UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE TECNOLOGIA E INGENIERÍA ELÉCTRICA

APLICACIÓN DEL ESTANDAR IEC-61970 AL SEN Y ANALISIS DE FLUJO DE CARGA EN SOFTWARE ABIERTO.

Por:

Silvana E. Delgado García

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
Como requisito parcial para optar al título de
Ingeniero Electricista

Sartenejas, Noviembre de 2013

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE TECNOLOGIA E INGENIERÍA ELÉCTRICA

APLICACIÓN DEL ESTANDAR IEC-61970 AL SEN Y ANALISIS DE FLUJO DE CARGA EN SOFTWARE ABIERTO

Por:

Silvana E. Delgado García

Realizado con la asesoría de:

Tutor Académico: Paulo De Oliveira
Tutor Industrial: Paulo De Oliveira

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
Como requisito parcial para optar al título de
Ingeniero Electricista

Sartenejas, Noviembre de 2013.

Por:
Silvana Elisa Delgado García.

RESUMEN

Desde mediados del siglo XX en Venezuela habían existido numerosas compañías eléctricas, cada una de ellas encargada de un sector en particular del sistema eléctrico nacional, sin embargo el gobierno nacional decidió integrar todas estas empresas para convertirlas en una gran corporación que estuviera al mando de toda la red eléctrica. Debido a que cada compañía funcionaba de manera independiente, los formatos de las bases de datos eran completamente diferentes, lo que representaba una gran limitación técnica a la hora de la fusión.

Como consecuencia de los inconvenientes a la hora de compartir los datos de las redes eléctricas entre los diferentes departamentos que constituyen una empresa de servicio eléctrico, o incluso entre varias compañías (como fue el caso de Venezuela), nace el estándar CIM (llamado así por sus siglas en siglas de Modelo de Información Común) que se certifica con la norma 61970 de la IEC (Comisión Electrotécnica Internacional). Este modelo constituye una pauta internacional para estructurar archivos de datos de sistemas eléctricos, los cuales son imprescindibles a la hora de planificar, mantener, operar y analizar cualquier red eléctrica.

Este proyecto de grado se enfoca en su implementación a una parte del sistema eléctrico nacional, lo que representaría la integración de los datos del sistema a los estándares internacionales. Además de esto se enfatiza la necesidad de fomentar la utilización de herramientas de software libre a través de la ejecución de un flujo de carga clásico a la red de 765kV del sistema eléctrico nacional. Dicha herramienta está programada en Python y que evidencia las ventajas de los programas de código abierto por sobre las de los privativos.

DEDICATORIA

A Dios y a la Santísima Virgen.

A mi familia.

A mi nuevo ángel de la guarda.

AGRADECIMIENTOS

En primer lugar agradezco a Dios por preservarme de todo mal y darme la fuerza para seguir adelante en cada situación. Que su presencia nunca me abandone y me dé la sabiduría para tomar las decisiones correctas en la vida.

Le doy gracias a mi madre, por su apoyo constante y su comprensión, por enseñarme que en la vida todo tiene solución y que lo peor que puede pasar generalmente no es tan malo. A mi padre y hermano por su preocupación silenciosa y sus lecciones de vida. A mi madrina, por su cariño y ayuda incondicional. Los amo.

A mi familia por elección, a quienes adoro con toda el alma y sin los cuales la vida universitaria no hubiese sido ni la mitad de buena; Francisco Ochoa, Yanilu Teneúd, Andreina Domingos y Yessica de Ascencao. Gracias por estar conmigo en las buenas y las malas.

A los que, más que compañeros se volvieron mis amigos después de la quinta hora de estudio consecutivo, Manuel Ruiz, Migdalys Bueno, Génesis Delnardo, Gabriel Russo, Mayerlin Márquez, Jean Nava. Los quiero mucho y gracias por todo.

Gracias especiales a mi tutor, el Ingeniero Paulo De Oliveira por su guía y apoyo durante la elaboración de este proyecto.

Finalmente, pero no por ello menos importante, a quien me brindo sus consejos y su amor de madre cuando más lo necesité, más que una profesora, una gran amiga. Gracias a mi queridísima Paloma Lobo (en paz descanse), siempre estarás en mi corazón. Nunca olvidaré nuestras tardes de café.

Por todo esto y más, a todos

¡Muchas Gracias!

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

LISTA DE ABREVIATURAS

AC	Alternate Currente "Corriente Alterna"
CADAFE	Compañia Anónima de Administración y Fomento Eléctrico
CDF	Common Data Format "Formato de Datos Común"
CIM	Common information model "Modelo de Información Común"
CORPOELEC	Corporación Eléctrica Nacional
CPF	Continuación de flujo de carga y/o análisis de estabilidad de voltaje
DC	Direct Current "Corriente Directa"
DIgSILENT	Digital Simulator and Electrical Network "Simulador digital y red eléctrica"
DTD	Document Type Definition "Definición de tipos del documento"
EA	Análisis de estabilidad de pequeña señal
EDC	Electricidad de Caracas
EDELCA	Electrificación del Caroní
EMT	Transitorio electromagnético
ENELBAR	Energía Eléctrica de Barquisimeto
ENELVEN	Energía Eléctrica de Venezuela
EPRI	Electric Power Research Institute "Instituto de Investigación para la Energía Eléctrica"
GUI	Interfaz Grafica de Usuario
HTML	HyperText Markup Language "Lenguaje de Etiquetas de hipertexto"
IEC	Comisión Electrotécnica Internacional
IEEE	Instituto de Ingenieros Eléctricos y Electricistas
InterPSS	Simulador de Sistemas de Potencia basado en tecnología de Internet
M.I.T.	Instituto de tecnología de Massachusetts
MATLAB	Matrix Laboratory "Laboratorio de Matrices"
MPPEE	Ministerio del Poder Popular para la Energía Eléctrica
ODM	Open Data Model "Modelos de datos abierto"
OOP	Object Oriented Programming
OPF	Optimal Power Flow
PF	Power Flow "Flujo de carga"
PQ	Tipo de barra a la que se le conoce la potencia y reactiva consumida
PSAT	Power System Analysis Toolbox "Caja de análisis de sistemas potencia"
PV	Tipo de barra con módulo de voltaje y potencia generada y consumida conocidos
SEN	Sistema Eléctrico Nacional
SGML	Standard Generalized Markup Language "Lenguaje de etiquetas estándar generalizado"
TC57	IEC Technical Committee
Tol	Tolerancia
UML	Unified Modelling Language "Lenguaje de modelación unificado"
UWPFLOW	Flujo de Carga de la Universidad de Waterloo
XML	eXtensible Markup Language "Lenguaje de Etiquetas Extensible"

LISTA DE SIMBOLOS

%.	Porcentaje
Hz.	Hertz
θ	Angulo del voltaje
G	Conductancia
Δ	Diferencial.Puede ser de potencia activa o reactiva
Y	elemento de la matriz de admitancias o la matriz en sí según el caso
Z.	Impedancia
kA.	Kilo Amper
kV.	Kilo Voltio
km.	Kilómetros
MW	Mega Vatio
MVA.	Mega Voltio Amper
Mvar.	Mega Voltio Amper Reactivo
Ω.	Ohmio
pu.	Por Unidad
P	Potencia activa
Pd	Potencia activa demandada
Pg	Potencia activa generada
P_{neta}	Potencia activa neta de una barra
Q	Potencia reactiva
Qd	Potencia reactiva demandada
Qg	Potencia reactiva generada
Q_{neta}	Potencia reactiva neta en una barra
B	Susceptancia
X_0	Vector de partida para los procesos iterativos.

INTRODUCCIÓN

En la actualidad la planificación, diseño, implantación, operación y mantenimiento de un sistema eléctrico, viene obligatoriamente vinculado con algún tipo de herramienta computacional, ya sea un software comercial o uno de código abierto. Las implicaciones que esto trae consigo van mucho más allá a simplemente utilizar el programa con un propósito, sino que con el tiempo han hecho de la programación y de los conocimientos computacionales partes imprescindible en la formación de los ingenieros electricistas. Bajo este mismo orden de ideas se desemboca en el tema de las bases de datos y los formatos de archivos de los programa de análisis de sistemas de potencia.

Con el fin de facilitar el manejo de la información del sistema y el intercambio de los datos del mismo, la EPRI (Instituto de Investigación para la Energía Eléctrica) desarrolló un Modelo de Información Común CIM que desencadenó en la creación en el año 2003 de la norma IEC 61970 para almacenar los datos de las redes. Desde entonces cientos de personas y empresas han dedicado recursos económicos y tecnológicos para lograr implementar esta norma a niveles internacionales sin embargo por tratarse de un estándar relativamente reciente aún no es lo suficientemente conocido, lo que dificulta su implementación y la obtención de datos concretos al respectos.

Empresas involucradas en el tema han desarrollado herramientas que sirven para estructurar los datos de los archivos CIM con mayor facilidad. Una de ellas es el programa Rational Rose, (disponible en http://cimug.ucaiug.org/) donde se muestran de forma gráfica y ordenada los diagramas de clases descritos en la norma de la IEC y busca facilitar la representación de elementos de una red bajo este modelo. Además de esto, diferentes investigadores de la Universidad Pontificia Comillas de España, como Rafael Santodomingo y José Rodriguez, se han dedicado a publicar ensayos en donde se explica la necesidad de implementar el estándar y utilizan ejemplos sumamente sencillos para demostrar su utilidad.

Lamentablemente en Venezuela el avance con respecto a este tema es muy limitado por lo que es necesario informar al público pertinente de manera que poco a poco se vayan adaptando

las aplicaciones a este estándar y se creen los archivos pertinentes para poder utilizar aquellas herramientas (en su totalidad internacionales) que ya soportan este formato.

En cuanto a las herramientas de análisis de sistemas de potencia, el incremento de la demanda por parte de las compañías eléctricas ha desatado una tendencia a desarrollar cada vez más aplicaciones tanto privadas como lo son MatLab (Laboratorio de Matrices) o PowerFactory, como de software libre, por ejemplo InterPSS (Simulador de Sistemas de Potencia basado en tecnología de Internet).

Este trabajo de grado pretende exponer algunos de los beneficios del uso de los programas abiertos a la hora de realizar análisis de sistemas de potencia, en especial aquellos que son escritos Python debido a la versatilidad y auge de este lenguaje y la importancia de estos para el aprendizaje y comprensión de los códigos de implementaciones científicas. Esto sumado al hecho de la adaptabilidad de las aplicaciones abiertas, hacen del software libre una alternativa sumamente atractiva a nivel tanto académico como empresarial.

Objetivos.

Como objetivo general de este trabajo se tiene la aplicación del modelo CIM descrito en la norma 61970 de la IEC a una parte del sistema eléctrico nacional, mientras que en los específicos se tiene la actualización de los datos disponibles del sistema, la creación de una bases de datos relacional de la red en software libre, la ejecución de un flujo de carga desarrollado en Python y el almacenamiento de los resultados de este análisis en diferentes formatos, entre los que se encuentra el HTML (lenguaje de etiquetas de hipertexto)

CAPÍTULO I

MARCO TEORICO

1.1. Compañías eléctricas y programas de procesamiento de datos.

Las compañías que se encargan del análisis, control y planificación de un sistema eléctrico en particular, utilizan diferentes tipos de herramientas computacionales que han sido desarrolladas con el fin de facilitar la ejecución de estas actividades de forma rápida y confiable. Cada una de estas herramientas requiere datos específicos del sistema eléctrico de potencia que deben ser almacenados digitalmente en un formato particular para que la misma pueda funcionar. Debido a esto si una compañía pretende utilizar cierto software para un tipo de análisis de la red, se debe crear una base de datos específicamente para esta aplicación que esté almacenada en el formato compatible con dicho software.

Dado que la mayoría de las bases de datos de un sistema eléctrico de potencia se crean para una aplicación específica, no pueden ser utilizadas con otras herramientas (Wu & Schulz) , por lo tanto si una empresa posee varios programas para realizar uno o más análisis de una misma red, lo cual es sumamente común ya que normalmente existen numerosos departamentos y cada uno de ellos se encarga de una función especifica utilizando el software que más le convenga, la compañía en cuestión se verá obligada a mantener tantas copias de la base de datos como sea necesario, lo cual es muy poco práctico y sumamente difícil de mantener sincronizado.

Sumado a esto se añade la complicación del control de versiones, ya que cuando una aplicación se actualiza el formato que requiere su base de datos, en muchos casos, cambia. En general, esto no trae grandes problemas al usuario porque al generar una nueva versión de un programa

normalmente se añaden herramientas que sirven para importar la base de datos de la versión anterior y convertirla al formato nuevo. El verdadero inconveniente viene dado cuando una empresa desea mantener una versión anterior de un software trabajando en paralelo con la nueva, ya que se ve obligado a mantener una copia de la base de datos no solo por cada aplicación sino también por cada versión que se tenga de la misma (Mc Morran , 2007) . Además si por algún motivo se necesita intercambiar datos entre empresas, no será posible a menos de que utilicen exactamente el mismo software en la misma versión.

Existen herramientas de software libre como InterPSS (Simulador de Sistemas de Potencia basado en tecnología de Internet), PSAT (del ingles Power System Analysis Toolbox) y UWPFLOW (Flujo de Carga de la Universidad de Waterloo), programas de los que se discutirá más adelante) que han desarrollado adaptadores para importar datos desde otros formatos y convertirlos a los suyos propios, más no para exportarlos. Estos adaptadores son de gran utilidad pero su desarrollo para formatos de código cerrado es sumamente complicado debido a que la información necesaria no se encuentra disponible gratuitamente y en algunos casos está incompleta (Milano, Zhou, & Hou, 2009) .

En (Mc Morran , 2007) se enumeran las principales opciones a las que las empresas pueden recurrir en caso de que deseen tener los datos de un sistema eléctrico disponible para ser usado por cada una de las diferentes aplicaciones o para ser compartidos con otras empresas. Estos son:

1. Mantener copias la base de dato en el formato de la cada aplicación y para las diferentes versiones que se utilicen lo cual, cómo se menciono antes es sumamente complicado.

2. Guardar los datos en un formato compatible con todas las aplicaciones (esto requiere eliminar datos para que algunas herramientas funcionen ya que de contener información adicional que el programa no reconoce el formato quedará invalido para dicha herramienta). Esta opción trae como consecuencia la pérdida de precisión en aquellas aplicaciones que sí utilizaban la información eliminada.

3. Crear un solo archivo que contenga toda la información necesaria para cada aplicación y/o versión de una herramienta. Este caso requiere desarrollar adaptadores que conviertan este

archivo a cada uno de los formatos requeridos por cada aplicación, dejando sólo la información pertinente al caso de manera que la herramienta reconozca todos los datos y no se invalide el formato.

4. Crear un solo archivo con todos los datos, pero en un formato tal que sea compatible con todas las aplicaciones y se pueda guardar información adicional sin invalidar el archivo.

Según lo publicado en éste mismo artículo (Mc Morran , 2007), para poder considerar válida la cuarta opción, es necesario que el formato utilizado permita generar un modelo altamente detallado de un sistema eléctrico, debe ser capaz de almacenar datos extendidos sin afectar la información principal y por último tendría que se adoptado por la mayor cantidad posible de compañías y desarrolladores de herramientas, ya sea por motivos económicos o regulativos.

1.2. Modelo de Información Común.

El Modelo de Información Común, o CIM por sus siglas en inglés, aplicado al ámbito eléctrico es un UML (Unified Modeling Language), es decir, un Lenguaje Unificado de Modelación, que se utiliza para representar la información de elementos del mundo real pertenecientes a un sistema eléctrico (Xtensible Solutions, 2008) de forma que pueda ser compartida e interpretada por diferentes grupos de personas que trabajan con dicho sistema.

Si bien la IEEE (Instituto de Ingenieros Eléctricos y Electricistas) ha desarrollado desde inicios de los años 70's el CDF (Common Data Format o Formato de Datos Común) cuyo objetivo es el mismo que el del CIM (permitir el intercambio de datos del sistema eléctrico de potencia entre diferentes empresas) éste tiene las desventajas de ser limitado a datos estáticos y ser poco flexible. Por ello se necesita un formato moderno, gratuito, flexible y bien documentado para intercambiar los datos necesarios para el análisis de sistemas de potencia (Milano, Zhou, & Hou, 2009) .

Como parte del comité técnico denominado TC57 de la Comisión Electrotécnica Internacional, mejor conocida por sus siglas en ingles como la IEC, el EPRI (Electric Power

Research Institute) desarrolló los estándares 61970-301 (IEC, 2003) y 61968-11 (IEC Draft) el CIM con el fin de cumplir con las condiciones antes mencionadas (Mc Morran , 2007) .

Además de esto, el CIM es un ODM, es decir, un Modelo de Datos Abierto, lo cual significa que la mayor parte de la documentación referente al mismo se encuentra publicada de forma gratuita y es de fácil acceso. Esto representa una gran ventaja, ya que la principal idea es que las empresas puedan adaptarse a este modelo sin ningún tipo de complicaciones.

1.3. CIM/XML.

Uno de los aspectos más importantes a la hora de desarrollar en CIM fue la elección del lenguaje a utilizar. Con el aumento continuo de las aplicaciones en la web, entre otras menos conocidas, los lenguajes en constante crecimiento y estudio eran del tipo SGML (Standard Generalized Markup Language) dado que su flexibilidad los estableció como la tecnología dominante para codificar documentos estructurados en nuevas aplicaciones. Particularmente el lenguaje de marcas extensible o XML por sus siglas en ingles (eXtensible Markup Language) se convirtió en el formato de elección a la hora de intercambiar documentos de datos ya fuese a través del Internet o de redes privadas (deVos, Widergren, & Zhu) .

Los documentos escritos en XML están formados por entidades que contienen datos de diferentes tipos. Para identificar los diferentes datos presentes en el documento se utilizan las etiquetas, a través de las cuales se define la estructura del documento y las clases de datos almacenadas en él (Bray, Paoli, Sperberg-McQueen, & Maler, 2006) . Este lenguaje permite construir archivos que contienen una gran cantidad de datos y que pueden ser leídos con cualquier analizador estándar.

La definición de las etiquetas presentes en un documento XML se realiza comúnmente con un Document Type Definition (DTD) sin embargo éste no permite establecer relaciones dentro del documento entre los objetos que lo componen, por ello en lugar de esto en CIM se utiliza el RDF (Resource Description Framework) cuya función general es la misma que la del DTD con la diferencia de ser un modelo abstracto que permite asignar un identificador único para cada entidad del documento y relacionarlos a través de ellos (Wu & Schulz) .

1.3.1. Propiedades principales del UML.

Gracias a que es un UML, el CIM permite la definición clases, subclases, asociaciones, agregaciones y composición de clases (Mc Morran , 2007) . Esto quiere decir que se pueden establecer tipos de objetos con atributos específicos, a esto se lo conoce como clase. Una subclase es otro objeto pero que hereda atributos de la clase superior. Luego se pueden establecer numerosas relaciones entre las clases. Esto forma parte de la tendencia conocida como OOP o Programación Orientada a Objetos, donde las variables que se definen son consideradas como objetos a los que se les asigna una identificación, atributos y se les aplican métodos (también llamadas funciones). De esta manera si dos objetos de definen como de una misma clase, entonces ya se conocen los atributos que los detallan y los métodos en los que puede estar involucrado (Alvarez, 2012).

En el capítulo dos (2) de (Mc Morran , 2007) se encuentra un ejemplo detallado de las clases y la herencia entre clases. Además se muestran algunos diagramas de clases de UML para hacer el ejemplo más ilustrativo, siendo uno de ellos la Figura 1.1 Ejemplo de diagrama UML Figura 1.1. Aquí cada uno de los cuadros representa una clase de objeto y las flechas entre ellos las relaciones. La flecha con punta triangular completa (por ejemplo entre las clases persona y estudiante) indica herencia, es decir, que el estudiante es una subclase de persona y por ende hereda sus atributos, por lo tanto un estudiante puede tener hasta cinco atributos: año, numero de estudiante y curso, que son sus atributos propios y nombre y género, que son los heredados.

La flecha de punta fina indica una asociación, luego un estudiante no graduado se puede asociar a cero o varias materias (denotado al final de la flecha cómo “0...*”) y una materia se puede asociar con uno o más estudiantes (denotado al inicio de la flecha cómo “1...*”).

Las agregaciones y composiciones se indican en el diagrama utilizando rombos vacíos y rellenos respectivamente. En ambos casos significa que el objeto agregado o que compone al otro, forma parte de él, con la diferencia de que en las asociaciones si uno de ellos es destruido el otro puede seguir existiendo mientras que en las composiciones no.

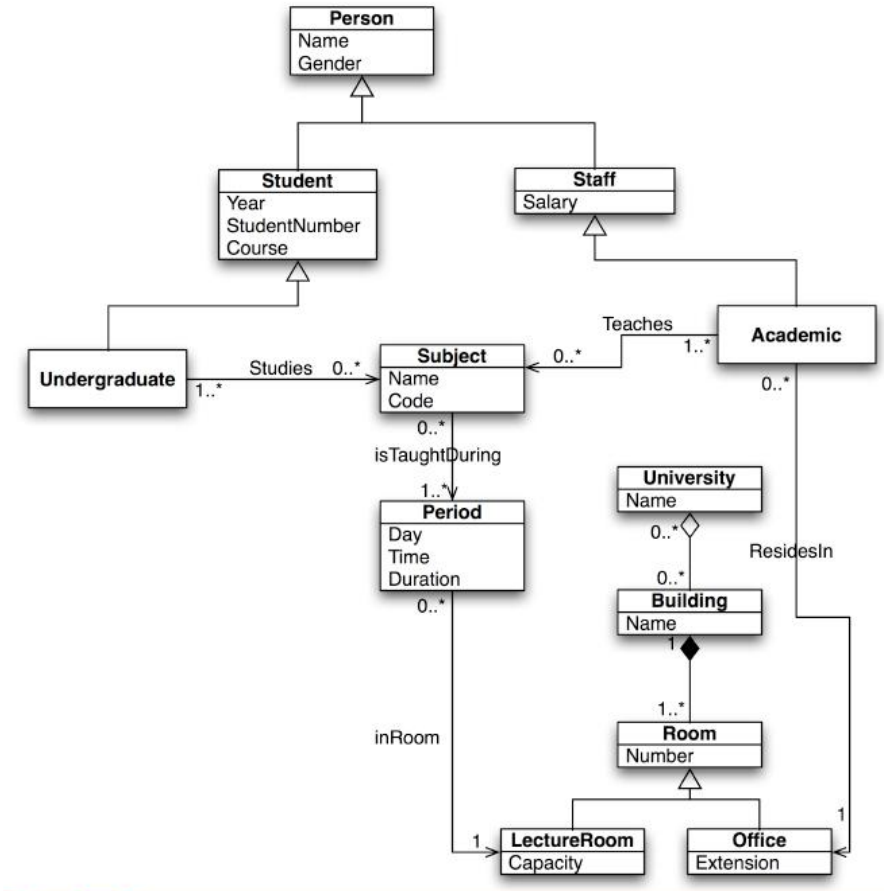


Figura 1.1 Ejemplo de diagrama UML (Mc Morran , 2007)

En el diagrama de ejemplo vemos que un edificio es agregado a una universidad, esto quiere decir que si la universidad como institución deja de existir, el edificio no se ve afectado. Por otro lado, un edificio está compuesto por salones y si es destruido también lo serán estos últimos.

Un aspecto importante del XML es, como su nombre lo indica, que es extensible, lo cual implica que a un documento dado se le puede agregar nuevas clases y asociaciones sin que éste pierda su validación, sólo se deben definir las nuevas etiquetas presentes en el archivo de modo que puedan ser identificas por el analizador (Bray, Paoli, Sperberg-McQueen, & Maler, 2006) . Gracias a esto el XML cumple con una de las principales características deseadas que era la flexibilidad.

1.4. Paquetes del CIM.

El Modelo de Información Común está diseñado para poder representar todos los elementos que constituyen un sistema eléctrico de potencia. Debido a que la cantidad de clases que se deben definir para lograr esto es sumamente extensa, tuvieron que ser agrupadas en paquetes para su mejor estructuración. Estos paquetes (numerados por sus nombres en inglés) son:

Core: Contiene la clase PowerSistemResource o traducido al español, Recursos del Sistema de potencia, que es de la cual nacen todas aquellas subclases concernientes a las propiedades físicas de la red. En este paquete se definen las clases correspondientes a base de voltaje, base de potencia, equipos conductores de corriente, entre otros.

Wires: define todas las piezas que se conectan eléctricamente a la red como por ejemplo breakers, fusibles, líneas, etc.

Generation: se divide en dos subclases: Production (se usa para definir tipos de generadores e incluso algunos componentes de generadores hidráulicos y térmicos) y GenerationDynamics (sirve para la descripción de rotores, tipos de turbinas, suplidores de vapor, etc.)

LoadModel: Datos referentes a las cargas. Incluye, entre muchas otras cosas, curvas de cargas y horarios de demanda.

Topology: incluye clases referentes a cómo los equipos se conectan entre ellos. Dos de sus subclases son BusNameMarker y TopologicalNode

Measurement: referente a medidas tomadas de algún elemento de la red. Se pueden catalogar en dos tipos, las que se asocian a un PowerSystemResource por lo que no son medidas eléctricas, sino características como peso o temperatura de un elemento de la red, y las asociadas a terminales (voltajes y corrientes). Esta clase no representa exclusivamente a equipos de medida en sí sino que puede estar asociada simplemente a una medida eléctrica en algún punto de la red.

Outage: referente a horarios para la planeación de la configuración de la red. Incluye clases asociadas a Switches para definir estado del elemento en un momento particular.

Protection: Parámetros y configuración de los elementos de protección que operan los Switches.

Todo esto acorde a la breve descripción encontrada en (Mc Morran , 2007) . Para la documentación completa acudir directamente a (IEC, 2003) y (IEC Draft) .

1.5. Análisis de Flujo de Carga.

El objetivo principal de un sistema eléctrico de potencia es proveer un servicio confiable e ininterrumpido a las cargas que alimentan, pero no solo eso, sino que además es imprescindible que los niveles de voltaje y el valor de la frecuencia se mantengan constantes en sus valores nominales o al menos lo más cercano posible a estos para que los elementos conectados a la red puedan funcionar correctamente (Fouad & Anderson, 2003) .

Dado que las características topológicas de una red son fijas (considerando que las ramas en servicio y los valores de impedancia de éstas no varían), el voltaje y la frecuencia de un sistema dependen de la carga que esté conectada a él y los flujos de potencia que se den entre las barras. El análisis de flujo de carga o PF por sus siglas en inglés (Power Flow) tiene como objetivo calcular estos voltajes en módulo y ángulo y con ellos determinar los flujos de potencia que circulan por las ramas.

A principios del siglo XX los sistemas de potencia eran mayormente radiales por lo que realizar el análisis de flujo de carga era relativamente sencillo, sin embargo con el aumento de las interconexiones y el crecimiento de las redes esto se complicó considerablemente. Para poder resolver este problema, en 1929 un grupo de ingenieros del M.I.T. (Massachusetts Institute of Technology) (Dunstan, 1947) desarrollaron los llamados analizadores de circuito o cuadros de cálculo C.A. (entiéndase, de Corriente Alterna) “en los cuales, resistencias, inductancias y capacitancias variables se conectaban para formar una réplica de una fase del sistema real con valores en escala” (Stevenson, 1975) de manera que al ponerlos en funcionamiento se podía determinar el comportamiento del sistema real y estimar con bastante aproximación los voltajes y los flujos de potencia. Sin embargo el proceso de utilización de estos cuadros era largo y tedioso, por lo que al surgir las computadoras digitales, no pasó mucho tiempo antes de que se aprovecharan para empezar a resolver este problema. Actualmente las herramientas

computacionales han reemplazado prácticamente por completo el uso de los analizadores de circuito en la mayor parte de las aplicaciones donde eran utilizadas, pero muy especialmente en los análisis de flujos de carga.

1.5.1. Planteamiento del modelo clásico de Flujo de Carga.

El modelo clásico para resolver el problema de flujo de carga se basa en establecer las ecuaciones de balance de potencia activa (P) y reactiva (Q) en cada una de las barras donde sea posible. Tomando como premisa que la topología de la red es conocida, si se tienen los datos de la potencia activa generada y demandada de una barra, entonces se puede establecer la ecuación de balance de potencia activa de dicha barra. Lo mismo ocurre con la potencia reactiva.

La disponibilidad de los datos de cada barra depende del tipo de elemento conectado a ella. Según esto cada barra puede ser clasificada como una de las siguientes cuatro opciones (Milano F. , 2010) :

Barras PQ: son aquellas barras donde no hay generación directamente conectada sino que sirven para suplir una carga. La potencia activa y reactiva consumida por una carga es dependiente del voltaje en los terminales de la misma, sin embargo dado que el voltaje equivalente de una barra de carga normalmente es regulado utilizando los cambiadores de tomas de los transformadores aledaños a ella, en condiciones normales de operación, la tensión se considera constante y se puede modelar la carga en régimen permanente con una demanda constante.

Barras PV: son aquellas barras donde hay generación conectada y puede o no haber carga. La potencia generada por un generador sincrónico se controla a través del gobernador de velocidad y el voltaje en terminales con el regulador automático de voltaje. Luego la potencia generada y el voltaje son datos conocidos, además en caso de haber carga esta se modela, al igual que en la barras PQ como una demanda constante y conocida de potencia activa y reactiva.

Barra SLACK: uno de los generadores es definido como fuente independiente de voltaje, es decir, con magnitud de voltaje y ángulo de referencia constante. Para resolver cualquier problema

de un sistema eléctrico en corriente alterna, es necesario establecer la fase de un voltaje cualquiera como referencia, comúnmente se toma ángulo cero en algún punto estratégico de la red. En este caso la barra slack es la que se elige como referencia. Dato que al iniciar el análisis se desconocen las pérdidas de potencia activa del sistema, si bien esta barra tiene conectado un generador sincrónico con su correspondiente gobernador de velocidad, no se le asigna valor a la potencia activa generada sino que se deja como variable para que tome el valor necesario para cumplir con los balances de potencia.

Acorde a la clasificación anterior, en cada barra tendremos dos incógnitas y dos datos dependiendo del tipo de barra que sea. Estos se pueden observar en la Figura 1.2.

El caso particular de una barra sin carga y sin generación, será tomado como PQ donde Pd y Qd toman valor nulo.

Barras PQ	Potencia activa y reactiva demandada: Pd y Qd	Voltaje con modulo y ángulo:|V| y θ
Barras PV	Potencia activa generada y demandada en caso de haber carga y modulo del voltaje en terminales: Pg, Pd y |V|	Potencia reactiva generada y ángulo del voltaje: Qg y θ
Barras Slack	Voltaje con modulo y angulo:|V| y θ	Potencia activa y reactiva generada: Pg y Qg

Figura 1.2 Datos e incógnitas según el tipo de barra

Además de establecer los tipos de barras y los datos conocidos de cada una de ellas, es necesario conocer plenamente la topología de la red. En el modelo clásico de flujo de carga, las líneas de transmisión y los transformadores se modelan con parámetros concentrados y modelo π. Teniendo todos los datos del sistema, se puede construir la matriz de admitancias nodales asociada a la red.

Se debe recordar que al realizar casi cualquier análisis de un sistema de potencia, por cuestión de simplificación de los cálculos, la forma más común de representar potencia, voltaje, corriente e impedancia es utilizando valores en por unidad, y el flujo de carga no es la excepción. En el caso de los ángulos, la unidad a utilizar será grados.

Finalmente se pueden escribir las ecuaciones de balance de potencia de aquellas barras donde se conoce la inyección de potencia activa o reactiva. Esto quiere decir que:

Para una barra PQ se establecen las ecuaciones de balance de potencia tanto de potencia activa como reactiva.

En una barra PV sólo se establece la ecuación de balance para la potencia activa.

Para una barra SLACK no se establece ninguna ecuación de balance porque se desconocen tanto los datos de potencia activa como de reactiva.

Esto quiere decir que por cada barra PQ se tienen dos ecuaciones de balance y por cada PV sólo una.

1.5.2. Ecuaciones.

Al inicio de la sección 1.5.1 se estableció que para resolver el problema del flujo de carga era necesario plantear las ecuaciones de balance de potencia en aquellas barras donde era posible.

Básicamente el balance de potencia es la verificación de que se cumpla la Ley de la conservación de la energía, la cual establece que “la energía no se crea ni se destruye; sólo se puede transformar de una forma a otra” (Seese & Daub, 2005) . Aplicado al caso que se esta estudiando, se traduce en que en una barra cualquiera la potencia generada (en caso de haber generador) menos la potencia consumida (si hay carga conectada) debe ser igual a la potencia neta, es decir, la inyectada al resto del sistema.

De forma general, para hallar las respuestas del flujo de carga de un sistema de potencia se debe resolver el conjunto de las dos (2) ecuaciones siguientes para cada una de las barras que lo requiera:

(1.1)

(1.2)

Donde el subíndice "i" indica el número que identifica a la barra y P_g, P_d y P_{neta} son la potencia activa generada, demandada e inyectada respectivamente. De forma análoga ocurre con los subíndices de la ecuación referente al balance de potencia reactiva 1.2.

Luego para que se cumpla el balance de energía los términos ΔP_i y ΔQ_i deben ser iguales a cero. Dado que para aquellas barras donde se pueden plantear las ecuaciones 1.1 y 1.2 se suponen conocidas las potencias generadas y consumidas, la solución del problema depende enteramente de las potencias inyectadas en la barra, cuyas expresiones se presentan a continuación:

(1.3)

(1.4)

Donde n es el numero total de barras, G_{ik} y B_{ik} son la conductancia y susceptancia correspondientes al elemento Y_{ik} de la matriz de inductancia de la red y θ_{ik} es la diferencia angular entre el voltaje de la barra i y la barra k. Puesto que las características de la red se suponen conocidas, quedan como incógnitas P_{neta} , Q_{neta} , V y θ de cada barra.

Estas ecuaciones nacen de desarrollar la expresión de la potencia aparente en una barra, la cual establece que:

(1.5)

Donde I_i representa la suma de las corrientes inyectadas a cada un de las barras del sistema, y se calcula:

(1.6)

Sustituyendo la ecuación 1.6 en 1.5 y conjugando a ambos lados se tiene:

(1.7)

Expresando el termino de la impedancia como la suma de la conductancia y susceptancia y el voltaje en forma polar (módulo y ángulo se obtiene:

(1.8)

Dado que el término V_i es constante se puede introducir dentro de la sumatoria y realizar el producto de los exponenciales:

(1.9)

Aplicando el Teorema de Euler para expresar el término $e^{-j\theta ik}$ en función de senos y cosenos se tiene:

(1.10)

Finalmente desarrollando la ecuación 1.10, considerando el caso particular cuando i=k y colocándolo fuera de la sumatoria e igualando las partes reales y complejas del resultado final, se obtienen las expresiones 1.3 y 1.4.

La deducción de estas ecuaciones se puede estudiar con mayor detalle en la sección 3.3 de (Gómez-Expósito, Conejo, & Cañizares , 2009) o la 4.3 de (Milano F. , 2010).

1.5.3. Métodos numéricos aplicados al flujo de carga.

Al sustituir las expresiones 1.3 y 1.4 en las ecuaciones de balance de potencia se obtiene un sistema de ecuaciones no lineal dependiente del módulo y el ángulo del voltaje en cada barra, cuya solución no puede ser calculada analíticamente. Para poder resolver este tipo de problemas

es necesario recurrir a los métodos numéricos iterativos, siendo los más comunes en éste campo el método de Jacobi, el de Gauss-Seidel y el Newton-Raphson.

Si bien los dos primeros métodos han sido usados ampliamente desde hace décadas para resolver el problema del flujo de carga, gracias a que las limitaciones computacionales han disminuido considerablemente en los últimos años, es posible ejecutar una técnica que requiera una cantidad de recursos de procesamiento mayor a cambio de obtener beneficios en cuanto al tiempo de ejecución (Milano F. , 2010). Este es el caso del método de Newton-Raphson también conocido como Newton-Fourier, el cual es uno de los más usados en la actualidad pero que en los años setenta se consideraba computacionalmente limitante ya que requería de lo que en aquella época se consideraba un gran poder de cálculo.

El método de Newton tiene la ventaja de que "el número de iteraciones requeridas… es prácticamente independiente del número de barras"(Stevenson, 1975), a diferencia del Gauss-Seidel cuyo tiempo de ejecución aumenta casi directamente con estas. Esto es sumamente conveniente si se considera el hecho de que la mayoría de herramientas computacionales que se desarrollan para realizar análisis de flujo de carga están pensadas para manejar redes de gran tamaño y es uno de los principales motivos por los que se decidió aplicarlo en este estudio.

1.5.3.1 Método de Newton-Raphson

Este método se basa "en la expansión de series de Taylor para una función de dos o más variables…Las derivadas de orden superior a uno se desprecian" (Stevenson, 1975).

La serie de Taylor establece que para una función de la forma F(x)=0 se cumple que:

— – — (1.11)

Los términos de orden superior se desprecian ya que no tienen grandes efectos en el resultado final por lo que se obtiene:

— (1.12)

Considerando que F(x)=0, se puede igualar la ecuación 1.12 a este valor y despejar la variable x, tal como se muestra en la ecuación 1.13.

(1.13)

Donde x_0 es un valor inicial alrededor del cual se encontrará la solución y J es la matriz Jacobiana de la función, es decir, contiene las derivadas parciales de las ecuaciones del sistema.

Normalmente el valor inicial x_o se establece en lo que se conoce como un perfil plano de tensiones, es decir, módulo del voltaje uno en por unidad (1pu.) y ángulo 0°. Para cualquier caso de estudio bien planteado estos valores representan un buen punto de partida para las iteraciones. Existen programas de análisis de flujo de carga que primero ejecutan el método de Gauss-Seidel para obtener un valor inicial más cercano al real y así disminuir el número de iteraciones necesarias para hallar el resultado final con el Newton-Raphson.

Al igual que en todos los métodos iterativos, la metodología se repite hasta que se cumpla cierta condición. En el caso del flujo de carga, se desea que:

(1.14)

Donde la tolerancia depende del criterio de la persona que realice el estudio. Por lo general toma valores entre $1x10^{-4}$ y $1x10^{-8}$.

La construcción de la matriz Jacobiana es la parte que más recursos computacionales ocupa a la hora de programar un flujo de carga utilizando el método de Newton-Raphson, ya que cada barra PQ aporta dos ecuaciones (balance de potencia activa y reactiva) y dos incógnitas (módulo y ángulo del voltaje) al sistema y cada barra PV aporta una (balance de potencia activa y ángulo del voltaje). Luego, al construir la matriz se debe derivar cada ecuación tantas veces como variables haya. En otras palabras, si una red tiene n_{PQ} barras del tipo PQ y n_{PV} barras del tipo PV, el Jacobiano será una matriz cuadrada de dimensión $2*n_{PQ}+n_{PV}$, lo cual puede significar un tamaño bastante considerable dependiendo de la red a estudiar.

Ahora que se tienen algunas nociones básicas del método de Newton-Raphson se puede discutir el planteamiento general del flujo de carga bajo este método.

La variable x será un vector de dimensión $(2*n_{PQ}+n_{PV})*1$ compuesto por los ángulos y los módulos de los voltajes de aquellas barras donde se desconocen estos datos y la función F será un vector con la misma dimensión de x compuesto por las ecuaciones de balance de potencia activa y reactiva de las barras correspondientes.

Esto quiere decir que para un caso cualquiera x y F serían de la forma:

(1.15)

Donde las barras i, k serían del tipo PQ (ya que se plantean ecuaciones de balance de potencia activa y reactiva y se tiene tanto módulo como el ángulo del voltaje como incógnita) y la barra k sería de tipo PV (sólo se plantea el balance de potencia activa y de incógnita se tiene el ángulo de la tensión en esa barra). Por comodidad se recomienda que $i<j<k$, es decir, ordenar las ecuaciones de balance y las incógnitas en orden ascendente respecto a la numeración de las barras.

Paralelo a esto, el Jacobiano sería tal como se muestra en la Figura 1.3 y se seguiría el proceso iterativo descrito por la ecuación 1.13 hasta cumplir con la condición de la tolerancia.

(1.16)

Figura 1.3 Matriz Jacobiana para el método del Newton Raphson

CAPITULO II

PROGRAMAS PARA EL ANÁLISIS DE SISTEMAS ELECTRICOS.

En la primera parte del capítulo anterior se habló de que existe una variedad de herramientas computacionales que sirven para realizar análisis de sistemas de potencia. En esta sección se discutirá un poco más al respecto, específicamente de la distinción entre el software libre y el no libre.

Como se sabe, las empresas que realizan análisis de sistemas eléctricos utilizan algún tipo software para la mayoría de sus estudios. Debido al gran desarrollo en el sector eléctrico que se ha dado en numerosos países, cada vez hay mayor demanda por este tipo de programas, sin embargo gran parte de los que han surgido en los últimos años son del tipo privativo (también llamados software propietario o privado), es decir, que para poder utilizarlos se debe realizar un pago para adquirir una licencia y al instalarlo no se tiene acceso a su código fuente por lo que modificarlo es prácticamente imposible. Por ser programas comerciales que pueden ser instalados por múltiples clientes, tienen una estructura preestablecida que no siempre se adapta a las necesidades del usuario.

Debido a los inconvenientes surgidos por utilizar las aplicaciones comerciales, las empresas han creado grupos de investigación y desarrollo cuyo objetivo principal es diseñar programas de análisis de sistemas tales que se pudiese prescindir de aquellos que requieren licencia. Al hacerlo se pueden satisfacer todas las necesidades de la compañía sin tener que adaptarse al modo de trabajo de un software ajeno, además, gracias a que los desarrolladores de la herramienta trabajan para la empresa, es posible realizar cambios de forma rápida, eficiente y económica.

Existen casos en los que programas de código abierto se vuelven disponibles para el uso del público en general, de tal forma que cualquier persona puede no solo tener acceso al software sino también a su código.

2.1. Herramientas de tipo Software Libre.

Antes que nada se debe tener claro cuáles son las condiciones indispensables de un software libre. Richard Stallman (Stallman, 2004) establece que los usuarios deben tener la libertad para "ejecutar, copiar, distribuir estudiar, cambiar y mejorar el software". Más específicamente define cuatro (4) características:

> Libertad 0: la libertad para ejecutar el programa sea cual sea nuestro propósito
>
> Libertad 1: la libertad para estudiar el funcionamiento del programa y adaptarlo a tus necesidades —el acceso al código fuente es condición indispensable para esto.
>
> Libertad 2: la libertad para redistribuir copias y ayudar así a tu vecino.
>
> Libertad 3: la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto.

Estas libertades le proporcionan al software libre una serie de ventajas con respecto a otros tipos. Algunas de estas son:

Tienen un muy bajo costo (en algunos casos nulo) de adquisición y son de libre uso.

Se encuentran al escrutinio público, lo que facilita encontrar y corregir errores además al haber una mayor cantidad de personas involucradas en un proyecto se incrementa la innovación tecnológica que se puede alcanzar.

Los programas de software libre son independientes de proveedores particulares, por lo que tienen menos requisitos de hardware y por ende mayor durabilidad en el tiempo. A diferencia de los programas privativos, una herramienta de software libre no requiere de un sistema operativo específico, "lo que se traduce en un mejor soporte -de manera general- para las versiones antiguas de software y de plataformas de hardware o software más minoritarias" (Culebro, Gómez, & Torres, 2006) .

Aumentan las posibilidades de aprendizaje de los usuarios. Cuando una persona utiliza un software privado, se encuentra limitado a lo que sea que el programa le arroje, en cambio si se

tiene acceso a la estructura interna se puede analizar y comprender de mejor manera cómo funciona el programa e incluso aplicarle mejorías que faciliten su uso.

Una consecuencia directa del punto anterior, es que el software libre tiene mayores posibilidades de adaptación con el usuario.

Utilizar un software libre en empresas o instituciones que manejen datos sensibles de otras personas, como bancos o instituciones gubernamentales, ayudaría a garantizar la independencia de dichas instituciones de los proveedores de software privado, sin mencionar que se disminuye el riesgo a un sabotaje debido a vulnerabilidades deliberadas o al chantaje por información almacenada en formatos de propietarios.

Estas son algunas de las ventajas que exponen M. Culebro y el resto de los autores de "Software libre vs software propietario" (Culebro, Gómez, & Torres, 2006) . En éste libro y en (INDENE-USB, 2010) también se describen ciertas desventajas de software libre, algunas de las cuales se resumen a continuación:

El software libre no tiene garantía, por lo que el usuario lo utiliza bajo su propio riesgo. Por ende se debe monitorear las correcciones de errores que se hacen públicas y arreglarlas a la brevedad posible porque representan fuentes potenciales de intrusión. Se debe acotar si se tiene cierto conocimiento de programación los errores pueden ser solucionados personalmente ya que normalmente cuando se descubren y publican también se indica cómo resolverlos, mientras que con el software privado se debe esperar a que la compañía arregle el problema y publique la nueva versión, en muchos casos teniendo que pagar para obtenerla.

Por el momento las interfaces gráficas de usuario y la multimedia se consideran un poco rudimentarias en comparación con algunas de software privado, lo que hace que la curva de aprendizaje sea un poco mayor que las de aquellas herramientas que poseen interfaces más amigables.

En muchos casos el usuario debe tener nociones básicas de programación ya que la administración del sistema no se realiza de forma gráfica y la configuración del hardware no siempre es intuitiva. Además la diversidad de distribuciones, métodos de empaquetamientos,

licencias de uso, herramientas con un mismo fin, etc., pueden crear confusión en personas poco experimentadas en el área de la informática.

Únicamente los proyectos importantes y de trayectoria tiene buen soporte, ya que la documentación que se publica proviene tanto de desarrolladores como usuarios, por lo que si una herramienta no tiene gran interés público, su crecimiento será menor.

Un tipo de software que se encuentra muy ligado al libre, es el llamado “open source” o de código abierto. Cómo su nombre lo indica en este tipo de programas el usuario tiene acceso al código fuente sin embargo pueden existir limitaciones con respecto a lo que se puede o no hacer con dicho código (Stallman, 2004) . Esto clasifica al open source justo en el límite entre el software libre y el privativo, sin embargo dado que ofrece mayores libertades que el resto de los programas de propietario se ha decidido nombrarlo en esta sección en lugar de bajo software privado.

En el ámbito eléctrico existen varios programas de software libre y de código abierto que se encuentran disponibles. Algunos de ellos se encuentran reflejados en la Figura 2.1 donde se especifica cuales de los más importantes análisis se puede realizar con cada uno.

En (Milano F. , 2010) se definen las columnas de Figura 2.1 según el nombre en ingles del estudio:

- PF: Flujo de carga convencional.
- CPF: Continuación de flujo de carga y/o análisis de estabilidad de voltaje.
- OPF: Flujo de carga óptimo.
- EA: Análisis de estabilidad de pequeña señal.
- TDS: Simulaciones en el dominio del tiempo para análisis de estabilidad transitoria.
- EMT: Transitorio electromagnético.

Package	Language	PF	CPF	OPF	EA	TDS	EMT	GUI	CAD
AMES	Java	✓		✓					
EST	Matlab	✓			✓	✓			✓
InterPSS	Java	✓				✓		✓	✓
MatDyn	Matlab	✓				✓			
MatEMTP	Matlab					✓	✓	✓	✓
Matpower	Matlab	✓		✓					
ObjectStab	Modelica	✓			✓	✓		✓	✓
OpenDSS	Delphi	✓				✓		✓	✓
PAT	Matlab	✓			✓	✓			✓
PSAT	Matlab, Octave	✓	✓	✓	✓	✓		✓	✓
PST	Matlab	✓	✓		✓	✓			
Pylon	Python	✓		✓				✓	✓
UWPFLOW	C	✓	✓						
VST	Matlab	✓	✓		✓	✓		✓	

Figura 2.1 Paquetes de código abierto y libre para análisis de sistemas de potencia. (Milano F. , 2010)

Las últimas dos columnas corresponden a aspectos estéticos de la aplicación: sí posee interfaz de usuario gráfica (GUI) o editor de diagrama unifilar asistido por ordenador (CAD).

Entre los de software libre se encuentran InterPSS y MatPower.

2.2. Herramientas de tipo Software Privativo.

En contraparte al software libre esta el de tipo privativo que “Se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o que su código fuente no está disponible o el acceso a éste se encuentra restringido” (Culebro, Gómez, & Torres, 2006) , es decir, que no cumplen con una o ninguna de las cuatro libertades establecidas en la sección anterior.

Si bien las libertades del software libre poseen grandes beneficios para el usuario, los software con propietario también poseen ventajas (y desventajas) que no se deben olvidar,

algunas de estas se encuentran en (Culebro, Gómez, & Torres, 2006) y se pueden resumir de la siguiente manera:

Las compañías que los desarrollan contratan personal altamente capacitado tanto para la creación de la herramienta como para el departamento de control de calidad, de manera que el producto pasa por una serie de pruebas antes de ser liberado al público, lo cual garantiza su funcionalidad. Además existe personal dedicado exclusivamente al soporte al usuario, para que si alguno experimenta cualquier problema haya personas capacitadas para ayudarle.

Los programas de propietario más comunes son usados por muchas personas por lo que es sencillo encontrar documentación acerca de su uso.

Proveen aplicaciones para usos específicos que muchas veces no son desarrollados en el mundo del software libre. Algunas de estas herramientas llegan a ser tan conocidas y utilizadas en su ámbito, que las empresas suelen ofrecer versiones educativas gratuitas o a muy bajos precios, de manera que los estudiantes se puedan familiarizar con el programa aún antes de sumergirse en el campo laboral.

Normalmente los recursos obtenidos por la compañía proveedora son destinados al mejoramiento de la herramienta lo que permite generar versiones cada vez más modernas y eficientes.

Entre las desventajas se tiene:

En muchas ocasiones cuando una empresa desarrolladora es comprada por otra que no tiene interés en mantener todas las herramientas creadas por la primera, el software es descontinuado y se cancela el soporte técnico para los usuarios por lo que eventualmente se vuelve obsoleto y los clientes se ven obligados a buscar otras alternativas para sustituirlo.

Se tiene una dependencia con el proveedor para poder adaptar legalmente el software a las necesidades propias del consumidor. En caso de ser posibles, son costosas y las realizará el fabricante dentro de sus posibilidades.

Es ilegal copiar o compartir el programa, lo cual dificulta la instalación del mismo por grandes grupos de personas o instituciones ya que se debe adquirir una licencia cada vez que se vaya a instalar.

Los usuarios no pueden mejorar la aplicación ellos mismos sino que en caso de tener una idea innovadora deben elegir entre vendérsela al fabricante o crear su propia aplicación.

Acorde a (INDENE-USB, 2010) algunos de los softwares privados que existen en la industria de la energía eléctrica son:

- AGORA - Advanced Grid Observation Reliable Algorithms - http://www.elequant.com/products/agora/ Software allowing the electric power industry to manage power grids and restore power after blackouts.
- AMTECH Power Software - http://www.amtech-power.com/ Design and verification software for high and low-voltage distribution systems. Covers IEC and NEC standards, protective device coordination and lightingdesign.
- IPSA Power Ltd Power System Analysis Software - http://www.ipsa-power.com software for the design, operation and planning of electrical generation, transmission and distribution systems.
- Electranix Corporation - http://www.electranix.com PSCAD is a platform for building simulations of electric power and power electronic systems, controls and protections.
- Fractal Power Engineering Software - http://www.fractal.hr/index_en.htm Power engineering software development, load flow, and short circuit analysis, with a graphical user interface.

...

- Powerex - http://powerex.150m.com Power distribution software, load flow and short circuit studies.
- Operation Technology, Inc. - http://www.etap.com Designers and developers of ETAP PowerStation,software for analysis and design of electrical power systems and PSMS, real-time power management system.
- PLECS - Power Electronics Modelling - http://www.plexim.com Simulation of electrical circuits within the MATLAB/Simulink environment. It is specially designed for power electronic and drive systems.
- Distribution Networks Real Time Restoration - http://digilander.libero.it/stockbroker/adiscon.html Software engine for real time restoration and reconfiguration of large scale electrical distribution networks.

Por último, pero no menos importante, está DigSILENT (Digital Simulator and Electrical Network) de Power Factory del cuál se discutirá más adelante y cuya información básica se puede encontrar en http://www.digsilent.de/

CAPITULO III

REDES ELECTRICAS.

Los sistemas eléctricos de potencia están compuestos esencialmente por una o más redes eléctricas conectadas entre sí. “La red eléctrica es un elemento para convertir y transportar energía...se compone de tres partes principales: las centrales generadoras, las líneas de transmisión y las redes de distribución.” (Stevenson, 1975)

Basándose en (Pacheco) y en el 1er capitulo de (Stevenson, 1975) se puede resumir que un sistema eléctrico está compuesto por cuatro partes o etapas:

Generación: en éste punto se utilizan máquinas eléctricas para transformar diferentes fuentes de energía (potencial, térmica, nuclear, etc.) en energía eléctrica. Las instalaciones donde ocurre éste proceso se conocen como centrales eléctricas y pueden ser hidráulicas, termoeléctricas o de energías alternativas. Normalmente el nivel de tensión al que la energía es producida se encuentra entre los 12 y los 29kV.

Transporte: esta etapa no está presente en todos los sistemas eléctricos, sino principalmente en aquellos donde la generación se encuentra alejada de los centros de consumo. Aquí se utilizan las líneas de transmisión para transportar bloques de potencia a altos niveles de tensión, 115, 230, 400 y 765kV en el caso de Venezuela, esto con el fin de minimizar las pérdidas.

Distribución: es la parte del sistema eléctrico que se encarga de llevar la energía a los consumidores finales. Los niveles de tensión son bastante menores que en la transmisión y van disminuyendo a medida que se acercan a la carga acorde al nivel de voltaje que la misma requiera.

Consumo: como su nombre lo indica representa a los consumidores finales de la energía, es decir, quienes hacen uso de la misma. Normalmente se dividen en pequeños consumidores (principalmente cargas residenciales) que requieren bajos niveles de tensión y las cargas industriales, que en su mayoría utilizan altos niveles de tensión según la actividad que realizan.

A lo largo de estas cuatro etapas siempre se encuentran presentes las máquinas eléctricas, del tipo rotativas en la generación y el consumo (generadores sincrónicos y motores, por nombrar sólo un par de ellas) y los transformadores en el transporte y distribución (Pacheco) .

3.1. Sistema Eléctrico Nacional

A lo largo de las últimas décadas, el sistema eléctrico de Venezuela, también conocido como SEN (Sistema Eléctrico Nacional), ha destacado a nivel mundial por varias razones, siendo dos de las más importantes el hecho de haber sido uno de los primeros en integrar a la red un sistema de transmisión a 765 kV y por tener una de las centrales hidroeléctricas más grandes del mundo (Central Hidroeléctrica Simón Bolívar, ubicada en el estado Bolívar y con una capacidad instalada de 10.000 MW).

Según (Ministerio del Poder Popular para la Energía Eléctrica., 2010) para diciembre del año 2010 la capacidad instalada a nivel nacional era de 24.838 MW, de los cuales 10.216 MW correspondían a centrales térmicas y el resto a hidráulicas. Se debe acotar que con las medidas tomadas por el gobierno nacional, se esperaba que para finales del 2012 se hubiesen incorporado 9.077 MW a través de numerosos proyectos que se iban a llevar a cabo en diferentes partes del país, estos incluían la puesta en marcha de varios centros de generación distribuida y la creación de nuevas centrales térmicas e hidráulicas (Ministerio del Poder Popular para la Energía Eléctrica., 2010) .

Se puede observar en diversas fuentes, incluidas (Ministerio del Poder Popular para la Energía Eléctrica., 2010) y (Ministerio del Poder Popular para la Energía Eléctrica., 2011) , que el Ministerio del Poder Popular para la Energía Eléctrica o MPPEE ha desarrollado múltiples proyectos enfocados primordialmente en incrementar la capacidad de generación del país con el fin de satisfacer la demanda eléctrica nacional que se encuentra en constante crecimiento.

Como se sabe, un aumento significativo en el consumo eléctrico sin un incremento de la generación, puede llevar al sistema a un punto de operación crítico, es decir, que el voltaje y la frecuencia se encuentren tan alejados de sus valores de operación normal y/o que una o más líneas de transmisión estén tan sobrecargadas, que podría llegar a ser necesario realizar un bote de carga.

3.1.1. Datos del SEN en software privado.

Para poder determinar si bajo ciertas condiciones especificas el sistema alcanza un punto de operación como el descrito en la sección anterior, se debe realizar un estudio de flujo de carga para calcular, entre otras cosas, los voltajes en cada barra, la frecuencia del sistema y los flujos de potencia que circularían por las ramas si las condiciones se dan en la vida real.

Considerando que en los últimos años se han realizado y planificado múltiples cambios al SEN, se puede suponer que la empresa encargada del manejo de las principales redes del país, en este caso CORPOELEC, ha tenido que realizar diversos cambios a la base de datos que contiene la información de todo el sistema para poder llevar a cabo los estudios pertinentes, pero al mismo tiempo manteniendo los datos iniciales intactos para tener la información sincronizada con la realidad. Esto lo ha logrado gracias a que utiliza Power Factory de DigSilent para los análisis del SEN.

Este software contiene numerosas aplicaciones que sirven para planificar, estudiar y operar los aspectos técnicos de un sistema eléctrico. Su desarrollo comenzó en el año 1976 utilizando programación orientada a objetos y el lenguaje de programación C++. Actualmente es un software de suma popularidad, en gran parte gracias a que a estas alturas del proyecto sus algoritmos al alcanzado grandes niveles de eficiencia y confiabilidad lo que prácticamente garantiza su buen funcionamiento.

3.1.1.1 Aspectos relevantes de Power Factory

Esta herramienta funciona de tal manera que permite establecer diferentes casos de estudio, escenarios de operación y variaciones (entre muchas otras cosas) para una o varias redes que

conforman un proyecto, lo cual “Evita que el usuario deba organizar los análisis de forma externa a la herramienta en distintos archivos y carpetas” (Alvarez, 2012) .Cada proyecto contiene al menos una red, la cual puede estar interconectada con muchas otras.

Para crear un proyecto se establece un caso de estudio base en el cual se van a almacenar todos los datos de los elementos que conforman la(s) red(es) del sistema a estudiar, esto incluye los datos de demanda, estatus de los switches, si el elemento está o no en servicio, etc. A un caso de estudio base se le pueden aplicar diferentes variaciones, éstas establecen cambios topológicos que se le efectúan al caso en cuestión y pueden ser de cualquier tipo, desde inclusiones o puestas fuera de servicio de líneas de transmisión o barras hasta conexiones con otras redes, entre otras. A cada variación se le asigna una fecha, de modo que si se realiza un análisis en un tiempo posterior a dicha fecha, éste tomará en cuenta todas las variaciones que estén diseñadas para estar activas en ese momento. Un aspecto primordial de esto es que la fecha del estudio no está ligada con la fecha real en la que se realiza el mismo, sino que puede ser especificada de tal forma que se activen las variaciones deseadas. Cada vez que se aplica una variación, se crea un nuevo caso de estudio donde la red tiene la topología especificada en la variación. De esta forma se pueden realizar los estudios para determinar los efectos de un cambio topológico en la red.

Paralelo a las variaciones, DigSilent también permite crear escenarios de operación, los cuales funcionan de manera muy similar a estas. En ellos se pueden realizar principalmente cambios en los datos de la demanda. Uniendo esto a las variaciones se pueden estudiar los casos más críticos del sistema, por ejemplo creando escenarios de carga máxima ligados a la pérdida de una o varias líneas. Se debe recordar que con sólo cambiar la fecha de estudio y/o desactivar un escenario de operación, se tendrán nuevamente los datos originales. Vale acotar que a diferencia de las variaciones, a los escenarios de operación no se les establece una fecha sino que se activa o desactivan.

Gráficamente estas aplicaciones de DigSilent las podemos ver en la Figura 3.1. Esta muestra parte de la interfaz del programa, donde se puede observar el administrador de datos con todos los detalles un proyecto en particular. Aquí se ilustran los casos de estudio y variaciones que se han creado para este proyecto. En este caso no se han establecido escenarios de operación, pero de ser necesarios podría hacerse.

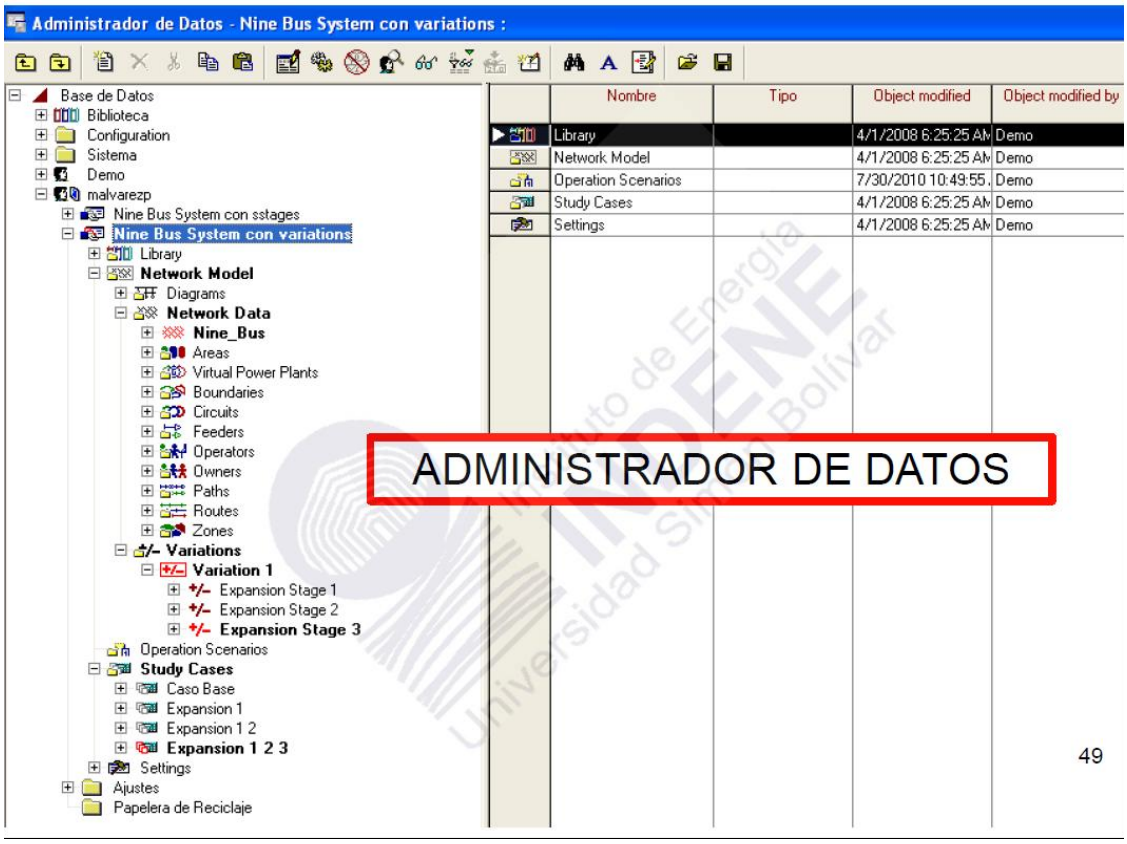


Figura 3.1 Ejemplo del Administrador de Datos de DigSilent (Alvarez, 2012).

A través de esta interfaz se pueden visualizar y modificar todos lo elementos que conforman las redes del sistema, lo cual es una gran ventaja en comparación con otros programas que se manejan únicamente con el diagrama unifilar del sistema, ya que permite modificar y estudiar los datos más fácilmente.

3.1.1.2 Exportación de los datos desde DigSilent.

Dado que DigSilent es un software privado, las bases de datos de cada proyecto que se crean en él tienen un formato tal que sólo pueden ser utilizadas en éste programa. Sin embargo, el administrador de datos permite ver la información de los elementos del sistema en forma de tablas que se clasifican según su contenido en; Datos Flexibles, Datos Básicos, Flujo de Carga, Corto Circuito Completo, entre otras. En la Figura 3.2 podemos ver cómo se visualizan los datos de un grupo de terminales de cierto proyecto.

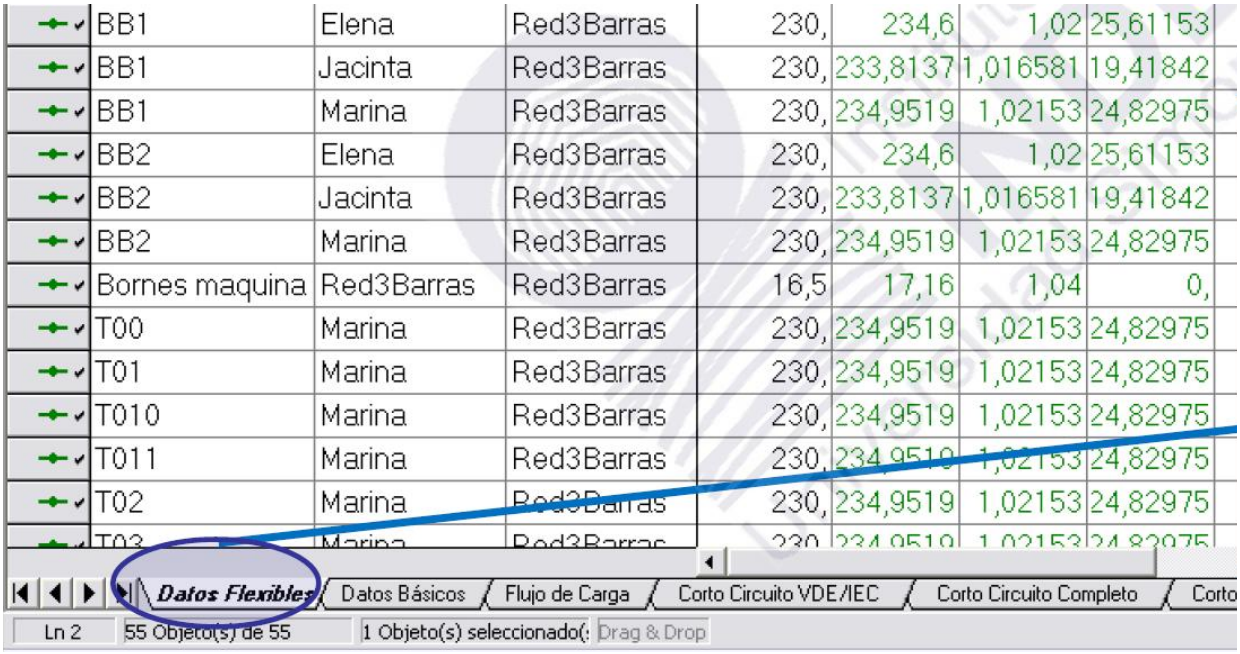


Figura 3.2 Opciones de visualización de datos en el administrador de datos de DigSilent (Alvarez, 2012)

La pestaña llamada “Datos Flexibles” y que se ve resaltada en la Figura 3.2 es sumamente importante, ya que en ella se puede elegir qué característica del componente se quieren visualizar. De esta forma si sólo se necesitan ciertos datos de un tipo de elemento en particular, se pueden obtener a utilizando esta opción.

Luego de tener los datos deseados presentes en pantalla, todos o parte de ellos pueden ser exportados hacia un archivo del tipo hoja de cálculo o pueden ser copiados incluyendo los encabezados. Para ello se seleccionan los elementos que se quieren incorporar a la hoja de cálculo, se hace click derecho sobre uno de ellos y en el menú que se despliega se elige la opción adecuada tal y como se muestra en la Figura 3.3 obtenida en (DigSILENT GmbH, 2010)

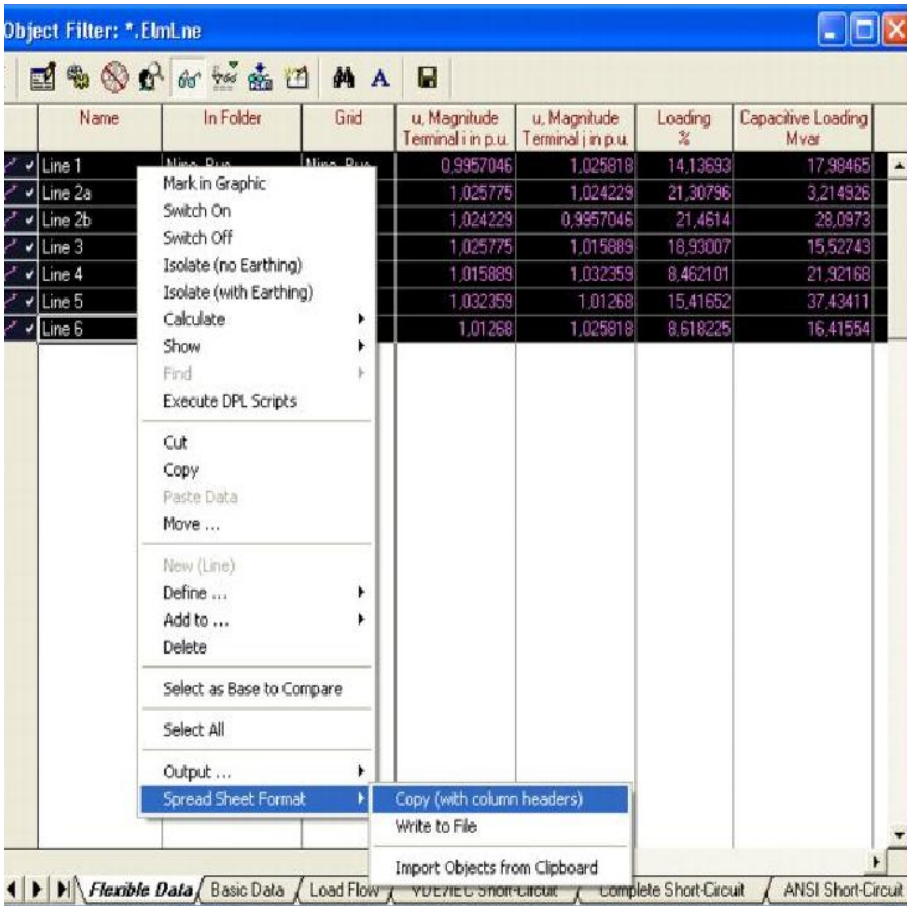


Figura 3.3 Opción para copiar datos con encabezado (DigSILENT GmbH, 2010)

En caso de seleccionar “Write to File”, es decir, escribir en un archivo, se desplegará una ventana donde se debe elegir el archivo existente en el cual se copiara los datos seleccionados.

Si bien el copiar los datos de esta forma a una hoja de cálculo es bastante ventajoso, tiene ciertos inconvenientes. Entre ellos se encuentran:

Como es de suponerse, la pestaña de datos flexibles sólo se puede utilizar para visualizar datos de un mismo tipo de componente a la vez, ya que en muchos de los casos las variables que se eligen pueden estar definidas para un tipo de elemento en concreto y no tendría sentido que se mostrara ese dato para otro tipo de componente. Esto trae como consecuencia que no se pueda copiar toda la información deseada en un solo paso, sino que se debe realizar el procedimiento tantas veces como tipos de elementos se deseen copiar.

Cuando se selecciona la opción de escribir en un archivo, se abre el mismo en un modo de escritura tal que se sobrescriben las primeras líneas del documento hasta almacenar todos los

datos que habían sido seleccionados en el programa. Esto quiere decir que si se desea crear un sólo archivo con información de diferentes elementos, deberán crearse tantos documentos como tipo de componentes sean y después unificarlo manualmente (debido a que por la particularidad mencionada en el punto anterior, si se escriben los datos de las líneas y luego los de las carga, las segundas reemplazarán la información de las primeras). Otra opción es escribir todos los datos en el documento de una sola vez, lo cual sólo es posible si no se utiliza la pestaña de datos flexibles, esto implica que se copiaría solo la información que se muestra predeterminadamente en alguna de las otras pestañas; finalmente otra alternativa sería utilizar la opción de copiar con los encabezados y pegar la información directa y manualmente en el archivo deseado un tipo de elemento a la vez.

En algunos casos la información mostrada en los recuadros del administrador de datos no son simples números reales, sino que son matrices. Luego al copiar uno de estos recuadros en una hoja de cálculo, sólo se guarda el primer elemento de la diagonal, lo cual, si no se considera puede traer errores en cálculos que se realicen con estos datos.

3.1.2. Datos del SEN en software libre.

Si bien DigSilent tiene grandes beneficios, no se debe olvidar que es un software privado y como tal tiene todas las limitaciones nombradas en la sección 2.2 de esta tesis. Es debido a estas desventajas que surge la necesidad de crear una aplicación que permita realizar la mayor cantidad de funciones posibles pero en software libre. Obviamente este proceso de creación es sumamente largo, complicado y requerirá la colaboración de grandes grupos de personas, sin embargo considerando los beneficios a obtener es un trabajo que debe ser hecho. Esta tesis pretende ser solo un pequeño aporte en un proyecto de tal envergadura para incentivar el uso de software libre para el análisis de sistemas de tal forma que en el futuro se reduzca la dependencia que se tiene actualmente de las grandes compañías y sus costosas licencias

Apuntando es estos objetivos, se decidió crear una base de datos del sistema eléctrico nacional utilizando una herramienta de software libre como lo es SQLite (disponible en www.sqlite.org). Éste permite crear bases de datos relacionales de forma rápida y sencilla de manera que fue ideal para ese trabajo.

CAPITULO IV

ESTÁNDAR CIM APLICADO AL SISTEMA ELÉCTRICO NACIONAL DE VENEZUELA

Hasta hace unos años el sistema eléctrico nacional estaba conformado por diversas empresas, entre las que se pueden mencionar CADAFE (Compañía Anónima de Administración y Fomento Eléctrico), ENELVEN (Energía Eléctrica de Venezuela), Electricidad de Caracas y por supuesto EDELCA (Electrificación del Caroní), sin embargo el gobierno nacional decidió facilitar el manejo de sistema eléctrico fusionando todas estas compañías y creando lo que hoy en día se conoce como CORPOELEC (Corporación Eléctrica Nacional). Cada una de las empresas que existían hasta ese momento, estaban encargadas de planificar, diseñar, implantar, operar y mantener una parte del SEN, por lo que debían compartir los datos entre cada uno de los departamentos en los que se ejecutaban estas acciones.

En el momento de la fusión se crea la necesidad de que los datos de toda la red estén disponibles y puedan ser analizados como un todo y no parcialmente como se hacía antes. Es en este punto donde se destaca la importancia de un modelo de información común que permita realizar fusiones de empresas o simplemente compartir datos con mayor facilidad.

Esta tesis pretende ser tan sólo un ejemplo de cómo aplicar parcialmente la norma IEC 61970 (IEC, 2003) a la red de 765kV del sistema eléctrico nacional. Se debe aclarar que se aplica sólo parcialmente ya que la norma es sumamente completa y compleja y para poder crear un archivo CIM/XML que cumpla cabalmente con los requisitos de la norma, se tendría que tener acceso a información de la red que no está disponible al público.

Dado que este trabajo se realiza con fines académicos y que los datos de la red que se tienen son limitados, se decidió crear un archivo que manejara sólo la información básica del sistema, es decir, la suficiente como para efectuar un flujo de carga clásico y obviara, por ejemplo, los datos transitorios de la máquinas, tipos de excitatrices, datos económicos.

Vale señalar que el hecho de que un documento CIM/XML no contenga todos los datos que se especifican en la norma no quiere decir que el archivo no cumpla con el estándar, ya que se debe recordar que estos son extensibles y por ello se pueden adaptar a las necesidades de cada usuario según la información que necesite.

4.1. Actualización.

Una parte primordial de esta tesis es que se quiso trabajar con los datos reales y al día del SEN. Para ello se realizó una labor de investigación con el fin de actualizar una los datos que se tenían (contenidos en un archivo de DigSilent Power Factory de principios del año 2011) hasta la fecha de aquel momento (Julio 2012).

Este proceso de actualización comprendió la inclusión de máquinas sincrónicas, líneas y reactores a la base de datos del sistema. Lamentablemente no existe forma de confirmar si el sistema que se obtuvo es totalmente fiel al sistema eléctrico actual, pero sirve para cumplir con los objetivos de esta tesis sin alejarse considerablemente de la realidad.

4.2. Extracción de datos.

Gracias a que el archivo del que se disponía contenía un caso de estudio del SEN en Power Factory, se pudo hacer uso de las opciones de extracción de datos descritas en la sección 3.1.1.2 para guardar los datos temporalmente en documento tipo un hoja de cálculo y luego introducirlos en la base de datos de SQLite.

4.3. Creación del archivo CIM/XML

Luego de tener disponible y organizada en la bases de datos la información necesaria y actualizada del sistema eléctrico se pudo proceder al proceso de escritura del archivo XML. Para ello se programó un código que creaba el documento XML y leía los datos de los diferentes elementos de la red y los transcribía en el archivo de destino utilizando el siguiente esquema:

1. Encabezado: todos los documentos del tipo XML tienen un encabezado especifico que permite al analizador identificarlos como tal y en este caso particular es de la siguiente forma:

   <?xml version="1.0" encoding="UTF-8"?>

   Evidentemente el texto se encuentra entre los signos '<' y '>' para identificar el objeto como una etiqueta (al igual que cualquier dato que se escriba en un archivo de este tipo) dentro de la cual se especifica que el documento es del tipo xml primera versión y codificado en UTF-8. La codificación permite establecer correctamente las letras y símbolos que se encuentran presentes en el archivo, de manera que de ser abiertos en otro país con una codificación diferente puedan ser leídos sin problema alguno a pesar de la configuración del idioma del computador.

2. Nombre de espacio: en segundo lugar se declara el nombre de espacio de xml (mejor conocido como xmlns) del que derivan las etiquetas del documento. Para poder identificar cada uno de los elementos de la red según el formato CIM, se debe colocar al inicio de la etiqueta una especie de prefijo que indica que el objeto es de un tipo en particular y que por ende hereda todas las características de dicho tipo, tal y como se explicó en el capítulo 1.3.1. Un ejemplo de este tipo de declaración seria:

   <cim:cim xmlns:cim="http://packages.python.org/PyCIM/">

   Esto quiere decir que existe un nombre de espacio o clase llamado "cim" que sigue los parámetros definidos en la página web que se nombre a continuación. Luego:

   <cim:case> </cim:case>

Son las etiquetas de apertura y cierre que contienen un objeto llamado “case” que es del tipo cim.

Se debe recordar que no todas las etiquetas presentes en el documento deben corresponder a un nombre de espacio en particular, sino que como el lenguaje es extensible permite ser bastante flexible en cuanto a este tema.

3. Declaración de objetos: luego de especificadas las clases a utilizar, sólo resta escribir el resto del documento siguiendo el estándar CIM y por supuesto considerando cada elemento del sistema como un objeto, es decir, que al declarar por ejemplo un generador del sistema, se le asigna un nombre, un id o número de referencia (único para cada objeto) y por supuesto cierta cantidad de atributos que lo definen.

La parte más trabajosa de la creación del documento tiene que ser, sin duda alguna, la declaración de los objetos, ya que para identificarlos correctamente se debe conocer específicamente a que clase pertenecen y como se enuncian cada uno de sus atributos de forma que cumpla con la norma y puedan ser reconocidos por cualquier persona que lea el documento. En caso de que un parámetro de la clase no sea conocido se puede obviar y se por el contrario se tiene un dato del elemento adicional a los establecidos en la norma, se puede incluir en la declaración, siempre y cuando se identifique con una etiqueta que lo represente en el archivo.

Uno de los recursos que se utilizó para poder realizar la declaración correctamente fue la documentación encontrada en la referencia (Lincoln, 2010). Richard Lincoln es el autor de un paquete de programación escrito en Python llamado PyCIM cuyo objetivo es permitir la implementación del modelo de información común de la IEC en un entorno en el cual las clases, sus principales atributos y relaciones ya se encuentran definidas. La documentación de este módulo especifica, entre otras cosas, los elementos que se encuentran en cada uno de los subpaquetes del CIM descritos en capitulo 1.4 por lo que sirvieron de guía para la elaboración de este trabajo.

De forma general se definieron los elementos básicos a declarar en el documento y los atributos de cada uno, por ejemplo para una barra cualquiera del sistema, los parámetros de interés son nombre, voltaje nominal, potencia activa y reactiva demanda y tipo de barra (PV, PQ, SLACK o aislada) entre otros. Se busco el elemento barra en la referencia especificada, y se examinaron los nombres de los atributos de esta clase, de manera que dichos nombres pudiesen ser utilizados como etiquetas al declarar el objeto y sus parámetros en el archivo. En muchas ocasiones, dado que es estándar es sumamente complejo, los elementos y sus atributos no se definen directamente en la norma, sino que se dividen en subclases cuyos atributos sí coinciden nos los datos buscados. En dichos casos se hicieron adaptaciones de manera que los elementos de interés pudiesen ser declarados sin agregar datos innecesarios al sistema.

Un ejemplo particular de este problema fueron lo transformadores, ya que en la norma estos se definen simplemente como elementos que contienen devanados y no como componentes que conduzcan corriente. Los datos nominales se les asignan a cada devanado en lugar de al transformador en sí. Esto quiere decir que para declarar un elemento de este tipo, se tenían que crear al menos tres (3) objetos, los dos devanados y el transformador que los contiene. Esto es sumamente engorroso y nada práctico para la aplicación que se está estudiando, por lo que se hicieron ciertas adaptaciones para poder establecer los datos de una forma más concisa y entendible. Otros casos de este tipo se encontraron a lo largo del proyecto y se resolvieron de la misma manera.

Se debe recordar que el estándar CIM es un formato que busca facilitar el manejo e intercambio de los datos, por lo que su aplicación se ejecuta usando el lenguaje de marcas extensibles, lo que le permite ser dinámico y por sobre todas las cosas, flexible.

CAPITULO V

APLICACIÓN DE SOFTWARE DE ANÁLISIS AL SEN

El principal objetivo de crear un documento con los datos del sistema eléctrico nacional implementando el estándar CIM, es que se tenga toda la información de la red en un solo documento que pueda ser utilizado para crear la base de datos necesaria para utilizar cierto programa de análisis de sistemas de potencia, en lugar de varias bases de datos que deban ser cambiadas de formato para funcionar con cada programa.

En líneas generales el objetivo de una base de datos del SEN es poder realizar algún tipo de análisis del sistema, es por ello que se decidió complementar esta tesis aprovechando los datos de la red para ejecutar un flujo de carga por el método de Newton Raphson, pero que al igual que el resto de este trabajo, resaltara la importancia que han ganado las herramientas de software libre.

En el capítulo 2 se discutió acerca de los dos tipos de programas disponibles para el análisis de sistemas de potencia, los de software libre y los privativos, y se nombraron algunas de las más importantes herramientas que se han desarrollado hasta el momento. Sin embargo, el lenguaje de programación en el cual se escriben los programas también es un factor a considerar ya que de esto depende en gran parte la eficiencia del programa y la facilidad o dificultad que representaría en un futuro efectuar cambios o siquiera analizar el código fuente, suponiendo que se tuviese acceso al mismo, es decir, suponiendo que la herramienta es de código abierto o software libre.

5.1. Selección del lenguaje de implementación.

Antes de poder definir qué programa o paquete computacional usar para realizar el análisis de flujo de carga, se debe establecer el lenguaje de programación del mismo.

Existen diferentes aspectos que se deben considerar en este punto, algunos de los más resaltantes son:

1. Nivel del lenguaje: de forma básica, el nivel del lenguaje determina que tan parecido al lenguaje de máquina se realiza la programación, es decir, que tan directas son las instrucciones que se ejecutan en la herramienta. En los lenguajes de bajo nivel los comandos es escriben lo más parecido posible al lenguaje de máquina o ASM (del ingles assembly language), por lo que su ejecución es directa, mientras que en los de alto nivel los algoritmos son más didácticos y se entienden con mayor facilidad por los humanos. Sin embargo todos los niveles de lenguaje, tienen sus ventajas y desventajas, o que nos lleva al siguiente punto:

2. Rapidez: Aquí se deben considerar diferentes aspectos, todos ellos dependientes del nivel de lenguaje, entre otras cosas y se pueden clasificar en tres (3):

    - Rapidez de corrida: depende, además del nivel de lenguaje y las características del harware computacional en el que se ejecute el programa, de la calidad de los algoritmos y la eficiencia del código.

    - Rapidez de compilación: es el tiempo que le toma al compilador realizar la traducción del lenguaje de programación utilizado al lenguaje de máquina.

    - Rapidez de escritura del programa: esto depende de la experiencia del programador y el nivel del lenguaje. En la mayoría de los casos el segundo factor es incluso más determinante que el primero, ya que para un mismo programa se requieren muchas más líneas de código utilizando por ejemplo lenguaje ASM que C++ o Python.

3. Disponibilidad de librerías: las librerías son paquetes computacionales en donde se definen funciones especificas que son de uso común. Estas facilitan en gran parte el trabajo a la hora de escribir un programa ya que se aprovechan los códigos creados con anterioridad. Particularmente se deben mencionar las librerías gráficas y las de

análisis numérico, que permiten generar gráficos de alta calidad en dos (2) dimensiones y manipular arreglos multidimensionales como vectores y matrices entre otras cosas.

4. Popularidad: este factor ayuda a determinar la posibilidad de encontrar documentación impresa o en línea acerca del lenguaje. Del nivel de popularidad de un lenguaje se puede definir si es conveniente usarlo o no, ya que si este nivel es muy bajo, el lenguaje tiene altas posibilidades de entrar en desuso.

Estos son algunos de los factores más relevantes a considerar, sin embargo se pueden encontrar diversas discusiones acerca del tema en diferentes fuentes. Particularmente en el anexo 14 de (INDENE-USB, 2010) se discute profundamente al respecto y se hace una comparación exhaustiva de los principales lenguajes de programación actuales. Dicho cotejo se refleja en la Figura 5.1, en la cual se evalúan ciertos criterios con el fin de calcular una puntuación que considere cada uno de ellos. Se debe acotar que dicha comparación se hizo con fines muy similares a los de esta tesis, encontrar el mejor lenguaje para los programas de análisis de sistemas de potencia.

Criterio	C/C++	Java	PHP	Python	Ruby	.NET
Medición según Tiobe	5	5	4	2	0	3
Interfaces Usuario	3	5	2	5	4	5
Interoperatividad OS	2	4	3	5	5	1
Neutralidad y Estándares	5	3	2	4	2	3
Plataformas MVC	1	4	3	5	5	1
Dinamismo	2	3	4	5	5	3
Entornos de Desarrollo	5	5	3	4	4	3
Documentación/Soporte	5	5	4	5	4	5
Componentes/Escalabilidad	5	3	4	5	4	3
Métricas/Rendimiento	3	4	3	4	3	4
Puntaje total	36	41	32	44	35	32

Figura 5.1 Comparación de lenguajes de programación. (INDENE-USB, 2010)

Donde la medición según Tiobe representa el índice de popularidad según dicha compañía para el año 2010 (acorde al número de búsquedas relacionadas con el lenguaje), interfaces de usuario califica la calidad con la que el usuario puede llegar a interactuar con el programa,

interoperatividad OS que se refiere a la capacidad del lenguaje de ser utilizado en diferentes sistemas operativos o OS (operating sytem), neutralidad tecnológica y estándares mide la independencia del lenguaje de aquellas decisiones corporativas que pudiesen afectar el proyecto en desarrollo, plataformas MVC estima la posibilidad de implementar plataformas del tipo Modelo-Vista-Controlador, el dinamismo que constituye la facilidad de uso del lenguaje, los entornos de desarrollo que igualmente son herramientas que proveen un ambiente más cómodo de escritura para el programador, documentación y soporte, referente a la disponibilidad de información para el usuario, componentes y escalabilidad que califica la cantidad de librerías que se pueden obtener y la posibilidad de llevar un programa a nuevas escalas (como consecuencia de la obsolescencia de la plataforma o del programa) y finalmente las métricas y rendimientos que califican al lenguaje según el consumo de recursos y la velocidad de ejecución de un programa comparándolo con porciones de código que cumplan la misma función y que hayan sido desarrollados en otros lenguajes.

Acorde a ese análisis se puede observar que el lenguaje que suma la mayor cantidad de ventajas es Python, aunque no haya obtenido la mayor puntuación en todas las categorías.

Además de esta comparación también se tomó en cuenta los criterios dados por el profesor Federico Milano y que expone en el capítulo 3 de su libro (Milano F. , 2010), donde califica a Python como una de las mejores opciones a la hora de realizar análisis de sistemas de potencia, entre otros motivos por considerarlo de escritura segura, dinámica y fuerte, es decir, que no permite operaciones o conversiones que conlleven a resultados inesperados, realiza el chequeo del código durante el tiempo de corrida en lugar de durante el de compilado y bloquea el éxito de operaciones que involucren argumentos con incorrecto tipo de valor; por otro lado permite la metaprogramación, lo que significa que los programas creados con este lenguaje pueden escribir o manipular otros programas, realiza introspecciones, o lo que es igual determina el tipo de objeto durante la corrida. Se menciona también la ventaja del gran número de librerías que hay disponibles, tanto de los mismos creadores de Python como de terceras personas y el beneficio que representa que sea de software libre y basado en la estructuración de clases, lo que permite crear y mantener códigos de programas orientados a objetos.

En conclusión el lenguaje de programación recomendado por diferentes expertos en el área de la ingeniería eléctrica para las aplicaciones de análisis de sistemas de potencia es Python, motivo por el cual se utilizó en esta tesis.

5.2. Implementación del módulo PyPower al SEN.

Para ejecutar el análisis de flujo de carga a la red de 765kV del sistema eléctrico nacional, se utilizó un paquete de programación de Python llamado PyPower (Lincoln, Richard ; Power System Engineering Research Center, 2009) el cual permite realizar el análisis de flujo de carga clásico y optimo de grandes redes de potencia ya sea con el método de Newton Raphson, Gauss Seidel o Fast Decoupled.

En líneas generales para correr el flujo de carga se tuvo que crear un caso de estudio particular donde los datos estuviesen en el formato adecuado. Dado que dicho caso se puede considerar un poco largo como para mostrarlo con lujo de detalle en este libro, por simplicidad se explicará el código que genera el caso de estudio del problema de cuatro (4) barras y dos generadores presente en el libro Power System Analysis (Stevenson & Grainger, 1994) y que viene predeterminadamente incluido como ejemplo en PyPower.

5.3. Formato de los casos de estudio de PyPower

Para correr un caso de estudio cualquiera, PyPower recibe como parametro lo que en Python se define como un diccionario, en el cual la información se estructura a pares de datos, es decir, se crea lo que comúnmente se conoce como una ‘llave’ y se asocia a cierto contenido. Las llaves que deben estar presentes en el caso de estudio para que el mismo sea válido son bastante intuitivas y se enumeran a continuación:

1. ‘baseMVA’: el contenido asociado con esta llave es un número real y representa la potencia base a la que se realizará el estudio expresada en MVA. Ejemplo:

```
ppc["baseMVA"] = 100.0
```

2. ‘bus’: Se relaciona con un arreglo que contiene los datos de todas las barras del sistema. La columnas de cada arreglo debe contener específicamente el siguiente orden;
    - Número de la barra (entero positivo y único para cada barra).
    - Tipo de barra: 1 si es PQ, 2 si es PV, 3 para la barra slack y 4 en caso de estar aislada.
    - Potencia activa demandada [MW].
    - Potencia reactiva demandada [MVAr].
    - Conductancia shunt demandada en la barra cuando V=1.0 p.u. [MW].
    - Susceptancia shunt inyectada en la barra cuando V=1.0 p.u. [MVAr].
    - Número del área (entero positivo).
    - Magnitud del voltaje [p.u.].
    - Ángulo del voltaje [°].
    - Voltaje base [kV].
    - Número de la zona (entero positivo).
    - Voltaje máximo permitido en la barra [p.u.].
    - Voltaje mínimo permitido en la barra [p.u.].

    La forma como se declaran las barras en el ejemplo especificado es la siguiente:

```
ppc["bus"] = array([
                [0, 3, 50,  30.99,  0, 0, 1, 1, 0, 230, 1, 1.1, 0.9],
                [1, 1, 170, 105.35, 0, 0, 1, 1, 0, 230, 1, 1.1, 0.9],
                [2, 1, 200, 123.94, 0, 0, 1, 1, 0, 230, 1, 1.1, 0.9],
                [3, 2, 80,  49.58,  0, 0, 1, 1, 0, 230, 1, 1.1, 0.9]
                 ])
```

3. ‘gen’: es también un arreglo con los datos de los generadores del sistema. El orden de los datos del arreglo debe ser el siguiente:
    - Número de la barra en la que se encuentra conectado el generador (entero positivo).
    - Potencia activa generada [MW].
    - Potencia reactiva generada [MVAr].

- Límite máximo de potencia reactiva [MVAr].
- Límite mínimo de potencia reactiva [MVAr].
- Consigna de voltaje del generador [p.u.].
- Base de potencia aparente del generador [MVA].
- Estado de servicio: mayor a cero para generadores en servicio y menor o igual a cero para generadores fuera de servicio.
- Límite máximo de potencia activa [MW].
- Límite mínimo de potencia activa [MW].
- El resto de los datos de este arreglo son referentes a características más específicas de la máquina y sus curvas de capacidad, a las cuales no se tiene acceso, por lo que se establecen en cero para ser obviadas.

Para el caso del ejemplo antes mencionado de las cuatro barras, los generadores se declaran de la siguiente manera:

```
ppc["gen"] = array([
        [3, 318, 0, 100, -100, 1.02, 100, 1, 318, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0,   0, 100, -100, 1,    100, 1, 0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        ])
```

4. ‘branch’: finalmente un último arreglo donde se especifican las características de las ramas de la red en el siguiente orden:
    - Número de la barra de salida (entero positivo).
    - Número de la barra de llegada (entero positivo).
    - Resistencia de la rama [p.u.].
    - Reactancia de la rama [p.u.].
    - Susceptancia total de la rama [p.u.].
    - Límite de potencia de régimen permanente [MVA].
    - Límite de potencia a corto plazo [MVA].
    - Límite de potencia de emergencia [MVA].
    - Relación de transformación (cero para las líneas y menor a uno (1) para los transformadores).

- Angulo de desfasaje (cero para las líneas y según el tipo de conexión para los transformadores).
- Estado de servicio: uno (1) para las ramas en servicio y cero (0) para las que se encuentran fuera de servicio.
- Ángulo mínimo de diferencia entre la fase del voltaje en la barra de salida y en la barra de destino [°].
- Ángulo máximo de diferencia entre la fase del voltaje en la barra de salida y en la barra de destino [°].

En el caso tomado como ejemplo se declaran las ramas del sistema de la siguiente manera:

```
ppc["branch"] = array([
      [0, 1, 0.01008, 0.0504, 0.1025, 250, 250, 250, 0, 0, 1, -360, 360],
      [0, 2, 0.00744, 0.0372, 0.0775, 250, 250, 250, 0, 0, 1, -360, 360],
      [1, 3, 0.00744, 0.0372, 0.0775, 250, 250, 250, 0, 0, 1, -360, 360],
      [2, 3, 0.01272, 0.0636, 0.1275, 250, 250, 250, 0, 0, 1, -360, 360]
   ])
```

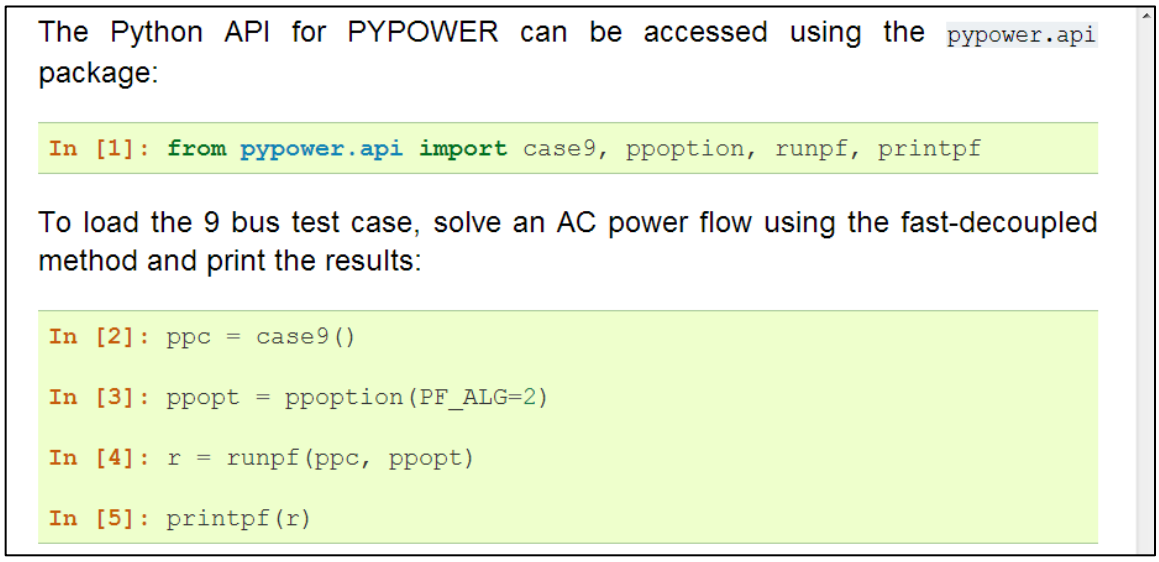
Un último elemento puede ser incluido en el diccionario en caso de que se conozcan los costos de la generación en cada una de las zonas que se definieron, sin embargo eso no es pertinente a este estudio y no fue tomado en cuenta.

Uniendo los códigos de ejemplo se construye en caso de cuatro barras y dos generadores de (Stevenson & Grainger, 1994). Dicho código se puede estudiar en su totalidad en el APÉNDICE A de esta tesis.

5.4. Módulos más importantes de PyPower

Como se explicó de forma general en el capítulo 1.5, el análisis de flujo de carga es relativamente complejo y tiene un peso computacional considerable, por ende su programación es igualmente compleja.

Si se toma como referencia las instrucciones de uso del módulo tal y como se muestran en su página web principal en el apartado etiquetado como ‘Usage’ (Lincoln, PyPower, 2011) (Figura 5.2), se ve que se utilizan aparentemente sólo dos funciones, ppoption y runpf, sin embargo si se estudia el código fuente se podrá observar que en realidad el proceso de ejecución del programa fue dividido en múltiples funciones que se importan paulatinamente en cada una de las funciones que la requiere.


The Python API for PYPOWER can be accessed using the `pypower.api` package:

```
In [1]: from pypower.api import case9, ppoption, runpf, printpf
```

To load the 9 bus test case, solve an AC power flow using the fast-decoupled method and print the results:

```
In [2]: ppc = case9()

In [3]: ppopt = ppoption(PF_ALG=2)

In [4]: r = runpf(ppc, ppopt)

In [5]: printpf(r)
```

Figura 5.2 Instrucciones básicas para correr un flujo de carga utilizando PyPower. (Lincoln, PyPower, 2011)

Para facilitar la comprensión del funcionamiento de este paquete se describirán algunas de las funciones que se consideraron más relevantes.

5.4.1. Módulo ppoption.

Esta función crea un diccionario donde se especifican las características principales del flujo de carga a ejecutar. En caso de ser llamada sin ningún parámetro, asigna los valores por defecto establecidos por su creadores, aunque siendo un software libre es posible modificar estos valores a nuestra conveniencia para futuras implementaciones. Algunos de los valores por defecto que toma la herramienta son:

- Algoritmo de flujo de carga: dado que este módulo permite usar diferentes algoritmos para realizar el flujo de carga, se debe especificar cuál de ellos se quiere utilizar. Para ello se especifica la llave 'PF_ALG' igual a uno (1) para el método de Newton Raphson (valor por defecto), sin embargo se dispone del dos (2) o del tres (3) para Fast Decoupled versión XB o BX respectivamente o cuatro (4) para el método de Gauss Siedel.

- Tolerancia del problema: como se sabe los métodos numéricos iterativos repiten el proceso de cálculo hasta que la última solución hallada difiere de la solución anterior en un factor conocido como tolerancia. En caso de que este valor no sea especificado, el módulo ppoption los define por defecto en 1e-8, es decir, 0.000001% de error. Para que el usuario defina otro valor de tolerancia, basta con invocar la función ppoption dándole como parámetro PF_TOL=x, donde x es la tolerancia deseada.

- Número máximo de iteraciones: existen casos (como los descritos en el capítulo 4.2 de (Milano F. , 2010)) en los que los datos establecidos son tales que el método numérico no es capaz de encontrar la solución al problema, por lo que de no ser limitado se ve atrapado en un lazo infinito y nunca deja de iterar. Para evitar esto se deben definir un número máximo de iteraciones, después de las cuales el programa distinga que el problema no tuvo solución y se aborte. En el caso de PyPower este número se establece en 10, pero en caso de querer aumentarlo o disminuirlo se invoca la función ppoption dándole como parámetro PF_MAX_IT=x, donde x es el número máximo de iteraciones permitidas. Se debe acotar que un número máximo de iteraciones muy bajo puede perjudicar el desempeño del programa ya que podría incurrir en abortar la ejecución antes de llegar a la solución.

5.4.2. Módulo runpf

Este módulo es el que se encarga en sí de correr el flujo de carga. Recibe cuatro (4) parámetros:

- Caso se estudio: comprende el diccionario con los datos del sistema tal y como se describió en la sección 5.3. En caso de que este parámetro no sea dado a la función, se importará el ejemplo académico basado en el sistema de nueve (9) barras y tres (3) generadores de Joe Chow.

- Características del flujo de carga: se especifican en un diccionario con las asignaciones respectivas al tipo de algoritmo, tolerancia, etc. De forma general este diccionario se debe crear antes de invocar a la función runpf haciendo uso de la función ppoption, pero en caso de que no haya sido así, dentro del código runpf se incluye una llamada a ppoption para asignar los valores por defecto.

- Nombre para archivo de salida caso 1: de ser dado runpf crea un archivo con dicho nombre, o sobrescribe en caso de existir, en el cual imprime los datos de salida del flujo de carga, es decir, el módulo y ángulo del voltaje y la potencia activa y reactiva consumida y demandada en cada barra y los flujos de potencia circulando por las ramas. Si este parámetro no es dado, los resultados simplemente se imprimen por pantalla en lugar de crear un documento con ellos.

- Nombre para archivo de salida caso 2: en el segundo caso disponible para la salida de datos, se crea un archivo de extensión .py que contiene un código que permite crear un caso de estudio con los datos del sistema incluyendo la solución del problema, es decir, un archivo muy similar al del APÉNDICE A pero con cuatro (4) columnas extras en el arreglo de las ramas para incluir los datos de potencia activa y reactiva inyectados desde la barra de salida y los recibidos en la barra de destino, y los voltajes y potencias generadas no son los asignados antes del flujo de carga, sino los resultados del mismo.

Este módulo se puede dividir en tres (3) etapas básicas, lectura y adecuación de datos, procesamiento del flujo de carga y salida final.

Durante la lectura y adecuación de datos, se reestructuran los arreglos de las barras, ramas y generadores, de manera que estén ordenados de una forma más conveniente para el programa. Entre otras cosas se eliminan momentáneamente las barras aisladas y los generadores y ramas

que estén fuera de servicio, se reenumeran las barras empezando en cero y se organizan los generadores según en número de la barra en la que estén conectados. Una llave adicional es agregada al diccionario del caso de estudio para que al terminar el análisis los resultados puedan ser expresados al usuario utilizando los números de barras originales.

Esta reestructuración de los datos le permite al programa hacer uso de (entre muchos otros) dos módulos en particular, que son makeYbus.py y makeSbus.py. Como sus nombres en ingles lo indican, estos módulos sirven para crear la matriz de admitancias y de inyecciones de potencia asociadas al sistema, respectivamente. Como se debe recordar estos son datos básicos para ejecutar el flujo de carga. Además de estos dos módulos durante las primeras etapas del programa se invocan otros se leen los datos del sistema y crean dos arreglos donde se identifican los números de las barras PQ, los números de las barras PV y se aísla en número de la barra slack.

Luego de tener el sistema bien estructurado, se utilizan los datos contenidos en el diccionario ppopt creado con la función ppoption para establecer el procedimiento a seguir y seleccionar el módulo adecuado para el análisis acorde a los requerimientos dados. Si el usuario lo específica se puede ejecutar un flujo de carga de corriente directa o DCPF por sus siglas en ingles, antes del flujo de carga AC, obteniendo un punto inicial más efectivo y en muchos casos reduciendo el número de iteraciones necesarias para hallar la solución. De lo contrario simplemente se utiliza el módulo newtonpf.py (en caso de desear ejecutar el método de Newton Raphson) para resolver el problema de flujo de carga clásico.

El módulo newtonpf.py contiene programado el método numérico por el cual lleva su nombre y recibe todos los datos de la red más los del diccionario ppopt para resolver el problema del flujo de carga. Vale acotar que no recibe directamente el diccionario con los datos del sistema sino la matriz de admitancias y de inyecciones de potencia y los datos de los tipos de barras del sistema. Este módulo devuelve los datos de la última iteración y una variable que especifica si el proceso de iteración fue exitoso o no, es decir, si el error fue menor a la tolerancia.

Finalmente teniendo los resultados del flujo de carga se procede a actualizar los datos del sistema, es decir, asignar en el diccionario de la red los valores de voltaje y potencia obtenidos e imprimirlos en pantalla o en documentos alternos según se haya especificado, no sin antes hacer

uso de la llave especial mencionada anteriormente para estructurar los datos de las ramas, barras y generadores tal y cómo se enumeraron por el usuario.

Nuevamente se debe resaltar la importancia y las facilidades que brindan a los usuarios las herramientas de software libre, ya que como se pudo ver en este caso, el acceso al código fuente tiene un aporte vital a esta tesis y permite estudiar la estructura del programa de flujo de carga e inclusive en del método numérico sin ningún tipo de restricciones. Más aún, existe la posibilidad de cambiar datos del sistema de manera que se establezcan los valores por defecto que el consumidor prefiera o también imprimir en pantalla variables que de lo contrario serían totalmente internas al programa, como por ejemplo la matriz de admitancias o los vectores que especifican las barras del tipo PV o PQ entre otras cosas. Esto no sería ni remotamente posible en una herramienta privativa.

CAPITULO VI

RESULTADOS

6.1. Implementación del estándar CIM al SEN.

Para almacenar los datos de la red de 765kV del sistema eléctrico nacional implementando la norma IEC-61970, se generaron cinco códigos en Python. El primero de ellos creaba el archivo XML y le coloca el encabezado, luego invoca gradualmente a los otros cuatro, que se encargan de leer la información desde la base de datos y transcribirla correctamente al documento XML correspondiente. Al implementar este código se obtuvo un documento con más de tres (3) mil líneas que por razones obvias no se puede incluir completamente dentro de éste trabajo, sin embargo se mostrara parte de él para dar una idea general al lector de cómo deben estructurarse los datos en un archivo CIM/XML.

6.1.1. Encabezado.

El encabezado del archivo CIM/XML es tal y como se muestra en la Figura 6.1 y muestra, además de los elementos mencionados en a sección 4.3, los datos básicos que identifican la red estudiada; se le asigna un nombre, una potencia base en caso de requerir realizar cálculos en por unidad y especifica la red como un sistema de transmisión.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cim:cim xmlns:cim="http://packages.python.org/PyCIM/">
3  <cim:case>
4  <cim:id>SEN_765kV</cim:id>
5  <cim:networkCategory>Transmission</cim:networkCategory>
6  <cim:Core.BasePower>
7      <cim:basePower>100</cim:basePower>
8      <cim:Core.Unit>MVA</cim:Core.Unit>
9  </cim:Core.BasePower>
```

Figura 6.1 Encabezado del archivo CIM/XML

6.1.2. Elementos del sistema.

En el cuerpo del documento se declararon por separado las subestaciones, generadores, transformadores y líneas. Además se aprovecharon los atributos de la programación orientada a objetos para definir tipos de generadores, transformadores y líneas, de forma que se redujo el tamaño del archivo sin perder información.

6.1.2.1 Subestaciones o barras.

```
<cim:Core.Substationt>
    <cim:Core.IdentifiedObject>
        <cim:Core.IdentifiedObject.aliasname>Guri_765</cim:Core.IdentifiedObject.aliasname>
        <cim:Core.IdentifiedObject.name>id_Guri_765</cim:Core.IdentifiedObject.name>
        <status>0</status>
    </cim:Core.IdentifiedObject>
    <sistem>AC</sistem>
    <phases>ABC</phases>
    <in_s-e>id_Guri</in_s-e>
    <bustype>SWING</bustype>
    <cim:Core.Substationt.VoltageLevels>
        <cim:BaseVoltage>765</cim:BaseVoltage>
        <cim:Core.Unit>kV</cim:Core.Unit>
    </cim:Core.Substationt.VoltageLevels>
    <cim:LoadModel.LoadGroup>
        <cim:ActivePowerDemand>
            <cim:ActivePower>100</cim:ActivePower>
            <cim:Core.Unit>MW</cim:Core.Unit>
        </cim:ActivePowerDemand>
        <cim:ReactivePowerDemand>
            <cim:ReactivePower>100</cim:ReactivePower>
            <cim:Core.Unit>MVAr</cim:Core.Unit>
        </cim:ReactivePowerDemand>
    </cim:LoadModel.LoadGroup>
</cim:Core.Substationt>
```

Figura 6.2 Ejemplo de datos de barras en el archivo CIM/XML

6.1.2.2 Generadores y tipos de generadores.

La Figura 6.3 y la Figura 6.4 muestran un ejemplo de la estructura de un generador del sistema y su tipo de generador correspondiente en el archivo CMI/XML. Crear los tipos de generadores, transformadores y barras disminuye el tamaño del documento ya que se evita repetir la información una y otra vez. Normalmente los sistemas eléctricos contienen elementos comprados a una misma empresa o en general con características similares, al definir un tipo de elemento se le especifican los parámetros y luego simplemente se hace referencia a él en lugar de agregar los mismos parámetros una y otra vez.

```
<cim:gen_list>
    <cim:Dynamics.Generators>
        <cim:Core.IdentifiedObject>
            <cim:Core.IdentifiedObject.aliasname>Guri_gen_01</cim:Core.IdentifiedObject.aliasname>
            <cim:Core.IdentifiedObject.name>id_Guri_gen_01</cim:Core.IdentifiedObject.name>
            <status>0</status>
        </cim:Core.IdentifiedObject>
        <cim:ConnectivityNode>
            <idRef>id_Guri_g01</idRef>
        </cim:ConnectivityNode>
        <cim:Core.Generators.VoltageLevels>
            <cim:BaseVoltage>18</cim:BaseVoltage>
            <cim:Core.Unit>kV</cim:Core.Unit>
        </cim:Core.Generators.VoltageLevels>
        <VoltageSetpoint>
            <setpointVoltage>1.03</setpointVoltage>
            <cim:Core.Unit>p.u.</cim:Core.Unit>
        </VoltageSetpoint>
        <ActivePowerSetpoint>
            <ActivePower>100</ActivePower>
            <cim:Core.Unit>MW</cim:Core.Unit>
        </ActivePowerSetpoint>
        <ReactivePowerSetpoint>
            <ReactivePower>100</ReactivePower>
            <cim:Core.Unit>MVAr</cim:Core.Unit>
        </ReactivePowerSetpoint>
        <central>Central_Hidro_Guri</central>
        <casa_maq>C.M._I</casa_maq>
        <gentype>
            <idRef>symtip_Guri_1-3</idRef>
        </gentype>
    </cim:Dynamics.Generators>
</cim:gen_list>
```

Figura 6.3 Ejemplo de datos de un generador en el archivo CIM/XML.

```
<gentype>
    <cim:Core.IdentifiedObject.name>symtip_Guri_1-3</cim:Core.IdentifiedObject.name>
    <RatedPower>
        <ratedS>185</ratedS>
        <unit>MVA</unit>
    </RatedPower>
    <PowerFactor>0.95</PowerFactor>
    <connectionType>YN</connectionType>
    <cim:OperationalLimit.ActivePowerLimits>
        <cim:MaxActivePower>175.75</cim:MaxActivePower>
        <cim:Core.Unit>MW</cim:Core.Unit>
    </cim:OperationalLimit.ActivePowerLimits>
    <cim:OperationalLimit.ReactivePowerLimits>
        <cim:MaxReactivePower>185</cim:MaxReactivePower>
        <cim:MinReactivePower>-185</cim:MinReactivePower>
        <cim:Core.Unit>MW</cim:Core.Unit>
    </cim:OperationalLimit.ReactivePowerLimits>
    <Impedance>
        <xd>0.01308</xd>
        <xq>0.01308</xq>
        <cim:Core.Unit>p.u.</cim:Core.Unit>
    </Impedance>
</gentype>
```

Figura 6.4 Ejemplo de datos de tipo un generador en el archivo CIM/XML.

6.1.2.3 Líneas y tipos de líneas.

En la Figura 6.5 y Figura 6.6 se muestra un ejemplo de línea y tipo de línea del documento.

```
<cim:Wires.ACLineSegment>
  <cim:Core.IdentifiedObject>
    <cim:Core.IdentifiedObject.aliasname>Guri_765_to_Malena_765</cim:Core.IdentifiedObject.aliasname>
    <cim:Core.IdentifiedObject.name>Linea_1</cim:Core.IdentifiedObject.name>
    <status>0</status>
  </cim:Core.IdentifiedObject>
  <cim:ConnectivityNode>
    <idRef>id_Guri_765</idRef>
  </cim:ConnectivityNode>
  <cim:ConnectivityNode>
    <idRef>id_Terminal_Guri_1</idRef>
  </cim:ConnectivityNode>
  <cim:Wire.Conductor>
    <cim:length>152.8</cim:length>
    <cim:Core.Unit>km</cim:Core.Unit>
  </cim:Wire.Conductor>
  <lineType>
    <idRef>lnetyp_001_L1</idRef>
  </lineType>
</cim:Wires.ACLineSegment>
```

Figura 6.5 Ejemplo de datos de una línea en el archivo CIM/XML.

```
<lineType>
    <cim:Core.IdentifiedObject.name>lnetyp_001_L1</cim:Core.IdentifiedObject.name>
    <RatedCurrent>
        <ratedI>3.916</ratedI>
        <unit>kA</unit>
    </RatedCurrent>
    <RatedVoltage>
        <voltage>765</voltage>
        <unit>kV</unit>
    </RatedVoltage>
    <Impedance>
        <resistance>
            <r>0.01308</r>
            <unit>ohmperkm</unit>
        </resistance>
        <reactance>
            <x>0.01308</x>
            <unit>ohmperkm</unit>
        </reactance>
        <susceptance>
            <b>0.01308</b>
            <unit>uSperkm</unit>
        </susceptance>
    </Impedance>
</lineType>
```

Figura 6.6 Ejemplo de datos de un tipo de línea en el archivo CIM/XML.

6.1.2.4 Transformadores de potencia y tipos de transformadores.

Finalmente en la Figura 6.7 y la Figura 6.8 se ejemplifican las secciones del documento donde se describen un transformador y un tipo de transformador respectivamente.

```
<trx_list>
    <cim:Wires.PowerTransformer>
        <cim:Core.IdentifiedObject>
            <cim:Core.IdentifiedObject.aliasname>Guri_g01_to_Guri_230</cim:Core.IdentifiedObject.aliasname>
            <cim:Core.IdentifiedObject.name>id_Guri_trx_gen_1</cim:Core.IdentifiedObject.name>
            <status>0</status>
        </cim:Core.IdentifiedObject>
        <cim:Wires.TransformerWinding>
            <side>LV</side>
            <cim:ConnectivityNode>
                <idRef>id_Guri_g01</idRef>
            </cim:ConnectivityNode>
        </cim:Wires.TransformerWinding>
        <cim:Wires.TransformerWinding>
            <side>HV</side>
            <cim:ConnectivityNode>
                <idRef>id_Guri_230</idRef>
            </cim:ConnectivityNode>
        </cim:Wires.TransformerWinding>
        <trxType>
            <idRef>tr2typ elev. Guri 1-3</idRef>
        </trxType>
        <tap_info>
            <tap_position>0</tap_position>
            <tap_side>HV</tap_side>
        </tap_info>
    </cim:Wires.PowerTransformer>
</trx_list>
```

Figura 6.7 Ejemplo de datos de un transformador en el archivo CIM/XML.

```
<trxType>
    <cim:Core.IdentifiedObject.name>tr2typ elev. Guri 1-3</cim:Core.IdentifiedObject.name>
    <RatedPower>
        <ratedS>212</ratedS>
        <unit>MVA</unit>
    </RatedPower>
    <RatedHV>
        <HV>230</HV>
        <unit>kV</unit>
    </RatedHV>
    <RatedLV>
        <HV>18</HV>
        <unit>kV</unit>
    </RatedLV>
    <connectionType>YNd0</connectionType>
    <x>0.0833</x>
    <cim:Wire.TapChanger>
        <side>HV</side>
        <cim:stepVoltageIncrement>2.5</cim:stepVoltageIncrement>
        <cim:highStep>3</cim:highStep>
        <cim:lowStep>-1</cim:lowStep>
    </cim:Wire.TapChanger>
    <cim:Wire.TapChanger>
        <side>LV</side>
        <cim:stepVoltageIncrement>0</cim:stepVoltageIncrement>
        <cim:highStep>0</cim:highStep>
        <cim:lowStep>-0</cim:lowStep>
    </cim:Wire.TapChanger>
</trxType>
```

Figura 6.8 Ejemplo de datos de un tipo de transformador en el archivo CIM/XML.

6.2. Flujo de Carga clásico con PyPower.

Como se menciono anteriormente los archivos que se generan con la solución del flujo de carga son un poco extensos por lo que para poder expresar los resultados de una manera más ordenada, se realizó un estudio de menores dimensiones cuyos resultados se muestran en esta sección.

Todas las tablas incluidas en este capítulo fueron generadas por PyPower luego de correr el programa. Para mantener la información lo más fiel posible las únicas modificaciones que se les hicieron fueron de formato, más el contenido es el mismo (razón por la cual son en idioma ingles).

La Tabla 1 es un resumen de la red, donde se especifica el número total de barras, generadores, cargas, potencia generada, entre otras cosas.

Tabla 1 Resumen de la red.

How many?		How much?	P (MW)	Q (MVAr)
Buses	24	Total Gen Capacity	8494.5	-4330.0 to 5830.0
Generators	17	On-line Capacity	5987.0	-3080.0 to 4130.0
Committed Gens	12	Generation (actual)	5445.8	-1036.2
Loads	5	Load	5394.2	1813.4
Fixed	5	Fixed	5394.2	1813.4
Dispatchable	0	Dispatchable	0.0 of 0.0	0.0
Shunts	0	Shunt (inj)	0.0	0.0
Branches	30	Losses (I^2 * Z)	51.60	2039.33
Transformers	20	Branch Charging (inj)	-	4888.9
Inter-ties	0	Total Inter-tie Flow	0.0	0.0
Areas	1			

Tabla 2En la Tabla 6.2 se muestran los valores máximos y mínimos de voltaje y en que barras ocurren. Además se especifican las ramas donde ocurren las mayores pérdidas de potencia activa y reactiva.

Tabla 2 Condiciones mínimas y máximas.

	Minimum	Maximum
Voltage Magnitude	0.970 p.u. @ bus 11	1046p.u. @bus 22
Voltage Angle	-26.03 deg @ bus 25	3.78deg @bus 4
P Losses (I^2*R)	-	7.65MW@line22-23
Q Losses (I^2*X)	-	199.72MVAr@line22-23

El flujo de carga permite obtener el estado de los generadores, lo cual permite determinar si se encuentran trabajando fuera de sus valores nominales o en un rango que pueda causar daños a la máquina. En la Tabla 3 se pueden observar la potencia activa y reactiva inyectada por cada generador del sistema.

Tabla 3 Condición de los generadores.

Gen	Bus	Status	Pg	Qg
	#	#	(MW)	(MVAr)
1	2	1	301.36	-181.86
2	4	1	651.20	-150.45
3	6	1	620.40	-154.15
4	7	1	634.80	-152.44
6	9	1	631.70	-152.81
7	10	1	590.70	-157.55
8	11	1	618.40	-179.44
10	15	1	210.70	-88.58
12	17	1	207.70	-33.15
13	18	1	323.40	71.06
15	20	1	326.10	71.39
16	21	1	329.30	71.79
		Total:	5445.76	-1036.20

Las últimas dos tablas se pueden considerar como las más importantes que devuelve el programa, ya que es donde se muestran los valores del módulo y ángulo de los voltajes de las barras (como se muestran en la Tabla 4) y se calculan los flujos de potencia por las ramas (Tabla 5)

Tabla 4. Voltajes y potencia generada y consumida en cada barra.

Bus	Voltage		Generation		Load	
#	Mag(pu)	Ang(deg)	P(MW)	Q (MVAr)	P(MW)	Q (MVAr)
1	1.011	-3.242	-	-	-	-
2	0.975	0.000	-	-	-	-
3	1.000	0.000	-	-	-	-
4	0.975	3.777	651.20	-150.45	-	-
6	0.975	3.444	620.40	-154.15	-	-
7	0.975	3.600	634.80	-152.44	-	-
8	1.000	0.000	-	-	-	-
9	0.975	3.566	631.70	-152.81	-	-
10	0.975	3.122	590.70	-157.55	-	-
11	0.970	3.457	618.40	-179.44	-	-
12	1.000	0.000	-	-	-	-
13	1.033	-3.559	-	-	38.93	3.60
14	1.020	-4.301	-	-	1527.18	43.53
15	1.000	1.148	210.70	-88.58	-	-
16	1.000	0.000	-	-	-	-
17	1.000	1.209	207.70	-33.15	-	-
18	1.040	2.659	323.40	71.06	-	-
19	1.000	0.000	-	-	-	-
20	1.040	2.717	326.10	71.39	-	-
21	1.040	2.786	329.30	71.79	-	-
22	1.046	-9.679	-	-	-	-
23	1.045	-18.568	-	-	1774.30	589.92
24	1.015	-23.865	-	-	194.76	527.00
25	1.010	-26.031	-	-	1858.99	649.33
		Total:	5445.76	-1036.20	5394.16	1813.38

Tabla 5 Resultado de flujos de potencia por las ramas.

Brnch	From	To	From Bus Injection		To Bus Injection		Loss (I^2 * Z)	
#	Bus	Bus	P (MW)	Q (MVAr)	P (MW)	Q (MVAr)	P (MW)	Q (MVAr)
0	1	22	1315.03	-591.14	-1308.79	299.21	6.239	162.93
1	1	22	1315.03	-591.14	-1308.79	299.21	6.239	162.93
2	1	22	1249.60	-584.17	-1243.67	260.31	5.928	154.83
3	22	23	1288.98	-285.64	-1281.34	-205.76	7.648	199.72
4	22	23	1288.98	-285.64	-1281.34	-205.76	7.648	199.72
5	22	23	1283.28	-287.44	-1275.66	-207.91	7.614	198.84
6	23	25	1125.34	-0.54	-1119.42	-471.56	5.924	154.71
7	23	24	938.69	30.04	-935.42	-473.50	3.280	94.23
8	24	25	740.66	-53.50	-739.57	-177.77	1.083	28.29
9	1	13	56.30	42.59	-56.30	-42.10	-0.000	0.49
10	1	13	56.30	42.59	-56.30	-42.10	-0.000	0.49
11	1	13	56.30	42.59	-56.30	-42.10	-0.000	0.49
12	1	8	0.00	0.00	0.00	0.00	0.000	0.00
13	1	2	-301.36	205.97	301.36	-181.86	-0.000	24.11
14	1	4	-651.20	237.38	651.20	-150.45	-0.000	86.93
15	1	10	-590.70	230.28	590.70	-157.55	-0.000	72.73
16	1	3	0.00	0.00	0.00	0.00	0.000	0.00
17	1	12	0.00	0.00	0.00	0.00	0.000	0.00
18	1	6	-620.40	233.68	620.40	-154.15	-0.000	79.53
19	1	7	-634.80	235.39	634.80	-152.44	-0.000	82.94
20	1	9	-631.70	235.02	631.70	-152.81	-0.000	82.20
21	1	11	-618.40	260.96	618.40	-179.44	-0.000	81.52
22	14	16	0.00	0.00	0.00	0.00	0.000	0.00
23	14	15	-210.70	113.22	210.70	-88.58	-0.000	24.64
24	14	17	-207.70	54.02	207.70	-33.15	-0.000	20.87
25	14	19	0.00	0.00	0.00	0.00	0.000	0.00
26	14	18	-323.40	-30.76	323.40	71.06	-0.000	40.30
27	14	20	-326.10	-30.42	326.10	71.39	-0.000	40.97
28	14	21	-329.30	-30.03	329.30	71.79	-0.000	41.76
29	14	13	-129.98	-119.56	129.98	122.71	-0.000	3.15
						Total:	51.602	2039.33

Para complementar los resultados dados por el programa se debe tener acceso a la base de datos donde se especifican no solo el número de la barra sino también en nombre de dicha barra en el mundo real, lo cual es indispensable para poder hacer uso de los resultados. En el estudio

mostrado en éste capítulo cada barra corresponde a los terminales de los generadores o a las subestaciones según se especifica en la Tabla 6

Tabla 6 Relación entre el número del id y el nombre real de la barra.

id	Barra
1	Guri_765
2	Guri_g11
3	Guri_g12
4	Guri_g13
6	Guri_g14
7	Guri_g15
8	Guri_g16
9	Guri_g17
10	Guri_g18
11	Guri_g19
12	Guri_g20
13	Guri_B_400
14	Guri_A_400
15	Guri_g04
16	Guri_g05
17	Guri_g06
18	Guri_g07
19	Guri_g08
20	Guri_g09
21	Guri_g10
22	Malena_765
23	San_Geronimo_765
24	O.M.Z._765
25	La_Horqueta_765

CONCLUSIONES Y RECOMENDACIONES

En la actualidad las herramientas computacionales son imprescindibles para el análisis de sistemas de potencia, por lo que la gran mayoría de las empresas se ven obligadas a adquirir licencias que le permitan utilizan algún software para realizar los procedimientos pertinentes, ya sean de diseño, planificación, implantación, operación o mantenimiento de una red. Sin embargo cada uno de estos programas trabaja con un formato de almacenamiento de datos particular, normalmente exclusivo de la empresa proveedora. Esto impide utilizar la información generada por uno de ellos en otros programas, lo que obliga a la compañía a utilizar productos provenientes de un mismo distribuidor o en el peor de los casos a generar la base de datos de forma paralela para poder utilizar diferentes software en el análisis de un mismo sistema. Esto trae como consecuencia que se tengan grandes inconvenientes en el manejo de los datos, incurriendo en la creación de múltiples copias de una misma información sólo para que sea compatible con cada programa.

En el caso particular de Venezuela este problema se evidenció por el hecho de que el sistema eléctrico nacional se encontraba a cargo de aproximadamente veinte (20) compañías diferentes, cada una de ellas con su propia base de datos almacenada en un formato en particular. Por ello, a la hora de integrar los datos para conocer el estado general de la red, ya fuese antes de después de la fusión que unió todas estas empresas convirtiéndolas en Corpoelec, se generaban grandes complicaciones que dificultaban el trabajo.

En la búsqueda para resolver este problema que se daba a nivel mundial, el Instituto de Investigación para la Energía Eléctrica, mejor conocido como EPRI por sus siglas en ingles (Electric Power Research Institute) elaboró el modelo de información común CIM que se convirtió en la norma 61970 de la Comisión Electrotécnica Internacional o IEC. Este estándar permite aprovechar los beneficios de los lenguajes de etiquetas extensibles o XML para almacenar los datos de los sistemas eléctricos de potencia en un formato que es internacionalmente aceptado y facilita el manejo de la información.

Los archivos XML son mundialmente conocidos, en especial desde que el incremento de las aplicaciones web resaltó la utilidad de los lenguajes de etiquetas. Además existen múltiples

analizadores que con la configuración adecuada muestran el contenido del archivo de una forma muy eficiente y organizada. Es debido a estos motivos, entre muchos otros, que este lenguaje fue seleccionado como el apropiado para la implementación del estándar.

Este proyecto de grado se basó en la aplicación de éste modelo a los datos del sistema eléctrico nacional, específicamente a la red de 765kV compuesta por las subestaciones Yaracuy, La Arenosa, La Horqueta, O.M.Z., San Gerónimo, Malena y el complejo Generador Simón Bolívar, mejor conocido como Guri.

La aplicación del estándar CIM a todo el sistema eléctrico nacional permitiría adaptar los datos a la normativa internacional, lo que representaría una paso más para mantener al país a la vanguardia tecnológica en el área de la electricidad y facilitaría el manejo de los datos de la red.

Lamentablemente en la actualidad existen políticas de restricción con respecto a la publicación de los datos del sistema eléctrico, por lo que la población en general no tiene acceso a ellos. Esto trae como consecuencia que cualquier persona ajena al estado que desee realizar un trabajo de investigación o desarrollo tecnológico en base al SEN, deberá remitirse a informes oficiales de años anteriores o información publicada por los medios de comunicación que no es del todo confiable.

Sumado a esto se tiene que la complejidad del estándar es tal que la implementación de la norma se vio limitada a una menor parte del sistema de lo que en realidad se pretendía, ya que este modelo es tan extenso y abarca tanta información, que sería imposible para personas ajenas al estado aplicarlo considerando toda la información que en él se plantea, pues se deben conocer datos que son sumamente específicos y que en general sólo son conocidos por aquellas personas que llevan mucho tiempo manejando la información del sistema. Sin embargo la importancia de la estandarización del SEN no debe ser dejada de lado sólo por estas limitaciones, sino que sería ideal plantear un proyecto de mayor envergadura donde se forme un grupo de trabajo que pueda tener acceso a los datos completos, actualizados y reales de red para poder aplicar la norma a cabalidad y aprovechar todos sus beneficios.

Se debe resaltar que el modelo CIM no es en sí un estándar de bases de datos, sino un formato bajo el cual estructurar archivos de texto donde se almacene la información de un sistema eléctrico de la forma más completa posible. En base a este documento se pueden construir las bases de datos en el formato adecuado y con la data pertinente para un programa de análisis en particular.

Pese a las limitaciones nombradas, durante la elaboración de este proyecto se consiguió elaborar un archivo que contiene los datos básicos de la red de 765kV del sistema eléctrico venezolano bajo la estructura del lenguaje de etiquetas siguiendo los parámetros expuestos en (Lincoln, PyCIM, 2010).

Una de las principales ventajas de este formato es que permite tener toda la información almacenada en un solo archivo, de esta manera se puede mantener el control de un proyecto aunque involucre la participación de varios grupos de trabajo de diferentes departamentos. Sin embargo en el caso de aplicaciones académicas y algunas de investigación, la implementación de este formato no representa grandes beneficios, por el contrario si los datos son reducidos, no está planteado compartirlos con otras empresas y/o se tiene seguridad de que sólo se va a utilizar un único software para el análisis del sistema, la aplicación del modelo representaría una complicación innecesaria.

Uno de los análisis más relevantes que se le realizan a una red es el de flujo de carga, que permite conocer los voltajes de las barras y la consigna de los generadores (entre otras cosas) en el régimen permanente bajo ciertas condiciones de carga. Esto sirve para saber si el sistema tendrá parámetros aceptables o no en esa situación.

En años anteriores el peso computacional necesario para ejecutar este análisis era tal que se tuvieron que desarrollar métodos alternativos para hallar la solución, sin embargo hoy en día la capacidad de procesamiento de las computadoras es tan grande que la preocupación ya no es cómo lograr correr un flujo de carga, sino de qué forma es más eficiente. En lo referente a este tema se tomaron como referencia diferentes fuentes y se llegó a la conclusión de que los programas escritos en Python son algunos de los más apropiados a la hora de desarrollar herramientas de análisis de sistemas de potencia, ya que la cantidad de librerías disponibles en

este lenguaje, tanto matemáticas como gráficas, se encuentran en aumento, su popularidad es cada vez mayor, pueden ser utilizados en diferentes sistemas operativos, tienen un alto nivel de rendimiento, soportan aplicaciones del tipo modelo-vista-controlador, etc. Debido a estos motivos se utilizó un programa para resolver problemas de flujos de carga escrito en Python cuyos resultados arrojados fueron medianamente satisfactorios, ya que al compararlos con los de Power Factory (disponibles en el Apendice C) no coincidirieron al 100% sino que hubo un pequeño porcentaje de error (<5% tanto para módulo como para ángulo del voltaje) sin embargo se presume que dicho error viene dado por posibles inconsistencias de los datos ya que al correr el flujo de carga en Power Factory, el programa asigna valores por defecto y hace simplificaciones que sólo los usuarios experimentados son capaces de reconocer y modificar.

Si bien algunos programas de software libre no están tan avanzados ni son tan sofisticados como otros de propietario, con el tiempo se ha ido incrementando el interés en ellos, haciendo que muchas personas incluso consideren al software libre como un derecho que tiene la sociedad, lo que puede suponer que en los próximos años la calidad y cantidad de programas de este tipo van a aumentar significativamente. Además este tipo de herramienta tiene la ventaja académica de permitir al usuario estudiar y modificar el código, lo que le permite al estudiante analizar el algoritmo de solución ya sea con intenciones de mejorarlo o reproducirlo.

En conclusión el estándar CIM es un formato que permite el almacenamiento de datos de una forma eficiente y completa y que facilita el intercambio de información sin necesidad de transformar la data de un formato de base de datos a otro, lo que se traduce en un ahorro de tiempo, dinero y herramientas computacionales, pero que requiere de un alto conocimiento de la red en estudio y de un consenso mundial para que todos los programas de análisis de ahora en más incluyan la opción de exportar e importar archivos CIM.

REFERENCIAS BIBLIOGRÁFICAS

Alvarez, M. (Marzo de 2012). Presentación para PDVSA. Curso DigSilent. Caracas, Venezuela.

Bray, T., Paoli, J., Sperberg-McQueen, C., & Maler, E. (2006). Extensible Markup Language (XML). *1.1 Second Edition*. Word Wide Web Consortium.

CORPOELEC. (2009). *Sistema de Transmisión a 765kV EDELCA.* Caracas.

Culebro, M., Gómez, W., & Torres, S. (2006). *Software libre vs software propietario. Ventajas y desventajas.* Creative Commons.

deVos, A., Widergren, S., & Zhu, J. (s.f.). XML for CIM Model Exchange.

DigSILENT GmbH. (Julio de 2010). Power Factory Manual .

Dunstan, L. A. (Enero de 1947). Machine Computation of Power Network Performance. *Transactions of the American Institute of Electrical Engineers* , 610-624.

Fouad, A. A., & Anderson, P. M. (2003). *Power System Control and Stabibity* (2da Edición ed.). Wiley-Interscience.

Gómez-Expósito, A., Conejo, A. J., & Cañizares , C. (2009). *Electric Energy Systems. Analysis and Operation.* CRC Press.

IEC. (Noviembre de 2003). IEC 61970 Energy Management system application program interface (EMS-API)-Part 301:Common Information Model (CIM) Base. *Edición 1.0*.

IEC Draft. (s.f.). IEC 61968 Application integration at electric utilities- System interfaces for distribution management- Part 11: Common Information Model (CIM).

INDENE-USB. (2010). *Anteproyecto para el desarrollo de herramienta computacional para estudios de planificación corto-mediano plazo de sistemas eléctricos de distribución.* Caracas.

Lincoln, R. (13 de Noviembre de 2010). *PyCIM*. Recuperado el 18 de Agosto de 2012, de http://packages.python.org/PyCIM/

Lincoln, R. (2011). *PyPower*. Recuperado el Septiembre de 2012, de http://www.pypower.org/

Lincoln, Richard ; Power System Engineering Research Center. (2009). *PyPower.* Recuperado el Septiembre de 2012, de http://pypi.python.org/pypi/PYPOWER/4.0.1

Mc Morran , A. W. (2007, Enero). An Introduction to IEC 61970-301 & 61968-11: The Common Information Model. Glasgow, Reino Unido.

Milano, F. (2010). *Power System Modelling and Scripting.* Springer.

Milano, F., Zhou, M., & Hou, G. (2009). Open Model For Exchanging Power System Data. *IEEE PES General Meeting.*

Ministerio del Poder Popular para la Energía Eléctrica. (2010). *Anuario Estadístico del Sector Eléctrico Nacional.* Caracas.

Ministerio del Poder Popular para la Energía Eléctrica. (2010). *Proyectos de Generación. Periodo 2010-2015. Informe de Seguimiento de Obras.* Caracas.

Ministerio del Poder Popular para la Energía Eléctrica. (2011). *Memoria y Cuenta 2011.Tomo I.* Caracas.

Ochoa, F. J. (Marzo de 2012). Estudios de Cortocircuito balanceado en sistemas de distribución sobre una plataforma libre. Caracas, Miranda, Venezuela.

OPSIS. (2007). *Sistema Eléctrico Nacional.* Caracas.

Pacheco, J. R. (s.f.). Fundamentos sobre generación, transporte y distribución de energía eléctrica. Caracas, Venezuela.

Seese, W. S., & Daub, G. W. (2005). *Química.* Pearson Educación.

Stallman, R. M. (2004). *Software libre para una sociedad libre.* Traficantes de Sueños.

Stevenson, W. (1975). *Análisis de Sistemas Eléctricos de Potencia.* McGraw-Hill.

Stevenson, W., & Grainger, J. (1994). *Power System Analysis.* McGraw-Hill.

Wu, J., & Schulz, N. (s.f.). Overview of CIM-Oriented Database Design and Data Exchanging in Power System Applications.

Xtensible Solutions. (2008). Welcome to the CIM User Group. *CIM User Group Meeting.* Vesteras, Sweden.

APÉNDICE A

Ejemplo de elementos presentes en un archive tipo CIM/XML

```
<?xml version="1.0" encoding="UTF-8"?>
<cim:cim xmlns:cim="http://packages.python.org/PyCIM/">
<cim:case>
<cim:id>SEN_765kV</cim:id>
<cim:netwockCategory>Transmission</cim:netwockCategory>
<cim:Core.BasePower>
    <cim:basePower>100</cim:basePower>
    <cim:Core.Unit>MVA</cim:Core.Unit>
</cim:Core.BasePower>

<cim:substations_list>
    <cim:Core.Substationt>
        <cim:Core.IdentifiedObject>
    <cim:Core.IdentifiedObject.aliasname>Guri_765</cim:Core.IdentifiedObject.aliasname>
    <cim:Core.IdentifiedObject.name>id_Guri_765</cim:Core.IdentifiedObject.name>
            <status>0</status>
        </cim:Core.IdentifiedObject>
        <sistem>AC</sistem>
        <phases>ABC</phases>
        <in_s-e>id_Guri</in_s-e>
        <bustype>SWING</bustype>
        <cim:Core.Substationt.VoltageLevels>
            <cim:BaseVoltage>765</cim:BaseVoltage>
            <cim:Core.Unit>kV</cim:Core.Unit>
        </cim:Core.Substationt.VoltageLevels>
        <cim:LoadModel.LoadGroup>
            <cim:ActivePowerDemand>
                <cim:ActivePower>100</cim:ActivePower>
                <cim:Core.Unit>MW</cim:Core.Unit>
            </cim:ActivePowerDemand>
            <cim:ReactivePowerDemand>
                <cim:ReactivePower>100</cim:ReactivePower>
                <cim:Core.Unit>MVAr</cim:Core.Unit>
            </cim:ReactivePowerDemand>
        </cim:LoadModel.LoadGroup>
    </cim:Core.Substationt>
</cim:substations_list>

<cim:gen_list>
    <cim:Dynamics.Generators>
        <cim:Core.IdentifiedObject>
```

```
        <cim:Core.IdentifiedObject.aliasname>Guri_gen_01</cim:Core.IdentifiedObject.aliasna
me>
        <cim:Core.IdentifiedObject.name>id_Guri_gen_01</cim:Core.IdentifiedObject.name>
                        <status>0</status>
                </cim:Core.IdentifiedObject>
                <cim:ConnectivityNode>
                        <idRef>id_Guri_g01</idRef>
                </cim:ConnectivityNode>
                <cim:Core.Generators.VoltageLevels>
                        <cim:BaseVoltage>18</cim:BaseVoltage>
                        <cim:Core.Unit>kV</cim:Core.Unit>
                </cim:Core.Generators.VoltageLevels>
                <VoltageSetpoint>
                        <setpointVoltage>1.03</setpointVoltage>
                        <cim:Core.Unit>p.u.</cim:Core.Unit>
                </VoltageSetpoint>
                <ActivePowerSetpoint>
                        <ActivePower>100</ActivePower>
                        <cim:Core.Unit>MW</cim:Core.Unit>
                </ActivePowerSetpoint>
                <ReactivePowerSetpoint>
                        <ReactivePower>100</ReactivePower>
                        <cim:Core.Unit>MVAr</cim:Core.Unit>
                </ReactivePowerSetpoint>
                <central>Central_Hidro_Guri</central>
                <casa_maq>C.M._I</casa_maq>
                <gentype>
                        <idRef>symtip_Guri_1-3</idRef>
                </gentype>
        </cim:Dynamics.Generators>
    </cim:gen_list>

    <gentype>
        <cim:Core.IdentifiedObject.name>symtip_Guri_1-3</cim:Core.IdentifiedObject.name>
        <RatedPower>
                <ratedS>185</ratedS>
                <unit>MVA</unit>
        </RatedPower>
        <PowerFactor>0.95</PowerFactor>
        <connectionType>YN</connectionType>
        <cim:OperationalLimit.ActivePowerLimits>
                <cim:MaxActivePower>175.75</cim:MaxActivePower>
                <cim:Core.Unit>MW</cim:Core.Unit>
        </cim:OperationalLimit.ActivePowerLimits>
        <cim:OperationalLimit.ReactivePowerLimits>
                <cim:MaxReactivePower>185</cim:MaxReactivePower>
                <cim:MinReactivePower>-185</cim:MinReactivePower>
```

```
                <cim:Core.Unit>MW</cim:Core.Unit>
            </cim:OperationalLimit.ReactivePowerLimits>
            <Impedance>
                <xd>0.01308</xd>
                <xq>0.01308</xq>
                <cim:Core.Unit>p.u.</cim:Core.Unit>
            </Impedance>
        </gentype>

        <branch_list>
            <cim:Wires.ACLineSegment>
                <cim:Core.IdentifiedObject>
                    <cim:Core.IdentifiedObject.aliasname>Guri_765_to_
Terminal_Guri_1_cirId1 </cim:Core.IdentifiedObject.aliasname>

            <cim:Core.IdentifiedObject.name>Linea_1</cim:Core.IdentifiedObject.name>
                    <status>0</status>
                </cim:Core.IdentifiedObject>
                <cim:ConnectivityNode>
                    <idRef>id_Guri_765</idRef>
                </cim:ConnectivityNode>
                <cim:ConnectivityNode>
                    <idRef>id_Terminal_Guri_1</idRef>
                </cim:ConnectivityNode>
                <cim:Wire.Conductor>
                    <cim:length>152.8</cim:length>
                    <cim:Core.Unit>km</cim:Core.Unit>
                </cim:Wire.Conductor>
                <lineType>
                    <idRef>lnetyp_001_L1</idRef>
                </lineType>
            </cim:Wires.ACLineSegment>
        </branch_list>

        <lineType>
            <cim:Core.IdentifiedObject.name>lnetyp_001_L1</cim:Core.IdentifiedObject.name>
            <RatedCurrent>
                <ratedI>3.916</ratedI>
                <unit>kA</unit>
            </RatedCurrent>
            <RatedVoltage>
                <voltage>765</voltage>
                <unit>kV</unit>
            </RatedVoltage>
            <Impedance>
                <resistance>
                    <r>0.01308</r>
                    <unit>ohmperkm</unit>
```

```
                </resistance>
                <reactance>
                        <x>0.01308</x>
                        <unit>ohmperkm</unit>
                </reactance>
                <susceptance>
                        <b>0.01308</b>
                        <unit>uSperkm</unit>
                </susceptance>
        </Impedance>
    </lineType>

    <trx_list>
        <cim:Wires.PowerTransformer>
                <cim:Core.IdentifiedObject>

        <cim:Core.IdentifiedObject.aliasname>Guri_g01_to_Guri_230_cirId1</cim:Core.Identifi
edObject.aliasname>

        <cim:Core.IdentifiedObject.name>id_Guri_trx_gen_1</cim:Core.IdentifiedObject.name>
                        <status>0</status>
                </cim:Core.IdentifiedObject>
                <cim:Wires.TransformerWinding>
                        <side>LV</side>
                        <cim:ConnectivityNode>
                                <idRef>id_Guri_g01</idRef>
                        </cim:ConnectivityNode>
                </cim:Wires.TransformerWinding>
                <cim:Wires.TransformerWinding>
                        <side>HV</side>
                        <cim:ConnectivityNode>
                                <idRef>id_Guri_230</idRef>
                        </cim:ConnectivityNode>
                </cim:Wires.TransformerWinding>
                <trxType>
                        <idRef>tr2typ elev. Guri 1-3</idRef>
                </trxType>
                <tap_info>
                        <tap_position>0</tap_position>
                        <tap_side>HV</tap_side>
                </tap_info>
        </cim:Wires.PowerTransformer>
    </trx_list>

    <trxType>
        <cim:Core.IdentifiedObject.name>tr2typ elev. Guri 1-
3</cim:Core.IdentifiedObject.name>
        <RatedPower>
```

```
            <ratedS>212</ratedS>
            <unit>MVA</unit>
        </RatedPower>
        <RatedHV>
            <HV>230</HV>
            <unit>kV</unit>
        </RatedHV>
        <RatedLV>
            <HV>18</HV>
            <unit>kV</unit>
        </RatedLV>
        <connectionType>YNd0</connectionType>
        <x>0.0833</x>
        <cim:Wire.TapChanger>
            <side>HV</side>
            <cim:stepVoltageIncrement>2.5</cim:stepVoltageIncrement>
            <cim:highStep>3</cim:highStep>
            <cim:lowStep>-1</cim:lowStep>
        </cim:Wire.TapChanger>
        <cim:Wire.TapChanger>
            <side>LV</side>
            <cim:stepVoltageIncrement>0</cim:stepVoltageIncrement>
            <cim:highStep>0</cim:highStep>
            <cim:lowStep>-0</cim:lowStep>
        </cim:Wire.TapChanger>
    </trxType>

    </cim:case>
    </cim:cim>
```

APÉNDICE B.

Formato de los casos de estudio de PyPower. Caso cuatro (4) barras.

```
# Copyright (C) 1996-2011 Power System Engineering Research Center
# Copyright (C) 2010-2011 Richard Lincoln
#
# PYPOWER is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published
# by the Free Software Foundation, either version 3 of the License,
# or (at your option) any later version.
#
# PYPOWER is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with PYPOWER. If not, see <http://www.gnu.org/licenses/>.

"""Power flow data for 4 bus, 2 gen case from Grainger & Stevenson.
"""

from numpy import array

def case4gs():
    """Power flow data for 4 bus, 2 gen case from Grainger & Stevenson.
    Please see L{caseformat} for details on the case file format.

    This is the 4 bus example from pp. 337-338 of I{"Power System Analysis"},
    by John Grainger, Jr., William Stevenson, McGraw-Hill, 1994.

    @return: Power flow data for 4 bus, 2 gen case from Grainger & Stevenson.
    """
    ppc = {"version": '2'}

    ##-----  Power Flow Data  -----##
    ## system MVA base
    ppc["baseMVA"] = 100.0

    ## bus data
    # bus_i type Pd Qd Gs Bs area Vm Va baseKV zone Vmax Vmin
```

```
ppc["bus"] = array([
   [0, 3, 50,  30.99,  0, 0, 1, 1, 0, 230, 1, 1.1, 0.9],
   [1, 1, 170, 105.35, 0, 0, 1, 1, 0, 230, 1, 1.1, 0.9],
   [2, 1, 200, 123.94, 0, 0, 1, 1, 0, 230, 1, 1.1, 0.9],
   [3, 2, 80,  49.58,  0, 0, 1, 1, 0, 230, 1, 1.1, 0.9]
])

## generator data
# bus, Pg, Qg, Qmax, Qmin, Vg, mBase, status, Pmax, Pmin, Pc1, Pc2,
# Qc1min, Qc1max, Qc2min, Qc2max, ramp_agc, ramp_10, ramp_30, ramp_q, apf
ppc["gen"] = array([
   [3, 318, 0, 100, -100, 1.02, 100, 1, 318, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
   [0, 0,   0, 100, -100, 1,    100, 1, 0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
])

## branch data
#fbus, tbus, r, x, b, rateA, rateB, rateC, ratio, angle, status, angmin, angmax
ppc["branch"] = array([
   [0, 1, 0.01008, 0.0504, 0.1025, 250, 250, 250, 0, 0, 1, -360, 360],
   [0, 2, 0.00744, 0.0372, 0.0775, 250, 250, 250, 0, 0, 1, -360, 360],
   [1, 3, 0.00744, 0.0372, 0.0775, 250, 250, 250, 0, 0, 1, -360, 360],
   [2, 3, 0.01272, 0.0636, 0.1275, 250, 250, 250, 0, 0, 1, -360, 360]
])

return ppc
```

APÉNDICE C.

Resultados del flujo de carga en Power Factory de parte de la red de 765kV.

Nombre	[%] Carga	Barra	P [MW]	Q [Mvar]
Guri_02	96.97	Guri g02	178.400	-18.822
Guri_03	94.79	Guri g03	175.300	-4.532
Guri_04	91.61	Guri g04	210.700	-0.323
Guri_06	91.96	Guri g06	207.700	-40.000
Guri_07	85.93	Guri g07	323.400	45.096
Guri_09	78.39	Guri g09	326.100	45.423
Guri_10	79.16	Guri g10	329.300	45.815
Guri_11	89.43	Guri g11	596.632	-189.466
Guri_13	90.14	Guri g13	601.832	-189.466
Guri_14	85.95	Guri g14	571.032	-189.466
Guri_15	87.90	Guri g15	585.432	-189.466
Guri_17	87.48	Guri g17	582.332	-189.466
Guri_18	81.93	Guri g18	541.332	-189.466
Guri_19	85.68	Guri g19	569.032	-189.466

	Voltaje Nominal	Resultado	Angulo
	[V]	[p.u.]	[°]
Guri g04	18.00	1.00	-1.74
Guri g05	18.00		
Guri g06	18.00	1.01	-1.99
Guri g07	18.00	1.04	-0.52
Guri g08	18.00		
Guri g09	18.00	1.04	-0.46
Guri g10	18.00	1.04	-0.39
Guri g11	18.00	0.98	17.60
Guri g12	18.00		
Guri g13	18.00	0.98	17.60
Guri g14	18.00	0.98	17.61
Guri g15	18.00	0.98	17.61
Guri g16	18.00		
Guri g17	18.00	0.98	17.61
Guri g18	18.00	0.98	17.62
Guri g19	18.00	0.98	17.61
Guri g20	18.00		
Guri B	400.00	1.04	-6.68
Guri A	400.00	1.03	-7.41
Guri 765	765.00	1.02	-6.35
La Horqueta	765.00	1.03	-28.50
O.M.Z.	765.00	1.04	-26.42
San Gerónimo	765.00	1.06	-21.31
Malena	765.00	1.06	-12.67

APÉNDICE D.

Archivo de salida del flujo de carga en formato pypower.

```
def estudio_parcial_765kV():

    ## PYPOWER Case Format : Version 2
    ppc = {'version': '2'}

    ##-----  Power Flow Data  -----##
    ## system MVA base
    ppc['baseMVA'] = 100

    ## bus data
    # bus_i type Pd Qd Gs Bs area Vm Va baseKV zone Vmax Vmin
    ppc['bus'] = array([
        [1, 1, 0, 0, 0, 0, 1, 1.011125, -3.241942, 765, 1, 1.05, 0.95],
        [2, 3, 0, 0, 0, 0, 1, 0.975, 0, 18, 1, 1.05, 0.95],
        [3, 2, 0, 0, 0, 0, 1, 1, 0, 18, 1, 1.05, 0.95],
        [4, 2, 0, 0, 0, 0, 1, 0.975, 3.7772342, 18, 1, 1.05, 0.95],
        [6, 2, 0, 0, 0, 0, 1, 0.975, 3.4436929, 18, 1, 1.05, 0.95],
        [7, 2, 0, 0, 0, 0, 1, 0.975, 3.5996053, 18, 1, 1.05, 0.95],
        [8, 2, 0, 0, 0, 0, 1, 1, 0, 18, 1, 1.05, 0.95],
        [9, 2, 0, 0, 0, 0, 1, 0.975, 3.5660365, 18, 1, 1.05, 0.95],
        [10, 2, 0, 0, 0, 0, 1, 0.975, 3.1222793, 18, 1, 1.05, 0.95],
        [11, 2, 0, 0, 0, 0, 1, 0.97, 3.4565498, 18, 1, 1.05, 0.95],
        [12, 2, 0, 0, 0, 0, 1, 1, 0, 18, 1, 1.05, 0.95],
        [13, 1, 38.93, 3.6, 0, 0, 1, 1.0327412, -3.5592149, 400, 1, 1.05, 0.95],
        [14, 1, 1527.18, 43.53, 0, 0, 1, 1.020351, -4.30131, 400, 1, 1.05, 0.95],
        [15, 2, 0, 0, 0, 0, 1, 1, 1.1482488, 18, 1, 1.05, 0.95],
        [16, 2, 0, 0, 0, 0, 1, 1, 0, 18, 1, 1.05, 0.95],
        [17, 2, 0, 0, 0, 0, 1, 1, 1.2085844, 18, 1, 1.05, 0.95],
        [18, 2, 0, 0, 0, 0, 1, 1.04, 2.6587912, 18, 1, 1.05, 0.95],
        [19, 2, 0, 0, 0, 0, 1, 1, 0, 18, 1, 1.05, 0.95],
        [20, 2, 0, 0, 0, 0, 1, 1.04, 2.7171908, 18, 1, 1.05, 0.95],
        [21, 2, 0, 0, 0, 0, 1, 1.04, 2.7864146, 18, 1, 1.05, 0.95],
        [22, 1, 0, 0, 0, 0, 1, 1.0460482, -9.6792151, 765, 1, 1.05, 0.95],
        [23, 1, 1774.3, 589.92, 0, 0, 1, 1.0448341, -18.567636, 765, 1, 1.05, 0.95],
        [24, 1, 194.76, 527, 0, 0, 1, 1.0149054, -23.864603, 765, 1, 1.05, 0.95],
        [25, 1, 1858.99, 649.33, 0, 0, 1, 1.0101786, -26.031428, 765, 1, 1.05, 0.95],
    ])

    ## generator data
    # bus Pg Qg Qmax Qmin Vg mBase status Pmax Pmin Pc1 Pc2 Qc1min Qc1max Qc2min
Qc2max ramp_agc ramp_10 ramp_30 ramp_q apf
```

```
ppc['gen'] = array([
    [2, 301.362, -181.86, 350, -200, 0.975, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [3, 0, 0, 350, -200, 0.975, 700, 0, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [4, 651.2, -150.45, 350, -200, 0.975, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [6, 620.4, -154.151, 350, -200, 0.975, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [7, 634.8, -152.444, 350, -200, 0.975, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [8, 0, 0, 350, -200, 0.975, 700, 0, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [9, 631.7, -152.815, 350, -200, 0.975, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [10, 590.7, -157.548, 350, -200, 0.975, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [11, 618.4, -179.439, 350, -200, 0.97, 700, 1, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [12, 0, 0, 350, -200, 0.98, 700, 0, 630, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [15, 210.7, -88.5816, 230, -230, 1, 230, 1, 218.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [16, 0, 0, 230, -230, 1, 230, 0, 218.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [17, 207.7, -33.1495, 230, -230, 1, 230, 1, 218.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [18, 323.4, 71.0603, 380, -380, 1.04, 380, 1, 342, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [19, 0, 0, 420, -420, 1.04, 420, 0, 399, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [20, 326.1, 71.3914, 420, -420, 1.04, 420, 1, 399, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [21, 329.3, 71.7873, 420, -420, 1.04, 420, 1, 399, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
])

## branch data
# fbus tbus r x b rateA rateB rateC ratio angle status angmin angmax Pf Qf Pt Qt
ppc['branch'] = array([
    [1, 22, 0.000341597, 0.00892105, 4.29814, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1315.0254, -591.1428, -1308.7865, 299.2059],
    [1, 22, 0.000341597, 0.00892105, 4.29814, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1315.0254, -591.1428, -1308.7865, 299.2059],
    [1, 22, 0.000359481, 0.00938812, 4.52317, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1249.6013, -584.1672, -1243.6728, 260.3090],
    [22, 23, 0.000502558, 0.0131247, 6.32344, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1288.9848, -285.6382, -1281.3371, -205.7559],
    [22, 23, 0.000502558, 0.0131247, 6.32344, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1288.9848, -285.6382, -1281.3371, -205.7559],
    [22, 23, 0.000504794, 0.0131831, 6.35157, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1283.2763, -287.4443, -1275.6625, -207.9087],
    [23, 25, 0.000471707, 0.012319, 5.93525, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
1125.3419, -0.5356, -1119.4179, -471.5584],
    [23, 24, 0.000367127, 0.0105488, 5.06856, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360, 360,
938.6948, 30.0360, -935.4152, -473.5035],
    [24, 25, 0.000201202, 0.00525454, 2.53163, 2995.74, 2995.74, 2995.74, 0, 0, 1, -360,
360, 740.6552, -53.4965, -739.5721, -177.7716],
    [1, 13, 0, 0.0105333, 0, 1500, 1500, 1500, 0.975, 0, 1, -360, 360, 56.3033, 42.5907, -
56.3033, -42.1025],
    [1, 13, 0, 0.0105333, 0, 1500, 1500, 1500, 0.975, 0, 1, -360, 360, 56.3033, 42.5907, -
56.3033, -42.1025],
    [1, 13, 0, 0.0105333, 0, 1500, 1500, 1500, 0.975, 0, 1, -360, 360, 56.3033, 42.5907, -
56.3033, -42.1025],
    [1, 8, 0, 0.0185, 0, 800, 800, 800, 1, 0, 0, -360, 360, 0.0000, 0.0000, 0.0000, 0.0000],
```

```
        [1, 2, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -301.3621, 205.9710, 301.3621, -181.8605],
        [1, 4, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -651.2000, 237.3813, 651.2000, -150.4502],
        [1, 10, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -590.7000, 230.2831, 590.7000, -157.5484],
        [1, 3, 0, 0.0185, 0, 800, 800, 800, 1, 0, 0, -360, 360, 0.0000, 0.0000, 0.0000, 0.0000],
        [1, 12, 0, 0.0185, 0, 800, 800, 800, 1, 0, 0, -360, 360, 0.0000, 0.0000, 0.0000, 0.0000],
        [1, 6, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -620.4000, 233.6801, 620.4000, -154.1515],
        [1, 7, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -634.8000, 235.3879, 634.8000, -152.4436],
        [1, 9, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -631.7000, 235.0169, 631.7000, -152.8147],
        [1, 11, 0, 0.0185, 0, 800, 800, 800, 1, 0, 1, -360, 360, -618.4000, 260.9606, 618.4000, -179.4385],
        [14, 16, 0, 0.0471698, 0, 265, 265, 265, 1, 0, 0, -360, 360, 0.0000, 0.0000, 0.0000, 0.0000],
        [14, 15, 0, 0.0471698, 0, 265, 265, 265, 0.975, 0, 1, -360, 360, -210.7000, 113.2237, 210.7000, -88.5816],
        [14, 17, 0, 0.0471698, 0, 265, 265, 265, 1, 0, 1, -360, 360, -207.7000, 54.0166, 207.7000, -33.1495],
        [14, 19, 0, 0.0397619, 0, 420, 420, 420, 1, 0, 0, -360, 360, 0.0000, 0.0000, 0.0000, 0.0000],
        [14, 18, 0, 0.0397619, 0, 420, 420, 420, 1, 0, 1, -360, 360, -323.4000, -30.7554, 323.4000, 71.0603],
        [14, 20, 0, 0.0397619, 0, 420, 420, 420, 1, 0, 1, -360, 360, -326.1000, -30.4244, 326.1000, 71.3914],
        [14, 21, 0, 0.0397619, 0, 420, 420, 420, 1, 0, 1, -360, 360, -329.3000, -30.0285, 329.3000, 71.7873],
        [14, 13, 0, 0.0105, 0, 2000, 2000, 2000, 0, 0, 1, -360, 360, -129.9800, -119.5620, 129.9800, 122.7076],
    ])

    return ppc
```

APÉNDICE E

Código para correr el flujo de carga.

```
from pypower.api import ppoption, runpf
from generar_html import generar_html
import sqlite

def correr():
    """
    corre sqlite.caso_765 y el flujo de carga con pypower
    """

    ##Creación del caso de estudio. Dicionadio con datos##
    ppc=sqlite.caso_765()

    ##Características para el flujo de carga##
    ppopt=ppoption(PF_ALG=1, PF_MAX_IT= 15, TOL=1e-6)

    #Nombres de archivos para almacenar los resultados#
    nombre=raw_input("""Indique el nombre deseado para el archivo de
    salida   del flujo de carga: """)
    caso=raw_input('''Indique el nombre deseado para el archivo de
    salida del flujo de carga(datos y resultados formato pypower)s: ''')

    #Corrida del flujo de carga#
    r=runpf(ppc,ppopt, fname=nombre, solvedcase=caso)

    #¿Desea generar un archivo html con los resultados?#
    generar=raw_input('Desea generar el archivo con la salida html? y/n: ' )
    if generar=='y':
            caso+='.py'
            f=open(caso, 'r').read()
            respaldo=open('respaldo.py', 'w')
            respaldo.write('from numpy import array\n')
            for i in f:
                    respaldo.write(i)
            respaldo.close()
            import respaldo
            caso=caso.replace('.py', '')
            ejecutar='respaldo.%s()'%caso
            exec 'ppc=%s' %ejecutar
            generar_html(ppc)
    a=raw_input("finalizado")
```