0.2] kvs=[19.92  0.12  0.12] kVAs=[15  15  15] conns=[wye  wye  wye]
new transformer.312012_T phases=1 windings=3 Xhl=5.34 Xht=5.34 Xlt=3.56 %Rs=[0.98  1.95  1.95]
%noloadloss=0.5 %imag=1.0 Buses=[MT_AREA_426634_358111.1.0 BT_AREA_426634_358111.1.0
BT_AREA_426634_358111.0.2] kvs=[19.92  0.12  0.12] kVAs=[10  10  10] conns=[wye  wye  wye]
new transformer.325514_T phases=1 windings=3 Xhl=5.89 Xht=5.89 Xlt=3.93 %Rs=[0.68  1.36  1.36]
%noloadloss=0.34 %imag=1.0 Buses=[MT_AREA_434415_731283.1.0 BT_AREA_434415_731283.1.0
BT_AREA_434415_731283.0.2] kvs=[19.92  0.12  0.12] kVAs=[50  50  50] conns=[wye  wye  wye]
```

Fuente: elaboración propia.

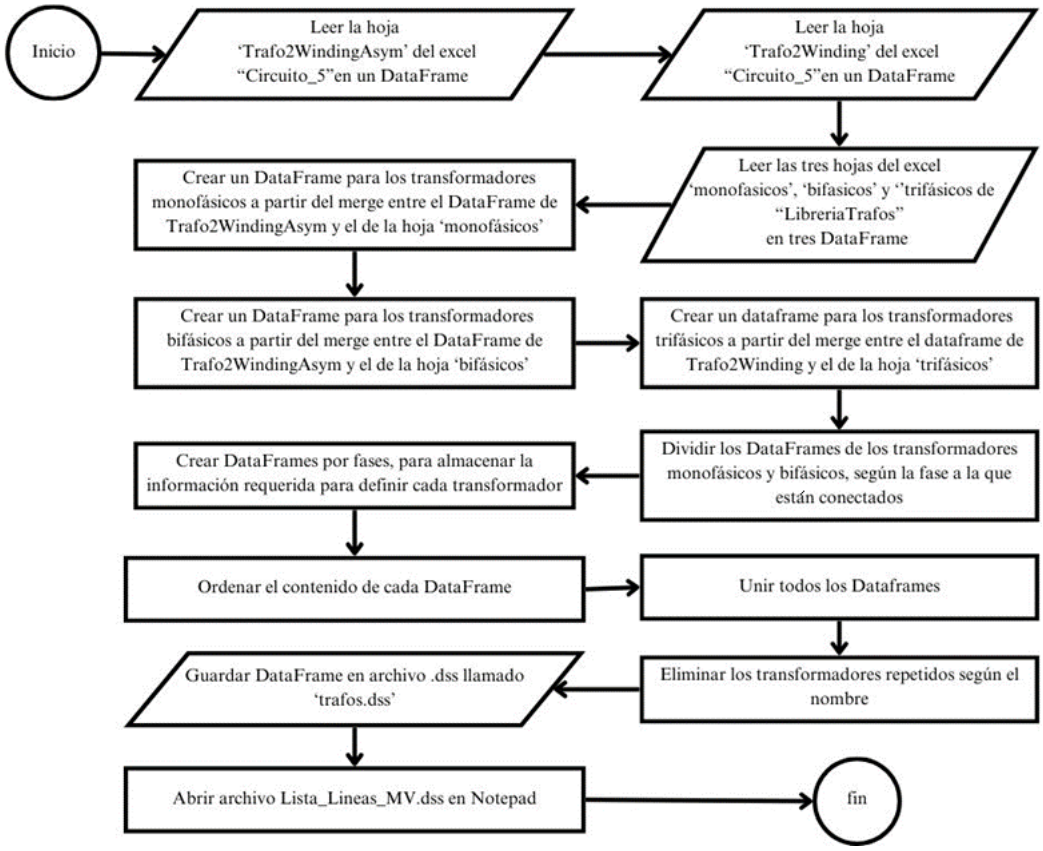
‘Archivo de transformadores’ al archivo “GeneradorArchivoDSS.py". Esta sección procesa los datos de los transformadores y genera un archivo de texto en el formato DSS utilizando la misma lógica y pasos que se emplearon en la sección para definir las líneas.

Como se puede observar en el apéndice A, figura A.5, el proceso inicia definiendo la ruta del archivo Excel llamado “LibreriaTrafos”, creado previamente. Luego, se crean tres DataFrames, uno para cada una de las tres hojas (‘monofásicos’, ‘bifásicos’ y ‘trifásicos’) de este archivo. Posteriormente, se lee la hoja ‘Trafo2WindingAsym’ del archivo “Circuito_5”, la cual contiene la información sobre los transformadores monofásicos y bifásicos. A partir de esta lectura, se crean dos DataFrames: uno para los transformadores monofásicos y otro para los transformadores bifásicos. Esto se logra utilizando la función merge, la cual fusiona automáticamente los datos de ‘Trafo2WindingAsym’ con los parámetros de la hoja ‘monofásicos’, para el DataFrame de los transformadores monofásicos y después fusiona los datos de ‘Trafo2WindingAsym’ con los parámetros de la hoja ‘bifásicos’, para el DataFrame de los transformadores bifásicos. Luego se creó otro DataFrame, para los transformadores trifásicos, a partir de la lectura de la hoja ‘Trafo2Winding’ y utilizando la función merge se fusionan los datos de esta hoja con los parámetros de la hoja ‘Trifásicos’.

Después, se procedió a dividir los DataFrames de los transformadores monofásicos y bifásicos en función de la fase a la que estuvieran conectados en su devanado primario. Para cada fase se creó un nuevo DataFrame con el propósito de definir los transformadores, asignando valores a diferentes columnas. Estos valores incluyen el nombre del transformador, el número de fases, el número de devanados, la reactancia de alta tensión (Xhl), la reactancia de alta tensión (Xht), la reactancia de baja tensión (Xlt), la pérdida en vacío (%Rs), la pérdida sin carga (%noloadloss), la corriente de magnetización (%imag), los buses a los que están conectados el transformador, la tensión nominal (kVs), la capacidad nominal (kVAs) y las conexiones de los devanados. Esto se realizó a partir de los DataFrames de los transformadores por fase (observar apéndice A, figuras A.5 y A.6). En la figura 3.13 se muestra un diagrama de flujo del código que genera el archivo de transformadores.

Figura 3.13

Diagrama de Flujo del Código para la Creación del Archivo que Define los Transformadores



Fuente: Elaboración propia.

Los transformadores bifásicos se modelaron utilizando dos transformadores monofásicos. Por lo tanto, para cada transformador bifásico hay dos unidades monofásicas. La diferencia entre los nombres de estas unidades monofásicas se encuentra en la fase a la que están conectadas. Por ejemplo, si el transformador bifásico está conectado en las fases AB, el primer transformador se nombra como 'name_A' y el segundo como 'name_B', además en las barras se indica la fase a la que están conectadas por su sufijo. Para los transformadores trifásicos de la misma manera se creó un DataFrame que para definir los parámetros de los transformadores. Luego estos transformadores se acomodaron en el orden indicado y se unieron todos los DataFrames en uno solo para guardar los datos en un archivo llamado "trafos.dss"

Luego de crear este archivo, se añaden los transformadores al archivo "Master.dss", usando el comando 'Redirect'. En el apéndice C, figura C.2 se muestra esta modificación del archivo.

3.5. Modelado de las Líneas de Distribución Secundarias

3.5.1. Cálculo de las Matrices de Impedancia

Para comenzar el modelado de las líneas de distribución secundarias en OpenDSS, al igual que con las líneas primarias, primero se llevaron a cabo los cálculos de las matrices de impedancia faltantes, en este caso faltaban todas las líneas secundarias aéreas. Este proceso se realizó utilizando OpenDSS. Para esto se definieron tres objetos clave: WireData (para las características de los conductores), spacing (para el espaciado entre conductores) y Geometry (para la geometría de los conductores); se generó un LineCode para estas líneas, el cual se puede observar en la figura 3.14.

Figura 3.14

LineCode de Líneas de Distribución Secundarias

```
!--- OpenDSS Linecodes file generated from Show LINECONSTANTS command
!--- Frequency = 60 Hz, Earth resistivity = 100 ohm-m
!--- Earth Model = Carson

New Linecode.bt_123.3aaac_aaac_123.3_1 nphases=2  Units=m
~ Rmatrix=[0.000181696  |6.35487E-005  0.000189473  ]
~ Xmatrix=[0.000482839  |0.00021401  0.00041147  ]
~ Cmatrix=[0.0129506  |-0.00601109  0.0150906  ]

New Linecode.bt_1/0co_tr_co_tr_1/0_1 nphases=2  Units=m
~ Rmatrix=[0.000475449  |0.000135757  0.000494534  ]
~ Xmatrix=[0.000614424  |0.000278754  0.000555397  ]
~ Cmatrix=[0.0102251  |-0.00417494  0.0113971  ]

New Linecode.bt_1/0acsr_acsr_2_1 nphases=2  Units=m
~ Rmatrix=[0.00091218  |0.000214589  0.00094231  ]
~ Xmatrix=[0.00073931  |0.000426335  0.000706965  ]
~ Cmatrix=[0.010386  |-0.0043314  0.0115607  ]

New Linecode.bt_1/0aaac_aaac_2_1 nphases=2  Units=m
~ Rmatrix=[0.000839018  |0.000214589  0.000869148  ]
~ Xmatrix=[0.000743584  |0.000426335  0.000711239  ]
~ Cmatrix=[0.010386  |-0.0043314  0.0115607  ]

New Linecode.bt_1/0aaac_aaac_1/0_1 nphases=2  Units=m
~ Rmatrix=[0.000821038  |0.000195681  0.00084929  ]
~ Xmatrix=[0.000669398  |0.000345342  0.000622822  ]
~ Cmatrix=[0.0104079  |-0.0042912  0.0116346  ]
```

Fuente: IEEE Power Engineering Society (1988).

El código de OpenDSS mostrado en el apéndice B, figuras B.2 y B.3 es el que se utilizó para calcular las impedancias para todos los tipos de línea faltantes. En primer lugar, este código establece la configuración del circuito eléctrico, incluyendo la base de tensión, ángulo, frecuencia y otras características relevantes. Luego, se detallan las propiedades de los conductores mediante la referencia a archivos que contienen información específica, como la resistencia en corriente alterna (Rac), el radio medio geométrico (GMR), el diámetro externo del conductor (Diam) y la corriente nominal del conductor (Normamps).

Posteriormente, se define el espaciado entre conductores y se define la geometría de la línea para cada tipo de línea faltante. Luego, se calculan las impedancias y capacitancias correspondientes y se resuelve el sistema para obtener las matrices de impedancia. Este proceso se repite para todas las líneas faltantes de matriz de impedancias y cuando se resuelve el código, todas estas se guardan en un archivo de LineCode. Este LineCode se nombró 'LineCodeSecundarias' y define las características específicas de las líneas.

3.5.2. Programa en Python para la Definición de las Líneas Distribución Secundarias

La sección del código, que se muestra en el Anexo A, figuras A.8 y A.9, es la encargada de generar el archivo donde se definen las líneas de distribución secundarias. Esta sección de código al igual que la sección de las líneas de media tensión trabaja con el DataFrame 'line_asym_df', porque como se mencionó en la sección de obtención de datos, la hoja 'LineAsym' del documento "Circuito_5" contiene los datos de todas las líneas del circuito. Para poder definir las líneas secundarias o de baja tensión primero se crean distintos DataFrames según el tipo y características de la línea. Primero, se hace un DataFrame para las líneas subterráneas, seguido otras líneas secundarias aéreas, otro para las líneas acometidas tríplex y finalmente otro para las líneas acometidas de tipo cuádruplex. Los conductores tríplex y cuádruplex son dos tipos de cables, el tríplex consiste en tres cables conductores aislados y entrelazados, mientras que el cuádruplex está compuesto por cuatro cables conductores del mismo tipo. (Thue, 2017). Estos cables se utilizan principalmente para la conexión entre el transformador secundario y el consumidor.

Posteriormente, se hace un nuevo DataFrame para las líneas secundarias que contiene la información necesaria para la definición de las líneas. En OpenDSS, como se indicó anteriormente, para definir una línea se utiliza el objeto 'new line.' y luego se debe especificar la siguiente información:

- Name: Nombre único asignado a la línea.
- Phases: Indica el número de fases de la línea.
- Bus1 y Bus2: Nombres de los buses y los nodos donde la línea está conectada (para las líneas secundarias se indicó que todas estaban conectadas a las fases A y B, por lo que los nodos para todas eran .1.2).

- Length: Longitud de la línea en km.
- Linecode: Nombre del código de línea asociado que define los parámetros eléctricos de la línea, como la resistencia, la reactancia y la capacidad de cortocircuito.
- Units: Unidades de medida asociadas a los parámetros de la línea, como kilómetros o millas para la longitud.

En el caso de las líneas acometidas (TRIPLEX y CUADRUPLEX) y líneas subterráneas, antes de crear un nuevo DataFrame con los datos de las líneas, se crearon tres funciones para asignarles un tipo de LineCode a cada una. En el caso de las acometidas, se le asignó un nombre especificado en el archivo 'LineCodeAcometidas.dss' dado por el ICE, que contiene las características de las líneas acometidas tríplex y cuádruplex, a cada designación de la columna 'LibraryType' de 'LineAsym' (se usaron dos funciones distintas para simplificar el proceso de asignación del LineCode).

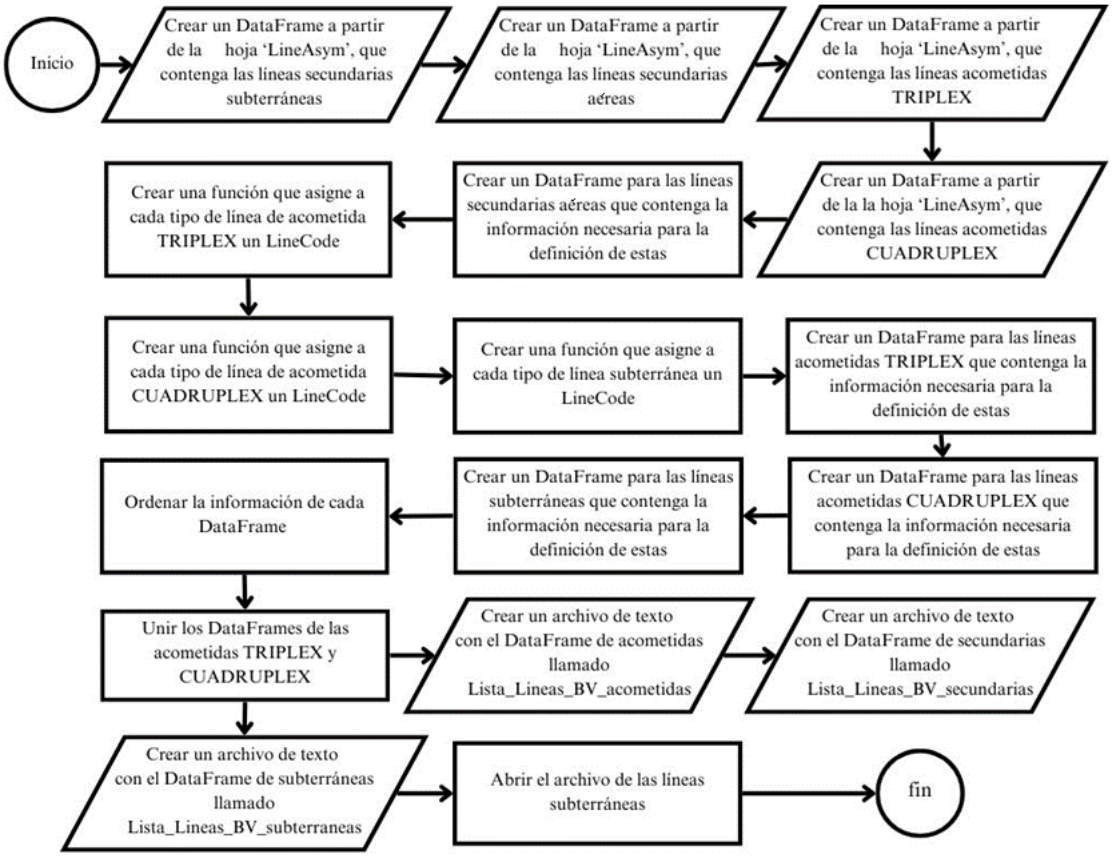
Luego, para asignar un Linecode a las líneas subterráneas, a cada tipo de línea especificado en la columna 'LibraryType' de 'LineAsym', se le asignó el nombre de LineCode equivalente del archivo 'LineCodeSubterraneas.dss' dado por el ICE. Para las líneas secundarias aéreas se aseguró que cuando se realizó el LineCode, en la sección de cálculo de las matrices de impedancias para las líneas secundarias, el nombre del LineCode para cada tipo de línea coincidiera con el nombre asignado en la columna 'LibraryType' de 'LineAsym'.

Luego se hacen los nuevos DataFrames con los datos necesarios para la definición de las líneas secundarias subterráneas y las acometidas. Una vez organizados todos los DataFrames y completadas las columnas necesarias, se concatenan y se eliminan duplicados. Luego, los datos son exportados a 3 archivos de texto con extensión .dss llamados: "Lista_Lineas_BV_acometidas", "Lista_Lineas_BV_secundarias" y "Lista_Lineas_BV_subterraneas". Por último, el archivo de líneas subterráneas se abre automáticamente en el Bloc de notas para asegurarse de que estos se hicieron correctamente.

Para facilitar el entendimiento de este código se realizó un diagrama de flujo que se observa en la figura 3.15.

Figura 3.15

Diagrama de Flujo para la Definición de Líneas de Baja Tensión



Fuente: elaboración propia.

Luego de crear estos tres archivos, se añaden las líneas secundarias al archivo ”Master.dss”, usando el comando ’Redirect’. En el apéndice C, figura C.3 se muestra esta modificación del archivo.

Figura 3.16

Parte del Archivo de Líneas de Baja Tensión Lista_Lineas_BV_acometidas.dss

```
new line.401129___0 bus1=BT_AREA_429663_8184.1.2 bus2=BT_AREA_429665_7786.1.2
length=0.004479549358021148 units=km Linecode=tpx_6_Hippa
new line.401129___1 bus1=BT_AREA_429665_7786.1.2 bus2=BT_AREA_429635_6291.1.2
length=0.04863058473331125 units=km Linecode=tpx_6_Hippa
new line.401146___0 bus1=BT_AREA_430811_771892.1.2 bus2=BT_AREA_430811_771892.1.2
length=0.005094513631549532 units=km Linecode=tpx_1/0_Leda
```

Fuente: elaboración propia.

3.5.3. Programa en Python para la Unión de las Líneas de Distribución Secundarias

Al incorporar las líneas de distribución secundaria al circuito, se observa que 29 de estas líneas se encuentran desconectadas. Para abordar esta situación, se desarrolló un nuevo programa destinado a conectar estas líneas. En este proceso, se empleó el archivo generado por OpenDSS que muestra la topología del circuito llamado “circuito5_TopoTree”, facilitando la identificación de las líneas aisladas. En la figura 3.17 se muestra el resumen de la topología del circuito y en la figura 3.18 se muestra una parte del árbol de la topología del circuito donde están las líneas aisladas.

Figura 3.17

Resumen Topología del Circuito Luego de Conectar las Líneas de Baja Tensión

```
Topology analysis for switch control algorithms

952 Levels Deep
404 Loops
0 Parallel PD elements
29 Isolated PD components
0 Controlled Switches
```

Fuente: OpenDSS (2024).

Figura 3.18

Parte del Árbol de Topología del Circuito

Fuente: OpenDSS (2024).

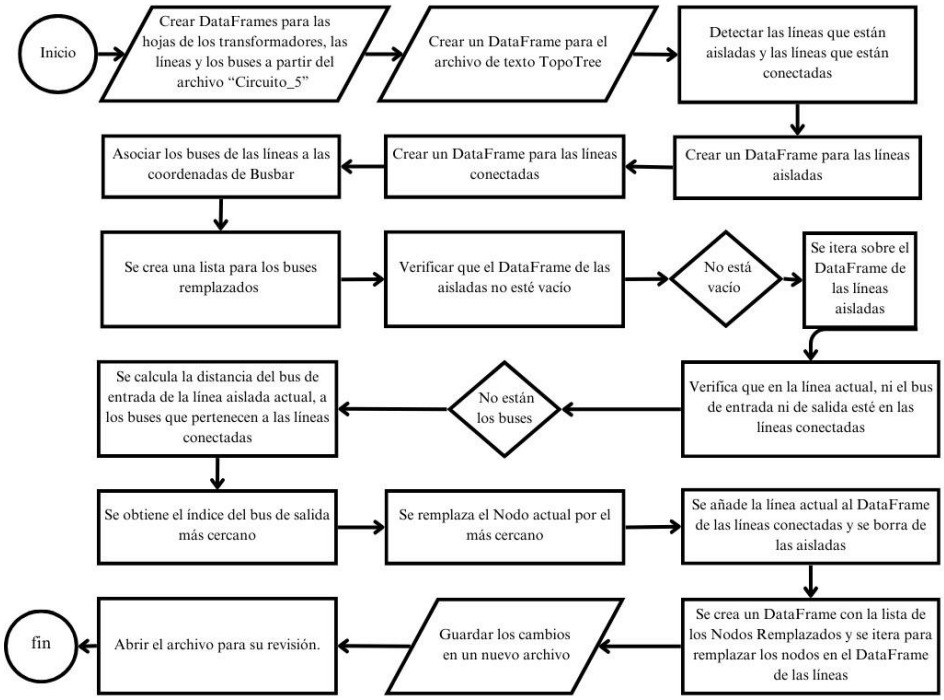
El código de unión de líneas, que se muestra en el anexo F, figuras F.1 y F.2, comienza importando las bibliotecas necesarias para el análisis de datos (pandas), el cálculo de distancias (scipy.spatial.distance) y la ejecución de comandos del sistema (subprocess). Luego, se definen las rutas de los archivos Excel y de texto que contienen la información del circuito y las líneas aisladas, respectivamente. A continuación, se leen los datos del archivo Excel que contienen información sobre los nodos ('Busbar'), las líneas del circuito ('LineAsym') y los transformadores ('Trafo2WindingAsym' y 'Trafo2Winding'), así como del archivo de texto que proporciona detalles sobre las líneas aisladas.

Posteriormente, se filtran las líneas aisladas y no aisladas (en las líneas no aisladas se incluyen los transformadores, porque los buses de los transformadores unen las líneas de baja tensión con las líneas de media tensión) del archivo de texto y se almacenan en DataFrames separados. Luego se asocian las coordenadas de los nodos a las líneas aisladas y las líneas conectadas del circuito; y se empieza con la conexión de las líneas aisladas, para esto se itera en las líneas aisladas y se busca el nodo más cercano en las líneas conectadas para reemplazarlo.

Finalmente, se guardan los cambios en un nuevo archivo Excel y se abre automáticamente el archivo actualizado. Luego de esto se actualiza el Excel 'Circuito_5' de manera manual remplazando la hoja LineAsym y las líneas quedan unidas, para facilitar la comprensión de este proceso se realizó un diagrama de flujo que se observa en la figura 3.19. En la figura 3.20 se puede observar un resumen de la topología del circuito luego de la unión de las líneas.

Figura 3.19

Diagrama de Flujo del Código que une las Líneas de Baja Tensión



Fuente: OpenDSS (2024).

Figura 3.20

Resumen Topología del Circuito Luego de Conectar las Líneas Aisladas de Baja Tensión

```
Topology analysis for switch control algorithms

952 Levels Deep
404 Loops
0 Parallel PD elements
0 Isolated PD components
0 Controlled Switches
```

Fuente: OpenDSS (2024).

3.6. Modelado de las Cargas

Para modelar las cargas del sistema se llevaron a cabo dos análisis distintos, con el objetivo de fortalecer la comprensión del circuito. En primer lugar, se simularon las cargas como estáticas, empleando los consumos promedio habituales mensuales proporcionados por el ICE. Esta fase permitió obtener unavisión general de la distribución de la carga en el sistema. Posteriormente, en una segunda etapa, se incorporaron seis curvas de demanda dinámica para simular las variaciones y picos de consumo a lo largo del día. Estas curvas representaban las fluctuaciones esperadas basadas en diversos escenarios, desde consumos bajos hasta situaciones de alto consumo.

3.6.1. Programa en Python para la Definición de las Cargas Estáticas

En el apéndice A, figura A.10 se muestra la sección del código que genera el archivo para definir las cargas estáticas. Este código en Python comienza leyendo la hoja ‘Load’ en un DataFrame llamado “cargas_df”. Luego, se crea un nuevo DataFrame llamado “new_df_cargas_1”, donde se especifica la información necesaria para definir las cargas; en OpenDSS las cargas se definen con el objeto ‘new load.’ e inmediatamente después del punto es necesario especificar la siguiente información:

- Name: identificador único de la carga.
- Bus1: nombre de la barra a la que está conectada la carga.
- kV: tensión nominal de la carga.
- Model: modelo de la carga (en este proyecto se usó el modelo de potencia constante).

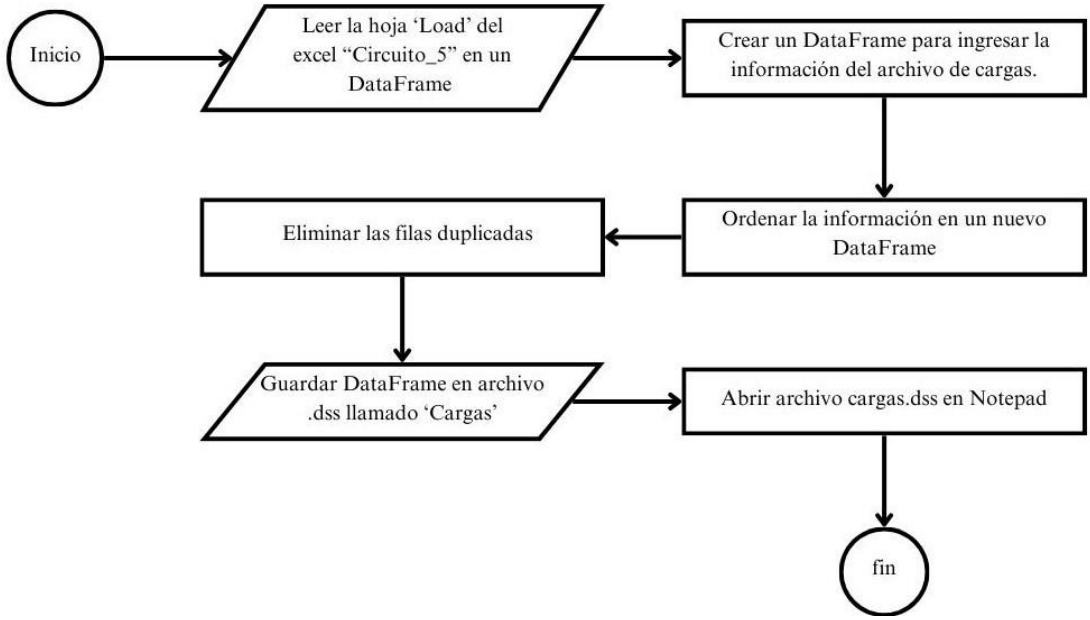
- Conn: tipo de conexión de la carga (delta, wye).
- kW: potencia nominal de la carga.
- pF: factor de potencia de la carga.
- Status: fija o variable (en este caso va a ser fija).
- Phases: número de fases de la carga (1 o 3 para monofásica o trifásica).

Después, se crea otro DataFrame llamado ‘output_cargas’ que se encarga de ordenar las columnas del DataFrame anterior y luego se elimina cualquier fila duplicada basada en la columna ‘Name’ y se almacena el resultado en un nuevo DataFrame llamado ‘output_cargas_df’.

Posteriormente, el DataFrame ‘output_cargas_df’ se guarda en un archivo llamado “Cargas.dss”, utilizando la función ‘to_csv’ de pandas. Por último, el código utiliza la biblioteca “subprocess” para abrir automáticamente el archivo en Notepad. En la figura 3.21 se muestra un diagrama de flujo que explica el funcionamiento de este código. Por otra parte, en la figura 3.22 se muestra una parte del archivo de cargas estáticas.

Figura 3.21

Diagrama de Flujo del Código Generador del Archivo de Cargas Estáticas



Fuente: elaboración propia.

Figura 3.22

Parte del Archivo de Cargas Estáticas cargas.dss

```
new load.87140___0 bus1=BT_AREA_431310_873973 kV=0.24 model=1 conn=wye kW=1.31666666666667
pF=0.95 status=fixed phases=1
new load.87165___1 bus1=BT_AREA_427720_355503 kV=0.24 model=1 conn=wye kW=4.96111111111111
pF=0.95 status=fixed phases=1
new load.91777___2 bus1=BT_SUB_427709_420947 kV=0.24 model=1 conn=wye kW=21.7597222222223
pF=0.95 status=fixed phases=1
```

Fuente: elaboración propia.

En el anexo C, figura C.4 se puede observar el archivo “Master.dss” con la unión del archivo de definición de las cargas.

Por otra parte, se configuró en este código un monitor para seguir la tensión en la fuente de tensión llamada ‘Vsource.source’ en su terminal 1. Se establece en modo de análisis diario con un paso de tiempo de 1 minuto y un total de 24 pasos.

Luego, se creó un medidor de energía llamado ‘EnergyMeter.sub’ en la línea identificada como ‘Line.445411___0’ en su terminal 1. Estas adiciones al código se realizaron con el fin de obtener resultados visuales del circuito. La creación del monitor y del medidor de energía permite analizar comportamiento de la tensión y la energía en diferentes partes del sistema. Al establecer el modo de análisis en diario con un paso de tiempo definido, se garantiza la recopilación de datos a lo largo del tiempo, lo que facilita la visualización de tendencias y patrones en el rendimiento del circuito, mientras que la exportación de los datos del monitor facilita la generación de gráficos y la realización de análisis posteriores.

3.6.2. Programa en Python para la Definición de las Cargas Dinámicas

La definición de las cargas dinámicas se basa esencialmente en la misma información que se utiliza para las cargas estáticas, con la diferencia de que se establece el ‘status’ como variable. Además, se incorpora un parámetro adicional denominado ‘daily’, el cual incluye la curva de demanda diaria. En cuanto al código que se utilizó para definir las cargas dinámicas (ver apéndice A, figura A.11) genera seis documentos, uno para cada curva de demanda utilizada. Cada uno de los documentos fueron nombrados de la siguiente manera:

- cargasdinamicas_1 cuyo parámetro daily es loadshape_1.
- cargasdinamicas_2 cuyo parámetro daily es loadshape_2.
- cargasdinamicas_3 cuyo parámetro daily es loadshape_3.
- cargasdinamicas_4 cuyo parámetro daily es loadshape_4.
- cargasdinamicas_5 cuyo parámetro daily es loadshape_5.

- cargasdinamicas_6 cuyo parámetro daily es loadshape_6.

Para generar estos archivos primero se hizo una lista vacía llamada ‘new_df_cargasdinamicas_list’, donde se almacenan DataFrames individuales. Luego, se utiliza un bucle ‘for’ para establecer el contenido de los 6 documentos. Dentro de este bucle, se crea un DataFrame denominado ‘new_df_cargasdinamicas’, que establece las características de las cargas, obtenidos del DataFrame ‘cargas_df’. Estos DataFrames individuales se agregan a la lista ‘new_df_cargasdinamicas_list‘.

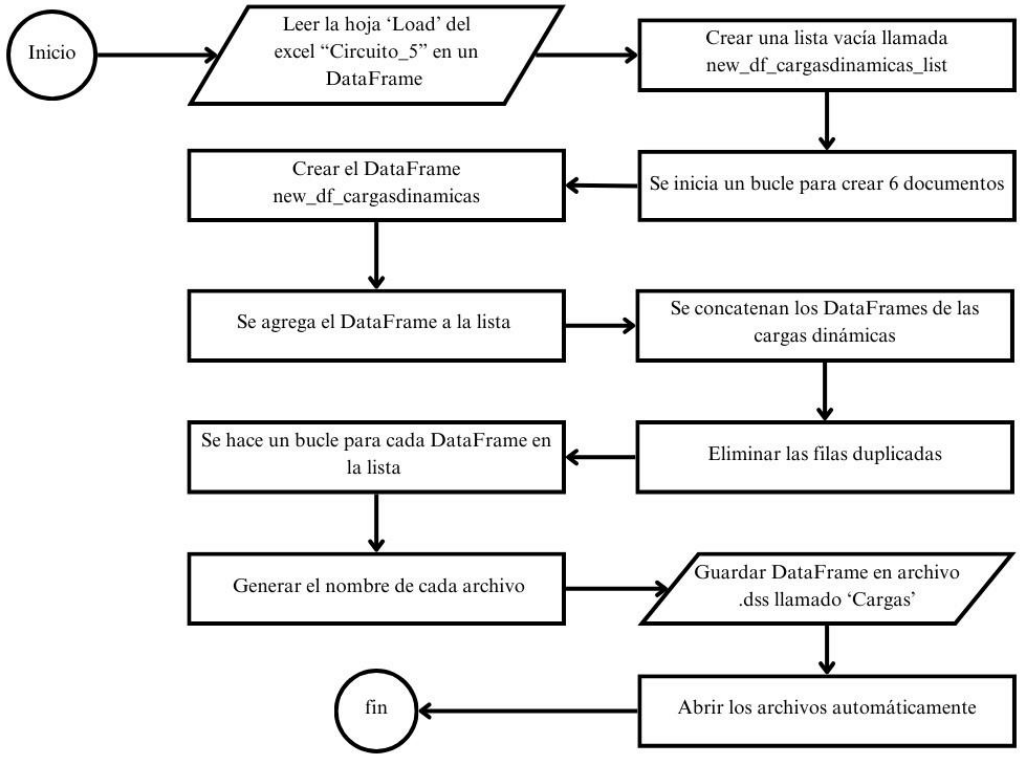
Posteriormente, se realiza la concatenación de todos los DataFrames en la lista ‘new_df_cargasdinamicas_list’ en un solo DataFrame llamado ‘output_cargasdinamicas’, utilizando el método ‘pd.concat ()’. Se procede a eliminar las filas duplicadas del DataFrame resultante, basándose en la columna ’Name’, para garantizar la integridad de los datos.

Finalmente, se define un sufijo ‘output_file_cargasdinamicas_prefix’ para los nombres de los archivos de salida y se itera sobre los DataFrames en ‘new_df_cargasdinamicas_list’ utilizando otro bucle ‘for’. En cada iteración, se genera el nombre del archivo de salida. Los datos del DataFrame se escriben en el archivo de texto con un formato específico utilizando ‘df.to_csv()’.

Para finalizar, se abre automáticamente cada archivo .dss en el Bloc de notas mediante ‘subprocess.run()’. En la figura 3.23 se encuentra un diagrama de flujo que describe este código y en la figura 3.24 se encuentra una parte de uno de los archivos generados por este código.

Figura 3.23

Diagrama de Flujo del Código que Genera los Archivos de las Cargas Dinámicas



Fuente: elaboración propia.

Figura 3.24

Parte del Archivo de Cargas Dinámicas cargasdinamicas_1.dss

```
new load.87140___0 bus1=BT_AREA_431310_873973 kV=0.24 model=1 conn=wye kW=1.31666666666667
pF=0.95 status=variable phases=1 daily=loadshape_1
new load.87165___1 bus1=BT_AREA_427720_355503 kV=0.24 model=1 conn=wye kW=4.96111111111111
pF=0.95 status=variable phases=1 daily=loadshape_1
new load.91777___2 bus1=BT_SUB_427709_420947 kV=0.24 model=1 conn=wye kW=21.7597222222223
pF=0.95 status=variable phases=1 daily=loadshape_1
```

Fuente: elaboración propia.

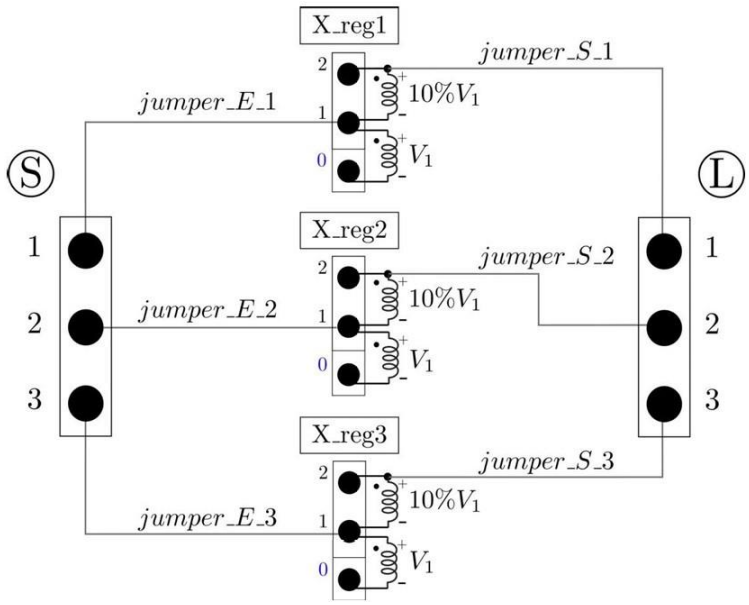
3.7. Modelado de los Reguladores de Tensión

En este circuito se emplearon dos reguladores de tensión de tipo A denominados 8011_R (ver apéndice G, figura G.1) y 8012_R (ver apéndice G figura G.2) para estabilizar el sistema en OpenDSS. La simulación de un regulador de tensión tipo A se logra conectando un transformador como autotransformador, como se muestra en la figura 3.25. En OpenDSS, se utiliza el objeto new RegControl. para modelar las acciones de control de un regulador de tensión. Los parámetros necesarios para definir un regulador de tensión son:

- Name: Nombre del regulador. No pueden haber 2 o más RegControl con el mismo nombre.
- Transformer: Nombre del transformador al cual el RegControl le controlará la relación de vueltas.
- Winding: Devanado que controla el RegControl. 1 para primario, 2 para secundario.
- Vreg: Tensión de referencia. Generalmente es 120 V.
- Band: Banda de tensión deseada de la tensión controlada. Si es 2, la tensión de ese transformador estará entre 119 y 121 V.
- Ptratio: Relación de transformación del transformador de potencial. El producto de ptratio * Vreg debe de ser igual a la tensión deseada en el devanado del trafo.
- Ptphase: Fase por monitorear. Puede ser 1, 2 o 3 para A, B o C respectivamente. También MIN, MAX o AVG

Figura 3.25

Autotransformador Usado como Regulador de Tensión Tipo A



Fuente: ARESEP (2023).

Para agregar un nuevo RegControl es necesario haber definido previamente el modelo del autotransformador. Para un regulador trifásico, el autotransformador se define a partir de tres unidades monofásicas.

Para cada una de estas unidades, se especifica su nombre, el número de fases, así como otros parámetros relevantes como la impedancia de dispersión (xhl), el porcentaje de pérdidas en carga (%loadloss), entre otros. (ARESEP, 2023).

El autotransformador debe tener la siguiente relación de vueltas:

$$n_t = \frac{V_{\text{serie}}}{V_{\text{shunt}}} \tag{3.1}$$

Como en este caso se desea dos reguladores a 34.5 kV (19.92 kV de fase) cada uno de ellos con regulación de ±10 %, V_{shunt} = 19,92 kV y V_{serie} = 1,992 kV → n_t = 0,1.

Cada transformador debe tener la siguiente capacidad:

$$\text{KVA}_{\text{trafo}} = \text{KVA}_{\text{auto}} \frac{n_t}{1 + n_t} \tag{3.2}$$

En este caso la capacidad del regulador trifásico es de 3000 kVA (o 1000 kVA por fase), por lo que la potencia del transformador monofásico usado como autotransformador es:

$$\mathrm{KVA}_{\mathrm{trafo}} = 1000\frac{0{,}1}{1+0{,}1} = 90{,}91 \text{ kVA} \tag{3.3}$$

El ptratio utilizado para los dos reguladores de tensión de este circuito se calculó de la siguiente manera:

$$ptratio = \frac{34500}{\sqrt{3}} \cdot \frac{1}{120} = 166 \tag{3.4}$$

Capítulo 4

Análisis y evidencias gráficas

Tras la finalización del modelado del circuito eléctrico de distribución, se procedió con un breve análisis utilizando OpenDSS y OpenDSS-G, con la intención de validar la funcionalidad del modelo desarrollado. OpenDSS y OpenDSS-G son herramientas versátiles que permiten realizar una amplia gama de estudios. Estas herramientas implementan reportes detallados que brindan información valiosa sobre el comportamiento del sistema, como tensiones línea-línea, tensiones línea-neutro, tensiones en los diferentes elementos, corrientes de línea, potencias activas y reactivas, entre otros.

Además de los reportes numéricos, OpenDSS y OpenDSS-G son capaces de generar gráficos de alta calidad que facilitan la visualización e interpretación de los resultados. Entre los gráficos más relevantes se encuentran los perfiles de tensión, que muestran la variación de los niveles de tensión a lo largo del circuito y permiten identificar áreas críticas donde las tensiones se desvían de los rangos aceptables. Además, se pueden generar mapas de calor que representan visualmente la distribución espacial de variables como tensión, corriente y potencia en los diferentes elementos del sistema, lo que facilita la detección de puntos problemáticos o desbalances en la red.

En este capítulo se presentarán algunos de los gráficos y resultados más significativos obtenidos a partir de las simulaciones realizadas en el software OpenDSS para el circuito eléctrico modelado. Estos gráficos incluirán perfiles de tensión, mapas de calor, así como otras visualizaciones relevantes que permitan evaluar el desempeño y la calidad del sistema de distribución eléctrica.

4.1. Análisis del Circuito con Cargas Estáticas

4.1.1. General

Para comenzar el análisis de las cargas estáticas, primero se revisaron las pérdidas del sistema. En la figura 4.1 se muestra un resumen generado por OpenDSS sobre las pérdidas del circuito que detalla las pérdidas en las líneas eléctricas, las cuales son de 33,9 kW. Estas pérdidas ocurren debido a la resistencia eléctrica natural de los conductores de las líneas. Además, se registran las pérdidas en los transformadores, que son 112,5 kW. Estas pérdidas provienen de los transformadores de potencia y pueden ser causadas tanto por las resistencias en el devanado como por las corrientes parásitas y las pérdidas en el núcleo. Sumando ambos componentes, las pérdidas totales del sistema son de 146,3 kW.

En este resumen también se proporciona información sobre la potencia total de carga en el sistema, que es de 960,7 kW, para calcular el porcentaje de pérdidas para el circuito completo dividiendo las pérdidas totales entre la potencia de carga total, siendo el porcentaje de pérdidas de este circuito 15.23 %.

Figura 4.1

Resumen de las Pérdidas del Circuito de Cargas Estáticas

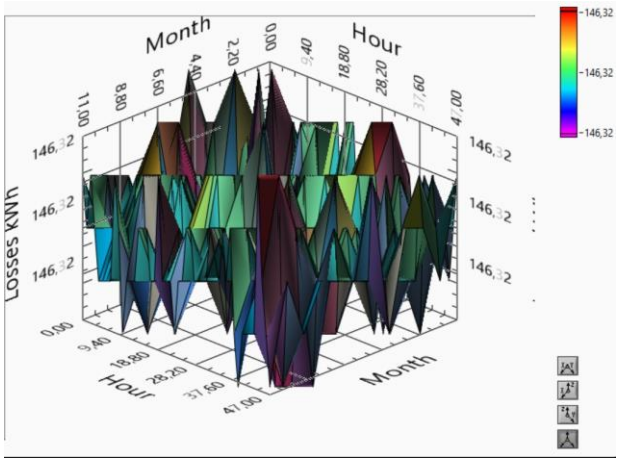
```
LINE LOSSES=                        33.9 kW
TRANSFORMER LOSSES=                112.5 kW

TOTAL LOSSES=                      146.3 kW

TOTAL LOAD POWER =                 960.7 kW
Percent Losses for Circuit =     15.23 %
```

Fuente: OpenDSS (2024).

Además, se crearon dos gráficas para ilustrar las pérdidas. En la primera (figura 4.2) se presentan las pérdidas en kilovatios-hora (kWh).

Figura 4.2

Gráfico Pérdidas kWh



Fuente: OpenDSS-G (2024).

El gráfico 3D que se muestra en la figura 4.2 representa el comportamiento de las pérdidas de potencia activa (kW) en el sistema eléctrico a lo largo del tiempo. Los ejes muestran el mes, la hora del día y los valores numéricos correspondientes de pérdidas. Se observa un patrón periódico con variaciones significativas en las pérdidas de potencia. Estas fluctuaciones exhiben picos más altos durante ciertos meses y horas específicas, lo cual está directamente relacionado con los patrones de demanda de energía eléctrica en el sistema. Las pérdidas más elevadas se producen durante los períodos de mayor consumo, mientras que las pérdidas más bajas ocurren en momentos de menor demanda. La escala de colores representa la magnitud de las pérdidas, donde los tonos cálidos, como rojos y anaranjados, indican las pérdidas más altas y los tonos fríos, como los azules y verdes, las perdidas más bajas. Esta visualización permite identificar rápidamente los períodos críticos de mayores pérdidas en el sistema.

4.1.2. Perfil de Tensión

Para iniciar el análisis del sistema, se procedió a examinar detenidamente el perfil de tensión. Este proceso se dividió en tres etapas distintas: en primer lugar, se llevó a cabo una revisión inicial antes de conectar cualquier carga al sistema. Posteriormente, se realizó una segunda evaluación una vez que todas las cargas estuvieron debidamente conectadas. Finalmente, se efectuó una última revisión después de la instalación de los reguladores de tensión, con el objetivo de determinar su impacto en el perfil eléctrico global.

En el análisis de sistemas eléctricos de potencia, las gráficas de perfil de tensión son herramientas visuales fundamentales que permiten analizar y comprender el comportamiento de la tensión a lo largo de un

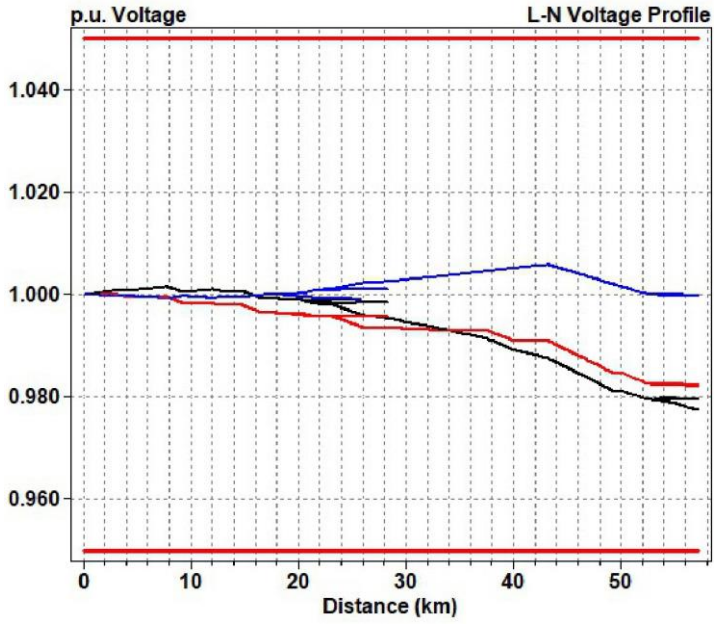
circuito. Estas gráficas representan la variación de la tensión en función de la distancia o ubicación en el circuito, proporcionando información crucial para garantizar un suministro de energía adecuado y estable. En OpenDSS, la gráfica de perfil de tensión presenta tres líneas de colores distintivos (rojo, negroy azul) y brindan información sobre la tensión en cada una de las fases del circuito trifásico.

En el gráfico mostrado en la figura 4.3 se muestra el perfil de tensión del circuito modelado cuando aún no se han conectado las cargas. Cuando las cargas no están conectadas, se puede observar que la tensión se mantiene estable y dentro de los límites establecidos, lo que indica que el sistema está funcionando correctamente sin demanda de energía adicional.

Sin embargo, cuando se conectan las cargas, como se muestra en la figura 4.4, la tensión de la fase B, es decir la línea azul, se mantiene dentro de los límites establecidos. Sin embargo, la línea roja, que representa la fase A, cae por debajo de los límites establecidos a partir del kilómetro 24 aproximadamente, lo que indica una situación más crítica que requiere atención. Esta caída de tensión puede deberse a diversos factores, como una mayor demanda de energía en esa fase o la presencia de cargas desequilibradas. Para mitigar esta situación, se implementaron reguladores de tensión, los cuales, como se observa en la figura4.5, logran aumentar los niveles de tensión en su mayoría. No obstante, en la fase A aún se registran tensiones por debajo de los límites establecidos, posiblemente debido a que es la fase con más carga del circuito.

Figura 4.3

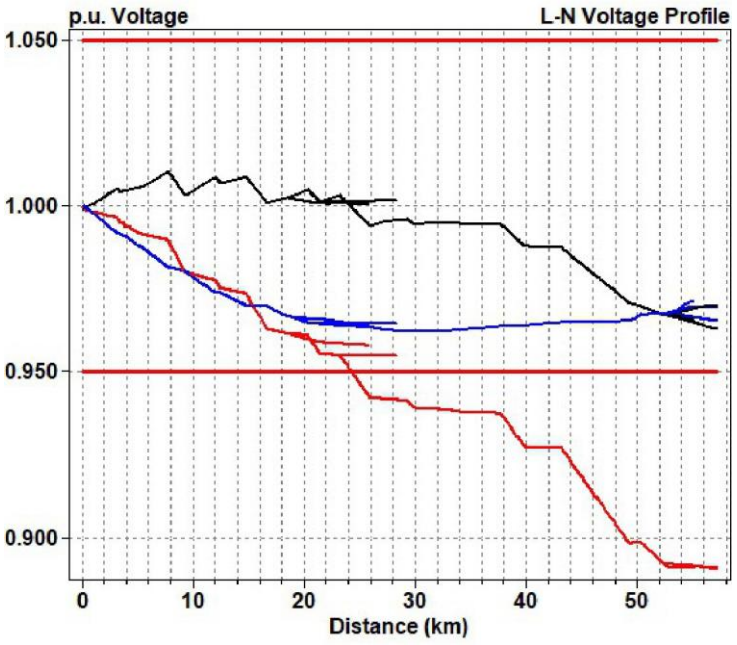
Gráfica de Perfil de Tensiones del Circuito sin Cargas



Fuente: OpenDSS-G (2014).

Figura 4.4

Gráfica de Perfil de Tensiones del Circuito con Cargas y sin Reguladores de Tensión

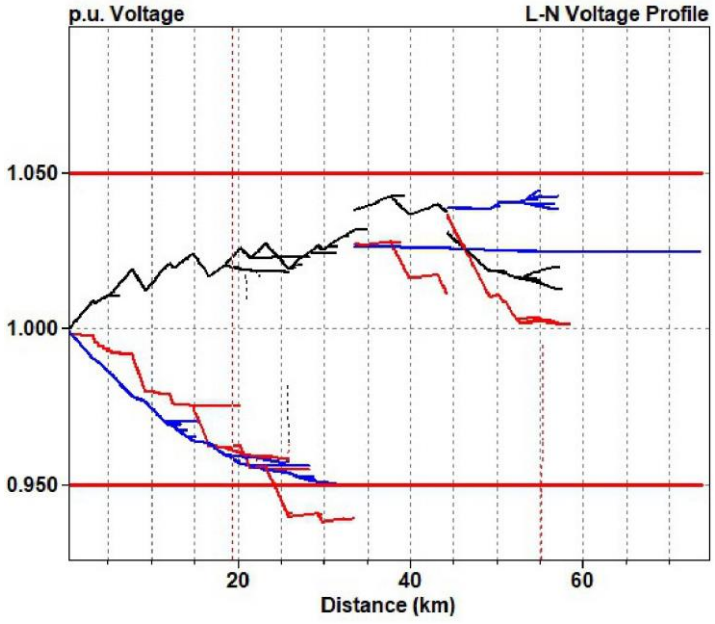


Fuente: OpenDSS-G (2024)

Figura 4.5

Gráfica de Perfil de Tensiones del Circuito con Cargas y con Reguladores de Tensión

Fuente: OpenDSS-G (2024).



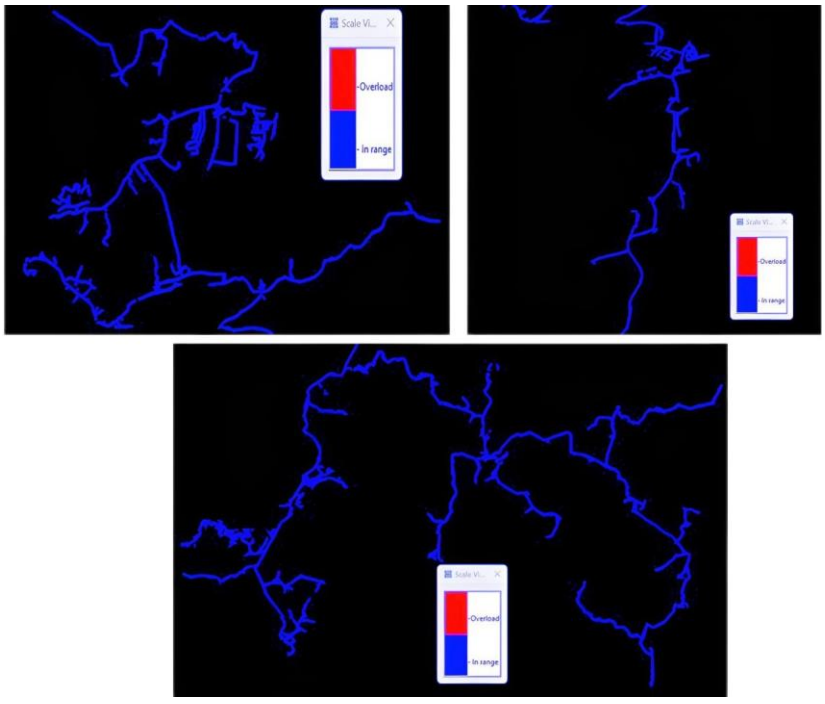
4.1.3. Mapas de Calor

Después de generar los gráficos de perfil de tensión, se procedió a realizar mapas de calor utilizando la herramienta OpenDSS. Los mapas de calor son representaciones visuales que permiten identificar de manera intuitiva las áreas o secciones del circuito eléctrico donde se presentan los mayores desafíos en términos de caídas de tensión o sobretensiones. Estos mapas tienen una representación por color de las áreas problemáticas y las que se encuentran en el área aceptable.

Para empezar, se llevó a cabo la creación de un mapa de calor, con el fin de identificar las sobrecargas del sistema. En la figura 4.6 se presenta este mapa, cuyo gráfico muestra el circuito en tres secciones más pequeñas, con el fin de tener una visualización más clara. La escala de colores en el cuadro lateral explica el significado de los colores utilizados en el mapa. El color rojo indica áreas con sobrecarga o exceso de demanda eléctrica, mientras que el azul representa áreas donde la carga se mantiene dentro del rango aceptable o normal. Es importante destacar que en ninguna parte del circuito se registra una sobrecarga, ya que no hay ninguna sección coloreada en rojo.

Figura 4.6

Mapa de Calor de Sobrecarga



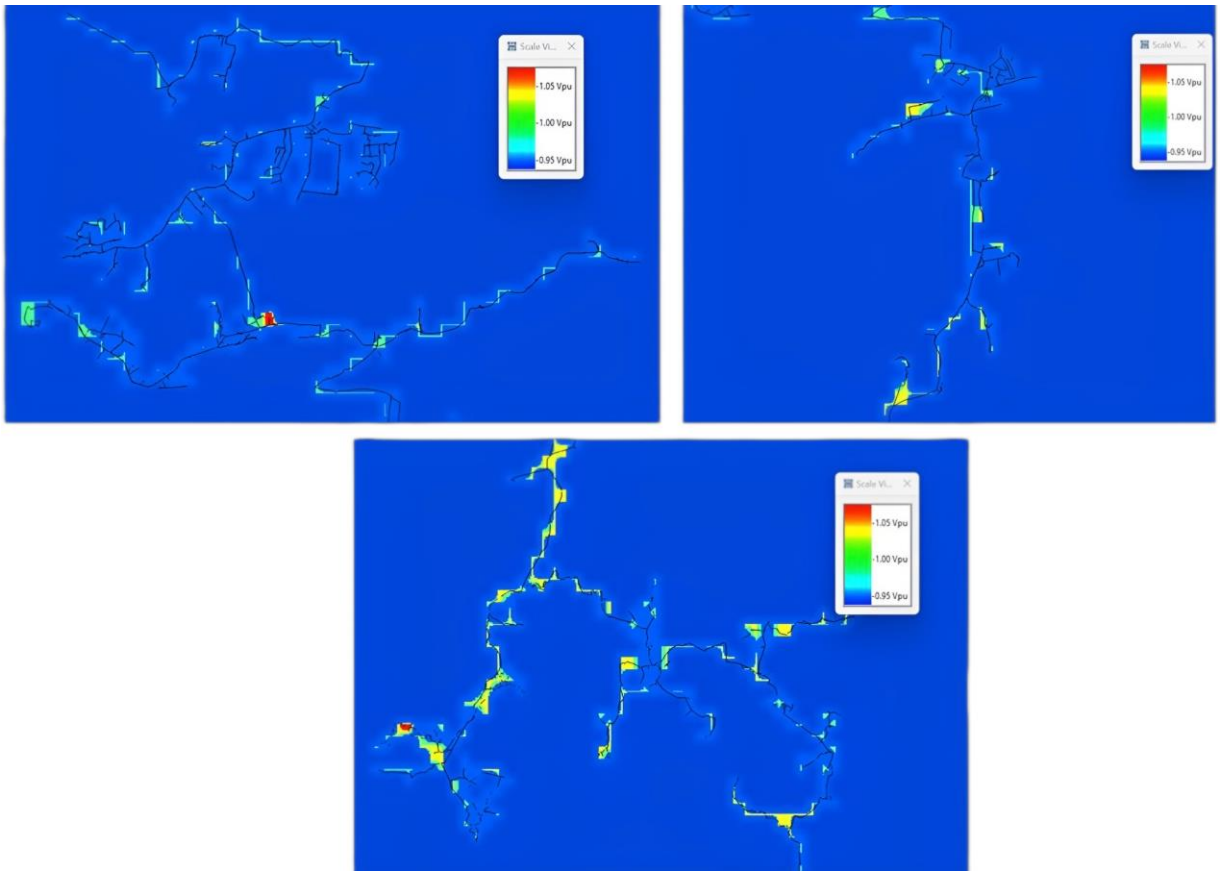
Fuente: OpenDSS-G (2024).

Luego se generó el mapa de calor de Overvoltage (sobre tensiones). Este mapa de color muestra una representación visual de los niveles de sobretensión en el sistema de distribución. Las áreas de color rojo intenso indican las ubicaciones donde se presentan las mayores sobretensiones, lo que puede ser perjudicial

para los equipos y dispositivos conectados. Por otro lado, las zonas de color azul oscuro o verde representan áreas con niveles de tensión más bajos o cercanos a los valores nominales. Este mapa de calor permite identificar rápidamente los puntos críticos del sistema y tomar medidas correctivas adecuadas. Como se puede observar en la figura 4.7, la mayor parte del sistema se encuentra en azul y verde; sin embargo, hay pequeñas secciones del sistema en rojo, lo que parece indicar que hay sobretensiones.

Figura 4.7

Mapa de Calor de Sobretensiones

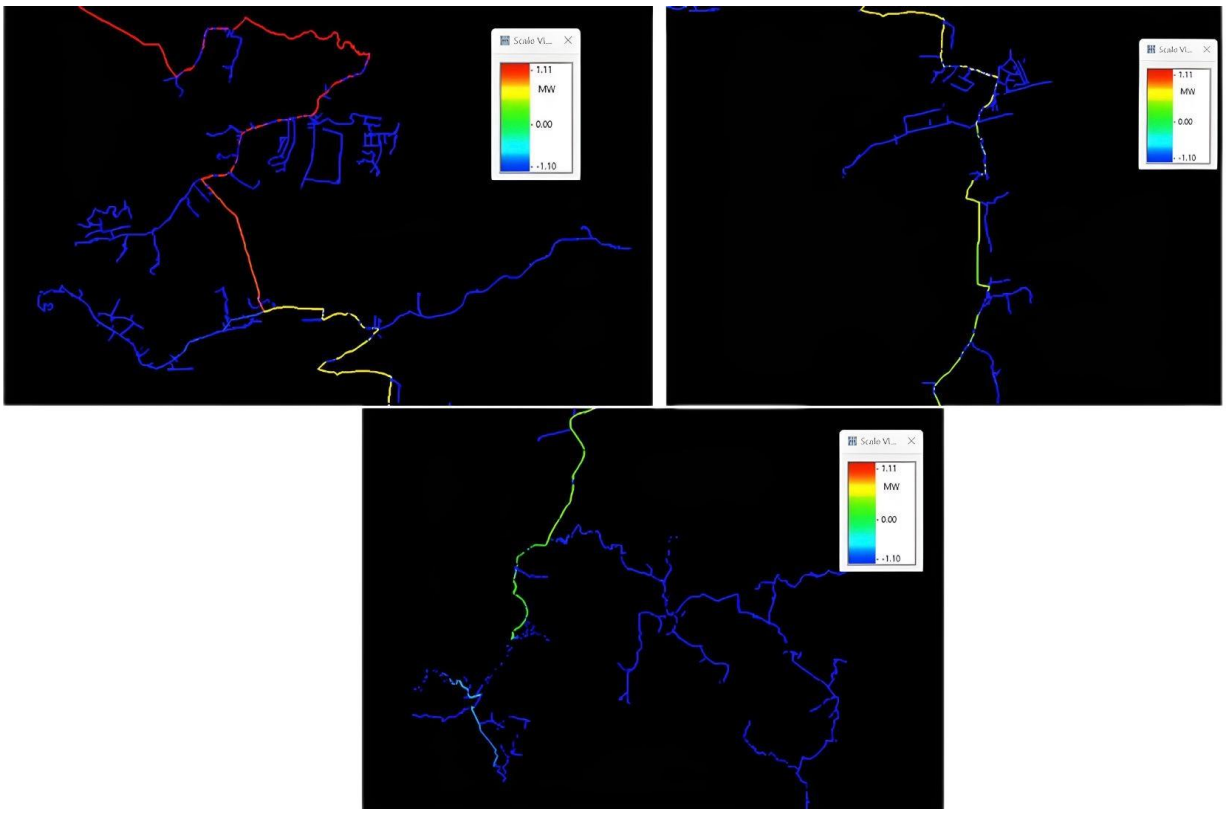


Fuente: OpenDSS-G (2024).

Después se generó el mapa de calor de PowerFlow (flujo de potencia), el cual brinda una visualización de los flujos de potencia a través del sistema de distribución. En este caso como se observa en la figura 4.8, las áreas de color amarillo o naranja intenso indican las ramas o secciones del sistema que están experimentando altos niveles de flujo de potencia, lo que puede ser un indicador de pérdidas excesivas. Por el contrario, las zonas de color azul oscuro representan ramas con flujos de potencia más bajos y el color verde indica un flujo de potencia estable.

Figura 4.8

Mapa de Calor de Flujo de Potencia



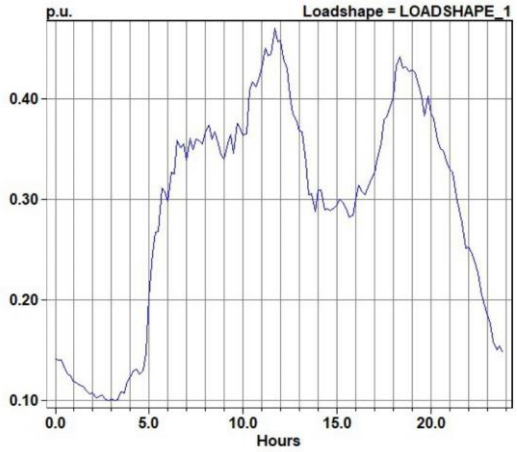
Fuente: OpenDSS-G (2024).

4.2. Análisis del Circuito con Cargas Dinámicas: LoadShape_1

En la figura 4.9 se encuentra la curva de demanda que representa el perfil de carga más bajo de los seis analizados. Se caracteriza por presentar valores de demanda relativamente bajos durante todo el día, con picos a medio día y entre las 6: 00 p.m. y 7: 00 p.m.

Figura 4.9

Curva de Demanda 1



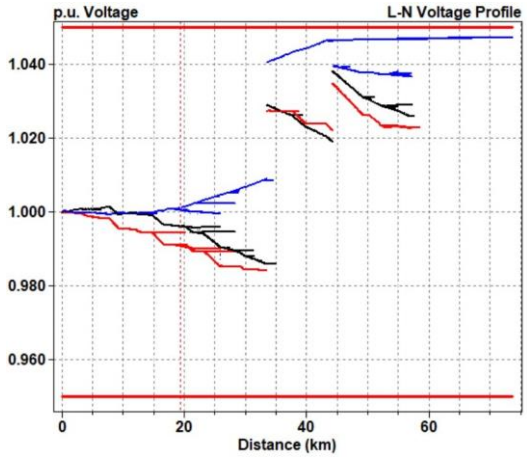
Fuente: OpenDSS-G (2024).

4.2.1. Perfil de Tensión

El perfil de tensión para el circuito con las cargas dinámicas y la curva de demanda uno, que se muestra en la figura 4.10, se mantienen dentro de los márgenes permitidos a lo largo de todo el circuito. Esto se puede deber a los niveles relativamente bajos de demanda, por lo que las caídas de tensión son mínimas.

Figura 4.10

Perfil de Tensión para las Cargas Dinámicas con la Curva de Demanda 1



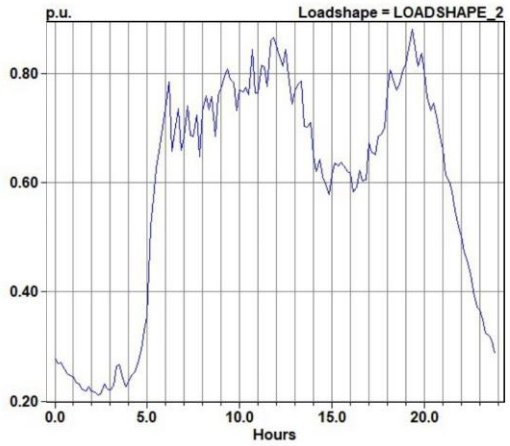
Fuente: OpenDSS-G (2024).

4.3. Análisis del Circuito con Cargas Dinámicas: LoadShape_2

La curva de demanda 2, que se muestra en la figura 4.11, exhibe un perfil de carga moderado, con valores intermedios en comparación con las demás curvas analizadas. Los picos de demanda se dan al medio día y alrededor de las 7: 00 p.m. para después disminuir.

Figura 4.11

Curva de Demanda 2



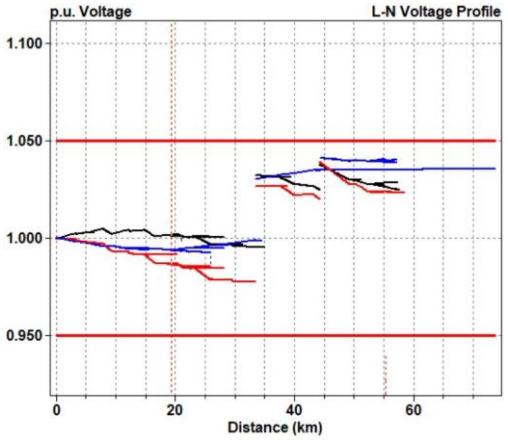
Fuente: OpenDSS-G (2024).

4.3.1. Perfil de Tensión

Al igual que con la curva de demanda uno, el perfil de tensión para el circuito con las cargas dinámicas y la curva de demanda dos, representada en la figura 4.12, se mantiene dentro de los márgenes permitidos a lo largo de todo el circuito. Es probable que este comportamiento se deba a factores similares que incluyen niveles moderados de demanda, mínimas caídas de tensión y un equilibrio adecuado entre las tres fases del sistema.

Figura 4.12

Perfil de Tensión para las Cargas Dinámicas con la Curva de Demanda 2



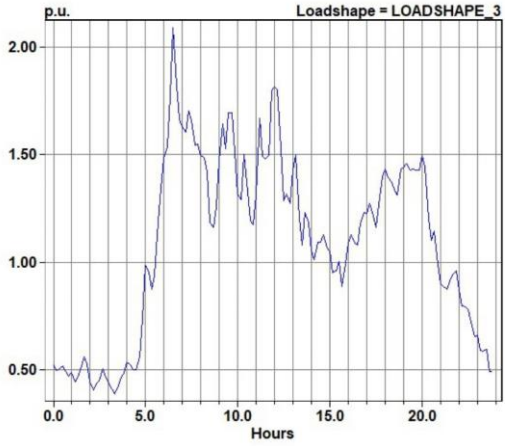
Fuente: OpenDSS-G (2024).

4.4. Análisis del Circuito con Cargas Dinámicas: LoadShape_3

La curva de demanda 3, que se muestra en la figura 4.13, exhibe niveles intermedios de consumo. Se distingue un aumento notable de la demanda a partir de las 5: 00 a.m., llegando a un pico pronunciado entre las 6: 00 a.m. y 7: 00 a.m., para luego ir descendiendo gradualmente a medida que avanza el día.

Figura 4.13

Curva de Demanda 3



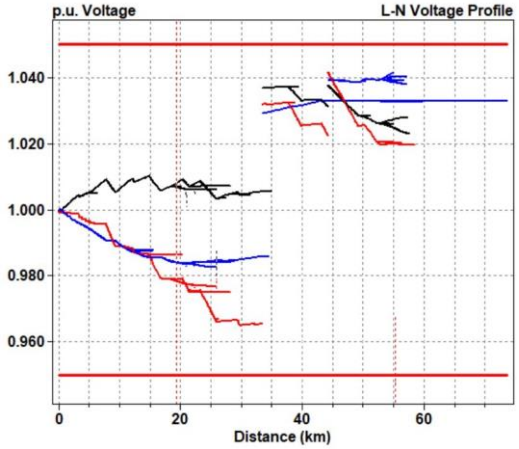
Fuente: OpenDSS-G (2024).

4.4.1. Perfil de Tensión

En el caso de las cargas dinámicas con la curva de demanda tres, el perfil de tensión, representado en la figura 4.14, se mantiene dentro de los márgenes permitidos a lo largo de todo el circuito. No obstante, se observa una ligera caída de tensión en las fases A y B, posiblemente por el aumento en la demanda eléctrica.

Figura 4.14

Perfil de Tensión para las Cargas Dinámicas con la Curva de Demanda 3



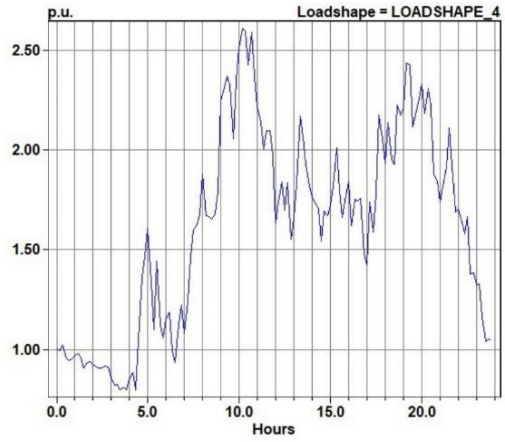
Fuente: OpenDSS-G (2024).

4.5. Análisis del Circuito con Cargas Dinámicas: LoadShape_4

En la figura 4.15 se observa la curva de demanda 4. Esta curva presenta niveles de demanda más altos en comparación con las anteriores. Se aprecia un incremento significativo en el consumo desde las primeras horas del día, alcanzando su pico máximo al rededor del mediodía y manteniéndose en niveles elevados durante gran parte de la curva.

Figura 4.15

Curva de Demanda 4



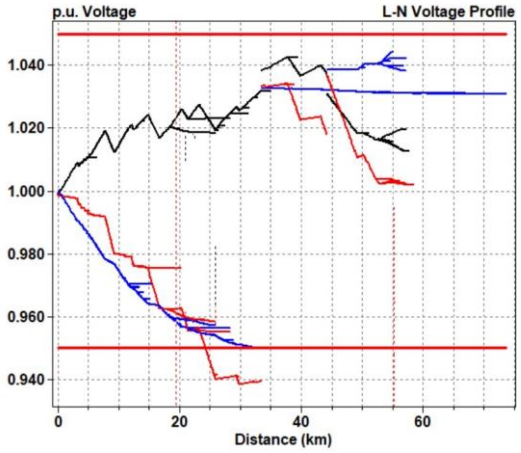
Fuente: OpenDSS-G (2024).

4.5.1. Perfil de Tensión

En el caso de las cargas dinámicas con la curva de demanda cuatro, se observa en el perfil de tensión representado en la figura 4.16, que la tensión en la fase B experimenta una caída considerable, aunque se mantiene al límite permitido. Por otro lado, en la fase A, la caída es aún más pronunciada, quedando por debajo de los niveles aceptables. Estos resultados evidencian claramente cómo el aumento de la carga afecta negativamente al circuito, comprometiendo su estabilidad y eficiencia.

Figura 4.16

Perfil de Tensión para las Cargas Dinámicas con la Curva de Demanda 4



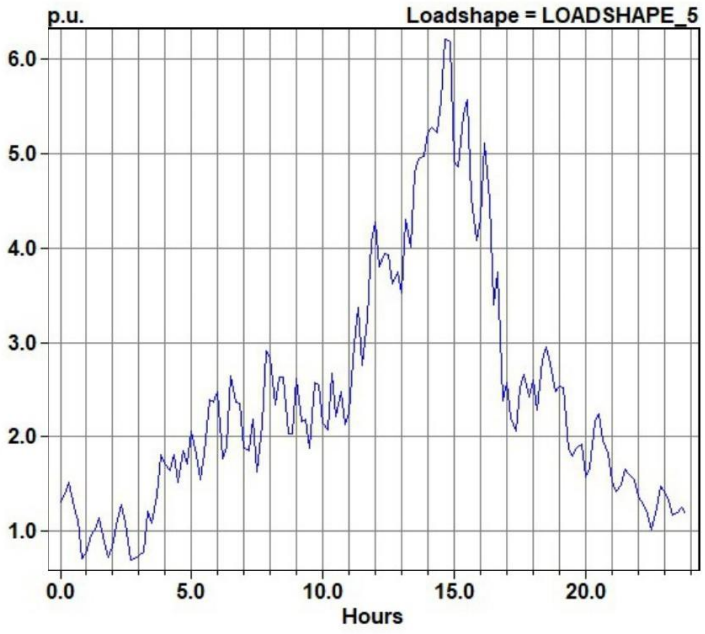
Fuente: OpenDSS-G (2024).

4.6. Análisis del Circuito con Cargas Dinámicas: LoadShape_5

La curva de demanda 5 muestra uno de los perfiles de carga más altos. Se distingue un pico de demanda muy pronunciado alrededor de las 3: 00 p.m., seguido de varios picos secundarios durante las horas posteriores. A lo largo de toda la curva la demanda se mantiene alta.

Figura 4.17

Curva de Demanda 5



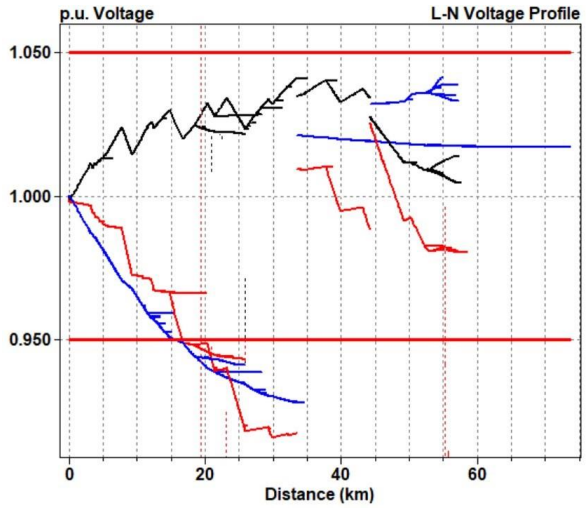
Fuente: OpenDSS-G (2024).

4.6.1. Perfil de Tensión

Para las cargas con la curva de demanda 5, en el perfil de tensión del circuito que se muestra en la figura 4.18 se observa que las tensiones de las fases A y B caen por debajo de los límites permitidos, mientras que la fase C se mantiene constante. Esta información revela que, ante niveles de demanda alta, las fases del circuito se desequilibran significativamente. Por lo que, para garantizar un funcionamiento óptimo y seguro del circuito, es crucial corregir este desequilibrio mediante métodos como la redistribución de la carga.

Figura 4.18

Perfil de Tensión para las Cargas Dinámicas con la Curva de Demanda 5



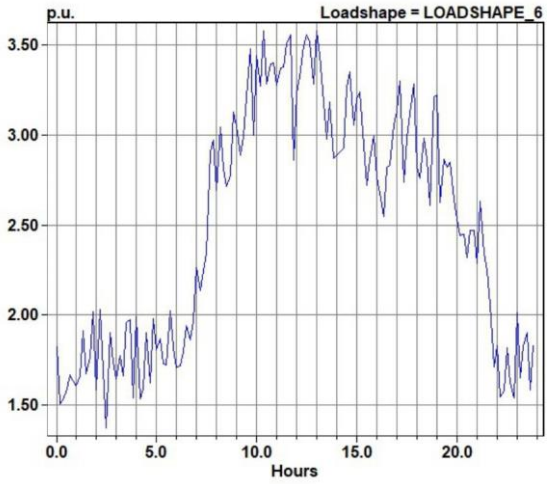
Fuente: OpenDSS-G (2024).

4.7. Análisis del Circuito con Cargas Dinámicas: LoadShape_6

Esta es la curva con los niveles de demanda más altos de todas. Se caracteriza por mantener valores de consumo extremadamente elevados durante la mayor parte del día, con múltiples picos muy pronunciados.

Figura 4.19

Curva de Demanda 6



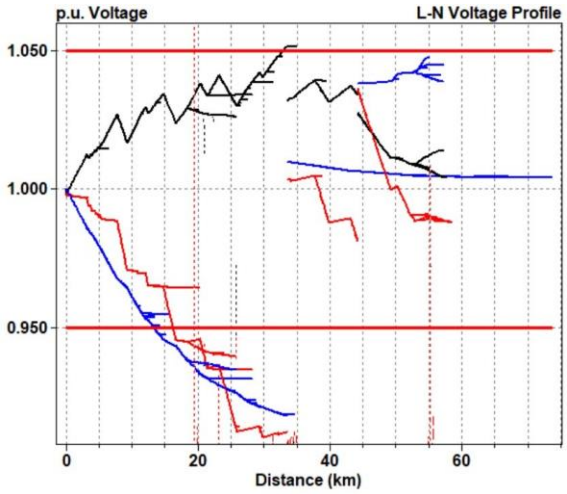
Fuente: OpenDSS-G (2024).

4.7.1. Perfil de Tensión

Para las cargas con la curva de demanda 6, el perfil de tensión del circuito, representado en la figura 4.20, muestra una situación interesante: mientras que las tensiones de las fases A y B caen por debajo de los límites permitidos, la fase C experimenta un aumento en la tensión. Por lo que se puede confirmar que este circuito al enfrentarse a situaciones de carga más altas, se debe modificar para distribuir de mejor manera la carga entre las fases del sistema eléctrico, disminuyendo la carga en las fases A y B y aumentando en la fase C.

Figura 4.20

Perfil de Tensión para las Cargas Dinámicas con la Curva de Demanda 6



Fuente: OpenDSS-G 8 (2024).

Capítulo 5

Conclusiones y Recomendaciones

5.1. Conclusiones

Con respecto del objetivo 1:

1. El desarrollo de programas en Python para la limpieza y transformación de datos agilizó significativamente el proceso de preparación de datos para la simulación eléctrica, optimizando el proceso de transferencia de información.
2. La combinación de datos geográficos y eléctricos proporcionados por el ICE permitió una integración completa en el modelado del circuito eléctrico, garantizando la compatibilidad y utilidad de los datos para análisis y simulaciones futuras.
3. La minería de datos realizada durante la fase de recolección y depuración garantizó la integridad y consistencia de la información recopilada.

Con respecto del objetivo 2:

4. La implementación de programas en Python para la generación de archivos DSS automatizó la definición de los elementos del circuito eléctrico, demostrando un enfoque eficiente para la preparaciónde datos para simulaciones.
5. Desde la definición de buses y líneas hasta la configuración de transformadores y cargas, se logró una completa incorporación de todos los elementos del circuito eléctrico, garantizando la representación precisa del sistema para análisis y simulaciones futuras.

Con respecto del objetivo 3:

6. A pesar de la falta de información en algunos archivos proporcionados, se superaron las limitaciones mediante el cálculo de impedancias faltantes y la creación de características de líneas secundarias, asegurando la completitud del modelo del circuito eléctrico.

7. La colaboración estrecha con el ICE facilitó el acceso a los datos necesarios y proporcionó la orientación adecuada para abordar los desafíos específicos del modelado del circuito eléctrico, destacando la importancia de las relaciones de trabajo colaborativas en proyectos de esta naturaleza.

Con respecto del objetivo 4:

8. La validación de los datos transformados con OpenDSS mediante pruebas de integración garantizó la compatibilidad y precisión del modelo del circuito eléctrico, asegurando la utilidad de los resultados para futuros análisis y simulaciones.

Adicionalmente:

9. Durante el desarrollo del proyecto, se enfrentaron desafíos significativos con el uso operativo y la interacción general con el software utilizado para llevar a cabo las tareas de simulación y análisis de resultados en OpenDSS y OpenDSS-G. Estos desafíos se derivaron principalmente de la rigidez de dichos programas, lo que dificultó incluso las tareas más simples, como realizar zoom para examinar o modificar elementos específicos del circuito, por lo que tareas como la selección de elementos en la red resultó ser una tarea costosa en términos de tiempo y esfuerzo. En contraste, existen programas alternativos como ETAP y CYME ofrecen una experiencia de usuario más fluida y proporcionan capacidades adicionales para el análisis. Esto hace que el modelado de la red sea más eficiente y eficaz en comparación con las limitaciones encontradas en OpenDSS y OpenDSS-G.

5.2. Recomendaciones

A continuación, se presenta una serie de recomendaciones para proyectos futuros:

Con respecto del objetivo 1:

1. Implementar procedimientos de validación para garantizar la integridad y precisión de los datos recopilados, así como de los resultados obtenidos en las simulaciones eléctricas.

2. Registrar detalladamente los pasos realizados durante la preparación de datos y el modelado del circuito, incluyendo cualquier decisión tomada y los criterios utilizados.

3. Establecer canales de comunicación efectivos con la empresa distribuidora y otros colaboradores para asegurar la claridad en los requisitos de datos y en la interpretación de los resultados.

Con respecto del objetivo 2:

4. Aplicar buenas prácticas de programación al desarrollar programas en Python, como la modularidad, el uso de comentarios descriptivos y la validación de entrada de datos.
5. Ejecutar pruebas exhaustivas en los programas desarrollados para la preparación de datos y la generación de archivos DSS, asegurando su funcionamiento correcto y la coherencia de los resultados.

Con respecto del objetivo 3:

6. Investigar y aprovechar herramientas y bibliotecas de Python que faciliten la automatización de tareas repetitivas o de procesamiento de datos a gran escala.
7. Priorizar la seguridad y la privacidad de los datos en todas las etapas del proyecto, asegurando el cumplimiento de las regulaciones y políticas pertinentes.

Con respecto del objetivo 4:

10. Estar abierto a la retroalimentación y buscar continuamente oportunidades de mejora en los procesos y herramientas utilizados para el modelado del circuito eléctrico, con el objetivo de optimizar la eficiencia y la precisión de los resultados.
11. Realizar una actualización periódica de los datos geográficos y eléctricos en el modelo, para asegurar que este refleje los cambios y expansiones en la red de distribución. Esto permitirá mantener la precisión y utilidad del modelo a largo plazo.
12. Se recomienda comparar los resultados obtenidos mediante OpenDSS, con datos de campo para validar la precisión y confiabilidad del modelo del circuito.

Adicionalmente

13. Exportar el circuito a un software especifico de Sistema de Información Geográfico, como ArcGIS, para optimizar la visualización y el análisis geoespacial de la red eléctrica. Esto facilitará las actualizaciones y la toma de decisiones.

Bibliografía

ARESEP (2023). Estudios de capacidad de alojamiento de der en circuitos de distribución.

Chen, C., Wang, J. and Ton, D. (2017). Modernizing distribution system restoration to achieve grid resiliency against extreme weather events: An integrated solution. *Proceedings of the IEEE*, 105(7):1267–1288.

Contraloría General de la República (2019). Informe de la auditoría operativa coordinada sobre energías renovables en el sector eléctrico. Technical report, Costa Rica.

Derakhshan, G., Milani, K. R., Etemadi, A., Shayanfar, H. and Sarafraz, U. (2013). Management and operation of electricity distribution networks on geographic information system platform. pages 1–4.

Dugan, R. C. (2013). *Reference Guide: The Open Distribution System Simulator (OpenDSS)*. Electric Power Research Institute, Inc., Sr. Technical Executive.

EPERLab. (2017). Biblioteca de parámetros de transformadores.

Hernández, M. F. (2018). *OpenDSS Viewer User Manual*. Electric Power Research Institute, Inc.

Gallego, L. A., López Lezama, J. M. and Mejía Giraldo, D. A. (2009). Flujo de potencia trifásico desbalanceado en sistemas de distribución con generación distribuida. *Scientia et Technica*, XV(43):23–38.

Gallego Rendón, R. A., Escobar Zuluaga, A. H. and Granada Echeverri, M. (2016). *Flujo de carga en sistemas de transmisión: modelamiento y análisis*. Colección Textos Académicos. Editorial Universidad Tecnológica de Pereira, Pereira.

Geo Ingeniería Ingenieros Consultores S.A. (2024). *Análisis Comparativo del Marco Regulatorio, Incentivos y Sistema Tarifario de Precios Existentes, para la compra/generación de Electricidad de plantas de Energía Renovable en Centroamérica y Panamá*. Banco Centroamericano de Integración Económica, Apartado Postal 772, Tegucigalpa, M.D.C., Honduras, C.A.

GeoPandas.org. (2013). Geopandas. https://geopandas.org/en/stable/. [Último acceso: 10 octubre 2023].

Glover, J. D., Overbye, T. J. and Sarma, M. S. (2015). *Power System Analysis & Design*. Cengage Learning, Boston, MA, sixth edition.

Gómez-Ramírez, G. A. (2016). Evolución y tendencias de índices de confiabilidad en sistemas eléctricos de potencia. *Revista Tecnología en Marcha*, 29(2):3–13.

Gómez-Ramírez, G. A., Luévano-Reyes, I. A., Meza, C. and García-Santander, L. (2021). Demand response improvement using storage power systems: Case study of honduras. In *2021 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6.

Gómez-Ramírez, G. A., Luévano-Reyes, I. A., Mora-Jiménez, G., García-Santander, L., Laskano, M. Z. and Meza, C. (2022). Increasing distribution network capacity through storage in central american countries: A case study. In *2022 IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, pages 1–6.

Gómez-Ramírez, G. A., Meza, C., Mora-Jiménez, G., Morales, J. R. R. and García-Santander, L. (2023a). The central american power system: Achievements, challenges, and opportunities for a green transition. *Energies*, 16(11).

Gómez-Ramírez, G. A., Mora-Jiménez, G., and Meza, C. (2023b). Simulación del sistema de interconexión eléctrica de los países de américa central usando etap. *Revista Tecnología en Marcha*, 36(2):Pág. 50–58.

Hay, S. and Ferguson, A. (2015). A review of power system modelling platforms and capabilities. *The Institution of Engineering and Technology*.

Hormazábal Sanhueza, R. and Ramírez Estrada, M. (2014). Integración de un sistema de información geográfica en la planificación y gestión de los sistemas de distribución eléctrica. *Ingeniare. Revista chilena de ingeniería*, 22(1):6–13.

IEEE. (2000). Ieee standard general requirements for liquid-immersed distribution, power, and regulating transformers. Standard IEEE Std C57.12.00-2000, IEEE.

IEEE. Power Engineering Society (1988). Ieee standard requirements, terminology, and test code for step-voltage and induction-voltage regulators. Standard ANSI/IEEE Std C57.15-1986, IEEE.

Instituto Costarricense de Electricidad. (2021). Quiénes somos: historia.

Instituto Costarricense de Electricidad. (2023). Principios corporativos.

Instituto Costarricense de Electricidad, División Transmisión. (2021). Plan de expansión de la transmisión 2021-2031. Technical report.

Instituto Costarricense de Electricidad (ICE). (2023). Informe de atención de demanda y producción de electricidad con fuentes renovables - Costa Rica 2023. Informe técnico, División Operación y Control del Sistema Eléctrico, Instituto Costarricense de Electricidad, Costa Rica.

Kersting, W. H. (2001). *Distribution System Modeling and Analysis*. Electric Power Engineering Series. CRC Press.

La Gaceta. (2023). Alcance N°174 a La Gaceta N°169.

Lumbreras, S. and Ramos, A. (2016). The new challenges to transmission expansion planning. Survey ofrecent practice and literature review. *Electric Power Systems Research*, 134:19–29.

Matulic, I. (2003). Introducción a los sistemas eléctricos de potencia. *RevActaNova*, 02(02):208–215.

Montenegro, D. and Dugan, R. (2020). *Program on Technology Innovation: OpenDSS-G - A Graphical View of OpenDSS*. 3420 Hillview Avenue, Palo Alto, California 94304-1338.

Moreno Muñoz, A. and Cordobes Córcoles, S. (2019). *Python Práctico*. RAMA EDITORIAL, Madrid.

Nayeripour, M., Rezaee, N., Roosta, A. and Niknam, T. (2010). Role of gis in distribution power systems. *World Applied Sciences Journal*, 08(05):614–621.

Ortega Candel, J. M. (2022). *Big data, machine learning y data science en Python*. RA-MA S.A. Editorial y Publicaciones, Madrid.

PyPI. (2024). Shapely. https://pypi.org/project/shapely/.

Resolución RJD-070-2015. (2015). *Supervisión de la calidad del suministro eléctrico en baja y media tensión (AR-NT-SUCAL)*.

Santos Preciado, J. M. (2020). *Sistemas de información geográfica*. UNED, San José.

Schavemaker, P. and Van Der Sluis, L. (2017). *Electrical Power System Essentials*. John Wiley & Sons, Inc., Chichester, West Sussex, 2nd edition.

Short, T. (2004). *Electric Power Distribution Handbook*. CRC Press LLC.

Thue, W. A. (2017). *Ingeniería de cables de energía eléctrica*. CRC Press, 1st edition.

Viganò, G., Clerici, D., Michelangeli, C., Moneta, D., Bosisio, A., Morotti, A., Greco, B. and Caterina, P. (2021). Using gis to assess the impact of electric vehicles on electrical distribution networks: a study applied to the city of brescia. pages 1–6.

Anexo A

Configuraciones de conductores eléctricos ICE

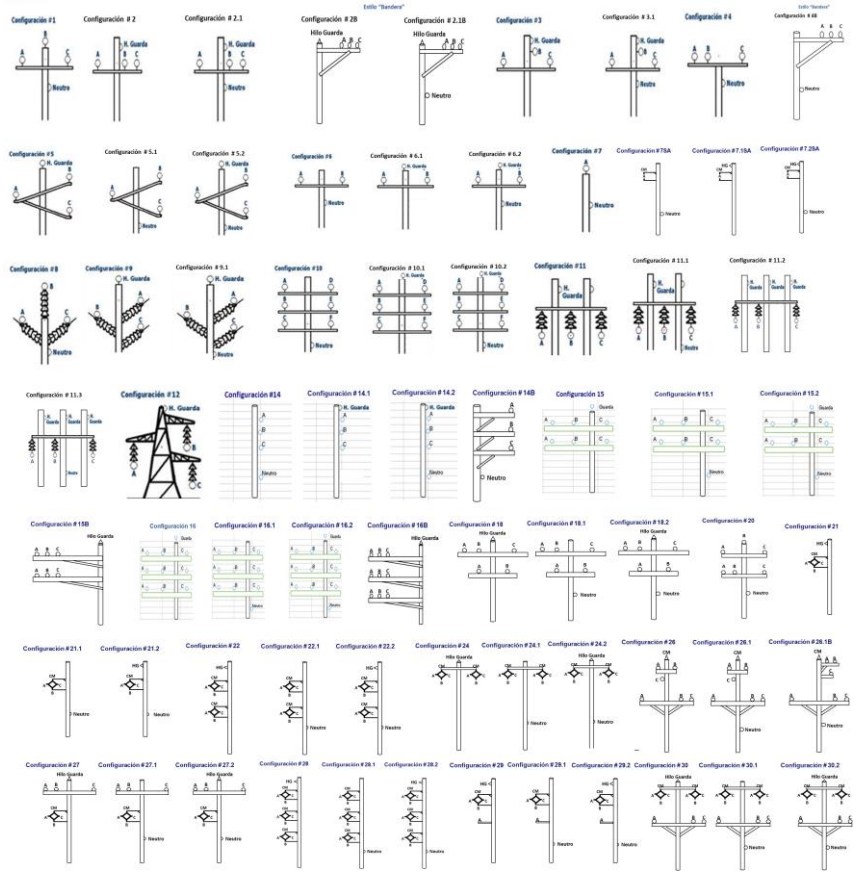


Figura A.1: Configuraciones de líneas aéreas
Fuente: SIRDE.

Anexo B

Wiredata AAAC

New WireData.AAAC_6_7STR Rac=0.785 Runits=kft GMR=0.0060 GMRunits=ft DIAM=0.198 Radunits=in Normamps=107

New WireData.AAAC_4_7STR Rac=0.4928 Runits=kft GMR=0.0076 GMRunits=ft DIAM=0.250 Radunits=in Normamps=143

New WireData.AAAC_2_7STR Rac=0.3098 Runits=kft GMR=0.0095 GMRunits=ft DIAM=0.316 Radunits=in Normamps=191

New WireData.AAAC_1/0_7STR Rac=0.1947 Runits=kft GMR=0.0120 GMRunits=ft DIAM=0.398 Radunits=in Normamps=256

New WireData.AAAC_2/0_7STR Rac=0.1545 Runits=kft GMR=0.0135 GMRunits=ft DIAM=0.447 Radunits=in Normamps=296

New WireData.AAAC_3/0_7STR Rac=0.1227 Runits=kft GMR=0.0152 GMRunits=ft DIAM=0.502 Radunits=in Normamps=342

New WireData.AAAC_4/0_7STR Rac=0.0973 Runits=kft GMR=0.0170 GMRunits=ft DIAM=0.563 Radunits=in Normamps=395

New WireData.AAAC_266.8_19STR Rac=0.0769 Runits=kft GMR=0.0203 GMRunits=ft DIAM=0.642 Radunits=in Normamps=460

New WireData.AAAC_336.4_19STR Rac=0.0610 Runits=kft GMR=0.0227 GMRunits=ft DIAM=0.720 Radunits=in Normamps=532

New WireData.AAAC_397.5_19STR Rac=0.0518 Runits=kft GMR=0.0247 GMRunits=ft DIAM=0.783 Radunits=in Normamps=590

New WireData.AAAC_477_19STR Rac=0.0431 Runits=kft GMR=0.0271 GMRunits=ft DIAM=0.858 Radunits=in Normamps=663

New WireData.AAAC_556.5_19STR Rac=0.0371 Runits=kft GMR=0.0292 GMRunits=ft DIAM=0.927 Radunits=in Normamps=729

New WireData.AAAC_123.3_7STR Rac=0.195 Runits=kft GMR=0.0398 GMRunits=ft DIAM=0.398 Radunits=in Normamps=256

Anexo C

Wiredata AAC

New Wiredata.AAC_6_7STR GMR=0.00555 DIAM=0.184 Rac=0.805 Normamps=103 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_4_7STR GMR=0.00700 DIAM=0.232 Rac=0.506 Normamps=138 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_2_7STR GMR=0.00883 DIAM=0.292 Rac=0.318 Normamps=185 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_1_7STR GMR=0.0099 DIAM=0.328 Rac=0.252 Normamps=214 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_1/0_7STR GMR=0.0111 DIAM=0.368 Rac=0.2000 Normamps=247 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_2/0_7STR GMR=0.0125 DIAM=0.414 Rac=0.1590 Normamps=286 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_3/0_7STR GMR=0.0140 DIAM=0.464 Rac=0.1260 Normamps=331 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_4/0_7STR GMR=0.0158 DIAM=0.522 Rac=0.0999 Normamps=383 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_250_7STR GMR=0.0171 DIAM=0.567 Rac=0.0846 Normamps=425 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_250_19STR GMR=0.0181 DIAM=0.574 Rac=0.0846 Normamps=426 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_266.8_7STR GMR=0.0177 DIAM=0.586 Rac=0.0793 Normamps=443 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_266.8_19STR GMR=0.0187 DIAM=0.592 Rac=0.0793 Normamps=444 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_300_19STR GMR=0.0198 DIAM=0.628 Rac=0.0706 Normamps=478 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_336.4_19STR GMR=0.0210 DIAM=0.665 Rac=0.0630 Normamps=513 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_350_19STR GMR=0.0214 DIAM=0.679 Rac=0.0605 Normamps=526 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_397.5_19STR GMR=0.0228 DIAM=0.723 Rac=0.0534 Normamps=570 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_450_19STR GMR=0.0228 DIAM=0.769 Rac=0.0472 Normamps=616 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_477_19STR GMR=0.0250 DIAM=0.792 Rac=0.0445 Normamps=639 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_477_37STR GMR=0.0254 DIAM=0.795 Rac=0.0445 Normamps=639 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_500_19STR GMR=0.0256 DIAM=0.811 Rac=0.0425 Normamps=658 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_500_37STR GMR=0.0260 DIAM=0.814 Rac=0.0425 Normamps=658 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_556.5_19STR GMR=0.0270 DIAM=0.856 Rac=0.0382 Normamps=703 Runits=kft Radunits=in GMRunits=ft

New Wiredata.AAC_556.5_37STR GMR=0.0275 DIAM=0.858 Rac=0.0382 Normamps=704 Runits=kft Radunits=in GMRunits=ft

Anexo D

WireData ACSR

New WireData.ACSR_6_6STR Rac=0.806 Runits=kft GMR=0.0064 GMRunits=ft DIAM=0.198 Radunits=in Normamps=105

New WireData.ACSR_4_6STR Rac=0.515 Runits=kft GMR=0.0080 GMRunits=ft DIAM=0.250 Radunits=in Normamps=140

New WireData.ACSR_4_7STR Rac=0.519 Runits=kft GMR=0.0085 GMRunits=ft DIAM=0.257 Radunits=in Normamps=140

New WireData.ACSR_2_6STR Rac=0.332 Runits=kft GMR=0.0101 GMRunits=ft DIAM=0.316 Radunits=in Normamps=184

New WireData.ACSR_2_7STR Rac=0.338 Runits=kft GMR=0.0107 GMRunits=ft DIAM=0.325 Radunits=in Normamps=184

New WireData.ACSR_1_6STR Rac=0.268 Runits=kft GMR=0.0113 GMRunits=ft DIAM=0.354 Radunits=in Normamps=212

New WireData.ACSR_1/0_6STR Rac=0.217 Runits=kft GMR=0.0127 GMRunits=ft DIAM=0.398 Radunits=in Normamps=242

New WireData.ACSR_2/0_6STR Rac=0.176 Runits=kft GMR=0.0143 GMRunits=ft DIAM=0.447 Radunits=in Normamps=276

New WireData.ACSR_3/0_6STR Rac=0.144 Runits=kft GMR=0.0161 GMRunits=ft DIAM=0.502 Radunits=in Normamps=315

New WireData.ACSR_4/0_6STR Rac=0.119 Runits=kft GMR=0.0180 GMRunits=ft DIAM=0.563 Radunits=in Normamps=357

Anexo E

WireData Cobre

New WireData.CU_14_SLD Rdc=2.626 Runits=kft GMR=0.00208 GMRunits=ft DIAM=0.0641 Radunits=in Normamps=20

New WireData.CU_8_SLD Rdc=0.653 Runits=kft GMR=0.00416 GMRUnits=ft DIAM=0.1285 Radunits=in Normamps=95

New WireData.CU_6_SLD Rdc=0.411 Runits=kft GMR=0.00526 GMRunits=ft DIAM=0.1620 Radunits=in Normamps=125

New WireData.CU_4_SLD Rdc=0.258 Runits=kft GMR=0.00663 GMRunits=ft DIAM=0.2043 Radunits=in Normamps=170

New WireData.CU_2_SLD Rdc=0.163 Runits=kft GMR=0.00836 GMRunits=ft DIAM=0.2576 Radunits=in Normamps=225

New WireData.CU_2_7STR Rdc=0.1660 Runits=kft GMR=0.00883 GMRunits=ft DIAM=0.2920 Radunits=in Normamps=230

New WireData.CU_1/0_7STR Rdc=0.1042 Runits=kft GMR=0.01113 GMRunits=ft DIAM=0.3680 Radunits=in Normamps=310

New WireData.CU_1/0_19STR Rdc=0.1040 Runits=kft GMR=0.01178 GMRunits=ft DIAM=0.3730 Radunits=in Normamps=310

New WireData.CU_2/0_7STR Rdc=0.08267 Runits=kft GMR=0.01252 GMRunits=ft DIAM=0.4140 Radunits=in Normamps=355

New WireData.CU_3/0_7STR Rdc=0.06556 Runits=kft GMR=0.01404 GMRunits=ft DIAM=0.4640 Radunits=in Normamps=410

New WireData.CU_4/0_19STR Rdc=0.05199 Runits=kft GMR=0.01668 GMRunits=ft DIAM=0.5280 Radunits=in Normamps=480

New WireData.CU_250_19STR Rdc=0.04400 Runits=kft GMR=0.01813 GMRunits=ft DIAM=0.5740 Radunits=in Normamps=530

New WireData.CU_350_19STR Rdc=0.03143 Runits=kft GMR=0.02140 GMRunits=ft DIAM=0.6790 Radunits=in Normamps=650

New WireData.CU_500_37STR Rdc=0.02200 Runits=kft GMR=0.02605 GMRunits=ft DIAM=0.8140 Radunits=in Normamps=810

Apéndice A

Código de python

GeneradorArchivoDSS.py

```
'''
Autor : Keitlyn Valeria Montiel Arce
Fecha : 24/01/2024
Generador de archivos DSS
'''
# Importar bibliotecas
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point
import subprocess
import csv

# Ruta del archivo Excel
file_path = r'C:\Users\nunz1\OneDrive\Documentos\Universidad\2024\TFG

\Circuito\Circuito_5.xlsx '

"Creacion del archivo de buses"

# Leer el archivo Excel en un DataFrame
Busbar_df = pd.read_excel(file_path, 'Busbar')

# Crear un GeoDataFrame con las coordenadas originales en epsg:5367
geometry_orig = [Point(x, y) for x,y in
zip(Busbar_df['CoordX1'], Busbar_df['CoordY1'])]
gdf_orig = gpd.GeoDataFrame(Busbar_df, geometry=geometry_orig, crs='epsg:5367')

# Convertir las coordenadas a epsg:4326
gdf_converted = gdf_orig.to_crs('epsg:4326')

# Crear un nuevo DataFrame con las columnas requeridas
output_df = pd.DataFrame({
    'Name': Busbar_df['Name'],
    'X': gdf_converted['geometry'].x,
    'Y': gdf_converted['geometry'].y,
    })

# Guardar el DataFrame en un archivo de texto con extension .dss
BusCoords = 'BusCoords.dss'

# Abrir el archivo en modo escritura ('w')
with open(BusCoords, 'w') as archivo:
    archivo.write(output_df.to_csv(sep=',', index=False,
    header=False).replace(',', ' '))

# Abrir automaticamente el archivo .dss en notas
subprocess.run(['notepad.exe', BusCoords])
```

Figura A.1: Sección de definición de buses en el archivo GeneradorArchivoDSS
Fuente: Elaboración propia.

```
"Creacion de los archivos de lineas de Media Tension"

# Leer el archivo Excel en un DataFrame para la hoja LineAsym
line_asym_df = pd.read_excel(file_path, 'LineAsym')
# Ruta del archivo Excel para la hoja Libreria Lineas
line_library_file_path = r'C:\Users\nunz1\OneDrive\Documentos\Universidad\2024\TFG\Circuito
\Informacion del circuito\Libreria_Lineas.xlsx'

# Leer el archivo Excel en un DataFrame para la hoja Libreria Lineas
line_library_df = pd.read_excel(line_library_file_path)

# Elimina todas las lineas que salen y entran al mismo bus
without_equal_df1 = line_asym_df[line_asym_df['Node1'] != line_asym_df['Node2']]

# Realizar merge con las columnas LibraryType
without_equal_df = pd.merge(without_equal_df1, line_library_df, on='LibraryType', how='left')

# Agregar sufijos a las columnas node1 y node2 segun el tipo de LibraryType
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_ABC:|MT_ABC:'), ['Node1', 'Node2']]
+= '.1.2.3'
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_A:|MT_A:'), ['Node1', 'Node2']]
+= '.1'
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_B:|MT_B:'), ['Node1', 'Node2']]
+= '.2'
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_C:|MT_C:'), ['Node1', 'Node2']]
+= '.3'
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_AB:|MT_AB:'), ['Node1', 'Node2']]
+= '.1.2'
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_AC:|MT_AC:'), ['Node1', 'Node2']]
+= '.1.3'
without_equal_df.loc[without_equal_df['LibraryType'].str.contains('MT_SUB_BC:|MT_BC:'), ['Node1', 'Node2']]
+= '.2.3'

# Dataframe para media tension trifasico
df_mt_abc = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_ABC:|MT_ABC:')]
# Dataframe para media tension monofasico fase a
df_mt_a = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_A:|MT_A:')]
# Dataframe para media tension monofasico fase b
df_mt_b = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_B:|MT_B:')]
# Dataframe para media tension monofaico fase c
df_mt_c = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_C:|MT_C:')]
# Dataframe para media tension bifasico fase ab
df_mt_ab = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_AB:|MT_AB:')]
# Dataframe para media tension bifasico fase bc
df_mt_bc = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_BC:|MT_BC:')]
# Dataframe para media tension bifasico fase ac
df_mt_ac = without_equal_df[without_equal_df['LibraryType'].str.contains('MT_SUB_AC:|MT_AC:')]

# Crear DataFrame para el primer conjunto de condiciones
new_df_lista_lineas_mt_abc = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + df_mt_abc['Name'],
    'phase': 'phases=3',
    'Node1': 'bus1=' + df_mt_abc['Node1'],
    'Node2': 'bus2=' + df_mt_abc['Node2'],
    'Length': 'length=' + df_mt_abc['Length'].astype(str),
    'Units': 'units=km',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_abc['R_RR'].astype(str) + '|' +
               df_mt_abc['R_RS'].astype(str) + ' ' + df_mt_abc['R_SS'].astype(str) +
               '|' + df_mt_abc['R_RT'].astype(str) + ' ' + df_mt_abc['R_ST'].astype(str) + ' ' +
               df_mt_abc['R_TT'].astype(str) + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_abc['X_RR'].astype(str) + '|' +
               df_mt_abc['X_RS'].astype(str) + ' ' + df_mt_abc['X_SS'].astype(str) +
               '|' + df_mt_abc['X_RT'].astype(str) + ' ' + df_mt_abc['X_ST'].astype(str) + ' ' +
               df_mt_abc['X_TT'].astype(str) + ']',
    'MatrixC': 'xmatrix=' + ' '+ '[' + df_mt_abc['C_RR'].astype(str) + '|' +
               df_mt_abc['C_RS'].astype(str) + ' ' + df_mt_abc['C_SS'].astype(str) +
               '|' + df_mt_abc['C_RT'].astype(str) + ' ' + df_mt_abc['C_ST'].astype(str) + ' ' +
               df_mt_abc['C_TT'].astype(str) + ']',
    })

    # Crear DataFrame para el segundo conjunto de condiciones
new_df_lista_lineas_mt_a = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + df_mt_a['Name'],
    'Node1': 'bus1=' + df_mt_a['Node1'],
    'Node2': 'bus2=' + df_mt_a['Node2'],
    'Length': 'length=' + df_mt_a['Length'].astype(str),
    'Units': 'units=km',
    'phase': 'phases=1',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_a['R_RR'].astype(str) + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_a['X_RR'].astype(str) + ']',
    'MatrixC': 'cmatrix=' + ' '+ '[' + df_mt_a['C_RR'].astype(str) + ']',
    })
```

Figura A.2: Sección de definición de líneas de media tensión GeneradorArchivoDSS parte 1
Fuente: Elaboración propia.

```
    # Crear DataFrame para el tercer conjunto de condiciones
new_df_lista_lineas_mt_b = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + df_mt_b['Name'],
    'Node1': 'bus1=' + df_mt_b['Node1'],
    'Node2': 'bus2=' + df_mt_b['Node2'],
    'Length': 'length=' + df_mt_b['Length'].astype(str),
    'Units': 'units=km',
    'phase': 'phases=1',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_b['R_SS'].astype(str) + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_b['X_SS'].astype(str) + ']',
    'MatrixC': 'cmatrix=' + ' '+ '[' + df_mt_b['C_SS'].astype(str) +  ']',
    })

# Crear DataFrame para el cuarto conjunto de condiciones
new_df_lista_lineas_mt_c = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + df_mt_c['Name'],
    'Node1': 'bus1=' + df_mt_c['Node1'],
    'Node2': 'bus2=' + df_mt_c['Node2'],
    'Length': 'length=' + df_mt_c['Length'].astype(str),
    'Units': 'units=km',
    'phase': 'phases=1',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_c['R_TT'].astype(str) + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_c['X_TT'].astype(str) + ']',
    'MatrixC': 'cmatrix=' + ' '+ '[' + df_mt_c['C_TT'].astype(str) +  ']',
    })
# Crear DataFrame para el quinto conjunto de condiciones
new_df_lista_lineas_mt_ab = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + df_mt_ab['Name'],
    'Node1': 'bus1=' + df_mt_ab['Node1'],
    'Node2': 'bus2=' + df_mt_ab['Node2'],
    'Length': 'length=' + df_mt_ab['Length'].astype(str),
    'Units': 'units=km',
    'phase': 'phases=2',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_ab['R_RR'].astype(str) + '|' +
               df_mt_ab['R_RS'].astype(str) + ' ' + df_mt_ab['R_SS'].astype(str)+ ' ' + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_ab['X_RR'].astype(str) + '|' +
               df_mt_ab['X_RS'].astype(str) + ' ' + df_mt_ab['X_SS'].astype(str) + ' '+ ']',
    'MatrixC': 'cmatrix=' + ' '+ '[' + df_mt_ab['C_RR'].astype(str) + '|' +
               df_mt_ab['C_RS'].astype(str) + ' ' + df_mt_ab['C_SS'].astype(str) + ' '+ ']',
    })

# Crear DataFrame para el sexto conjunto de condiciones
new_df_lista_lineas_mt_bc = pd.DataFrame({
    'Name': 'new' + ' ' +  'line.' + df_mt_bc['Name'],
    'Node1': 'bus1=' + df_mt_bc['Node1'],
    'Node2': 'bus2=' + df_mt_bc['Node2'],
    'Length': 'length=' + df_mt_bc['Length'].astype(str),
    'Units': 'units=km',
    'phase': 'phases=2',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_bc['R_SS'].astype(str) + '|' +
               df_mt_bc['R_ST'].astype(str) + ' ' + df_mt_bc['R_TT'].astype(str)+ ' ' + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_bc['X_SS'].astype(str) + '|' +
               df_mt_bc['X_ST'].astype(str) + ' ' + df_mt_bc['X_TT'].astype(str)+ ' ' + ']',
    'MatrixC': 'cmatrix=' + ' '+ '[' + df_mt_bc['C_SS'].astype(str) + '|' +
               df_mt_bc['C_ST'].astype(str) + ' ' + df_mt_bc['C_TT'].astype(str) + ' ' +  ']',
    })
# Crear DataFrame para el septimo conjunto de condiciones
new_df_lista_lineas_mt_ac = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + df_mt_ac['Name'],
    'Node1': 'bus1=' + df_mt_ac['Node1'],
    'Node2': 'bus2=' + df_mt_ac['Node2'],
    'Length': 'length=' + df_mt_ac['Length'].astype(str),
    'Units': 'units=km',
    'phase': 'phases=2',
    'MatrixR': 'rmatrix=' + ' ' + '[' + df_mt_ac['R_RR'].astype(str) + '|' +
               df_mt_ac['R_RT'].astype(str) + ' ' + df_mt_ac['R_TT'].astype(str) + ' ' + ']',
    'MatrixX': 'xmatrix=' + ' '+ '[' + df_mt_ac['X_RR'].astype(str) + '|' +
               df_mt_ac['X_RT'].astype(str) + ' ' + df_mt_ac['X_TT'].astype(str) + ' ' + ']',
    'MatrixC': 'cmatrix=' + ' '+ '[' + df_mt_ac['C_RR'].astype(str) + '|' +
               df_mt_ac['C_RT'].astype(str) + ' ' + df_mt_ac['C_TT'].astype(str) + ' ' + ']',
    })

    # Ordenar los dataframe
output_df_Lineas 1 = pd.concat([new_df_lista_lineas_mt_abc['Name'], new_df_lista_lineas_mt_abc['phase'],
    new_df_lista_lineas_mt_abc['Node1'], new_df_lista_lineas_mt_abc['Node2'],
    new_df_lista_lineas_mt_abc['Length'], new_df_lista_lineas_mt_abc['Units'],
    new_df_lista_lineas_mt_abc['MatrixR'], new_df_lista_lineas_mt_abc['MatrixX'],
    new_df_lista_lineas_mt_abc['MatrixC'],
    ], axis=1)
output_df_Lineas 2 = pd.concat([new_df_lista_lineas_mt_a['Name'], new_df_lista_lineas_mt_a['phase'],
    new_df_lista_lineas_mt_a['Node1'], new_df_lista_lineas_mt_a['Node2'],
    new_df_lista_lineas_mt_a['Length'], new_df_lista_lineas_mt_a['Units'],
    new_df_lista_lineas_mt_a['MatrixR'], new_df_lista_lineas_mt_a['MatrixX'],
    new_df_lista_lineas_mt_a['MatrixC'],
    ], axis=1)
```

Figura A.3: Sección de definición de líneas de media tensión GeneradorArchivoDSS parte 2
Fuente: Elaboración propia.

```
output_df_Lineas 3  = pd.concat([new_df_lista_lineas_mt_b['Name'], new_df_lista_lineas_mt_b['phase'],
    new_df_lista_lineas_mt_b['Node1'], new_df_lista_lineas_mt_b['Node2'],
    new_df_lista_lineas_mt_b['Length'], new_df_lista_lineas_mt_b['Units'],
    new_df_lista_lineas_mt_b['MatrixR'], new_df_lista_lineas_mt_b['MatrixX'],
    new_df_lista_lineas_mt_b['MatrixC'],
    ], axis=1)
output_df_Lineas 4= pd.concat([new_df_lista_lineas_mt_c['Name'], new_df_lista_lineas_mt_c['phase'],
    new_df_lista_lineas_mt_c['Node1'], new_df_lista_lineas_mt_c['Node2'],
    new_df_lista_lineas_mt_c['Length'], new_df_lista_lineas_mt_c['Units'],
    new_df_lista_lineas_mt_c['MatrixR'], new_df_lista_lineas_mt_c['MatrixX'],
    new_df_lista_lineas_mt_c['MatrixC'],
    ], axis=1)
output_df_Lineas 5=  pd.concat([ new_df_lista_lineas_mt_ab['Name'], new_df_lista_lineas_mt_ab['phase'],
    new_df_lista_lineas_mt_ab['Node1'],new_df_lista_lineas_mt_ab['Node2'],
    new_df_lista_lineas_mt_ab['Length'], new_df_lista_lineas_mt_ab['Units'],
    new_df_lista_lineas_mt_ab['MatrixR'], new_df_lista_lineas_mt_ab['MatrixX'],
    new_df_lista_lineas_mt_ab['MatrixC'],
    ], axis=1)
output_df_Lineas 6= pd.concat([new_df_lista_lineas_mt_bc['Name'], new_df_lista_lineas_mt_bc['phase'],
    new_df_lista_lineas_mt_bc['Node1'], new_df_lista_lineas_mt_bc['Node2'],
    new_df_lista_lineas_mt_bc['Length'], new_df_lista_lineas_mt_bc['Units'],
    new_df_lista_lineas_mt_bc['MatrixR'], new_df_lista_lineas_mt_bc['MatrixX'],
    new_df_lista_lineas_mt_bc['MatrixC'],
    ], axis=1)

output_df_Lineas 7=  pd.concat([ new_df_lista_lineas_mt_ac['Name'], new_df_lista_lineas_mt_ac['phase'],
    new_df_lista_lineas_mt_ac['Node1'], new_df_lista_lineas_mt_ac['Node2'],
    new_df_lista_lineas_mt_ac['Length'], new_df_lista_lineas_mt_ac['Units'],
    new_df_lista_lineas_mt_ac['MatrixR'], new_df_lista_lineas_mt_ac['MatrixX'],
    new_df_lista_lineas_mt_ac['MatrixC'],
    ], axis=1)

#Unir todos los dataframes monofasicos
output_df_Lineas_mv  =  pd.concat([output_df_Lineas1, output_df_Lineas2,
output_df_Lineas3, output_df_Lineas4, output_df_Lineas5, output_df_Lineas6, output_df_Lineas7])

without_duplicates_df_mv  =  output_df_Lineas_mv.drop_duplicates(subset=['Name'])
without_duplicates_df_mv  =  without_duplicates_df_mv.drop_duplicates(subset=['Node1','Node2'], keep='first')
without_duplicates_df_mv  =  without_duplicates_df_mv[without_duplicates_df_mv['Length'] != 0]

# Guardar el DataFrame en un archivo de texto con extension .dss para las lineas trifasicas
output_file_Lineas_mv = 'Lista_Lineas_MV.dss'

with  open(output_file_Lineas_mv, 'w') as archivo:
    without_duplicates_df_mv.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE,
    escapechar=' ')

# Abrir automaticamente el archivo .dss en notas
subprocess.run(['notepad.exe', output_file_Lineas_mv])
```

Figura A.4: Sección de definición de líneas de media tensión GeneradorArchivoDSS parte 3
Fuente: Elaboración propia.

```
''Archivo de transformadores''
# Ruta del archivo Excel
libreria_trafo = r'C:\Users\nunz1\OneDrive\Documentos\Universidad\2024\TFG\Circuito\Informacion del
circuito\LibreriaTrafos.xlsx'

# Leer el archivo Excel libreriatrafo hoja monofasicos
libreria_trafo_mono_df = pd.read_excel(libreria_trafo, 'monofasicos')
# Leer el archivo Excel libreriatrafo hoja bifasicos
libreria_trafo_bi_df = pd.read_excel(libreria_trafo, 'bifasicos')
# Leer el archivo Excel libreriatrafo hoja trifasicos
libreria_trafo_tri_df = pd.read_excel(libreria_trafo, 'trifasicos')

# Leer el archivo Excel y crear un DataFrame
trafo_mono = pd.read_excel(file_path, sheet_name='Trafo2WindingAsym ')

#Unir libreria de trafos con merge
trafo_mono_df = pd.merge(trafo_mono, libreria_trafo_mono_df, on='Library Type',how='left')
#Unir libreria de trafos con merge
trafo_bi_df = pd.merge(trafo_mono, libreria_trafo_bi_df, on='LibraryType', how='left')

# Leer el archivo Excel en un DataFrame para la hoja Trafo2Windingt
rafo_tri = pd.read_excel(file_path, 'Trafo2Winding')

trafo_tri_df = pd.merge(trafo_tri, libreria_trafo_tri_df, on='LibraryType', how='left')#

filtrar trafos monofasicos
trafo_mono_df_a = trafo_mono_df.loc[trafo_mono_df['Library Type'].astype(str).str.startswith('A_')]
trafo_mono_df_b = trafo_mono_df.loc[trafo_mono_df['Library Type'].astype(str).str.startswith('B_')]
trafo_mono_df_c = trafo_mono_df.loc[trafo_mono_df['LibraryType'].astype(str).str.startswith('C_')]

# filtrar trafos monofasicos estrella-delta abierta
trafo_mono_df_ab = trafo_bi_df.loc[trafo_bi_df['Library Type'].astype(str).str.match(r'^(AB|BA)_')]
trafo_mono_df_bc = trafo_bi_df.loc[trafo_bi_df['Library Type'].astype(str).str.match(r'^(BC|CB)_')]
trafo_mono_df_ac = trafo_bi_df.loc[trafo_bi_df['LibraryType'].astype(str).str.match(r'^(AC|CA)_')]

# Crear DataFrame para el formato del archivo de transformadores monofasicos
new_df_trafo_mono_a = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_a['Name'],
    'phases': 'phases=1',
    'windings': 'windings=3',
    'Xhl': 'Xhl=' + trafo_mono_df_a['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_a['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_a['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_a['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_a['%noloadloss'].astype(str),
    '%imag':'%imag=' + trafo_mono_df_a['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_a['Node1'] + '.1.0' + ' ' +
    trafo_mono_df_a['Node2'] +'.1.0' + ' ' + trafo_mono_df_a['Node2'] +'.0.2' + ']',
    'kvs': 'kvs=' + trafo_mono_df_a['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_a['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye wye]',

})

new_df_trafo_mono_b = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_b['Name'],
    'phases': 'phases=1',
    'windings': 'windings=3',
    'Xhl': 'Xhl=' + trafo_mono_df_b['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_b['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_b['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_b['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_b['%noloadloss'].astype(str),
    '%imag':'%imag=' + trafo_mono_df_b['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_b['Node1'] + '.2.0' + ' ' +
    trafo_mono_df_b['Node2'] + '.1.0' + ' ' + trafo_mono_df_b['Node2'] + '.0.2' + ']',
    'kvs': 'kvs=' + trafo_mono_df_b['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_b['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye wye]',
})

new_df_trafo_mono_c = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_c['Name'],
    'phases': 'phases=1',
    'windings': 'windings=3',
    'Xhl': 'Xhl=' + trafo_mono_df_c['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_c['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_c['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_c['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_c['%noloadloss'].astype(str),
    '%imag':'%imag=' + trafo_mono_df_c['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_c['Node1'] + '.3.0' + ' ' +
    trafo_mono_df_c['Node2'] + '.1.0' + ' ' + trafo_mono_df_c['Node2'] + '.0.2' + ']',
    'kvs': 'kvs=' + trafo_mono_df_c['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_c['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye wye]',
})
```

Figura A.5: Sección del código GeneradorArchivoDSS.py para definición de los transformadores parte 1
Fuente: Elaboración propia.

```
# Crear DataFrame para el formato del archivo de transformadores bifasicos
new_df_trafo_mono_ab = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_ab['Name'] + '_A',
    'phases': 'phases=1',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_mono_df_ab['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_ab['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_ab['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_ab['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_ab['%noloadloss'].astype(str),
    '%imag': '%imag=' + trafo_mono_df_ab['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_ab['Node1'] + '.1.0' + ' ' + trafo_mono_df_ab['Node2']
+ '.2.1' + ']',
    'kvs': 'kvs=' + trafo_mono_df_ab['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_ab['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye]',
    })
new_df_trafo_mono_ab2 = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_ab['Name'] + '_B',
    'phases': 'phases=1',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_mono_df_ab['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_ab['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_ab['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_ab['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_ab['%noloadloss'].astype(str),
    '%imag': '%imag=' + trafo_mono_df_ab['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_ab['Node1'] + '.2.0' + ' ' + trafo_mono_df_ab['Node2']
+ '.3.2' + ']',
    'kvs': 'kvs=' + trafo_mono_df_ab['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_ab['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye]',
})

new_df_trafo_mono_bc = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_bc['Name'] + '_B',
    'phases': 'phases=1',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_mono_df_bc['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_bc['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_bc['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_bc['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_bc['%noloadloss'].astype(str),
    '%imag': '%imag=' + trafo_mono_df_bc['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_bc['Node1'] + '.2.0' + ' ' + trafo_mono_df_bc['Node2'] +'.2.1' + ']',
    'kvs': 'kvs=' + trafo_mono_df_bc['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_bc['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye]',
    })
new_df_trafo_mono_bc2 = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_bc['Name'] + '_C',
    'phases': 'phases=1',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_mono_df_bc['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_bc['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_bc['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_bc['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_bc['%noloadloss'].astype(str),
    '%imag': '%imag=' + trafo_mono_df_bc['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_bc['Node1'] + '.3.0' + ' ' + trafo_mono_df_bc['Node2'] +'.3.2' + ']',
    'kvs': 'kvs=' + trafo_mono_df_bc['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_bc['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye]',
})

new_df_trafo_mono_ac = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_ac['Name'] + '_A',
    'phases': 'phases=1',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_mono_df_ac['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_ac['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_ac['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_ac['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_ac['%noloadloss'].astype(str),
    '%imag': '%imag=' + trafo_mono_df_ac['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_ac['Node1'] + '.1.0' + ' ' + trafo_mono_df_ac['Node2'] +'.2.1' + ']',
    'kvs': 'kvs=' + trafo_mono_df_ac['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_ac['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye]',
    })
new_df_trafo_mono_ac2 = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_mono_df_ac['Name'] + '_C',
    'phases': 'phases=1',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_mono_df_ac['Xhl'].astype(str),
    'Xht': 'Xht=' + trafo_mono_df_ac['Xht'].astype(str),
    'Xlt': 'Xlt=' + trafo_mono_df_ac['Xlt'].astype(str),
    '%Rs': '%Rs=' + trafo_mono_df_ac['%Rs'].astype(str),
    '%noloadloss': '%noloadloss=' + trafo_mono_df_ac['%noloadloss'].astype(str),
    '%imag': '%imag=' + trafo_mono_df_ac['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_mono_df_ac['Node1'] + '.3.0' + ' ' + trafo_mono_df_ac['Node2'] +'.3.2' + ']',
    'kvs': 'kvs=' + trafo_mono_df_ac['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_mono_df_ac['kVAs'].astype(str),
    'conns': 'conns=' + '[wye wye]',
})
```

Figura A.6: Sección del código GeneradorArchivoDSS.py para definición de los transformadores parte 2
Fuente: Elaboración propia.

```
# Crear DataFrame para el formato del archivo de transformadores trifasicos
new_df_trafo_tri = pd.DataFrame({
    'Name': 'new' + ' ' + 'transformer.' + trafo_tri_df['Name'],
    'phases': 'phases=3',
    'windings': 'windings=2',
    'Xhl': 'Xhl=' + trafo_tri_df['Xhl'].astype(str),
    '%Rs': '%Rs=' + trafo_tri_df['%Rs'].astype(str) ,
    '%noloadloss': '%noloadloss=' + trafo_tri_df['%noloadloss'].astype(str) ,
    '%imag':'%imag=' + trafo_tri_df['%imag'].astype(str),
    'Buses': 'Buses=' + '['+ trafo_tri_df['Node1'] + '.1.2.3' + ' ' + trafo_tri_df['Node2'] + '.1.2.3' + ']',
    'kvs': 'kvs=' + trafo_tri_df['kvs'].astype(str),
    'kVAs': 'kVAs=' + trafo_tri_df['kVAs'].astype(str),
    'conns': trafo_tri_df['LibraryType'].apply(lambda x: 'conns=[wye, wye]' if 'Estrella_Estrella'
    in x else ('conns=[wye, delta]' if 'Estrella_Fase_Partida' in x else 'conns=')),
})

#dataframe con los datos ordenados para monofasicos
output_df_trafo_mono_a = pd.concat([new_df_trafo_mono_a['Name'], new_df_trafo_mono_a['phases'],
    new_df_trafo_mono_a['windings'], new_df_trafo_mono_a['Xhl'], new_df_trafo_mono_a['Xht'],
    new_df_trafo_mono_a['Xlt'], new_df_trafo_mono_a['%Rs'], new_df_trafo_mono_a['%noloadloss'],
    new_df_trafo_mono_a['%imag'], new_df_trafo_mono_a['Buses'], new_df_trafo_mono_a['kvs'],
    new_df_trafo_mono_a['kVAs'], new_df_trafo_mono_a['conns'],
    ], axis=1)
output_df_trafo_mono_b = pd.concat([new_df_trafo_mono_b['Name'], new_df_trafo_mono_b['phases'],
    new_df_trafo_mono_b['windings'], new_df_trafo_mono_b['Xhl'], new_df_trafo_mono_b['Xht'],
    new_df_trafo_mono_b['Xlt'], new_df_trafo_mono_b['%Rs'], new_df_trafo_mono_b['%noloadloss'],
    new_df_trafo_mono_b['%imag'], new_df_trafo_mono_b['Buses'], new_df_trafo_mono_b['kvs'],
    new_df_trafo_mono_b['kVAs'], new_df_trafo_mono_b['conns'],
    ], axis=1)
output_df_trafo_mono_c = pd.concat([new_df_trafo_mono_c['Name'], new_df_trafo_mono_c['phases'],
    new_df_trafo_mono_c['windings'], new_df_trafo_mono_c['Xhl'],
    new_df_trafo_mono_c['Xht'],
    new_df_trafo_mono_c['Xlt'], new_df_trafo_mono_c['%Rs'], new_df_trafo_mono_c['%noloadloss'],
    new_df_trafo_mono_c['%imag'], new_df_trafo_mono_c['Buses'], new_df_trafo_mono_c['kvs'],
    new_df_trafo_mono_c['kVAs'], new_df_trafo_mono_c['conns'],
    ], axis=1)
output_df_trafo_mono_ab = pd.concat([new_df_trafo_mono_ab['Name'], new_df_trafo_mono_ab['phases'],
    new_df_trafo_mono_ab['windings'], new_df_trafo_mono_ab['Xhl'], new_df_trafo_mono_ab['Xht'],
    new_df_trafo_mono_ab['Xlt'], new_df_trafo_mono_ab['%Rs'], new_df_trafo_mono_ab['%noloadloss'],
    new_df_trafo_mono_ab['%imag'], new_df_trafo_mono_ab['Buses'], new_df_trafo_mono_ab['kvs'],
    new_df_trafo_mono_ab['kVAs'], new_df_trafo_mono_ab['conns'],
    ], axis=1)
output_df_trafo_mono_bc = pd.concat([new_df_trafo_mono_bc['Name'], new_df_trafo_mono_bc['phases'],
    new_df_trafo_mono_bc['windings'], new_df_trafo_mono_bc['Xhl'],      new_df_trafo_mono_bc['Xht'],
    new_df_trafo_mono_bc['Xlt'], new_df_trafo_mono_bc['%Rs'], new_df_trafo_mono_bc['%noloadloss'],
    new_df_trafo_mono_bc['%imag'], new_df_trafo_mono_bc['Buses'], new_df_trafo_mono_bc['kvs'],
    new_df_trafo_mono_bc['kVAs'], new_df_trafo_mono_bc['conns'],
    ], axis=1)
output_df_trafo_mono_ac = pd.concat([new_df_trafo_mono_ac['Name'], new_df_trafo_mono_ac['phases'],
    new_df_trafo_mono_ac['windings'], new_df_trafo_mono_ac['Xhl'], new_df_trafo_mono_ac['Xht'],
    new_df_trafo_mono_ac['Xlt'], new_df_trafo_mono_ac['%Rs'], new_df_trafo_mono_ac['%noloadloss'],
    new_df_trafo_mono_ac['%imag'], new_df_trafo_mono_ac['Buses'], new_df_trafo_mono_ac['kvs'],
    new_df_trafo_mono_ac['kVAs'], new_df_trafo_mono_ac['conns'],
    ], axis=1)
output_df_trafo_mono_ab2 = pd.concat([new_df_trafo_mono_ab2['Name'], new_df_trafo_mono_ab2['phases'],
    new_df_trafo_mono_ab2['windings'], new_df_trafo_mono_ab2['Xhl'], new_df_trafo_mono_ab2['Xht'],
    new_df_trafo_mono_ab2['Xlt'], new_df_trafo_mono_ab2['%Rs'], new_df_trafo_mono_ab2['%noloadloss'],
    new_df_trafo_mono_ab2['%imag'], new_df_trafo_mono_ab2['Buses'], new_df_trafo_mono_ab2['kvs'],
    new_df_trafo_mono_ab2['kVAs'], new_df_trafo_mono_ab2['conns'],
    ], axis=1)
output_df_trafo_mono_bc2 = pd.concat([new_df_trafo_mono_bc2['Name'], new_df_trafo_mono_bc2['phases'],
    new_df_trafo_mono_bc2['windings'], new_df_trafo_mono_bc2['Xhl'], new_df_trafo_mono_bc2['Xht'],
    new_df_trafo_mono_bc2['Xlt'], new_df_trafo_mono_bc2['%Rs'], new_df_trafo_mono_bc2['%noloadloss'],
    new_df_trafo_mono_bc2['%imag'], new_df_trafo_mono_bc2['Buses'], new_df_trafo_mono_bc2['kvs'],
    new_df_trafo_mono_bc2['kVAs'],new_df_trafo_mono_bc2['conns'],
   ], axis=1)
output_df_trafo_mono_ac2 = pd.concat([new_df_trafo_mono_ac2['Name'], new_df_trafo_mono_ac2['phases'],
    new_df_trafo_mono_ac2['windings'], new_df_trafo_mono_ac2['Xhl'], new_df_trafo_mono_ac2['Xht'],
    new_df_trafo_mono_ac2['Xlt'], new_df_trafo_mono_ac2['%Rs'], new_df_trafo_mono_ac2['%noloadloss'],
    new_df_trafo_mono_ac2['%imag'], new_df_trafo_mono_ac2['Buses'], new_df_trafo_mono_ac2['kvs'],
    new_df_trafo_mono_ac2['kVAs'], new_df_trafo_mono_ac2['conns'],
    ], axis=1)

output_df_trafo_bi = pd.concat([output_df_trafo_mono_ab, output_df_trafo_mono_ab2    , output_df_trafo_mono_bc,
output_df_trafo_mono_bc2    , output_df_trafo_mono_ac    , output_df_trafo_mono_ac2])

# Ordenar el DataFrame segun la columna 'Name'
output_df_trafo_bi = output_df_trafo_bi.sort_values(by='Name')
output_df_trafo_mono = pd.concat([output_df_trafo_mono_a, output_df_trafo_mono_b    , output_df_trafo_mono_c ])

#Unir todos los DataFrames
output_df_trafo = pd.concat([output_df_trafo_mono, output_df_trafo_bi , output_df_trafo_tri])

#elimina las lineas repetidas
output_df_trafo = output_df_trafo.drop_duplicates(subset=['Name'])

# Guardar el DataFrame en un archivo de texto con extension .dss
output_file_trafos = 'trafos.dss'

with open(output_file_trafos, 'w') as archivo:
    output_df_trafo.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE, escapechar=' ')
```

Figura A.7: Sección del código GeneradorArchivoDSS.py para definición de los transformadores parte 3
Fuente: Elaboración propia.

```
 Filtrar filas que contienen lineas de baja tension secundarias subterraneas
subterraneas = line_asym_df[line_asym_df['LibraryType'].str.startswith('BT_SUB')]

# Filtrar filas que contienen lineas de baja tension secundarias
secundarias = line_asym_df[line_asym_df['LibraryType'].str.startswith('BT_') &
                  ~line_asym_df['LibraryType'].str.contains('BT_SUB', na=False) &
                  line_asym_df['LibraryType'].str.endswith('_1')]

# Filtrar filas que contienen lineas de baja tension acometidas
acometidas = line_asym_df[line_asym_df['LibraryType'].str.startswith('BT_') &
                 ~line_asym_df['LibraryType'].str.contains('BT_SUB', na=False) &
                 (line_asym_df['LibraryType'].str.endswith('_3') |line_asym_df['LibraryType'].str.endswith('_6'))]

# Filtrar filas que contienen lineas de baja tension acometidas QUADRUPLEX
acometidas4 = line_asym_df[line_asym_df['LibraryType'].str.startswith('BT_')&
                 ~line_asym_df['LibraryType'].str.contains('BT_SUB', na=False) &
                 (line_asym_df['LibraryType'].str.endswith('_4'))]

# Crear DataFrame para las lineas secudarias
new_df_secundarias = pd.DataFrame({
    'Name': 'new' + ' ' + 'line.' + secundarias['Name'],
    'Node1': 'bus1=' + secundarias['Node1'] + '.1.2',
    'Node2': 'bus2=' + secundarias['Node2'] + '.1.2',
    'Length': 'length=' + secundarias['Length'].astype(str),
    'Units': 'units=km',
    'geometry': 'Linecode=' + secundarias['LibraryType'].str.lower().str.replace(' ', ''),
})

# Condiciones y valores correspondientes para libreria lineas de acometidas
def get_linecode(library_type):
    if library_type.startswith('BT_6 AAAC'):
        return 'tpx_6_Hippa'
    elif library_type.startswith('BT_1/0 AAAC'):
        return 'tpx_1/0_Leda'
    elif library_type.startswith('BT_6 ACSR'):
        return 'tpx_6_Voluta'
    elif library_type.startswith('BT_1/0 CO'):
        return 'tpx_1/0_Corinthian'
    elif library_type.startswith('BT_123.3 AAAC'):
        return 'tpx_1/0_Leda'
    elif library_type.startswith('BT_2 ACSR'):
        return 'tpx_2_Conch'
    elif library_type.startswith('BT_6 CO'):
        return 'tpx_6_Tudor'
    elif library_type.startswith('BT_2 CO'):
        return 'tpx_2_Minion'
    elif library_type.startswith('BT_8 ACSR'):
        return 'tpx_6_Voluta'
    elif library_type.startswith('BT_1/0 ACSR'):
        return 'tpx_1/0_Cenia'
    elif library_type.startswith('BT_8 CO'):
        return 'tpx_8_Pica'
    elif library_type.startswith('BT_4 ACSR'):
        return 'tpx_4_Periwinkle'
    else:
        return 'Unknown'

# Condiciones y valores correspondientes para libreria lineas cuadruplex
def get_linecode4(library_type):
    if library_type.startswith('BT_1/0 CO'):
        return 'qdpx_1/0_Nashville'
    elif library_type.startswith('BT_6 ACSR'):
        return 'qdpx_6_Chola'
    elif library_type.startswith('BT_1/0 ACSR'):
        return 'qdpx_1/0_Costena'
    elif library_type.startswith('BT_2 ACSR'):
        return 'qdpx_2_Palomino'
    elif library_type.startswith('BT_123.3 AAAC'):
        return 'qdpx_1/0_Shetland'
    elif library_type.startswith('BT_1/0 AAAC'):
        return 'qdpx_1/0_Shetland'
    else:
        return 'Unknown'

    # Definir funcion para definir el Library Type
def get_linecode_subterraneas(library_type):
    if '250 COBRE' in library_type:
        return 'UGSC1F_250'
    elif '1/0 COBRE' in library_type:
        return 'UGSC1F_1/0'
    elif '2 COBRE' in library_type:
        if library_type.startswith('BT_SUB_ABC'):
            return 'UGSC3F_2'
        else:
            return 'UGSC1F_2'
    else:
        return 'Unknown'
```

Figura A.8: Sección del código GeneradorArchivoDSS.py para definir las líneas de baja tensión parte 1
Fuente: Elaboración propia.

```
#Definir la estructura del archivo de acometidas y lineas secundarias
new_df_acometidas = pd.DataFrame({
    'Name': 'new line.' + acometidas['Name'],
    'Node1': 'bus1=' + acometidas['Node1'] + '.1.2',
    'Node2': 'bus2=' + acometidas['Node2'] + '.1.2',
    'Length': 'length=' + acometidas['Length'].astype(str),
    'Units': 'units=km',
    'geometry': 'Linecode=' + acometidas['LibraryType'].apply(get_linecode)
})
new_df_acometidas4 = pd.DataFrame({
    'Name': 'new line.' + acometidas4['Name'],
    'Node1': 'bus1=' + acometidas4['Node1'] + '.1.2',
    'Node2': 'bus2=' + acometidas4['Node2'] + '.1.2',
    'Length': 'length=' + acometidas4['Length'].astype(str),
    'Units': 'units=km',
    'geometry': 'Linecode=' + acometidas4['LibraryType'].apply(get_linecode4)
})

# Definir la estructura del archivo de subterraneas
new_df_subterraneas = pd.DataFrame({
    'Name': 'new line.' + subterraneas['Name'],
    'Node1': 'bus1=' + subterraneas['Node1'] + '.1.2',
    'Node2': 'bus2=' + subterraneas['Node2'] + '.1.2',
    'Length': 'length=' + subterraneas['Length'].astype(str),
    'Units': 'units=km',
    'geometry': 'Linecode=' + subterraneas['LibraryType'].apply(get_linecode_subterraneas)
})
# Ordenar los dataframe
output_df_secundarias = pd.concat([new_df_secundarias['Name'], new_df_secundarias['Node1'],
    new_df_secundarias['Node2'], new_df_secundarias['Length'],
    new_df_secundarias['Units'], new_df_secundarias['geometry'],
    ], axis=1)
output_df_acometidas3 = pd.concat([new_df_acometidas['Name'], new_df_acometidas['Node1'],
    new_df_acometidas['Node2'], new_df_acometidas['Length'],
    new_df_acometidas['Units'], new_df_acometidas['geometry'],
    ], axis=1)
output_df_acometidas4 = pd.concat([new_df_acometidas4['Name'], new_df_acometidas4['Node1'],
    new_df_acometidas4['Node2'], new_df_acometidas4['Length'],
    new_df_acometidas4['Units'], new_df_acometidas4['geometry'],
    ], axis=1)
output_df_subterraneas = pd.concat([new_df_subterraneas['Name'], new_df_subterraneas['Node1'],
    new_df_subterraneas['Node2'], new_df_subterraneas['Length'],
    new_df_subterraneas['Units'], new_df_subterraneas['geometry'],
    ], axis=1)
output_df_acometidas =pd.concat([output_df_acometidas3, output_df_acometidas4])
output_df_acometidas = output_df_acometidas.drop_duplicates(subset=['Name'])

# Guardar el DataFrame en un archivo de texto con extension .dss para lineas monofasicas
output_file_Lineas_BV_subterraneas= 'Lista_Lineas_BV_subterraneas.dss'
with open(output_file_Lineas_BV_subterraneas, 'w') as archivo:
    output_df_subterraneas.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE,
    escapechar=' ')
# Guardar el DataFrame en un archivo de texto con extension .dss para lineas monofasicas
output_file_Lineas_BV_secundarias= 'Lista_Lineas_BV_secundarias.dss'
with open(output_file_Lineas_BV_secundarias, 'w') as archivo:
    output_df_secundarias.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE,
    escapechar=' ')
# Guardar el DataFrame en un archivo de texto con extension .dss para lineas monofasicas
output_file_Lineas_BV_acometidas= 'Lista_Lineas_BV_acometidas.dss'
with open(output_file_Lineas_BV_acometidas, 'w') as archivo:
    output_df_acometidas.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE,
    escapechar=' ')

# Abrir automaticamente el archivo .dss en notas
subprocess.run(['notepad.exe', output_file_Lineas_BV_subterraneas])
```

Figura A.9: Sección del código GeneradorArchivoDSS.py para definir las líneas de baja tensión parte 2
Fuente: Elaboración propia.

```
# Leer el archivo Excel en un DataFrame
cargas_df = pd.read_excel(file_path, 'Load')

# Crear DataFrame para el formato del archivo de cargas
new_df_cargas_1= pd.DataFrame({
    'Name': 'new' + ' ' + 'load.' + cargas_df['Name'],
    'bus': 'bus1=' + cargas_df['Node1'],
    'kV': 'kV=' + cargas_df['Un'].astype(str),
    'model': 'model=1',
    'conn': 'conn=wye',
    'kW': 'kW=' + cargas_df['S'].astype(str),
    'pf': 'pF=' + cargas_df['CosPhi'].astype(str),
    'status': 'status=fixed',
    'phases': 'phases=1',
})

#dataframe con los datos ordenados cargas tipo 1
output_cargas = pd.concat([new_df_cargas_1['Name'], new_df_cargas_1['bus'], new_df_cargas_1['kV'],
    new_df_cargas_1['model'], new_df_cargas_1['conn'], new_df_cargas_1['kW'],
    new_df_cargas_1['pf'], new_df_cargas_1['status'], new_df_cargas_1['phases'],
    ], axis=1)

output_cargas_df =output_cargas.drop_duplicates(subset=['Name'])

# Guardar el DataFrame en un archivo de texto con extension .dss
output_file_cargas = 'cargas.dss'
with open(output_file_cargas, 'w') as archivo:
    output_cargas_df.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE, escapechar=' ')

# Abrir automaticamente el archivo .dss en notas
subprocess.run(['notepad.exe', output_file_cargas])
```

Figura A.10: Seccion del codigo GeneradorArchivoDSS.py para definicion de las cargas.
Fuente: Elaboracion propia.

```
# Crear DataFrame para el formato del archivo de cargas
new_df_cargasdinamicas_list =[]

for i in range(1, 7):  # Generar 6 documentos
    new_df_cargasdinamicas = pd.DataFrame({
        'Name': 'new load.' + cargas_df['Name'],
        'bus': 'bus1=' + cargas_df['Node1'],
        'kV': 'kV=' + cargas_df['Un'].astype(str),
        'model': 'model=1',
        'conn': 'conn=wye',
        'kW': 'kW=' + cargas_df['S'].astype(str),
        'pf': 'pF=' + cargas_df['CosPhi'].astype(str),
        'status': 'status=variable',
        'phases': 'phases=1',
        'daily': 'daily=loadshape_' + str(i)
    })
    new_df_cargasdinamicas_list.append(new_df_cargasdinamicas)

# Concatenar los DataFrames de cargas dinamicas
output_cargasdinamicas = pd.concat(new_df_cargasdinamicas_list)

# Eliminar duplicados
output_cargasdinamicas = output_cargasdinamicas.drop_duplicates(subset=['Name'])

# Guardar el DataFrame en un archivo de texto con extension .dss
output_file_cargasdinamicas_prefix ='cargasdinamicas_'
for i, df in enumerate(new_df_cargasdinamicas_list):
    output_file_cargasdinamicas = output_file_cargasdinamicas_prefix + str(i+1) + '.dss'
    with open(output_file_cargasdinamicas, 'w') as archivo:
        df.to_csv(archivo, sep=' ', index=False, header=False, quoting=csv.QUOTE_NONE, escapechar=' ')

    # Abrir automaticamente el archivo .dss en notas
    subprocess.run(['notepad.exe', output_file_cargasdinamicas])
```

Figura A.11: Sección del código GeneradorArchivoDSS.py para definición de las cargas dinámicas.
Fuente: Elaboración propia.

Apéndice B

Código para el cálculo de impedancias en OpenDSS

```
//MT_A: 1/0 AAAC_AAAC_1/0_NT_NT_7_N
clear
New Circuit.Line basekv=34.5 pu=1.0 angle=0 frequency=60 phases=3 Mvasc3=2000 Mvasc1=1500

set EarthModel=Carson


// Define las caracteristicas de los conductores
Redirect WireDataAAAC.dss
Redirect WireDataAAC.dss
Redirect WireDataACSR.dss
Redirect WireDataCU.dss

// Define el espaciamento estre conductores
new LineSpacing.1F_MTH nconds=2 nphases=1 units=m x=[0 0] h=[10.74 6.7]

// Define la geometria de la linea
new Linegeometry.1FMV1/0AAAC_AAAC_1/0_NT_NT_7_N nconds=2 nphases=1 spacing=1F_MTH
~ wires =[AAAC_1/0_7STR AAAC_1/0_7STR] Reduce=Y

// muestra las impedancias y capacitancias de linea
show lineconstants [60] [km]
solve
```

Figura B.1: Ejemplo del cálculo de la matriz impedancia de una línea primaria monofásica.
Fuente: Elaboración propia.

```
Clear

New Circuit.3FAer basekv=0.240 pu=1.0 angle=0 frequency=60 phases=3 Mvasc3=2000 Mvasc1=1500
set EarthModel=Carson

Redirect WireDataACSR.dss
Redirect WireDataAAAC.dss
Redirect WireDataAAC.dss
Redirect WireDataCU.dss
Redirect CNData_CU_EPR133.dss

//Define el espaciamiento entre conductores
 new LineSpacing.1F_3WIRE nconds=3 nphases=2 units=m x=[0.20 0.20 0.20] h=[6.7 6.9 7.1]

//Define la geometria de las lineas
new Linegeometry.BT_123.3AAAC_AAAC_123.3_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_123.3_7STR AAAC_123.3_7STR AAAC_123.3_7STR] Reduce=Y
```

Figura B.2: Cálculo de matrices de impedancia para líneas secundarias parte 1
Fuente: Elaboración propia.

```
new Linegeometry.BT_1/0CO_TR_CO_TR_1/0_1  nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_1/0_7STR CU_1/0_7STR CU_1/0_7STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR AAAC_2_7STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_AAAC_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR AAAC_2_7STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_AAAC_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR AAAC_1/0_7STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_AAAC_123.3_1  nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR AAAC_123.3_7STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR ACSR_2_7STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_ACSR_6_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR ACSR_6_6STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_NE_NE_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR AAAC_1/0_7STR] Reduce=Y

new Linegeometry.BT_1/0AAAC_CO_TR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_1/0_7STR AAAC_1/0_7STR CU_1/0_7STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_AAAC_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR AAAC_2_7STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_ACSR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR ACSR_1/0_6STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_ACSR_2/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR ACSR_2/0_6STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_ACSR_6_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR ACSR_6_6STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_NE_NE_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR ACSR_6_6STR] Reduce=Y

new Linegeometry.BT_1/0ACSR_CO_TR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_1/0_6STR ACSR_1/0_6STR CU_1/0_7STR] Reduce=Y

new Linegeometry.BT_1/0CO_SO_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_1/0_7STR CU_1/0_7STR ACSR_2_6STR] Reduce=Y

new Linegeometry.BT_1/0CO_TR_AAAC_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_1/0_7STR CU_1/0_7STR AAAC_2_7STR] Reduce=Y

new Linegeometry.BT_1/0CO_TR_ACSR_123.3_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_1/0_7STR CU_1/0_7STR CU_1/0_7STR] Reduce=Y

new Linegeometry.BT_1/0CO_TR_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_1/0_7STR CU_1/0_7STR ACSR_2_6STR] Reduce=Y

new Linegeometry.BT_123.3AAAC_ACSR_123.3_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_123.3_7STR AAAC_123.3_7STR AAAC_123.3_7STR] Reduce=Y

new Linegeometry.BT_123.3AAAC_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_123.3_7STR AAAC_123.3_7STR ACSR_2_6STR] Reduce=Y

new Linegeometry.BT_123.3CO_TR_CO_TR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_1/0_7STR CU_1/0_7STR CU_1/0_7STR] Reduce=Y

new Linegeometry.BT_2AAAC_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[AAAC_2_7STR AAAC_2_7STR ACSR_2_6STR] Reduce=Y

new Linegeometry.BT_2ACSR_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_2_6STR ACSR_2_6STR ACSR_2_6STR] Reduce=Y

new Linegeometry.BT_2ACSR_ACSR_6_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_2_6STR ACSR_2_6STR ACSR_6_6STR ] Reduce=Y

new Linegeometry.BT_2ACSR_NE_NE_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_2_6STR ACSR_2_6STR ACSR_6_6STR] Reduce=Y

new Linegeometry.BT_2CO_TR_CO_TR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_2_7STR CU_2_7STR CU_1/0_7STR] Reduce=Y

new Linegeometry.BT_2/0ACSR_ACSR_2/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_2/0_6STR ACSR_2/0_6STR ACSR_2/0_6STR] Reduce=Y

new Linegeometry.BT_6ACSR_ACSR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_6_6STR ACSR_6_6STR ACSR_1/0_6STR] Reduce=Y

new Linegeometry.BT_6ACSR_ACSR_2_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_6_6STR ACSR_6_6STR ACSR_2_6STR] Reduce=Y

new Linegeometry.BT_6ACSR_ACSR_6_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[ACSR_6_6STR ACSR_6_6STR ACSR_6_6STR] Reduce=Y

new Linegeometry.BT_6CO_TR_CO_TR_1/0_1 nconds=3 nphases=2 spacing=1F_3WIRE
 ~ wires=[CU_6_SLD CU_6_SLD CU_1/0_7STR] Reduce=Y

!!! Muestra resultados
Show lineconstants freq=60 units=m
```

Figura B.3: Cálculo de matrices de impedancia para líneas secundarias parte 2
Fuente: Elaboración propia.

Apéndice C

Archivo Maestro del circuito en OpenDSS

```
//  Este script de OpenDSS comienza limpiando cualquier configuracion anterior
clear
//crea un nuevo circuito
new circuit.circuito5
Set  DefaultBaseFrequency=60
//Redirige la salida del script a un archivo llamado V_source.dss
Redirect Vsource.dss
//Redirige la entrada del script a un archivo de lista de lineas de media tension (MV)
Redirect Lista_Lineas_MV.dss
//Crea una lista de buses en el sistema
MakeBusList
//Importa las coordenadas de los buses desde un archivo llamado BusCoords.dss
Buscoord BusCoords.dss
Redirect BusVoltageBases.dss

solve
show topology
```

Figura C.1: Código Master solo con las líneas primarias
Fuente: Elaboración propia.

```
//  Este script de OpenDSS comienza limpiando cualquier configuracion anterior
clear
//crea un nuevo circuito
new circuit.circuito5
Set  DefaultBaseFrequency=60
//Redirige la salida del script a un archivo llamado V_source.dss
Redirect Vsource.dss
//Redirige la entrada del script a un archivo de lista de lineas de media tension (MV)
Redirect Lista_Lineas_MV.dss
//Redirige la entrada del script a un archivo de transformadores
Redirect trafos.dss
//Crea una lista de buses en el sistema
MakeBusList
//Importa las coordenadas de los buses desde un archivo llamado BusCoords.dss
Buscoord BusCoords.dss
Redirect BusVoltageBases.dss

solve
show topology
```

Figura C.2: Código Master con las líneas primarias y los transformadores
Fuente: Elaboración propia.

```
//  Este script de OpenDSS comienza limpiando cualquier configuracion anterior
clear
//crea un nuevo circuito
new circuit.circuito5
Set  Default Base Frequency=60

//Redirige la salida del script a un archivo llamado V_source.dss
Redirect Vsource.dss
//Redirige la entrada del script a un archivo de lista de lineas de media tension (MV)
Redirect Lista_Lineas_MV.dss
//Redirige las geometrias de las lineas de baja tension secundarias
Redirect LineCodeSecundarias.dss
//Redirige las LineCode de las lineas de baja tension de las acometidas
Redirect LineCodeAcometidas.dss
//Redirige las LineCode de las lineas de baja tension de las subterraneas
Redirect LineCodeSubterraneas.dss
//Redirige la entrada del script a un archivo de transformadores
Redirect trafos.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) secundarias
Redirect Lista_Lineas_BV_secundarias.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) acometidas
Redirect Lista_Lineas_BV_acometidas.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) subterraneas
Redirect Lista_Lineas_BV_subterraneas.dss

//Crea una lista de buses en el sistema
MakeBusList
//Importa las coordenadas de los buses desde un archivo llamado BusCoords.dss
Buscoord BusCoords.dss
Redirect BusVoltageBases.dss

solve
show topology
```

Figura C.3: Código Master con las líneas primarias, transformadores y las líneas secundarias
Fuente: Elaboración propia.

```
//  Este script de OpenDSS comienza limpiando cualquier configuracion anterior
clear
//crea un nuevo circuito
new circuit.circuito5
Set  Default Base Frequency=60

//Redirige la salida del script a un archivo llamado V_source.dss
Redirect Vsource.dss
//Redirige la entrada del script a un archivo de lista de lineas de media tension (MV)
Redirect Lista_Lineas_MV.dss
//Redirige las geometrias de las lineas de baja tension secundarias
Redirect LineCodeSecundarias.dss
//Redirige las LineCode de las lineas de baja tension de las acometidas
Redirect LineCodeAcometidas.dss
//Redirige las LineCode de las lineas de baja tension de las subterraneas
Redirect LineCodeSubterraneas.dss
//Redirige la entrada del script a un archivo de transformadores
Redirect trafos.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) secundarias
Redirect Lista_Lineas_BV_secundarias.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) acometidas
Redirect Lista_Lineas_BV_acometidas.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) subterraneas
Redirect Lista_Lineas_BV_subterraneas.dss
//Redirige al archivo de cargas
Redirect cargas.dss

//Crea una lista de buses en el sistema
MakeBusList
//Importa las coordenadas de los buses desde un archivo llamado BusCoords.dss
Buscoord BusCoords.dss
Redirect BusVoltageBases.dss

New monitor.monitor element=Vsource.source Terminal=1 Mode=1 ppolar=no
set mode=daily stepsize=1m number=24
New "Energy Meter.sub" element=Line.445411__0 Terminal=1
solve
export monitor monitor
show topology
```

Figura C.4: Código Master con las líneas primarias, transformadores, líneas secundarias y cargas estáticas.
Fuente: Elaboración propia.

```
//  Este script de OpenDSS comienza limpiando cualquier configuracion anterior
clear
//crea un nuevo circuito
new circuit.circuito5
Set  Default Base Frequency=60
//Redirige la salida del script a un archivo llamado V_source.dss
Redirect Vsource.dss

//Redirige la entrada del script a un archivo de lista de lineas de media tension (MV)
Redirect Lista_Lineas_MV.dss
//Redirige las geometrias de las lineas de baja tension secundarias
Redirect LineCodeSecundarias.dss
//Redirige las LineCode  de las lineas de baja tension de las acometidas
Redirect LineCodeAcometidas.dss
//Redirige las LineCode  de las lineas de baja tension de las subterraneas
Redirect LineCodeSubterraneas.dss
//Redirige la entrada del script a un archivo de transformadores
Redirect trafos.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) secundarias
Redirect Lista_Lineas_BV_secundarias.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) acometidas
Redirect Lista_Lineas_BV_acometidas.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) subterraneas
Redirect Lista_Lineas_BV_subterraneas.dss
//Redirige al archivo de cargas
Redirect cargas.dss
Redirect 8011_R.dss
Redirect 8012_R.dss

//Crea una lista de buses en el sistema
MakeBusList
//Importa las coordenadas de los buses desde un archivo llamado BusCoords.dss
Buscoord BusCoords.dss
Redirect BusVoltageBases.dss

New monitor.monitor element=Vsource.source Terminal=1 Mode=1 ppolar=no
set mode=daily stepsize=1m number=24
New "Energy Meter.sub" element=Line.4454411__0 Terminal=1
solve
export monitor monitor
show topology
```

Figura C.5: Código Master con las líneas primarias, transformadores, líneas secundarias, cargas y reguladores de tensión

Fuente: Elaboración propia.

```
//  Este script de OpenDSS comienza limpiando cualquier configuracion anterior
clear
//crea un nuevo circuito
new circuit.circuito5
Set  Default Base Frequency =60
//Redirige la salida del script a un archivo llamado V_source.dss
Redirect Vsource.dss

//Redirige la entrada del script a un archivo de lista de lineas de media tension (MV)
Redirect Lista_Lineas_MV.dss
//Redirige las geometrias de las lineas de baja tension secundarias
Redirect LineCodeSecundarias.dss
//Redirige las LineCode  de las lineas de baja tension de las acometidas
Redirect LineCodeAcometidas.dss
//Redirige las LineCode  de las lineas de baja tension de las subterraneas
Redirect LineCodeSubterraneas.dss
//Redirige la entrada del script a un archivo de transformadores
Redirect trafos.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) secundarias
Redirect Lista_Lineas_BV_secundarias.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) acometidas
Redirect Lista_Lineas_BV_acometidas.dss
//Redirige la entrada del script a un archivo de lista de lineas de baja tension (BV) subterraneas
Redirect Lista_Lineas_BV_subterraneas.dss
New LoadShape.loadshape_1 npts =144 minterval=10 mult=(file=loadshape_1.txt) useactual=no New
LoadShape.loadshape_2 npts  =144 minterval=10 mult=(file=loadshape_2.txt) useactual=no New
LoadShape.loadshape_3 npts  =144 minterval=10 mult=(file=loadshape_3.txt) useactual=no New
LoadShape.loadshape_4 npts  =144 minterval=10 mult=(file=loadshape_4.txt) useactual=no New
LoadShape.loadshape_5 npts  =144 minterval=10 mult=(file=loadshape_5.txt) useactual=no New
LoadShape.loadshape_6 npts =144 minterval=10 mult=(file=loadshape_6.txt) useactual=no
//Redirige al archivo de cargas
Redirect cargasdinamicas_1.dss
//Redirect cargasdinamicas_2.dss
//Redirect cargasdinamicas_3.dss
//Redirect cargasdinamicas_4.dss
//Redirect cargasdinamicas_5.dss
//Redirect cargasdinamicas_6.dss

//Crea una lista de buses en el sistema
MakeBusList
//Importa las coordenadas de los buses desde un archivo llamado BusCoords.dss
Buscoord BusCoords.dss
Redirect BusVoltageBases.dss

New monitor.monitor element=Vsource.source Terminal=1 Mode =1 ppolar =no
set mode=daily stepsize=1m number=24
New "Energy Meter.sub" element=Line.445411___0 Terminal=1
solve
export monitor monitor
show topology
```

Figura C.6: Código Master con las líneas primarias, transformadores, líneas secundarias y cargas dinámicas.

Fuente: Elaboración propia.

Apéndice D

Código para unir las líneas primarias

```
'''
Autor : Keitlyn Valeria Montiel Arce
Fecha : 21/02/2024
Generador de archivos DSS
'''
import pandas as pd
import subprocess

# Leer el archivo Excel
excel_file = r'C:\\Users\\nunz1\\OneDrive\\Documentos\\Universidad\\2024\\TFG\\Circuito
\\Limpieza de datos de circuito\\Circuito_5Original.xlsx'
busbaroriginal_df = pd.read_excel(excel_file, sheet_name='Busbar')
lineas_asym_df = pd.read_excel(excel_file, sheet_name='LineAsym')
# Copiar el DataFrame original para no modificar los datos originales
busbar_df = busbaroriginal_df.copy()
busbar_df = busbar_df[busbar_df['Name'].str.startswith('MT')]
#  Redondear las columnas CoordX1 y CoordY1 a un decimal decimales
busbar_df['CoordX1'] = busbar_df['CoordX1'].round(4)
busbar_df['CoordY1'] = busbar_df['CoordY1'].round(4)

# Identificar filas duplicadas basadas en CoordX1, CoordY1 y Un
duplicados_primero = busbar_df.duplicated(subset=['CoordX1', 'CoordY1', 'Un'], keep='first')
duplicados_segundo = busbar_df.duplicated(subset=['CoordX1', 'CoordY1','Un'], keep='last')

# Crear DataFrame conservando el primer dato repetido y borrando el segundo
primer_dato_repetido = busbar_df[~duplicados_primero]
# Crear DataFrame borrando el primer dato y conservando el segundo
segundo_dato_repetido = busbar_df[~duplicados_segundo]

# Data Frame con unicamente los primeros datos repetidos
first_df = pd.merge(busbar_df, segundo_dato_repetido, indicator=True, how='left').query('_merge ==
"left_only"').drop('_merge', axis=1)
# Data Frame con unicamente los segundos datos repetidos
second_df = pd.merge(busbar_df, primer_dato_repetido, indicator=True, how='left').query('_merge ==
"left_only"').drop('_merge', axis=1)

# Crear un nuevo DataFrame para almacenar los resultados modificados de LineAsym
nuevo_lineas_asym_df = lineas_asym_df.copy()
# Iterar sobre las filas de  second_df
for index, row in second_df.iterrows():
    # Buscar coincidencias en Node1 de LineAsym
    match_row = lineas_asym_df[lineas_asym_df['Node1'] == row['Name']]
    if not match_row.empty:
        matching_first_df_row = first_df[(first_df['CoordX1'] == row['CoordX1']) & (first_df['CoordY1']
        == row['CoordY1'])]
        if not matching_first_df_row.empty:
            nuevo_lineas_asym_df.loc[nuevo_lineas_asym_df['Node1']
            == row['Name'], 'Node1'] = matching_first_df_row.iloc[0]['Name']
# Iterar sobre las filas de second_df para Node2
for index, row in second_df.iterrows():
    match_row = lineas_asym_df[lineas_asym_df['Node2'] == row['Name']]
    if not match_row.empty:
        matching_first_df_row = first_df[(first_df['CoordX1'] == row['CoordX1']) & (first_df['CoordY1']
        == row['CoordY1'])]
        if not matching_first_df_row.empty:
            nuevo_lineas_asym_df.loc[nuevo_lineas_asym_df['Node2']
            == row['Name'], 'Node2'] = matching_first_df_row.iloc[0]['Name']

# Guardar los cambios en un nuevo archivo Excel
output_file = 'Circuito5_actualizado.xlsx'
with pd.ExcelWriter(output_file) as writer:
    primer_dato_repetido.to_excel(writer, sheet_name='Busbar', index=False)
    nuevo_lineas_asym_df.to_excel(writer, sheet_name='LineAsym', index=False)
# Abrir automaticamente el archivo Excel
subprocess.run(['start', output_file], shell=True)
```

Figura D.1: Código LimpiezaDatos.py para limpieza de datos
Fuente: Elaboración propia.

Apéndice E

Código para generar un archivo de librería de transformadores

```
'''
Autor : Keitlyn Valeria Montiel Arce
Fecha : 17/03/2024
Generador de archivos de librerias de transformadores
'''
import pandas as pd

# Funcion para asignar los valores correspondientes a las columnas Xhl, Xht y Xlt
def asignar_valores(library_type):
    if '10.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.34, 'Xht': 5.34, 'Xlt': 3.56, '%Rs': '[0.98 1.95 1.95]',
        '%noloadloss': 0.50, '%imag': 1, 'kVAs': '[10 10 10]', 'kvs': '[19.92 0.12 0.12]'}
    elif '15.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.52, 'Xht': 5.52, 'Xlt': 3.68, '%Rs': '[0.92 1.83 1.83]',
        '%noloadloss': 0.47, '%imag': 1, 'kVAs': '[15 15 15]', 'kvs': '[19.92 0.12 0.12]'}
    elif '25.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.65, 'Xht': 5.65, 'Xlt': 3.77, '%Rs': '[0.81 1.62 1.62]',
        '%noloadloss': 0.36, '%imag': 1, 'kVAs': '[25 25 25]', 'kvs': '[19.92 0.12 0.12]'}
    elif '37.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.8, 'Xht': 5.8, 'Xlt': 3.86, '%Rs': '[0.73 1.45 1.45]',
        '%noloadloss': 0.35, '%imag': 1, 'kVAs': '[37 37 37]', 'kvs': '[19.92 0.12 0.12]'}
    elif '50.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.89, 'Xht': 5.89, 'Xlt': 3.93, '%Rs': '[0.68 1.36 1.36]',
        '%noloadloss': 0.34, '%imag': 1, 'kVAs': '[50 50 50]', 'kvs': '[19.92 0.12 0.12]'}
    elif '75.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.94, 'Xht': 5.94, 'Xlt': 3.96, '%Rs': '[0.61 1.21 1.21]',
        '%noloadloss': 0.27, '%imag': 1, 'kVAs': '[75 75 75]', 'kvs': '[19.92 0.12 0.12]'}
    elif '100.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.98, 'Xht': 5.98, 'Xlt': 3.98, '%Rs': '[0.56 1.12 1.12]',
        '%noloadloss': 0.25, '%imag': 1, 'kVAs': '[100 100 100]', 'kvs': '[19.92 0.12 0.12]'}
    elif '167.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.04, 'Xht': 6.04, 'Xlt': 4.02, '%Rs': '[0.50 1.00 1.00]',
        '%noloadloss': 0.24, '%imag': 1, 'kVAs': '[167 167 167]', 'kvs': '[19.92 0.12 0.12]'}
    elif '250.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.07, 'Xht': 6.07, 'Xlt': 4.05, '%Rs': '[0.50 1.00 1.00]',
        '%noloadloss': 0.23, '%imag': 1, 'kVAs': '[250 250 250]', 'kvs': '[19.92 0.12 0.12]'}
    elif '333.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.1, 'Xht': 6.1, 'Xlt': 4.06, '%Rs': '[0.50 1.00 1.00]',
        '%noloadloss': 0.23, '%imag': 1, 'kVAs': '[333 333 333]', 'kvs': '[19.92 0.12 0.12]'}
    elif '500.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.12, 'Xht': 6.12, 'Xlt': 4.08, '%Rs': '[0.50 1.00 1.00]',
        '%noloadloss': 0.22, '%imag': 1, 'kVAs': '[500 500 500]', 'kvs': '[19.92 0.12 0.12]'}
    elif '667.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.12, 'Xht': 6.12, 'Xlt': 4.08, '%Rs': '[0.50 1.00 1.00]',
        '%noloadloss': 0.22, '%imag': 1, 'kVAs': '[667 667 667]', 'kvs': '[19.92 0.12 0.12]'}
    elif '833.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.12, 'Xht': 6.12, 'Xlt': 4.08, '%Rs': '[0.50 1.00 1.00]',
        '%noloadloss': 0.22, '%imag': 1, 'kVAs': '[833 833 833]', 'kvs': '[19.92 0.12 0.12]'}
    else:
        return {}

# Ruta del archivo Excel
file_path = r'C:\Users\nunzi\OneDrive\Documentos\Universidad\2024\TFG\Circuito\Circuito_5.xlsx'
df = pd.read_excel(file_path, 'Trafo2WindingAsym')
#Lista para almacenar las nuevas filas
new_rows = []
```

Figura E.1: Código para crear un archivo de Librería de transformadores parte 1.
Fuente: Elaboración propia.

```
# Iteracion sobre las filas del DataFrame original
for index, row in df.iterrows():
    library_type = row['LibraryType']
    values = asignar_valores(library_type)
   # Verificar si library_type comienza con 'A_', 'B_' o 'C_'
    if library_type.startswith(('A_', 'B_', 'C_')):
        values = asignar_valores(library_type)
        if values:
            new_rows.append({
                'LibraryType': row['LibraryType'],
                'Xhl': values['Xhl'],
                'Xht': values['Xht'],
                'Xlt': values['Xlt'],
                '%Rs': values['%Rs'],
                '%noloadloss': values['%noloadloss'],
                '%imag': values['%imag'],
                'kvs':  values['kvs'],
                'kVAs': values['kVAs'],
            })

# Crear un nuevo DataFrame a partir de la lista de diccionarios
new_df = pd.DataFrame(new_rows)
# Funcion para asignar los valores correspondientes a las columnas Xhl, Xht y Xlt
def asignar_valores2(library_type):
    if '75.0_34.5_kV_.480_kV' in library_type:
        return {'Xhl': 5.11 , '%Rs': '[0.77 0.77]' , '%noloadloss': 0.36, '%imag':
        1, 'kVAs': '[75 75]', 'kvs': '[34.5 0.48]'}
    elif '150.0_34.5_kV_.240' in library_type:
        return {'Xhl': 5.24, '%Rs': '[0.64 0.64]' , '%noloadloss': 0.30, '%imag':
        1, 'kVAs': '[150 150]', 'kvs': '[34.5 0.24]'}
    elif '225.0_34.5_kV_.208_kV' in library_type:
        return {'Xhl': 5.30, '%Rs': '[0.58 0.58]' , '%noloadloss': 0.28, '%imag':
        1, 'kVAs': '[225 225]', 'kvs': '[34.5 0.208]'}
    elif '225.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.30, '%Rs': '[0.58 0.58]' , '%noloadloss': 0.28, '%imag':
        1, 'kVAs': '[225 225]', 'kvs': '[34.5 0.24]'}
    elif '225.0_34.5_kV_.480_kV' in library_type:
        return {'Xhl': 5.30, '%Rs': '[0.58 0.58]' , '%noloadloss': 0.28, '%imag':
        1, 'kVAs': '[225 225]', 'kvs': '[34.5 0.48]'}
    elif '300.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.32, '%Rs': '[0.54 0.54]' , '%noloadloss': 0.25, '%imag':
        1, 'kVAs': '[300 300]', 'kvs': '[34.5 0.24]'}
    elif '501.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.37, '%Rs': '[0.52 0.52]' , '%noloadloss': 0.24, '%imag':
        1, 'kVAs': '[500 500]', 'kvs': '[34.5 0.24]'}
    elif '501.0_34.5_kV_.480_kV' in library_type:
        return {'Xhl': 5.37, '%Rs': '[0.52 0.52]' , '%noloadloss': 0.24, '%imag':
        1, 'kVAs': '[500 500]', 'kvs': '[34.5 0.48]'}
    elif '750.0_34.5_kV_.480_kV' in library_type:
        return {'Xhl': 5.40, '%Rs': '[0.518 0.518]' , '%noloadloss': 0.23, '%imag':
        1, 'kVAs': '[750 750]', 'kvs': '[34.5 0.48]'}
    else:
        return {}
df2 = pd.read_excel(file_path, 'Trafo2Winding')
#Lista para almacenar las nuevas filas
new_rows2 = []

#Iteracion sobre las filas del DataFrame original
for index2, row2 in df2.iterrows():
    library_type = row2['LibraryType']
    values2 = asignar_valores2(library_type)
    if values2:
            new_rows2.append({
            'Library Type': row2['Library Type'],
            'Xhl': values2['Xhl'],
            '%Rs': values2['%Rs'],
            '%noloadloss': values2['%noloadloss'],
            '%imag': values2['%imag'],
            'kvs':  values2['kvs'],
            'kVAs': values2['kVAs'],
        })
#Crear un nuevo DataFrame a partir de la lista de diccionarios
new_df2 = pd.DataFrame(new_rows2)
```

Figura E.2: Código para crear un archivo de Librería de transformadores parte 2.
Fuente: Elaboración propia.

```
#Funcion  para asignar los valores correspondientes a las columnas Xhl, Xht y Xlt
def asignar_valores3(library_type):
    if '10.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.34, 'Xht': 5.34, 'Xlt': 3.56, '%Rs': '[0.98 1.95]',
        '%noloadloss': 0.50, '%imag': 1, 'kVAs': '[10 10]', 'kvs': '[19.92 0.24]'}
    elif '15.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.52, 'Xht': 5.52, 'Xlt': 3.68, '%Rs': '[0.92 1.83]',
        '%noloadloss': 0.47, '%imag': 1, 'kVAs': '[15 15]', 'kvs': '[19.92 0.24]'}
    elif '25.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.65, 'Xht': 5.65, 'Xlt': 3.77, '%Rs': '[0.81 1.62]',
        '%noloadloss': 0.36, '%imag': 1, 'kVAs': '[25 25]', 'kvs': '[19.92 0.24]'}
    elif '37.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.8, 'Xht': 5.8, 'Xlt': 3.86, '%Rs': '[0.73 1.45]',
        '%noloadloss': 0.35, '%imag': 1, 'kVAs': '[37 37]', 'kvs': '[19.92 0.24]'}
    elif '50.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.89, 'Xht': 5.89, 'Xlt': 3.93, '%Rs': '[0.68 1.36]',
        '%noloadloss': 0.34, '%imag': 1, 'kVAs': '[50 50]', 'kvs': '[19.92 0.24]'}
    elif '75.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.94, 'Xht': 5.94, 'Xlt': 3.96, '%Rs': '[0.61 1.21]',
        '%noloadloss': 0.27, '%imag': 1, 'kVAs': '[75 75]', 'kvs': '[19.92 0.24]'}
    elif '100.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 5.98, 'Xht': 5.98, 'Xlt': 3.98, '%Rs': '[0.56 1.12]',
        '%noloadloss': 0.25, '%imag': 1, 'kVAs': '[100 100]', 'kvs': '[19.92 0.24]'}
    elif '167.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.04, 'Xht': 6.04, 'Xlt': 4.02, '%Rs': '[0.50 1.00]',
        '%noloadloss': 0.24, '%imag': 1, 'kVAs': '[167 167]', 'kvs': '[19.92 0.24]'}
    elif '250.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.07, 'Xht': 6.07, 'Xlt': 4.05, '%Rs': '[0.50 1.00]',
        '%noloadloss': 0.23, '%imag': 1, 'kVAs': '[250 250]', 'kvs': '[19.92 0.24]'}
    elif '333.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.1, 'Xht': 6.1, 'Xlt': 4.06, '%Rs': '[0.50 1.00]',
        '%noloadloss': 0.23, '%imag': 1, 'kVAs': '[333 333]', 'kvs': '[19.92 0.24]'}
    elif '500.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.12, 'Xht': 6.12, 'Xlt': 4.08, '%Rs': '[0.50 1.00]',
        '%noloadloss': 0.22, '%imag': 1, 'kVAs': '[500 500]', 'kvs': '[19.92 0.24]'}
    elif '667.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.12, 'Xht': 6.12, 'Xlt': 4.08, '%Rs': '[0.50 1.00]',
        '%noloadloss': 0.22, '%imag': 1, 'kVAs': '[667 667]', 'kvs': '[19.92 0.24]'}
    elif '833.0_34.5_kV_.240_kV' in library_type:
        return {'Xhl': 6.12, 'Xht': 6.12, 'Xlt': 4.08, '%Rs': '[0.50 1.00]',
        '%noloadloss': 0.22, '%imag': 1, 'kVAs': '[833 833]', 'kvs': '[19.92 0.24]'}

new_rows3 = []
# Iteracion sobre las filas del DataFrame original
for index3, row3 in df.iterrows():
    library_type = row3['LibraryType']
    values3 = asignar_valores3(library_type)
   # Verificar si library_type comienza con 'AB_', 'BC_' o 'AC_'
    if library_type.startswith(('AB_', 'BC_', 'AC_')):
        values3 = asignar_valores3(library_type)
        if values:
            new_rows3.append({
                'Library Type': row3['Library Type'],
                'Xhl': values['Xhl'],
                'Xht': values['Xht'],
                'Xlt': values['Xlt'],
                '%Rs': values['%Rs'],
                '%noloadloss': values['%noloadloss'],
                '%imag': values['%imag'],
                'kvs':  values['kvs'],
                'kVAs': values['kVAs'],
            })
# Crear un nuevo DataFrame a partir de la lista de diccionarios
new_df3 = pd.DataFrame(new_rows3)
# Guardar el nuevo DataFrame en un nuevo archivo Excel en la hoja "monofasicos"
with pd.ExcelWriter('LibreriaTrafos.xlsx', engine='xlsxwriter') as writer:
    new_df.to_excel(writer, index=False, sheet_name='monofasicos')
    new_df3.to_excel(writer, index=False, sheet_name='bifasicos')
    new_df2.to_excel(writer, index=False, sheet_name='trifasicos')
```

Figura E.3: Código para crear un archivo de Libreria de transformadores parte 3.

Apéndice F

Código para unir las líneas secundarias

```
'''
Autor: Keitlyn Valeria Montiel Arce
Fecha: 20/03/2024
Unir lineas aisladas
'''
import pandas as pd
from scipy.spatial import distance
import subprocess

# Ruta del archivo Excel
excel_file = r'C:\Users\nunz1\OneDrive\Documentos\Universidad\2024\TFG\Circuito\Circuito_5.xlsx'

# Leer el archivo Excel
busbar_df = pd.read_excel(excel_file, sheet_name='    Busbar    ')
lineas_df_original = pd.read_excel(excel_file, sheet_name='LineAsym')
df_trafo3 = pd.read_excel(excel_file, sheet_name='Trafo2Winding')
df_busbar = pd.read_excel(excel_file, sheet_name='Busbar')
df_trafo = pd.read_excel(excel_file, sheet_name='Trafo2WindingAsym ')

lineas_df = lineas_df_original.copy()

# Ruta del archivo de texto
archivo_texto = r'C:\Users\nunz1\OneDrive\Documentos\Universidad\2024\TFG\Circuito\circuito5_TopoTree.txt'

# Leer el archivo de texto
with open(archivo_texto, 'r') as file:
    lines = file.readlines()

# Filtrar las lineas que contienen la palabra "Isolated"
isolated_lines = [line.strip().replace('Isolated: Line.', '') for line in lines if 'Isolated' in line]

# Crear un dataframe a partir de las lineas filtradas
aisladas_df = pd.DataFrame(isolated_lines, columns=['Isolated Lines'])

# Filtrar las lineas que no contienen la palabra "Isolated" y eliminar el prefijo "   Line   .   "
non_isolated_lines = [line.strip().replace('Line.', '') for line in lines if 'Isolated' not in line]

# Crear un DataFrame a partir de las lineas filtradas
sin_aisladas_df = pd.DataFrame(non_isolated_lines, columns=['Non Isolated Lines'])

# Crear DataFrame filtrando las lineas aisladas en LineAsym
lineas_aisladas_df = lineas_df.merge(aisladas_df, how='inner', left_on='Name', right_on='Isolated Lines')

# Crear DataFrame filtrando las lineas no aisladas en LineAsym
lineas_sin_aisladas_df = lineas_df.merge(sin_aisladas_df, how='inner', left_on='Name',
right_on='Non Isolated Lines')
lineas_sin_aisladas_df = lineas_sin_aisladas_df[lineas_sin_aisladas_df['Node2'].str.startswith('BT')]

lineas_sin_aisladas_df = pd.concat([lineas_sin_aisladas_df, df_trafo3, df_trafo])

# Asociar las coordenadas de los nodos en lineas_aisladas_df y lineas_sin_aisladas_df
def associate_coordinates(df, busbar_df):
    df = df.merge(busbar_df[['Name', 'CoordX1', 'CoordY1']], how='left', left_on='Node1', right_on='Name')
    df = df.rename(columns={'CoordX1': 'CoordX1_Node1', 'CoordY1': 'CoordY1_Node1'})
    df = df.merge(busbar_df[['Name', 'CoordX1', 'CoordY1']], how='left', left_on='Node2', right_on='Name')
    df = df.rename(columns={'CoordX1': 'CoordX1_Node2', 'CoordY1': 'CoordY1_Node2'})
    return df

lineas_aisladas_df = associate_coordinates(lineas_aisladas_df, busbar_df)
lineas_sin_aisladas_df = associate_coordinates(lineas_sin_aisladas_df, busbar_df)

# Crear una lista para almacenar los nodos reemplazados y los nodos por los que se reemplazaron
nodos_reemplazados_list = []
```

Figura F.1: Código para unir las líneas de baja tensión parte 1
Fuente: Elaboración propia.

```
# Verificar si lineas_aisladas_df no esta vacio antes de iterar
while not lineas_aisladas_df.empty:
    nodo_reemplazado = False  # Variable para verificar si se realizo algun reemplazo en este ciclo

    # Iterar sobre las filas de lineas_aisladas_df
    for index, row in lineas_aisladas_df.iterrows():
        node1_present = row['Node1'] in lineas_sin_aisladas_df['Node1'].values or row['Node1']
        in lineas_sin_aisladas_df['Node2'].values
        node2_present = row['Node2'] in lineas_sin_aisladas_df['Node1'].values or row['Node2']
        in lineas_sin_aisladas_df['Node2'].values

        # Si ninguno de los nodos esta presente en lineas_sin_aisladas_df
        if not (node1_present or node2_present):
            # Coordenadas de Node1 de la linea actual
            coordX_Node1 = row['CoordX1_Node1']
            coordY_Node1 = row['CoordY1_Node1']

            # Calcular las distancias a los nodos en lineas_sin_aisladas_df
            distances_node2 =
            lineas_sin_aisladas_df.apply(lambda x: distance.euclidean((coordX_Node1, coordY_Node1),
            (x['CoordX1_Node2'], x['CoordY1_Node2'])), axis=1)

            # Encontrar el indice del nodo mas cercano en lineas_sin_aisladas_df
            closest_node_index = distances_node2.idxmin()

            #  Obtener el nodo mas cercano y reemplazar Node1 (asegurandonos de que no sea igual a Node1)
            closest_node = lineas_sin_aisladas_df.loc[closest_node_index, 'Node2']
            if closest_node != row['Node1']:
                # Agregar los nodos reemplazados y los nodos por los que se reemplazaron a la lista
                nodos_reemplazados_list
                nodos_reemplazados_list.append({'Node1': row['Node1'], 'closest_node': closest_node})

                lineas_aisladas_df.at[index, 'Node2'] = closest_node

                # Concatenar la fila correspondiente a lineas_sin_aisladas_df
                lineas_sin_aisladas_df = pd.concat([lineas_sin_aisladas_df, row.to_frame().T],
                ignore_index=True)

                # Eliminar la fila procesada de lineas_aisladas_df
                lineas_aisladas_df.drop(index, inplace=True)

                nodo_reemplazado = True  # Se realizo un reemplazo en este ciclo
                print(f"Nodo reemplazado: {row['Node1']} por {closest_node}")
                 # Salir del ciclo for para reiniciar el ciclo while
                break
    # Si no se realizo ningun reemplazo en este ciclo, salimos del bucle while
    if not nodo_reemplazado:
        break

# Convertir la lista de nodos reemplazados a un DataFrame
nodos_reemplazados_df = pd.DataFrame(nodos_reemplazados_list)

# Mostrar los nodos reemplazados y los nodos por los que se reemplazaron
print("Nodos reemplazados y nodos por los que se reemplazaron:")
print(nodos_reemplazados_df)

# Iterar sobre los elementos de nodos_reemplazados_list
for node_reemplazado_info in nodos_reemplazados_list:
    #  Obtener el nodo reemplazado y el nodo por el que se reemplazo en este elemento
    node_reemplazado = node_reemplazado_info['Node1']
    closest_node = node_reemplazado_info['closest_node']

    # Buscar las casillas en lineas_df_original['Node1'] que coincidan con node_reemplazado
    matching_indices = lineas_df_original[lineas_df_original['Node1'] == node_reemplazado].index

    # Reemplazar los valores en las casillas correspondientes con el valor de closest_node
    lineas_df_original.loc[matching_indices, 'Node1'] = closest_node

#  Guardar los cambios en un nuevo archivo Excel
output_file = 'Circuito5_actualizado2.xlsx'
with pd.ExcelWriter(output_file) as writer:
    busbar_df.to_excel(writer, sheet_name='Busbar', index=False)
    lineas_df_original.to_excel(writer, sheet_name='LineAsym', index=False)

print("Archivo guardado correctamente:", output_file)

subprocess.run(['start', output_file], shell=True)
```

Figura F.2: Código para unir las líneas de baja tensión parte 2
Fuente: Elaboración propia.

Apéndice G

Definición en OpenDSS de los reguladores de tensión

```
\\REGULADOR TIPO A
set maxcontroliter=30

\\Jumpers de entrada (baja impedancia)
New Reactor.Jumper_Reg1_E1 phases=1 bus1=MT_AREA_435112_952647.1 bus2=MT_AREA_435110_859936_1.1 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg1_E2 phases=1 bus1=MT_AREA_435112_952647.2 bus2=MT_AREA_435110_859936_2.1 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg1_E3 phases=1 bus1=MT_AREA_435112_952647.3 bus2=MT_AREA_435110_859936_3.1 X=0.0001 R=0.0001

\\Unidades transformadoras
New Transformer.8011_T_A phases=1 windings=2 bank=reg0 xhl=0.01 %loadloss=0.1
  ~ wdg=1 Bus=MT_AREA_435110_859936_1.1.0 kV=19.9186 kVA=90.91
  ~ wdg=2 Bus=MT_AREA_435110_859936_1.2.1 kV=1.99186 kVA=90.91 Maxtap=1.0 Mintap=-1.0 tap=0 numtaps=32
New Transformer.8011_T_B phases=1 windings=2 bank=reg0 xhl=0.01 %loadloss=0.1
  ~ wdg=1 Bus=MT_AREA_435110_859936_2.1.0 kV=19.9186 kVA=90.91
  ~ wdg=2 Bus=MT_AREA_435110_859936_2.2.1 kV=1.99186 kVA=90.91 Maxtap=1.0 Mintap=-1.0 tap=0 numtaps=32
New Transformer.8011_T_C phases=1 windings=2 bank=reg0 xhl=0.01 %loadloss=0.1
  ~ wdg=1 Bus=MT_AREA_435110_859936_3.1.0 kV=19.9186 kVA=90.91
  ~ wdg=2 Bus=MT_AREA_435110_859936_3.2.1 kV=1.99186 kVA=90.91 Maxtap=1.0 Mintap=-1.0 tap=0 numtaps=32

\\Jumpers de salida
New Reactor.Jumper_Reg1_S1 phases=1 bus1=MT_AREA_435110_859936_1.2 bus2=MT_AREA_435110_859936.1 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg1_S2 phases=1 bus1=MT_AREA_435110_859936_2.2 bus2=MT_AREA_435110_859936.2 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg1_S3 phases=1 bus1=MT_AREA_435110_859936_3.2 bus2=MT_AREA_435110_859936.3 X=0.0001 R=0.0001

\\Controladores del regulador
New regcontrol.8011_R_A transformer=8011_T_A winding=2 bus=MT_AREA_435110_859936_1.2 vreg=124.0 band=2.0
ptratio=166.0 maxtapchange=1
New regcontrol.8011_R_B transformer=8011_T_B winding=2 bus=MT_AREA_435110_859936_2.2 vreg=124.0 band=2.0
ptratio=166.0 maxtapchange=1
New regcontrol.8011_R_C transformer=8011_T_C winding=2 bus=MT_AREA_435110_859936_3.2 vreg=124.0 band=2.0
ptratio=166.0 maxtapchange=1
```

Figura G.1: Código para la definición del regulador de tensión 8011_R
Fuente: Elaboración propia.

```
\\ REGULADOR TIPO A
set maxcontroliter=30

\\ Jumpers de entrada (baja impedancia)
New Reactor.Jumper_Reg2_E1 phases=1 bus1=MT_AREA_431474_646134.1 bus2=MT_AREA_431407_6761_1.1 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg2_E2 phases=1 bus1=MT_AREA_431474_646134.2 bus2=MT_AREA_431407_6761_2.1 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg2_E3 phases=1 bus1=MT_AREA_431474_646134.3 bus2=MT_AREA_431407_6761_3.1 X=0.0001 R=0.0001

\\ Unidades transformadoras
New Transformer.8012_T_A phases=1 windings=2 bank=reg0 xhl=0.01 %loadloss=0.1
  ~ wdg=1 Bus=MT_AREA_431407_6761_1.1.0 kV=19.9186 kVA=90.91
  ~ wdg=2 Bus=MT_AREA_431407_6761_1.2.1 kV=1.99186 kVA=90.91 Maxtap=1.0 Mintap=-1.0 tap=0 numtaps=32
New Transformer.8012_T_B phases=1 windings=2 bank=reg0 xhl=0.01 %loadloss=0.1
  ~ wdg=1 Bus=MT_AREA_431407_6761_2.1.0 kV=19.9186 kVA=90.91
  ~ wdg=2 Bus=MT_AREA_431407_6761_2.2.1 kV=1.99186 kVA=90.91 Maxtap=1.0 Mintap=-1.0 tap=0 numtaps=32
New Transformer.8012_T_C phases=1 windings=2 bank=reg0 xhl=0.01 %loadloss=0.1
  ~ wdg=1 Bus=MT_AREA_431407_6761_3.1.0 kV=19.9186 kVA=90.91
  ~ wdg=2 Bus=MT_AREA_431407_6761_3.2.1 kV=1.99186 kVA=90.91 Maxtap=1.0 Mintap=-1.0 tap=0 numtaps=32

\\ Jumpers de salida
New Reactor.Jumper_Reg2_S1 phases=1 bus1=MT_AREA_431407_6761_1.2 bus2=MT_AREA_431407_6761.1 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg2_S2 phases=1 bus1=MT_AREA_431407_6761_2.2 bus2=MT_AREA_431407_6761.2 X=0.0001 R=0.0001 New
Reactor.Jumper_Reg2_S3 phases=1 bus1=MT_AREA_431407_6761_3.2 bus2=MT_AREA_431407_6761.3 X=0.0001 R=0.0001

\\ Controladores del regulador
New regcontrol.8012_R_A transformer=8012_T_A winding=2 bus=MT_AREA_431407_6761_1.2 vreg=124.0 band=2.0
ptratio=166.0 maxtapchange=1
New regcontrol.8012_R_B transformer=8012_T_B winding=2 bus=MT_AREA_431407_6761_2.2 vreg=124.0 band=2.0
ptratio=166.0 maxtapchange=1
New regcontrol.8012_R_C transformer=8012_T_C winding=2 bus=MT_AREA_431407_6761_3.2 vreg=124.0 band=2.0
ptratio=166.0 maxtapchange=1
```

Figura G.2: Código para la definición del regulador de tensión 8012_R
Fuente: Elaboración propia.