Electrical Grid Simulator: Software de Simulación de Sistemas Eléctricos de Potencia con Orientación a la Investigación

Electrical Grid Simulator: Research-Oriented Electrical Power Systems Simulation Software

Presentación: 13 y 14 de septiembre de 2023

Ariel Loyarte
Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Centro de Investigación y Desarrollo en Ingeniería Eléctrica y Sistemas Energéticos (CIESE)
aloyarte@frsf.utn.edu.ar

Carlos Sanseverinatti
Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Centro de Investigación y Desarrollo en Ingeniería Eléctrica y Sistemas Energéticos (CIESE)
cisanseverinatti@frsf.utn.edu.ar

Emmanuel Sangoi
Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Centro de Investigación y Desarrollo en Ingeniería Eléctrica y Sistemas Energéticos (CIESE)
esangoi@frsf.utn.edu.ar

Resumen

Este trabajo describe las primeras etapas del desarrollo del Electrical Grid Simulator, un software de simulación de sistemas eléctricos de potencia destinado al ámbito de la investigación. El mismo pretende incluir funciones de cálculo habituales, como flujos de potencia y cortocircuito, y asimismo ser expandible con complementos que pueden ser incorporados por investigadores. Para ello, un sistema de extensiones provee las herramientas necesarias para añadir nuevos algoritmos, implementados como paquetes individuales, lo que facilita su distribución y la posibilidad de compartirlos con otros investigadores. Entre sus características generales, se destaca la adopción de una licencia gratuita, de código abierto y libre, compatibilidad con prácticamente la totalidad de los sistemas operativos modernos, y una interfaz gráfica ligera y sencilla de usar, con modernas líneas de diseño. A modo de ejemplo, este trabajo presenta complementos para resolver flujos óptimos de potencia y estudios de riesgo por arco eléctrico.

Palabras clave: redes eléctricas, simulación, software libre

Abstract

This work describes the first development stages of the Electrical Grid Simulator, a power system simulation software for research purposes. It is intended to include common calculation functions, such as power flow and short-circuit analysis, and also to be expandable with add-ons that can be incorporated by researchers. For this purpose, an extension system provides the necessary tools to add new algorithms, implemented as individual packages, which facilitates their distribution and the possibility of sharing them with other researchers. Among its general characteristics, it is free and it has a free and open source license, compatibility with most modern operating systems, and a light and easy-to-use graphical interface, with modern design lines. As an example, this work presents extensions to solve the optimal power flow problem and arcflash studies.

Keywords: electrical networks, simulation, free software

Introducción

En el campo de los sistemas eléctricos de potencia, es habitual el uso de software de simulación para el análisis de diversos problemas, escenarios y configuraciones de una red (Pfenninger et al., 2018). Es por ello que se ha desarrollado una amplia variedad de programas que pueden agruparse en al menos cuatro categorías, según su finalidad: a) dimensionar y proyectar la red, b) estudiar fenómenos transitorios, c) simular redes de transporte, y d) simular redes de distribución. Ejemplos reconocidos de las últimas dos categorías son: Siemens PSS/E (Palmer, 2016: 1-4), NEPLAN (NEPLAN, 2023) y ETAP (ETAP, 2023). Estos cuentan con la capacidad de resolver flujos de potencia (FP) y corrientes de cortocircuito, entre sus principales opciones, y presentan una interfaz gráfica que simplifica su uso. Sin embargo, también acusan limitaciones. En primer lugar, usan una licencia comercial; y aunque ofrezcan versiones especiales gratuitas, estas restringen el tamaño de la red y/o funcionalidades, lo que dificulta su utilización en tareas de investigación. Asimismo, son programas de código cerrado, limitados a las opciones que la empresa desarrolladora decide ofrecer, y se disponen casi exclusivamente para la plataforma Microsoft Windows.

En algunos casos, estas aplicaciones se complementan por medio de paquetes que se acoplan al programa base. Por ejemplo, ETAP posibilita importar modelos de MATLAB o combinar funciones con scripts en lenguaje Python (Kulkarni y Sontakke, 2015). De forma similar, MATLAB puede simular redes eléctricas, incorporando el complemento gratuito *MATPOWER* (Zimmerman et al., 2009), aunque carente de interfaz gráfica.

Por otra parte, se ha evidenciado en los últimos tiempos un aumento en la popularidad del lenguaje de programación Python, en su orientación al cálculo científico (Thangarajah, 2019; Nagpal y Gabrani, 2019). La tendencia ha impulsado el desarrollo de *PYPOWER*, una versión portada de *MATPOWER* para dicho lenguaje. Aunque el proyecto fue abandonado, otros surgieron partiendo de su código fuente, destacándose *PyPSA* (Brown et al., 2017) y *pandapower* (Thurner et al., 2018). Estos no operan como programas independientes, sino como librerías de modelado y simulación. Entre sus ventajas, son gratuitas y de código abierto, aportando flexibilidad en tareas de investigación. Este formato permite reutilizar algunas de sus capacidades para generar nuevos modelos y/o algoritmos orientados a resolver problemas no previstos por el paquete original. Entre sus puntos desfavorables, y en contraste con el software comercial, no cuentan con interfaz gráfica, por lo que su manejo puede no resultar intuitivo para el estudiante o investigador inexperto, y demanda el dominio de un lenguaje de programación.

En razón de la dificultad para disponer de un software gratuito con interfaz gráfica que permita la simulación de redes eléctricas en régimen estático, y al mismo tiempo resulte ampliable de acuerdo con algoritmos que surjan en contexto de actividades de investigación, se propone el desarrollo de Electrical Grid Simulator (EGS), una herramienta informática presentada en este trabajo, y para la cual se resolvió:

i) Utilizar la licencia libre MIT, que permite modificar el código fuente y reutilizarlo con cualquier propósito.
ii) Emplear componentes multiplataforma, compatibles con Microsoft Windows, GNU/Linux y Apple MacOS.
iii) Utilizar los modelos y métodos de cálculo que provee el paquete *pandapower*, dada su amplia difusión y activo ciclo de desarrollo. Su inclusión asegura buena proyección y soporte para el mediano y largo plazo.
iv) Utilizar la librería *PySide2* (Qt for Python, 2023) para generar interfaces gráficas modernas.
v) Emplear el lenguaje Python, destacado en aplicaciones de inteligencia computacional, lo que facilitaría generar extensiones abocadas a problemas de optimización, machine learning y procesamiento de datos.
vi) Diseñar un sistema de extensiones que permita empaquetarlas en el formato estándar de Python, de modo que su difusión e instalación resulte sencilla, por medio del repositorio PyPI (medio oficial, público y gratuito).

El presente trabajo describe la composición estructural de EGS y el diseño del sistema de extensiones. Este último se evalúa con dos complementos iniciales, uno destinado a resolver flujos óptimos de potencia (OPF), y otro empleado para realizar estudios de riesgo por arco eléctrico (Sangoi et al., 2022).

Desarrollo

Estructura del software

La Figura 1a esquematiza, de manera sintética, la estructura de EGS. El paquete *PySide2* se utiliza en Python para generar la interfaz gráfica y, asimismo, gestionar el proceso de manera integral, con posibilidades de generar hilos secundarios cuya ejecución en paralelo se destina a cálculos con alta demanda de cómputo. Internamente, se emplea una composición similar a la que suele ofrecer el software comercial, almacenando en memoria una variable destinada a representar el esquema gráfico de la red, y otra correspondiente al modelo. El software se vale del paradigma de orientación a objetos, de modo que cada una de estas variables presenta, a su vez, una estructura compleja con múltiples atributos (otras variables en su interior) y métodos (funciones

asociadas). El esquema unifilar de la red se construye por medio del paquete libre *NodeGraphQt* (NodeGraphQt, 2023), que provee un componente gráfico compatible con *PySide2*, de modo que puede insertarse sobre la ventana principal del programa. De este modo, la red se simboliza con un grafo en el que cada nodo representa a una barra, una línea, un generador, una carga, etc. Las conexiones (aristas del grafo), efectuadas con el puntero del mouse, permiten relacionar dichos componentes. Los nodos pueden arrastrarse con el puntero, similar a lo que ocurre con otras aplicaciones del rubro.

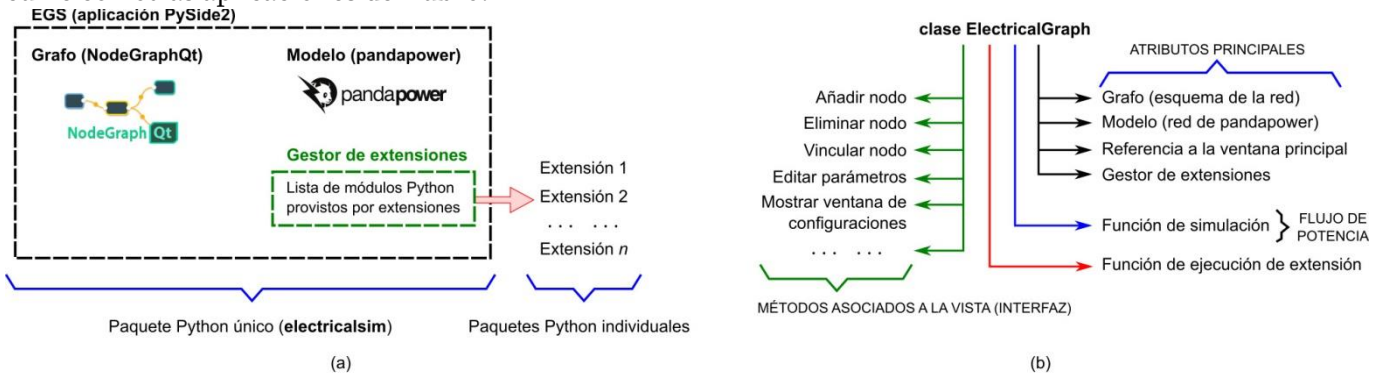

Figura 1. (a) Esquema simplificado de la estructura de EGS. (b) Componentes fundamentales de la clase **ElectricalGraph**.

El modelo de la red se construye con *pandapower*, que registra los parámetros y variables del modelo en tablas. Todos los elementos por esta soportados se disponen en EGS (21 diferentes), incluyendo varios tipos de generadores, cargas, interruptores, sistemas de almacenamiento y líneas de corriente continua. EGS sincroniza el grafo con el modelo, de forma que toda modificación en el primero se traslada automáticamente al segundo.

Un gestor de extensiones se implementa como un listado de paquetes externos que añaden otras funciones. Por un lado, EGS se dispone gratuitamente en PyPI (paquete **electricalsim**), lo que implica que su instalación es idéntica a la de cualquier paquete regular. Por otro lado, cada extensión se implementa como un paquete independiente. Si el desarrollador (investigador) optase por difundirla abiertamente, sólo tendría que compartirla en el repositorio.

Para mayores detalles, la Figura 1b resume los componentes principales de la clase **ElectricalGraph** (definición de acuerdo con la orientación a objetos). Tras su ejecución, EGS genera una variable del tipo **ElectricalGraph** que incluye el grafo (esquema), el modelo (tablas de *pandapower*), el gestor de extensiones y una referencia a la ventana principal (Figura 2), de modo que cualquier acción sobre el modelo puede tener consecuencias sobre la interfaz gráfica. Por ej., esta relación se utiliza para que los resultados de simulación se muestren indicados sobre el grafo.

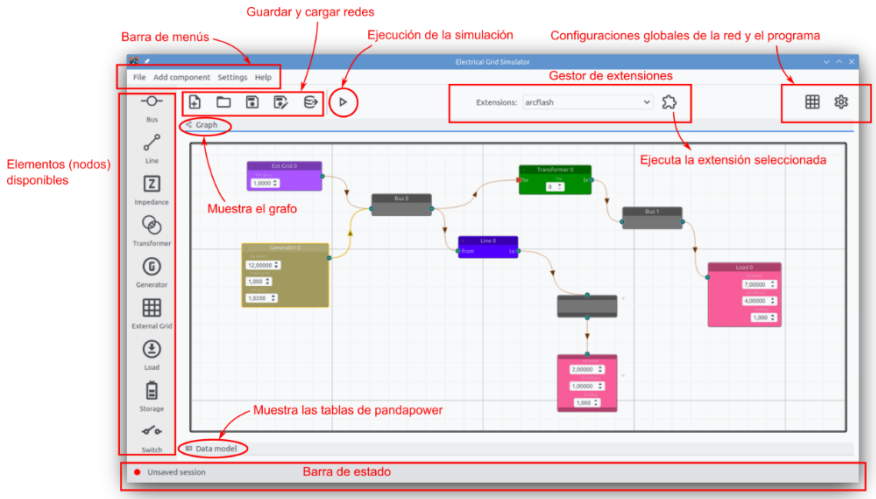

Figura 2. Ventana principal de EGS.

La función de simulación abre una ventana para computar FP y exhibir los resultados en tablas (formato de *pandapower*) y gráficos (Figura 3a); se grafican: niveles de tensión en diagramas de barra y caja, estado de carga en líneas y transformadores, caídas de tensión en líneas y potencias reactivas generadas. Junto a ellos se incorporan resúmenes estadísticos y se resaltan niveles fuera de los rangos configurados como aceptables.

Otros métodos disponibles en **ElectricalGraph** se ejecutan al accionar sobre un botón u opción de la ventana principal. Entre otras posibilidades, estos añaden nodos, los vinculan, eliminan, o permiten editarlos en una ventana (Figura 3b). Algunos nodos incluyen atajos para editar sus parámetros más importantes. Por ejemplo, la Figura 3b señala que un generador, en modo de control de tensión, contiene atajos para la potencia activa inyectada (escalable por un factor) y la tensión. Un doble click sobre el nodo muestra, en una ventana separada, todas las opciones disponibles, pudiendo incluir parámetros no utilizados en el problema del FP, pero necesarios para otras funciones de *pandapower*, incorporadas con una extensión. La función para ejecutar extensiones se detalla a continuación.

(a) (b)

Figura 3. (a) Ventana de simulación (cálculo de flujo de potencia). (b) Parametrización de un nodo en el grafo.

Sistema de extensiones

Para crear extensiones, EGS provee la definición de una clase **ExtensionBase**. El desarrollador programa una nueva clase **Extension** que deriva de esta. Luego, EGS podrá manipularla a partir de los métodos y atributos heredados:

- Un atributo booleano que solicita o rechaza el uso de la ventana estándar de extensiones. Puede optarse por usar una ventana pre-armada, ya disponible en EGS, para mostrar los resultados de la extensión.
- Un atributo booleano que solicita o rechaza la ejecución del algoritmo de cálculo en un hilo secundario. Si los cómputos son demandantes, esta opción puede ofrecer un mejor desempeño. Además, permite mostrar resultados parciales en la ventana estándar, aún si el cálculo no ha finalizado, ya que la interfaz se gestiona en el hilo principal, y el cálculo se dirige a un hilo paralelo (evita la secuencialidad).
- Un método **before_running()** que se ejecuta previo al cálculo y en el hilo principal.
- El método especial **__call__()** donde el desarrollador programa la lógica principal de la extensión, a ejecutar sobre el hilo principal o secundario, según se configure.
- Un método **finish()** que se ejecuta luego de finalizada la función **__call__()**, sobre el hilo principal.
- Otros métodos y atributos convenientes, como una referencia al objeto **ElectricalGraph**, lo que da acceso al grafo, al modelo y a la interfaz. Por esta razón, las capacidades de una extensión son ilimitadas.
- Un método **print()** para imprimir mensajes en la ventana estándar.

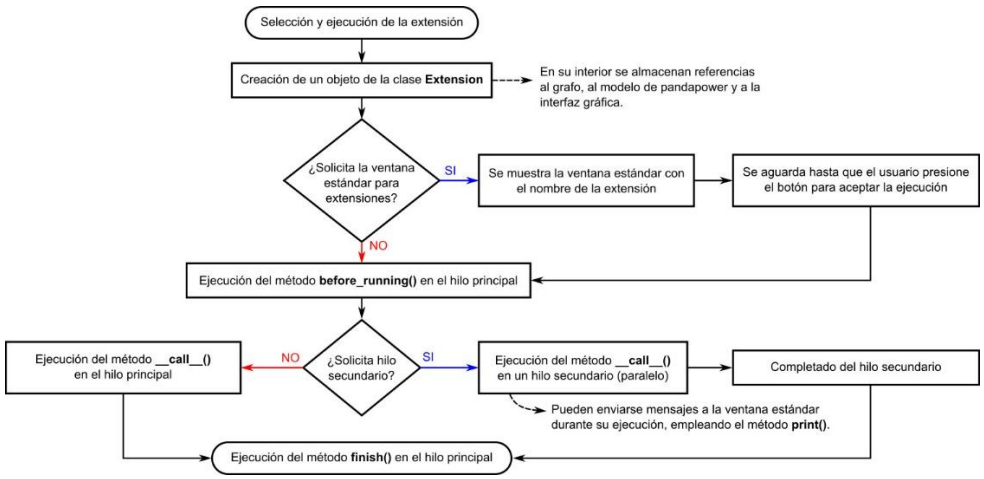

Figura 4. Lógica de ejecución de una extensión.

El investigador debe indicar los atributos booleanos citados y programar la lógica en el método **__call__()** (sobrescribir el heredado de **ExtensionBase**). El uso de los otros métodos es opcional. Cada vez que el usuario ejecuta una extensión en la barra superior de la ventana principal (Figura 2), se efectúa la secuencia de la Figura 4. El método **before_running()** se emplea normalmente para configurar valores iniciales y eventualmente mostrar una ventana de diálogo, de forma que **__call__()** se ejecute luego de aceptado el mismo. Por su parte, **finish()** se suele emplear para acciones posteriores a los resultados, como guardarlos o graficarlos sobre ventanas adicionales.

La extensión se empaqueta siguiendo el procedimiento Python oficial, por lo que puede publicarse en el repositorio PyPI, e incluye metadatos que son utilizados por EGS para detectar su instalación en forma automática.

Resultados

EGS está disponible de manera gratuita en PyPI, como así también una primera extensión destinada a OPF (paquete **electricalsim-opf-quadratic**), capaz de resolver un despacho económico de generación, con costos cuadráticos para las potencias activas y reactivas, generadas y demandadas, para las trasmitidas en líneas de corriente continua y las transferidas desde/hacia medios de almacenamiento. Aplica un método de punto interior integrado en *pandapower* para optimizar el costo global, con restricciones de carga en líneas, transformadores y generadores, límites aceptables de tensión, y balance según el modelo del FP. También ofrece una variante simplificada (linealizada). Dispone de una ventana para configurar coeficientes de costos y parámetros del método numérico (Figura 5a), mientras que utiliza la ventana estándar para presentar los resultados (Figura 5b). Dada la construcción modular impulsada por el sistema, la extensión tiene un tamaño aproximado de sólo 12 kilobytes.

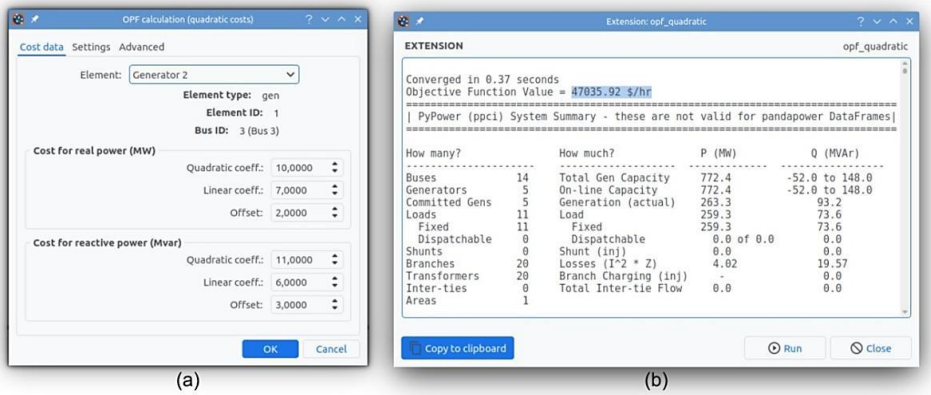

Figura 5. Ventana principal de la extensión para computar OPF (a) y resultados en la ventana estándar (b).

Las pruebas continuaron con el desarrollo de una segunda extensión, destinada a realizar estudios de riesgo por arco eléctrico. Para ello, el complemento solita los parámetros del análisis y computa corrientes mínima y máxima de cortocircuito en la barra seleccionada, a partir de las cuales determina el nivel de riesgo y genera, de manera automatizada y en formato PDF, la etiqueta de advertencia a colocar sobre el tablero, de acuerdo con la

normativa AEA 92606 (Asociación Electrotécnica Argentina, 2016). Asimismo, genera un reporte en PDF con un estudio de sensibilidad sobre los parámetros y variables más relevantes. Para mayores detalles, consultar el trabajo de Sangoi et al. (2022), que profundiza en el estudio y el algoritmo desarrollado, integrado luego como extensión en EGS.

Conclusiones

La aplicación se encuentra aún en una etapa temprana de desarrollo. Sin embargo, presenta buena estabilidad y desempeño, con buen grado de avance sobre las metas proyectadas. El sistema de extensiones es reciente y, por tanto, con posibilidades de sufrir modificaciones y añadidos en el corto plazo. No obstante, las pruebas realizadas con los complementos de OPF y estudios de riesgo por arco eléctrico exhibieron resultados satisfactorios. La propuesta demostró que el sistema de extensiones permite un desarrollo veloz, exceptuando la complejidad propia del algoritmo de cálculo o simulación, que dependerá de la aplicación concreta. Para referencia, la adaptación de la función de optimización de *pandapower* para su conversión a extensión de OPF en EGS, requirió aproximadamente 4 días de desarrollo, sin experiencia previa en este tipo de complementos y con el trabajo de un único investigador.

A futuro se prevé difundir una plantilla e instructivo que facilitarán el desarrollo de extensiones. Asimismo, se proyecta que EGS incorpore cálculos de cortocircuito según la norma IEC, como parte de las funciones de simulación.

Referencias

Asociación Electrotécnica Argentina. (2016). "AEA 92606: Arco eléctrico. Cálculo de magnitudes de los efectos térmicos y su protección".

Brown, T., Hörsch, J., and Schlachtberger, D. (2018). "PyPSA: Python for Power System Analysis". *Journal of Open Research Software*, 6(1).

ETAP - Electrical Power System Analysis & Operation Software. (Accedido: 06-2023). *ETAP Microgrid Management & Control* (https://etap.com/renewable-energy/etap-microgrid-fundamentals).

Kulkarni, S. y Sontakke, S. (2015). "Power system analysis of a microgrid using ETAP". *International Journal of Innovative Science and Modern Engineering (IJISME)*, 3(5).

Nagpal, A. y Gabrani, G. (2019). "Python for Data Analytics, Scientific and Technical Applications". *Amity International Conference on Artificial Intelligence (AICAI)*,140–145.

NEPLAN. (Accedido: 06-2023). *NEPLAN – Power System Analysis*. (https://www.neplan.ch/).

NodeGraphQt. (Accedido: 06-2023). *A node graph UI framework written in Python that can be implemented and re-purposed into applications supporting PySide2*. (http://chantonic.com/NodeGraphQt/api/index.html).

Palmer, E. (2016). "The Use of Siemens PTI PSS/E in Undergraduate Teaching at Central Queensland University", *Australasian Universities Power Engineering Conference (AUPEC)*, 1–4.

Pfenninger, S., Hirth, L., Schlecht, I., Schmid, E., F. Wiese, Brown, T., Davis, C., Gidden, M., Heinrichs, H., Heuberger, C., Hilpert, S., Krien, U., Matke, C., Nebel, A., Morrison, R., Müller, B., Pleßmann, G., Reeg, M., Richstein, J. C., Shivakumar, A., Staffell, I., Tröndle, T., y Wingenbach, C. (2018). "Opening the black box of energy modelling: Strategies and lessons learned". *Energy Strategy Reviews*, 19:63–71.

Qt for Python. (Accedido: 06-2023). *Design GUI with Python: Python Bindings for Qt*. (https://www.qt.io/qt-for-python).

Sangoi, E., Manassero, U., Loyarte, A. S., Fernandez, J. P., Rossi, L. D., y Cea, M. M. (2022). *Estudio de Riesgo por Arco Eléctrico según AEA 92606. Análisis de Sensibilidad de Resultados*. Congreso Bienal de IEEE Argentina (ARGENCON).

Thangarajah, V. (2019). "Python current trend applications-an overview". *International Journal of Advance Engineering and Research Development*, 6(10).

Thurner, L., Scheidler, A., Schäfer, F., Menke, J., Dollichon, J., Meier, F., Meinecke, S., y Braun, M. (2018). "Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems". *IEEE Transactions on Power Systems*, 33(6), 6510–6521.

Zimmerman, R. D., Murillo-Sanchez, C. E., y Thomas, R. J. (2009). "MATPOWER's extensible optimal power flow architecture". *IEEE Power Energy Society General Meeting*, 1–7.