Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Modelado y simulación de Smart Grid con OpenDSS y Matlab

Autor: Juan Pablo Claro Báez

Tutora: Ascensión Zafra Cabeza

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Modelado y simulación de Smart Grid con OpenDSS y Matlab

Autor:

Juan Pablo Claro Báez

Tutora:

Ascensión Zafra Cabeza

Profesora Contratada Doctora

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado: Modelado y simulación de Smart Grid con OpenDSS y Matlab

Autor: Juan Pablo Claro Báez
Tutora: Ascensión Zafra Cabeza

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Este trabajo representa el fin de una etapa de mi vida muy importante para mí. Atrás quedan muchos años de duro esfuerzo y dedicación. Muchos momentos y experiencias enriquecedoras que no se ven eclipsadas por etapas menos gratificantes en las que no conseguía los resultados esperados. Durante todos estos años, muchas son las personas que me han ayudado día a día para poder continuar adelante. Aunque se hace imposible nombrar a todas ellas, me gustaría agradecer profundamente a todas las personas que, de alguna manera, son parte de la culminación de este proyecto.

En primer lugar, debo agradecer a mi tutora, Ascensión Zafra, por darme la oportunidad de realizar este proyecto y por su guía en la realización del mismo. También a Luis Valverde, que con su ayuda hizo más fácil la realización de este trabajo. Además de nombrar a profesores que han dejado su huella a lo largo de la carrera, tales como Miguel Pérez-Saborid, Rafael Vázquez Valenzuela, José Manuel Gordillo, Federico Paris, Carlos Gómez Camacho y Juana María Mayo.

Agradezco también a todos mis amigos y compañeros, que he ido haciendo tanto en mi etapa en el instituto como en la Escuela de Ingeniería, porque siempre han sido un sustento y han conformado un ambiente que ha hecho posible llevar adelante mi trayectoria académica. En este punto tengo que mencionar especialmente a mi grupo de amigos *"Los Pimientos"*, con los que he pasado la mayoría de los días de estos últimos años: Jesús, Ángel, Márquez, Cobos, Javi y Rubén. Tampoco puedo pasar sin mencionar a Oihana, por estar siempre ahí y hacer más llevaderos los últimos meses de tensión.

Por último, lo más importante. Mis más sinceros agradecimientos a mi padre y a mi madre, por el enorme esfuerzo que han realizado para poner a mí disposición todo lo que he necesitado para hacer posible que esta etapa termine con éxito. Mis padres y mis hermanos, con su apoyo incondicional y sus ánimos han sido esenciales. Sin ellos, nada de esto hubiera sido posible.

A todos, gracias de corazón.
Juan Pablo Claro Báez.

Resumen

En el presente trabajo se presenta la herramienta OpenDSS, un simulador de sistemas eléctricos, para su utilización a través de Matlab en las tareas de modelado, diseño y optimización de sistemas y circuitos eléctricos. Se muestra cómo modificar un script de un circuito para adaptarlo a las necesidades del usuario, añadiendo dispositivos como centrales eólicas, plantas solares fotovoltaicas o sistemas de almacenamiento entre otros. Por último, se utiliza OpenDSS y Matlab para modelar y simular una microgrid doméstica real basada en hidrógeno y energías renovables.

Índice

Resumen III

Índice de Figuras VII
Índice de Tablas IX
Índice de Códigos XI

1. Introducción **1**
1.1. Motivaciones 1
1.2. Objetivos 2
1.3. Estructura del documento 2

2. Software para modelado y simulación **5**
2.1. Introducción. 5
2.2. Qué es OpenDSS. 5
2.2.1. Principales utilidades de OpenDSS 6
2.2.2. Modos de solución 6
2.2.3. Elementos básicos en OpenDSS 6
2.3. OpenDSS como aplicación independiente. 7
2.3.1. Script de un circuito en OpenDSS 7
Pasos previos 8
Configuración básica de elementos y dispositivos 8
Monitores, medidores y modos de solución 14
Interpretación de resultados. 15
Ejecución del código 16

3. Sistema de distribución en OpenDSS **17**
3.1. Sistema de distribución IEEE-37 17
3.1.1. Modelo Inicial 17
3.1.2. Modificación para simulación 20
3.2. Modelado e implementación de fuentes generadoras de energía renovable en OpenDSS 20
3.2.1. Modelado Planta Solar Fotovoltaica 20
Consideraciones sobre el modelo PV en OpenDSS y características generales 21
Usando el modelo 22
Soluciones estáticas. Snapshot 23
Simulaciones secuenciales en el tiempo 23
3.2.2. Implementación de una Planta Solar Fotovoltaica 23
3.2.3. Modelado de un Generador eólico 26
3.2.4. Implementación de un Generador eólico 28
3.2.5. Modelado de Baterías. *Storage Element* 29
Modos de funcionamiento. 30
3.2.6. Implementación de Baterías 33
3.3. Otros elementos 35

3.3.1. Coche eléctrico conectado a la red. PEV. 35

4. Control de OpenDSS desde Matlab 37
4.1. Control de OpenDSS por Matlab 37
4.1.1. Iniciar la interfaz COM 38
4.1.2. Compilando el circuito 38
4.1.3. Obtener información de OpenDSS desde Matlab 39
4.1.4. Elementos activos 40
4.1.5. Ejecutar comandos 40
4.1.6. Añadiendo/Editando elementos 41
4.1.7. Búsqueda de información en los circuitos 41
Trabajar con las estructuras de datos desde Matlab 42
4.1.8. Funciones para comprobaciones 42
Ejecutar la función de comprobación de circuito 43
Interpretar las comprobaciones de resultados 43
4.1.9. Creación de gráficas en OpenDSS desde Matlab 48
Interacción con el circuito 48
Edición de gráficas 49
Identificación de las gráficas 50
4.1.10. Ejemplo 51

5. Modelado y simulación de una planta experimental doméstica basada en hidrógeno renovable 53
5.1. Introducción al modelo de la Smart Grid 54
5.2. Modelado de la red en OpenDSS 55
5.2.1. Modelo de dispositivos 56
Generador fotovoltaico 56
Electrolizador 57
Pila de combustible PEM 59
Pack de baterías 59
Demandas de energía doméstica 60
5.3. Gestión del modelo de OpenDSS en Matlab 60
5.3.1. Estrategia de control 61
5.3.2. Generación y consumo de hidrógeno 63
5.3.3. Eficiencia del electrolizador y pila de combustible 64
5.4. Resultados de la simulación 66
5.4.1. Central solar 66
5.4.2. Demanda de la red 66
5.4.3. Carga de la batería 67
5.4.4. Producción de hidrógeno 67

6. Conclusiones 69
6.1. Conclusiones sobre la utilización de OpenDSS 69
6.1.1. Conclusiones sobre modelado de sistemas de generación distribuida 69
6.1.2. Conclusiones sobre modelado de Smart-Grid basada en hidrógeno 69
6.2. Objetivos futuros 70

A. Códigos de OpenDSS 73

B. Modelado en OpenDSS 77

C. Códigos de Matlab 83

Bibliografía 91

Índice de Figuras

2.1. Interfaz gráfica de OpenDSS 7
2.2. Circuito de ejemplo 8
2.3. Valores de los componentes simétricos de un conductor tipo 336 MCM ACSR 10
2.4. Matrices de impedancias 10
2.5. Definición del objeto capacitador 12
2.6. Ejecutar el script desde OpenDSS 16

3.1. Esquema básico del IEEE 37 Node Test Feder 17
3.2. Esquema del IEEE 37 Node Test Feder 20
3.3. Diagrama de bloques de un sistema con placas solares en OpenDSS 21
3.4. Resultados monitor para panel solar 25
3.5. Potencias medidas por el monitor m1 25
3.6. Representación gráfica de pérdidas en el circuito 26
3.7. Diagrama de la creación de curvas modelo para generadores eólicos 26
3.8. Potencia entregada por un generador eólico en función de la velocidad del viento 27
3.9. Circuito radial de prueba de reguladores de tensión y bancos capacitadores para un generador eólico 28
3.10. Representación gráfica de pérdidas en el circuito con sistema eólico (nodo 722) y solar (nodo 730) 29
3.11. Esquema básico del Elemento de Almacenamiento 29
3.12. Ilustración del modo de funcionamiento DEFAULT 31
3.13. Ejemplo de una simulación diaria de carga y descarga en el modo FOLLOW 32
3.14. Ejemplo de resultados de simulación 32
3.15. Storage LoadShape diseñado en Matlab 34
3.16. Nivel de carga de la batería (kWh) en una simulación diaria 34
3.17. Carga, descarga y pérdidas de la batería 34

4.1. Estructura OpenDSS 37
4.2. Comunicación Matlab y OpenDSS 38
4.3. Seleccionar un elemento con el click izquierdo 48
4.4. Seleccionar un elemento con el click izquierdo y activar vista de los nodos 49
4.5. Seleccionar un elemento con el click derecho 49
4.6. Evitar usar "Show Plot Tools" 49
4.7. Usar "Property Editor" para modificar gráficas 50
4.8. Volver a la vista por defecto 50
4.9. Gráfica por defecto y después de modificar la marca de los capacitadores con la estructura de identificación ("handles") 50

5.1. Vista del montaje experimental 53
5.2. Red doméstica basada en hidrógeno 54
5.3. Características del equipamiento experimental 55
5.4. Esquema de configuración de la planta experimental 55
5.5. Curva de irradancia utilizada en el modelo experimental 57

5.6. Potencias del electrolizador y convertidor DC/DC experimental 58
5.7. Rendimientos sistema electrolizador-convertidor DC/DC 58
5.8. Curva experimental funcionamiento conjunto electrolizador-convertidor DC/DC 59
5.9. Demandas de energía variable de la red 61
5.10. Estrategia de control por banda de histéresis. (Ulleberg, 2004) 62
5.11. Potencia entregada por los paneles solares por hora 66
5.12. Demanda de la red por hora 66
5.13. Carga de la batería durante la simulación 67
5.14. Comparación simulación OpenDSS con el documento de referencia 68

B.1. Diagrama de la creación de curvas modelo para panel solar 77
B.2. Diagrama de la creación de curvas modelo para generadores eólicos 77
B.3. Script de ejemplo para modelar un panel solar 79

Índice de Tablas

3.1. Dispositivos implementados y nodos 20
3.2. Pérdidas del circuito 24

5.1. Características eléctricas ISOFOTON 150 para irradiancia 1000 W/m^2. Datos de web del fabricante 56
5.2. Características de operación ISOFOTON 150 56
5.3. Parámetros del modelo de batería 60

B.1. Propiedades de la configuración de elementos de almacenaje 78

Índice de Códigos

3.1. Script en OpenDSS del sistema de distribución IEEE 37 sin modificar 18
3.2. Dos métodos equivalentes para la creación del objeto XYcurve 22
3.3. Definición de curvas para implementación de panel solar en OpenDSS 23
3.4. Implementación de panel solar en OpenDSS 24
3.5. Implementación de generador eólico en OpenDSS 28
3.6. Implementación de baterías en OpenDSS 33

4.1. Código de Matlab para iniciar el COM Server 51

5.1. Obtención de intensidades y almacenamiento en Matlab 60
5.2. Control por banda de histéresis en Matlab 62
5.3. Obtención de intensidades y almacenamiento en Matlab 64
5.4. Obtención de intensidades y almacenamiento en Matlab 64

A.1. Script del circuito de ejemplo 73
A.2. Script en OpenDSS del sistema de distribución IEEE 37 sin modificar 74
A.3. Dos métodos equivalentes para la creación del objeto XYcurve 75

C.1. Código de Matlab para el modelado y simulación de la planta experimental de energías renovables basada en hidrógeno 83

1 Introducción

1.1 Motivaciones

En el mundo actual, donde existe cada vez una demanda de energía mayor en cada vez más lugares, algunos de ellos aislados, se está tendiendo a la descentralización de la generación de energía. La construcción de plantas generadoras de gran tamaño no resulta actualmente la mejor opción para atender a un aumento de la demanda y se estudian otras alternativas más eficientes.

Además, se hace esencial tener presente los factores medioambientales y económicos que hacen necesario utilizar otras formas de generación y almacenamiento de energía más respetuosas con el entorno y factibles económicamente. Algunos de estos factores a tener en cuenta son:

- **Agotamiento de los combustibles fósiles.** El agotamiento gradual de los recursos energéticos fósiles, especialmente de aquellos llamados convencionales o de fácil extracción, es un proceso en marcha ampliamente reconocido. Los estudios acerca de las perspectivas de agotamiento de los diversos combustibles fósiles son relativamente abundantes en la literatura científica [1].

- **Calentamiento global y cambio climático.** El consumo de energía es el principal factor generador de emisiones de gases de efecto invernadero, especialmente CO_2, que está provocando el aumento de las temperaturas en todo el planeta (calentamiento global). Este aumento de temperaturas está generando cambios en todas las regiones geográficas: la atmósfera y los océanos se están calentando, el alcance y el volumen de la nieve y el hielo están aumentando, los niveles del mar están subiendo y los patrones climáticos están cambiando [14].

- **Conflictos bélicos y tensiones sociales.** Los principales yacimientos de combustibles fósiles, en vías de agotamiento, están concentrados en lugares del planeta muy determinados. El control y explotación de los yacimientos es causa de conflictos, guerras y tensiones sociales.

- **Desequilibrio social.** A las evidentes diferencias existentes entre los países con control directo o indirecto de los recursos energéticos globales y los que no pueden acceder a dichos recursos, se suma otro dato: aproximadamente la cuarta parte de la población mundial consume las 3/4 partes del total de la energía primaria producida [9].

Estos inconvenientes graves sólo se pueden superar de forma efectiva mediante un cambio global en la generación de energía: descentralizándola, utilizando fuentes de energía renovables y mejorando las técnicas de almacenaje del exceso de producción energético.

Las fuentes de energía renovables, si bien resta mucho camino por recorrer en su desarrollo, son tecnologías estudiadas durante décadas y que han demostrado su viabilidad y sus ventajas respecto a las fuentes de energía tradicionales; por otro lado, los sistemas de optimización para la producción y almacenamiento de energía es un mundo completamente nuevo por descubrir y que es esencial para el desarrollo de numerosos campos

de la ingeniería, desde la viabilidad de medios de transportes respetuosos con el medio ambiente hasta la descentralización de generación de energía, pasando por el desarrollo de viviendas y edificios autosuficientes.

Está claro que la energía que se se suministra cerca de las centrales no crea demasiados problemas, pero en ocasiones existen zonas donde debe ser transportada hasta casi 20.000 km de distancia, con unas pérdidas del 3 % cada 1000 km [19], con los costes adicionales que ello supone, tanto en pérdidas como en infraestructura. Además de otros aspectos como la diferencia de oferta y demanda, así como la variación de precio de la energía, principalmente de noche, podría ser salvada mediante el almacenamiento de energía. En este contexto, diversos investigadores han encontrado una posible solución parcial en el uso de hidrógeno como medio de almacenamiento y transporte energético. La tecnología del hidrógeno tiene una gran eficiencia energética, ya que la energía química del hidrógeno puede ser convertida de forma directa en electricidad sin necesidad de emplear un ciclo termodinámico intermedio. Esta transformación directa se puede llevar a cabo en pilas de combustibles. Sin embargo, el hidrógeno no es un recurso energético que se encuentre disponible en la naturaleza como tal, sino que ha de ser producido, siendo necesario para ello partir de otras fuentes de energía. Es en estas fuentes de energía donde hay que englobar a las tecnologías energéticas renovables (energía solar, eólica, etc.) [19].

El hidrógeno se puede producir a partir de agua mediante electrolizadores que empleen energía eólica o solar fotovoltaica, obteniendo una forma de almacenamiento tan flexible y útil como los derivados del petróleo, sin los inconvenientes medioambientales, geopolíticos y de escasez de recursos asociados a éstos. El uso a gran escala del hidrógeno podría suponer el desplazamiento estratégico del control del suministro energético del sector transporte de países inestables, al dominio local, mediante la producción propia del hidrógeno. De esta forma se podrían resolver algunos conflictos internacionales y proporcionar soberanía energética a los países y regiones.

Hay numerosas aplicaciones de pilas de combustible basadas en el hidrógeno que se están desarrollando. En este documento se tratará una planta experimental localizada en las instalaciones de la Escuela Técnica Superior de Ingeniería de Sevilla. Se describirá su funcionamiento en el Capítulo 5.

1.2 Objetivos

El objetivo del presente Trabajo de Fin de Grado es la introducción al modelado y simulación de sistemas de generación distribuida, así como el modelado y control de una planta experimental de energía renovable basada en el pilas de combustible de hidrógeno con la que se ha trabajado para implementar algoritmos de control. Para ello se utilizarán herramientas de simulación y paquetes software que permitan el post-procesamiento de datos.

1.3 Estructura del documento

Segundo capítulo

En el segundo capítulo se introduce la herramienta que se utilizará a lo largo de este documento. Se trata de OpenDSS, un software de simulación de sistemas de distribución de energía eléctrica. Se presentan sus capacidades y características como una herramienta de análisis y planeación de sistemas de distribución. Se desarrolla un ejemplo a través del cual es explicado el funcionamiento de OpenDSS como aplicación independiente.

Tercer capítulo

En el tercer capítulo se implementa en OpenDSS un sistema de distribución de energía eléctrica de la IEEE con 37 nodos, añadiendo las diferentes modificaciones que se plantean en el documento [17]. Entre ellas está el modelado de fuentes de energía renovable tales como plantas solares fotovoltaicas y generadores eólicos, así como elementos de almacenamiento.

Cuarto capítulo

El cuarto capítulo tiene como objetivo explicar cómo utilizar conjuntamente OpenDSS y Matlab sin necesidad de ejecutar la interfaz gráfica de OpenDSS. Además se exponen mediante ejemplos algunas de las características más importantes que será necesario utilizar.

Quinto capítulo

En el quinto capítulo se introduce la planta experimental doméstica basada en hidrógeno renovable. Se hará una breve descripción de los dispositivos que forman parte de ella y se presentará un modelo simplificado de la planta en OpenDSS a través de Matlab. Una vez modelado se implementará una estrategia de control por histéresis mediante estas herramientas.

Sexto capítulo

En el sexto capítulo se detallan las conclusiones que se han alcanzado tras utilizar el software OpenDSS junto con el paquete Matlab para trabajar con diferentes tipos de sistemas eléctricos, además de los posibles objetivos futuros para este proyecto.

2 Software para modelado y simulación

En el presente capítulo se hace una descripción del software que ha sido utilizado para el desarrollo del trabajo, describiendo las características y funciones básicas que se han utilizado de dicho software.

2.1 Introducción.

Los programas que se van a utilizar son *OpenDSS* y *Matlab*. Esta introducción se centrará especialmente en OpenDSS por ser un software más desconocido y menos extendido en los ámbitos ingenieriles. En cambio Matlab es ampliamente utilizado, por lo que se dará por sentado que el lector sabe utilizarlo y el presente texto no será tan exhaustivo en la explicación del código desarrollado.

2.2 Qué es OpenDSS.

OpenDSS [5] es un software de simulación de sistemas de distribución de energía eléctrica desarrollado por EPRI (Electric Power Research Institute). Se trata de un programa del tipo software libre utilizado en la simulación de propiedades eléctricas en el dominio de la frecuencia con las funcionalidades propias de los simuladores comerciales. Es un software aún en desarrollo que va implementando progresivamente nuevas herramientas que tienen en cuenta futuras necesidades relacionadas con los esfuerzos de modernización de las redes actuales. El programa fue originalmente pensado como una herramienta para el análisis de la interconexión de generación distribuida, pero su continua evolución ha llevado al desarrollo de otras funcionalidades que son perfectas, por ejemplo, para estudios de eficiencia en el suministro de energía y estudios de armónicos [3].

Una de las características innovadoras respecto a otras herramientas típicas de análisis de sistemas de distribución de energía eléctrica, es su modo de solución cuasiestática, es decir, las simulaciones en tiempo secuencial. Actualmente ya existen varias herramientas con capacidades similares, pero OpenDSS desde el principio fue diseñado para llevar a cabo simulaciones de ciclos de trabajo con la periodicidad deseada: anuales, diarios, etc. Cada elemento del sistema eléctrico bajo estudio puede tener una única curva de carga o funcionamiento; esta curva es una característica importante porque los contadores de energía modernos pueden proporcionar datos de un intervalo de demanda para cada cliente. La facilidad con la que OpenDSS realiza esta tarea, con sus opciones *Monitor* y *EnergyMeter*, quizás lo haga único en la industria, ya que puede capturar los resultados de series temporales en largas simulaciones. Estas opciones pueden ser muy útiles para el análisis con recursos en la utilización de energías renovables, almacenamiento, vehículos eléctricos, etc., cuya solución es difícil de obtener sin modelar el comportamiento del sistema como función del tiempo.

Otra de las ventajas de utilizar OpenDSS, es que el código fuente es libre, para que los investigadores de redes inteligentes que necesiten capacidades avanzadas de simulación que aún no hayan sido implementadas, puedan modificar dicho código para desarrollarlas.

2.2.1 Principales utilidades de OpenDSS

Algunas de las utilidades para las que ya ha sido utilizado OpenDSS:

- Modelado y análisis de redes de distribución.
- Análisis de circuitos AC polifásicos.
- Análisis de interconexión de generación distribuida.
- Simulación anuales de generación y carga.
- Simulación de plantas eólicas.
- Mejoramiento de la eficiencia en redes de distribución.
- Estudios de armónicos.
- Otros muchos.

2.2.2 Modos de solución

Una vez modelado el tipo de sistema eléctrico que se desea estudiar, OpenDSS permite varios modos de solución para diferentes tipos de análisis. Algunos de ellos serán utilizados posteriormente y se explicarán en más detalle.

- **Snapshot**. Modo estático para flujo de potencia.
- **Directo**. Modo no iterativo.
- **Daily Mode**. Simulación de 24 horas con incrementos de 1 hora por defecto.
- **Yearly Mode**. Simulación de 8760 horas (un año) en incrementos de una hora por defecto.
- **Duty cycle**. Simulación de ciclos de trabajo con incrementos de 1 a 5 segundos. Útil para el análisis de la generación de energía de las renovables por ejemplo.
- **Dynamics**. Simulación de transitorios electromecánicos.
- **Fault Study**. Informes de corrientes y voltajes en todas las fases para todos los tipos de fallos. Útil para la depuración de modelos de circuitos.
- **MonteCarlo fault study**. El usuario define los fallos y los lugares del modelo donde estos deben ocurrir. El programa los selecciona uno a uno y los analiza.
- Análisis de armónicos.
- Modo de solución definida por el usuario.

2.2.3 Elementos básicos en OpenDSS

Como se ha dicho anteriormente, el programa OpenDSS se utiliza para modelar circuitos y sistemas eléctricos; para ello utiliza una serie de elementos básicos que pueden ser configurados con una gran flexibilidad. Algunos de los elementos más utilizados son:

- **Elementos de suministro de potencia.**
  - Lineas. Todo tipo de líneas y cables para distribución.
  - Transformadores. Multifase y multibobinados.
  - Condensadores. En serie y en derivación.
- **Elementos de conversión de potencia.**
  - Generadores.
  - Cargas disipadoras de energía.
  - PVSysyem. Sistemas de energía fotovoltaica incluyendo panel e inversor.

  - Almacenaje.
- **Medidores.**
  - EnergyMeter. Permite medir potencias y pérdidas.
  - Monitor. Permite medir cantidades en un determinado punto del circuito.
  - Sensores.

2.3 OpenDSS como aplicación independiente.

El software OpenDSS está diseñado para funcionar de forma autónoma mediante el archivo ejecutable *OpenDSS.exe*, esto es, con su propia interfaz de usuario del programa (ver Figura 2.1). A través de dicha interfaz se pueden generar los códigos de diseño de circuitos, simulación y otras funciones ya mencionadas. Además ofrece multitud de herramientas de control y análisis de cada elemento del circuito.

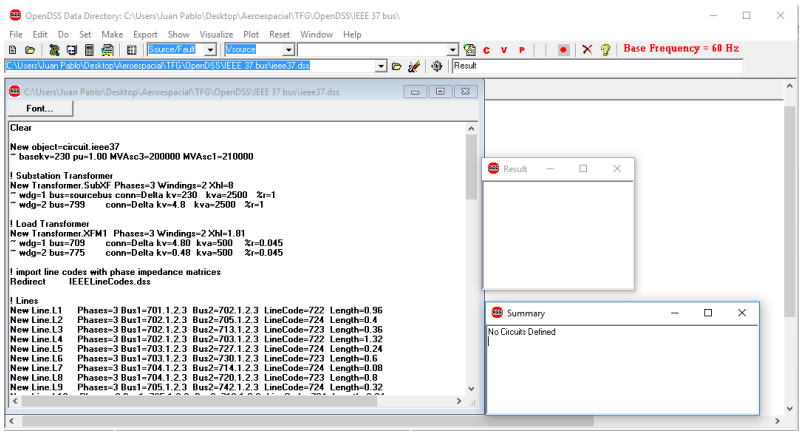


Figura 2.1 Interfaz gráfica de OpenDSS.

OpenDSS está diseñado para recibir instrucciones en forma de texto permitiendo mayor flexibilidad al usuario. Puede accederse al programa tanto a través de una aplicación independiente como a través del módulo COM server. La aplicación independiente cuenta con una interfaz de usuario muy básica, aunque funcional, que permite interactuar con el programa; el COM server permite conectar OpenDSS con otros programas tales como Matlab, Visual Basic, Python o Excel, proporcionando de esa manera una gran capacidad de análisis de la información.

2.3.1 Script de un circuito en OpenDSS

El programa OpenDSS incluye varios ejemplos de diferentes tipos de circuitos. En este apartado se va a utilizar el *ejemplo 1* (Figura 2.2) de la guía de referencia del programa (*Reference Guide: The Open Distribution System Simulator – OpenDSS* [4]), explicando su desarrollo y simulación posterior.

En la Figura 2.2 se comprueba que el circuito de ejemplo consiste en una alimentador que parte de un transformador trifásico y que alimenta a distintas cargas puntuales que se encuentran repartidas a diferentes distancias (expresadas en ft). En el circuito también se dispone de un generador eólico de 8 MW. En el apéndice (Ver A.1) se incluye el código DSS que modela este circuito y posteriormente se comentará sentencia por sentencia para entender su contenido. Este código puede ser escrito directamente en una de las ventanas de la interfaz gráfica del software o en un documento plano de texto guardado con la extensión *.dss*.

A continuación se pretende explicar dicho código para entender cómo se modela un circuito sencillo como éste y cuáles son algunas de las opciones más importantes que permite el software OpenDSS. Además se ha adaptado otra sintaxis más intuitiva, añadiendo el nombre de los parámetros definidos para facilitar su entendimiento; ambas son igualmente válidas para el software. Por ejemplo, si la primera instrucción en el código original es:

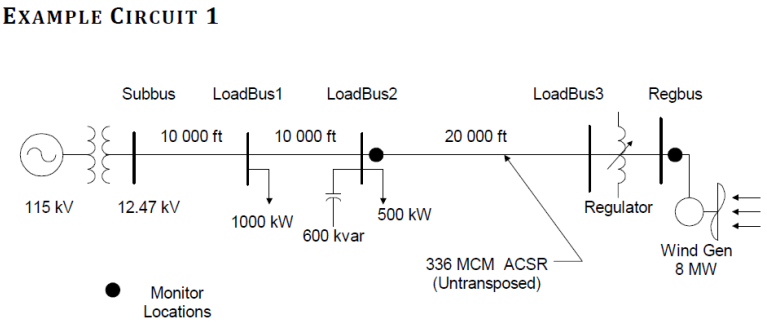


Figura 2.2 Circuito de ejemplo.

```
New object= circuit .DSSLLibtestckt
~ basekv=115 1.00 0.0 60.0 3 20000 21000 4.0 3.0
```

Esta línea de comandos es completamente análoga a la siguiente, donde se indican los nombres de los parámetros utilizando los comandos asignados por OpenDSS para dichas variables:

```
New object= circuit .DSSLLibtestckt
~ basekv=115 pu=1.00 Angle=0.0 Frequency=60.0 Phases=3 Mvasc3=20000 Mvasc1=21000
~ x1r1=4.0 x0r0=3.0
```

Estos nombres se pueden consultar con el comando Help o en el Manual de usuario de OpenDSS. La ventaja de utilizar los nombres en la definición de los parámetros es que no hay que seguir un orden específico en su configuración; en el caso de no indicar los nombres (como en el ejemplo anterior), habrá que seguir escrupulosamente el orden que se indica en el Manual de usuario para que OpenDSS lo pueda interpretar correctamente.

Pasos previos

- **Primero, usar la frecuencia correcta.** Antes de empezar es aconsejable configurar la frecuencia base por defecto a la que se vaya a utilizar en el circuito normalmente. En caso contrario, si el circuito tiene especificada una frecuencia diferente a la frecuencia base, no funcionará.

```
Set DefaultBaseFrecuency = 60
```

- **No olvidar el *"sourcebus"*.** Es importante configurar el *sourcebus* (o bus de referencia) cuando sea apropiado utilizarlo. Según las necesidades del modelo se puede hacer a través de una línea o un transformador.
- **OpenDSS imita redes reales.** Por lo tanto se necesita colocar monitores y medidores.

Configuración básica de elementos y dispositivos

- **Comando *"Clear"*.**

  Este comando debe encabezar todos los scripst de OpenDSS para evitar fallos y errores al compilar. Su función es eliminar de memoria todas las configuraciones, simulaciones y archivos temporales que pudiesen existir.

- Configurar la ruta de trabajo. Esto es especialmente útil cuando se trabaja con varios scripts que no necesariamente se encuentran en el mismo fichero. Su sintaxis es:

```
set datapatch = C:\OpenDSS\CircuitosEjemplo\
```

- **Crear el nuevo circuito.**

Se crea el circuito como un nuevo objeto de OpenDSS con el comando *New*. Posteriormente se configura el circuito y la fuente de tensión principal. En este caso se trata de una fuente alterna de 3 fases con 115 kV de tensión trabajando a una frecuencia de 60 Hz. Con un ángulo de fase nulo.

```
New object= circuit .DSSLLibtestckt
~ basekv=115 pu=1.00 Angle=0.0 Frecuency=60.0 Phases=3 Mvasc3=20000 Mvasc1=21000
~ x1r1=4.0  x0r0=3.0
```

El resto de parámetros no esenciales se utilizan para un modelado más exacto de la fuente de tensión. Por ejemplo, *Mvasc3* y *Mvasc1* hacen referencia a tensiones en cortocircuito para 3 y 1 fase respectivamente; mientras que *x1r1* y *x0r0* se utilizan para configurar relaciones de impedancias internas.

- **Curvas LoadShape.**

  El objeto *LoadShape* se utiliza para simulaciones temporales (diarias, anuales, etc). Consiste en una serie de multiplicadores que típicamente varían de 0.0 a 1.0; estos multiplicadores se aplican a los valores de la carga en kW para representar la variación de la carga respecto al tiempo. Se aplica generalmente a un intervalo de tiempo fijo, pero también puede ser variable. En cualquier caso se debe especificar tanto el tiempo (en horas) como los multiplicadores. Un objeto *loadshape* es, al fin y al cabo, un vector cuyas componentes son los multiplicadores. El *loadshape* puede ser introducido directamente en la línea de comandos o pueden ser almacenado en otro tipo de archivo que se almacenan en memoria (generalmente, tablas Excel).

  En el circuito de ejemplo se incluyen dos vectores *LoadShape*.

```
New loadshape.day 8  3.0
~ mult=(.3  .36  .48  .62  .87  .95  .94  .60)
```

  Como con cualquier objeto en OpenDSS, lo primero es crearlo con el comando *New*, seguido del tipo de objeto(*LoadShape*) y el nombre (*day*). El valor 8 representa el número de puntos para definir la curva. El 3.0 el tiempo de intervalo en horas. En este caso se ha realizado una modificación con respecto al código original que tomaba un vector de 24 componentes (1 por hora) para que se observen diferentes formas de proporcionar este objeto a OpenDSS. Por último se añaden los multiplicadores.

  En el segundo *LoadShape* se definen las cargas para el generador eólico, en el que ya se han tenido en cuenta las características de la turbina y una estimación de la velocidad del viento para un día medio. Este vector se proporciona a través de un fichero externo *zavwind.csv* que contiene 2400 puntos, por lo que se realiza una extrapolación para obtener los 24 necesarios (1 por hora) para la simulación diaria.

```
New loadshape.wind 2400 {1  24  /}
~ mult=( file =zavwind.csv)    action=normalize
```

  El comando *normalize* escala los multiplicadores para que el valor pico sea igual a 1.

- **Líneas**

  El elemento *Línea* de OpenDSS se utiliza para modelar la mayoría de cables o líneas multifase. En el software se trata como un elemento de potencia que se caracteriza por su impedancia. Las impedancias de una línea pueden ser especificadas mediante valores de componentes simétricos, una matriz o haciendo referencia a un objeto *linecode* existente. También es posible especificar la geometría y que OpenDSS calcule las impedancias de la línea.

  En el caso de que se especifique la propiedad *Geometría*, las matrices de impedancias o el objeto *linecode* serán ignorados. Con la geometría introducia, el software calcula las matrices de impedancia cada vez que cambia de frecuencia. Las longitudes de línea pueden ser introducidas en cualquier unidad, pero el cambio de unidad hay que declararlo o se introducirán en la unidad por defecto (m).

  El objeto de línea por defecto es una línea aérea de 1000 pies de longitud con un conductor del tipo 336 MCM ACSR en una cruceta de 8 pies. Se define por valores de los componentes simétricos:

```
R1 = 0.0580  ohms per 1000 ft
X1 = 0.1206  ohms per 1000 ft
R0 = 0.1784  ohms per 1000 ft
X0 = 0.4047  ohms per 1000 ft
C1 = 3.4e-9 nF per 1000ft
C0 = 1.6e-9 nF per 1000ft
```

Figura 2.3 Valores de los componentes simétricos de un conductor tipo 336 MCM ACSR.

Los objetos ***linecode*** son librerías que contienen características de impedancia para las líneas y cables. En la mayoría de los programas de análisis de distribución, se puede describir una línea por su *linecode* y su longitud.

En última instancia, la impedancia de una línea se describe con su matriz de impedancia y su matriz nodal capacitiva. Estas matrices se pueden especificar directamente o pueden ser generadas mediante los datos de los componentes simétricos.

$$Z = R + jX = \begin{bmatrix} Z_{11}+Zg & Z_{12}+Zg & Z_{13}+Zg \\ Z_{21}+Zg & Z_{22}+Zg & Z_{23}+Zg \\ Z_{31}+Zg & Z_{32}+Zg & Z_{33}+Zg \end{bmatrix}$$

Figura 2.4 Matrices de impedancias.

Hay que señalar que no es necesario utilizar *linecode*, ya que las impedancias se pueden introducir manualmente, pero sí es aconsejable hacerlo cuando hay un número considerable de líneas con las mismas características o ya se tienen almacenadas en la librería, que suele ser lo habitual.

En el ejemplo se crea y configura un nuevo *linecode*. Como se puede ver, se sigue el mismo procedimiento que para crear otro tipo de objetos: se crea la línea con el comando *New*, se le asigna un nombre (*336matrix*) y el número de fases. Se va a caracterizar la línea mediante matrices triangulares inferiores, tal que: $Z = R + jX$, donde R es la resistencia y X es la reactancia medida en $\Omega/(unidad\,de\,longitud)$. Posteriormente se crea la matriz de capacitancia en $nanofaradios/(unidad\,de\,longitud)$ y se configura la corriente máxima (*Normamps*) y la corriente máxima de emergencia (*Emergamps*).

```
New linecode.336matrix  nphases=3
~ rmatrix=(0.0868455 |  0.0298305 0.0887966 |  0.0288883 0.0298305 0.0868455)  % \Omega/1000ft
~ xmatrix=(0.2025449 |  0.0847210 0.1961452 |  0.0719161 0.0847210 0.2025449)  % \Omega/1000ft
~ cmatrix=(2.74  |  −0.70 2.96  |  −0.34 −0.71 2.74)  % nf/1000ft
~ Normamps=400 Emergamps=600
```

Una vez definido y configurado el *linecode*, se crean las líneas:

```
New line. line1  bus1=subbus bus2=loadbus1 linecode=336matrix length=10
New line. line2  bus1=loadbus1 bus2=loadbus2 linecode=336matrix length=10
New line. line3  bus1=loadbus2 bus2=loadbus3 linecode=336matrix length=20
```

El procedimiento una vez creada la línea con el comando *New*, es designar los buses entre los que se instalará, añadir el *linecode* y asignar la longitud.

- **Cargas.**

  Una carga es un elemento de conversión de energía que es fundamental en muchos análisis. Básicamente se define por sus kW nominales y su factor de potencia (PF) o sus kW y sus $kvar$. Una vez definida puede ser modificada por una serie de multiplicadores (*loadshapes*).

  La configuración predeterminada hace que la carga se comporte como una fuente de inyección de corriente. Por lo tanto, su matriz primitva Y contiene sólo la impedancia que podría existir desde el neutro de una carga conectada en estrella a tierra.

Sin embargo, se puede cambiar a un modo de admitancia con el comando *Set LoadModel*; haciendo que la carga se convierta en una admitancia que es incluida en la matriz Y del sistema. Éste sería el modelo utilizado para los estudios de fallos en los que no se alcanza la convergencia a causa de las bajas tensiones. Las cargas se suponen equilibradas para el número de fases especificadas. Si se desean cargas desequilibradas, será necesario introducir las cargas monofásicas por separado.

Hay tres formas de caracterizar las cargas:

1. kW, PF

2. kW, kvar

3. kVA, PF

Para asegurarse de que el funcionamiento sea el deseado, hay que introducir esas propiedades en el orden que se indica.

Para definir la carga del ejemplo se recurre al siguiente código:

```
New load.load1 bus1=loadbus1 phases=3 kv=12.47 kw=1000 pf=0.88 model=1 class=1 duty=day

New load.load2 bus1=loadbus2 phases=3 kv=12.47 kw=500 pf=0.88 model=1 class=1 duty=day ~ conn=delta %
      Idem a la carga anterior. Conn indica la configuración delta o estrella
```

Hay 8 modelos diferentes; se ha seleccionado el 1 que es aquel donde P y Q son constantes. *Class* es el número entero de segregación de la carga y *duty* es el nombre del ciclo de trabajo, que se ha hecho coincidir con el *loadshape* previamente creado.

- **Transformador.**

  Un transformador se implementa como un elemento de suministro de energía con dos o más terminales. Consta de dos o más bobinados que pueden estar conectados de diferentes maneras (por defecto con una conexión estrella-triángulo). Los transformadores tienen una o más fases y el número de conductores por terminal es siempre el número de fases más uno. Para bobinados conectados en estrella (*wye*) el conductor adicional es el punto neutro; para bobinados conectados en triángulo, el terminal adicional está abierto internamente.
  Para incorporar una subestación transformadora se utiliza un código similar a éste:

```
New transformer.subxfrm phases=3 windings=2
~ buses=(SourceBus subbus)
~ conns='delta wye' kvs=(115 12.47)
~ kvas=(20000 20000) XHL=7

 New transformer.reg1 phases=3 windings=2
~ buses=(loadbus3 regbus) conns='wye wye'
~ kvs=(12.47 12.47) kvas=(8000 8000) XHL=1
```

  - La línea *New transformer.subxfrm* crea un nuevo objeto de tipo transformador con nombre *subxfrm*.
  - El parámetro *windings* hace referencia al número de bobinados y el parámetro *phases* al número de fases.
  - La línea *buses=(SourceBus subbus)* indica entre qué dos buses está instalada la subestación, siendo equivalente a definirlos como *bus1=SourceBus bus2=subbus*.
  - El parámetro *conns* sirve para configurar la conexión en delta o en estrella.
  - El comando *kvs=(115 12.47)* se utiliza para indicar el ratio de transformación. Mientras que *kvas=(20000 20000)* indica los KVA de referencia.
  - El parámetro *XHL=7* representa el porcentaje de reactancia de alto a bajo (del bobinado 1 al 2).

Otra opción que permite OpenDSS es incorporar un control que hace la función de regulador de tensión (*RegControl*) para un transformador. Este control se conecta a un determinado bobinado del transformador. Siendo también posible controlar el votaje de un bus remoto para emular varios dispositivos de

una red inteligente.

En el ejemplo, los reguladores de tensión se configuran así:

```
New regcontrol.subxfrmCtrl  transformer=subxfrm windings=2 vreg=125
~ band=3  ptratio=60 delay=10

New regcontrol.reg1Ctrl  transformer=reg1  windings=2 vreg=122
~ band=3  ptratio=60 delay=15
```

En la configuración se indica el número de bobinados con la sentencia *windings*. El parámetro *Vreg* es la configuración del regulador de tensión; multiplicando este valor por el *ptratio* se debe obtener el voltaje a través del bobinado bajo control. El parámetro *Band* hace referencia al ancho de banda en voltios para el bus controlado. *Ptratio* es la relación entre la tensión del bobinado y del regulador. El parámetro *Delay* es el retraso en segundos, que se usa para determinar qué controlador actúa primero.

- **Capacitadores.**

  El modelo de condensador se implementa, básicamente, como un elemento de suministro de potencia de dos terminales. Sin embargo, si no se especifica una conexión para el segundo bus, se toma por defecto el *nodo*0 (referencia de tierra) del mismo bus al que está conectado el primer terminal. Es decir, el valor predeterminado es una batería de condensadores de derivación a tierra en estrella (*wye*).

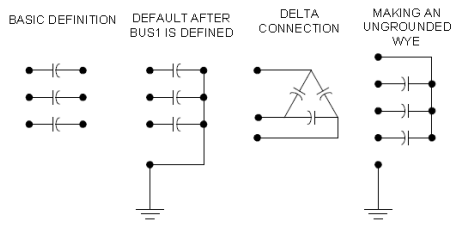


Figura 2.5 Definición del objeto capacitador.

Si se especifica que la conexión es de tipo *"delta"*, entonces el segundo terminal se elimina. Para configurar un condensador en serie, sólo se tiene que especificar una conexión para el segundo bus. Por último, para un condensador en estrella sin conexión a tierra, se deben ajustar todos los segundos terminales a un nodo vacío en el terminal del primer bus, por ejemplo:

```
Bus1=B1 bus2 = B1.4.4.4 % Para un capacitador de 3 fases
```

Cualquier otra conexión es posible especificando correctamente la conexión de los terminales.

Aunque el objeto *capacitor* puede ser tratado como si fuera una sola batería de condensadores, en realidad se implementa como un banco de filtros de varios pasos sintonizado.

En el caso de que se quiera representar un banco de filtros, hay que seleccionar la propiedad *XL* o la *Harm*. Cuando un objeto *Capcontrol* conmuta un condensador, lo hace para aumentar o disminuir el paso activo.

En el ejemplo se incorpora un capacitador con un sistema de control *CapControl*, el código utilizado es el siguiente:

```
New capacitor.Cap1 bus1=loadbus2 phases=3 kvar=600 kv=12.47
New capcontrol.Cap1Ctrl element=line.line3  terminal=1  capacitor=Cap1
~     type=current   ctratio=1 ONsetting=60 OFFsetting=55 delay=2
```

Como no se ha especificado una conexión para el segundo bus, se toma por defecto la referencia a tierra.

Con *Capcontrol* se configura el control del condensador. Básicamente se puede monitorear la tensión y la corriente de un elemento y enviar mensajes de conmutación a un condensador.

En la configuración se indica el elemento que va a ser monitoreado (*line3*), el terminal (el 1), el condensador bajo control (*Cap1*) y el tipo de control (que puede ser de corriente, tensión o tiempo, en el ejemplo se utiliza un tipo de control en corriente). El parámetro *ONsetting* da el valor para cambiar el condensador a *ON*, *OFFsetting* hace lo mismo con el estado *OFF*, las unidades de ambos parámetros dependen del tipo de control (en este caso amperios). *Delay* es el retraso en el tiempo (en segundos) desde el momento en que se arma el control hasta que se envía la orden de conmutación.

- **Generador.**

  El objeto *Generator* en OpenDSS es un elemento de conversión de energía similar a una carga. Se define básicamente por su kW nominales y PF o su kW y $kvar$. Pudiendo ser modificados por una serie de multiplicadores o *LoadShape*.

  Para los estudios de flujo de potencia, el generador es esencialmente una carga negativa, es decir, una carga que genera potencia. Para el modo *Armónicos*, el generador se convierte en una fuente de tensión. Para el modo *Dinámico*, el generador se comporta como una fuente de tensión detrás de una impedancia, dependiendo esta impedancia del modelo elegido.

  Si el valor de *"envío"* (propiedad *DispValue*) es 0, el generador siempre sigue la curva de funcionamiento asociada, que es simplemente un objeto *Loadshape*. Si este valor es mayor que 0, $DispValue > 0$, entonces el generador se enciende sólo cuando el multiplicador de la carga supera el valor del parámetro *DispValue*. Cuando el generador está encendido, siempre sigue la curva de funcionamiento para el tipo de solución que se está realizando.

  Si se desea modelar un generador que está funcionando siempre al mismo nivel, hay que añadir el comando *"Status = Fixed"*. El valor por defecto del estado (*Status*) es "Status = Variable", es decir, que se sigue una curva *Loadshape* por defecto. Otra opción para que el generador se comporte siempre igual es definir una curva *LoadShape* cuyos multiplicadores sean todos igual a 1.0.

  Los generadores tienen sus propios medidores de energía que graban los siguientes parámetros:

  1. Total *kwh*.

  2. Total *kvarh*.

  3. Máximo kW.

  4. Máximo kVA.

  5. Horas de operación.

  6. $ (Precio * energía generada).

  Los diferentes modelos de generador para las simulaciones de flujo de potencia son:

  1. P constante, Q (dependiente del *LoadShape*).

  2. Z Constante (Para una aproximación de una solución simple).

  3. P constante, $|V|$ como un flujo de corriente estándar con magnitudes de tensión y ángulos como variables.

  4. P constante, Q fijada. P sigue el *LoadShape* mientras que Q es siempre la misma.

  5. P constante, la reactancia fijada. P sigue el *LoadShape* y Q se calcula como si se tratara de una reactancia fija.

  6. Modelo escrito por el usuario.

  7. Corriente limitada y modelo de P y Q constantes. Comportamiento como el de algunos inversores.

  En el ejemplo se configura un generador eólico de 8 MW como un objeto generador:

```
New generator.gen1 bus1=regbus kV=12.47 kW=8000 pf=1 conn=delta duty=wind Model=1
```

Como se puede ver, se le asocia el ciclo de trabajo al *LoadShape* llamado *wind* que se generó al principio.

Monitores, medidores y modos de solución

- **Monitor:**

  Un monitor es un objeto en OpenDSS que se conecta al terminal de otro elemento del circuito. Toma datos de tensión, intensidad o potencia con respecto al tiempo en todas las fases. En esencia funciona como un monitor de potencia real. Se puede configurar de varias formas para obtener diferentes parámetros.

  Para su configuración, se selecciona el elemento y terminal bajo estudio. El parámetro *mode* es un código para describir qué es lo que el monitor guarda. Puede guardar dos tipos de cantidades: A) Tensión y corriente; B) Alimentación.

  Algunos de los modos son:

  - 0. Modo estándar: V, I en cada fase;
  - 1. Potencia en cada fase (kW y kVARs);
  - 2. Tomas del transformador. Conexión a un bobinado;
  - 3. Variables de estado conectado a un PCElement (Power Conversion Element);
  - Etc.

  En el ejemplo siguiente se incluyen 3 monitores: el primero para analizar las potencias generadas en el generador eólico; el segundo para monitorizar tensión y corrientes en la segunda carga; y el tercero para monitorizar la secuencia de tensiones y corrientes en la línea 3.

```
New Monitor.gen1 element=generator.gen1 terminal=1 mode=1
New Monitor.loadbus2 element=load.load2 terminal=1 mode=0
New Monitor.line3 element=line.line3 terminal=1 mode=48
```

- **Energy Meter:** Un *energymeter* es un medidor inteligente conectado a un terminal de un elemento del circuito. Simula el comportamiento de un contador de energía real, sin embargo, tiene más capacidad, ya que puede acceder a los valores de otros puntos del circuito, no sólo donde está instalado. Puede medir potencias y energías, pero también pérdidas y valores de sobrecarga dentro de una región definida del circuito.

  Tiene 2 tipos de registro: A) Valores de energía. B) Potencia máxima.

  En el ejemplo se incorpora un *Energy meter* para obtener datos de una línea y hacer posteriormente las gráficas:

```
New Energymeter.em1 element=line.line1
```

- **Modo de control. *ControlMode*:**

  Se utiliza con el modo *dutycycle* o *dynamic*. *Controlmode* puede tomar valores [OFF | STATIC | EVENT | TIME].

  El valor por defecto es *STATIC*. En el modo *STATIC* el tiempo no avanza. Las acciones de control se ejecutan en orden desde la de menor tiempo de actuación hasta que todas las acciones se realizan. Se utiliza este modo para soluciones de flujo de potencia que requieran varios cambios del regulador por solución.

  Modo *EVENT*. Es un tipo de solución orientada a eventos. Sólo se ejecutan las acciones de control cercanas al evento y el tiempo se hace avanzar automáticamente hasta la hora de éste.

  Modo *TIME*. Es un tipo de solución orientada a control del tiempo. Las acciones de control se ejecutan cuando el tiempo de espera de la acción se alcanza o se supera. Se utiliza el modo *TIME* para modelar

un control y un modo de solución externo al software OpenDSS, tales como *Daily* o *Dutycycle* que avanzan en el tiempo, o cuando se ajusta el tiempo (horas y segundos) explícitamente desde el programa externo (por ejemplo Matlab).

En el ejemplo, como paso previo a seleccionar el modo de control, se ejecutan dos comandos:

```
Set voltagebases =(115 12.47 .48)

Calcvoltagebases
```

El primero sirve para definir las bases de tensiones del circuito y que los informes sean expresados respecto a la unidad. El segundo, *Calcvoltagebases*, genera la lista de buses y calcula las tensiones base de éstos.

Posteriormente, se selecciona los modos de control de la simulación para hacer que las funciones de captura y registro funcionen de manera sincronizada con el resto de la simulación:

```
Set controlmode=time

Set controlmode=duty number=86400 hour=0 stepsize=1 sec=0
```

La segunda configuración es para una simulación basada en tiempo. Durante 24 horas (86400 segundos), con intervalos de 1 segundo, empezando en la hora 0 y segundo 0.

- **Ejecución de la simulación.**

  Para llevar a cabo la simulación se ejecuta el comando que resuelve el circuito:

```
Solve
```

  Este comando ejectua la solución en el modo especificado en el paso anterior, *Set Mode = comando*. Se puede ejecutar una o cientos de soluciones. La "solución" es un objeto de DSS asociado al circuito activo y que se puede consultar.

Interpretación de resultados.

Hay varias formas de mostrar los resultados en pantalla: se puede hacer uso de gráficas, exportar los resultados a documentos .txt o excel, mostrar diferentes parámetros en cada dispositivo, etc.

- **Gráficas. Comando *Plot*.**

  Hay muchas opciones para el comando *Plot*, de forma que tiene su propia rama en el árbol de ayuda de la aplicación OpenDSS. Básicamente es una herramienta de representación gráfica de resultados. Algunas opciones que aquí se utilizan son:

  - **Selección del tipo de gráfica. "type":** Circuit | Monitor | Daisy | AutoAdd | General (bus data) | Loadshape | Tshape | Princeshape | Profile .

    Un *plot* de un circuito requiere que se definan las coordenadas de los buses. Por defecto, el grosor de las líneas del circuito se dibuja proporcional a la potencia. También se pueden realizar gráficas a partir de los monitores, dibujando uno o más canales del elemento que está siendo monitorizado.

  - **Selección de la cantidad mostrada. "quantity":** Voltage | Current | Power | Losses | Capacity . Opción que sirve para indicar qué valor o cantidad será representada en la gráfica plot.
  - **Selección del objeto de OpenDSS utilizado. " object"** = [Nombre de la zona que se graficará | Nombre del monitor | Nombre del Bus o Circuito | Nombre del Loadshape]. Sirve para especificar qué objeto será el usado en la función plot.
  - **Canales que se mostrarán. "Channels":** Sirve para indicar qué canales serán graficados en el plot.

Hay más opciones adicionales que se pueden consultar en HELP o en el manual.

En el ejemplo se utilizan los siguientes comandos:

```
Plot type=monitor, object=loadbus2, Channels=(1,3,5)
Visualize element=line.line1 what=powers

Plot profile
```

El primer *Plot* sirve para mostrar los valores registrados por el monitor instalado en la segunda carga en diferentes canales; en particular, se van a visualizar las potencias.

Por último, el comando *Plot profile* , muestra el perfil de la tensión en el alimentador al final de la simulación (último intervalo).

En el anexo de códigos se podrá ver el script completo del circuito (Ver A.1).

Ejecución del código

Para ejecutar el código desde OpenDSS, éste deberá estar seleccionado, luego se debe hacer clic derecho sobre él y selecciona la opción *Do*; una operación análoga es pulsar *Ctrl + D* cuando el código esté seleccionado (Ver Figura 2.6). Al realizar esta acción se resolverá el sistema realizando el tipo de simulación que se haya programado. Cuando el sistema ha sido resuelto es posible generar un gran número de archivos de salida que contienen información acerca del estado del sistema y de todos los elementos, además de otras salidas de tipo gráfico. Un dato importante, es que para ejecutar la opción *Do* (*Ctrl +D*), se puede seleccionar una o más o líneas de código, siendo solo ejecutado la parte del código seleccionado. Esto es útil para seguir realizando operaciones una vez que el circuito ha sido ya resuelto. Para que el script de ejecute sin problemas, es necesario que los archivos a los que se llame desde el mismo estén en carpeta o en el directorio indicado a OpenDSS. En el ejemplo anterior, habría que tener en la carpeta del script los *LoadShape* utilizados, en este caso el archivo para el generador eólico *zavwind.csv*

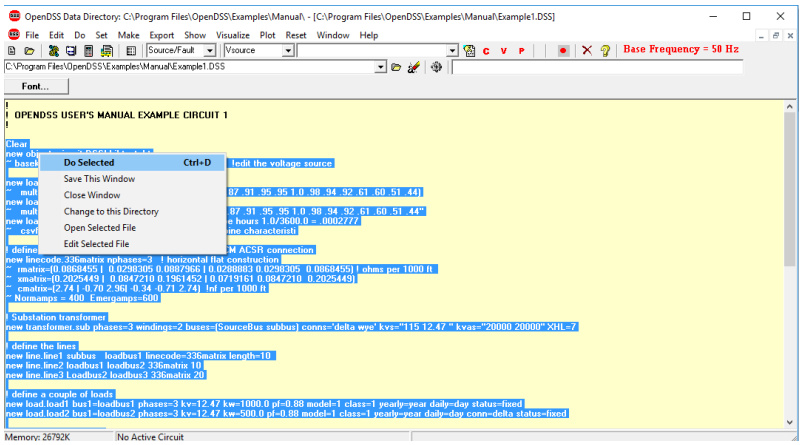


Figura 2.6 Ejecutar el script desde OpenDSS.

OpenDSS genera un archivo de texto en el cual se encuentran las potencias que fluyen a través de todos los elementos del sistema en forma de componentes simétricas; también presenta información acerca de la potencia de sobrecarga de cada elemento y las pérdidas totales del sistema. Para generar este archivo se utiliza el comando:

```
Show Powers
```

Todas las opciones que presenta OpenDSS se pueden ejecutar tanto desde su interfaz gráfica como mediante comandos escritos en el Script del circuito o en la pantalla principal.

3 Sistema de distribución en OpenDSS

En este capítulo se presentará un sistema de distribución a modelar, explicando someramente el script del circuito y cómo modelar e incorporar diferentes elementos como generadores eólicos, plantas solares o baterías en el sistema de distribución bajo estudio. Una vez presentado el modelo, se modificará el script para incorporar estos nuevos elementos al entorno OpenDSS.

3.1 Sistema de distribución IEEE-37

El *IEEE 37 Node Test Feeder* es uno de los cuatro modelos de distribución estándar desarrollados por la IEEE [7]. Este modelo de alimentación de prueba tiene sólo cables subterráneos y da un estándar para probar este tipo de distribución, que es propia de las zonas urbanas densas. Las modificaciones que hay realizar en el script del circuito incluyen el modelado y la implementación de los elementos de la tabla 3.1.

3.1.1 Modelo Inicial

En la figura 3.1 se puede ver un esquema unifilar del sistema de distribución IEEE 37. Se trata de un sistema real de California, con una tensión de servicio de $4.8kV$. Se caracteriza por la configuración delta, todos los segmentos de línea son subterráneas y la regulación de tensión de la subestación se realiza mediante dos reguladores monofásicos en triángulo abierto. Dispone de cargas puntuales y muy desequilibradas.

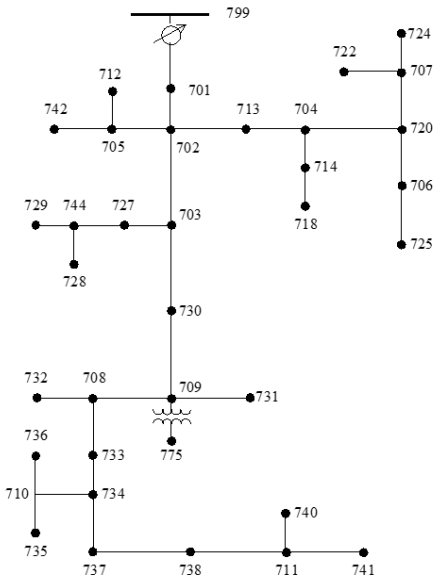


Figura 3.1 Esquema básico del IEEE 37 Node Test Feder.

El script del modelo inicial es el siguiente:

Código 3.1 Script en OpenDSS del sistema de distribución IEEE 37 sin modificar .

```
Clear

New object= circuit . ieee37
~ basekv=230 pu=1.00 MVAsc3=200000 MVAsc1=210000

! Substation Transformer
New Transformer.SubXF Phases=3 Windings=2 Xhl=8
~ wdg=1 bus=sourcebus conn=Delta kv=230  kva=2500  %r=1
~ wdg=2 bus=799       conn=Delta kv=4.8  kva=2500  %r=1

! Load Transformer
New Transformer.XFM1 Phases=3 Windings=2 Xhl=1.81
~ wdg=1 bus=709       conn=Delta kv=4.80  kva=500  %r=0.045
~ wdg=2 bus=775       conn=Delta kv=0.48  kva=500  %r=0.045

! import line codes with phase impedance matrices
Redirect          IEEELineCodes.dss

! Lines
New Line.L1     Phases=3 Bus1=701.1.2.3  Bus2=702.1.2.3  LineCode=722 Length=0.96
New Line.L2     Phases=3 Bus1=702.1.2.3  Bus2=705.1.2.3  LineCode=724 Length=0.4
New Line.L3     Phases=3 Bus1=702.1.2.3  Bus2=713.1.2.3  LineCode=723 Length=0.36
New Line.L4     Phases=3 Bus1=702.1.2.3  Bus2=703.1.2.3  LineCode=722 Length=1.32
New Line.L5     Phases=3 Bus1=703.1.2.3  Bus2=727.1.2.3  LineCode=724 Length=0.24
New Line.L6     Phases=3 Bus1=703.1.2.3  Bus2=730.1.2.3  LineCode=723 Length=0.6
New Line.L7     Phases=3 Bus1=704.1.2.3  Bus2=714.1.2.3  LineCode=724 Length=0.08
New Line.L8     Phases=3 Bus1=704.1.2.3  Bus2=720.1.2.3  LineCode=723 Length=0.8
New Line.L9     Phases=3 Bus1=705.1.2.3  Bus2=742.1.2.3  LineCode=724 Length=0.32
New Line.L10    Phases=3 Bus1=705.1.2.3  Bus2=712.1.2.3  LineCode=724 Length=0.24
New Line.L11    Phases=3 Bus1=706.1.2.3  Bus2=725.1.2.3  LineCode=724 Length=0.28
New Line.L12    Phases=3 Bus1=707.1.2.3  Bus2=724.1.2.3  LineCode=724 Length=0.76
New Line.L13    Phases=3 Bus1=707.1.2.3  Bus2=722.1.2.3  LineCode=724 Length=0.12
New Line.L14    Phases=3 Bus1=708.1.2.3  Bus2=733.1.2.3  LineCode=723 Length=0.32
New Line.L15    Phases=3 Bus1=708.1.2.3  Bus2=732.1.2.3  LineCode=724 Length=0.32
New Line.L16    Phases=3 Bus1=709.1.2.3  Bus2=731.1.2.3  LineCode=723 Length=0.6
New Line.L17    Phases=3 Bus1=709.1.2.3  Bus2=708.1.2.3  LineCode=723 Length=0.32
New Line.L18    Phases=3 Bus1=710.1.2.3  Bus2=735.1.2.3  LineCode=724 Length=0.2
New Line.L19    Phases=3 Bus1=710.1.2.3  Bus2=736.1.2.3  LineCode=724 Length=1.28
New Line.L20    Phases=3 Bus1=711.1.2.3  Bus2=741.1.2.3  LineCode=723 Length=0.4
New Line.L21    Phases=3 Bus1=711.1.2.3  Bus2=740.1.2.3  LineCode=724 Length=0.2
New Line.L22    Phases=3 Bus1=713.1.2.3  Bus2=704.1.2.3  LineCode=723 Length=0.52
New Line.L23    Phases=3 Bus1=714.1.2.3  Bus2=718.1.2.3  LineCode=724 Length=0.52
New Line.L24    Phases=3 Bus1=720.1.2.3  Bus2=707.1.2.3  LineCode=724 Length=0.92
New Line.L25    Phases=3 Bus1=720.1.2.3  Bus2=706.1.2.3  LineCode=723 Length=0.6
New Line.L26    Phases=3 Bus1=727.1.2.3  Bus2=744.1.2.3  LineCode=723 Length=0.28
New Line.L27    Phases=3 Bus1=730.1.2.3  Bus2=709.1.2.3  LineCode=723 Length=0.2
New Line.L28    Phases=3 Bus1=733.1.2.3  Bus2=734.1.2.3  LineCode=723 Length=0.56
New Line.L29    Phases=3 Bus1=734.1.2.3  Bus2=737.1.2.3  LineCode=723 Length=0.64
New Line.L30    Phases=3 Bus1=734.1.2.3  Bus2=710.1.2.3  LineCode=724 Length=0.52
New Line.L31    Phases=3 Bus1=737.1.2.3  Bus2=738.1.2.3  LineCode=723 Length=0.4
New Line.L32    Phases=3 Bus1=738.1.2.3  Bus2=711.1.2.3  LineCode=723 Length=0.4
New Line.L33    Phases=3 Bus1=744.1.2.3  Bus2=728.1.2.3  LineCode=724 Length=0.2
New Line.L34    Phases=3 Bus1=744.1.2.3  Bus2=729.1.2.3  LineCode=724 Length=0.28
New Line.L35    Phases=3 Bus1=799r.1.2.3 Bus2=701.1.2.3  LineCode=721 Length=1.85

! Regulator - open delta with C leading, A lagging, base LDC setting is 1.5 + j3
new transformer.reg1a phases=1 windings=2 buses=(799.1.2 799r.1.2) conns='delta delta' kvs="4.8 4.8" kvas="2000
    2000" XHL=1
new regcontrol.creg1a transformer=reg1a winding=2 vreg=122 band=2 ptratio=40 ctprim=350 R=-0.201 X=3.348
new transformer.reg1c like=reg1a buses=(799.3.2 799r.3.2)
new regcontrol.creg1c like=creg1a transformer=reg1c R=2.799 X=1.848
New Line.Jumper Phases=1 Bus1=799.2     Bus2=799r.2     r0=1e-3 r1=1e-3 x0=0 x1=0 c0=0 c1=0

! spot loads
% En el nodo 701 se implementará el DG
New Load.S701a    Bus1=701.1.2 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S701b    Bus1=701.2.3 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S701c    Bus1=701.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 350.0 kVAR= 175.0
```

```
New Load.S712c      Bus1=712.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S713c      Bus1=713.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S714a      Bus1=714.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 17.0  kVAR=  8.0
New Load.S714b      Bus1=714.2.3 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 21.0  kVAR= 10.0
New Load.S718a      Bus1=718.1.2 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0

% En el nodo 720 se implementará la batería.
New Load.S720c      Bus1=720.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0

% En el nodo 722 se implementará el Generador eólico
New Load.S722b      Bus1=722.2.3 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S722c      Bus1=722.3.1 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 21.0  kVAR= 10.0

New Load.S724b      Bus1=724.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S725b      Bus1=725.2.3 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S727c      Bus1=727.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S728       Bus1=728   Phases=3 Conn=Delta Model=1 kV= 4.800 kW= 126.0 kVAR= 63.0
New Load.S729a      Bus1=729.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 42.0  kVAR= 21.0

% En el nodo 730 se implementará el Generador Solar.
New Load.S730c      Bus1=730.3.1 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S731b      Bus1=731.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S732c      Bus1=732.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S733a      Bus1=733.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S734c      Bus1=734.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0

New Load.S735c      Bus1=735.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S736b      Bus1=736.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 42.0  kVAR= 21.0

% En el nodo 737 se implementará el PEV (coche eléctrico )
New Load.S737a      Bus1=737.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S738a      Bus1=738.1.2 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 126.0 kVAR= 62.0
New Load.S740c      Bus1=740.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S741c      Bus1=741.3.1 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S742a      Bus1=742.1.2 Phases=1 Conn=Delta Model=2 kV= 4.800 kW=  8.0  kVAR=  4.0
New Load.S742b      Bus1=742.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S744a      Bus1=744.1.2 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0

Set VoltageBases = "230,4.8,0.48"
CalcVoltageBases
BusCoords IEEE37_BusXY.csv

! solve mode=direct
set maxiterations =100
solve

! show voltages LL Nodes
! show currents residual =y elements
! show powers kva elements
! show taps
```

Como se observa, tiene una estructura casi idéntica a la que se explicó en la sección 2.3.1. Se comienza haciendo un borrado de memoria con el comando *Clear*, se crea el nuevo circuito *ieee37* y se configura. Inmediatamente después se crean dos unidades transformadoras cuyas características se ven en el script. Se carga un *lineCode* o código de línea que consiste en una librería con diferentes configuraciones de impedancia que posteriormente se utilizan para definir cada una de las líneas del circuito. Tras las 35 líneas definidas, se configuran 2 nuevos transformadores con sus respectivos reguladores de tensión. Por último en el modelado, se configuran todas las *cargas* puntuales conectadas a sus respectivos buses.

Una vez configurados todos los elementos del circuito, se hace calcular las tensiones base y se introducen las coordenadas de todos los buses para poder realizar las gráficas posteriormente. Las coordenadas se introducen mediante una tabla de Excel (*BusCoordsIEEE37_ BusXY.csv*) que se encuentra entre los ficheros de OpenDSS.

Tras esto ya se puede ejecutar el comando *solve* en el modo que se requiera para obtener la solución del circuito.

3.1.2 Modificación para simulación

Las modificaciones que se han realizado en el sistema de distribución IEEE 37 están basadas en el esquema unifilar del artículo [17] (Ver figura 3.2). Para ello se va a proceder al modelado y a la implementación de los elementos de la tabla 3.1 en el script del circuito IEEE 37 (Ver Código 3.1).

Tabla 3.1 Dispositivos implementados y nodos.

Generador diesel	Nodo 701
Generador eólico	Nodo 722
Generador solar	Nodo 730
Batería	Nodo 720
PEV (Coche eléctrico)	Nodo 737

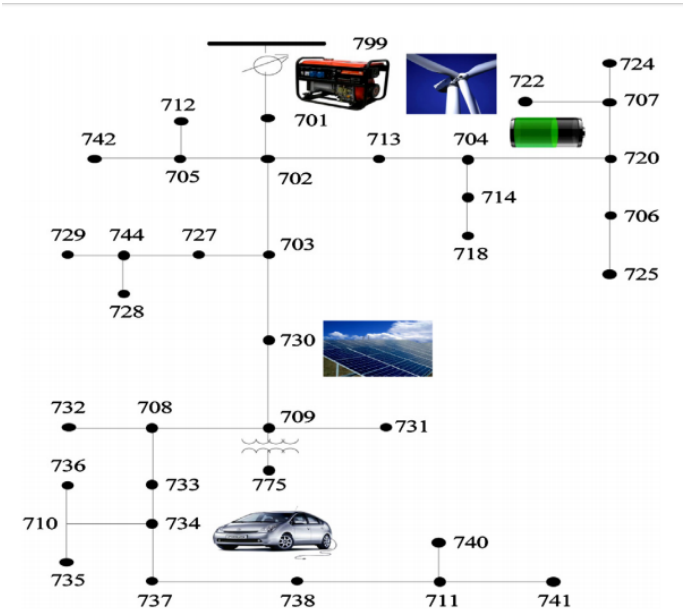


Figura 3.2 Esquema del IEEE 37 Node Test Feder.

La modificaciones realizadas no afectan a la estructura del código, sólo se sustituyen las cargas puntuales de los nodos 701, 722, 730, 720 y 737 por los dispositivos necesarios para convertir este modelo de red en un sistema de distribución con energías renovables (planta solar y generador éolico), baterías, un generador diésel y un coche eléctrico que deberá cargar sus baterías. Para ello, lo primero es modelar este tipo de elementos en OpenDSS.

3.2 Modelado e implementación de fuentes generadoras de energía renovable en OpenDSS

3.2.1 Modelado Planta Solar Fotovoltaica

La aplicación OpenDSS permite modelar plantas solares fotovoltaicas de forma simplificada (ver referencia [6]) y realizar numerosos tipos de análisis en un sistema de distribución con este tipo de dispositivos. Un esquema del modelado en OpenDSS se puede ver en la figura 3.3.

Por lo tanto, lo primero que hay que hacer es definir o generar varias curvas:

- Perfiles de carga
- Irradancia solar

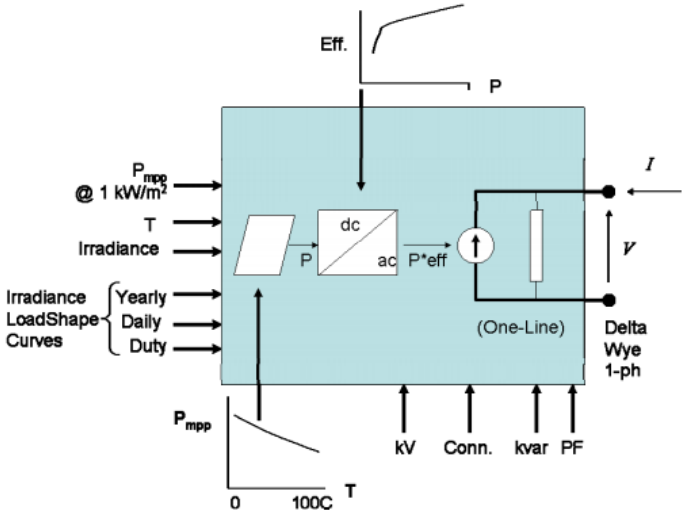


Figura 3.3 Diagrama de bloques de un sistema con placas solares en OpenDSS.

- Temperatura de operación fotovoltaico
- Potencia de generador

Sin embargo, antes de hacerlo, se va a resumir cómo OpenDSS modela un sistema de placas solares (PV).

Consideraciones sobre el modelo PV en OpenDSS y características generales

La interfaz para el modelo del circuito de un panel solar es como la de un elemento de conversión de energía (PC, *Power Converter*) en OpenDSS. La configuración del panel solar se realiza de forma similar a cualquier otro dispositivo cuya carga, producción o capacidad de consumo dependa de una función. En este caso, la potencia activa, P, es una función de la irradiancia, la temperatura T y la potencia nominal P_{mpp} depende de una temperatura seleccionada y una irradiación de $1.0 kW/m^2$. Además, se aplica la eficiencia del inversor en la potencia de funcionamiento y el voltaje.

Potencia reactiva

La potencia reactiva se especifica por separado de la potencia activa y ambas pueden ser definidas de tal forma que sean fijas. Si se especifica la propiedad *PF*, el modelo tiene un factor de potencia de salida constante hasta que se cambie esta propiedad (es el modo por defecto). Si se especifica la propiedad *kvar*, se supone que el inversor intentará mantener ese valor a pesar del valor actual de la energía del panel. La salida de *kvar* real se deja caer si se supera la potencia nominal del inversor (*kVA*).

Estos son los únicos dos modos de potencia reactiva disponibles en la actual versión. La mayoría de los inversores de sistemas fotovoltaicos pueden ajustar la potencia reactiva para regular el voltaje de salida. Por lo tanto, las futuras versiones podrán ser modelados para regular el voltaje. (Esa capacidad se puede configurar en un elemento generador genérico con el modo $model = 3$).

LoadShapes y TShapes

Para simulaciones diarias, anuales y de ciclo de trabajo, la *irradiancia* y la temperatura, T, pueden ser incorporados al modelo con los objetos *LoadShape* y *Tshape* respectivamente, correspondiente al tipo de simulación que se realiza. Esto es lo que permite proporcionar una entrada de energía variable al modelo para las simulaciones de tiempo secuenciales. Si no se define *LoadShapes* para el modo de simulación ciclo de trabajo o anual, se toma de forma predeterminada el *LoadShape* del modo diario. Si éste tampoco está definido, el valor predeterminado del *LoadShape* es un multiplicador constante de 1,0, lo que quiere decir,

que no habrá variación de energía entregada.

Relación Potencia-Temperatura y Eficiencia. Objetos *XYCurve*
Los modelos de sistemas *PVsystem* usan objetos *XYcurve* para describir ciertas características de los paneles fotovoltaicos y los inversores. Este objeto *XYcurve* que representa a una curva, puede ser dada como una matriz de puntos o matrices separadas de los valores x e y. Se muestran dos ejemplos en totalmente equivalentes en el Código 3.2. Los valores de la matriz pueden estar separados por comas o por espacios en blanco. Los puntos en los objetos *XYcurve* se interpolan linealmente para determinar el valor real. Para las curvas utilizadas en el modelo de placa solar, por lo general es suficiente definir 4 ó 5 puntos, porque las curvas son relativamente suaves y monótonas.

Código 3.2 Dos métodos equivalentes para la creación del objeto XYcurve .

```
// curve in separate x, y array
New XYCurve.MyEff npts=4 xarray=[0.1 0.2 0.4 1.0] yarray=[0.86 0.9 0.93 0.97]
// curve as array of x,y values, in sequence
New XYCurve.MyEff npts=4 points=[0.1, 0.86 0.2, 0.9 0.4, 0.93 1.0, 0.97]
```

El objeto *XYcurve* se introduce para describir cómo la potencia (P_{mpp}) varía con la temperatura relativa (T). Dicha relación se establece entre la temperatura elegida y la potencia nominal máxima en kW/m^2. Escalando los valores de la curva *XYcurve* de modo que el factor 1 es el que define la temperatura en la que la potencia generada es máxima. Por lo general, esta potencia disminuye para temperaturas más altas y aumenta para temperaturas más bajas.

También se utiliza otra matriz de puntos para representar la curva de rendimiento para el inversor. En este caso se trata de una familia de curvas en función de la tensión del bus de corriente continua, el modelo en OpenDSS utiliza sólo una única curva hasta la fecha. Este modelo puede ser más sofisticado en futuras revisiones del software, sin embargo, este modelo simplificado parece adecuada para los estudios de impacto de distribución.

Variables de estado Como en otros elementos de conversión de potencia (PC) en OpenDSS, el *PVsystem* tiene variables de estado internas que se pueden consultar y observar. Los nombres de las variables presentes son:

1. ***Irradancia:*** irradiancia neta después de aplicar el factor de forma de la carga para el modo de simulación actual.
2. ***PanelkW:*** potencia neta, kW, que sale del panel teniendo en cuenta la irradiancia y la temperatura.
3. ***PTFactor:*** factor interpolado a partir de la curva de potencia-temperatura para la presente solución. Esto se aplica a la base P_{mpp} en referencia a la temperatura de referencia para calcular la potencia del panel (kW).
4. ***Efficiency:*** factor de eficiencia del inversor.

Se pueden observar los valores de estas variables de estado durante las simulaciones mediante la colocación de un elemento *monitor* (en el *modo*3) en el terminal del elemento PV. Además, para una condición de carga estática, se puede emitir el comando *"Show Variables"* para ver los valores de todas las variables de estado en el sistema.

Usando el modelo

Aquí se van a presentar los datos básicos para un modelo. Sin embargo se puede consultar cómo crear unas curvas de caracterización más realistas en [20] (Apéndice B ver figura B.1).

Los datos básicos del modelo son:

- La potencia media por panel en $1kW/m^2$ de irradancia a una temperatura constante entre 25º y 50 ºC.
- La variación con respecto a la unidad de P_{mpp} frente a la temperatura por cada $1kW/m^2$ de irradiancia.
- Una curva de eficiencia característica del inversor. Unidad de eficiencia por unidad de potencia.

Se pueden consultar algunos ejemplos de estas curvas en [6].

Soluciones estáticas. Snapshot

Para obtener una solución estática (instantánea), se debe configurar el valor de la irradancia en kW/m^2 y la temperatura del panel. Configurar la potencia activa (PF) o reactiva (kvar) si el factor de potencia por unidad por defecto no es satisfactorio. Por último, ejecutar el comando *Solve*. El programa iterará hasta encontrar la solución de la potencia de salida requerida. Si es necesario, la potencia reactiva (kvar) será recortada para mantener la potencia total por debajo del límite de *kVA*.

Simulaciones secuenciales en el tiempo

Para hacer simulaciones temporales, es necesario crear y asignar un objeto *LoadShape* con unas determinadas propiedades en función si la simulación es diaria, anual o de otro periodo de tiempo. Este *LoadShape* consiste en un vector, cuyos valores se multiplican por el valor base especificado para la irradiancia. Por otro lado, hay que crear y asignar un objeto *Tshape* para definir los valores de temperatura del panel que corresponda al *LoadShape* de la simulación, por lo tanto tiene que tener las mismas propiedades correspondientes a la simulación diaria, anual u otro ciclo de trabajo según corresponda.

Tras esto, se deberá configurar el modo de solución y el tamaño y número de intervalos de tiempo de la simulación. Ejecutar el comando *Solve* y observar los resultados de los monitores.

Se debe comprobar el script de ejemplo adjunto en el apéndice en la página 79 y las propiedades configurables del sistema de Panel Solar en OpenDSS.

3.2.2 Implementación de una Planta Solar Fotovoltaica

Se va a utilizar un modelo sencillo de generador solar para una primera prueba/modificación del script del circuito IEEE-37. Habría que añadir las líneas del código 3.3 al script original. Una curva que relacione potencia y temperatura, otra de la eficiencia del panel solar por unidad de potencia, la curva de irradancia y una última con las temperaturas a la largo del día (para 24 horas).

Código 3.3 Definición de curvas para implementación de panel solar en OpenDSS .

```
// La curva P-T representa Pmpp por unidad frente a la temperatura.
// En este caso simple es un Pmpp por cada 25 grados.
New XYCurve.MyPvsT npts=4 xarray=[0 25 75 100] yarray=[1.2 1.0 0.8 0.6]

// La curva de eficiencia representa unidad de eficiencia por unidad de potencia.
New XYCurve.MyEff npts=4 xarray=[.1 .2 .4 1.0] yarray=[.86 .9 .93 .97]

// Curva por unidad de irradiancia
New Loadshape.MyIrrad npts=24 interval=1 mult=[0 0 0 0 0 0 .1 .2 .3 .5 .8 .9 1.0 1.0 .99 .9 .7 .4 .1 0 0
    0 0 0]

// Curva de temperatura para 24 horas
New Tshape.MyTemp npts=24 interval=1 temp=[25, 25, 25, 25, 25, 25, 25, 25, 35, 40, 45, 50 60 60 55 40 35 30
    25 25 25 25 25 25]
```

Posteriormente, habrá que definir el panel solar como un *PVSystem* en OpenDSS, para ello se tiene que añadir al código las líneas de comandos 3.4, que tienen que contener:

- Bus al que se conectará el panel solar. En este caso y siendo fiel a la figura 3.2, se instalará en el nodo 730.
- La potencia activa y reactiva.
- La potencia máxima: Pmpp.
- La temperatura de funcionamiento.
- Se le asignan la curva de eficiencia, la curva P-T, la curva de irradiancia y temperaturas diarias previamente definidas en el código 3.3.

Código 3.4 Implementación de panel solar en OpenDSS .

```
// Definición del panel solar (PV) y conexión al nodo 730.
New PVSystem.PV phases=3 bus1=730 kV=12.47 kVA=500 irrad=0.8 Pmpp=500 temperature=25 PF=1
 ~ effcurve=Myeff P-TCurve=MyPvsT Daily=MyIrrad TDaily=MyTemp

solve ! Se resuelve para la especificada temperatura e irradiancia .

new monitor.m1 PVSystem.PV 1 mode=1 ppolar=no
new monitor.m2 PVSystem.PV 1

new monitor.mvars PVSystem.PV 1 mode=3


solve
solve mode=daily trapezoidal=yes

Export pvsystem /m
show mon m1

show mon m1
show mon m2
show mon mvars


Export monitors m1
Plot monitor object= m1 channels=(1 )
Export monitors m2
Plot monitor object= m2 channels=(1 ) base=[7200]
Export monitors m2
Plot monitor object= m2 channels=(9 )
```

Adicionalmente, sin ser esto necesario, se han añadido tres monitores que proporcionarán información sobre la simulación, se ha ejectudo el comando *Solve* para realizar la simulación y se ha definido el modo de ésta. Por último, se ejecuta el comando para mostrar los resultados de los diferentes monitores, y se indica que el resultado se muestre mediante gráficas (*plots*) y exportando los resultados numéricos a archivos de texto.

Una vez añadidas las anteriores sentencias en el script del IEEE-37, ya está modelado de una forma sencilla el primer panel solar y se ha configurado una simulación diaria de 24 horas. Se muestran de ejemplo algunas de las gráficas obtenidas mediante la herramienta *Plot* en la Figura 3.4.

Por otro lado, el comando *Show mon m1*, por ejemplo, muestra los resultados del monitor *m1*, que al estar configurado en el Código 3.4 en su modo de funcionamiento 1, muestra las potencias (activas y reactivas) en cada fase (Ver Figura 3.5). Si el modo de funcionamiento fuese 0 (por defecto), mostraría tensiones e intensidades o si fuese 3, se mostraría en pantalla las variables de estado en el caso de estar conectado a un elemento de tipo PC (*Power Conversion*); es decir, que en función de cómo se configure el monitor se obtendrán unos resultados u otros.

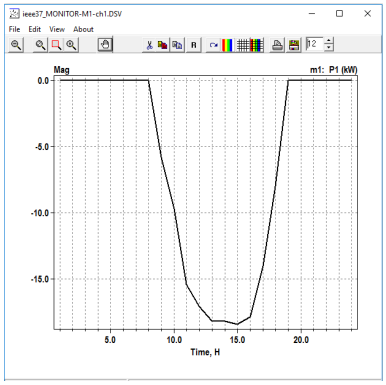
También se pueden obtener las pérdidas del circuito, que serán necesarias para implementar más tarde un algoritmo de optimización. El comando que hay que ejecutar es: *show losses*. Ver Tabla 3.2.

Tabla 3.2 Pérdidas del circuito.

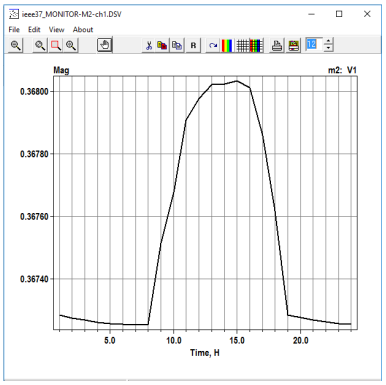
LINE LOSSES	60.0 kW
TRANSFORMER LOSSES	86.9 kW
TOTAL LOSSES	146.9 kW
TOTAL LOAD POWER	2441.8 kW
Percent Losses for Circuit	6.02

Una gráfica del circuito en la que vienen representadas las pérdidas (losses) en función del grosor de las líneas, se consigue simplemente con el comando: *Plot Type=Circuit Quantity=Losses Dots=Y Labels=Y*, obteniendo el resultado de la Figura 3.6.

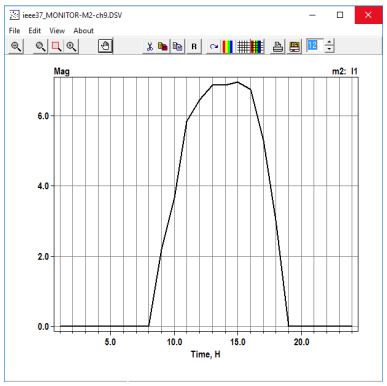
Aquí sólo se han mostrado algunas de las opciones gráficas y de obtención de resultados numéricos que ofrece OpenDSS, pero las opciones y variedades son muy altas. Para más información consultar el manual



(a) Potencia (kW) frente a hora del día.



(b) Tensión frente a hora del día.



(c) Intensidad.

Figura 3.4 Resultados monitor para panel solar.

ieee37_Mon_m1: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
hour, t(sec), P1 (kW), Q1 (kvar), P2 (kW), Q2 (kvar), P3 (kW), Q3 (kvar), P4 (kW), Q4 (kvar)
1, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
2, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
3, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
4, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
5, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
6, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
7, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
8, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
9, 0.00000, -5.79259, -0.000154441, -6.38422, -3.89316E-005, -6.1859, 9.65603E-005, -0, 0
10, 0.00000, -9.68967, -0.000358801, -10.6761, -0.000164123, -10.3466, -0.000246516, -0, 0
11, 0.00000, -15.4579, -0.000594798, -17.0264, -0.000398106, -16.5033, -0.000976591, 0, -0
12, 0.00000, -17.0907, -0.000195302, -18.8227, -0.000129394, -18.2449, -0.00035505, 0, -0
13, 0.00000, -18.2014, -0.000135929, -20.0448, -9.08676E-005, -19.4299, -0.000248832, 0, -0
14, 0.00000, -18.2018, -3.43627E-006, -20.045, -1.43604E-006, -19.4301, -5.5886E-006, 0, -0
15, 0.00000, -18.4514, -2.98832E-005, -20.3197, -2.02336E-005, -19.6964, -5.49247E-005, 0, -0
16, 0.00000, -17.8904, 6.41186E-005, -19.7023, 4.37867E-005, -19.0978, 0.000117906, 0, -0
17, 0.00000, -14.0608, 0.000344669, -15.4871, 0.000239202, -15.0111, 0.000627595, 0, -0
18, 0.00000, -7.99849, 0.000307483, -8.8122, 0.000224272, -8.54054, 0.000500258, 0, -0
19, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
20, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
21, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
22, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
23, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
24, 0.00000, 0, 0, 0, -0, -0, 0, 0, 0
```

Figura 3.5 Potencias medidas por el monitor m1.

de usuario del software.

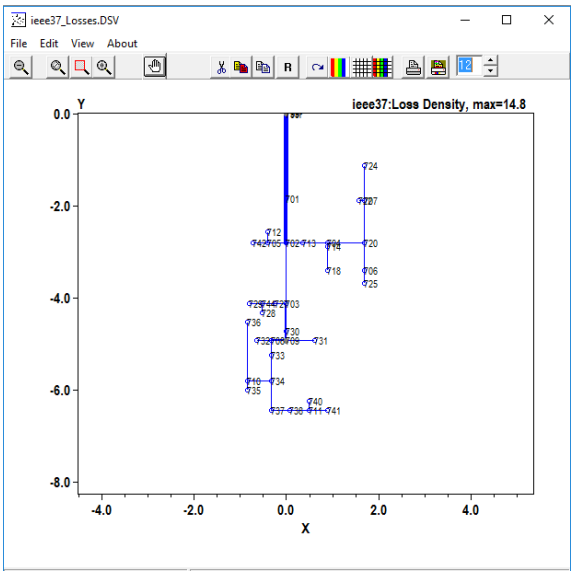


Figura 3.6 Representación gráfica de pérdidas en el circuito.

3.2.3 Modelado de un Generador eólico

OpenDSS permite también modelar generadores eólicos. Se comenzará con un modelo inicial de un generador eólico muy básico. Para ello, habría que definir dos curvas *LoadShape*, una que contenga los multiplicadores que simulen la potencia generada por el generador en función de la velocidad del viento (modelo del generador-turbina) y otro *LoadShape* temporal en el que, sobre el valor unidad, contenga los multiplicadores que modelan la velocidad del viento en los diferentes intervalos de tiempo en los que se pretenda simular. Obviamente, este último *LoadShape* siempre estará sujeto a incertidumbre, pues no se puede predecir con precisión cuál será la velocidad del viento. Podrá ser construido con mayor o menor precisión en función del modelo que se tome. Esto queda fuera del alcance de este trabajo, aunque si es del interés del lector, se puede consultar el trabajo [20], donde se indica cómo construir un modelo mediante modelos estadísticos y correcciones por variación de densidad en función de la altura y de la eficiencia del generador que puede ser implementado en OpenDSS. A modo de resumen, se adjunta la figura 3.7 extraída de este trabajo.

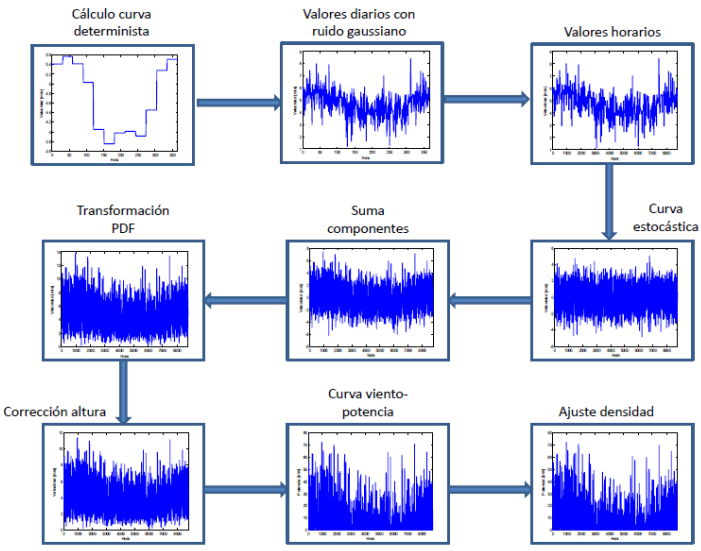


Figura 3.7 Diagrama de la creación de curvas modelo para generadores eólicos.

Para el caso del modelo de turbina-generador, en este documento se utilizará el proporcionado por OpenDSS, que se encuentra en formato tabla de Excel entre los archivos auxiliares del programa con el nombre *Zav-*

Wind.csv, que es el generalmente utilizado para este tipo de simulaciones. Este archivo consiste en un vector de valores experimentales que simulan la potencia generada por el generador en función de la velocidad del viento, que será proporcionado a OpenDSS mediante otro *LoadShape*.

Por lo tanto, para un modelo inicial de generador eólico, en la construcción del *LoadShape* se debería considerar tanto la potencia generada a partir de los valores corregidos de la velocidad del viento y la curva viento-potencia del conjunto turbina-generador (ver figura 3.8, como la velocidad del viento en función de la hora del día (modelo estadístico) para poder realizar una simulación temporal con OpenDSS.

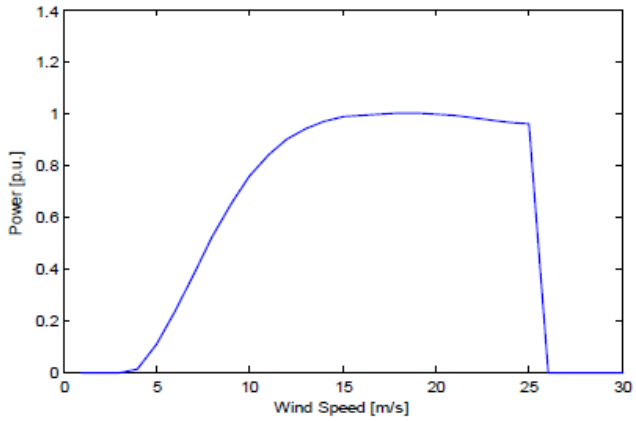


Figura 3.8 Potencia entregada por un generador eólico en función de la velocidad del viento.

Sin embargo, OpenDSS permite realizar modelos más avanzados de generadores eólicos para analizar y optimizar su penetración en un sistema de distribución. Se recomienda la consulta de [2], en el citado documento, se analiza y se muestran ejemplos en OpenDSS de la regulación del voltaje para la utilización de aerogeneradores en sistemas de distribución, considerando los cuatro tipos de tecnologías de aerogeneradores normalmente utilizadas: turbinas eólicas de velocidad fija, turbina eólica de velocidad variable, turbina eólica con doble generador de inducción y turbinas eólicas con convertidor completo. Cada tecnología tiene un método de control de potencia reactiva diferente y por tanto, un método de regulación de voltaje. Además la regulación de voltaje es frecuentemente un factor limitante en el objetivo de obtener sistemas eólicos de alta penetración, ya que en este tipo de sistemas puede haber caídas de voltaje al conectarse a la red, además de las variaciones de la velocidad del viento que pueden generar fluctuaciones en la generación de energía. Por ello, con el fin de regular la tensión de salida, se recomienda equipar al generador con equipos de reguladores de tensión y bancos capacitores. Tras esto, será necesario incorporar un transformador que adapte la tensión de salida del conjunto aerogenerador y regulador de tensión a la tensión de trabajo del sistema de distribución. Todo esto puede ser implementado en OpenDSS.

El sistema de pruebas que utilizan los autores del documento [2] es el que se muestra en la figura 3.9. Como se puede observar, se han incorporado los reguladores de tensión y bancos de capacitadores.

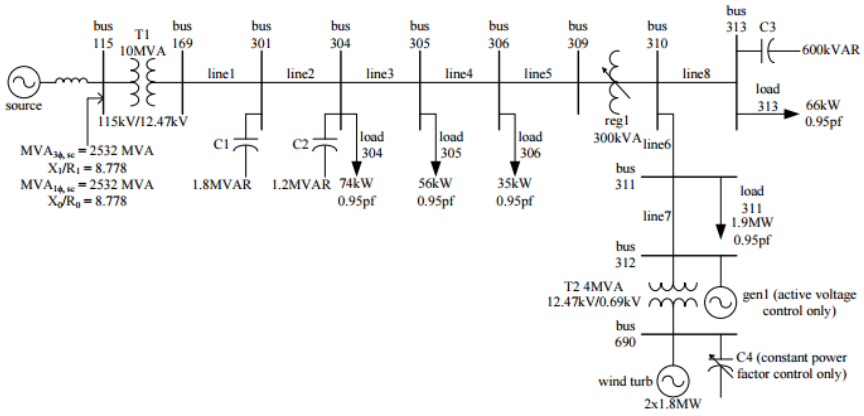


Figura 3.9 Circuito radial de prueba de reguladores de tensión y bancos capacitadores para un generador eólico.

3.2.4 Implementación de un Generador eólico

De forma análoga al caso anterior, primero se definen los *LoadShapes* necesarios para el modelo. Posteriormente se crea y configura el generador eólico como un *genwind*, y por último se incluyen los monitores o elementos de control que se quieran instalar (estos últimos son opcionales).

Código 3.5 Implementación de generador eólico en OpenDSS .

```
! Configuración del loadshape para el generador eólico.
New Loadshape.day 24 1.0
~ mult=[.3 .3 .3 .35 .36 .39 .41 .48 .52 .59 .62 .94 .87 .91 .95 .95 1.0 .98 .94 .92 .61 .60 .51 .44]
New Loadshape.wind 2400 0.00027777 mult=(file=ZavWind.csv) action=normalize

// Definición del generador eólico de 8 MW
New generator.genwind bus1=722 kV=12.47 kW=8000 pf=1 conn=wye duty=wind Model=1

new monitor.mw1 generator.genwind 1 mode=1 ppolar=no
new monitor.mw2 generator.genwind 1

solve
solve mode=daily trapezoidal =yes

show mon mw1
show mon mw2
```

Una vez modelado el generador eólico y los diferentes monitores de control, se obtienen algunas gráficas que representan tensiones de trabajo, intensidades y potencias generadas. Se observa que la potencia generada y las intensidades de trabajo del generador eólico de 8 MW es mucho mayor que la potencia e intensidad del sistema de panel solar. También las pérdidas de potencia en la zona donde se ha instalado el generador eólico son mucho mayores (ver Figuras 3.10). Se deberá ajustar mejor la configuración de todos los elementos para obtener unos mejores resultados y considerar el instalar reguladores de tensión y bancos capacitadores.

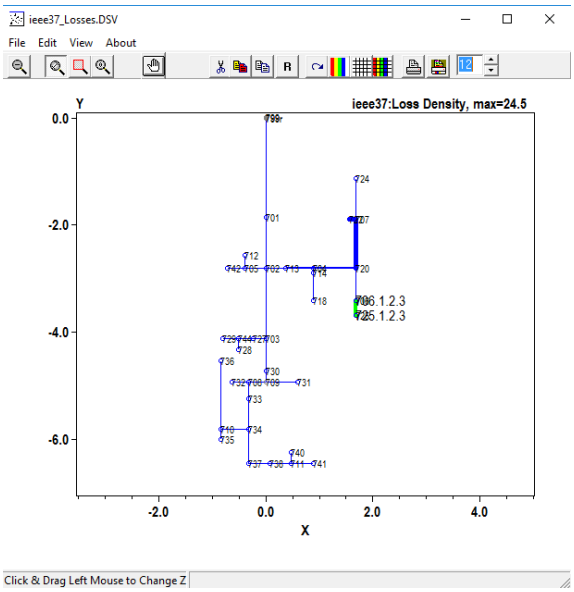


Figura 3.10 Representación gráfica de pérdidas en el circuito con sistema eólico (nodo 722) y solar (nodo 730).

3.2.5 Modelado de Baterías. *Storage Element*

El *Storage Element* o elemento de almacenamiento en OpenDSS es esencialmente un generador que puede ser utilizado para producir energía (fase de descarga de batería) o para consumir energía (fase de carga de batería) dentro de su rango de potencias y su capacidad de almacenar carga. Por ello, el modelo de las baterías ha sido desarrollado a partir del elemento *generador*. Así, ha heredado algunas de sus características como el contador de energía incorporado.

Un elemento de almacenamiento puede actuar de forma independiente o ser controlado por un elemento *StorageController*, es decir un controlador de almacenamiento, y también a través del servidor COM como se verá en el capítulo 5.

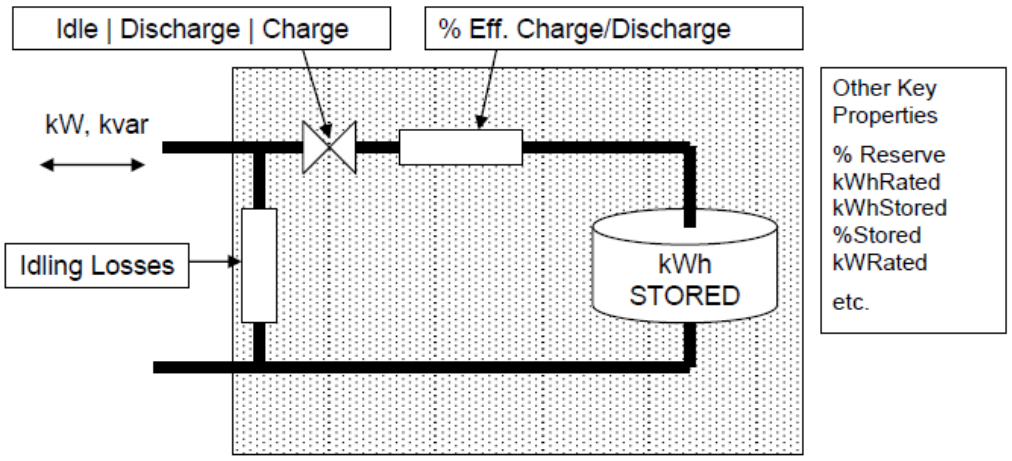


Figura 3.11 Esquema básico del Elemento de Almacenamiento.

El modelo de la batería puede ser utilizado en un modo de simulación ***Snapshot*** para el control del flujo de potencia del elemento en uno de sus estados posibles (Véase la Figura 3.11). En este caso, sólo se tendría que establecer uno de los estados [IDLING | CHARGING | DISCHARGING] , que hacen referencia ha

estado de ralentí o reposo, carga y descarga respectivamente, y luego solo habría que resolver.

Hay que tener en cuenta que sólo habrá una descarga de potencia si el nivel de carga actual (*kWhStored*) es mayor que el nivel de reserva que se ha especificado (VEASE LA TABLA DE PROPIEDADES DE BATERÍAS. Tabla B.1). Por otro lado, solo habrá una carga efectiva cuando el valor de la energía almacenada (*kWhStored*) es menor que la capacidad de almacenamiento, como es lógico. También es importante tener en cuenta que se puede especificar la velocidad de descarga con la propiedad *%Discharge* y la tasa de cargas de la batería con la propiedad *%Charge*.

Sin embargo, los modos de simulación más adecuados para este modelo son ***Daily*** (modo diario), ***Yearly*** (modo anual) y ***DutyCycle*** (modo ciclo de trabajo). Lo más habitual es utilizar los modos *Yearly* y *Daily* para analizar los movimientos de energía durante un periodos de tiempo que puede ir de los minutos a las horas. El modo *DutyCycle* se utiliza para estudiar la eficacia de almacenamiento para compensar las variaciones de energía en un circuito a corto plazo, por ejemplo para analizar el efecto de una nube pasajera sobre la generación de energía solar fotovoltaica.

El elemento de almacenamiento también puede producir o absorber potencia reactiva (VAR) en función del nivel de KVA del inversor. Es decir, si un objeto *StorageController* solicita una cierta cantidad de reactancia (kvar), el elemento de almacenamiento se lo proporcionará si el inversor tiene capacidad para hacerlo. Con respecto a esto, también es importante tener en cuenta que en el modo *IDLING* (ralentí), se puede producir o absorber reactancia.

Las pérdidas son importantes en la evaluación de los sistemas de almacenamiento, por ello, el modelo permite establecer una eficiencia de carga y otra de descarga. Los valores predeterminados son del 90 % para la carga y un 81 % para la descarga. Estos valores se pueden modificar en las propiedades del elemento. Además, las pérdidas del modo *ralentí* pueden ser especificadas. Estas pérdidas representan la energía requerida por los controladores internos, calentadores, refrigeradores, etc., para mantener la temperatura adecuada de la batería. Esta pérdida se especifica como un único valor medio expresado en tanto por ciento de la potencia nominal (por defecto = 1 %) y se modela como una impedancia constante en paralelo con el sistema de alimentación. Por supuesto, este valor se puede ir modificando sobre la marcha con un script.

En el modo de *carga* y *descarga*, la batería es generalmente modelada como un modelo simple constante (P + JQ) (Por defecto, $model = 1$). También está disponible un modelo de Z constante, ($model = 2$).

Modos de funcionamiento.

El usuario elige el modo de funcionamiento de la batería entre los siguientes:

[DEFAULT | FOLLOW | EXTERNAL | LOADLEVEL | PRICE]

Los valores de referencia para carga (***ChargeTrigger***) y para descarga (***DischargeTrigger***) intervienen de forma diferente en cada modo. Básicamente, cuando un nivel de referencia es superado la batería puede cambiar a otro modo.

- **Modo DEFAULT:** Modo por defecto. Los niveles de referencia siguen los valores especificados en el *LoadShape* correspondiente al presente modo de funcionamiento (Daily, Yearly, Dutycycle). Por ejemplo, en el caso de una solución de tipo Yearly, el dispositivo de almacenamiento descargará hasta el ratio específico de descarga (fijado) cuando el valor del *LoadShape* asociado al modo Yearly sobrepase el *DischargeTrigger* (valor de referencia de descarga). El dispositivo permanecerá en el modo de descarga hasta que el valor del *LoadShape* sea inferior al nivel de referencia o hasta que el dispositvo agote su energía almacenada hasta el nivel de reserva. Cuando el valor del *LoadShape* cae por debajo del valor de referencia de carga (*ChargeTrigger*), la batería empezará a cargarse a la velocidad especificada, hasta que esté al 100 % o se requiera otro ciclo de descarga.

  Hay un tiempo predeterminado para iniciar el ciclo de carga (*TimeChargeTrigger*) que anulará el criterio de nivel de referencia para carga (*ChargeTrigger*) si se especifica. Esto es para asegurar que el proceso de carga se llevará a cabo incluso si el nivel de carga almacenada no se encuentra por debajo del valor *ChargeTrigger* en cualquier momento durante el día. Para desactivar esta característica hay que establecer un valor de *TimeChargeTrigger* negativo, si no se especifica, se mantendrá el tiempo de carga por defecto.

Por ejemplo, el tiempo de carga por defecto es a las 2 AM. En ese momento, la batería intentará cargar incluso si la carga no ha caído por debajo del valor de *ChargeTrigger*. Esta es una estrategia para asegurar que el dispositivo de almacenamiento está siempre totalmente cargado para el pico de energía requerido del día siguiente.

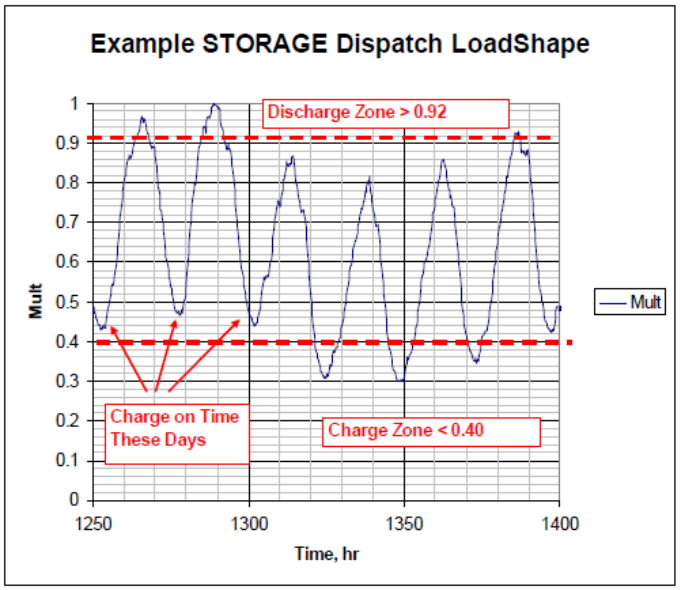


Figura 3.12 Ilustración del modo de funcionamiento DEFAULT.

En la Figura 3.12 se muestra gráficamente en que rangos habría carga o descarga en función de los valores de referencia(para descarga es 0.92 y para carga es 0.4) y del *LoadShape* asignado (línea azul en el gráfico). Cuando el valor del *LoadShape* supera el valor de 0.92, la batería empezará a descargarse a la velocidad de descarga que esté previsto que lo haga. Del mismo modo, cuando el valor del *LoadShape* cae por debajo de 0.4, comienza el ciclo de carga. Los días en el que el valor del *LoadShape* no cae por debajo del 0.4, a las 2 AM (tiempo de carga por defecto) se iniciará la carga con la velocidad asignada.

- **Modo FOLLOW:** La salida de potencia activa (kW) y de potencia reacitva (kvar) de la batería sigue los valores definidos en el *LoadShape* activo hasta que la batería esté o bien agotada o completa. La batería descarga para valores positivos y carga para los valores negativos del *LoadShape*. La carga y descarga son proporcionales a la propiedad *kWrated* de la batería. Esto se ilustra en la Figura 3.13.

  El ciclo de descarga se configura para seguir nominalmente la forma del pico diario que se produce aproximadamente a las 17:00 horas. Si se tiene una batería de 1000 kWh con un inversor de 250 kW. Con un período de 4,5 horas, esto sería casi la descarga de la batería, alcanzando un pico de 250 kW a 5 de la tarde (las 17:00 horas). A las 2 de la mañana (02:00 AM) del día siguiente, el ciclo de carga comenzaría con un aumento gradual de hasta 250 kW durante 0,5 horas y luego continuaría hasta las 6 PM si es necesario. Obviamente, habría que simular al menos 2 días para ver si esto funciona de la manera que se quiere, porque la unidad de almacenamiento comienza completamente cargada en la simulación, por lo tanto no habría ninguna carga en el primer ciclo.

  La figura 3.14 muestra un resultado que podría venir de la simulación descrita. El *LoadShape* en la figura 3.13 se superpone sobre la potencia entre los terminales de la batería. Un valor positivo denota que la batería se está descargando (OpenDSS utiliza como criterio que una potencia negativa representa que está saliendo del terminal).

  En esta simulación la batería se descarga hasta el nivel de reserva del 20 %. El ciclo de carga especificado es un poco más extenso de lo necesario para recargar la batería la completo. Esta simulación diaria se podría utilizar sin problemas para una simulación anual, sólo habría que repetir una y otra vez. Es importante tener en cuenta que en el modo FOLLOW no hay ninguna garantía de carga, es necesario que el *LoadShape* esté definido negativamente durante determinados periodos para forzarla. En general, se modela con algo más de área bajo la curva de carga que de descarga, para así compensar las pérdidas.

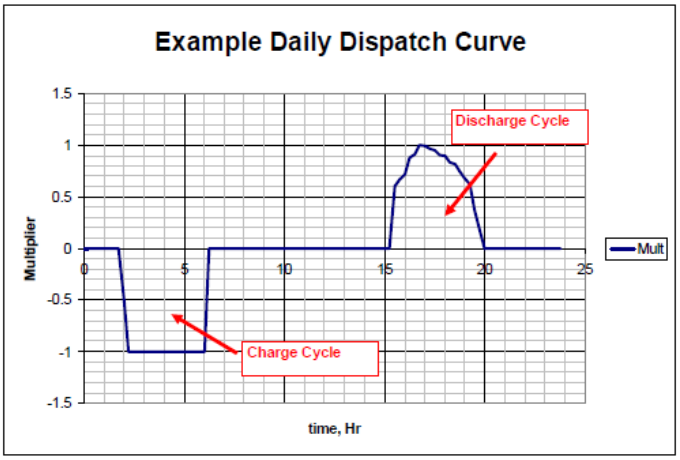


Figura 3.13 Ejemplo de una simulación diaria de carga y descarga en el modo FOLLOW.

Si la batería no tiene capacidad suficiente (kWh) para completar el ciclo de descarga, su estado cambiará al ralentí. Pasará lo mismo si el dispositivo se llena antes de que finalice el ciclo de carga. Las pérdidas de carga y descarga se contabilizan durante estos ciclos aunque no se completen.

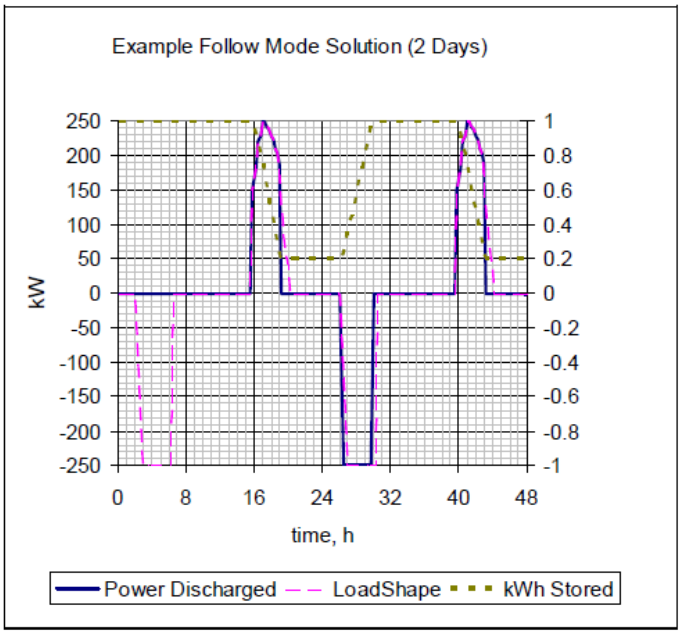


Figura 3.14 Ejemplo de resultados de simulación.

Aunque no se muestra, hay flujo tanto de potencia activa (kW) como reactiva (kvar). Primero se distribuye la potencia activa (kW) y luego la reactiva (kvar) hasta que se completa la potencia aparente, kVA, de la batería. Al igual que con todos los objetos *Loadshape*, si no se especifica la propiedad *"Qmult"*, los multiplicadores de potencia reactiva se pondrán por defecto con el mismo valor que *"Mult"* (multiplicadores del *LoadShape* para la potencia activa).

El script de OpenDSS (3.6) es el que fue usado para generar los resultados de la figura 3.14.

(Nota: el *LoadShape* utilizado fue el representado en la figura 3.13 y cargado en el archivo "storagetestshape.csv").

- **Modo LOADLEVEL o PRICE:** En este modo, el cambio de estado de la batería no está controlado por un *LoadShape* asociado a la batería en particular, sino que el algoritmo puede estar basado en un nivel de carga global (LOADLEVEL) o en un precio (PRICE). (Consultar *Help* para más información sobre las siguientes opciones).

– El valor del nivel de carga global (LOADLEVEL) se fija con el siguiente comando:

Set LoadMult= *valorporunidad*
Set DefaultDaily=MyDailyLoadShapeName (simulaciones diarias)
Set DefaultYearly=MyYearlyLoadShapeName (simulaciones anuales)

– Análogamente, el valor del PRICE global se fija con el comando: Set PriceSignal= *valor*
Set PriceCurve=MyPriceCurveName

La curva de precios (PriceCurve) son definidas como un objeto *PriceShape* en lugar de hacerlo con un *LoadShape*.

Con cualquiera de los tres modos de funcionamiento que se han descrito, se pueden generar una gran diversidad de simulaciones con diferentes comportamientos de carga y descarga, por ejemplo definiendo diferentes *LoadShapes* con Excel o Matlab. Se puede usar este procedimiento para simular el efecto de tener un controlador de almacenamiento central, incluso si no se define, mediante la asignación de las mismas formas de carga (LoadShape) a una serie de baterías dispersas. Las posibilidades son muy variadas.

- **Modo EXTERNAL:** La particularidad de este modo de funcionamiento es que el dispositivo de almacenamiento no determina su propio estado, en lugar de ello, un elemento controlador de almacenamiento (*StorageController*) determinará las funciones y estado del dispositivo bajo control. Este modo se establece automáticamente cuando un controlador de almacenamiento toma el control de un dispositivo de almacenamiento.

  Sin embargo, aunque el control del dispositivo lo hará el *StorageController*, se pueden utilizar *LoadShapes* para compensar zonas de simulación que se salgan del controlador, de seguimiento de carga, etc. Obviamente, para este modo, se requiere un *StorageController* para realizar las simulaciones en tiempo real.

  Además en este modo de funcionamiento, es posible controlar el estado del dispositivo mediante secuencias de comandos, a través de un script o bajo el control de un programa como Matlab y la interfaz COM que ya se describió en el capítulo 2.

3.2.6 Implementación de Baterías

La herramienta OpenDSS da la opción de configurar y modelar un sistema de almacenamiento de energía (*Storage Element*). Se van a describir algunas de las opciones que el software ofrece, para luego configurarla en función de las necesidades de la simulación.

La batería se implementará en el nodo 720 del sistema de distribución IEEE 37. Para un primer ejemplo, se considerará además un *LoadShape* propio para el control de la batería, aunque como ya se ha descrito, son posibles otras formas de control de la batería. Para ello se recurre a los siguientes comandos (3.6)

Código 3.6 Implementación de baterías en OpenDSS .

```
Clear
New Loadshape.DailyShape npts=96 minterval=15 mult=[ file = storagetestshape .csv]
New Storage. Battery  phases=3 Bus1=720 kV=12.47 kW=250 kWrated=250 kWhrated=1000
~ dispmode=follow daily=dailyshape
set  voltagebase =[12.47]
calcv
new monitor.PQ storage . battery  1  ppolar=no mode=1
new monitor.Vars  storage . battery  1  mode=3
solve
solve  mode=daily step=15m number=(2 96 *)
show mon PQ
show mon vars
```

Como primera aproximación, el *StorageLoadShape* ha sido creado mediante Matlab y almacenándolo en un archivo .csv, ajustándolo para que la batería se recargue cuando la planta solar modelada esté generando los picos de potencia (de 10 a 16 horas aproximadamente) y se descargue durante las horas sin sol (de 19 hasta descarga). En la figura 3.15 se puede ver gráficamente. Esto se puede hacer de una forma más eficiente mediante los objetos *StorageController* (Ver [13]).

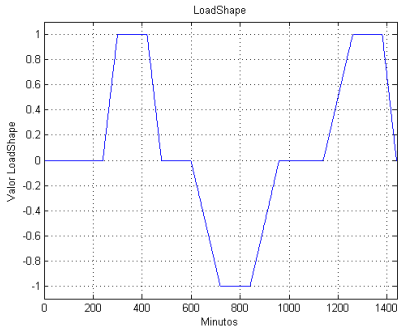


Figura 3.15 Storage LoadShape diseñado en Matlab.

En la figura 3.16 se muestra la carga de la batería en una simulación con el *LoadShape* anterior.

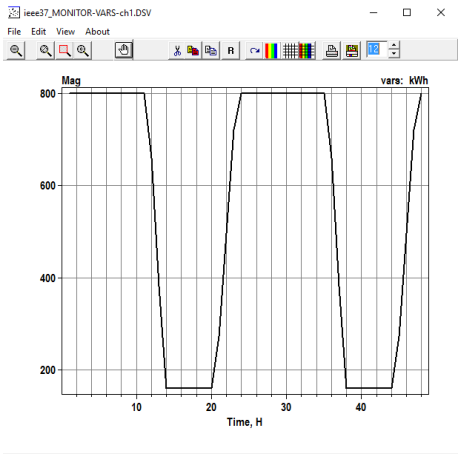


Figura 3.16 Nivel de carga de la batería (kWh) en una simulación diaria.

Y en la siguiente figura (3.17) se muestra las potencias de carga (curva negra), de descarga (curva roja) y las pérdidas (curva azul).

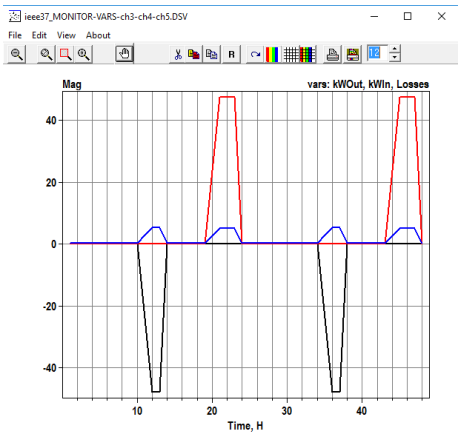


Figura 3.17 Carga, descarga y pérdidas de la batería.

3.3 Otros elementos

3.3.1 Coche eléctrico conectado a la red. PEV.

El objetivo es modelar el coche eléctrico conectado a la red con un modelo con incertidumbre como el descrito en [17]. Sin embargo, se comenzará con un modelo simple al que se le irán incorporando características posteriormente. Para optimizar la potencia utilizada por el sistema de distribución y reducir los costes económicos en un sistema real con coches eléctricos, es necesario incorporar medidores inteligentes y sensores que controlen la energía consumida en cada nodo; y un intercambio de información con un centro de control a través de una red de comunicación. Además se puede incluir otro tipo de información como los parkings disponibles para este tipo de vehículos, información de la compañía gestora, identificación de los consumidores, etc.. El centro de control, tras procesar esa información, puede enviar a los dispositivos de control la potencia que se debe consumir en las cargas variables, la potencia suministrada por los diferentes generadores, la energía almacenada en las baterías y otra información relevante para la gestión del sistema (más información detallada sobre las redes de comunicación en redes inteligentes con coches eléctricos (PEV) puede ser encontrada en [15]). Sin embargo, el control y procesamiento de esta información escapa del propósito del presente documento, que se centrará solo en el modelado del sistema de distribución de potencia y en su simulación.

En este documento, en concordancia con el ya citado [17], sólo se considerarán dos tipos de cargas para los coches eléctricos, y en ambos tipos todos los coches deben estar completamente cargados antes de desconectarse de la red:

- **Carga no controlada:** El proceso de carga comienza inmediatamente al llegar el vehículo. El proceso de carga será a la velocidad máxima que permita el sistema y no parará hasta que la batería esté completamente cargada, una vez que esto ocurra, el cargador de batería cambiará al estado de espera (*stand-by*).
- **Carga limitada:** La demanda de carga de la batería se satisfará de forma distribuida en todo el periodo en el que el vehículo esté disponible para cargar.

Para el modelado que se muestra en este documento, se usarán unas simplificaciones en la estimación de carga y en el modelo predictivo. Se pueden encontrar trabajos de desarrollo e investigación sobre estimación de tiempos de carga de coches eléctricos en condiciones reales de tráfico y el comportamiento de los usuarios en [16].

Con respecto al modelado en OpenDSS de un coche eléctrico, la forma más sencilla de hacerlo es mediante un elemento de tipo *Load* y asignarle un *LoadShape* que tenga las características de carga. También está la opción de modelarlo como un generador o como un elemento de almacenamiento. Debido a las características del modelo de PEV que aquí se tiene en cuenta, el más adecuado es el de elemento de almacenamiento, ya que esto permite controlar los ratios de carga y descarga, aunque aumenta significativamente la complejidad de la configuración.

El coche eléctrico será conectado al nodo 737. Para empezar habrá que definir las características de la batería: ratio de carga y descarga, eficiencia, capacidad, etc. Por otro lado, para el *LoadShape* o el sistema que se elija para definir el comportamiento de la batería, habrá que elegir (o estimar) unos patrones de carga y descarga que simule el comportamiento de los usuarios al hacer uso de los coches eléctricos. También se deberán definir dos escenarios en los que la velocidad de carga sea diferente, una de carga de alta velocidad y otra distribuida en función del tiempo de conexión.

4 Control de OpenDSS desde Matlab

En este capítulo se tiene como objetivo el explicar mediante ejemplos cuáles son algunas de las cuestiones básicas que hay que conocer para utilizar conjuntamente OpenDSS y Matlab sin necesidad de ejecutar OpenDSS directamente.
La principal fuente consultada para el desarrollo del siguiente capítulo-tutorial de uso de Matlab ha sido el documento de Sandia Corporation [10], del que se ha realizado una adaptación, obteniendo gran parte del código y figuras.

4.1 Control de OpenDSS por Matlab

Efectivamente, se puede hacer uso de OpenDSS sin necesidad de utilizar su interfaz gráfica (a partir de ahora OpenDSS GUI), esto es a través de una aplicación independiente, en este caso se utilizará Matlab (pero también por Visual Basic, Excel o Python) a través del COM Server, el cual ha sido desarrollado como una aplicación DLL.

Matlab utiliza su servidor ActiveX integrado para comunicarse con el COM Server de OpenDSS, de esta manera dicho servidor será la interfaz entre ambos programas (Ver figura 4.2). Esta característica proporciona una gran capacidad de análisis de la información, permitiendo incluso implementar algoritmos que utilicen los resultados de las simulaciones de OpenDSS. La figura 4.1 muestra cómo interactúan los diferentes módulos dentro de la estructura de OpenDSS [11].

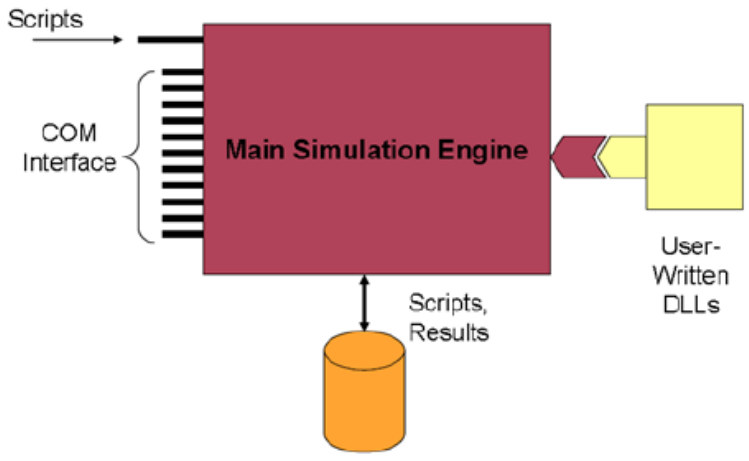


Figura 4.1 Estructura OpenDSS.

La inicialización del COM Server es realizada a través de una función creada en el espacio de trabajo de Matlab. El código utilizado para crear dicha función se puede ver en el siguiente fragmento de código (4.1). Una vez que OpenDSS ha sido inicializado, se puede compilar un circuito mediante su script (.DSS). En el siguiente ejemplo se carga el circuito IEEE de 37 buses. Una vez compilado, es posible enviarle instrucciones

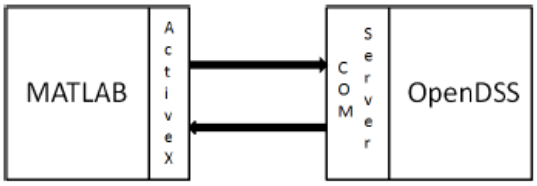


Figura 4.2 Comunicación Matlab y OpenDSS.

utilizando el comando *DSS.TextCommand* =, el cual debe estar acompañado por una cadena que contenga la instrucción que OpenDSS debe realizar con la sintaxis que reconoce este software.

Hay mucha documentación para cada función contenida en la herramienta, a la que se puede acceder de la forma habitual en la que se hace en Matlab:

```
help getBusInfo
```

4.1.1 Iniciar la interfaz COM

El primer paso para iniciar la interfaz COM es hacer una llamada la función *DSSStartup*. Hay varias formas de hacerlo, pero la siguiente es la más habitual:

```
[DSSCircObj, DSSText, gridpvPath] = DSSStartup;
```

DSSStartup inicia OpenDSS en segundo plano y devuelve un puntero con tres salidas:

- ***DSSCircObj***, que es el puntero de la interfaz COM. Contiene el circuito activo (*DSSCirObj.ActiveCircuit*), que no ha sido aún compilado y la interfaz de texto para OpenDSS (*DSSCircObj.Text*). *DSSCirObj* estará vacío hasta que un circuito sea compilado.
- ***DSSText***, es la interfaz de texto contenida dentro de *DSSCirObj*. Ha sido definido de esta manera para facilitar el uso dentro de la ventana de comandos de Matlab. *DSSCirObj.Text.Command*, *Text.command* y *DSSText.Command* devolverán el mismo resultado en forma de interfaz de texto.
- ***gridpvPath***, en el ejemplo es una cadena que contiene la ruta de localización en el ordenador para la herramienta.

DSSStartup devolverá un error si MATLAB no es capaz de crear la conexión con OpenDSS. El motivo más común es que OpenDSS no esté correctamente instalado en el ordenador o si es una versión antigua.

Hay que notar que el programa OpenDSS que se conecta con MATLAB por el servidor COM es diferente al ejecutable con interfaz gráfica de OpenDSS (GUI). Es decir, ninguna información, circuito, solución o parámetro configurado en la interfaz gráfica de OpenDSS será mostrado por la versión del servidor COM de OpenDSS y viceversa.

4.1.2 Compilando el circuito

Para abrir un circuito creado en OpenDSS desde Matlab, se usa la interfaz de texto con el código "compile", indicándole la ruta en el caso de que el circuito en OpenDSS y el archivo .m de Matlab se encuentren en directorios diferentes:

```
DSSText.Command = 'Compile "D:\Aeroespacial\TFG\Prueba OpenDSS\SimpleC.DSS"';
```

Es recomendable utilizar siempre la ruta del circuito en formato .DSS con el comando "compile" para evitar errores.

El tener abierto en segundo plano OpenDSS con un circuito compilado no impide tener también abierto y utilizar la interfaz gráfica de OpenDSS (GUI), pero como ya se indicó, ambos programas son independientes.

Cambios en el circuito en el OpenDSS GUI no serán reflejados en el circuito compilado en MATLAB. Para ello habría que guardar los cambios en el circuito en OpenDSS GUI y volver a compilar desde Matlab ese mismo circuito. Otra opción es hacer esos mismos cambios directamente desde Matlab con la interfaz de texto *DSSText*.

4.1.3 Obtener información de OpenDSS desde Matlab

Ahora que la interfaz COM ha sido iniciada y el circuito compilado, se puede comenzar a utilizar la ventana de comandos para la interactuación entre los dos programas.

Se puede llamar a la función *DSSCircObj.methods* (o equivalente si se ha definido como *DSSObj*) para ver los métodos disponibles que se pueden utilizar para interactuar con la interfaz. Usar la instrucción *DSSCircObj.get* para ver la interfaz principal. También se puede consultar la documentación de OpenDSS para más información.

El comando *DSSCircObj.get* devuelve algunos otros punteros de objetos de la interfaz del servidor COM en OpenDSS. Uno de estos sub-punteros es la interfaz de *ActiveCircuit*, que hace mención al circuito compilado en OpenDSS y contiene todos los parámetros y todas las soluciones de flujo de potencia. Es útil redefinir este puntero para utilizarlo con comandos más cortos que cumplirán exactamente la misma función sin tener que llamar a la función completa *DSSCircObj*:

```
DSSCircuit = DSSCircObj.ActiveCircuit;
```

Ahora llamando a *DSSCircuit.methods*, se verán los "métodos" pertinentes al circuito resuelto. Otra vez, usando *DSSCircuit.get*, se verán todos los diferentes campos e interfaces presentes en el circuito:

```
DSSCircuit.methods
DSSCircuit.get
```

En este caso, se obtendrán algunos punteros de interfaces más de OpenDSS COM, cada uno referido a un elemento específico en el circuito. Análogamente, como ya se ha hecho con *ActiveCircuit*, se pueden volver a utilizar los comandos *get* y *methods* en estos elementos para obtener más información y posibilidades, incluso se pueden consultar campos específicos de dichos elementos:

```
DSSCircuit.Lines.methods
DSSCircuit.Lines.get
DSSCircuit.Lines.LineCode
DSSCircuit.Capacitors.get
DSSCircuit.Capacitors.Name
```

Sin embargo, se debe tener en cuenta también que la mayoría de estos campos en estas interfaces tienen información sobre un elemento individual, generalmente el primer elemento por defecto. Para obtener información de un elemento en particular (una línea o un capacitador concreto), tendrá que ser *activado* previamente en la interfaz.

Esta es una importante observación para entender cómo utilizar las iteraciones para obtener todos los datos de un circuito. Los métodos *.first* y *.next* que son devueltos al hacer la llamada *DSSCircuit.Lines.get* son usados para cambiar el índice del objeto. Se usa el método *.first* para asegurarse de que se resetea el índice y se empieza por el primer objeto en la lista de un determinado tipo de elemento (línea, capacitador, generador, etc.) y entonces se usa el método *.next* para ir iterando paso a paso en dicha lista. Por ejemplo:

```
% Se selecciona el primer elemento de la lista de transformadores
DSSCircuit.Transformers.first;
% Se obtiene el número total de transformadores
numXfmr = DSSCircuit.Transformers.count;
% Se asigna previamente
xfmrNames = cell(numXfmr, 1);
% Se itera
for ii = 1:numXfmr
  %Se obtiene el nombre del elemento correspondiente al índice ii
```

```
    xfmrNames{ii} = DSSCircuit.Transformers.Name;
    % Se continúa al siguiente transformador
    DSSCircuit.Transformers.Next;
end
```

Si se observa el uso de la variable *numXfmr*, se podría pensar que es más útil no utilizarla y hacer sólo la llamada *DSSCircuit.Transformers.count* en las dos ubicaciones en las que se ha utilizado *numXfmr*, sin embargo, la función *DSSCircuit.Transformers.count* tendría que pasar dos veces por el servidor COM y ralentizaría considerablemente el programa, por lo tanto es más eficiente llamar a esa función una sola vez y usar una variable del espacio de trabajo (en este caso Matlab).

El ejemplo anterior del transformador es simplemente una demostración de cómo utilizar la iteración con las interfaces. Pero este método no es el más útil para obtener todos los nombres de los transformadores. Esto pone de relieve otro hecho importante sobre la interfaz de los elementos, y es que a pesar de que muchos campos serán específicos para un solo elemento (*name* por ejemplo), hay varios métodos y campos que devuelven o contienen información global, lo que ayuda a evitar procesos de iteración y hacer más eficiente el código. El siguiente método hace exactamente lo mismo que el bucle anterior pero con una sola instrucción:

```
xfmrNames = DSSCircuit.Transformers.AllNames;
```

4.1.4 Elementos activos

Cuando se interactúa con el servidor COM, hay dos principales formas con las que obtener información de un determinado elemento del circuito. La primera es la mostrada en la sección anterior y supone utilizar la interfaz específica del tipo de elemento (por ejemplo la interfaz de línea o interfaz del capacitador). Otra interfaz, la del *elemento activo*, puede ser también usada para encontrar información de algún tipo de elemento.

Llamando a la función *DSSCircuit.ActiveCktElement.get* se verá una lista de campos que, individualmente, pueden o no ser aplicadas a cada tipo de elemento del circuito. También se verá que hay algunos datos correspondientes a un determinado tipo de elemento pero que no están presentes en la interfaz del elemento. Esta es la razón por la que el *elemento activo* es tan útil: contiene datos relevantes que no se pueden encontrar en otros lugares. En general, las interfaces de clase (líneas, transformadores, etc.) contiene información sobre el elemento del circuito (rangos de funcionamiento, conexiones, impedancias, etc.) y la interfaz de *elemento activo* contiene la solución de los flujos de energía/potencia.

Después de llamar a *DSSCircuit.ActiveCktElement.methods*, se puede comprobar que en este modo no existen los métodos *.First* o *Next*. Esto es porque la interfaz de *elemento activo* requiere que el elemento sea seleccionado manualmente mediante la función *DSSCircuit.SetActiveElement*. Se muestra un ejemplo a continuación:

```
%Se consiguen los nombres de línea y se monta la estructura
lineNames = DSSCircuit.Lines.AllNames;
Lines = struct('name',lineNames);
% Se itera y se recuperan los datos de los buses
for ii=1:length(Lines)
% Se nombra elemento activo a las diferentes líneas
DSSCircuit.SetActiveElement(['line.' Lines(ii).name]);
% Se obtiene el nombre de los buses, una matriz de 2 columnas
lineBusNames = DSSCircuit.ActiveElement.BusNames;
Lines(ii).bus1 = lineBusNames{1};
Lines(ii).bus2 = lineBusNames{2};
end
```

4.1.5 Ejecutar comandos

Aparte de la interfaz de circuito, la otra herramienta primaria para interactuar con el servidor COM es la interfaz de texto. Esta interfaz puede ser utilizada para pasar comandos en su propio lenguaje de programación a OpenDSS y que sean ejecutadas directamente en el propio programa. Por ejemplo:

```
DSSText.command = 'Set controlmode=static' ;
DSSText.command = 'Set mode=snapshot number=1 hour=0 h=1 sec=0';
DSSText.command = 'solve';
```

Aquí, la interfaz de texto ha sido usada para resolver el circuito después de configurar un determinado modo de control, el tiempo y el intervalo de tiempo (h). Esta cadena de comandos es compilada en OpenDSS, por lo que que la interfaz de texto puede ser utilizada para cualquier tarea que se pudiese hacer en OpenDSS con sus scripts.

Algo importante sobre las soluciones es que cuando se resuelve el circuito en un bucle, OpenDSS por defecto reinicia la configuración en cada paso. Para evitar esto y que exista continuidad, hay que asignar un valor para el intervalo. Tras ajustar $h = 1$ (h es el paso de tiempo en segundos), se consigue que el software realice las sucesivas soluciones sin reiniciar su contador temporal interno, algo muy útil para simulaciones temporales que será utilizado posteriormente.

4.1.6 Añadiendo/Editando elementos

Uno de los usos más comunes de la interfaz de texto es añadir y editar elementos del circuito. Usando los comandos de OpenDSS *"new"* y *"edit"*, diferentes elementos pueden ser añadidos, movidos y cambiados vía Matlab como se muestra en el siguiente ejemplo:

```
% Comando para asegurarse de que no existen generadores
DSSCircuit.Generators.get;
% Se añade un generador llamado PV
DSSText.command = 'new generator.PV bus1= n292757 phases=3 kv=34.5
kw=500 pf=1 enabled=true';
% Si se vuelve a ejecutar el comando, ahora sí saldrá el generador
DSSCircuit.Generators.get;
% Se selecciona como elemento activo y se comprueba la información de sus buses
DSSCircuit.SetActiveElement('generator.pv');
DSSCircuit.ActiveElement.BusNames
% Ahora se cambia a otro bus y se observa el cambio
DSSText.command = 'edit generator.PV bus1=n1325391 kv=13.2';
DSSCircuit.ActiveElement.BusNames
```

4.1.7 Búsqueda de información en los circuitos

En esta subsección se va a profundizar en cómo obtener información de los diferentes elementos de un circuito desde el servidor COM de Matlab con la función *get* ya introducida anteriormente. Las funciones *get* son herramientas muy útiles para automatizar algunos de los aspectos más tediosos de la interactuación con el servidor COM. Cuando se hace una llamada a esta función, se debe pasar el puntero que se quiere consultar y, opcionalmente, un vector con el nombre de los elementos a consultar. Si no se incluye el nombre de los elementos, serán devueltos por defecto todos los dispositivos habilitados; en cambio, si se incluye el nombre del elemento, se obtendrá información de él aunque este inhabilitado en el circuito.

```
% Llamada sin especificar nombres. Devuelve todos los buses.
Buses = getBusInfo(DSSCircObj);
% Llamada especificando nombres (No olvidar las llaves )
Buses = getBusInfo(DSSCircObj,{'N1311915'});
% Llamada con un vector de nombres
Buses = getBusInfo(DSSCircObj,{'N1311915', 'N312536'});
% Llamada especificando los nombres vía servidor COM
Buses = getBusInfo(DSSCircObj, DSSCircObj.ActiveCircuit.AllBusNames);
```

Las funciones *get* han sido diseñadas para devolver todos los parámetros posibles de cada objeto. Esto presenta una completa lista de propiedades, pero el resultado puede llevar demasiado tiempo para circuitos grandes con muchos elementos. Para aplicaciones donde el usuario tiene que hacer repetitivas llamadas para obtener datos de elementos es recomendable optimizar el proceso de llamada a la función para mejorar la velocidad de respuesta. Esto se puede hacer de dos formas, la primera se acaba de discutir: enviar directamente el nombre de los elementos cuyas propiedades se quiere conocer. La segunda es hacer la llamada

a la función *get* de los elementos necesarios y almacenarla en el entorno de trabajo, entonces sólo habría que consultar la variable creada ahorrando comunicarse repetitivamente con el servidor COM, esto es útil si dichos elementos tienen propiedades constantes en el tiempo, que por lo general suele ser lo habitual en la mayoría de componentes de un circuito. Por ejemplo, la función *getLineInfo* puede ser guardada como *getLineActuales* y consultarla cuando sea necesario, comentando las llamadas al servidor COM. Reducir el número de propiedades consultadas no tiene un impacto significativo cuando se hace una sola llamada, pero sí puede proporcionar ventajas para repetitivas llamadas en una larga simulación temporal.

Es importante comentar que las funciones *get* no devuelven punteros a otros objetos. Se obtienen estructuras que contienen datos *estáticos* correspondientes a la solución más reciente del circuito. Si el circuito se modifica o si hay una nueva solución del circuito, habrá que hacer una nueva llamada a las funciones *get* para actualizar las estructuras con los datos más recientes.

Trabajar con las estructuras de datos desde Matlab

Para hacer un uso más eficiente de las estructuras de datos que se obtienen con la función *get* es necesario recordar ciertos comandos de Matlab. El valor de cada campo en la estructura se accede poniendo el nombre del campo una vez que haya transcurrido al menos un periodo de la simulación. Los nombres de los campos de una estructura dada pueden ser consultadas en Matlab mediante la llamada *fieldnames()*. El resultado de una función *get* es una estructura en forma de matriz. Por ejemplo, en *Buses* están almacenados los parámetros de todos los buses, y cada bus individual es una estructura dentro de *Buses* y sus valores pueden ser accedidos mediante el adecuado índice.

```
% Encontrar todos los nombres en la estructura Buses
fields = fieldnames(Buses);
% Devuelve el nombre del primer bus
Buses(1).name
% Devuelve el número de fases del segundo bus
Buses(2).numPhases
% Devuelve cuántos buses hay en la estructura
length(Buses)
```

Cuando se intenta acceder a datos de múltiples elementos en una estructura hay que asegurarse se incluir en la llamada los paréntesis (o llaves en el caso de vectores) para obtener una matriz de resultados.

```
Cargas = getLoadInfo(DSSCircObj);
% Llamar sin paréntesis devuelve los kW de cada carga de forma separada
Cargas.kW
% Frente a llamar con paréntesis, que devuelve todos los kW en una matriz
[Cargas.kW]
% Lo mismo es válido para vectores
{Loads.name}
```

Este uso de Matlab es muy útil para filtrar resultados en base a ciertos criterios. Por ejemplo, se pueden filtrar las estructuras de datos de sólo las cargas de tres fases o las cargas por debajo de un determinado voltaje.

```
% Filtro para cargas con tres fases únicamente
Cargas_tres_fases = Loads([Loads.numPhases]==3);
% Filtro para cargas de bajo voltaje del sistema secundario
Cargas_secundarias = Cargas([Cargas.kV]<=0.24);
```

4.1.8 Funciones para comprobaciones

Una herramienta particularmente útil al comienzo de cualquier análisis de un circuito es la función *circuitCheck*. Esta función examinará el circuito de OpenDSS por si existen errores tipográficos o inconsistencias potenciales que puedan conducir a una solución un tanto extraña para OpenDSS. Es útil para solucionar errores desde Matlab y para detectar anomalías en el código que están creando errores aparentes aunque permiten que el código se compile. Se recomienda utilizar esta función antes de comenzar a trabajar con cualquier circuito.

La función *circuitCheck* comprueba numerosos errores del modelo del circuito. Un ejemplo puede ser el configurar una carga con un tamaño demasiado elevado, este hecho permitiría compilar el código perfectamente, pero sería un error de modelado que ocasionaría la sobrecarga de un transformador, por ejemplo. La comprobación dará el nombre del transformador sobrecargado para que el código puede ser corregido adecuadamente. Otro ejemplo puede ser tener una línea de un número de fases que no corresponde al número de fases del elemento al que está conectado. Este ejemplo concreto generaría un error en la solución de OpenDSS y no sería compilado el código, pudiendo ser comprobado previamente con la función *getLineInfo* que avisaría del error. Sin embargo, la ventaja de *circuitCheck* es que tiene programadas todo este tipo de comprobaciones de forma automática y devolverá el error, identificando la causa y el nombre del elemento u objeto que debe ser corregido.

Ejecutar la función de comprobación de circuito

Para ejecutar manualmente *circuitCheck*, primero se debe compilar y resolver el circuito. Entonces, incluir el puntero COM para los objetos y, opcionalmente, la opción de advertencias de la función *circuitCheck*. Esta opción de advertencia viene por defecto activada, pero se puede desactivar como se indica a continuación.

```
warnSt = circuitCheck(DSSCircObj,'Warnings', 'off');
warnSt = circuitCheck(DSSCircObj);
```

Si la opción de advertencias está activa, *warnSt.str* será mostrado en la ventana de comandos una vez las comprobaciones hayan sido hechas. Independientemente del ajuste de las advertencias, siempre se podrá acceder a *warnSt.offenders* desde el espacio de trabajo. Se puede abrir la variable *warnSt* desde Matlab haciendo doble click sobre ella y navegar por los errores encontrados en el circuito.

La función *circuitCheck* es llamada automáticamente en cualquiera de las funciones *get* si se detecta un error. Si hay problemas importantes en el circuito, OpenDSS puede no ser capaz de devolver una solución del flujo de potencia válido para el circuito. Entonces, *circuitCheck* comprobará varios errores potenciales que tiene programados, pero hay muchas cuestiones que no podrán ser analizadas.

Interpretar las comprobaciones de resultados

Como ya se ha mencionado, si las advertencias están activadas, cualquier problema con el circuito se mostrarán en la ventana de comandos. Por otro lado, independientemente de si las advertencias se encuentran activas o no, si se ejecuta la función *circuitCheck*, ésta siempre tendrá como salida una estructura con los errores, aunque no se muestre directamente en pantalla. Cada elemento de la estructura corresponde a un solo aviso y contendrá cierta información que describe la advertencia, así como una lista de los elementos que provocan ese error. Después de ejecutar *circuitCheck*, siempre hay que comprobar esta estructura para comenzar el diagnóstico del circuito (si las advertencias están desactivadas, comprobar si la estructura de salida está vacía).
Los límites predeterminados para cada comprobación se pueden cambiar mediante la edición de estos en la parte superior del archivo *circuitCheck.m*. El propósito de *circuitCheck* es identificar posibles problemas, pero no todas las advertencias indican necesariamente que algo está mal en el circuito. El usuario tendrá que inspeccionar la salida para determinar cuáles de las advertencias son en realidad errores en el circuito.
Las advertencias que pueden mostrarse se enumeran a continuación:

- **warnSt.IsolatedElem**
  - **Objetivo:** clasifica a los elementos que no están conectados a la red como *aislados*.
  - warnSt.str: "There are *n* isolated elements, *m* of which are enabled and without voltage."
  - **Explicación:** Esto no tiene por qué ser un error per se, puede haber elementos aislados o desconectados de la red intencionadamente. En ese caso de que sea intencionado, es aconsejable deshabilitarlos. Hay tres categorías de elementos *aislados*:
    1. "Isolated& Enabled without Voltage". Son elementos aislados que no están deshabilitados pero tampoco están conectados a la red.
    2. "Isolated& Disabled". Son elementos que OpenDSS devuelve como aislados a pesar de que estén deshabilitados.

3. "Isolated& Enabled with Voltage". Elementos que tienen tensión, generalmente forman parte de una red "isla", es decir, que no está conectada a la red principal.

– warnSt.offenders: Esta tabla incluye una columna para cada categoría mencionada, además de otra columna que contiene todos los elementos aislados.

- **warnSt.IsolatedNodes**
  - **Objetivo:** Para comprobar nodos aislados de la red.
  - warnSt.str: "There are *n* isolated nodes in the circuit. Investigate the node by using DSSText.command=”show busflow BUSNAME kVA elem”; where BUSNAME is the name of the isolated node without the decimal phasing."
  - **Explicación:** Esto detecta cualquier nodo que puede estar aislado de la red. Un ejemplo de esto puede ser una carga de tres fases conectado a una línea de una sola fase: dos de los nodos (creados por la carga) estarían aislados con un voltaje desconocido.
  - warnSt.offenders: cada fila de la tabla incluye el nombre del nodo aislado.
- **warnSt.Loops**
  - **Objetivo:** Comprobar si existe algún bucle en el circuito.
  - warnSt.str: "There are *n* loops in the circuit. The loops can also be viewed using DSSText.command=”show loops”;"
  - **Explicación:** Generalmente los alimentadores son diseñados de forma radial. Esto devuelve el resultado de un algoritmo de OpenDSS para detectar bucles.
  - warnSt.offenders: Cada fila de la tabla incluye la lista de elementos que forman parte de un bucle.
- **warnSt.InvalidLineBusName**
  - **Objetivo:** Comprueba que la nomenclatura de un bus para líneas específicas coincide con las fases de esa línea.
  - warnSt.str: "One or more line has a bus name that does not match the number of phases of the line. (e.g. A 2-phase line should have both bus 1 and bus 2 with names similar to ’BUSNAME.2.3’with 2 phases indicated in the decimal notation."
  - **Explicación:** La herramienta usa esta nomeclatura para ayudar a determinar las fases que tiene una determinada línea. El número de fases en esa línea debe coincidir con el número de fases del bus al que está conectada.
  - warnSt.offenders: Cada fila de la tabla incluye el nombre de la línea (*LineName*), el número de fases (*NumPhases*) y el nombre de cada bus. A partir de esto, debe ser obvio qué parte de la definición de las líneas está causando los errores.
- **warnSt.NoBusCoords**
  - **Objetivo:** Comprueba la presencia de las coordenadas del bus.
  - warnSt.str: "There are no bus coordinates with this compiled circuit. Toolbox functionality wil be severaly limited."
  - **Explicación:** La herramienta usa las coordenadas de los buses para realizar las gráficas del circuito así como la integración solar para algunas simulaciones.
  - warnSt.offenders: n/a.
- **warnSt.MissingBusCoords**
  - **Objetivo:** Comprueba para buses de medio voltaje (por debajo de 600V) que han perdido coordenadas.
  - warnSt.str: "There are *n* buses above 600V that are missing coordinates."

  - **Explicación:** En general, cualquier objeto que está conectado al bus sin coordenadas, no será mostrado en las gráficas.
  - warnSt.offenders: Lista de buses sin coordenadas.

- **warnSt.LineLength**
  - **Objetivo:** Comprueba si hay líneas con largas longitudes sin sentido en el circuito.
  - **Límites:** 5 km.
  - warnSt.str: "*n* of the lines exceed 5 km"
  - **Explicación:** Introducir accidentalmente largas longitudes puede generar resultados erróneos e irregularidades en las simulaciones de flujo de potencia.
  - warnSt.offenders: Nombre de la línea y longitud.

- **warnSt.LineOverLoading**
  - **Objetivo:** Comprueba si hay violaciones térmicas en las líneas.
  - **Límites:** 100 % .
  - warnSt.str: "*n* Lines are load more than 100 % . Visualize using *plotCircuitLines(DSSCircObj,'Coloring','lineLoading')* "
  - **Explicación:** Advierte que una línea está sobrecargada que puede ser consecuencia de una mala configuración de los parámetros en el circuito o en las líneas.
  - warnSt.offenders: Nombre de la línea y su porcentaje de carga.

- **warnSt.BusDistance**
  - **Objetivo:** Comprueba si se han introducido incorrectamente líneas que están excesivamente lejos de las líneas.
  - **Límites:** 25 km.
  - warnSt.str: "*n* of the bus distances exceeds 25 km from the substation. "
  - **Explicación:** Introducir parámetros del circuito de forma errónea accidentalmente, así como la longitud de la línea, puede llevar a que un bus esté excesivamente lejos de una subestación.
  - warnSt.offenders: Nombre de los buses y distancia.

- **warnSt.CapacitorRatingMismatch**
  - **Objetivo:** Comprueba si se han introducido accidentalmente datos de kV poco consistentes.
  - **Límites:** 5 %.
  - warnSt.str: "*n* of the capacitor kV ratings differs from its bus kV rating by more than 5 %. "
  - **Explicación:** Introducir incorrectamente datos de tensiones pueden causar irregularidades en la solución sin dar un error en la simulación. Lo más probable es que este problema se deba a que, en una fase, no se introdujo correctamente la tensión de línea a neutro; o para dos o tres fases, no se introdujo el valor línea a línea.
  - warnSt.offenders: Nombre de cada elemento y su tensión línea a línea, así como el nombre de todos los buses y su tensión línea a línea.

- **warnSt.LoadRatingMismatch**
  - **Objetivo:** Comprueba si se han introducido accidentalmente datos de kV poco consistentes.
  - **Límites:** 5 %.
  - warnSt.str: "*n* of the load kV ratings differs from its bus kV rating by more than 5 %. "

  - **Explicación:** Introducir incorrectamente datos de tensiones pueden causar irregularidades en la solución sin dar un error en la simulación. Lo más probable es que este problema se deba a que, en una fase, no se introdujo correctamente la tensión de línea a neutro; o para dos o tres fases, no se introdujo el valor línea a línea.
  - warnSt.offenders: Nombre de cada elemento y su tensión línea a línea, así como el nombre de todos los buses y su tensión línea a línea.

- **warnSt.GeneratorRatingMismatch**
  - **Objetivo:** Comprueba si se han introducido accidentalmente datos de kV poco consistentes.
  - **Límites:** 5 %.
  - warnSt.str: "*n* of the generator kV ratings differs from its bus kV rating by more than 5 %. "
  - **Explicación:** Introducir incorrectamente datos de tensiones pueden causar irregularidades en la solución sin dar un error en la simulación. Lo más probable es que este problema se deba a que, en una fase, no se introdujo correctamente la tensión de línea a neutro; o para dos o tres fases, no se introdujo el valor línea a línea.
  - warnSt.offenders: Nombre de cada elemento y su tensión línea a línea, así como el nombre de todos los buses y su tensión línea a línea.

- **warnSt.PVRatingMismatch**
  - **Objetivo:** Comprueba si se han introducido accidentalmente datos de kV poco consistentes.
  - **Límites:** 5 %.
  - warnSt.str: "*n* of the PV kV ratings differs from its bus kV rating by more than 5 %. "
  - **Explicación:** Introducir incorrectamente datos de tensiones pueden causar irregularidades en la solución sin dar un error en la simulación. Lo más probable es que este problema se deba a que, en una fase, no se introdujo correctamente la tensión de línea a neutro; o para dos o tres fases, no se introdujo el valor línea a línea.
  - warnSt.offenders: Nombre de cada elemento y su tensión línea a línea, así como el nombre de todos los buses y su tensión línea a línea.

- **warnSt.TransformerRatingMismatch**
  - **Objetivo:** Comprueba si se han introducido accidentalmente datos de kV poco consistentes.
  - **Límites:** 5 %.
  - warnSt.str: "*n* of the transformer kV ratings differs from its bus kV rating by more than 5 %. "
  - **Explicación:** Introducir incorrectamente datos de tensiones pueden causar irregularidades en la solución sin dar un error en la simulación. Lo más probable es que este problema se deba a que, en una fase, no se introdujo correctamente la tensión de línea a neutro; o para dos o tres fases, no se introdujo el valor línea a línea.
  - warnSt.offenders: Nombre de cada elemento y su tensión línea a línea, así como el nombre de todos los buses y su tensión línea a línea.

- **warnSt.TransformerOverloaded**
  - **Objetivo:** Comprueba si hay violaciones térmicas en las líneas.
  - **Límites:** 5 % .
  - warnSt.str: "*n* of the transformer kVA ratings differs from its bus1 power by more than 5 %. Check that the loads on the transformer are entered correctly."
  - **Explicación:** Advierte que un transformador está sobrecargado, pudiendo ser consecuencia de una mala configuración de los parámetros en el circuito.

  - warnSt.offenders: Nombre del transformador y su porcentaje de carga.

- **warnSt.TransformerNoLoad**
  - **Objetivo:** Comprueba si hay transformadores con ninguna carga aguas abajo.
  - **Límites:** flujos de potencia en el transformador inferiores al 1 % del ratio del transformador y ninguna carga aguas abajo.
  - warnSt.str: "*n* of the transformer have no load on them. Check that the loads on that transformer are entered correctly."
  - **Explicación:** Detecta cualquier problema durante la asignación de cargas cuando cargas no están asignadas a un transformador en servicio.
  - warnSt.offenders: Nombre del transformador y sus ratios de kVA.

- **warnSt.TransformerLowLoad**
  - **Objetivo:** Comprueba si hay transformadores con bajo flujo de potencia en comparación a su ratio de funcionamiento.
  - **Límites:** flujos de potencia en el transformador inferiores al 1 % del ratio del transformador y ninguna carga aguas abajo.
  - warnSt.str: "*n* of the transformer have less than 1 % power flow of their kVA rating. Check that the loads on that transformer are entered correctly."
  - **Explicación:** Detecta cualquier problema durante la asignación de cargas cuando cargas no están asignadas a un transformador en servicio.
  - warnSt.offenders: Nombre del transformador, sus ratios de kVA, suma de kW para todas las cargas instaladas aguas abajo y la lista de cargas aguas abajo.

- **warnSt.BusVoltage**
  - **Objetivo:** Comprueba si hay violación de tensiones (sobrevoltaje o déficit de voltaje).
  - **Límites:** 1 ± 0.05 pu.
  - warnSt.str: "*n* of the enabled bus voltages are outside of the range 1 ± 0.05 pu. Visualize using *plotVoltageProfile(DSSCircObj)*"
  - **Explicación:** Advierte sobre violaciones de voltaje que pueden ser causa de una incorrecta configuración de parámetros y pueden provocar grandes cambios de tensión.
  - warnSt.offenders: Nombre del bus y tensión (pu y kV).

- **warnSt.LineRatingMismatch**
  - **Objetivo:** Comprueba si se han introducido accidentalmente datos incorrectos en los códigos de línea.
  - **Límites:** 150 %.
  - warnSt.str: "*n* of the line ratings are 150 % the size of the inmediately upstream line. Visualize using *plotCircuitLines(DSSCircObj,'Thickness','lineRating')*"
  - **Explicación:** Los ratios de las líneas que se incrementan aguas abajo *pueden* ser indicativo de incorrectos parámetros de línea (o puede ser por el diseño).
  - warnSt.offenders: Nombre de la línea aguas arriba (la más pequeña) y el nombre de la línea aguas abajo (más grande), seguida por ratio de cada línea y su código de línea.

4.1.9 Creación de gráficas en OpenDSS desde Matlab

La generación de gráficas es relativamente sencilla y se puede encontrar documentación en los manuales de OpenDSS ([12]) y algunos ejemplos en la red ([8]). Las gráficas son útiles porque son representativas de la solución del flujo de potencia en la simulación más reciente. Es decir, que en caso de querer conocer la solución gráfica en un determinado instante de tiempo, habrá que configurar la simulación para que ese sea el último.

```
DSSText.command = 'Set mode=duty number=10 hour=13 h=1 sec=1800';
DSSText.command = 'Set controlmode = static ';
DSSText.command = 'solve';
figure ; plotCircuitLines (DSSCircObj);
```

Llamando a *plotCircuitLines* sin asignar ninguna propiedad, será llamada por defecto la interfaz gráfica de OpenDSS (GUI) para lanzar la función *plotCircuitLinesOptions*. Esta función está asociada a OpenDSS GUI y también puede ser llamada de la misma manera, sin embargo, no acepta ningún otro parámetro.

Las gráficas obtenidas serán generadas con OpenDSS, pero añadiendo el comando de Matlab *figure* antes de hacer la llamada a *plotCircuitLines*, se abrirá en el entorno gráfico de Matlab.

```
figure ; plotCircuitLinesOptions (DSSCircObj);
```

En cambio, sí es posible llamar a la función *plotCircuitLines* con un determinado número de parámetros:

```
figure ; plotCircuitLines (DSSCircObj,'Coloring','PerPhase','Thickness',3,'MappingBackground','hybrid');
```

Las funciones usadas para la creación de gráficas utilizan el nombre de los parámetros de Matlab; y los argumentos que se le dan, después de la llamada a *DSSCircObj*, van emparejados. Esto es, aunque el orden de las especificaciones que se le da a la función no importa, cada opción requiere dos entradas: el nombre de la configuración que se está modificando y la especificación de este cambio. Por ejemplo, en la línea de código anterior, el parámetro *'Coloring'* es configurado a *'PerPhase'* y la propiedad *'Thickness'* es igual a 3.

Interacción con el circuito

Usando cualquier herramienta para la creación de gráficas, el usuario tiene disponibles ciertas opciones de interacción que hacen el análisis gráfico de flujos de potencia en OpenDSS mucho más sencillo. Cualquier línea, transformador, capacitador, carga o sistema de paneles solares puede ser seleccionado dentro de la gráfica con el click izquierdo o derecho.

Por defecto, el click izquierdo selecciona el elemento, mostrando el nombre como se indica en la figura 4.3. También hay un botón de activación de la visata de nodos (*"Node View"*) en la barra de herramientas, que permite activar la opción de mostrar el nombre del bus del elemento cuando se hace click sobre él.

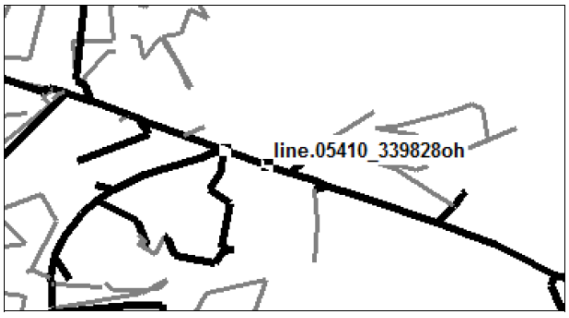


Figura 4.3 Seleccionar un elemento con el click izquierdo.

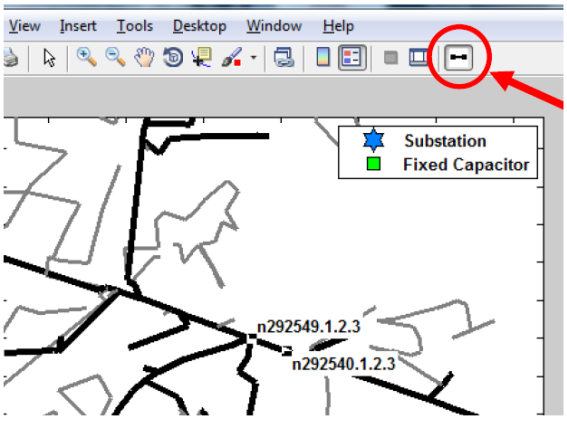


Figura 4.4 Seleccionar un elemento con el click izquierdo y activar vista de los nodos.

El click derecho abre el menú que se ve en la figura 4.5, que permite mostrar las propiedades, voltajes, corrientes y potencias para ese elemento. Este menú sólo está disponible si se hace click directamente sobre el elemento, por ello, a menudo es más fácil hacer clic derecho en un elemento del circuito que no está aún seleccionado.

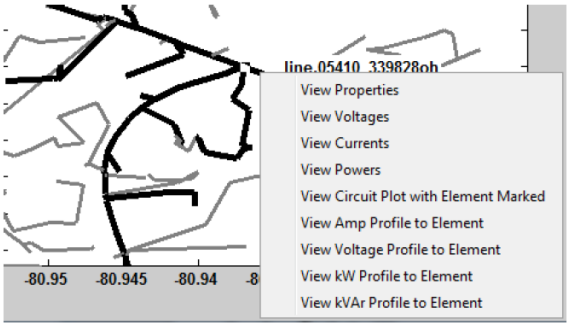


Figura 4.5 Seleccionar un elemento con el click derecho.

Seleccionar cualquier opción del menú que se abre con el click derecho, abrirá una ventana de OpenDSS con la información que se está solicitando. Estas ventanas (propiedades, tensiones, corrientes y potencias) son ventanas emergentes de OpenDSS, por lo que la opción que permite su despliegue debe estar activada en el software. Es decir, la propiedad *DSSCircObj.AllowForms* debe ser 1, que es el valor por defecto.

Edición de gráficas

Después de realizar la gráfica es posible editarlas. Los usuarios más experimentados en Matlab y sus opciones gráficas, tienden a usar el botón *"show plot tools"* mostrado en la figura 4.6. Por defecto, esto abrirá la "vista del explorador de gráficas". Pero esto generalmente está desaconsejado. La razón es que las gráficas generadas por la herramienta normalmente contienen un número muy elevado de líneas en la figura, y abrir la opción *"show plot tools"* puede ocasionar que Matlab se congele y sea necesario reiniciar el programa. Así que, a menos que el circuito sea pequeño (menos de 150 nodos), es desaconsejable utilizar esta opción.

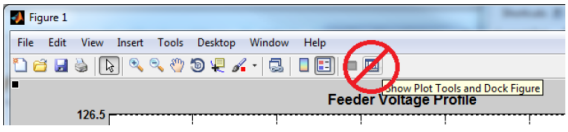


Figura 4.6 Evitar usar "Show Plot Tools".

Por tanto, la mejor forma de editar las gráficas es usando la vista del "Editor de propiedades" (*Property Editor*) que se muestra en la figura 4.7.

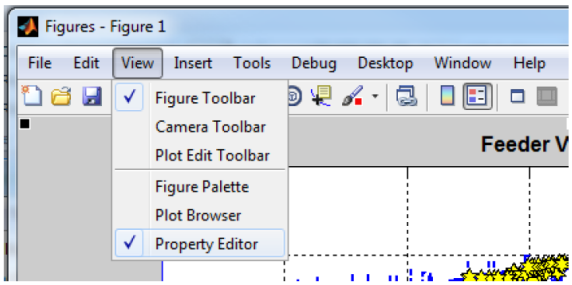


Figura 4.7 Usar "Property Editor" para modificar gráficas.

Después de seleccionar esta vista, se podrá seleccionar varios objetos de la gráfica para editar. El modo "Editor de Propiedades" puede ser utilizado para editar objetos en la gráfica (colores de línea y espesores), ejes y marcas.

Para volver a la vista estándar, sólo hay que seleccionar el botón *"Hide Plot Tools"* que se muestra en la figura 4.8.

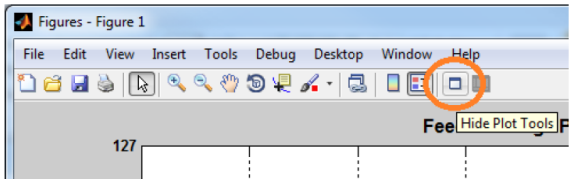


Figura 4.8 Volver a la vista por defecto.

Identificación de las gráficas

Cada función para crear gráficas devuelve una identificación para los objetos representados en el circuito, esto permite que cualquiera de ellos pueda ser modificado una vez se haya realizado la gráfica. Por ejemplo, los marcadores del capacitador pueden ser cambiados a círculos rojos haciendo:

```
figure; Handles = plotCircuitLines (DSSCircObj);
set(Handles. fixedCapacitor ,'MarkerFaceColor','r','MarkerSize',12,'Marke
r','o'); % Cambia la representación de los capacitadores a círculos rojos
```

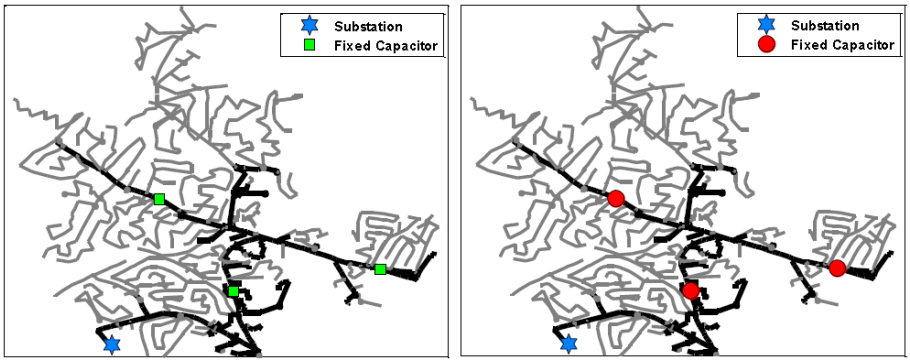


Figura 4.9 Gráfica por defecto y después de modificar la marca de los capacitadores con la estructura de identificación ("handles").

Como se ve, cualquier modificación de este tipo puede ser hecha después de haber realizado la gráfica. Una de las necesidades más comunes es representar gráficamente los buses que se han añadido a un determinado proyecto. En el ejemplo de abajo, se han añadido dos buses, pero el script podría ser usado para marcar cualquier número de buses. Se puede utilizar cualquier tipo de línea o marca en estas modificaciones.

```
figure; Handles = plotCircuitLines (DSSCircObj);
addBuses = [{'N1311915'}, {'N284022'}];
Buses = getBusInfo(DSSCircObj,addBuses,1);
BusesCoords = reshape([Buses. coordinates ],2,[]) ';
busHandle = plot(repmat(BusesCoords(:,2) ',2,1) ,repmat(BusesCoords(:,1) ',2,1) , 'ko','MarkerSize',10,'
    MarkerFaceColor','c','LineStyle','none','DisplayName',addBuses);
legend([Handles.legendHandles,busHandle'],[Handles.legendText,addBuses])
```

4.1.10 Ejemplo

Por último se incluye un ejemplo de cómo se podría compilar y ejecutar desde Matlab el circuito utilizado en el capítulo 3.

Código 4.1 Código de Matlab para iniciar el COM Server.

```
clc; clear all; close all;
%% ARRANQUE DE OPENDSS. Activación del COM Server y comunicación con ActiveX de Matlab.
% Se inicia el COM Server de OpenDSS. Matlab utiliza su servidor ActiveX
% integrado para comunicarse con el COM Server de OpenDSS, de esta manera
% dicho servidor será la interfaz entre ambos programas.
    DSSObj = actxserver('OpenDSSEngine.DSS');

% Se inicia la DSS. (Sólo necesario la primera vez que se inicia Matlab).
    DSSObj.Start(0) % Devolverá un 1 si OpenDSS está disponible y funcionando.

    DSSText = DSSObj.Text; % Se define la interfaz de texto. Con este comando se introducirán las instrucciones a
        OpenDSS desde Matlab
    DSSCircuit = DSSObj.ActiveCircuit; % Se define la interfaz del circuito.
    DSSSolution = DSSCircuit.Solution; % Se define la interfaz de la solución.

%% Compilación del script del circuito Ieee37 bus desde el directorio donde esté guardado
DSSText.Command = 'Compile "C:\example\Ieee37\ieee37.dss"';

%% Algunas instrucciones para OpenDSS
DSSText.Command = 'Show Losses' % Genera un archivo con las pérdidas de potencia
DSSText.Command = 'Show Powers' % Genera un archivo con la potencia de sobrecarga de cada elemento del circuito.
DSS.TextCommand = 'Show Voltages LN' % Comando que genera un archivo que contiene las tensiones simples de cada bus
    en el sistema

% Exportar voltajes a un archivo de excel
 DSSText.Command = 'Export Voltages';
 Filename = DSSText.Result;


%% Solución de circuito
DSSSolution.Solve();
if DSSSolution.Converged,
    disp('El circuito ha sido resuelto satisfactoriamente')
end
```

5 Modelado y simulación de una planta experimental doméstica basada en hidrógeno renovable

En este capítulo se pretende hacer un modelado simplificado, simulación e implementación de algoritmos de control mediante las herramientas OpenDSS y Matlab de una planta experimental equipada con una pila de combustible y un electrolizador. Este modelado está basado en el documento [18], donde se propone una solución para redes domésticas compuestas por energía fotovoltaica, almacenamiento de hidrógeno en hidruros metálicos con un electrolizador, pila de combustible de electrolito polimérico (PEM-FC) y la interconexión con redes vecinas y vehículos eléctricos. Este modelo ha sido validado mediante los resultados de una planta experimental diseñada, construida y operada por los autores del citado documento (Ver Figura 5.1). Dicha planta se encuentra en las instalaciones de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla (ETSI). Las principales características del equipamiento según sus desarrolladores se encuentra en la Figura 5.3, extraída del documento de referencia.

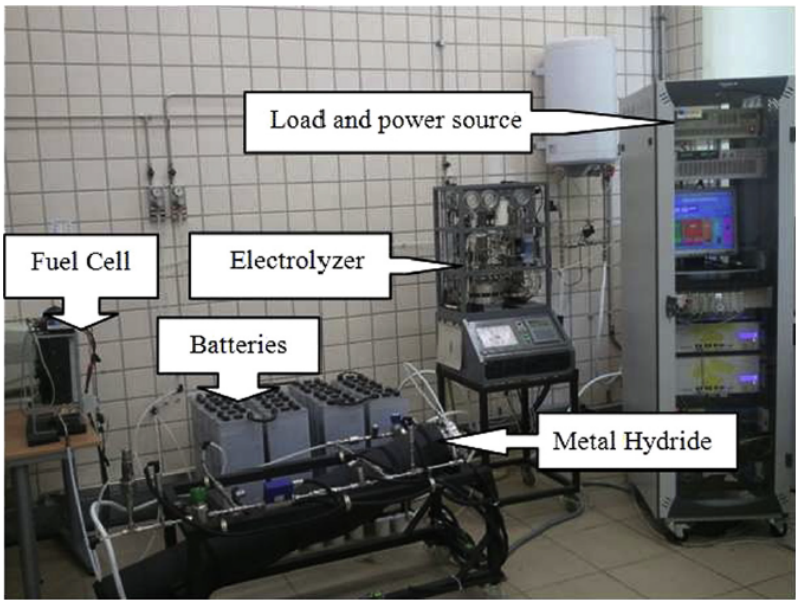


Figura 5.1 Vista del montaje experimental.

La motivación de utilizar pilas de combustible es la capacidad para almacenar energía a largo plazo, pudiendo convertir la energía eléctrica en hidrógeno mediante un proceso de electrólisis; posteriormente, el hidrógeno se convierte de nuevo en electricidad cuando la demanda de energía así lo solicite, proporcionando con ello una opción de almacenamiento de energía a largo plazo y fiable.

5.1 Introducción al modelo de la Smart Grid

Los elementos que se pretenden modelar para simular el modelo experimental son:

a) Paneles solares
b) Baterías.
c) Electrolizador
d) Pila de combustible
e) Transformadores AC/DC y DC/DC
f) Cargas
e) Coche eléctrico

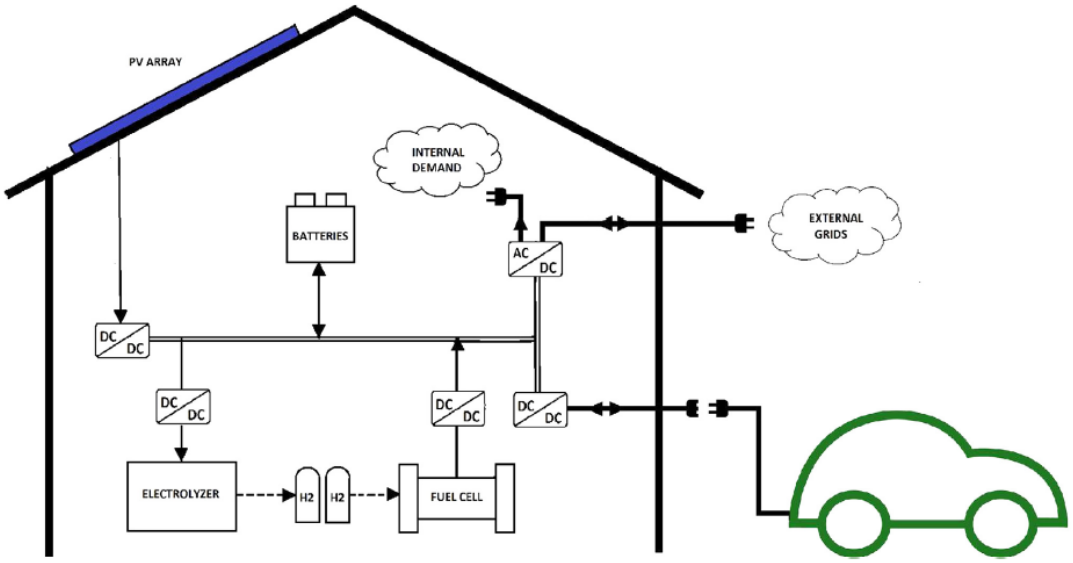


Figura 5.2 Red doméstica basada en hidrógeno.

Se puede ver un esquema de la red bajo estudio en la Figura 5.2. El panel fotovoltaico (*PV Array*) será la fuente principal de suministro de energía del sistema, pero para evitar interrupciones en el suministro causadas por la intermitencia de los recursos solares, un electrolizador estará conectado a la línea principal de distribución. En caso de exceso de potencia, la electricidad será usada para producir y almacenar hidrógeno. En caso contrario, la pila de combustible generará electricidad utilizando el hidrógeno almacenado cuando sea requerido. Un pack de baterías también forma parte del sistema, conectándose a la red principal para mantener un voltaje constante en la red, así se simplifica el diseño del convertidor. En este sentido, el diseño también contempla la incorporación de cinco convertidores: uno DC/AC para conectar las aplicaciones domésticas y las redes aledañas, otro bidireccional DC/DC para proporcionar y obtener energía de un vehículo eléctrico, y tres convertidores DC/DC unidireccionales para adaptar la potencia del conjunto fotovoltaico, la energía del electrolizador y los requisitos de energía de células de combustible, respectivamente. El coche eléctrico se modelará como una carga que consume energía del sistema.

En ese montaje experimental, la planta solar es emulada por un fuente de potencia electrónica programable, mientras que en el presente trabajo se modelará como se describió en el Capítulo 3, es decir, con el modelo PV que el software OpenDSS tiene implementado.
La instalación de hidrógeno experimental consiste en un electrolizador PEM de 3.2 kW de pico de potencia, un tanque de almacenamiento de hidruro de metal y una pila de combustible PEM. El hidrógeno es almacenado en un tanque de aleación de hidruro de metal $LaNi_5$ con una capacidad de 7 Nm^3. Además dispone de un sistema de enfriadores/calentadores, ya que es necesario calor para liberar el hidrógeno almacenado. Finalmente, una pila de combustible PEM de 1.2 kW completa la instalación.

Component	Rated capacity	Manufacturer
Electronic power source	6 kW	POWERBOX
Electronic load	2.5 kW	AMREL
PEM electrolyzer	0.23 Nm^3/h	HAMILTON-STD
Metal hydride storage	7 Nm^3, 5 barg	LABTECH
PEM fuel cell	20 Nl/min @ 1.5 kW	MES-DEA
Pb-acid battery pack	$C_{120,bt}$ 367 Ah	EXIDE
Water purification	3 l/h @ 15 MΩ	MILLIPORE
PLC	M340-Canbus	SCHNEIDER
DC/DC converters	1.5 kW, 1 kW	WINDINERTIA

Figura 5.3 Características del equipamiento experimental.

Todos los dispositivos electrónicos del montaje están conectados a un bus de 48 V de corriente continua. Las baterías tienen una capacidad de $C_{120,bt} = 367$ Ah.

5.2 Modelado de la red en OpenDSS

Se va a proceder a hacer un modelado básico del montaje experimental en el software OpenDSS. Para ello, lo primero será diseñar un esquema unifilar basado en el esquema de configuración del documento de referencia [18]. (Ver Figura 5.4).

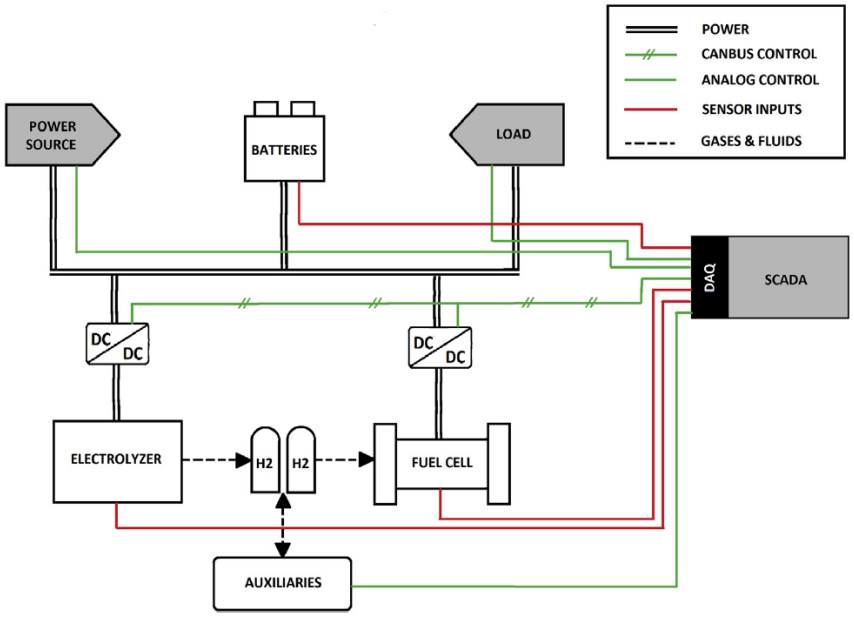


Figura 5.4 Esquema de configuración de la planta experimental.

Para realizar el esquema unifilar hay que definir los nodos/buses del circuito (en OpenDSS se pueden definir los buses como nodos, aunque un bus a su vez puede albergar varios nodos) y las coordenadas de dichos buses, si se desea, para utiliza la herramienta *Plot*, aunque esto no es necesario para la simulación en OpenDSS y sólo tiene sentido gráfico para interpretación de resultados.

Una cuestión importante sobre OpenDSS, es que es un software que sólo puede resolver una frecuencia de funcionamiento a la vez. Esto es, que en caso de que se quiera trabajar en sistemas de múltiples frecuencias (por ejemplo circuitos AC/DC), será necesario primero resolver uno y luego el otro. La solución del primer circuito, se utilizará como condición inicial para el segundo; después de la primera iteración, el segundo

sistema enviará las condiciones iniciales para el primero. Esto es una forma de combinar circuitos de diferentes frecuencias. Es importante tener esto en cuenta porque el circuito que se pretende aquí simular trabaja a corriente continua de 48 V, mientras que hay sistemas conectados en AC. Para simular circuitos de corriente continua en OpenDSS es necesario definir la frecuencia base de funcionamiento en 0 Hz, o en su defecto, a un valor muy bajo para evitar errores numéricos, por ejemplo, configurando la frecuencia a 0.001 Hz. Esto se puede realizar con el siguiente comando: $set \quad defaultbasefreq = 0.001$

Por otro lado, los elementos por defecto de OpenDSS son de 3 fases, en este caso, se va a considerar que sólo hay una fase (más la adicional a línea de tierra que no es necesario definir), por lo que será importante indicar el número de fases en cada elemento que sea modelado.

5.2.1 Modelo de dispositivos

Generador fotovoltaico

El modelado del generador fotovoltaico en OpenDSS se trató en la subsección 3.2.1. En esta sección se hará un modelo similar. Habrá que definir los perfiles de carga, la irradiancia solar, la temperatura de operación del panel fotovoltaico y la potencia del generador en función del modelo. La planta experimental bajo estudio, utiliza un modelo basado en 20 paneles fotovoltaicos *ISF-150* fabricados por ISOFOTON. Las características básicas de este modelo se observan en la siguiente tabla:

Tabla 5.1 Características eléctricas ISOFOTON 150 para irradiancia 1000 W/m^2. Datos de web del fabricante.

Parámetro	**Valor**
Potencia nominal	150 W
Tensión en circuito abierto	22.6 V
Corriente de cortocircuito Isc	8.70 A
Tensión en el punto de máxima potencia (Vmax)	18.5 V
Corriente en el punto de máxima potencia Imax	8.12 A
Eficiencia	15.0 %
Tolerancia de potencia (%Pmax)	0/+3 %

Tabla 5.2 Características de operación ISOFOTON 150.

Parámetro	**Valor**
Tensión máxima del sistema	1000 V
Límite de carga inversa	20 A
Temperatura de Operación Nominal de la Célula (TONC)	45 $\pm$2ºC
Temperatura de operación	-40 a +85ºC
Coeficiente de temperatura a Pmax	-0.464 %/K
Coeficiente de temperatura de Voc	-0.323 %/K
Coeficiente de temperatura de Isc	0.042 %/K

Para la curva de irradancia se tomarán los mismos valores que los utilizados en la planta experimental, interpolándose para obtener el número de puntos deseados (Ver Figura 5.5). En este caso se utilizarán las funciones *polyfit* y *polyval* de Matlab para obtener 24 puntos de esta curva para utilizarla como *LoadShape*. Teniendo en cuenta que estos datos de irradancia contemplan los valores desde las 4:45:00 GMT hasta las 20:10:34 GMT, por lo que habrá que incluir los puntos de las horas nocturnas, donde la irradancia tiene un valor igual a cero.

Por otro lado, debido a que la tensión de los paneles fotovoltaicos varía entre 18.5 y 22.6 VDC, será necesario incorporar un inversor. La eficiencia del inversor depende mucho del salto de tensión que tenga que realizar la electrónica. Se dispone de la curva de eficiencia real de los convertidores DC/DC experimentales, que será la misma que se tome para el inversor pero adaptando la eficiencia máxima, ya que las curvas de estos equipos se parecen mucho. Los inversores normalmente pueden tener más eficiencia que los convertidores DC/DC. Por ejemplo, los convertidores instalados en la planta experimental, como mucho tienen una eficiencia del 90 %, mientras que el inversor (que en el caso experimental es emulado mediante la fuente de

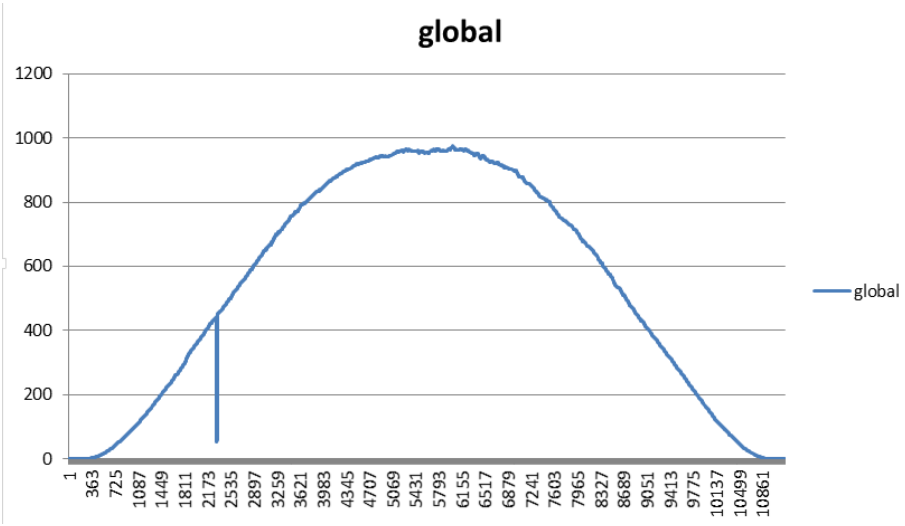


Figura 5.5 Curva de irradancia utilizada en el modelo experimental.

potencia) puede alcanzar 95 %. En el presente proyecto no se incorporarán convertidores DC/DC debido a que la herramienta OpenDSS aún no contempla este tipo de elementos (sí AC/DC o AC/AC), por lo que se considerará la curva de eficiencia experimental previamente mencionada.

Electrolizador

Un electrolizador utiliza el principio de la electrólisis para separar el hidrógeno y el oxígeno del agua aplicando una alta corriente a través de dos electrodos separados por un electrolito. La reacción electroquímica puede expresarse con la siguiente expresión:

$$2H_2O \quad (\text{líquida}) + \text{energía eléctrica} \longrightarrow H_2 \quad (\text{gas}) \quad + \frac{1}{2}O_2 \quad (\text{gas}) \tag{5.1}$$

Desde el punto de vista eléctrico, el electrolizador es un consumidor de potencia, por lo que en una primera aproximación, este dispositivo se modelará como una carga variable en función de una curva de funcionamiento basada en los modelos matemáticos del documento de referencia.

La generación de hidrógeno es directamente proporcional a la corriente del dispositivo. Ésta es la variable que se controla para gestionar la potencia que consume. La tensión viene dada por su curva de funcionamiento.

El electrolizador actual implementando en la planta experimental tiene una potencia pico de 3.2 kW. Este dispositivo trabaja a 32 VDC, es decir, que se necesita un convertidor DC/DC para conectarse al bus principal de 48 VDC. Se tendrán en cuenta tanto el rendimiento del electrolizador como del mencionado convertidor utilizando las curvas de rendimiento experimentales.

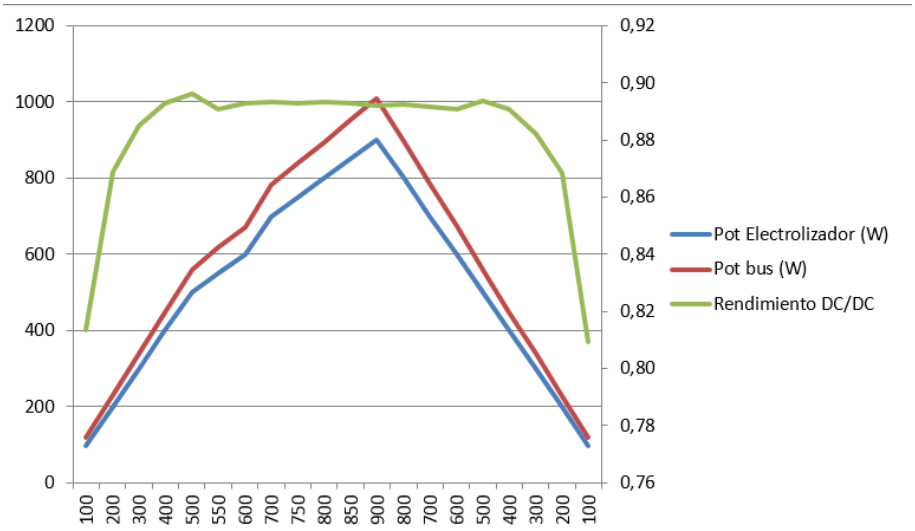


Figura 5.6 Potencias del electrolizador y convertidor DC/DC experimental .

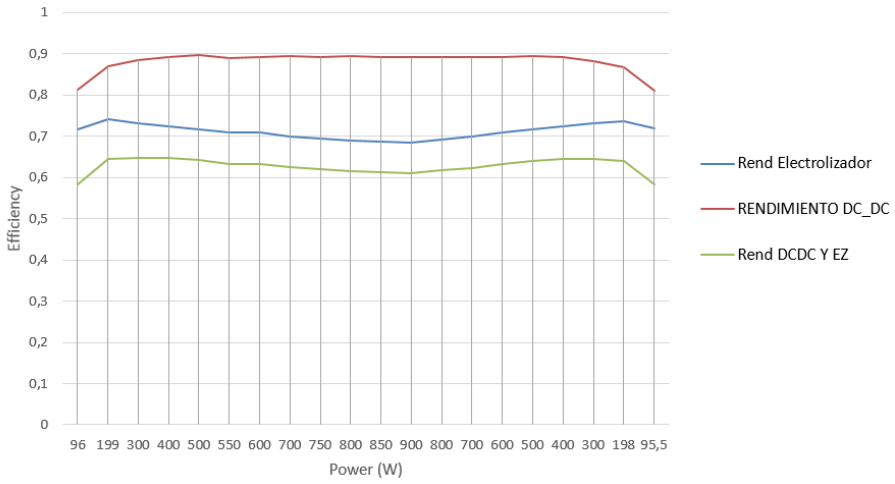


Figura 5.7 Rendimientos sistema electrolizador-convertidor DC/DC.

Obteniéndose la curva de funcionamiento del conjunto electrolizador-convertidor DC/DC.

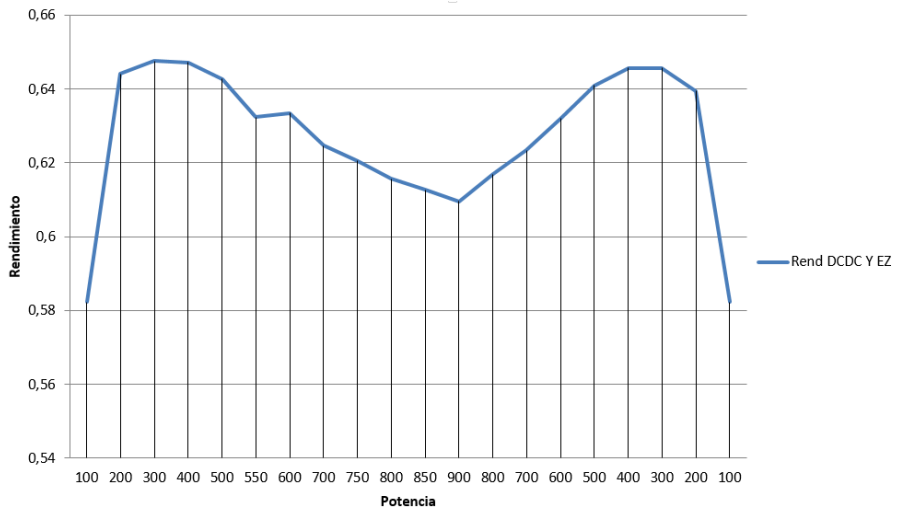


Figura 5.8 Curva experimental funcionamiento conjunto electrolizador-convertidor DC/DC.

El electrolizador tiene dos modos de funcionamiento: uno a carga parcial, donde el electrolizador (EZ) absorbe la diferencia entre la potencia generada y demandada. Otro modo a carga nominal. En éste caso el EZ funciona a potencia constante. Con la potencia que se deriva al equipo y su curva de funcionamiento V-I, se obtiene la intensidad en cada momento y por tanto, el hidrógeno producido.

Dispone de dos stacks de 30 celdas cada uno. Este dato será utilizado para calcular la cantidad de hidrógeno generado.

Pila de combustible PEM

El modelo inicial para la pila de combustible será el de un generador con una curva de funcionamiento. La potencia máxima es $P_{mx} = 1.2kW$.

La pila tiene también dos modos de funcionamiento (a P constante o variable). De nuevo, con su curva V-I se puede obtener el consumo de hidrógeno.

Pack de baterías

Las baterías se pueden modelar en OpenDSS como se se mostró en la subsección 3.2.5, adaptando las características del modelo a las propiedades del *storage element* de OpenDSS. El modelo dinámico que se va a seguir para la carga de la batería es el proporcionado en el documento ya citado [18]. El modelo utiliza una resistencia interna, R_i, y un control en tensión para mitigar el comportamiento exponencial de la batería:

$$V_{bt} = V_{bt,int} + R_i I_{bt} \tag{5.2}$$

donde V_{bt} será la tensión de la batería, $V_{bt,int}$ la tensión interna de la batería y I_{bt} la intensidad.

- Proceso de carga. Tensión interna:

$$V_{bt,int} = V_{bt,0} - K_{bt}\frac{C_{120,bt}}{C_{120,bt} - C_{out,t}} I^*_{bt} - K_{bt}\frac{C_{120,bt}}{C_{120,bt} - C_{out,t}} C_{out,t} + A_{bt} e^{B_{bt} C_{out,t}} \tag{5.3}$$

- Proceso de descarga. Tensión interna:

$$V_{bt,int} = V_{bt,0} - K_{bt}\frac{C_{120,bt}}{Cout,t + 0.1C_{120,bt}} + I^*_{bt} - K_{bt}\frac{C_{120,bt}}{C_{120,bt} - C_{out,t}} C_{out,t} + A_{bt} e^{B_{bt} C_{out,t}} \tag{5.4}$$

donde $V_{bt,0}$ es una tensión constante, I^*_{bt} es la intensidad de la batería filtrada por un filtro de primer orden, A_{bt} es la amplitud de la zona exponencial, B_{bt} representa la constante de tiempo de la zona exponencial, K_{bt} es la constante de polarización, $C_{out,t}$ es la capacidad extraída en Ah y $C_{120,bt}$ es la capacidad máxima de la

batería. Los parámetros son experimentales y se recogen en la Tabla 5.3. El tiempo de respuesta de la batería ha sido estimado en 31 s.

Tabla 5.3 Parámetros del modelo de batería.

Parámetro	Valor
$C_{120,bt}$	367 Ah
$V_{bt,0}$	51.58 V
K_{bt}	0.006215 V
A_{bt}	11.053 V
B_{bt}	2.452 Ah^{-1}

No hay una forma sencilla o directa de obtener el estado de carga de la batería en una simulación temporal, por lo que se recurre al siguiente código en Matlab:

Código 5.1 Obtención de intensidades y almacenamiento en Matlab .

```
%% BATERÍAS
        % Obtención de energía de las baterías.
        DSSCircuit.SetActiveElement('Storage.Battery');
        tempvoltage=DSSCircuit.ActiveElement.Powers;
        tempvoltage = reshape(tempvoltage,2,[]);
        tempvoltage = hypot(tempvoltage(1,:), tempvoltage(2,:));
        StoragePowerMag(i) = sum(tempvoltage(1));

        % Lectura del valor de la energía almacenada en las baterías
        DSSCircuit.SetActiveElement('Storage.Battery');
        b=DSSCircuit.ActiveElement.Name;
        command = '? Storage.Battery.kWhstored';
        DSSText.Command = command;
        DSSText.Result;
        energyStoredm(i) = str2num(DSSText.Result); % Valores actualizados al minuto
        energyStoredh(j)=str2num(DSSText.Result);   % Valores actualizados a la hora.
        SOC=energyStoredh(j); %Estado de carga para simulaciones con paso: 1 hora
        SOCm=energyStoredm(i); %Estado de carga para simulaciones con paso: 1 min.
```

Demandas de energía doméstica

Las demandas de energía domésticas de la smartgrid se considerarán como una carga variable en el tiempo. Esto se modela en OpenDSS con un *LoadShape* como ya se ha mencionado.

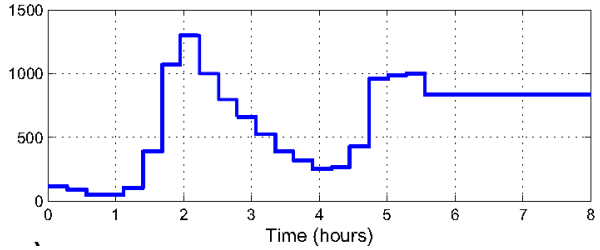
Por otro lado también se modelará la demanda de energía de un coche eléctrico que se conecte a la red. Por simplicidad, se considerará que sólo se carga una vez al día y a una velocidad de carga constante.

Estos perfiles de demanda de energía son los mismos que se han utilizado en la planta experimental. Como se puede comprobar en la figura 5.9, la demanda de energía se concentra principalmente en dos períodos (mediodía y final de la tarde). Por otro lado, el consumo de energía del coche eléctrico se ha modelado en la última mitad del día. Mientras que el aporte de energía a la red por parte de la planta solar se concentra en la primera mitad del día.

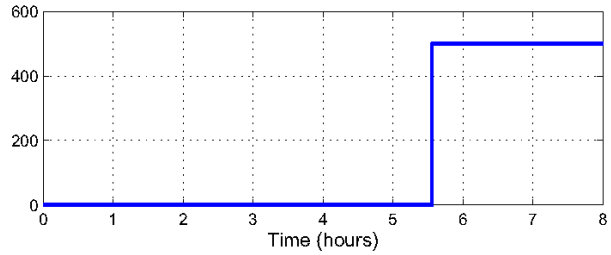
5.3 Gestión del modelo de OpenDSS en Matlab

Aunque el modelado de los dispositivos se hace mediante la herramienta OpenDSS, los comandos de modelado y control se realizarán desde Matlab generando un único script que permitirá modificar rápidamente cualquier parámetro del modelo. Es decir, aunque la simulación como tal se realiza por el programa OpenDSS, no es necesario tener abierto este software, ya que Matlab se encarga de hacer las llamadas necesarias a través del servidor COM.

El código completo de Matlab con el que se ha modelado y simulado la planta experimental se encuentra en el apéndice C (Código C.1).



(a) Demanda doméstica en W.



(b) Demanda vehículo eléctrico (W).

Figura 5.9 Demandas de energía variable de la red.

5.3.1 Estrategia de control

La estrategia de control para esta microgrid será la denominada en la literatura como *estrategia de control por histéresis*. Dicha estrategia se basa en controlar la activación o desactivación de los elementos del circuito en función del estado de carga de las baterías (SOC, *Status of charge*). Esto permite solventar los desajustes leves entre generación de electricidad y demanda de la misma.

La estrategia tiene como objetivo satisfacer la demanda de energía eléctrica siempre que sea posible, gestionando los excesos y defectos de energía mediante los sistemas de almacenamiento de que se disponga. En este caso se puede almacenar energía eléctrica directamente en el pack de baterías o en forma de hidrógeno para más tarde ser convertido en electricidad mediante al pila de combustible. Es decir, el exceso o defecto de energía durante largos periodos de tiempo, se compensa mediante el empleo del electrolizador y pila de combustible, cuyo sistema de control detecta dichos defectos o excesos en base al estado de carga de las baterías. (Valverde Isorna, Luis. *Estudio sobre el uso de convertidores DC/DC en instalaciones de almacenamiento de energía eléctrica de origen renovable*).

La técnica de introducción de una banda de histéresis en los límites del estado de carga de las baterías otorga gran flexibilidad de operación en el electrolizador, pila de combustible y las propias baterías. En este sentido, los componentes pueden ser protegidos de altos e innecesarios factores de utilización y de la operación a carga variables, reduciendo los encendidos y apagados frecuentes, prolongando notablemente la vida de los equipos. (Dimitris Ipsakis, 2008).

Un esquema de la estrategia se puede ver en la figura 5.10:

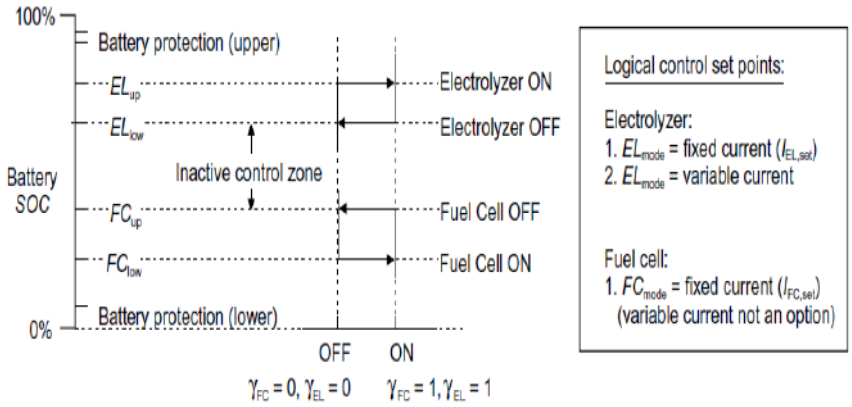


Figura 5.10 Estrategia de control por banda de histéresis. (Ulleberg, 2004).

El código preliminar creado en Matlab es el siguiente:

Código 5.2 Control por banda de histéresis en Matlab .

```
%% ESTRATEGIA DE CONTROL: CONTROL POR HISTÉRESIS
        StoragePowerMag(i)
            if SOC > 0.45*18.92 && SOC < 0.75*18.92

                disp('BATERÍA DENTRO DE LOS LÍMITES DE FUNCIONAMIENTO: 45%<SOC<75%')

                  t=1;
            else
                t=2;
                disp('BATERÍA FUERA DE LÍMITES NORMALES DE FUNCIONAMIENTO')
            end
                if t==1;
                    if SOC < 0.5*18.92 && SOC > 0.45*18.92
                        situacion(j)=2;
                        disp('Activar Pila de combustible para cargar batería: 45%<SOC<50%')
                        DSSText.command = 'batchedit Generator.FC.* enabled=yes kv=0.048';
                        DSSText.Command = ['Generator.FC.kW=' num2str(Potencia_PilaCombustible)];
                        DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state=
                            charging' ];
                        DSSText.Command = ['Storage.Battery. state =charging' ];
                        DSSText.Command = ['Storage.Battery.%charge=15'];
                        DSSText.Command = ['Storage.Battery.%reserve=' num2str(35)];
                        DSSText.command = 'batchedit load. Electrolizador .* enabled=no';


                        %DSSText.Command = ['Load.Electrolizador.kW=' num2str( Potencia_recibe_electrolizador ) ];
                    end
                    if SOC < 0.75*18.92 && SOC > 0.65*18.92
                        disp('Activar Electrolizador : 70%<SOC<75%')
                          situacion(j)=4;
                        DSSText.command = 'batchedit Generator.FC.* enabled=no';
                        DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state=
                            discharging' ];
                        DSSText.Command = ['Storage.Battery. state =discharging' ];
                        DSSText.Command = ['Storage.Battery.%reserve=' num2str(60)];
                        DSSText.Command = ['Storage.Battery.%discharge=10'];
                        DSSText.command = 'batchedit load. Electrolizador .* enabled=yes';
                        DSSText.Command = ['Load.Electrolizador.kW=' num2str( Potencia_recibe_electrolizador )];
                        DSSText.Command = ['Load.Carga.kW=' num2str(265)];

                    end

                    if SOC < 0.7*18.92 && SOC > 0.50*18.92
                        disp ('Zona de funcionamiento sin control: 50%<SOC<70%')
```

```
                        situacion (j)=3;
                        DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state =
                            IDLING'];
                        DSSText.Command = ['Storage.Battery. state =IDLING'];
                        DSSText.Command = ['Load.Carga.kW=' num2str(265)];
                        DSSText.Command = ['Storage.Battery.%IdlingkW=0.01'];
                        DSSText.command = 'batchedit Generator.FC.* enabled=no';
                        DSSText.command = 'batchedit load. Electrolizador .* enabled=no';
                    end

                end %Fin t=1

                if t==2;

                if SOC > 0.75*18.92
                    situacion (j)=5;
                disp('Batería con exceso de carga. Electrolizador se activa')
                DSSText.command = 'batchedit Generator.FC.* enabled=no';
                DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state =discharging' '
                    kv=0.048'];
                DSSText.Command = ['Storage.Battery.%discharge=25'];
                DSSText.Command = ['Storage.Battery.kwrated=0.5'];
                DSSText.Command = ['Storage.Battery.%reserve=' num2str(60)];
                DSSText.command = 'batchedit load. Electrolizador .* enabled=yes kv=0.048';
                DSSText.Command = ['Load.Electrolizador.kW=' num2str( Potencia_recibe_electrolizador )];
                DSSText.Command = ['Load.Carga.kW=' num2str(400)];
                end

            if SOC < 0.45*18.92
                disp('Batería con defecto de carga. Pila de combustible activada')
                situacion (j)=1;
                    DSSText.command = 'batchedit Generator.FC.* enabled=yes';
                    DSSText.Command = ['Generator.FC.kW=' num2str(Potencia_PilaCombustible)];
                    DSSText.command = 'batchedit load. Electrolizador .* enabled=no';
                    DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state =
                        charging' 'kv=0.048'];
                    DSSText.Command = ['Storage.Battery. state =charging'];
                    DSSText.Command = ['Storage.Battery.%charge=50'];
                    DSSText.Command = ['Storage.Battery.%reserve=' num2str(30)];
                    DSSText.Command = ['Load.Carga.kW=' num2str(250)];
                end
            end %Fin de zona t=2
        end
Batt=StoragePowerMag(i)
iter = iter +1;
```

De tal modo que el electrolizador se activa cuando el estado de carga de las baterías se encuentra en un nivel alto, lo que indica que la smartgrid tiene energía de sobra y es posible consumir energía en la generación de hidrógeno. Una vez que el electrolizador está conectado, se mantendrá así hasta que el estado de las baterías baje hasta cierto punto inferior al punto de encendido del electrolizador.

Por otro lado, la pila de combustible se enciende a su vez cuando el estado de carga de las baterías (SOC) está a un nivel bajo y se apaga cuando alcanza cierto nivel superior al punto de encendido de la pila de combustible.

En Matlab esto se ha realizado en el mismo script donde se ha modelado el circuito mediante varios bucles *if* dentro del bloque *solve* de OpenDSS. Esto es, se configura OpenDSS para realizar una simulación temporal diaria, con un paso entre simulaciones de 1 minuto, mientras que en Matlab se van analizando los resultados a cada hora mediante un bucle *for*, haciendo las correcciones y operaciones de control pertinentes en función de los resultados dados por OpenDSS, en un proceso de retroalimentación entre ambos software.

5.3.2 Generación y consumo de hidrógeno

Tal y como se ha mencionado previamente, la generación y el consumo de hidrógeno en el sistema es directamente proporcional a la intensidad de los elementos generadores y consumidores respectivamente.

La producción de hidrógeno en el electrolizador, $\dot{m}_{H_2,ez}$, se obtiene directamente de la intensidad del electrolizador con la relación:

$$\dot{m}_{H_2,ez} = n_{ez}\frac{I_{ez}}{F} \tag{5.5}$$

donde I_{ez} es la corriente del electrolizador, n_{ez} es el número de celdas del electrolizador y F es la constante de Faraday.

El consumo de hidrógeno en la pila de combustible se puede estimar de forma análoga como:

$$\dot{m}_{H_2,fc-cons} = \frac{I_{st}}{2FA_{fc}} \tag{5.6}$$

La energía producida en forma de electricidad es:

$$V_{st} = n_{st}V_{fc}, \qquad P_{st} = V_{st}I_{st} \tag{5.7}$$

siendo P_{st} la potencia de la pila, V_{st} la tensión y I_{st} la intensidad.

Para obtener en OpenDSS la intensidad de un elemento del circuito, simplemente hay que seleccionar como elemento activo el dispositivo y escribir un comando para que el programa registre las intensidades, pudiendo ser almacenadas en alguna variable de Matlab.

Código 5.3 Obtención de intensidades y almacenamiento en Matlab .

```
%INTENSIDAD EN ELECTROLIZADOR
DSSCircuit.SetActiveElement('Load. Electrolizador ');
 intensidad_electrolizador =DSSCircuit.ActiveElement.Currents;

%INTENSIDAD EN PILA DE COMBUSTIBLE
DSSCircuit.SetActiveElement('Generator.FC');
 intensidad_pilaCombustible =DSSCircuit.ActiveElement.Currents;
```

Quedando el código que calcula la generación y consumo de hidrógeno de esta manera:

Código 5.4 Obtención de intensidades y almacenamiento en Matlab .

```
%% PRODUCCIÓN Y CONSUMO DE HIDRÓGENO
            % Obtención de intensidades en electrolizador y pila de
            % combustible
            F=9.6496e4; %Constante de Faraday
            DSSCircuit.SetActiveElement('Load. Electrolizador ');
             inten_electr =DSSCircuit.ActiveElement.Currents;  %Vector con 4 componentes. Nos interesa sólo la
                 primera.
             inten_elect (j)= inten_electr (1);
            Hidrogeno_p(j)=30*inten_elect(j)/F;       % Hidrógeno producido

            DSSCircuit.SetActiveElement('Generator.FC');
            inten_FC=DSSCircuit.ActiveElement.Currents;
             inten_fct (j)=inten_FC(1);
            A_fc=1.4826; %Área efectiva de intercambio de la pila de combustible {m^2]
            Hidrogeno_c(j)= inten_fct (j)/(2*F*A_fc); % Hidrógeno consumido

            Hidrogeno_reserva=Hidrogeno_reserva+Hidrogeno_p(j)+Hidrogeno_c(j); % Hidrógeno almacenado
            Hidro_vector(j)=Hidrogeno_reserva; % Hidrógeno almacenado en cada iteraci ón.
```

5.3.3 Eficiencia del electrolizador y pila de combustible

Como ya se indicó anteriormente, se dispone de las curvas de eficiencia experimentales de un electrolizador de 1.5 kW junto con su convertidor(Ver Figura 5.6). Siendo dicha curva muy similar para la pila de combustible. En ambos casos se compara la potencia existente en el bus con el de la potencia efectiva que recibe el electrolizador o que cede la pila de combustible. Para poder modelar estas curvas, se definirá una función en

Matlab que tendrá como entrada la potencia del bus para el caso del electrolizador y como salida la potencia efectiva que recibe dicho elemento. Dentro de la función se compararán los valores de la simulación con los experimentales y se seleccionará la eficiencia correspondiente a dicha potencia. En caso de la pila de combustible será igual pero a la inversa, es decir, la entrada a la función será la potencia generada por la pila de combustible en la simulación y como salida será la potencia efectiva cedida al bus al que está conectada tras pasar por el convertidor y teniendo en cuenta la eficiencia de la propia pila de combustible.

5.4 Resultados de la simulación

5.4.1 Central solar

La potencia entregada por los paneles solares fotovoltaicos, considerando la media por hora, se puede observar en la figura 5.11.

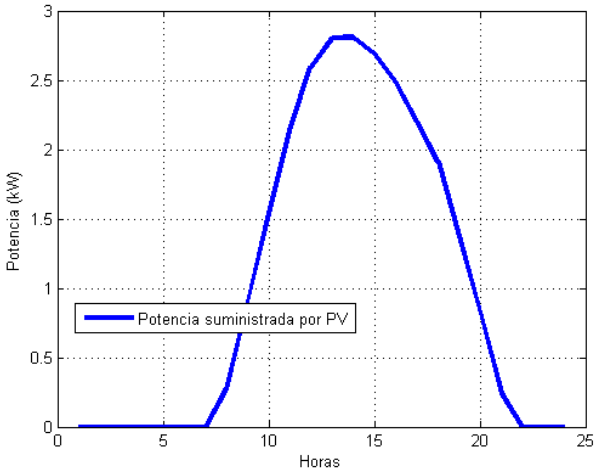


Figura 5.11 Potencia entregada por los paneles solares por hora.

Como puede observarse, los paneles solares comienzan a generar energía a partir de las 6 de la mañana (GMT) aproximadamente y dejan de hacerlo sobre las 20 horas (un día largo de verano). Sin embargo, sólo entre las 10 y las 16 horas se consiguen superar las 2 kW. La máxima potencia se entrega entre las 12 y las 13 horas, viéndose penalizado el rendimiento de las placas solares poco después del máximo debido al aumento de temperatura de éstas tras varias horas siendo irradiadas. Cuanto menor temperatura de la placa, mayor eficiencia.

5.4.2 Demanda de la red

Los perfiles de demanda se han generado basándose en los datos utilizados en la planta experimental, considerando como ya se indicó, la demanda doméstica y la del coche eléctrico que se carga durante las últimas horas del día.

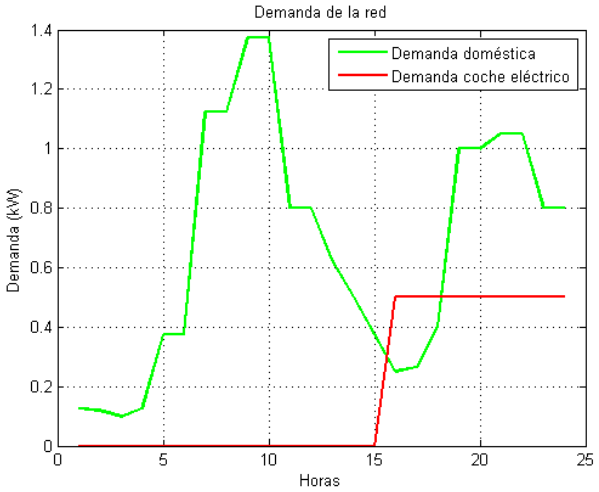


Figura 5.12 Demanda de la red por hora.

5.4.3 Carga de la batería

El pack de baterías tiene una importancia capital en esta simulación porque como se ha indicado, la estrategia de control está basada en este parámetro de las baterías (Ver Figura 5.10). La estrategia de control por banda de histéresis ha sido modelada en Matlab como puede verse en el Código 5.2 de la sección 5.3.1.

El resultado gráfico puede verse en la figura 5.13 generada por Matlab, donde se ha estimado que la carga máxima de las baterías es 18.92 kWh. La activación y desactivación del electrolizador y de la pila de combustible se lleva a cabo entre los valores de 70-75 % y 45-50 % respectivamente. Para generar la gráfica se ha almacenado en un vector el estado de carga (en *kWh*) en cada hora de la simulación; y se ha representado este vector frente al tiempo, ya que OpenDSS no muestra esta información en simulaciones temporales.

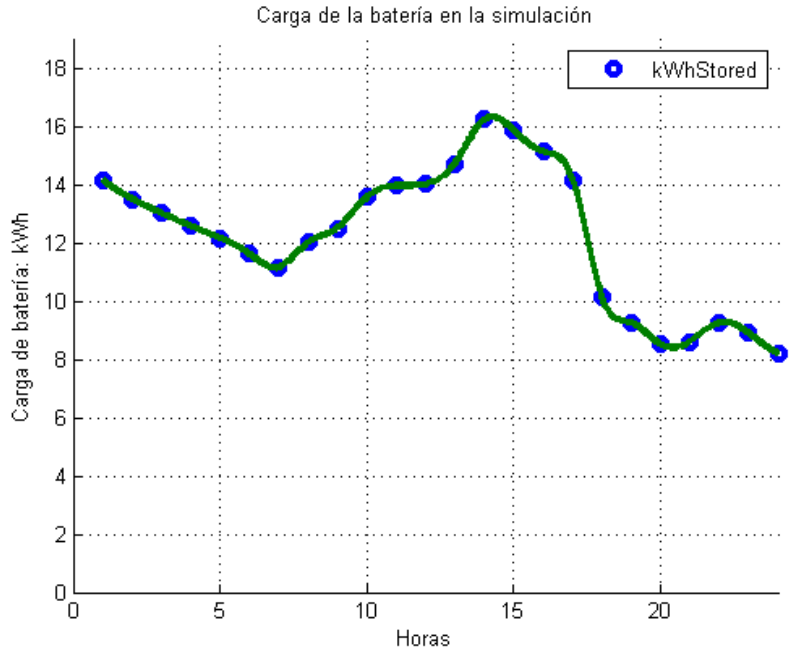


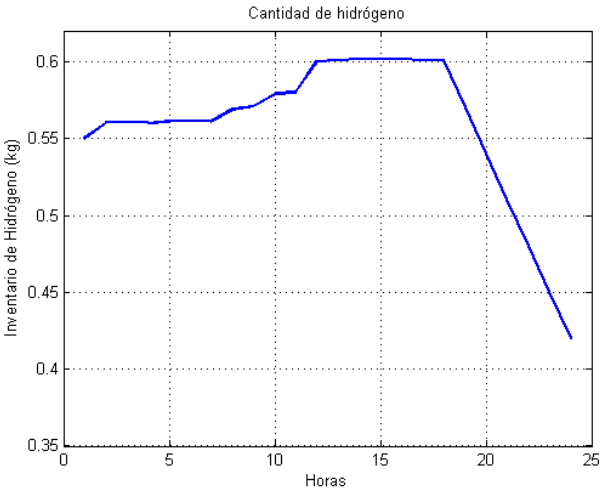
Figura 5.13 Carga de la batería durante la simulación.

Como puede verse, durante las primeras horas del día (donde se ha supuesto que la batería se encontraba en un nivel medio-alto de carga), la batería empieza a descargarse ya que los paneles solares aún no generan energía y la pila de combustible se encuentra desactivada. En cuanto el panel solar comienza a producir energía, las baterías comienzan a cargarse, llegando al máximo de carga admitido (nunca se cargan al 100 % como protección), por lo que después de este punto, el exceso de energía se comienza a utilizar para activar el electrolizador hasta que la caída de carga de la batería y la reducción de producción de energía solar, exige que la pila de combustible sea activada. Por eso se ve un pequeño repunte de la carga de la batería en las últimas horas del día.

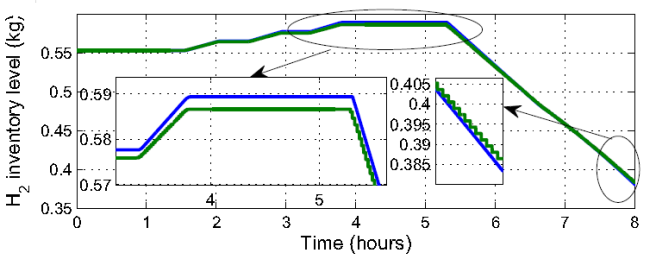
Hay que tener en cuenta que esta figura representa un comportamiento medio por hora de la batería, ya que al realizar la simulación con un paso de 1 minuto, por ejemplo, las oscilaciones de carga son mucho más acentuadas.

5.4.4 Producción de hidrógeno

La producción de hidrógeno según la simulación se puede ver en la figura 5.14 y una comparativa con el documento de referencia [18].



(a) Simulación OpenDSS - Matlab.



(b) Simulación paper.

Figura 5.14 Comparación simulación OpenDSS con el documento de referencia.

Como puede comprobarse, aunque no muy exactos, sí que se consiguen unos resultados cualitativamente válidos, aceptables teniendo en cuenta que el modelo realizado es más simple que el del citado documento.

6 Conclusiones

En este capítulo se van a enunciar las conclusiones que se han alcanzado tras utilizar el software OpenDSS junto con el paquete Matlab para trabajar con diferentes tipos de sistemas eléctricos.

6.1 Conclusiones sobre la utilización de OpenDSS

La selección de OpenDSS como herramienta de simulación presenta ventajas evidentes, ya que permite llevar a cabo estudios con carga constante (tipo *snapshot*), y por otro lado, estudios que tienen en cuenta cargas que varían con el tiempo (en este caso diarias), lo que proporciona la capacidad de contemplar condiciones más reales para el sistema. Además los modelos de los elementos de red utilizados por OpenDSS permiten analizar sistemas complejos con características reales. Otra de sus grandes ventajas es la capacidad de comunicarlo y retroalimentase con otros paquetes software, en este trabajo se ha utilizado Matlab, ampliando mucho las capacidades que ofrece el software.

Como contra, OpenDSS es un software aún en fase de desarrollo y que continúa solucionando errores y ampliando sus capacidades poco a poco, por lo que la información que se puede encontrar en internet es realmente reducida, casi inexistente, disminuyendo muchísimo la pendiente de la curva de aprendizaje. Se hace complicado familiarizarse con el entorno, el lenguaje y los métodos que utiliza el programa. El entorno gráfico es muy poco amigable para el usuario y está orientado a uso profesional, poco aconsejable para usuarios no familiarizados con este tipo de herramientas.

6.1.1 Conclusiones sobre modelado de sistemas de generación distribuida

El software OpenDSS está especialmente diseñado para el análisis y simulación de este tipo de sistemas eléctricos, tanto para pequeñas redes como para grandes sistemas de distribución, por lo que presenta ventajas evidentes si se desea utilizar para este fin. En el presente documento se ha presentado únicamente unas cuantas posibilidades (modelado de fuentes de energía renovable y elementos de almacenamiento) para un sistema de solo 37 nodos, pero las posibilidades para el estudio de los sistemas de generación distribuida es casi ilimitada. Se pueden realizar modelos realmente complejos y extensos (miles de nodos y distancias casi ilimitadas entre ellos), con la única limitación de ser ordenado y metódico en la tarea de definir y asignar elementos y buses para evitar errores difícilmente localizables.

6.1.2 Conclusiones sobre modelado de Smart-Grid basada en hidrógeno

Para el modelado de la planta experimental basada en hidrógeno, han aparecido varios inconvenientes. Para empezar el software no está orientado para los análisis en tensión continua y de tan bajo voltaje, lo que supone que las características de los *objetos* de OpenDSS estén diseñadas para medias y altas tensiones de alterna; tampoco dispone de convertidores DC/DC, sí AC/DC o DC/AC, lo que limita la veracidad del modelo, ya que hay que utilizar curvas experimentales para modelar las pérdidas y la eficiencia de la conversión de voltaje. Sin embargo, aunque en este documento se ha realizado un modelo más o menos simple, se han conseguido resultados aceptables, no demasiado dispares a los resultados experimentales, por lo que todo indica que se puede utilizar el software para este tipo de estudios, siendo consciente de sus limitaciones para este tipo de

redes de baja tensión en DC.

La labor de modelado de las baterías posiblemente sea la más compleja en esta tarea, pero la posibilidad de utilizar Matlab para controlar este elemento, facilita en cierta medida este modelado. Sin embargo no es nada trivial ajustar las propiedades del *storage element* de OpenDSS para obtener resultados similares al comportamiento de una batería real, ya de por si complejo. Para futuros modelos se recomienda utilizar otros objetos de OpenDSS diferentes al *StorageElement* que hagan más sencillo el modelado de las baterías, por ejemplo utilizando un objeto *generador* y otro *load*o *carga*, haciendo un poco más complejo la caracterización pero más simple el control de voltaje y de potencia entregada/absorbida. Ya que los resultados con el *StorageElement* no han sido satisfactorios en este control.

OpenDSS sí presenta ventajas a la hora de modelar los paneles solares, ya que dispone de modelos propios altamente configurables con resultados muy próximos a la realidad.

El electrolizador y la pila de combustible han sido modelados como una carga variable y un generador respectivamente. Pudiendo haber sido comunicados mediante la producción y consumo de hidrógeno a través de Matlab. En las simulaciones que se han presentado sólo se ha utilizado un modo de funcionamiento de los dos posibles para el electrolizador y pila de combustible; en concreto el modo a carga nominal, que es más simple de modelar. Para futuros modelos habría que implementar también el modo de funcionamiento a carga parcial, donde el electrolizador absorbe la diferencia entre la potencia generada y la demandada, haciendo más eficiente el funcionamiento.

6.2 Objetivos futuros

Con respecto a los sistemas de generación distribuida, existen en la actualidad diversos proyectos avanzados con OpenDSS en la optimización de este tipo de sistemas, por lo que es de especial interés el modelado de microgrids y smartgrids, mucho menos investigadas con este software. Por lo que algunos de los objetivos futuros podrían ser:

- Mejorar el modelo de la planta experimental:
  - Obtener más datos del comportamiento del pack de baterías y centrarse en modelar en OpenDSS y Matlab dicho comportamiento de una forma más realista, por ejemplo controlando la profundidad de las cargas y descargas, añadiendo los tiempos de actuación de las baterías (*delay*) o incluyendo los *StorageController* de los que dispone OpenDSS y que en el presente documento no se han considerado.
  - Probar otros modelos de baterías que no utilicen el objeto *StorageElement*, sino los objetos *Generador* y *Carga*. Haciendo más sencillo el control de tensión y potencias cedidas/absorbidas de la red.
  - Incluir los dos modos de funcionamiento del electrolizador y la pila de combustible: a carga parcial, donde el electrolizador absorbe la diferencia entre la potencia generada y demandada y la pila genera una potencia variable en función de las necesidad de la microred; y a carga nominal, funcionando el electrolizador y la pila de combustible a potencia constante (el modo considerado en el presente documento).
  - Modelar y considerar el comportamiento dinámico de las temperaturas de los elementos del sistema. Considerar las pérdidas en forma de calor así como la variación en la eficiencia de los dispositivos.
  - Modelar las pérdidas de la red. Para ello se podría modelar correctamente el cableado de la red para considerar las pérdidas que ahí se generan.
  - Probar algún modelo experimental de convertidor DC/DC. Existe un convertidor experimental llamado *VSConverter* que está diseñado para funcionar como interfaz ente secciones AC y DC de un circuito. Si hay alguna forma de modelar un convertidor DC/DC en OpenDSS, muy posiblemente sea con esta herramienta. Lamentablemente hay poco o nada de información en internet sobre su funcionamiento y uso.
  - Realizar análisis con todas las herramientas que permite OpenDSS para detectar fallos en el modelado y simulación con el fin de obtener resultados mejores y más cercanos a la realidad.

- Mejora de la simulación:
  - Utilizar otros modos de simulación que ofrece OpenDSS como el *Duty cycle*, que simula ciclos de trabajo con incrementos de 1 a 5 segundos, útil para el análisis de la generación de energía de las renovables o el comportamiento transitorio de ciertos dispositivos. O el modo *Dynamics* que permite la simulación de transitorios electromecánicos.
  - Control efectivo de la tensión del bus principal, con el objetivo de mantenerlo cercano a los 48 VDC.

A Códigos de OpenDSS

Código A.1 Script del circuito de ejemplo .

```
Clear
new object= circuit .DSSLLibtestckt
~ basekv=115 1.00 0.0 60.0 3 20000 21000 4.0 3.0 ! edit the voltage source

new loadshape.day 24 1.0
~ mult=(.3 .3 .3 .35 .36 .39 .41 .48 .52 .59 .62 .94 .87 .91 .95 .95 1.0 .98 .94 .92 .61 .60 .51 .44)
new loadshape.year 24 1.0 ! same as day for now
~ mult=".3 .3 .3 .35 .36 .39 .41 .48 .52 .59 .62 .94 .87 .91 .95 .95 1.0 .98 .94 .92 .61 .60 .51 .44"
new loadshape.wind 2400 0.00027777 ! unit must be hours 1.0/3600.0 = .0002777
~ csvfile =zavwind.csv action=normalize ! wind turbine characteristi

! define a linecode for the lines - unbalanced 336 MCM ACSR connection
new linecode.336matrix nphases=3 ! horizontal flat construction
~ rmatrix=(0.0868455 | 0.0298305 0.0887966 | 0.0288883 0.0298305 0.0868455) ! ohms per 1000 ft
~ xmatrix=(0.2025449 | 0.0847210 0.1961452 | 0.0719161 0.0847210 0.2025449)
~ cmatrix=(2.74 | -0.70 2.96| -0.34 -0.71 2.74) !nf per 1000 ft
~ Normamps = 400 Emergamps=600

! Substation transformer
new transformer.sub phases=3 windings=2 buses=(SourceBus subbus) conns='delta wye' kvs="115 12.47 " kvas="20000
    20000" XHL=7

! define the lines
new line.line1 subbus loadbus1 linecode=336matrix length=10
new line.line2 loadbus1 loadbus2 336matrix 10
new line.line3 Loadbus2 loadbus3 336matrix 20

! define a couple of loads
new load.load1 bus1=loadbus1 phases=3 kv=12.47 kw=1000.0 pf=0.88 model=1 class=1 yearly=year daily=day status=fixed
new load.load2 bus1=loadbus2 phases=3 kv=12.47 kw=500.0 pf=0.88 model=1 class=1 yearly=year daily=day conn=delta
    status=fixed

! Capacitor with control
new capacitor.C1 bus1=loadbus2 phases=3 kvar=600 kv=12.47
new capcontrol.C1 element=line.line3 1 capacitor=C1 type=current ctratio=1 ONsetting=60 OFFsetting=55 delay=2

! regulated transformer to DG bus
new transformer.reg1 phases=3 windings=2
~ buses=(loadbus3 regbus)
~ conns='wye wye'
~ kvs="12.47 12.47"
~ kvas="8000 8000"
~ XHL=1 ! tiny reactance for a regulator

! Regulator Control definitions
new regcontrol.sub transformer=sub winding=2 vreg=125 band=3 ptratio=60 delay=10
new regcontrol.reg1 transformer=reg1 winding=2 vreg=122 band=3 ptratio=60 delay=15

! define a wind generator of 8MW
```

```
New generator.gen1     bus1=regbus kV=12.47 kW=8000 pf=1 conn=delta duty=wind Model=1


! Define some monitors so's we can see what's happenin'

New Monitor.gen1a element=generator.gen1 1     mode=48
New Monitor.line3 element=line.line3 1        mode=48
New Monitor.gen1  element=generator.gen1 1 mode=32

! Define voltage bases so voltage reports come out in per unit
Set voltagebases="115 12.47 .48"
Calcv

Set controlmode=time
Set mode=duty number=2400 hour=0 h=1.0 sec=0  ! Mode resets the monitors
```

Código A.2 Script en OpenDSS del sistema de distribución IEEE 37 sin modificar .

```
Clear

New object= circuit.ieee37
~ basekv=230 pu=1.00 MVAsc3=200000 MVAsc1=210000

! Substation Transformer
New Transformer.SubXF Phases=3 Windings=2 Xhl=8
~ wdg=1 bus=sourcebus conn=Delta kv=230  kva=2500  %r=1
~ wdg=2 bus=799       conn=Delta kv=4.8  kva=2500  %r=1

! Load Transformer
New Transformer.XFM1 Phases=3 Windings=2 Xhl=1.81
~ wdg=1 bus=709       conn=Delta kv=4.80  kva=500  %r=0.045
~ wdg=2 bus=775       conn=Delta kv=0.48  kva=500  %r=0.045

! import line codes with phase impedance matrices
Redirect          IEEELineCodes.dss

! Lines
New Line.L1     Phases=3 Bus1=701.1.2.3  Bus2=702.1.2.3  LineCode=722 Length=0.96
New Line.L2     Phases=3 Bus1=702.1.2.3  Bus2=705.1.2.3  LineCode=724 Length=0.4
New Line.L3     Phases=3 Bus1=702.1.2.3  Bus2=713.1.2.3  LineCode=723 Length=0.36
New Line.L4     Phases=3 Bus1=702.1.2.3  Bus2=703.1.2.3  LineCode=722 Length=1.32
New Line.L5     Phases=3 Bus1=703.1.2.3  Bus2=727.1.2.3  LineCode=724 Length=0.24
New Line.L6     Phases=3 Bus1=703.1.2.3  Bus2=730.1.2.3  LineCode=723 Length=0.6
New Line.L7     Phases=3 Bus1=704.1.2.3  Bus2=714.1.2.3  LineCode=724 Length=0.08
New Line.L8     Phases=3 Bus1=704.1.2.3  Bus2=720.1.2.3  LineCode=723 Length=0.8
New Line.L9     Phases=3 Bus1=705.1.2.3  Bus2=742.1.2.3  LineCode=724 Length=0.32
New Line.L10    Phases=3 Bus1=705.1.2.3  Bus2=712.1.2.3  LineCode=724 Length=0.24
New Line.L11    Phases=3 Bus1=706.1.2.3  Bus2=725.1.2.3  LineCode=724 Length=0.28
New Line.L12    Phases=3 Bus1=707.1.2.3  Bus2=724.1.2.3  LineCode=724 Length=0.76
New Line.L13    Phases=3 Bus1=707.1.2.3  Bus2=722.1.2.3  LineCode=724 Length=0.12
New Line.L14    Phases=3 Bus1=708.1.2.3  Bus2=733.1.2.3  LineCode=723 Length=0.32
New Line.L15    Phases=3 Bus1=708.1.2.3  Bus2=732.1.2.3  LineCode=724 Length=0.32
New Line.L16    Phases=3 Bus1=709.1.2.3  Bus2=731.1.2.3  LineCode=723 Length=0.6
New Line.L17    Phases=3 Bus1=709.1.2.3  Bus2=708.1.2.3  LineCode=723 Length=0.32
New Line.L18    Phases=3 Bus1=710.1.2.3  Bus2=735.1.2.3  LineCode=724 Length=0.2
New Line.L19    Phases=3 Bus1=710.1.2.3  Bus2=736.1.2.3  LineCode=724 Length=1.28
New Line.L20    Phases=3 Bus1=711.1.2.3  Bus2=741.1.2.3  LineCode=723 Length=0.4
New Line.L21    Phases=3 Bus1=711.1.2.3  Bus2=740.1.2.3  LineCode=724 Length=0.2
New Line.L22    Phases=3 Bus1=713.1.2.3  Bus2=704.1.2.3  LineCode=723 Length=0.52
New Line.L23    Phases=3 Bus1=714.1.2.3  Bus2=718.1.2.3  LineCode=724 Length=0.52
New Line.L24    Phases=3 Bus1=720.1.2.3  Bus2=707.1.2.3  LineCode=724 Length=0.92
New Line.L25    Phases=3 Bus1=720.1.2.3  Bus2=706.1.2.3  LineCode=723 Length=0.6
New Line.L26    Phases=3 Bus1=727.1.2.3  Bus2=744.1.2.3  LineCode=723 Length=0.28
New Line.L27    Phases=3 Bus1=730.1.2.3  Bus2=709.1.2.3  LineCode=723 Length=0.2
New Line.L28    Phases=3 Bus1=733.1.2.3  Bus2=734.1.2.3  LineCode=723 Length=0.56
New Line.L29    Phases=3 Bus1=734.1.2.3  Bus2=737.1.2.3  LineCode=723 Length=0.64
New Line.L30    Phases=3 Bus1=734.1.2.3  Bus2=710.1.2.3  LineCode=724 Length=0.52
New Line.L31    Phases=3 Bus1=737.1.2.3  Bus2=738.1.2.3  LineCode=723 Length=0.4
```

```
New Line.L32     Phases=3 Bus1=738.1.2.3   Bus2=711.1.2.3   LineCode=723 Length=0.4
New Line.L33     Phases=3 Bus1=744.1.2.3   Bus2=728.1.2.3   LineCode=724 Length=0.2
New Line.L34     Phases=3 Bus1=744.1.2.3   Bus2=729.1.2.3   LineCode=724 Length=0.28
New Line.L35     Phases=3 Bus1=799r.1.2.3  Bus2=701.1.2.3   LineCode=721 Length=1.85

! Regulator - open delta with C leading, A lagging, base LDC setting is 1.5 + j3
new transformer.reg1a phases=1 windings=2 buses=(799.1.2 799r.1.2) conns='delta delta' kvs="4.8 4.8" kvas="2000
      2000" XHL=1
new regcontrol.creg1a transformer=reg1a winding=2 vreg=122 band=2 ptratio=40 ctprim=350 R=-0.201 X=3.348
new transformer.reg1c like=reg1a buses=(799.3.2 799r.3.2)
new regcontrol.creg1c like=creg1a transformer=reg1c R=2.799 X=1.848
New Line.Jumper Phases=1 Bus1=799.2       Bus2=799r.2       r0=1e-3 r1=1e-3 x0=0 x1=0 c0=0 c1=0

! spot loads
% En el nodo 701 se implementará el DG
New Load.S701a       Bus1=701.1.2 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S701b       Bus1=701.2.3 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S701c       Bus1=701.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 350.0 kVAR= 175.0

New Load.S712c       Bus1=712.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S713c       Bus1=713.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S714a       Bus1=714.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 17.0  kVAR=  8.0
New Load.S714b       Bus1=714.2.3 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 21.0  kVAR= 10.0
New Load.S718a       Bus1=718.1.2 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0

% En el nodo 720 se implementará la batería.
New Load.S720c       Bus1=720.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0

% En el nodo 722 se implementará el Generador eólico
New Load.S722b       Bus1=722.2.3 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S722c       Bus1=722.3.1 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 21.0  kVAR= 10.0

New Load.S724b       Bus1=724.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S725b       Bus1=725.2.3 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S727c       Bus1=727.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S728        Bus1=728   Phases=3 Conn=Delta Model=1 kV= 4.800 kW= 126.0 kVAR= 63.0
New Load.S729a       Bus1=729.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 42.0  kVAR= 21.0

% En el nodo 730 se implementará el Generador Solar.
New Load.S730c       Bus1=730.3.1 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S731b       Bus1=731.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S732c       Bus1=732.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S733a       Bus1=733.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S734c       Bus1=734.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0

New Load.S735c       Bus1=735.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S736b       Bus1=736.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 42.0  kVAR= 21.0

% En el nodo 737 se implementará el PEV (coche eléctrico)
New Load.S737a       Bus1=737.1.2 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 140.0 kVAR= 70.0
New Load.S738a       Bus1=738.1.2 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 126.0 kVAR= 62.0
New Load.S740c       Bus1=740.3.1 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S741c       Bus1=741.3.1 Phases=1 Conn=Delta Model=4 kV= 4.800 kW= 42.0  kVAR= 21.0
New Load.S742a       Bus1=742.1.2 Phases=1 Conn=Delta Model=2 kV= 4.800 kW=  8.0  kVAR=  4.0
New Load.S742b       Bus1=742.2.3 Phases=1 Conn=Delta Model=2 kV= 4.800 kW= 85.0  kVAR= 40.0
New Load.S744a       Bus1=744.1.2 Phases=1 Conn=Delta Model=1 kV= 4.800 kW= 42.0  kVAR= 21.0

Set VoltageBases = "230,4.8,0.48"
CalcVoltageBases
BusCoords IEEE37_BusXY.csv

! solve mode=direct
set maxiterations=100
solve

! show voltages LL Nodes
! show currents residual=y elements
! show powers kva elements
! show taps
```

Código A.3 Dos métodos equivalentes para la creación del objeto XYcurve .

```
// curve in separate x, y array
New XYCurve.MyEff npts=4 xarray=[0.1 0.2 0.4 1.0] yarray=[0.86 0.9 0.93 0.97]
// curve as array of x,y values, in sequence
New XYCurve.MyEff npts=4 points=[0.1, 0.86 0.2, 0.9 0.4, 0.93 1.0, 0.97]
```

B Modelado en OpenDSS

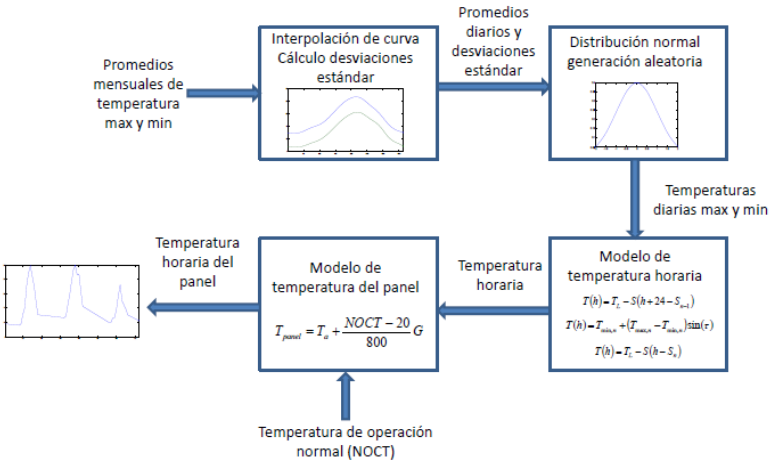


Figura B.1 Diagrama de la creación de curvas modelo para panel solar.

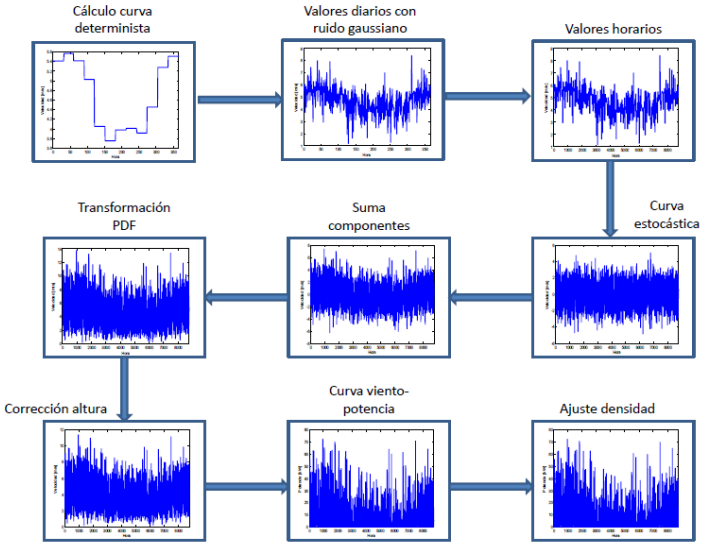


Figura B.2 Diagrama de la creación de curvas modelo para generadores eólicos.

Tabla B.1 Propiedades de la configuración de elementos de almacenaje.

%Charge	Ratio de carga (potencia de entrada) en porcentaje de la potencia nominal kW. Por defecto = 100.
%Discharge	Ratio de descarga (potencia de salida) en porcentaje de la potencia nominal. Por defecto = 100.
%EffCharge	Porcentaje de eficiencia para cargar la batería. Por defecto = 90.
%EffDischarge	Porcentaje de eficiencia para descargar la batería. Por defecto = 90. Las pérdidas en vacío son modeladas por **%IdlingkW**. Contempla tanto las pérdidas en carga y descarga como la eficiencia en el proceso de conversión de energía dentro de la unidad.
%Idlingkvar	Porcentaje de la potencia nominal consumida como reactancia (kvar) mientras está en vacío. Por defecto = 0.
IdlingkW	Porcentaje de la potencia nominal consumida mientras está en vacío. Por defecto = 1.
%reserve	Porcentaje de la capacidad de almacenamiento de kWh que se mantiene en reserva para un normal funcionamiento. Por defecto = 20. Esto se considera como el nivel mínimo de descarga de energía, a menos que haya una emergencia. Para la operación de emergencia establecer esta propiedad, que no puede ser menor que cero.
%stored	Cantidad actual de energía almacenada, porcentaje de los kWh nominales. El valor predeterminado es 100 %.
Bus1	Bus al que el elemento de almacenamiento está conectado. Puede incluir el nodo específico.
ChargeTrigger	Valor del *Trigger* para la carga de batería. El *Trigger* es el nivel de referencia. • Si = 0.0 el estado de la batería será modelado por el comando *State* o por el objeto de control *StorageController*. • Si es diferente de 0.0. El estado de la batería se establece en carga (*Charging*) cuando el nivel de *Trigger* es mayor que el especificado en la curva de Loadshape o la señal de precio o el valor global de *Loadlevel*, dependiendo del modo. Ver las propiedades de estado (*State property*) para más información.
Daily	Modo para las simulaciones diarias. Debe definirse previamente un *Loadshape* de 24 horas. En el modo predeterminado, el elemento de almacenamiento utiliza este *Loadshape* para desencadenar los cambios de estado.
DischargeTrigger	Valor del *Trigger* (referencia) para la decarga de batería. • Si = 0.0 el estado de la batería será modelado por el comando *State* o por el objeto de control *StorageController*. • Si es diferente de 0.0. El estado de la batería se cambiará a (*Discharging*) cuando el nivel de *Trigger* sobrepase el especificado en la curva de *Loadshape*, la señal de precio o el valor global de *Loadlevel*, dependiendo del modo de funcionamiento activo. Ver las propiedades de estado (*State property*) para más información.
State:	[IDLING | CHARGING | DISCHARGING] Determina/configura el estado actual de funcionamiento. En el modo *DISCHARGING*, la batería actúa como un generador y la propiedad kW es positiva. La batería continúa descargando con el nivel de potencia programado hasta que el nivel de almacenamiento alcanza el valor de reserva. A continuación, el estado vuelve a *IDLING*. En el estado *CHARGING*, la batería se comporta como una carga y la propiedad de kW es negativo. La batería continúa cargando hasta que se alcanza el máximo de almacenamiento kWh y luego cambia al estado *IDLING*. En el estado *IDLING*, el número de kW se mantiene en cero. Sin embargo, las pérdidas resistivas y reactivas se mantienen en el circuito, por lo que en el informe de flujo de potencia se mostrarán potencias consumidas.
TimeChargeTrig	Tiempo del día en horas fraccionadas (0230 = 2.5) en el que el elemento de almacenamiento pasará automáticamente al estado de carga. El valor predeterminado es 2,0. Introduzca un valor negativo de tiempo para desactivar esta función.

]mypages

Figura B.3 Script de ejemplo para modelar un panel solar.

Example Script for Exercising the PVSystem Model

This example defines a PV system with a panel Pmpp of 500 kW at 1 kW/m^2 irradiance and a panel temperature of 25°C. The inverter is rated at 500 kVA. A PF of 1.0 is assumed for this example.

```
clear

New Circuit.PVSystem  basekv=12.47  Isc3=1000 Isc1=900

// P-T curve is per unit of rated Pmpp vs temperature
// This one is for a Pmpp stated at 25 deg
New XYCurve.MyPvsT npts=4  xarray=[0  25  75  100]  yarray=[1.2 1.0 0.8  0.6]

// efficiency curve is per unit eff vs per unit power
New XYCurve.MyEff npts=4  xarray=[.1  .2  .4  1.0]  yarray=[.86  .9  .93  .97]

// per unit irradiance curve (per unit if "irradiance" property)
New Loadshape.MyIrrad npts=24 interval=1 mult=[0 0 0 0 0 0 .1 .2 .3  .5  .8  .9
1.0  1.0  .99  .9  .7  .4  .1 0  0  0  0  0]

// 24-hr temp shape curve
New Tshape.MyTemp npts=24 interval=1 temp=[25, 25, 25, 25, 25, 25, 25, 25, 35,
40, 45, 50  60 60  55 40  35  30  25 25 25 25 25 25]

// **** plot tshape object=mytemp

// take the default line
New Line.line1 Bus1=sourcebus bus2=PVbus  Length=2

// pv definition
New PVSystem.PV phases=3 bus1=PVbus kV=12.47  kVA=500  irrad=0.8  Pmpp=500
~ temperature=25 PF=1  effcurve=Myeff  P-TCurve=MyPvsT
~ Daily=MyIrrad  TDaily=MyTemp

set voltagebases=[12.47]
calcv

solve  ! solves at the specified irradiance and temperature

new monitor.m1 PVSystem.PV  1 mode=1 ppolar=no
new monitor.m2 PVSystem.PV  1

solve
solve mode=daily

show mon m1
show mon m2

Export monitors m1
Plot monitor object= m1 channels=(1 )
Export monitors m2
Plot monitor object= m2 channels=(1 ) base=[7200]
Export monitors m2
Plot monitor object= m2 channels=(9 )
```

Properties

Property	Description
(1) phases	Number of Phases, this PVSystem element. Power is evenly divided among phases.
(2) bus1	Bus to which the PVSystem element is connected. May include specific node specification.
(3) kv	Nominal rated (1.0 per unit) voltage, kV, for PVSystem element. For 2- and 3-phase PVSystem elements, specify phase-phase kV. Otherwise, specify actual kV across each branch of the PVSystem element. If 1-phase wye (star or LN), specify phase-neutral kV. If 1-phase delta or phase-phase connected, specify phase-phase kV.
(4) irradiance	Get/set the present irradiance value in kW/sq-m. Used as base value for shape multipliers. Generally entered as peak value for the time period of interest and the yearly, daily, and duty load shape objects are defined as per unit multipliers (just like Loads/Generators).
(5) Pmpp	Get/set the rated max power of the PV array for 1.0 kW/sq-m irradiance and a user-selected array temperature. The P-TCurve should be defined relative to the selected array temperature.
(6) Temperature	Get/set the present Temperature. Used as fixed value corresponding to PTCurve property. A multiplier is obtained from the Pmpp-Temp curve and applied to the nominal Pmpp from the irradiance to determine the net array output.
(7) pf	Nominally, the power factor for the output power. Default is 1.0. Setting this property will cause the inverter to operate in CONSTANT POWER FACTOR MODE. Enter negative when kW and kvar have opposite signs.
	A positive power factor signifies that the PVSystem element produces vars as is typical for a generator.
(8) conn	={wye|LN|delta|LL}. Default is wye.
(9) kvar	Get/set the present kvar value. Setting this property forces the inverter to operate in CONSTANT KVAR MODE.
(10) kVA	kVA rating of inverter. Used as the base for Dynamics mode and Harmonics mode values.
(11) %Cutin	% cut in power -- % of kVA rating of inverter. When the inverter is OFF, the power from the array must be greater than this for the inverter to turn on.
(12) %Cutout	% cut out power -- % of kVA rating of inverter. When the inverter is ON, the inverter turns OFF when the power from the array drops below this valye.

(13) EffCurve	An XYCurve object, previously defined, that describes the PER UNIT efficiency vs PER UNIT of rated kVA for the inverter. Inverter output power is discounted by the multiplier obtained from this curve.
(14) P-TCurve	An XYCurve object, previously defined, that describes the PV array PER UNIT Pmpp vs Temperature curve. Temperature units must agree with the Temperature property and the Temperature shapes used for simulations. The Pmpp values are specified in per unit of the Pmpp value for 1 kW/sq-m irradiance. The value for the temperature at which Pmpp is defined should be 1.0. The net array power is determined by the irradiance * Pmpp * f(Temperature)
(15) %R	Equivalent percent internal resistance, ohms. Default is 0. Placed in series with internal voltage source for harmonics and dynamics modes. Use a combination of %IdlekW and %EffCharge and %EffDischarge to account for losses in power flow modes.
(16) %X	Equivalent percent internal reactance, ohms. Default is 50%. Placed in series with internal voltage source for harmonics and dynamics modes. (Limits fault current to 2 pu.) Use %Idlekvar and kvar properties to account for any reactive power during power flow solutions.
(17) model	Integer code (default=1) for the model to use for power output variation with voltage. Valid values are: 1: PVSystem element injects a CONSTANT kW, kvar at specified power factor or kvar value 2: PVSystem element is modeled as a CONSTANT ADMITTANCE. 3: Compute load injection from User-written Model.
(18) Vminpu	Default = 0.90. Minimum per unit voltage for which the Model is assumed to apply. Below this value, the load model reverts to a constant impedance model.
(19) Vmaxpu	Default = 1.10. Maximum per unit voltage for which the Model is assumed to apply. Above this value, the load model reverts to a constant impedance model.
(20) yearly	Dispatch shape to use for YEARLY simulations. Must be previously defined as a Loadshape object. If this is not specified, the Daily dispatch shape, If any, is repeated during Yearly solution modes. In the default dispatch mode, the PVSystem element uses this loadshape to trigger State changes.
(21) daily	Dispatch shape to use for DAILY simulations. Must be previously defined as a Loadshape object of 24 hrs, typically. In the default dispatch mode, the PVSystem element uses this loadshape to trigger State changes.
(22) duty	Load shape to use for DUTY cycle dispatch simulations such as for solar ramp rate studies. Must be previously defined as a Loadshape object. Typically would have time intervals of 1-5 seconds. Designate the number of points to solve using the Set Number=xxxx command. If there are fewer points in the actual shape, the shape is assumed to repeat.
(23) Tyearly	Temperature shape to use for YEARLY simulations. Must be previously defined as a TShape object. If this is not specified, the Daily dispatch shape, If any, is

	repeated during Yearly solution modes. The PVSystem element uses this TShape to determine the Pmpp from the Pmpp vs T curve. Units must agree with the Pmpp vs T curve.
(24) Tdaily	Temperature shape to use for DAILY simulations. Must be previously defined as a TShape object of 24 hrs, typically. The PVSystem element uses this TShape to determine the Pmpp from the Pmpp vs T curve. Units must agree with the Pmpp vs T curve.
(25) Tduty	Temperature shape to use for DUTY cycle dispatch simulations such as for solar ramp rate studies. Must be previously defined as a TShape object. Typically would have time intervals of 1-5 seconds. Designate the number of points to solve using the Set Number=xxxx command. If there are fewer points in the actual shape, the shape is assumed to repeat. The PVSystem model uses this TShape to determine the Pmpp from the Pmpp vs T curve. Units must agree with the Pmpp vs T curve.
(26) class	An arbitrary integer number representing the class of PVSystem element so that PVSystem values may be segregated by class.
(27) UserModel	Name of DLL containing user-written model, which computes the terminal currents for Dynamics studies, overriding the default model. Set to "none" to negate previous setting.
(28) UserData	String (in quotes or parentheses) that gets passed to user-written model for defining the data required for that model.
(29) debugtrace	{Yes | No } Default is no. Turn this on to capture the progress of the PVSystem model for each iteration. Creates a separate file for each PVSystem element named "PVSystem_name.CSV".
(30) spectrum	Name of harmonic voltage or current spectrum for this PVSystem element. Current injection is assumed for inverter. Default value is "default", which is defined when the DSS starts.

C Códigos de Matlab

Código C.1 Código de Matlab para el modelado y simulación de la planta experimental de energías renovables basada en hidrógeno .

```
function Microgrid
clear all; close all; clc;
[DSSStartOK, DSSObj, DSSText] = DSSStartup('D:\Aeroespacial\TFG\Prueba OpenDSS');
    %Se comprueba si DSS se inicia correctamente
    if DSSStartOK
        %Compile the DSS circuit script
        DSSText.Command = 'Compile "D:\Aeroespacial\TFG\Prueba OpenDSS\SimpleC.DSS"';
        DSSCircuit = DSSObj.ActiveCircuit;
        DSSText = DSSObj.Text;          % Se define la interfaz de texto. Con este comando se introducirán las
            instrucciones a OpenDSS desde Matlab
        DSSCircuit = DSSObj.ActiveCircuit; % Se define la interfaz del circuito.
        DSSSolution = DSSCircuit.Solution; % Se define la interfaz de la solución.
        DSSMon=DSSCircuit.Monitors;     % Se define la interfaz de los monitores.


%% LoadShape
DSSText.command = 'New XYCurve.MyPvsT npts=4 xarray=[0 25 50 75] yarray=[1.2 1.0 0.8 0.6]'; %La curva P-T
    representa Pmpp por unidad frente a la temperatura.
DSSText.command = 'New XYCurve.MyEff npts=4 xarray=[.1 .2 .4 1.0] yarray=[.86 .9 .93 .97]'; %La curva de
    eficiencia representa unidad de eficiencia por unidad de potencia.
DSSText.command = 'New Loadshape.MyIrrad npts=24 interval=1 mult=[0 0 0 0 0 0 0.01 0.1 0.15 0.2 0.8 0.9 1.0
    1.0 0.99 0.95 0.8 0.2 0.01 0 0 0 0 0]';
DSSText.command = 'New Tshape.MyTemp npts=24 interval=1 temp=[15, 12, 12, 12, 10, 12, 15, 17, 21, 26, 30, 35 40
    45 50 48 40 35 30 25 20 15 15 15]';
irradnu=[0 0 0 0 0 0 0.01 0.1 0.15 0.2 0.8 0.9 1.0 1.0 0.99 0.95 0.8 0.2 0.01 0 0 0 0 0]; % Valores
    de irradancia
temp=[15, 12, 12, 12, 10, 12, 15, 17, 21, 26, 30, 35 40 45 50 48 40 35 30 25 20 15 15 15]; % Temperaturas.
rend_ez=[0.58233898, 0.644052402, 0.647657817, 0.647205357, 0.642767025, 0.632401619, 0.633488095, 0.624622846,
    0.620519048, 0.61574986, 0.612655462, 0.60960555, 0.616932515, 0.623429299, 0.632077209, 0.641043789,
    0.64576392, 0.645752941, 0.639526316, 0.582338983]; % Multiplicadores rendimiento conjunto Electrolizador -
    Convertidor DCDC

lsPEV=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1]; % Multiplicadores demanda coche eléctrico.
Dom=[125 120 100 125 375 375 1125 1125 1375 1375 800 800 625 500 375 250 265 400 1000 1000 1050 1050 800
    800]./1000; % Multiplicadores demanda doméstica


%% Obtención de la curva de irradancia a partir de archivo Excel.
Solh=xlsread(' radiacion_dia_soleado ', 'D2:D11217'); maxsol=max(Solh); Solh1=Solh./maxsol;
plot(0:length(Solh)-1,Solh1)
t=[0:1:length(Solh)-1]; p=polyfit(t',Solh1,4); f=polyval(p,t);
irrad=polyval(p,0:467:11216); irrad2=irrad(1:24); irrad=irrad2; irrad(1)=0;
mirrad=max(irrad); irrad=irrad./mirrad;
irradm=[0 0 0 0 0 0 76 254 450 649 813 917 961 963 934 811 655 450 258 73 0 0 0 0 0];
irrad=irradm./963;

%% Nuevas líneas para nuevos dispositivos
```

```
DSSText.command = 'New Line.Line1 Bus1=sourcebus Bus2=baux1.1 phases=1'
DSSText.command = 'New Line.Line2 Bus1=baux2.1 Bus2=b3.1 phases=1'
DSSText.command = 'New Line.Line3 Bus1=b1.1 Bus2=b2.1 phases=1'; %Pila de combustible (FC)

DSSText.command = 'New Line.Line4 Bus1=b4.1 Bus2=baux1.1 phases=1'; %Panel solar
DSSText.command = 'New Line.Line5 Bus1=b5.1 Bus2=b2.1 phases=1'; %Electrolizador
DSSText.Command = 'New Line.Line6 Bus1=b6.1 Bus2=baux2.1 phases=1'; %Coche eléctrico
DSSText.Command = 'New Line.Lineaux2 Bus1=baux2.1 Bus2=b2.1 phases=1';
DSSText.Command = 'New Line.Lineaux1 Bus1=baux1.1 Bus2=b2.1 phases=1';

%% Modelado de Dispositivos
DSSText.command = 'New PVSystem.PV phases=1 Bus1=b4.1 kV=0.096 Pmpp=3.2 temperature=20 PF=1 effcurve=Myeff P-
    TCurve=MyPvsT Daily=MyIrrad TDaily=MyTemp irrad=0.8 model=7'
DSSText.command = 'New Load.Electrolizador Bus1=b5.1 phases=1 kV=0.048 kW=3.2 PF=1 model=1 status=variable Vminpu
    =0.2 Vmaxpu=1.4'
DSSText.command = 'new Storage.Battery'
DSSText.Command = 'Storage.Battery.bus1=b3.1 phases=1 pf=1 kV=0.096 kWrated=1.2 kWhrated=18.92 kWhstored=18 %
    reserve=35 DispMode=External chargetrigger=0.0 dischargetrigger=0.0 state=discharging %IdlingkW=0.01 Vminpu
    =0.2 Vmaxpu=1. !%discharge=95 model=1 !kW=1.5 PF=1 %IdlingkW=0.1 %EffCharge=98 ' ; % difference in defining
    kwrated and kw %%stored and kwhstored are the same% DSSText.Command='New Monitor.Feeder1 line.lineA-B 2';
DSSText.command = 'New generator.FC phases=1 bus1=b1.1 kV=0.048 pf=1 basefreq=0.001 model=1 status=variable kW=1.2
     Vminpu=0.2 Vmaxpu=1.4'
DSSText.command = 'New Load.Carga Bus1=b2.1 phases=1 kV=0.048 PF=1 model=1 kW=350 status=variable Vminpu=0.2
    Vmaxpu=1.4 !model=3?'
DSSText.Command = 'New Load.Dom Bus1=b2.1 phases=1 kV=0.048 pf=1 model=1 kW=0.5 status=variable Vminpu=0.2
    Vmaxpu=1.4'
DSSText.Command = 'New Load.PEV Bus1=b6.1 phases=1 kV=0.048 PF=1 model=1 kW=0.5 status=variable Vminpu=0.2
    Vmaxpu=1.4'


%% Monitores
% Batería
DSSText.Command = 'new monitor.PQ Storage.battery 1 ppolar=no mode=1 ';
DSSText.Command = 'new monitor.Standard Storage.battery 1 mode=0';
DSSText.Command = 'new monitor.Vars Storage.battery 1 mode=3 ';

% Sistema paneles solares
DSSText.Command = 'new monitor.PV PVSystem.PV 1 ppolar=no mode=1 ';
DSSText.Command = 'new monitor.StandardPV PVSystem.PV 1 mode=0';
DSSText.Command = 'new monitor.VarsPV PVSystem.PV 1 mode=3 ';

% Pila de combustible (FC)
 DSSText.Command = 'new monitor.FCm generator.FC 1 mode=1';


%% Configuración de la simulación

    DSSText.Command = 'Set mode = daily stepsize=15 min' ;
    DSSText.Command = 'Set number=1 stepsize=15 min';
    DSSText.Command = 'Set hour=0 sec=0'
    DSSText.Command = 'Set Controlmode=dynamic';


%% Iteración y algoritmo
     iter =0;
    Hidrogeno_reserva=0.55;
        for i=1:1380
            j=floor(1+i/60)
            Hour(j)=j; % Horas
            Min(i)=i; % Minutos
                j
            Irrad=irrad(j);
            Temp=temp(j);
            PPev=0.5*lsPEV(j);
            PDom=Dom(j);

            % Se actualizan los valores de los elementos
            DSSText.Command = ['Load.Dom.kW=' num2str(PDom)];
            DSSText.Command = ['Load.PEV.kW=' num2str(PPev)];
            DSSText.command=['PVSystem.PV.irrad=' num2str(Irrad)];
            DSSText.command=['PVSystem.PV.temperature=' num2str(Temp)];
```

```
DSSText.Command = 'Set voltage="0.048"';

% Se inicia la solución
DSSText.Command = 'Solve';


%% BATERÍAS
% Obtención de energía de las baterías.
DSSCircuit.SetActiveElement('Storage.Battery');
tempvoltage=DSSCircuit.ActiveElement.Powers;
tempvoltage = reshape(tempvoltage,2,[]);
tempvoltage = hypot(tempvoltage(1,:), tempvoltage(2,:));
StoragePowerMag(i) = sum(tempvoltage(1));

% Lectura del valor de la energía almacenada en las baterías
DSSCircuit.SetActiveElement('Storage.Battery');
b=DSSCircuit.ActiveElement.Name;
command = '? Storage.Battery.kWhstored';
DSSText.Command = command;
DSSText.Result;
energyStoredm(i) = str2num(DSSText.Result); % Valores actualizados al minuto
energyStoredh(j)=str2num(DSSText.Result);   % Valores actualizados a la hora.
SOC=energyStoredh(j);
SOCm=energyStoredm(i);


%% PRODUCCIÓN Y CONSUMO DE HIDRÓGENO
% Obtención de intensidades en electrolizador y pila de
% combustible
F=9.6496e4; %Constante de Faraday
DSSCircuit.SetActiveElement('Load. Electrolizador');
 inten_electr =DSSCircuit.ActiveElement.Currents;   %Vector con 4 componentes. Nos interesa sólo la
       primera.
 inten_elect(j)= inten_electr(1);
Hidrogeno_p(j)=30*inten_elect(j)/F;        % Hidrógeno producido

DSSCircuit.SetActiveElement('Generator.FC');
inten_FC=DSSCircuit.ActiveElement.Currents;
 inten_fct(j)=inten_FC(1);
A_fc=1.4826; %Área efectiva de intercambio de la pila de combustible [m^2]
Hidrogeno_c(j)=80*inten_fct(j)/(2*F*A_fc); % Hidrógeno consumido

Hidrogeno_reserva=Hidrogeno_reserva+Hidrogeno_p(j)+Hidrogeno_c(j); % Hidrógeno almacenado
Hidro_vector(j)=Hidrogeno_reserva; % Hidrógeno almacenado en cada iteración.


%% Función para CURVAS DE EFICIENCIA de ELECTROLIZADOR y PILA DE COMBUSTIBLE
DSSCircuit.SetActiveElement('Line.Line5');
 Pot_bus_electrolizador =DSSCircuit.ActiveElement.Powers;
 Pot_bus_electrolizador ; %Vector con 4 componentes, nos interesa la última (es negativa = absorbe
       potencia)
 Pot_bus_electroliz(j)= Pot_bus_electrolizador(4); Potb_elec= Pot_bus_electrolizador(4);
[ Potencia_recibe_electrolizador ]=funcElec(Potb_elec);
 Potencia_recibe_electrolizadorv(j)= Potencia_recibe_electrolizador ;

DSSCircuit.SetActiveElement('Line.Line3');
Pot_FC_l=DSSCircuit.ActiveElement.Powers;
Pot_FC_lv(j)=Pot_FC_l(4); Pot_FCle=Pot_FC_l(4); %Vector con 4 componentes, nos interesa la primera (es
       negativa = cede potencia)
Potencia_PilaCombustible=funcFC(Pot_FCle);
Potencia_PilaCombustiblev(j)=Potencia_PilaCombustible; Potencia_PilaCombustible=1.2;
DSSCircuit.SetActiveElement('Storage.Battery');



%% ESTRATEGIA DE CONTROL: CONTROL POR HISTÉRESIS
StoragePowerMag(i)
    if SOC > 0.45*18.92 && SOC < 0.75*18.92

        disp('BATERÍA DENTRO DE LOS LÍMITES DE FUNCIONAMIENTO: 45%<SOC<75%')
```

```
        t=1;
    else
        t=2;
        disp('BATERÍA FUERA DE LÍMITES NORMALES DE FUNCIONAMIENTO')
    end
        if t==1;
            if SOC < 0.5*18.92 && SOC > 0.45*18.92
                situacion(j)=2;
                disp('Activar Pila de combustible para cargar batería: 45%<SOC<50%')
                DSSText.command = 'batchedit Generator.FC.* enabled=yes kv=0.048';
                DSSText.Command = ['Generator.FC.kW=' num2str(Potencia_PilaCombustible)];
                DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state=
                    charging'];
                DSSText.Command = ['Storage.Battery.state=charging'];
                DSSText.Command = ['Storage.Battery.%charge=15'];
                DSSText.Command = ['Storage.Battery.%reserve=' num2str(35)];
                DSSText.command = 'batchedit load.Electrolizador.* enabled=no';



                %DSSText.Command = ['Load.Electrolizador.kW=' num2str(Potencia_recibe_electrolizador)];
            end
            if SOC < 0.75*18.92 && SOC > 0.65*18.92
                disp('Activar Electrolizador: 70%<SOC<75%')
                situacion(j)=4;
                DSSText.command = 'batchedit Generator.FC.* enabled=no';
                DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state=
                    discharging'];
                DSSText.Command = ['Storage.Battery.state=discharging'];
                DSSText.Command = ['Storage.Battery.%reserve=' num2str(60)];
                DSSText.Command = ['Storage.Battery.%discharge=10'];
                DSSText.command = 'batchedit load.Electrolizador.* enabled=yes';
                DSSText.Command = ['Load.Electrolizador.kW=' num2str(Potencia_recibe_electrolizador)];
                DSSText.Command = ['Load.Carga.kW=' num2str(265)];

            end

            if SOC < 0.7*18.92 && SOC > 0.50*18.92
                disp('Zona de funcionamiento sin control: 50%<SOC<70%')
                situacion(j)=3;
                DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state=
                    IDLING'];
                DSSText.Command = ['Storage.Battery.state=IDLING'];
                DSSText.Command = ['Load.Carga.kW=' num2str(265)];
                DSSText.Command = ['Storage.Battery.%IdlingkW=0.01'];
                DSSText.command = 'batchedit Generator.FC.* enabled=no';
                DSSText.command = 'batchedit load.Electrolizador.* enabled=no';
            end

        end %Fin t=1

        if t==2;

        if SOC > 0.75*18.92
            situacion(j)=5;
        disp('Batería con exceso de carga. Electrolizador se activa')
        DSSText.command = 'batchedit Generator.FC.* enabled=no';
        DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state=discharging' '
            kv=0.048'];
        DSSText.Command = ['Storage.Battery.%discharge=25'];
        DSSText.Command = ['Storage.Battery.kwrated=0.5'];
        DSSText.Command = ['Storage.Battery.%reserve=' num2str(60)];
        DSSText.command = 'batchedit load.Electrolizador.* enabled=yes kv=0.048';
        DSSText.Command = ['Load.Electrolizador.kW=' num2str(Potencia_recibe_electrolizador)];
        DSSText.Command = ['Load.Carga.kW=' num2str(400)];
        end

    if SOC < 0.45*18.92
        disp('Batería con defecto de carga. Pila de combustible activada')
        situacion(j)=1;
            DSSText.command = 'batchedit Generator.FC.* enabled=yes';
```

```
                    DSSText.Command = ['Generator.FC.kW=' num2str(Potencia_PilaCombustible)];
                    DSSText.command = 'batchedit load. Electrolizador .* enabled=no';
                    DSSText.Command = ['Storage.Battery.kWhStored=' num2str(StoragePowerMag(i)) ' state =
                        charging' 'kv=0.048'];
                    DSSText.Command = ['Storage.Battery. state =charging' ];
                    DSSText.Command = ['Storage.Battery.%charge=50'];
                    DSSText.Command = ['Storage.Battery.%reserve=' num2str(30)];
                    DSSText.Command = ['Load.Carga.kW=' num2str(250)];
                end
            end %Fin de zona t=2
        end
Batt=StoragePowerMag(i)
iter = iter+1;

%% LECTURA DE MONITORES Y REPRESENTACIÓN GRÁFICA DE RESULTADOS
DSSText.Command = 'export mon vars';
    MonFileName = DSSText.Result;
    MyCSVb = csvread(MonFileName, 1, 0);
    kWh = MyCSVb(:,3);
     encab={'Hour','t(sec)','KWh','State','kWOut','kWIn','Losses','Idling','kWh'}
      variablesbat =MyCSVb;
      variablesbat ;


    figure(4);
    subplot(4,1,1)
    plot(Hour, MyCSVb(1:59:1380,3),'-k+',Hour, MyCSVb(1:59:1380,9),'-b+'); grid;
    legend('kWhStored','kWh');
    subplot(4,1,2) ;
    plot(Hour,MyCSVb(1:59:1380,4),'-r+');
    hold on
    subplot(4,1,3)
    %plot(Hour,energyStored2(1:59:1380)), grid
    plot(Min,energyStoredm), grid
    subplot(4,1,4)
    plot(Hour, situacion ), grid
    xlabel('Hour');

    hold off


    DSSText.Command = 'export mon Standard';
    MonFileName = DSSText.Result;
    MyCSVb = csvread(MonFileName, 1, 0);
    STANDAR=MyCSVb;


    % Monitor PVSystem
     DSSText.Command = 'export mon varsPV';
     MonFileName = DSSText.Result;
     MyCSV = csvread(MonFileName, 1, 0);
     variablesPV=MyCSV;
     PV_pt=variablesPV(1:59:1380,4);
     figure(5);
     plot(Hour,PV_pt,'linewidth',3), grid
     legend('Potencia suministrada por PV')
     xlabel('Horas'), ylabel('Potencia (kW)')

    % Monitor para Fuel Cell
    DSSText.Command = 'export mon FCm';
    MonFileName = DSSText.Result;
    MyFC = csvread(MonFileName,1,0);
    variablesFC=MyFC;
        end


        figure(6)
        plot(Hour,Dom,'g',Hour,0.5*lsPEV,'r','linewidth',2), grid
        xlabel('Horas'), ylabel('Demanda (kW)'), title('Demanda de la red'), legend('Demanda doméstica','Demanda
            coche eléctrico')
```

```
        figure(7)
        plot(Hour,Hidro_vector,'linewidth',2), grid, xlabel('Horas'), ylabel('Inventario de Hidrógeno (kg)'), title
            ('Cantidad de hidrógeno'), axis([0 25 0.35 0.6])


    DSSText.command='CalcVoltageBases';

%% TEST Y COMPROBACIONES DE OPENDSS
%           DSSText.Command = 'Show kVBaseMismatch'
%           DSSText.Command = 'Show mismatch'
%            DSSText.Command = 'Show Voltages LN Nodes'
%           DSSText.Command = 'show currents elem'
%            DSSText.Command = 'Show Voltage LN Elements'
%           DSSText.Command = 'Show Summary'
%           DSSText.command='Show Isolated'
%           DSSText.command='Show Elements'
%             DSSText.Command = 'Show Overloads'
%             DSSText.Command = 'Show Variables'

end

function [ Potencia_recibe_electrolizador ]=funcElec(Potb_elec)
% Tabla de datos experimentales para un Electrolizador de 1.5 kW.
% El instalado es de 3.2 kW, por lo que hay que escalar los valores.
Consigna=(3.2/1500)*[0 100 200 300 400 500 550 600 700 750 800 850 900 1500];
Rend=[0 0.582338983 0.644052402 0.647657817 0.647205357 0.642767025 0.632401619 0.633488095 0.624622846
      0.620519048 0.61574986 0.612655462 0.60960555 0.59];
p=polyfit(Consigna,Rend,length(Rend)-1);
 Potencia_recibe_electrolizador =polyval(p,Potb_elec)*Potb_elec;
end

function [Potencia_PilaCombustible]=funcFC(Pot_FCle)
% Tabla de datos experimentales para un Electrolizador de 1.5 kW.
% La curva de la pila de combustible es muy similar, pero con una potencia instalada de 1.2 kW, por lo que hay que
      escalar los valores.
Consigna=(1.2/1500)*[0 100 200 300 400 500 550 600 700 750 800 850 900 1000 1500 1800 2000 2250 2500 3000];
Rend=1.1*[0 0.582338983 0.644052402 0.647657817 0.647205357 0.642767025 0.632401619 0.633488095 0.624622846
      0.620519048 0.61574986 0.612655462 0.61060555 0.609960555 0.609 0.608 0.601 0.6005 0.6 0.61 ];
p=polyfit(Consigna,Rend,12);
t =[0:0.01:3];
Potencia_PilaCombustible=polyval(p,Pot_FCle)*Pot_FCle*1.2;
end


function [Start,Obj,Text] = DSSStartup(mydir)
    % Function for starting up the DSS
    % make sure we are in the proper directory
    cd(mydir);
    %
    % instantiate the DSS Object
    Obj = actxserver('OpenDSSEngine.DSS');
    %
    %Start the DSS. Only needs to be executed the first time w/in a
    %Matlab session
    Start = Obj.Start(0);
    % Define the text interface
    Text = Obj.Text;

end
function sa = ExtractMonitorData(mon, chan, base)
%ExtractMonitorData read OpenDSS binary monitor data into an array
%  mon is the interface from DSSMonitors (preselect by name or index)
%  chan is the channel number (0 to retrieve the independent variable)
%  base is a divisor for normalization
    n = mon.SampleCount;
    ba = mon.ByteStream;
    idata = typecast(ba(1:16),'int32');
    nrec = idata(3);
    mode = idata(4);
    [r c] = size(ba);
    sdata = typecast(ba(273:c),'single');
```

```
    y = reshape(sdata, nrec+2, n);
    if chan > 0   % normalized output value
        sa = y(chan+2,:) / base;
    else          % convert hour−sec to normalized time base
        sa = (3600*y(1,:) + y(2,:) ) / base;
    end
end
```

Bibliografía

[1] Iñigo Capellán-Pérez, Margarita Mediavilla, Carlos de Castro, Óscar Carpintero, and Luis Javier Miguel, *Agotamiento de los combustibles fósiles y escenarios socio-económicos: un enfoque integrado*, Ph.D. thesis, Universidad del Pais Vasco. Universidad de Valladolid., 2014, p. 62.

[2] P. Chirapongsananurak, S. Santoso, R. C. Dugan, and J. Smith, *Voltage regulation in distribution circuits with wind power*, 2012 IEEE Power and Energy Society General Meeting, IEEE, jul 2012, pp. 1–8.

[3] Roger Dugan, *What is Unique About OpenDSS?*, 2014, p. 1.

[4] Roger C Dugan, *Reference Guide The Open Distribution System Simulator (OpenDSS)*, (2013).

[5] EPRI, *OpenDSS Tutorial. Distribution System Simulator.*

[6] Electric Power Research Institute EPRI, *OpenDSS PVSystem Element Model*, 2011, p. 10.

[7] IEEE Power & Energy Society, *Distribution Test Feeders*.

[8] Insu Kim, Raeey Regassa, and Ronald G. Harley, *The modeling of distribution feeders enhanced by distributed generation in DIgSILENT*, 2015 IEEE 42nd Photovoltaic Specialist Conference (PVSC), IEEE, jun 2015, pp. 1–5.

[9] Organisation for Economic Co-operation and Development., *World energy outlook 2013*, International Energy Agency, 2013.

[10] Matthew Reno and Kyle Coogan, *Grid integrated distributed PV (GridPV).*, Tech. report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA (United States), aug 2013.

[11] Roger Dugan, *OpenDSS Introductory Training Level I.*

[12] Roger Dugan, EPRI, *OpenDSS Level 2 Training*, 2009.

[13] Trapezoidal Schedule, *TechNote StorageController Update*, 2015.

[14] Solomon S, Qin D, Manning M, Chen Z, Marquis M, Averyt K, Tignor MMB, and Miller HL, *IPCC (2007a) Climate Change 2007 — European Environment Agency*, Tech. report, European Environment Agency, 2007.

[15] Wencong Su, Habiballah Eichi, Wente Zeng, and Mo-Yuen Chow, *A Survey on the Electrification of Transportation in a Smart Grid Environment*, IEEE Transactions on Industrial Informatics **8** (2012), no. 1, 1–10.

[16] Wencong Su, Jianhui Wang, Kuilin Zhang, and Mo-Yuen Chow, *Framework for investigating the impact of PHEV charging on power distribution system and transportation network*, IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, IEEE, oct 2012, pp. 4735–4740.

[17] Wencong Su, Jianhui Wang, Kuilin Zhang, and Alex Q. Huang, *Model predictive control-based power dispatch for distribution system considering plug-in electric vehicle uncertainty*, Electric Power Systems Research **106** (2014), 29–35.

[18] L Valverde, F Rosa, A J Del Real, A Arce, and C Bordons, *Modeling, simulation and experimental set-up of a renewable hydrogen-based domestic microgrid*, International Journal of Hydrogen Energy **38** (2013), 11672–11684.

[19] Luis Valverde Isorna Tutor, Manuel Felipe Rosa Iglesias, and Luis Valverde Isorna, *Máster en Sistemas de Energía Térmica ESTUDIO SOBRE EL USO DE CONVERTIDORES DC/DC EN INSTALACIONES DE ALMACENAMIENTO DE ENERGÍA ELÉCTRICA DE ORIGEN RENOVABLE*.

[20] Juan A Martínez Velasco and Gerardo Guerra, *CONFERENCIA: Análisis y Diseño de Redes de Distribución Mediante Cálculo Paralelo.*, Universidad Politécnica de Catalunya.