



POLITÉCNICA

escuela técnica superior de
ingeniería
y diseño
industrial

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO
INDUSTRIAL**

**Grado en Ingeniería Electrónica Industrial y
Automática**

TRABAJO FIN DE GRADO

**Desarrollo de una interfaz gráfica en Matlab App
Designer para el simulador de sistemas fotovoltaicos
PVLite**

Autor: Marta Seisdedos Barrientos

Tutor:

Federico Javier Muñoz Cano

Departamento de Ingeniería
Eléctrica, Electrónica
Automática y Física Aplicada

Madrid, febrero, 2024

AGRADECIMIENTOS

A mi mamá, mi papá y mi ejemplo a seguir, mi hermano, las personas más importantes de mi vida. Por estar a mi lado y confiar en mí incondicionalmente. Por ayudarme a superar los momentos menos buenos y disfrutar conmigo los mejores. Sin vosotros no lo habría conseguido.

A Luci, por ayudar a que estos cuatro años hayan sido tan felices y menos duros, consiguiendo que siempre sienta su apoyo, y a todas mis amigas y amigos.

Y a mí, por mi esfuerzo y constancia.

ÍNDICE

AGRADECIMIENTOS.....	I
ÍNDICE.....	II
ÍNDICE DE FIGURAS	IV
Capítulo 1. Introducción.....	6
1.1. Resumen.....	6
1.2. Objetivos.....	8
1.3. Estado del arte	8
Capítulo 2. App Designer	14
2.1. Component library	15
2.2. Component Browser.....	18
2.3. Ventana de visualización de la interfaz.....	20
2.4. Herramientas.....	22
Capítulo 3. Desarrollo de la interfaz.....	24
3.1. Código implementado	24
3.1.1. Programación en App Designer.....	25
3.1.1.1. Explicación de cada componente en la Design View	25
3.1.1.2. Rangos permitidos para cada variable	40
3.1.1.3. Explicación del código de Code View	42
3.1.2. Programación en Matlab	44
3.1.3. Orden de ejecución del código	47
3.2. Funcionamiento de la interfaz	48
Capítulo 4. Conclusión y líneas futuras	62
4.1. Conclusión.....	62
4.2. Líneas futuras.....	63
BIBLIOGRAFÍA	64
ANEXO A. CÓDIGO IMPLEMENTADO EN APP DESIGNER.....	65
ANEXO B. CÓDIGO IMPLEMENTADO EN MATLAB.....	148
B.1. script newReadInputData	149

B.2. script PumpingModel	156
B.3. script pvlite	160
ANEXO C. TUTORIAL PARA AGREGAR O ELIMINAR COMPONENTES DE LA INTERFAZ	162
C.1. Eliminación de componentes	163
C.2. Creación de componentes	165

ÍNDICE DE FIGURAS

Ilustración 1. Interfaz del software PVsyst [2]	9
Ilustración 2. Interfaz del software Homer [4].....	10
Ilustración 3. Interfaz del software SAM [6].....	11
Ilustración 4. Interfaz del software PVWatts.....	13
Ilustración 5. Ventana principal en App Designer al crear una nueva aplicación en blanco.....	15
Ilustración 6. Component Library en App Designer	16
Ilustración 7. Component Browser (características) en App Designer	18
Ilustración 8. Component Browser (Callbacks) en App Designer.....	19
Ilustración 9. Design View en App Designer.....	21
Ilustración 10. Code View en App Designer	22
Ilustración 11. Herramientas disponibles en DESIGNER.....	22
Ilustración 12. Herramientas disponibles en CANVAS.....	23
Ilustración 13. Herramientas disponibles en EDITOR.....	23
Ilustración 14. Flujo de información entre App Designer y Matlab	25
Ilustración 15. Tab Group Site en Design View	26
Ilustración 16. Tab Group Meteo en Design View	27
Ilustración 17. Tab Group PVgen (1) en Design View	28
Ilustración 18. Tab Group PVgen (2) en Design View	29
Ilustración 19. Tab Group Inverter en Design View.....	30
Ilustración 20. Tab Group Wiring en Design View	31
Ilustración 21. Tab Group Battery en Design View.....	32
Ilustración 22. Tab Group Load en Design View	33
Ilustración 23. Tab Group GenSet en Design View	34
Ilustración 24. Tab Group Wind en Design View	35
Ilustración 25. Tab Group Pumping (1) en Design View.....	36
Ilustración 26. Esquema de las variables relacionadas con el Bombeo de agua	36
Ilustración 27. Tab Group Pumping (2) en Design View.....	37
Ilustración 28. Tab Group Pumping (3) en Design View.....	38
Ilustración 29. Tab Group Pumping (4) en Design View.....	39
Ilustración 30. Tab Group Options en Design View.....	40
Ilustración 31. Comando para abrir la interfaz.....	49
Ilustración 32. Interfaz funcionando: selección del proyecto a cargar	50
Ilustración 33. Interfaz en funcionamiento en la pestaña Site	50
Ilustración 34. Interfaz en funcionamiento en la pestaña Meteo	51
Ilustración 35. Interfaz en funcionamiento en la pestaña PVgen (1)	51
Ilustración 36. Interfaz en funcionamiento en la pestaña PVgen (2).....	52
Ilustración 37. Interfaz en funcionamiento en la pestaña Inverter.....	52
Ilustración 38. Interfaz en funcionamiento en la pestaña Wiring	53

Ilustración 39. Interfaz en funcionamiento en la pestaña Battery.....	53
Ilustración 40. Interfaz en funcionamiento en la pestaña Load	54
Ilustración 41. Interfaz en funcionamiento en la pestaña GenSet	54
Ilustración 42. Interfaz en funcionamiento en la pestaña Wind	55
Ilustración 43. Interfaz en funcionamiento en la pestaña Pumping (1).....	55
Ilustración 44. Interfaz en funcionamiento en la pestaña Pumping (2).....	56
Ilustración 45. Interfaz en funcionamiento en la pestaña Pumping (3).....	56
Ilustración 46. Interfaz en funcionamiento en la pestaña Pumping (4).....	57
Ilustración 47. Interfaz en funcionamiento en la pestaña Options.....	57
Ilustración 48. Error al introducir un valor fuera del rango permitido.....	58
Ilustración 49. Resultados de la simulación para determinados parámetros, aplicación tipo Stand Alone	59
Ilustración 50. Modificaciones del nuevo Proyecto (1)	60
Ilustración 51. Modificaciones del nuevo Proyecto (2)	60
Ilustración 52. Modificaciones del nuevo Proyecto (3)	61
Ilustración 53. Resultados para el nuevo proyecto (Water Pumping)	61

Capítulo 1. INTRODUCCIÓN

1.1. RESUMEN

La gran popularidad de los sistemas fotovoltaicos, no solo en los ambientes profesionales tecnológicos y de investigación sino también en nuestro día a día, hace que cada día haya más interés en el estudio de este tipo de sistemas. La posibilidad de poder atrapar la energía solar para su posterior reutilización en tareas cotidianas como puede ser mantener caliente el hogar con sistemas eléctricos que deben ser muy eficientes, como la aerotermia, es algo que, cada vez será más común para toda la población.

Para hacer esto posible y poder instalar un sistema fotovoltaico en cualquier lugar es esencial un estudio previo a partir de los datos meteorológicos, el tipo de aplicación y las características de los componentes que forman el sistema fotovoltaico. Por ejemplo, el tipo de aplicación del sistema que se desea instalar, la situación geográfica del lugar, la radiación solar y temperatura ambiente, las características del inversor del sistema, los rangos de potencias con los que pretende trabajar o las pérdidas. Es decir, cada sistema fotovoltaico será único y tendrá unas características muy específicas que se deben obtener previamente a cualquier montaje con la intención de reducir tiempos de trabajo y costos.

Existen diversos programas que permiten simular aplicaciones fotovoltaicas, tanto comerciales como libres.

Este trabajo parte de un programa de simulación ya desarrollado llamado *PVLite*, el cual permite simular diversas aplicaciones fotovoltaicas, como la conexión a la red o sistemas autónomos.

Para introducir los datos en *PVLite*, el programa utiliza una hoja Excel en la que se encuentran todos los parámetros necesarios para el estudio del sistema y la obtención de una serie de gráficas con las características principales de lo que sería ese sistema fotovoltaico en específico. El programa funciona correctamente, pero a la hora de valorar y comparar distintos sistemas con distintas características resulta muy lento y complicado ir modificando variable a variable en esta hoja Excel. Por ejemplo, hay que cerrar la hoja Excel del proyecto cada vez que se ejecuta una simulación y hay que volver a abrirla para modificar los datos de entrada.

Tras la valoración de las distintas opciones se concluyó con que la mejor solución para este problema era la implementación de una interfaz gráfica de usuario (*GUI, Graphical User Interface*). Una interfaz gráfica de usuario es un programa que sirve como conexión para la comunicación entre el hombre y la máquina. La principal característica de este tipo de interfaces es la presencia de imágenes y elementos muy visuales que hacen que esta comunicación sea muy sencilla, rápida e intuitiva. Es por ello, por lo que cada vez hay más interfaces gráficas de usuario en los elementos electrónicos de nuestro día a día.

En este trabajo se ha diseñado y programado una interfaz gráfica de usuario implementada con la ayuda de la herramienta App Designer de Matlab para el programa de simulación *PVLite*. Esta interfaz debe permitir la modificación de las variables de una manera sencilla, ser compatible con el código fuente del programa *PVLite* para que, tras la introducción de los parámetros se lance la simulación y se obtengan los resultados correctamente, y guardar y cargar proyectos tanto nuevos como los que estén previamente creados, además de cargar tantos proyectos como se quiera sin la necesidad de cerrar y volver a abrir la interfaz. Esta opción permitirá un gran ahorro de tiempo al usuario ya que podrá crear tantos proyectos como desee, con el nombre que considere y los parámetros que establezca en cada caso. También podrá guardar todos estos proyectos y, por lo tanto, cargarlos en la interfaz cuando desee para continuar con el estudio del sistema justo en el punto donde lo dejó.

1.2. OBJETIVOS

El objetivo de este proyecto es el diseño y la programación de una interfaz gráfica intuitiva que facilite la conexión entre el usuario y el programa de simulación de sistemas fotovoltaicos *PVLite*. Este simulador permite simular varias aplicaciones fotovoltaicas, como la conexión a la red eléctrica, con o sin autoconsumo, sistemas autónomos con baterías y sistemas de bombeo de agua. La interfaz permitirá la introducción y modificación de los parámetros necesarios para simular proyectos en cada tipo de aplicación y mostrarlos en forma de gráficos. Además, se implementará la funcionalidad de guardar, crear y cargar proyectos en la interfaz para poder continuar con el estudio del sistema cuando se desee sin perder toda la información. La función de cargar proyectos previamente guardados en la interfaz tantas veces como se desee, permitirá al usuario la opción de obtener distintas gráficas para distintos sistemas, y con ellas, comparar resultados valorando diferencias y similitudes en ellos. El programa concluirá con el lanzamiento de la simulación y, por lo tanto, la obtención de resultados del sistema que se pretende diseñar, caracterizado por los parámetros específicos para dicho sistema.

1.3. ESTADO DEL ARTE

Debido al gran desarrollo de los sistemas fotovoltaicos, hoy en día ya existe una gran variedad de programas dedicados a la simulación de dichos sistemas. Estos programas existentes pueden ser tanto comerciales como gratuitos, y como todo programa que necesita una comunicación hombre-máquina, cada uno de ellos tiene su respectiva interfaz gráfica de usuario. A continuación, se explican los más importantes.

PVsyst [1]

PVsyst es un software comercial que tiene como finalidad el estudio y diseño de sistemas fotovoltaicos, realizando los cálculos pertinentes a partir de los valores de entrada introducidos por el usuario. Los resultados mostrados por este programa son el cálculo de la producción eléctrica, el análisis de pérdidas y el análisis económico.

La interfaz de este programa, que permite la conexión entre hombre-máquina, se compone de diversos elementos que la caracterizan como visual y sencilla de utilizar. Estos elementos son principalmente botones que, al pulsarlos, abren nuevas ventanas donde se explican los fundamentos de cada parámetro de entrada y la forma correcta

de rellenarlo. Además, prácticamente por cada ventana y apartado, aparecen gráficas o dibujos que permiten una mayor facilidad a la hora de entender el funcionamiento de la interfaz.

Al abrir el programa, aparece una ventana principal en la que hay dos apartados muy diferenciados. Uno de ellos está enfocado a las herramientas del proyecto que se va a crear (nuevo proyecto, cargar proyecto anterior, borrar proyecto, guardar, características del proyecto como el nombre, etc.) y otra, más enfocada a la introducción de valores de entrada para el cálculo. En este apartado, existen varios campos que se organizan en tres pestañas: parámetros principales, parámetros opcionales y simulación (se puede visualizar en la Ilustración 1). Al pinchar sobre alguno de los parámetros, se abre una nueva ventana en la que el usuario deberá introducir los valores necesarios para el estudio de su sistema. Tras introducir todos los valores, el usuario clicará sobre el botón de *run simulation* y automáticamente, el programa ejecutará los comandos necesarios para la obtención de las gráficas resultantes. Este sistema de simulación requiere el uso de bases de datos como la localización donde se va a ubicar el sistema fotovoltaico en la tierra. La configuración de estas bases de datos se realiza de la misma forma que la introducción de los valores de entrada.

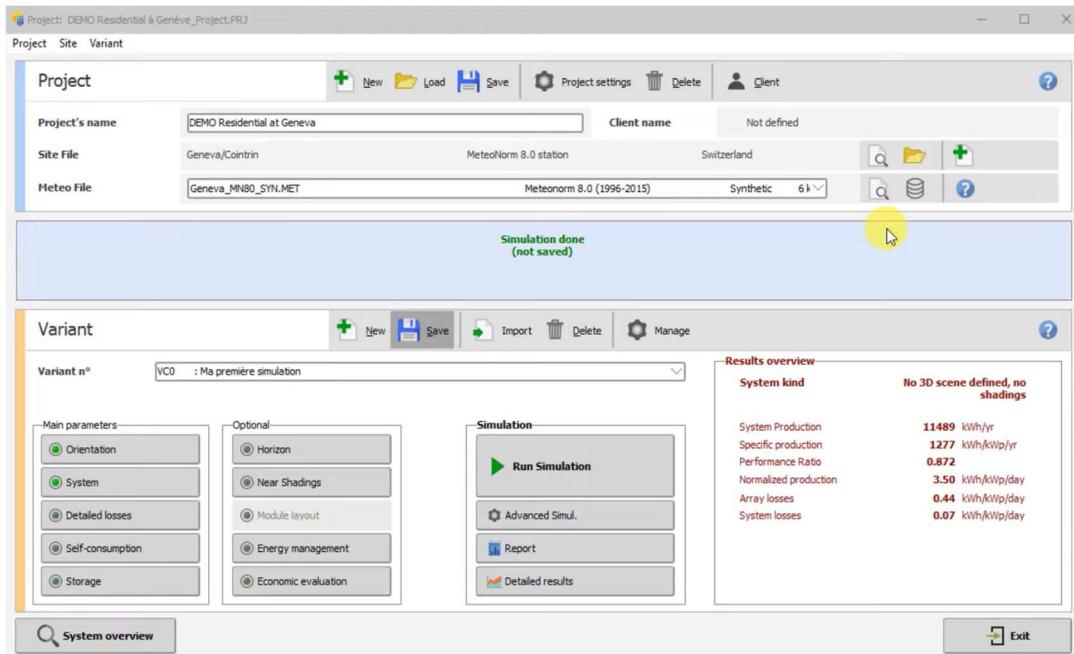


Ilustración 1. Interfaz del software PVsyst [2]

Capítulo 1

HOMER [3]

HOMER es otro programa comercial destinado a la simulación de sistemas fotovoltaicos. Su principal finalidad es garantizar un estudio profesional y real considerando las características propias de cada caso de simulación dependiendo de la ubicación, el precio del combustible, los cambios climáticos y todos los factores que puedan afectar al sistema que se va a diseñar.

En cuanto a la interfaz, también es considerada muy visual, compuesta por esquemas, dibujos y gráficas que permiten una mayor sencillez. En la ventana principal, en la parte izquierda, aparece un esquema con flechas e iconos. Al pulsar estos iconos, se abre una nueva ventana con la explicación de cada componente de dicho ícono, y la posibilidad de modificar los valores de entrada predeterminados. Algunos de estos iconos son las entradas de carga (con sus valores mensuales, cada hora, día a día, medias, máximos, gráficas que muestran todos los datos, etc.), los valores de entrada para un sistema PV (como los costos en \$ en función de las dimensiones en kW, el tipo de corriente, tiempo de vida, reflectancia del suelo, etc.), los parámetros de entrada para una turbina de viento (con su respectiva curva de potencia vs velocidad del viento, los costos vs la cantidad, etc.), las entradas del convertidor de potencia (también con su curva de costos, tiempo de vida y eficiencia), las entradas para un sistema fotovoltaico tipo Grid (expresando también una gráfica que muestra sus tasas en función de la hora del día y el mes del año), y los parámetros solares (como la latitud, altitud, zona horaria, datos base, media, etc.). Como se ha mencionado, en cada apartado de los numerados anteriormente, se muestran gráficas, y dibujos que representan visualmente todos los datos de entrada introducidos. Se muestra a continuación, en la Ilustración 2, la interfaz de este software a la hora de introducir las entradas de carga al sistema.

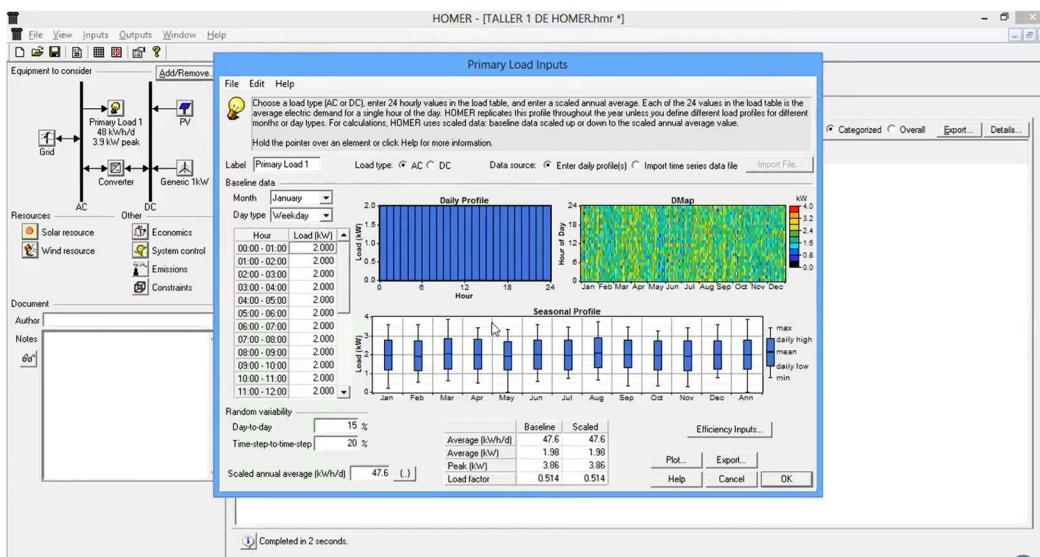


Ilustración 2. Interfaz del software Homer [4]

Como resultado, al simular el programa, el software calcula y diseña el sistema fotovoltaico y muestra los resultados en forma de valores numéricos y gráficas representativas para cada resultado.

SAM: System Advisor Model [5]

SAM es un programa gratuito que ofrece una gran ayuda a los profesionales que se dedican a la industria de energías renovables. Este software puede realizar estudios de distintos tipos de sistemas de energías renovables, pero únicamente vamos a centrarnos en los casos de sistemas fotovoltaicos.

El funcionamiento de la interfaz para este programa es similar al descrito en los casos a anteriores. Al crear un nuevo proyecto se abre una nueva ventana en la cual, en la parte izquierda, aparece un menú con distintos apartados. Al pulsar cada uno de los botones de dichos apartados, se abre en la parte derecha de la pantalla una nueva ventana en la que modificar los valores. Estos valores se pueden modificar bien numéricamente, pulsando sobre *check lists*, o desplegando menús y seleccionando una de las opciones disponibles, tiene diversidad de opciones a la hora de introducir los datos de entrada. También tiene gráficas que muestran los datos más importantes. Los apartados en los que se divide este programa son la localización y recursos, módulo, inversor, diseño del sistema, sombreado y trazado, pérdidas y límites de la cuadrícula. En la Ilustración 3 se observa la interfaz en el momento de introducción de datos de entrada relacionados con el diseño del sistema.

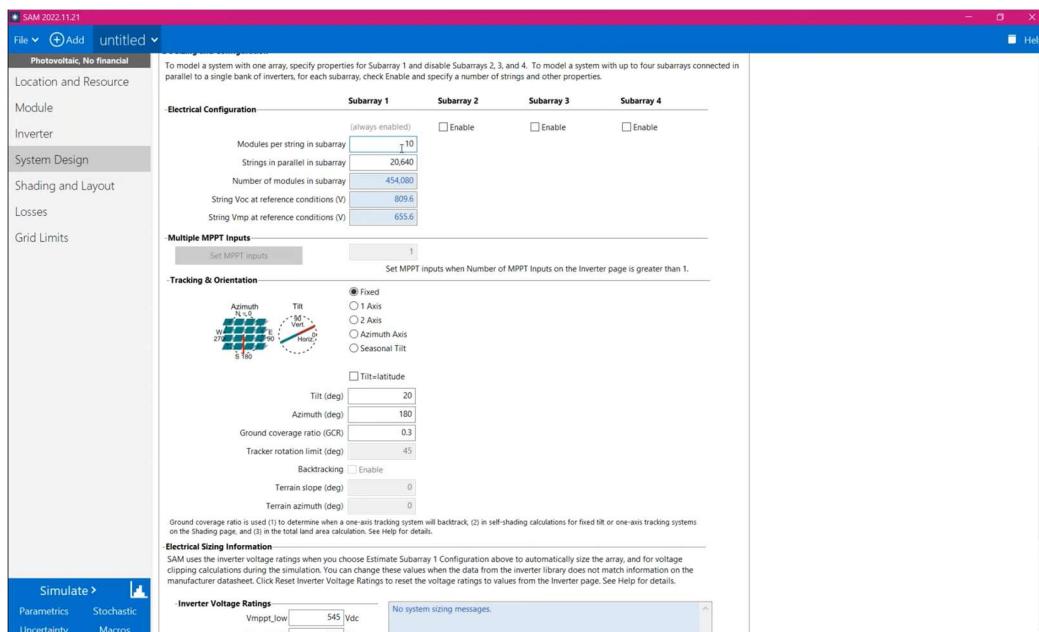


Ilustración 3. Interfaz del software SAM [6]

Capítulo 1

Una vez editados los valores necesarios, se simula el programa y como resultado aparecen gráficas y resultados numéricos divididos en secciones. La primera y más importante de las secciones es un resumen con los resultados más característicos del sistema. Otras de las secciones más representativas son las tablas de datos, las pérdidas representadas en un gráfico de flechas, los perfiles, estadísticas o mapas de calor.

PVWatts [7]

PVWatts es otro programa gratuito que realiza un estudio de la producción de energía fotovoltaica de sistemas conectados a la red, dando como resultados todos los cálculos y valores necesarios para la creación de un sistema fotovoltaico. Tiene también la opción de imprimir los resultados, tanto mensuales como resultados por hora.

Centrándonos en la interfaz, la manera de acceder a este programa de simulación es muy sencilla. En primer lugar, se pide la ubicación exacta del lugar donde se pretende instalar el sistema fotovoltaico. Al introducir la dirección, se avanza a la siguiente ventana en la que se deben llenar los datos de recursos solares. En esta pestaña, se muestra la longitud y latitud del sitio de datos de recursos solares y un mapa de todos estos recursos. El programa automáticamente asigna una cuadrícula en la que se encuentra la ubicación introducida anteriormente rodeada por los recursos más cercanos, pero si se desea, se pueden cambiar estos recursos, modificando así la cuadrícula asignada. A continuación, se procede a la información de sistema donde se asigna valores iniciales a cada parámetro y en caso de ser necesario, se pide al usuario cambiarlos. Cada campo tiene una opción de información donde se describe la función de cada parámetro. También existe un botón que permite estimar las dimensiones del área de tejado del sistema en un mapa. Una vez llenados todos los campos, se pulsa sobre el botón *PVWATTS Resultados* y entonces el programa realiza la simulación y con ella los cálculos y la obtención de resultados. Se expone en la Ilustración 4 la interfaz de este software a la hora de introducir los datos de información del sistema.

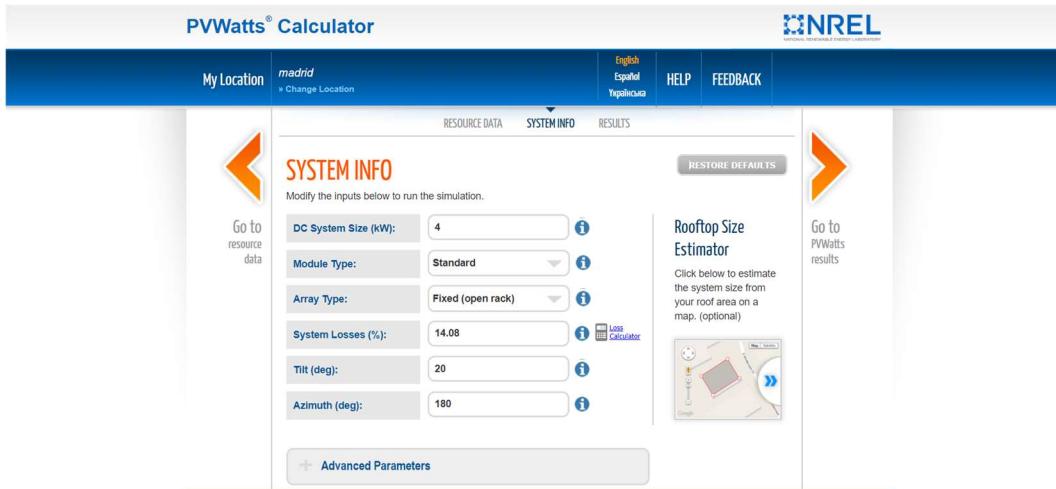


Ilustración 4. Interfaz del software PVWatts

PVLIB [8]

El último software descrito es el programa gratuito PVLIB Python, el cual tiene una gran relación con Matlab. Este software simula el rendimiento de los sistemas fotovoltaicos con la ayuda de una serie de funciones y clases desarrolladas en Matlab.

En este caso la comunicación entre el usuario y la máquina no es tan sencilla, ya que, al tratarse de un código implementado en Matlab, no existen botones o elementos visuales que permitan entender fácilmente cómo introducir cada valor de entrada al sistema. Como se ha dicho anteriormente, este software se basa en funciones y clases de Matlab, por lo que, para su funcionamiento se debe llamar a la función que se deseé en cada caso. Para modificar los valores se llama a través de la línea de comandos a la variable que contenga el valor que se deseé cambiar, y se cambia de la misma forma que se cambia cualquier variable en el espacio de trabajo de Matlab. A continuación, se guarda dicha variable y se sigue el mismo procedimiento para cada modificación. Posteriormente, se llama a nuevas funciones para que el programa se ejecute y se puedan obtener los resultados deseados. Este método de modificación de variables, además de ser más lento, es más dificultoso, por lo que se debe tener un conocimiento mínimo en Matlab para poder usar esta herramienta. La obtención de resultados es más parecida al resto, ya que, una vez modificados todos los valores, se ejecutan las funciones necesarias y finalmente se muestran las gráficas e imágenes con los resultados a la simulación del sistema, al igual que en los otros softwares de simulación descritos.

Capítulo 2. APP DESIGNER

App Designer es un entorno de desarrollo interactivo creado por Matlab que permite diseñar y programar una aplicación con interfaces gráficas de usuario (GUI). Para su correcto funcionamiento, necesita tener instalado el compilador de Matlab y en dicho compilador, la extensión de Matlab App Designer. [9]

Para acceder a App Designer (teniendo todos los paquetes necesarios instalados) es tan sencillo como abrir Matlab y en la ventana de comandos escribir “*appdesigner*”. Al introducir este comando, se abrirá la ventana principal de App Designer donde se da la opción, entre otras, de crear a una nueva aplicación en blanco, o abrir una ya creada para seguir editándola o para ejecutarla.

A continuación, se muestra en la Ilustración 5 lo que sería una aplicación vacía, preparada para comenzar a programar la interfaz deseada.

Como se puede observar, la pantalla se divide en una sección superior donde se encuentran las herramientas y otra inferior que se divide a su vez en tres ventanas muy diferenciadas. La de la izquierda es la *Component Library* (librería de componentes), en la que se muestran todos los componentes que se pueden añadir a la interfaz. La sección del medio sería la interfaz que se está creando que se divide en dos. Por un lado, está la *Design View* (vista de diseño), que es la visualización de lo que sería la aplicación y, por otro lado, la *Code View* (vista de código), es decir, el código de la aplicación. Se pueden alternar ambas vistas como se deseé. Por último, en la sección de la derecha, llamada *Component Browser* (explorador de componentes), se mostrarán todos los

componentes añadidos a la interfaz y pulsando sobre uno en específico, se podrán visualizar, más abajo, sus características.

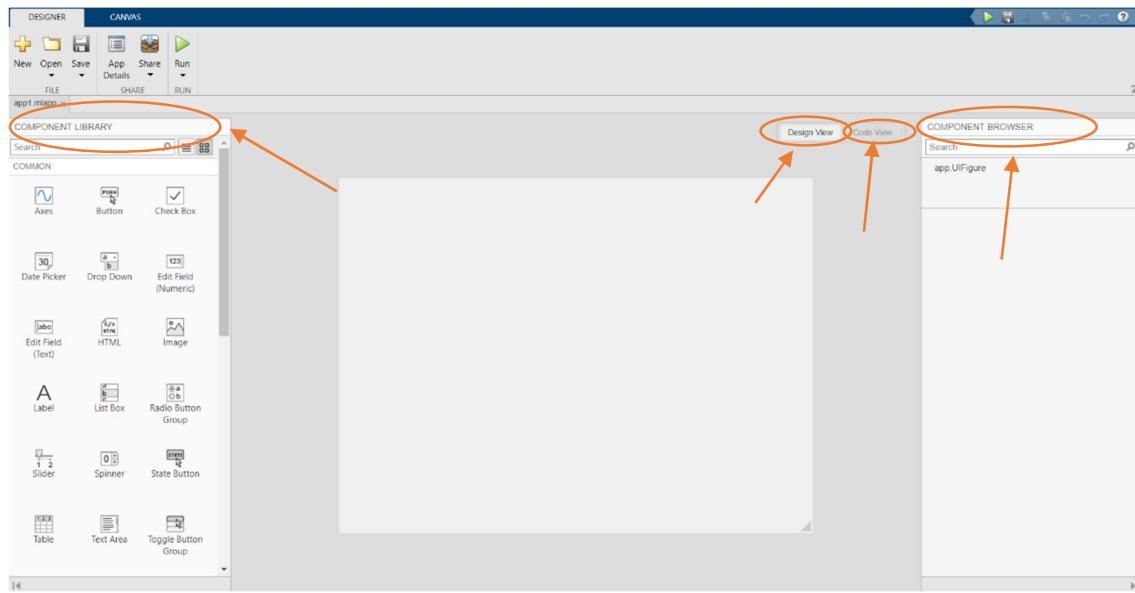


Ilustración 5. Ventana principal en App Designer al crear una nueva aplicación en blanco

Seguidamente, se van a describir más detalladamente estas tres secciones principales.

2.1. COMPONENT LIBRARY

La *Component Library*, representada en la Ilustración 6, está formada por todos los componentes existentes en App Designer que se pueden añadir a la interfaz.

Capítulo 2

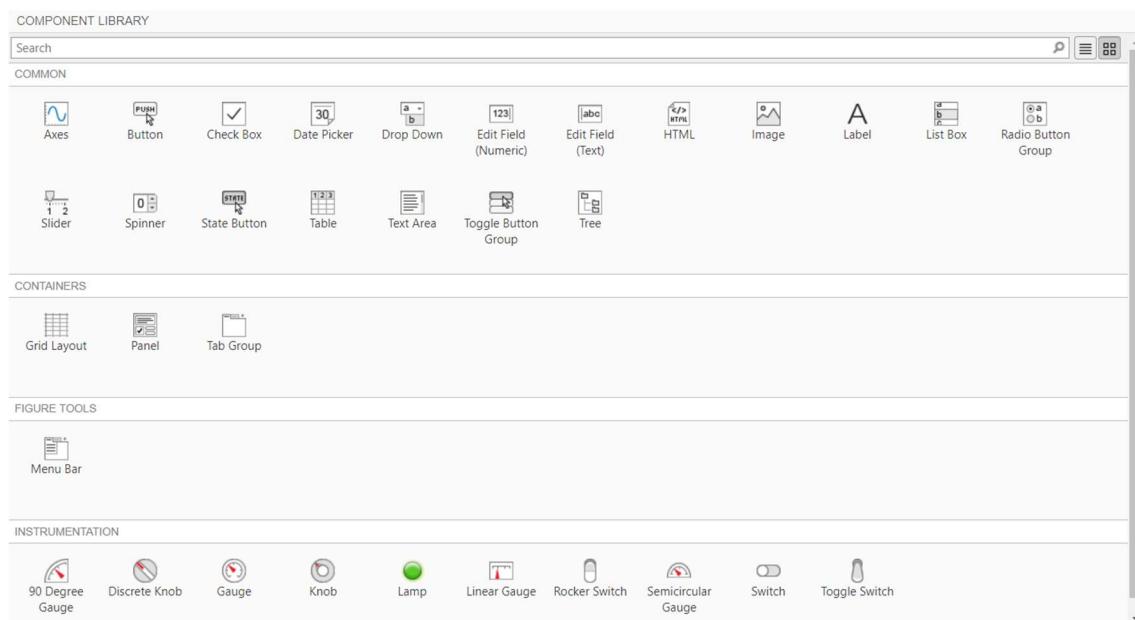


Ilustración 6. Component Library en App Designer

Como se puede comprobar, la variedad de componentes en esta plataforma es bastante amplia, lo que permite la obtención de una interfaz más elaborada y con la posibilidad de una estética mejorada. A pesar de ser tantos los elementos que existen, es cierto que hay muchos que comparten la misma funcionalidad, únicamente se distinguen en la estética. Para un mejor entendimiento de ellos, se muestra en la *Tabla 1: Tipos de componentes y sus funcionalidades*, la funcionalidad de los componentes más importantes y utilizados.

Tabla 1. Tipos de componentes y sus funcionalidades

COMPONENTE	FUNCIONALIDAD
Axes	Visualización de gráficas 2D. Permite visualizar datos introducidos por el usuario o datos ya predefinidos en el código. Se puede actualizar la gráfica en cualquier momento.
Button	Ejecución de una función al pulsar el botón. El botón no se mantiene activado al dejar de presionarlo.
<i>Check Box, State Button, Rocker Switch, Switch, Toggle Switch</i>	Selección de un estado excluyente de otro. Cuando está seleccionado, el programa tiene un comportamiento y cuando no, otro distinto.
Date Picker	Selección de fechas en un calendario interactivo.
<i>Drop Down, List Box, Radio Button Group, Toggle Button Group</i>	Selección de un estado excluyente de los demás. Se establece un listado con todas las opciones de selección y al seleccionar una de ellas, se excluyen las demás. Debe haber mínimo dos opciones de selección.
Edit Field (Numeric), Spinner	Introducción de datos numéricos por el usuario.
Edit Field (Text), Text Area	Introducción de información en forma de texto por el usuario.
HTML	Abrir en una nueva ventana una web con un enlace establecido.
Image	Visualización de una imagen.
Label	Etiqueta con texto (muy útil para instrucciones). El texto lo define el programador, no es editable por el usuario.
Slider	Introducción de datos numéricos por el usuario seleccionando entre un rango y valores establecidos.
Table	Visualización de datos con posibilidad de edición por parte del usuario, de tal manera que se tomen como valores de entrada a la aplicación. Los datos de las tablas pueden ser tanto numéricos como en forma de texto.
<i>90 Degree Gauge, Gauge, Linear Gauge, Semicircular Gauge</i>	Visualización de la medida de valores numéricos.
Lamp	Visualización de un estado.
Panel	Agrupación de varios componentes en un panel.
Tab Group	Agrupación de varios paneles, permitiendo visualizar uno u otro en función de la conveniencia del usuario.

2.2. COMPONENT BROWSER

En esta sección se listan todos los componentes que forman la aplicación con su nombre formado por el prefijo “app.” y seguido del nombre que le dé el programador a cada uno de ellos.

Si se pulsa sobre uno de los componentes, en la parte posterior al listado, se muestran las características y *Callbacks* (retrolllamadas) de dicho componente. Las *Callbacks* son funciones propias de cada componente que se ejecutan cuando el usuario interacciona con dicho componente

Se muestra un ejemplo en la Ilustración 7, en el que se visualizan las características de un *spinner* (componente dedicado a la introducción de un valor numérico en la interfaz), y a continuación, en la Ilustración 8, sus *Callbacks*.

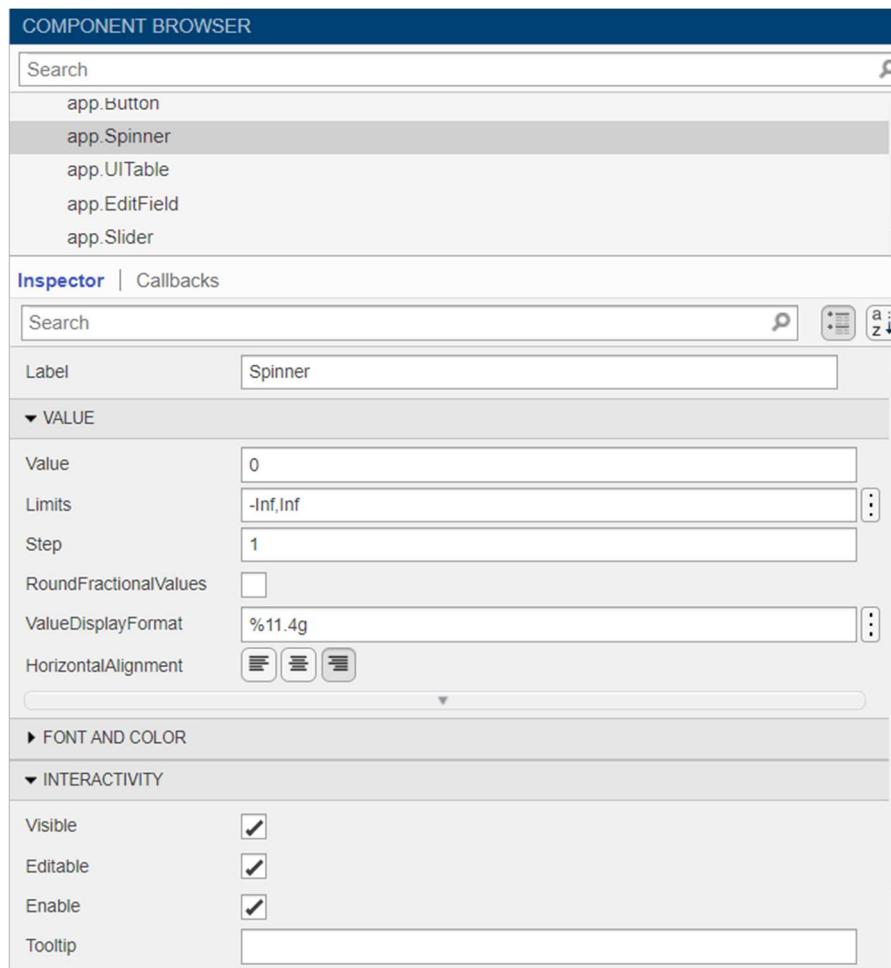


Ilustración 7. Component Browser (características) en App Designer

Estas características son modificables para cada caso, y las de cada tipo de componente son específicas y diferentes, ya que, como se ha dicho en el apartado anterior, cada componente tiene una funcionalidad muy distinta a los demás.

La primera parte de las características es donde se define el texto que se va a visualizar en la aplicación y su posición, principalmente. A continuación, en caso de que el componente lo requiera, se establecen los límites superiores e inferiores, los valores predefinidos en caso de desearlos, las marcas de los ejes horizontales y verticales, etc. Posteriormente se define la apariencia del componente, es decir, el tipo de letra, el color, si es cursiva o es negrita, etc. Por último, se define la interactividad, es decir, si el componente es visible, editable y está activado.

Todos los parámetros mencionados se deben definir manualmente por el programador siempre que se cree un nuevo componente ya que son los básicos e imprescindibles, pero hay otros tantos parámetros más específicos que se pueden definir si el programador lo desea.

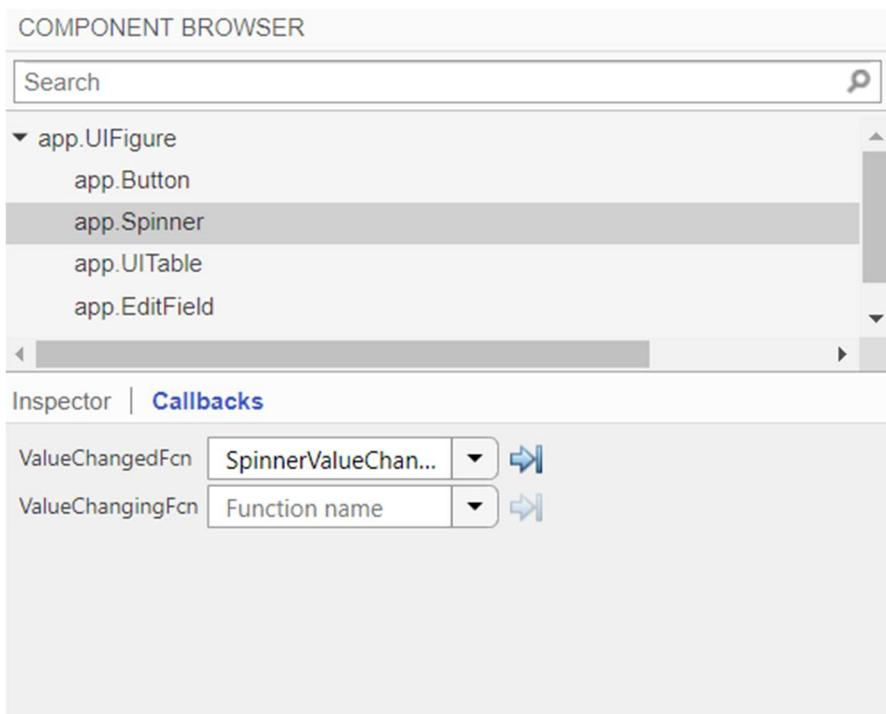


Ilustración 8. Component Browser (Callbacks) en App Designer

Las *Callbacks* se pueden añadir al código o no, es elección del programador en función del comportamiento que quiera para cada componente. En caso de querer definir una *Callback*, primero se debe añadir al código (ya sea a través de esta ventana, la *Component Browser*, o a través de otras ventanas como la *Design View* por ejemplo, se pueden tomar distintos caminos, todos ellos igual de válidos).

Capítulo 2

Cada tipo de componente tendrá unas *Callbacks* establecidas diferentes, ya que la interacción del usuario con los componentes es distinta si se habla de un *Spinner*, o de una *Table* (Tabla, componente dedicado a la visualización de una matriz de valores con posibilidad de edición por parte del usuario), por ejemplo.

2.3. VENTANA DE VISUALIZACIÓN DE LA INTERFAZ

En la sección intermedia de la ventana principal que engloba todo, se puede visualizar la interfaz de dos maneras muy distintas. Por un lado, existe la *Design View* en la que se visualiza de manera gráfica la interfaz con todos los componentes que forman parte de ella. Esta sería la visualización que tendría el usuario al compilar la aplicación y ejecutarla. Por otro lado, está la *Code View*, que solo podrá ver el programador. El usuario no tendrá acceso a ella ya que es la parte en la que se implementa todo el código que define el comportamiento de la aplicación.

Desde el punto de vista del programador, ambas visualizaciones son importantes, ya que este debe interactuar con todas ellas y, por tanto, conocer el funcionamiento de ambas.

Para la creación y definición de un nuevo componente, se usará en primer lugar la *Design View*, representada en la Ilustración 9. Si se quiere añadir un componente a la interfaz, el programador arrastrará el ícono del componente que se desea añadir desde la ventana de *Component Library* hasta el cuadrado de lo que sería la interfaz, es decir, la *Design View*. Una vez arrastrado a este cuadro, este componente se añadirá automáticamente a la interfaz y, por lo tanto, aparecerá en el listado de componentes de la ventana *Component Browser*. Desde la *Design View*, se puede mover el componente hasta que quede en el lugar deseado, eliminar, duplicar, agrupar con otros componentes y añadir una *Callback*.

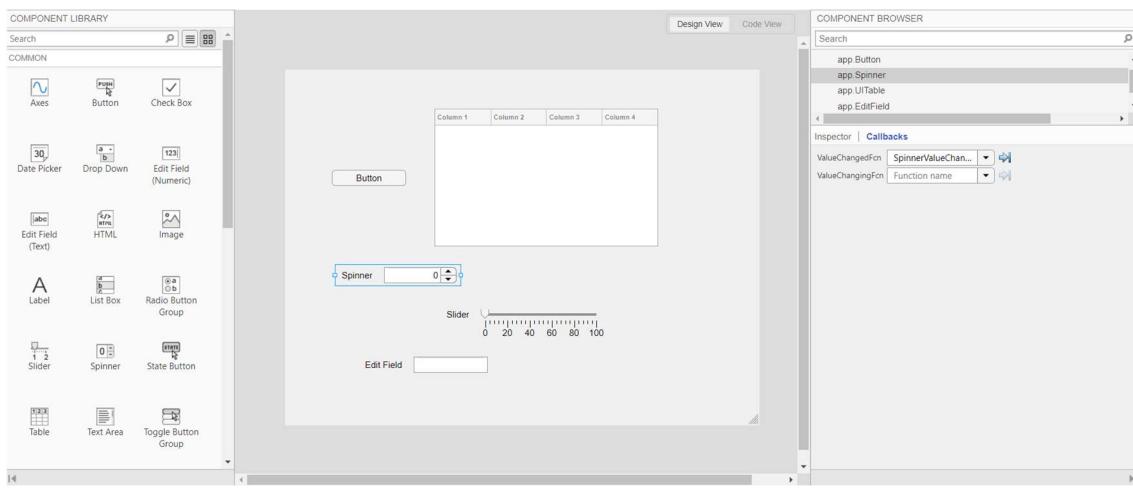


Ilustración 9. Design View en App Designer

Una vez hecho esto, el programador, deberá ir a la *Code View*. En esta vista desaparece la ventana de *Component Library* y en su lugar aparecen dos nuevas ventanas. La primera, llamada *Code Browser*, donde se listan todas las *Callbacks*, funciones y propiedades de la aplicación, y la segunda, llamada *App Layout*, donde se visualiza la interfaz en pequeño, básicamente lo que sería la *Design View*. Más abajo, se muestra en la Ilustración 10 la indicación de cada apartado en esta *Code View*.

En esta *Code View* aparece todo el código de la aplicación, pero este código se diferencia en dos tipos, uno que no es editable, y otro que sí. El código no editable (el mostrado con fondo gris clarito) es el que genera App Designer automáticamente al añadir un componente a la aplicación. Este código es el necesario para añadir los componentes y definir sus características iniciales. Por otro lado, existe el código editable por el programador (el mostrado con fondo blanco) que se compone de todas las *Callbacks* y funciones que debe implementar dicho programador para definir el comportamiento de la interfaz. Por ejemplo, al añadir la *Callback* a través de la *Design View*, automáticamente se cambiará a la *Code View*, donde se ha creado dicha *Callback* vacía. Es ahí, donde el programador llenará la *Callback* con el código necesario para que la interfaz actúe como se desea.

El código se estructura en distintas partes, una primera donde se implementan las propiedades (componentes de la interfaz y propiedades tanto públicas como privadas añadidas por el programador para su uso en las funciones posteriores del código), y en otra parte que la sigue donde se definen los métodos, es decir todas las funciones y *Callbacks* del código, tanto públicas como privadas.

Capítulo 2

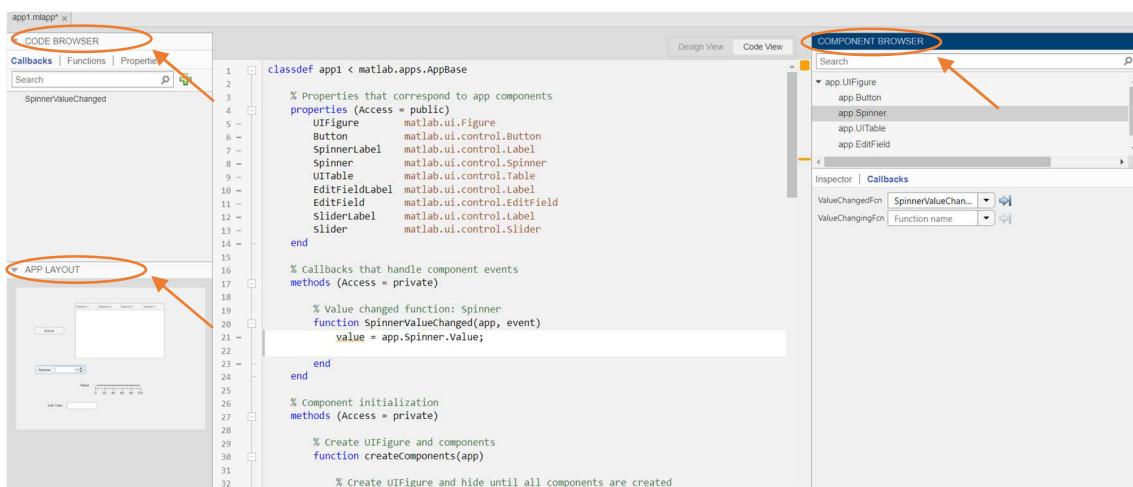


Ilustración 10. Code View en App Designer

2.4. HERRAMIENTAS

Por último, además de estas tres secciones descritas, existe un apartado en la parte superior en el que se encuentran todas las herramientas necesarias para facilitar la escritura del código y la simulación y ejecución del programa. Esta sección se compone de dos partes, una llamada *DESIGNER* que es común para ambas vistas (*Design View* y *Code View*), y otra que cambia en función de la vista en la que nos encontramos, siendo *CANVAS* para la *Design View*, y *EDITOR* para la *Code View*.

En el apartado de *DESIGNER* (expuesto en la Ilustración 11), las herramientas que aparecen son relacionadas con el proyecto/archivo creado. Está la opción de crear nuevo proyecto, guardar el actual y abrir uno nuevo ya creado. En esta sección está también la herramienta de *Run*, ícono que se pulsará siempre que se quiera simular y arrancar el programa desde esta aplicación.



Ilustración 11. Herramientas disponibles en DESIGNER

Por otro lado, en *CANVAS* (mostrado en la Ilustración 12) están las herramientas que facilitan la creación y colocación de los componentes en la *Design View*. En función de los componentes que se seleccionen, existe la posibilidad de alinearlos de distintas maneras, agruparlos como un único componente para una más sencilla colocación en la posición deseada, igualar el tamaño, etc. En este apartado también está la opción de Run mencionada con anterioridad.

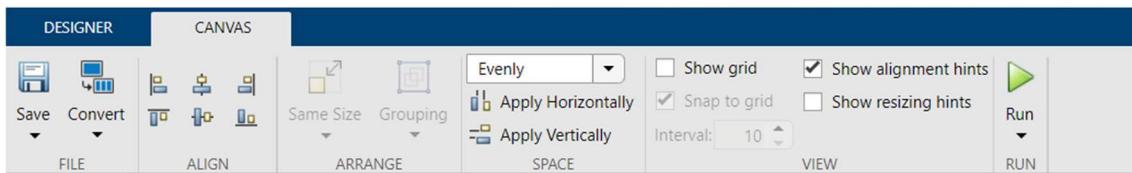


Ilustración 12. Herramientas disponibles en *CANVAS*

Por último, el apartado de *EDITOR* (representado en la Ilustración 13) se compone de todas las herramientas que ayudan al programador a la escritura del código, como la creación de funciones, propiedades o *Callbacks* o la posibilidad de ir a una línea de código en concreto, comentar dicha línea o descomentarla. Como en los otros dos apartados, también existe el icono de *Run*.



Ilustración 13. Herramientas disponibles en *EDITOR*

Capítulo 3. DESARROLLO DE LA INTERFAZ

3.1. CÓDIGO IMPLEMENTADO

Para la programación de la interfaz se ha implementado código en dos plataformas distintas, tanto en Matlab, como en App Designer. Para el desarrollo del proyecto, además de los manuales y tutoriales de App Designer, se ha obtenido información del foro de Matlab de la web del programador [10].

El código de App Designer se centra, principalmente, en la creación de cada componente y su definición, de tal manera que el usuario pueda introducir los valores de entrada que desee y el programa sea capaz de captar esos valores correctamente. Por otro lado, el código implementado en Matlab se centra más en la recepción de estos datos, ya que es en Matlab donde se encuentran los *scripts* que definen el comportamiento de la simulación del sistema fotovoltaico, y dan lugar a los resultados del programa completo. Estos *scripts* deben tener un espacio de almacenamiento desde el que poder leer los valores introducidos por el usuario, y, por lo tanto, este lugar debe ser accesible desde Matlab. La herramienta escogida para comunicar ambas plataformas y como se ha comentado, almacenar los valores para escribir y leer de ella, es la creación de un espacio de trabajo en Matlab (*Workspace*). De esta manera, App Designer, “escribirá” las variables que el usuario introduzca por la interfaz en el

Workspace, y Matlab “leerá” dichas variables para, posteriormente, pasarlas a los *scripts* de simulación ya existentes.

El esquema del flujo de información entre ambas plataformas sería el mostrado en la siguiente imagen (Ilustración 14).

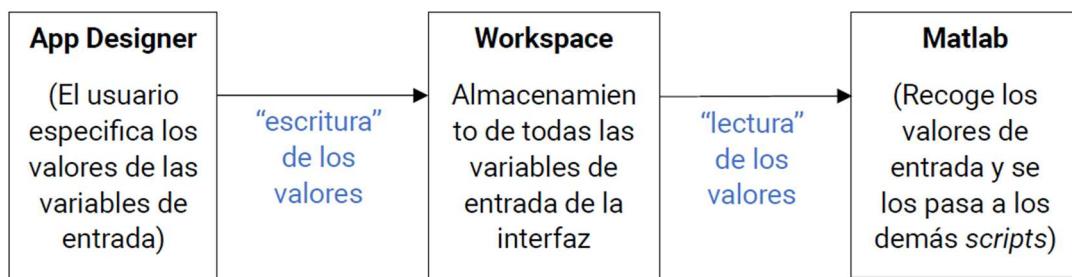


Ilustración 14. Flujo de información entre App Designer y Matlab

3.1.1. PROGRAMACIÓN EN APP DESIGNER

3.1.1.1. EXPLICACIÓN DE CADA COMPONENTE EN LA DESIGN VIEW

En primer lugar, se va a explicar el método de programación seguido en App Designer.

Para que el código de App Designer sea capaz de coger los valores de entrada introducidos por el usuario, lo primero es la creación de cada componente, uno para cada variable de entrada. La elección de cada tipo de componente es libre (dentro de las limitaciones que tenga cada variable en función de si se trata de una de entrada, de salida, de tipo texto, tipo número, etc.) por lo que se ha optado por una gran variedad de tipos de componentes, pudiendo así ver el funcionamiento de casi todas las opciones que nos proporciona App Designer.

La estructura general de la interfaz se compone de un Panel compuesto por quince pestañas llamadas *Tab Groups*, uno para cada aspecto que se valorará en la simulación del sistema, y cada *Tab Group* compuesto por diferentes componentes para cada valor de entrada. Además del Panel, en la página principal, existen un *Check Box* (componente destinado a la selección de un estado excluyente de otro) y un cuadro de texto que se puede editar, *Edit Field*, que permiten guardar los cambios que se hayan

Capítulo 3

hecho en el mismo proyecto en el que se encuentra o, si se prefiere, en uno nuevo creado con el nombre que especifique el usuario en el *Edit Field* destinado para ello. También existen dos *Button* que están destinados al cálculo y simulación del sistema fotovoltaico (llamado *CALCULATE*), y a la opción de cargar un proyecto previamente guardado, cuando ya está abierta la interfaz (llamado *LOAD PROJECT*).

A continuación, se va a explicar cada *Tab Group* y los componentes que forman parte de ellas.

Geographical data

En esta primera sección (Ilustración 15), se recogen los datos relacionados con la geografía del lugar donde se pretende ubicar el sistema fotovoltaico. Se ha optado por *Edit Fields* numéricos para las variables de *Altitude* (Altitud), *Longitude* (Longitud) y *Latitude* (Latitud) y un *Spinner* con saltos de quince unidades (grados en este caso) para la *Standard Longitude* (Longitud estándar). En cada variable aparece entre paréntesis la unidad de medida en la que se deben introducir los valores para evitar confusiones. Además, se ha implementado un *Switch* (componente destinado a la selección de un estado excluyente de otro) de *HELP*, con la funcionalidad de proporcionar una ayuda al usuario y que tendrá el mismo funcionamiento para todos los *Tab Groups*. Siempre estará inicialmente apagado. Si el usuario lo necesita, podrá encender este *Switch* y entonces aparecerá para cada variable un mensaje explicativo de su significado.

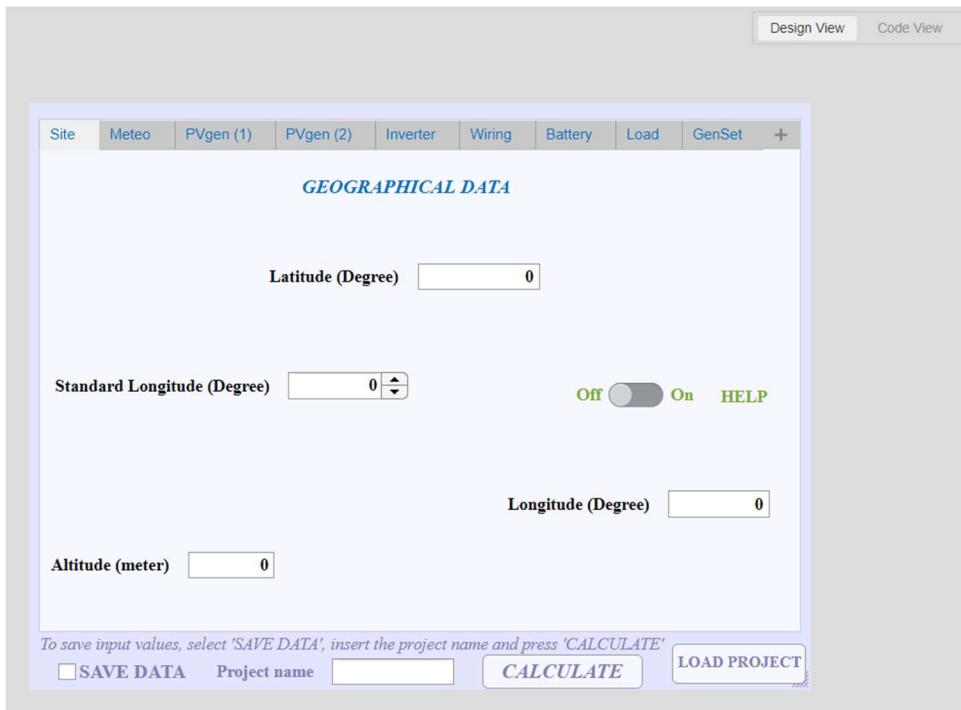


Ilustración 15. Tab Group Site en Design View

Meteorological input data

En esta segunda sección (Ilustración 16) los datos de entrada son los meteorológicos. Los componentes elegidos son una *List Box* (componente que muestra opciones entre las que escogerá uno el usuario) para el método de obtención de los datos teniendo cinco opciones, y un *Drop Down* (componente que despliega una serie de opciones entre las que tendrá que elegir uno el usuario) para las series temporales, teniendo como opciones *Mean Days* y *Aguiar*. En caso de que el usuario escoja la primera opción de la *List Box* (*Monthly averages from the table*), se deberán llenar también los datos de la tabla situada a la derecha. En esta tabla, cada fila corresponde a un mes del año y cada columna a un parámetro numérico distinto (*Gdm0* es la media mensual de la irradiación horizontal global diaria, *Tmm* la media mensual de la temperatura mínima diaria y *TMm* la media mensual de la temperatura máxima diaria), por lo tanto, todas las casillas de esta tabla son editables. El *Switch* de *HELP* tiene la misma función que la descrita en el primer *Tab Group*.

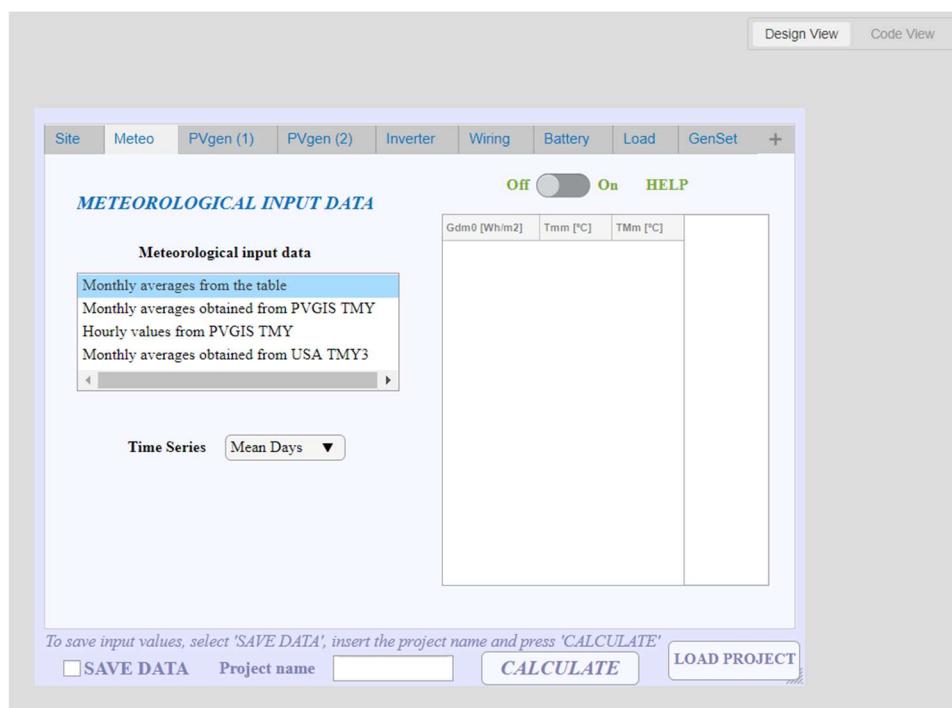


Ilustración 16. Tab Group Meteo en Design View

PV generator

Los siguientes dos apartados son los relacionados con el generador fotovoltaico. Como se puede observar en la Ilustración 17, todos los valores de la primera parte de esta sección son introducidos a través de *Edit Fields* numéricos y al igual que la mayoría de los componentes de este tipo, tienen indicadas las unidades de medida en las que se deben introducir los datos. Todas las casillas son editables por el usuario, pero la

Capítulo 3

relacionada con la resistencia térmica además de poder ser editada, tiene una fórmula específica para su obtención automática ($R_{th} = [NOCT - 20]/800$). Es decir, el valor inicial de esta variable se calcula automáticamente a partir de esta fórmula, y cada vez que se modifique la variable *NOCT*, se modificará *Rth* con ella, pero en caso de que el usuario quiera un valor específico para la resistencia térmica, también podrá introducirlo en el *Edit Field* y el programa lo admitirá como válido. Esta explicación viene detallada en la interfaz al activar el *switch HELP*.

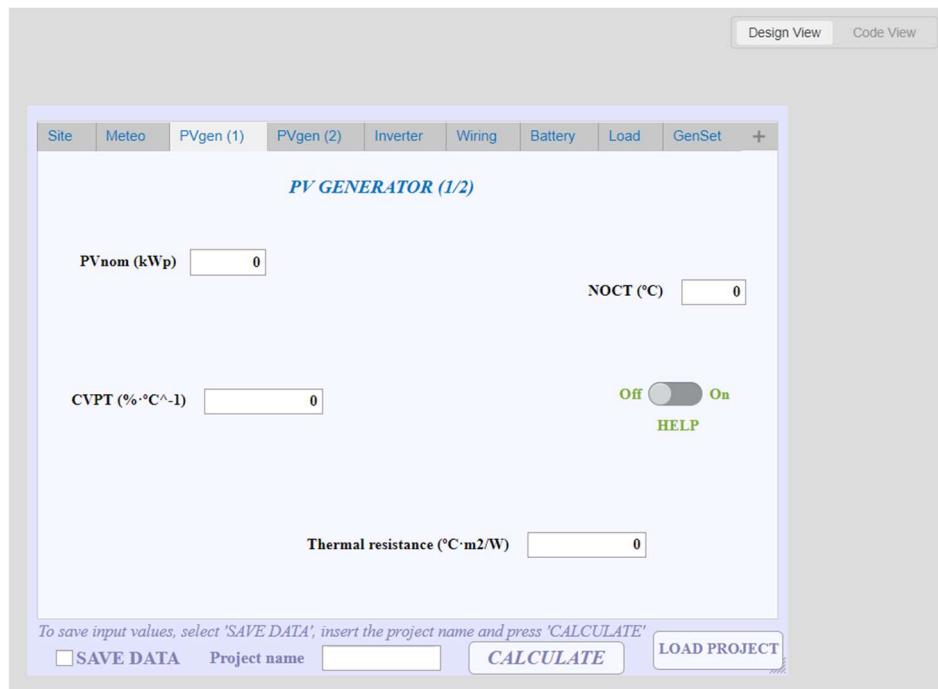


Ilustración 17. Tab Group PVgen (1) en Design View

El siguiente *Tab Group* del generador de sistemas fotovoltaicos (Ilustración 18) se centra en el tipo de estructura de montaje que se va a querer para el sistema. En primer lugar, el usuario deberá escoger una única opción entre los tipos de estructura disponibles en el *Radio Button Group* (no se permite escoger más de una opción). En función de la opción escogida, se deberán introducir las variables necesarias para cada tipo de montaje en los *Edit Fields* numéricos destinados para ello. Estas variables numéricas tienen ciertas limitaciones para que no se pueda introducir un valor fuera del rango establecido. Más adelante se indica una tabla con todos los valores mínimos y máximos permitidos para cada parámetro. La definición de los rangos y la explicación de cada variable se hará visible al activar el *Switch* de *HELP*, al igual que en todas las demás secciones de la interfaz.

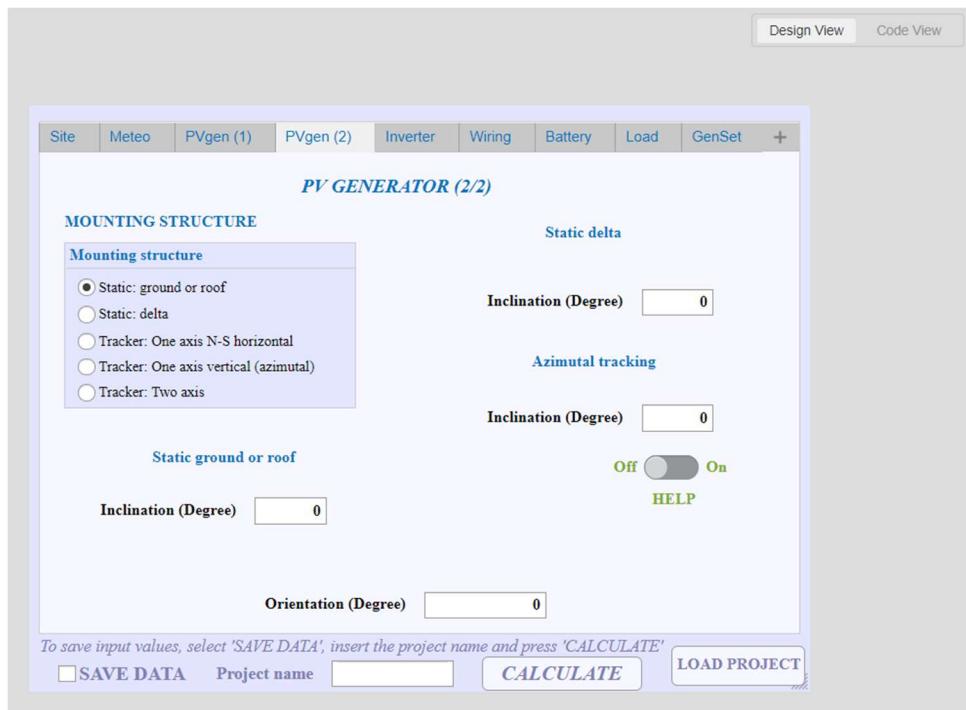


Ilustración 18. Tab Group PVgen (2) en Design View

Inverter

Esta quinta sección (Ilustración 19) es la del inversor y se compone de distintos tipos de variables, por lo tanto, diversos tipos de componentes. Para las potencias se ha optado por *Edit Fields* numéricos. Para la obtención de la curva del inversor se diferencian dos caminos que se pueden elegir con el *Toggle Button Group* (componente formado por un grupo de botones donde el usuario seleccionará uno de ellos excluyendo el resto). Si el usuario escoge la primera opción, *Model parameters*, deberá introducir los valores de los coeficientes k_0 , k_1 , y k_2 manualmente mediante los *Edit Fields* numéricos dispuestos para ello. Si, por el contrario, escoge la segunda opción, *Power Efficiency curve*, deberá introducir los valores de la eficiencia energética en la tabla. Esta tabla tiene, por lo tanto, todas las casillas editables, y en el mismo momento en que se modifica un valor de esta, la gráfica de su derecha se actualiza con la nueva curva basada en los nuevos puntos definidos. En esta opción, el programa calculará los coeficientes k_0 , k_1 y k_2 a través de la curva de eficiencia. Todos los *Edit Fields* numéricos que aparecen en este apartado tienen también un rango establecido entre el que se debe encontrar el valor introducido por el usuario, con el mismo procedimiento que el explicado en el apartado anterior. También está la opción de *HELP*.

Capítulo 3

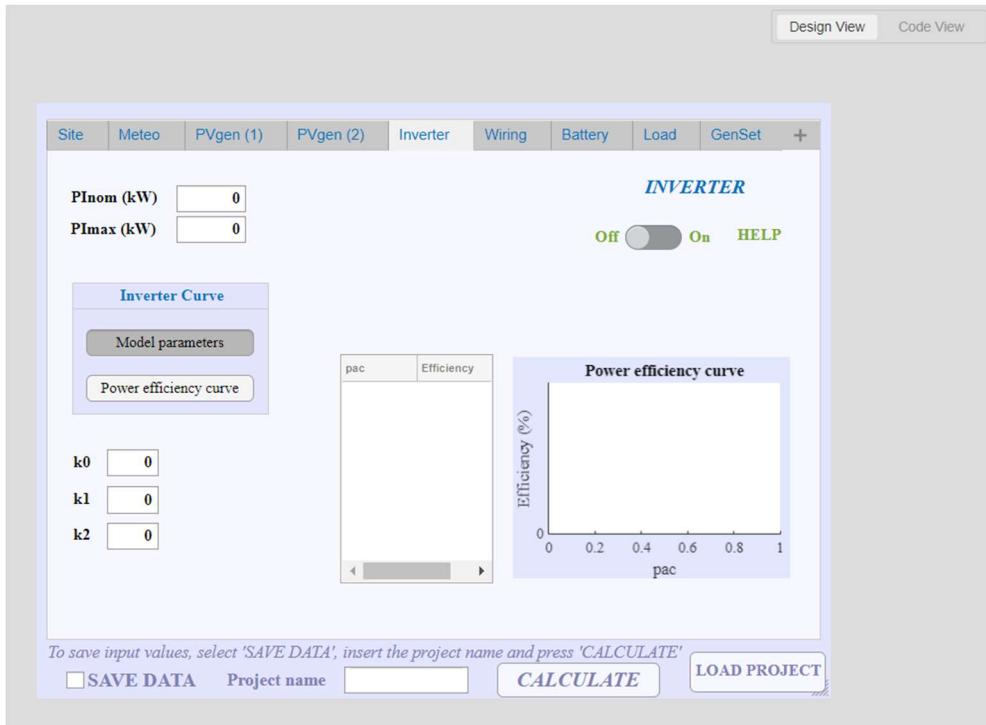


Ilustración 19. Tab Group Inverter en Design View

Wiring

El apartado de *Wiring* (Ilustración 20) se compone únicamente de dos *Edit Fields* numéricos, cada uno para un tipo de pérdidas, ambos limitados entre cero y diez y con la indicación de sus unidades de medida, en este caso %. Al ser una explicación muy breve de cada variable, no se ha optado por la adición de un *Switch* para ayudar al usuario como en otros *Tab Groups*.



Ilustración 20. Tab Group Wiring en Design View

Battery (Stand-alone PV systems)

Esta sección mostrada en la Ilustración 21, es únicamente para los sistemas con el tipo de aplicación *Stand-Alone* y está relacionada con la batería. Al igual que la anterior, esta se compone únicamente de *Edit Fields* numéricos, uno para cada variable, también con sus unidades de medida en caso de tenerlas. En este caso, sí que se ha optado por el *Switch* de *HELP* para ayudar al usuario con cada parámetro.

Capítulo 3

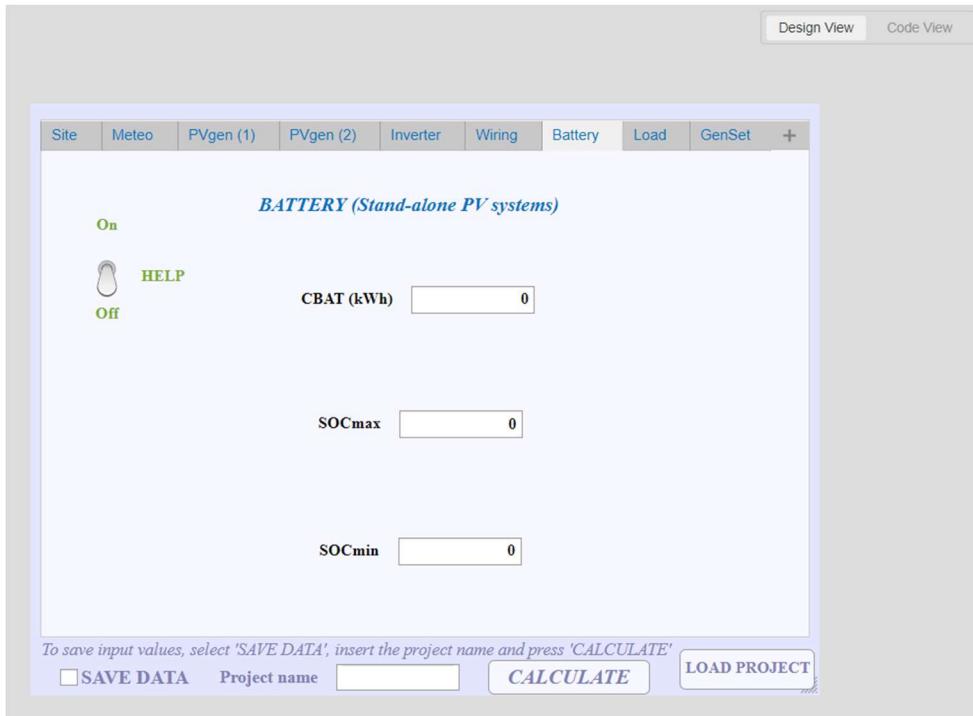


Ilustración 21. Tab Group Battery en Design View

Load Profiles

A continuación, en la Ilustración 22, aparece el apartado de carga de perfiles. Para la introducción de los valores de entrada, en este caso la media mensual del consumo diario (Ldm) y la fracción horaria de energía consumida cada hora ($F(h)$), se ha escogido la opción de Tables. Las casillas de ambas tablas son editables y el usuario podrá introducir los valores requeridos en ellas. En la primera de ellas, cada fila corresponde a cada mes del año y en la segunda, a cada hora del día. Al modificar los valores en las tablas, automáticamente se modificará la curva representada en cada gráfica. Estas gráficas representan los valores mensuales y horarios de los datos de entrada mencionados. Los dos *Edit Fields* numéricos que aparecen debajo de cada tabla son especiales, ya que no tienen la opción de ser editados por el usuario. Estos valores de modificarán automáticamente al cambiar los datos de las tablas ya que representan el total de cada una de ellas. Al igual que en la mayoría de los *Tab Groups*, en este caso también está la opción de *HELP*.

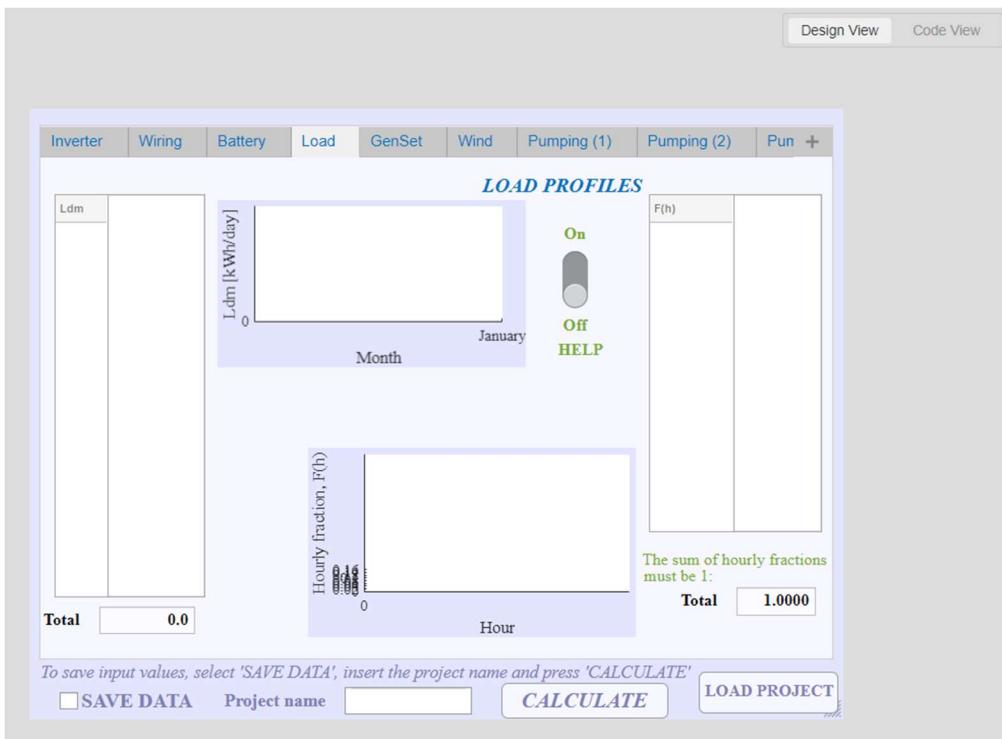


Ilustración 22. Tab Group Load en Design View

Generator set

Este *Tab Group* centrado en el grupo electrógeno (Ilustración 23) combina distintos tipos de elementos, como se puede observar en la imagen. Para los valores numéricos se ha optado por la elección de *Edit Fields* numéricos en los que se indican también las unidades con la implementación de *Labels* (etiquetas). El único parámetro en el que se define el rango permitido es el correspondiente a la potencia nominal del grupo. Por otro lado, para los puntos de la curva de consumo, al ser una colección de varios valores, la elección ha sido la de una *Table* junto con un *Axe* (componente destinado a la visualización de gráficas 2D) para su representación visual. Las casillas de la *Table* no son editables por el usuario ya que los puntos de la curva se calculan en función a los parámetros introducidos en la parte superior de la ventana ($b0i, b1i, b0s, b1s, PGENnom$). Al modificar uno de estos valores, los puntos en *Table* se actualizan y, por tanto, la gráfica que los representa también. También existe un botón de ayuda.

Capítulo 3

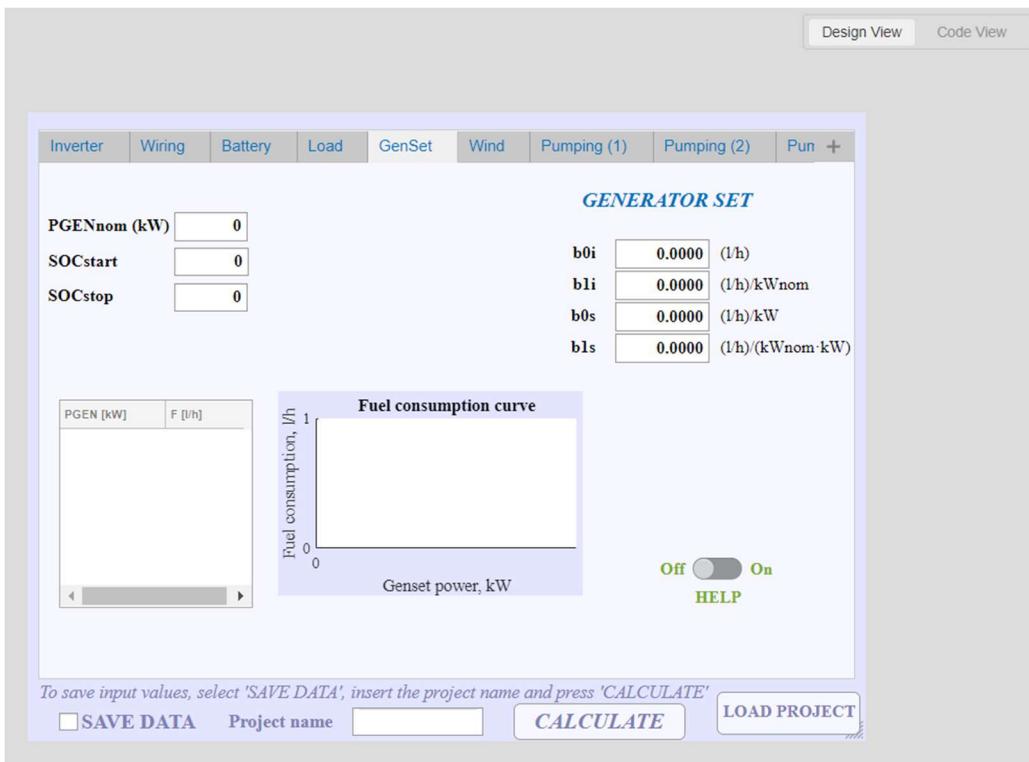


Ilustración 23. Tab Group GenSet en Design View

Wind turbine

En este caso mostrado en la Ilustración 24, ocurre algo similar que en el de GenSet. El usuario tiene la posibilidad de editar los valores numéricos de los *Edit Fields* de la parte superior excepto el parámetro denominado *Cpeq*. Esta variable no puede ser modificada por el usuario porque es un dato que se obtiene a partir de cálculos realizados a partir de las otras variables. Al modificar los cuatro primeros *Edit Fields*, se actualizan los de la tabla que miden la velocidad del viento y la potencia y el *Edit Field* de *Cpeq*. Con ellos, se actualiza también la gráfica que los representa. Las casillas de la *Table*, por lo tanto, no son editables. El *Switch* de *HELP* tiene la misma función que en los casos anteriores.

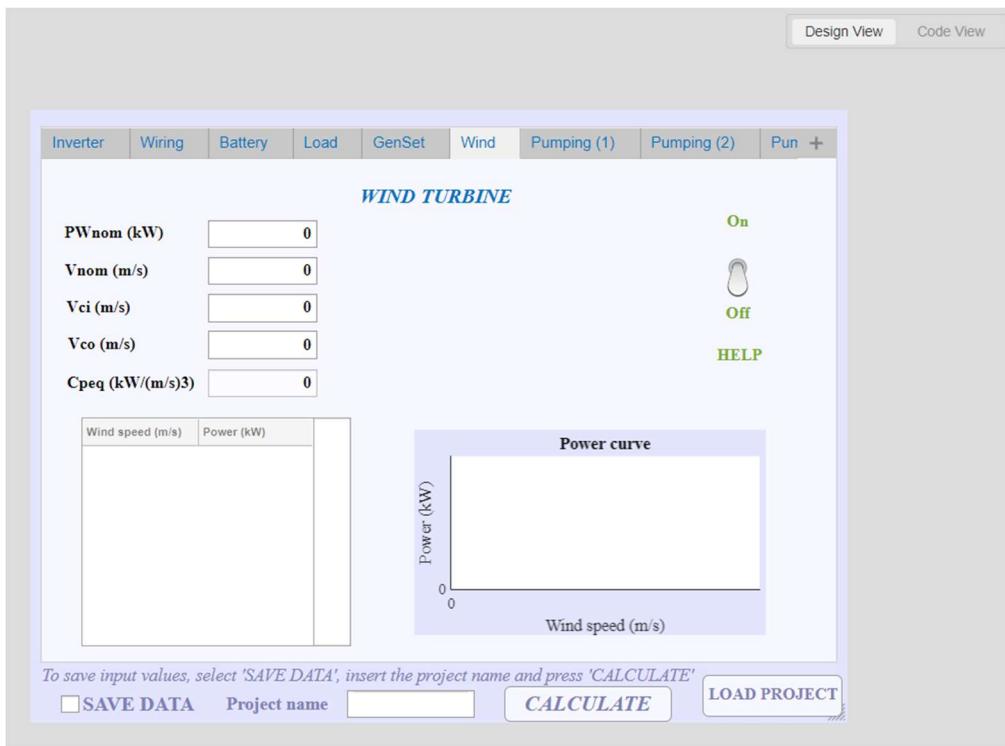


Ilustración 24. Tab Group Wind en Design View

Water pumping

El apartado relacionado con el Bombeo (*Pumping*) se ha dividido en cuatro *Tab Groups* para conseguir una representación más estética y menos recargada.

En la primera parte del bombeo (Ilustración 25) se definen los parámetros relacionados con el *Well/Borehole* (orificio de perforación), *Water Tank* (tanque de agua) y *Pump* (bomba), todos ellos con *Edit Fields* numéricos. Todas las casillas son editables por el usuario, pero en la variable *kw1* ocurre lo mismo que en la variable de la resistencia térmica del *PVgen*. Esta variable se define a partir de una fórmula ($kw1 = Hdr / Qtest$) y, por lo tanto, se actualizará cada vez que se modifique alguna de las variables de las que depende, pero también puede ser editada directamente por el usuario. Como se puede observar, cada variable viene indicada con las unidades de medida en las que debe ir, y al activar el *Switch* de *HELP* se visualizarán los mensajes explicativos de cada una de ellas.

Capítulo 3

Design View Code View

WELL / BOREHOLE

Hs (m)	0
Qtest (m ³ /h)	0
Hdr (m)	0
kwl (m/(m ³ /h))	0
kw2 (m/(m ³ /h) ²)	0

WATER TANK

Off On [HELP](#)

PUMP

WTC (m ³)	0
Qrated (m ³ /h)	0
Hrated (m)	0
Hr (m)	0
Liquid Density (kg/m ³)	0

To save input values, select 'SAVE DATA', insert the project name and press 'CALCULATE'

[SAVE DATA](#) Project name [CALCULATE](#) [LOAD PROJECT](#)

Ilustración 25. Tab Group Pumping (1) en Design View

Para un mejor entendimiento del significado de cada variable visualizar la Ilustración 26.

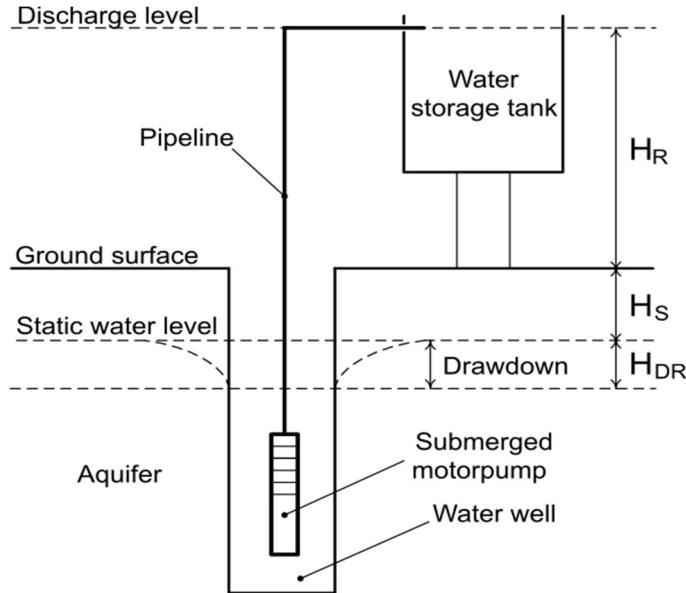


Ilustración 26. Esquema de las variables relacionadas con el Bombeo de agua

En la segunda parte (Ilustración 27) se definen a través de *Edit Fields* numéricos los parámetros necesarios para la composición de la *System curve* (curva del sistema). En este caso hay variedad en cuanto a la edición de estas variables. Por un lado, las variables llamadas H_f y k_{s1} son editables por el usuario y, por lo tanto, este deberá introducir el valor deseado para cada una de ellas. Por el contrario, las variables k_{s0} y k_{s2} no tienen habilitada la edición debido a que son variables que se calculan a partir de otros parámetros, por lo que, estas casillas se actualizarán a medida que se modifiquen los parámetros de los que depende cada una de ellas. Las fórmulas que definen su comportamiento son: $k_{s0}=H_r+H_s$ y $k_{s2}=[H_f/Q_{rated}^2]+k_w2$. Una vez se han completado debidamente estas variables, la *Table* que indica los puntos de la gráfica de *System Curve* se actualiza, y con ellos, dicha gráfica también.

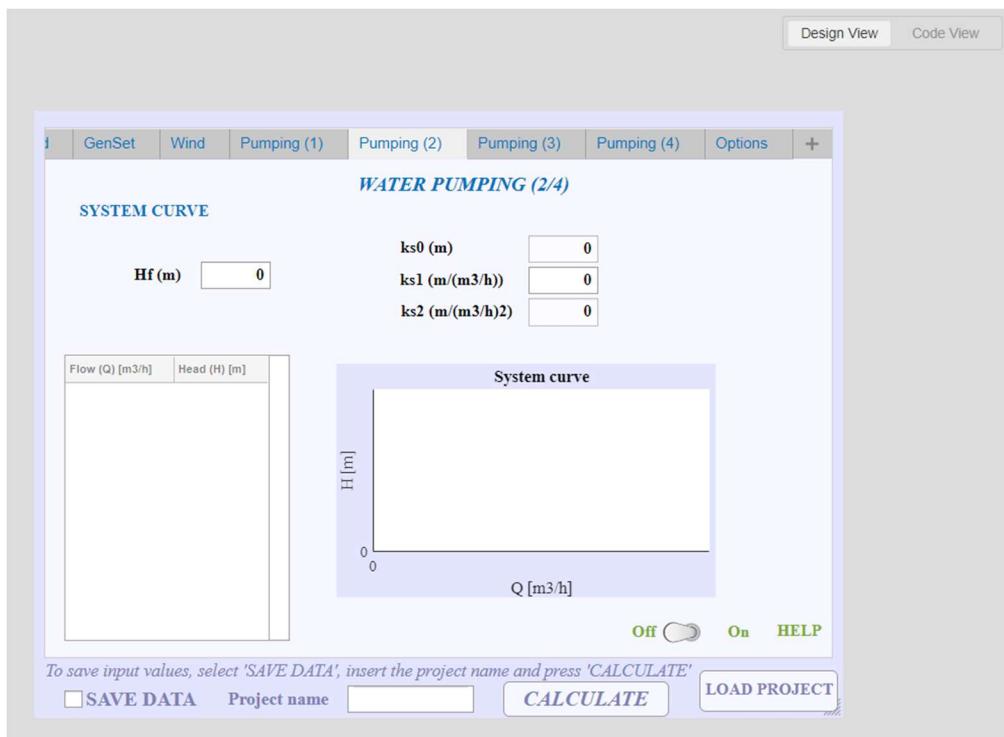


Ilustración 27. Tab Group Pumping (2) en Design View

La siguiente ventana, mostrada en la Ilustración 28, por el contrario, funciona de manera distinta, ya que los valores de las *Tables* son editables por el usuario, de tal manera que cuando el usuario modifica estos valores directamente en la tabla, las gráficas que corresponden a cada una de ellas se actualizan.

Capítulo 3

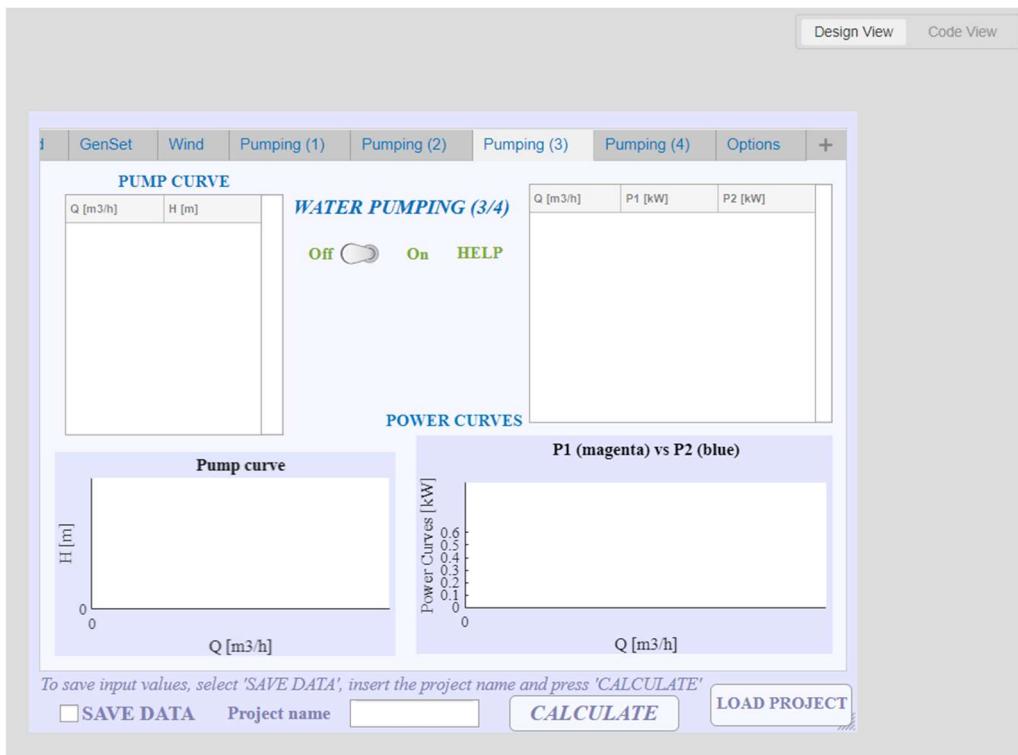


Ilustración 28. Tab Group Pumping (3) en Design View

Para la última de las partes (Ilustración 29), se ha optado por *Edit Fields* numéricos para los parámetros del motor y una *Table* con casillas editables por el usuario para los puntos de la *Power efficiency curve* (curva de eficiencia) que la acompaña. En esta última sección relacionada con el bombeo existen dos variables que están definidas por un rango permitido y que, al igual que en los demás casos de este tipo, al introducir un valor fuera del rango, aparece un mensaje de error.

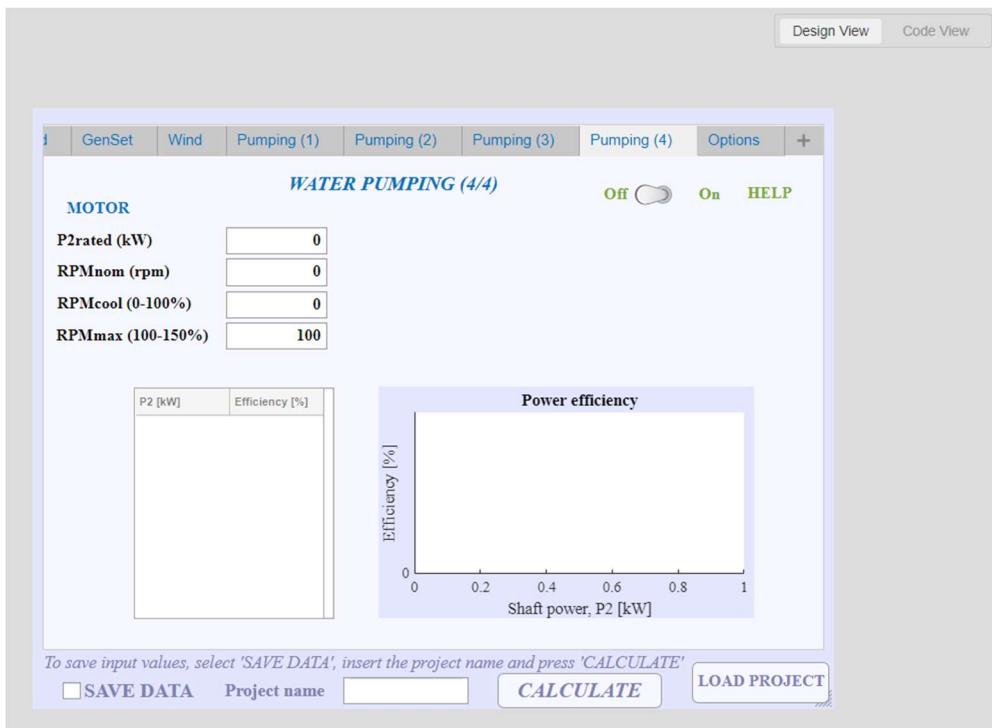


Ilustración 29. Tab Group Pumping (4) en Design View

Todas las ventanas disponen de un botón de ayuda con los mensajes explicativos necesarios para el perfecto entendimiento del usuario.

Options

Por último, el *Tab Group* de *Options*, mostrado en la Ilustración 30, es el más importante desde el punto de vista de simulación. En él, se debe seleccionar el tipo de aplicación del sistema pudiendo elegir en un *List Box* entre las cinco opciones más comunes de este tipo de sistemas (conexión a la red eléctrica, con o sin autoconsumo, sistemas autónomos con baterías y sistemas de bombeo de agua). También se deberá escoger el grado de suciedad a través de un *Toggle Button Group* compuesto por cuatro botones. La elección del modelo de cielo difuso se hará mediante un *Drop Down* con tres opciones y la correlación difusa mensual mediante un *Radio Button Group* con cuatro opciones entre las que el usuario podrá elegir. Para terminar, el usuario podrá escoger la reflectancia del suelo en un *Spinner* con saltos de 0.05 y límites entre 0 y 0.5 y los segundos de simulación con un *Edit Field* numérico entre 60 y 3600.

Capítulo 3

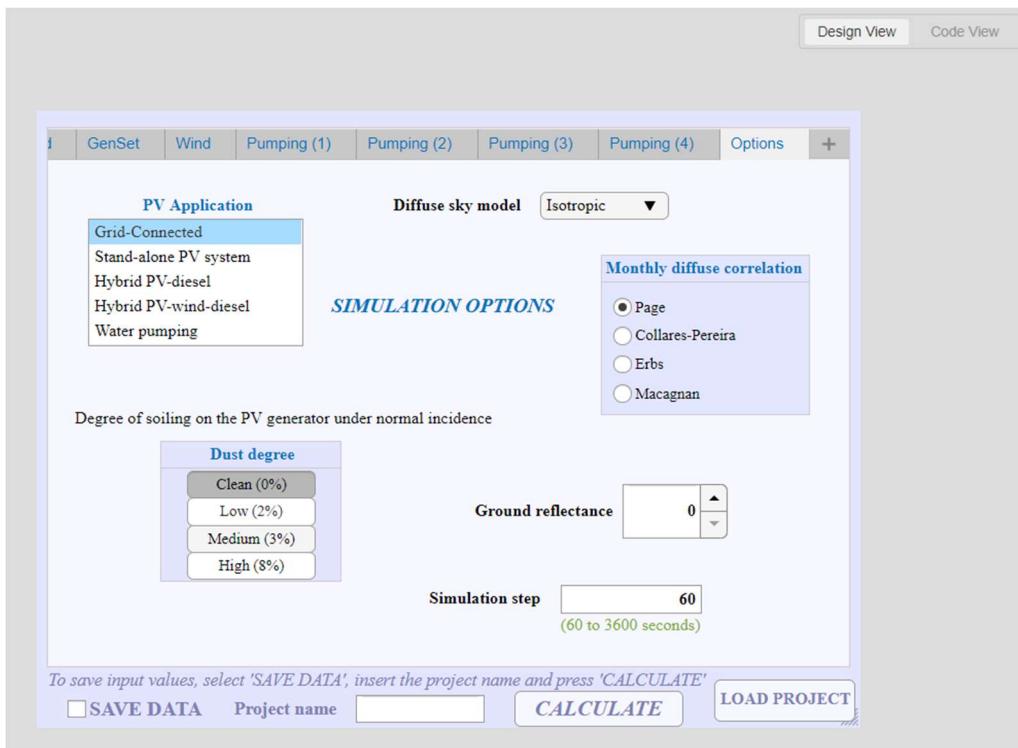


Ilustración 30. Tab Group Options en Design View

3.1.1.2. RANGOS PERMITIDOS PARA CADA VARIABLE

En la siguiente tabla (Tabla 2) se muestra un resumen de todas las variables numéricas que puede modificar el usuario a través de la interfaz con los valores mínimos y máximos permitidos entre los cuales debe estar el valor introducido por el usuario. También se indican las unidades de medida de cada una de ellas.

Tabla 2. Rangos y unidades de cada variable numérica

Variable	Rango (mínimo-máximo)	Unidades
Latitude	-	Grados
Standard Longitude	-	Grados
Altitude	-	Metros
Longitude	-	Grados
Inclination	0 - 90	Grados
NOCT	-	°C
PVnom	-	kWp
CVPT	0 - inf	% °C^-1
Thermal Resistance	-	°C m^2 / W
Orientation	-	Grados
Plnom	0 - inf	kW
Plmax	0 - inf	kW
k0	0 - inf	-
k1	0 - inf	-
k2	0 - inf	-
WDC	0 - 10	%
WAC	0 - 10	%
CBAT	-	kWh
SOCmin	-	-
SOCmax	-	-
PGENnom	0 - inf	kW
SOCstart	-	-
SOCstop	-	-
b0i	-	l/h
b1i	-	(l/h)/kWnom
b0s	-	(l/h)/kW
b1s	-	(l/h)/(kWnomKW)
PWnom	-	kW
Vnom	-	m/s
Vci	-	m/s
Vco	-	m/s
Cpeq	-	kW/(m/s)^3
Hs	-	m
Qtest	-	m^3/h
Hdr	-	m
kw1	-	m/(m^3/h)
kw2	-	m/(m^3/h)^2
WTC	-	m^3
Hr	-	m
Qrated	-	m^3/h
Hrated	-	m
Liquid Density	-	kg/m^3
Hf	-	m
ks0	-	m
ks1	-	m/(m^3/h)
ks2	-	m/(m^3/h)^2
P2rated	-	kW
RPMnom	-	rpm
RPMcool	0 - 100	%
RPMmax	100 - 150	%
Ground reflectance	0 - 0.5	-
Simulation step	60 - 3600	Segundos

Capítulo 3

3.1.1.3. EXPLICACIÓN DEL CÓDIGO DE CODE VIEW

Una vez creados todos los componentes y definidas sus características desde la ventana de *Design View*, se ha procedido a la escritura del código en la vista de *Code View*, donde se han implementado todas las funciones y *Callbacks* que definen el comportamiento de la interfaz. Se adjunta, como anexo, la totalidad del código implementado en esta plataforma para su consulta (Anexo A. Código implementado en App Designer).

En la primera parte del código, como se ha comentado con anterioridad, se crean todos los componentes como *Properties* del programa (los nombres de cada componente son el nombre de su variable con una “I” al final para distinguirla de las variables del *Workspace* de Matlab) además de las *Properties* agregadas para su utilización en las funciones posteriores del código. Las *Properties* correspondientes a los componentes de la interfaz se crean como *Properties* públicas, pero en cambio, las creadas para los cálculos internos dentro de la interfaz se crean como privadas. La diferencia entre ambas es que las públicas podrán ser vistas y accesibles tanto dentro como fuera de la aplicación, pero las privadas solo serán accesibles desde dentro de ella.

A continuación, se definen los *Methods* implementados, que son, principalmente, las funciones y *Callbacks*.

La primera función que se define es la llamada *startupFcn* que es la función que se ejecutará siempre que se inicie la simulación de la interfaz.

En esta función lo primero que se hace es solicitar al usuario el proyecto que desea cargar en la interfaz. Esto lo hace a través del comando *uigetfile* en el que se especifica que únicamente se podrá seleccionar un archivo con la extensión *.mat* (es decir, un *Workspace* de Matlab), el cual deberá estar ubicado en la misma carpeta en la que se encuentran todos los demás archivos relacionados con el programa. Inicialmente, ya hay creados tres proyectos distintos con sus respectivos parámetros, uno con valores típicos para un sistema con aplicación de tipo *Pumping* (*'DataPumping'*), otro para *Stand Alone* (*'DataStandAlone'*) y un último para *Grid-Connected* (*'Data'*). Estos son los proyectos iniciales, pero el usuario podrá modificarlos o crear nuevos, y estos nuevos se ubicarán en el mismo lugar en el que están estos.

Una vez solicitado el proyecto que se desea cargar, se valora la posibilidad de que no se seleccione ninguno o se cierre la ventana de selección de archivos, y entonces el programa se cerrará dando lugar a un error. En caso de no existir ningún error el código sigue ejecutándose, y lo siguiente que hace es cargar en App Designer el archivo seleccionado por el usuario. Al cargar todas las variables del *Workspace* se procede a la inicialización de cada parámetro en todos los componentes de la interfaz. Para el

establecimiento del valor inicial en el caso de los datos numéricos se le asigna el valor que tiene la variable en el *Workspace* directamente, pero para los casos en los que se debe escoger una opción entre varias (como los *Radio Button Group*, *Toggle Button Group*, *List Box*, *Drop Down* o *Button*) el procedimiento es distinto. Se ha optado por la asignación de un valor numérico para cada opción, para conseguir así una mayor sencillez a la hora de interpretar estas variables en funciones posteriores del código.

Una vez se ha definido para cada componente su valor inicial a partir de este espacio de trabajo de Matlab, se definen también los valores iniciales de cada casilla de todas las tablas distinguiendo las tablas que tienen un valor específico simplemente y las que se calculan a partir de operaciones con otras variables.

También, se dibujan las gráficas iniciales en función de los valores dados a estas tablas que las acompañan.

Por último en esta función, se le asigna a cada variable del espacio de trabajo el valor dado a su respectivo componente. Esto se hace para que, al comenzar el programa, haya una total comunicación y coordinación entre los espacios de trabajo de Matlab y App Designer, ya que, si hubiera una distorsión entre ambos, se produciría un problema que imposibilitaría la simulación de la interfaz y, por lo tanto, del programa entero.

Las siguientes funciones son las *Callbacks* propias de cada componente, es decir, las *Callbacks* que se van a ejecutar cuando el usuario interaccione con la interfaz. Cada vez que se modifique un componente, se asignará el nuevo valor a su respectiva variable del *Workspace* de Matlab. Dependiendo de si esta variable es un número, una opción, una tabla, etc. la variable del *Workspace* tendrá unas características y dimensiones distintas. Por ejemplo, para las tablas se ha optado por una variable de tipo matriz con tantas columnas y filas como tenga el componente *Table* de App Designer. Al modificar los valores que afecten a las gráficas, se vuelve a dibujar dicha gráfica ya actualizada, por lo que se debe llamar a este comando en estas *Callbacks* también. Para las *Callbacks* de los *Switches* de *HELP* se implementa un 'if', para que en caso de estar encendido se visualicen todas las *Labels* de ese *Tab Group* con la definición de cada variable, y, por el contrario, en caso de estar apagado, no se visualicen dichos mensajes informativos.

Al pulsar el botón *CALCULATE* se llama a la función *pvlite* ubicada en un *script* de Matlab, donde se procede a la realización de todos los cálculos del sistema para la posterior obtención de resultados. Si se desea guardar el proyecto en el que se está trabajando (o en uno nuevo), se deberá activar el *Check List* llamado *Save Data* (que al igual que todos los componentes, tiene su respectiva variable en el *Workspace* de Matlab, para, más tarde, interpretarla y valorarla), modificar o no el nombre del proyecto en el *Edit Field* de tipo texto asignado para ello (si se cambia el nombre se creará un nuevo proyecto con este nuevo nombre elegido por el usuario, con los nuevos

Capítulo 3

parámetros; y si se deja el mismo nombre, se reescribirá el proyecto actual con los nuevos valores) y posteriormente, pulsar CALCULATE. Es muy importante seleccionar el *Check List* de Save Data, ya que, si no se selecciona este parámetro, aunque se modifique el nombre del proyecto, no se guardarán los valores. Para guardar los datos es imprescindible seleccionar Save Data y pulsar CALCULATE.

Por otro lado, si una vez abierta la interfaz y por lo tanto un proyecto, se desea cambiar de proyecto y cargar otro distinto, el usuario pulsará el botón LOAD PROJECT. Al pulsar este botón, su *Callback* correspondiente llama a la función *startupFcn* (descrita anteriormente), y por lo tanto se vuelve a ejecutar lo que hay dentro de ella. Se pide de nuevo al usuario que escoja el proyecto que desea cargar en la interfaz, y se procede de la misma manera.

Los últimos *Methods* definidos en el código son los de inicialización de los componentes en los que se define su posición, tipo de letra, visibilidad, etc., es decir, las características que se completan en la sección de *Component Browser* de la *Design View*.

3.1.2. PROGRAMACIÓN EN MATLAB

Además del código implementado en App Designer, como se ha comentado anteriormente, es necesario un código complementario ubicado en Matlab para realizar la correcta conexión entre ambas plataformas. Este código implementado en Matlab se encarga de leer los datos desde el *Workspace* para, posteriormente, pasar estas variables a los demás *scripts* de Matlab y que estos, puedan trabajar con ellas y obtener los resultados de la simulación de sistema.

El *script* creado para las lecturas de estas variables es el llamado “*newReadInputData*” y se encuentra como anexo en el final de este documento para su consulta (Anexo B. Código implementado en Matlab (B.1.)). La llamada de este *script* se realiza en el inicio de la función *pvlite*, la cual se ejecuta cuando el usuario pulsa el botón CALCULATE en la interfaz creada en App Designer.

Este *script*, por lo tanto, se encarga de ir leyendo variable a variable, en el orden en el que están dispuestos los componentes en la interfaz creada (primero los relacionados con la geografía, luego meteorología, generador, inversor, cableado, batería, etc.). La lectura de cada variable se realiza con el comando *evalin*, y una vez leída se almacena en una variable local del programa creada por dicho *script* que, por lo tanto, estará visible para cualquier función implementada en el programa, hasta la finalización de la simulación. Se leen todo tipo de variables, tanto numéricas tipo *doubles*, como matrices, etc. Algunas de ellas necesitan una serie de cálculos internos tras ser leídas que también se realizan en este código. Además, el código dispone de

Líneas de comentarios para cada línea de código en la que se explican la variable y los cálculos, en caso de haberlos, que se han realizado a posteriori.

En el apartado de los datos geográficos el código lee todas las variables relacionadas con él, y realiza ciertos cálculos internos. Por ejemplo, con la latitud para pasarl a radianes, y con la *StandardLongitude* para la obtención de la zona horaria.

Para el caso de la meteorología se leen las variables también y se comparan realizando un 'if' para que, dependiendo de la elección del usuario, se ejecute un código u otro. En un caso se leen los datos de la tabla, y en otro, se llama a otras funciones implementadas en Matlab.

En el caso del generador en primer lugar se leen todas las variables de la primera ventana y el tipo de estructura de montaje. A continuación, se valora el montaje que se ha seleccionado para proceder a la lectura de unas u otras variables (ya que si se trata de un tipo se leerá una casilla de *Inclination* u otra, y se le asignará este valor a la variable *Inclination* del *Workspace*).

En el apartado del inversor se leen las variables y posteriormente se realiza una distinción de comportamiento en función de la opción escogida por el usuario. En caso de que se seleccione la opción de *Model Parameters*, el código lee los valores de los coeficientes. Por el contrario, si se escoge la otra opción, se llama a una función que se encarga de realizar un ajuste de la curva de eficiencia y con ella, el programa obtiene los coeficientes. Todo ello lo realiza a partir de la lectura de los valores de la tabla de las variables *pac* y *efficiency*.

Continuando con el apartado de cableado y batería, ocurre lo mismo que en el generador, únicamente se leen las variables del *Workspace*.

En la sección de *Load Profiles* se realiza la lectura de cada valor, y con alguno de ellos, en especial las matrices, se realizan una serie de cálculos como la trasposición para poder tratar con estos datos en otras funciones del programa que así lo requieran.

En el caso de *Options* se combina también la lectura de las variables con una serie de cálculos internos para la obtención del número de puntos de simulación por día o por hora, por ejemplo, o el número de días simulados.

A continuación, se valora el caso de *Pumping* que es el más complejo porque es el que más cálculos requiere. Para este caso se ha dividido el código implementado entre este *script* (*newReadInputData*) en el que se leen todas las variables de la interfaz, de igual manera que en los demás casos; y uno nuevo creado como *PumpingModel* (también adjuntado en el final de este documento: Anexo B. Código implementado en Matlab (B.2.)).

Capítulo 3

Este nuevo *script* será llamado en el inicio del *script mainPump*, para que solo se ejecute en caso de que el sistema sea de tipo Bombeo y, por lo tanto, no se realicen cálculos y almacenamientos de variables innecesarios, y se encargará de todos los cálculos necesarios para este tipo de aplicación. Se debe hacer un ajuste de cada curva mencionada en este tipo de aplicación, similar al realizado en el apartado del inversor. Este ajuste se realiza como cálculo interno ya que es necesario para la obtención de resultados, pero no se visualiza como resultado de la simulación en Matlab. Esta visualización no se representa en las gráficas para que no quede recargado, pero en todos los casos se implementa el código (de forma comentada) que se necesitaría para visualizar la comparación entre el ajuste de la curva y la curva real. Si se desea visualizar esta comparación es tan sencillo como descomentar dicho código. Para finalizar con esta aplicación, se siguen realizando cálculos internos para la obtención de otras variables necesarias para la simulación, y una vez terminados se almacenan todas ellas de manera local en Matlab.

Posteriormente, se estudian los apartados de *GenSet* y *Wind* que se encargan únicamente de leer las variables relacionadas para cada caso. El tratamiento de estas variables solo se realiza para determinadas aplicaciones, en el caso de *GenSet* para las aplicaciones de *Hybrid PV-Diesel* y *Hybrid PV-wind-diesel*, y para el caso de *Wind*, únicamente si se trata de un sistema tipo *Hybrid PV-wind-diesel*. Todo esto se valorará en el *script* de *pvlite* en el que se elige qué funciones se ejecutan en función del tipo de aplicación del sistema.

Por último, se analiza la opción de guardar proyectos. Esta valoración se realiza con la ayuda de las variables llamadas *saveDataApp* y *name*. La primera de ellas es la relacionada con el *Check List* de la interfaz en el que el usuario decide si se va a guardar el proyecto o no, y la segunda con el *Edit Field* de tipo texto en el que el usuario elige el nombre con el que desea guardarlo. La variable *name* inicialmente tiene como valor el nombre del proyecto actual, es decir, del *Workspace* en el que se encuentra.

En este *script* (*newReadInputData*) únicamente se leen las variables *saveDataApp* y *name*, para posteriormente, una vez realizados todos los cálculos, se valore en el final del *script pvlite* (adjuntado como anexo en este documento: Anexo B. Código implementado en Matlab (B.3.)) si la opción de guardar el proyecto ha sido seleccionada o no. En caso de estar seleccionada esta variable (*saveDataApp*), es decir, tener valor 1, se realiza una concatenación del nombre escogido por el usuario para el proyecto (variable *name*) y la extensión *.mat*, y posteriormente se guarda el archivo con este nombre. Si el usuario no ha cambiado el nombre, el proyecto se reescribirá con los nuevos valores, pero, en cambio, si ha decidido modificar el nombre, se creará un nuevo proyecto con este nuevo nombre y los nuevos parámetros seleccionados sin modificar el proyecto anterior.

Una vez ejecutado todo el código del script `newReadInputData`, se ejecutan las demás funciones implementadas en Matlab (el código fuente ya existente que define el comportamiento de la simulación del programa *PVLite*). Las funciones que llame el código serán distintas en función del tipo de sistema que se esté estudiando (tipo *Pumping*, tipo *StandAlone*, etc.) y darán lugar, por lo tanto, a distintos resultados en función de la diversidad de los parámetros seleccionados en la interfaz.

Cabe destacar que, al guardar las variables del proyecto tras finalizar todos los cálculos necesarios para la obtención de resultados, no solo se guardan las variables de entrada de la interfaz, sino que también se almacenan todas las variables internas que han sido necesarias para los numerosos cálculos del programa, lo que resulta de gran utilidad para el análisis de estos sistemas fotovoltaicos.

Durante la simulación, solo se pueden visualizar las variables de entrada a la interfaz en el *Workspace* de Matlab, pero en caso de que el usuario desee analizar los cálculos que se han realizado y por lo tanto revisar las variables internas que se han ido sacando a lo largo de la simulación, bastará con ver el archivo con extensión `.mat` donde se han guardado todas las variables (que como se ha comentado anteriormente, coincidirá con el nombre del proyecto escogido por el usuario).

Para visualizar el contenido del archivo se podrá hacer por dos métodos.

1. Escribir en la ventana de comandos de Matlab el comando `load` con el proyecto que se deseé analizar, por ejemplo "`load('Data.mat')`".
2. Desde documentos, pulsar sobre el archivo y seleccionar "abrir".

3.1.3. ORDEN DE EJECUCIÓN DEL CÓDIGO

Para concluir con este apartado en el que se ha explicado todo el código implementado en el programa, se va a representar a continuación, un esquema que resume el orden en que se ejecutará cada parte del código.

1. App Designer solicita al usuario la selección del proyecto que desea cargar.
2. El código de App Designer carga los valores iniciales del *Workspace* de Matlab seleccionado por el usuario, y se los asigna a cada componente de la interfaz.
3. El usuario modifica los valores de entrada que considere a través de la interfaz.

Capítulo 3

4. El código implementado en App Designer asigna los nuevos valores a las variables del *Workspace* de Matlab.
5. El usuario decide si desea guardar el proyecto (seleccionando *Save Data*) y en caso afirmativo, elige el nombre con el que desea guardarlo (el mismo si quiere sobrescribir el proyecto actual, o uno distinto si prefiere crear uno nuevo y no modificar el actual).
6. El usuario pulsa *CALCULATE*. Al pulsar *CALCULATE*, el código de App Designer llama al script *pvlite* de Matlab donde se empieza a ejecutar todo el código de Matlab.
7. *pvlite* empieza a ejecutarse en Matlab. El primer comando de este script es la llamada a *newReadInputData* donde se leen todas las variables del *Workspace*.
8. Una vez leídas todas las variables, *pvlite* ejecuta el resto de los comandos, en los que se llama a distintas funciones y *scripts* (unos encadenados a otros) en función de los valores leídos del *Workspace*. Al final de este script (*pvlite*) se ejecuta también, en caso de desearlo el usuario, el comando destinado a guardar el proyecto.
9. Una vez realizados todos los cálculos, se ejecutan los comandos de dibujo y visualización de resultados y el programa finaliza presentando todos los resultados de la simulación del sistema fotovoltaico.
10. Tras la obtención de resultados, sin cerrar el programa, se puede continuar simulando nuevos sistemas con nuevos parámetros de la misma manera que se ha hecho hasta ahora, tantas veces como se desee, y, por lo tanto, creando y guardando tantos proyectos como se quiera. También existe la posibilidad de cargar otro proyecto distinto en la interfaz en cualquier momento, sin la necesidad de cerrarla y volver a abrirla, simplemente pulsado sobre el botón “LOAD PROJECT”.

3.2. FUNCIONAMIENTO DE LA INTERFAZ

Cuando el usuario quiera comenzar a usar el programa lo primero que deberá hacer es abrir Matlab. Ubicándose en Matlab en la carpeta donde se encuentran todos los archivos necesarios para que el programa funcione (todos los *scripts* destinados a la obtención de resultados, el script de *newReadInputData* que permite la lectura de

datos, el archivo de App Designer donde se encuentra la interfaz y su respectivo código, los archivos tipo *Workspace* que almacenan todas las variables, etc.), el usuario escribirá en la ventana de comandos el nombre del archivo de App Designer (“*appInterface1*”) como se muestra en la Ilustración 31. Es muy importante que Matlab se ubique en dicha carpeta porque si no, no reconocerá el comando y dará error o se abrirá un archivo distinto al deseado.



Ilustración 31. Comando para abrir la interfaz

Al escribir este comando se abrirá automáticamente la ventana de la interfaz y comenzará a ejecutarse todo el código comenzando por la solicitud del proyecto a cargar. Una vez seleccionado el proyecto, se cargarán los datos iniciales y entonces se podrá comenzar a modificar los parámetros necesarios. En esta ventana el usuario podrá navegar por donde desee, alternando las pestañas en las que se quiera ubicar (*Site, Meteo, etc.*).

A continuación, se muestran imágenes de los pasos que se seguirán mientras se vayan ejecutando todas las líneas del código de esta aplicación, comenzando por la elección del proyecto a cargar inicialmente en la interfaz, como se muestra en la Ilustración 32.

Capítulo 3

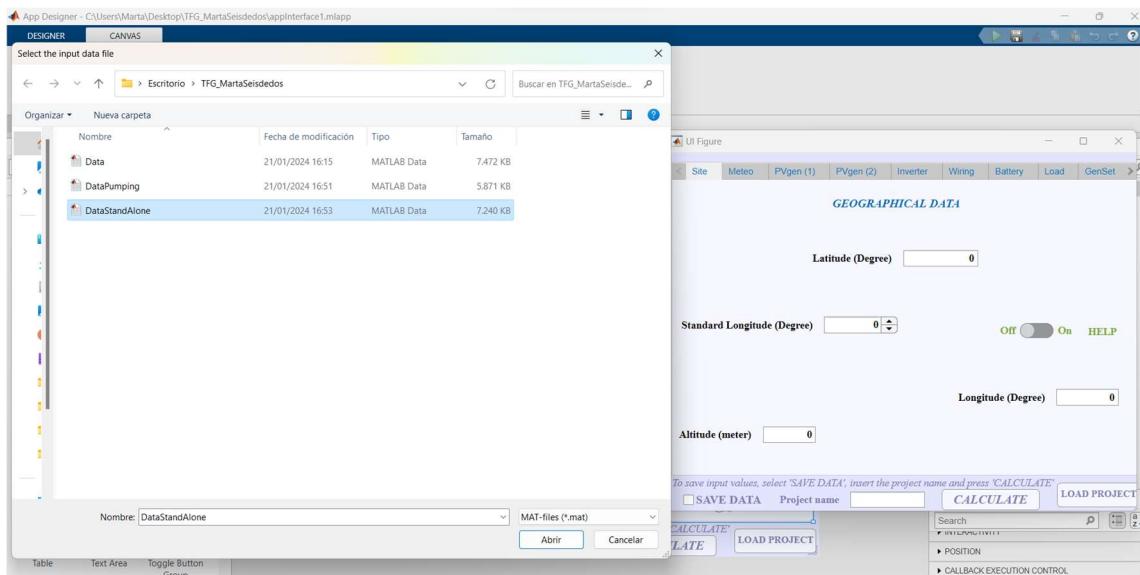


Ilustración 32. Interfaz funcionando: selección del proyecto a cargar

A continuación, se muestra una imagen de todas las pestañas con los datos cargados y los botones de ayuda al usuario activados.

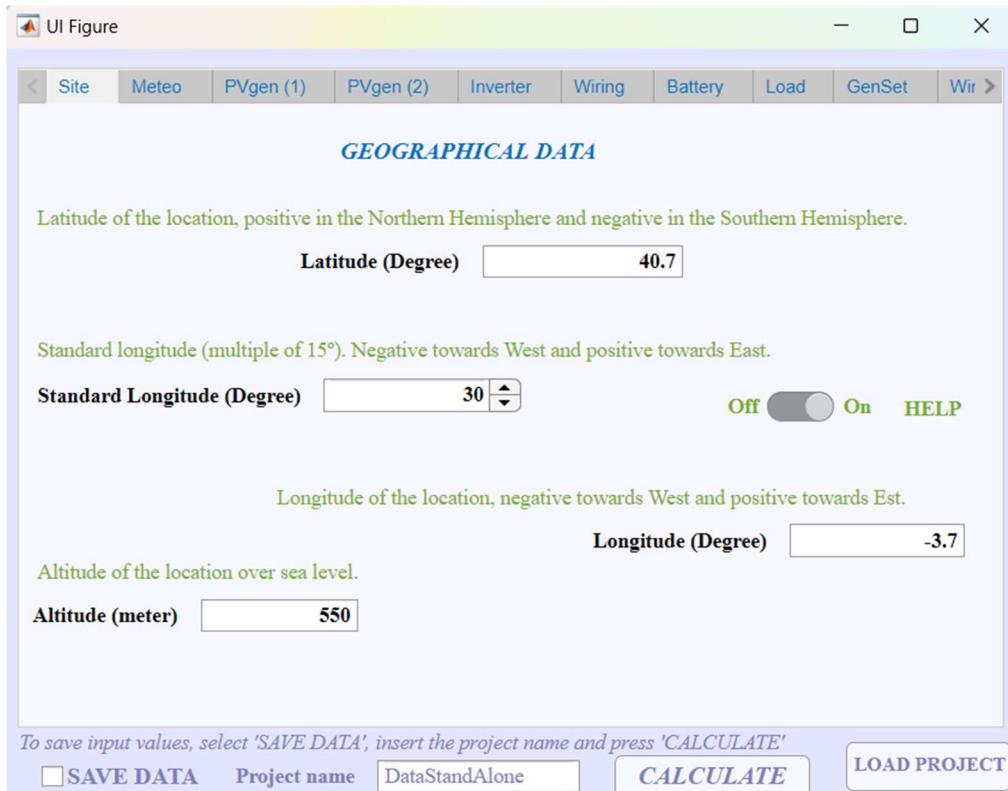


Ilustración 33. Interfaz en funcionamiento en la pestaña Site

Desarrollo de la interfaz

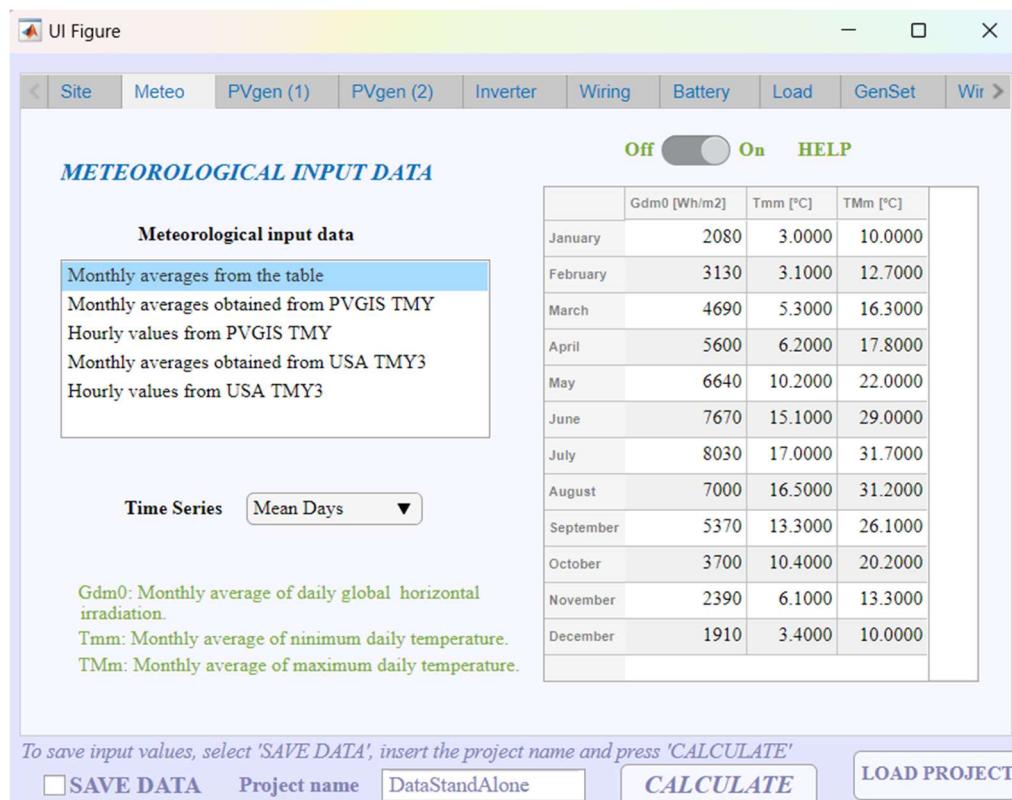


Ilustración 34. Interfaz en funcionamiento en la pestaña Meteo

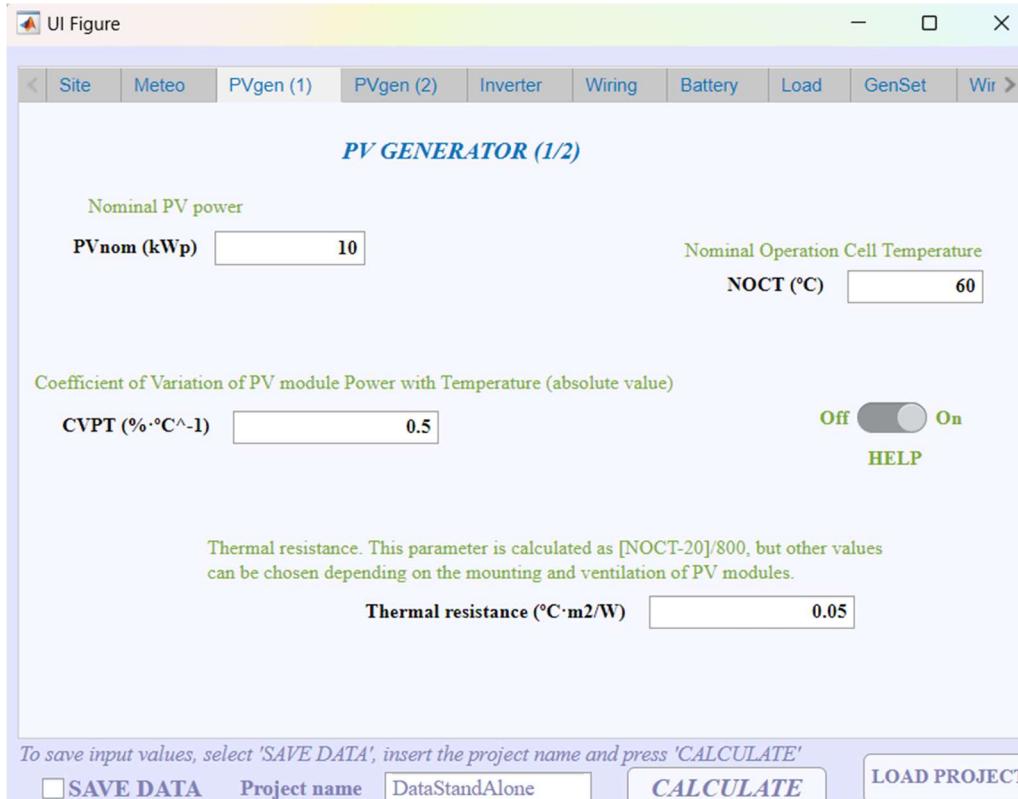


Ilustración 35. Interfaz en funcionamiento en la pestaña PVgen (1)

Capítulo 3

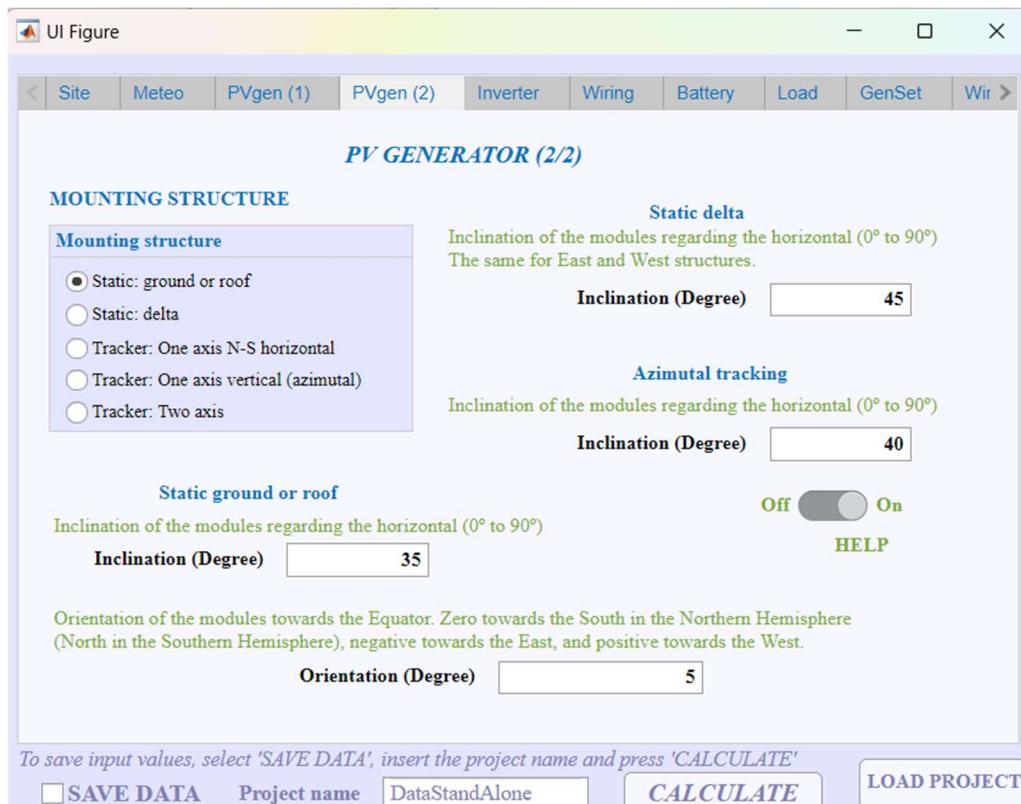


Ilustración 36. Interfaz en funcionamiento en la pestaña PVgen (2)

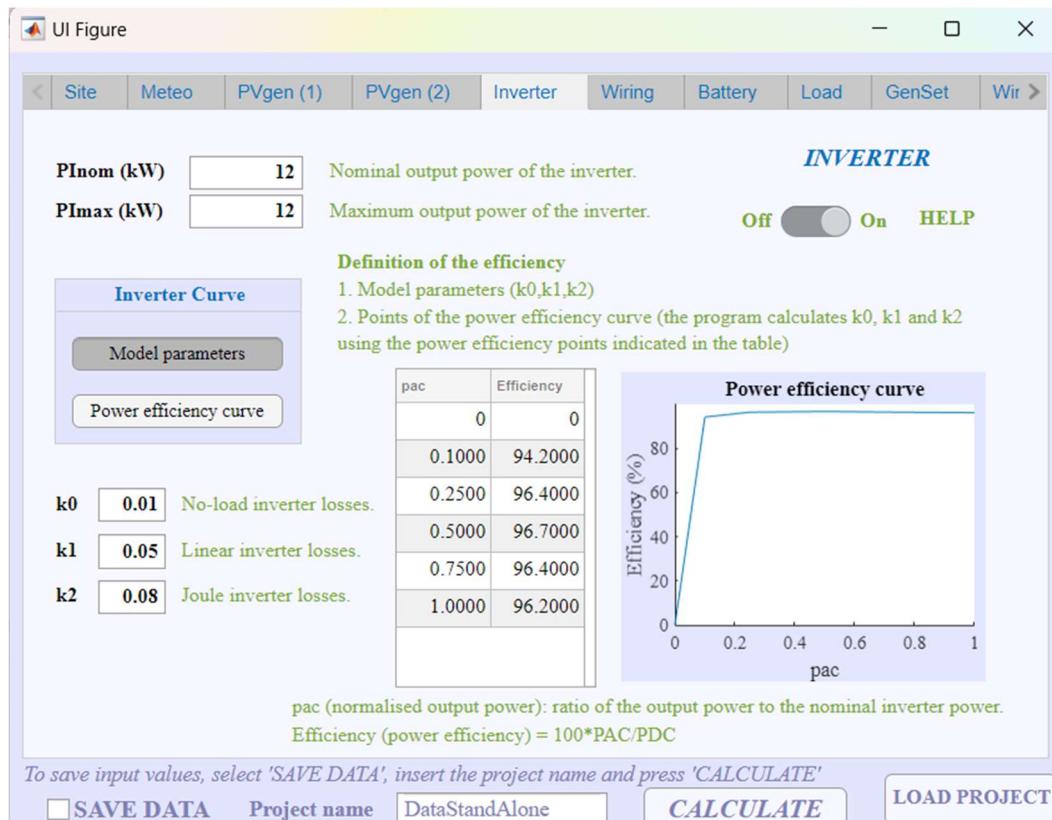


Ilustración 37. Interfaz en funcionamiento en la pestaña Inverter

Desarrollo de la interfaz

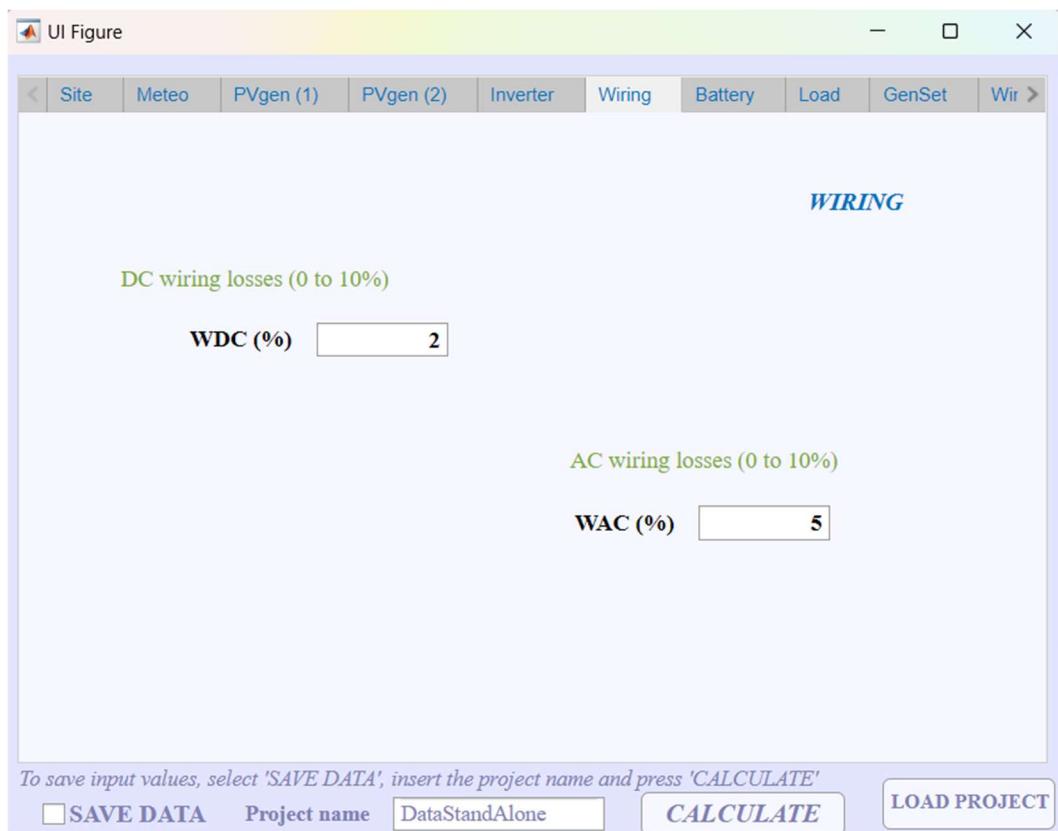


Ilustración 38. Interfaz en funcionamiento en la pestaña Wiring

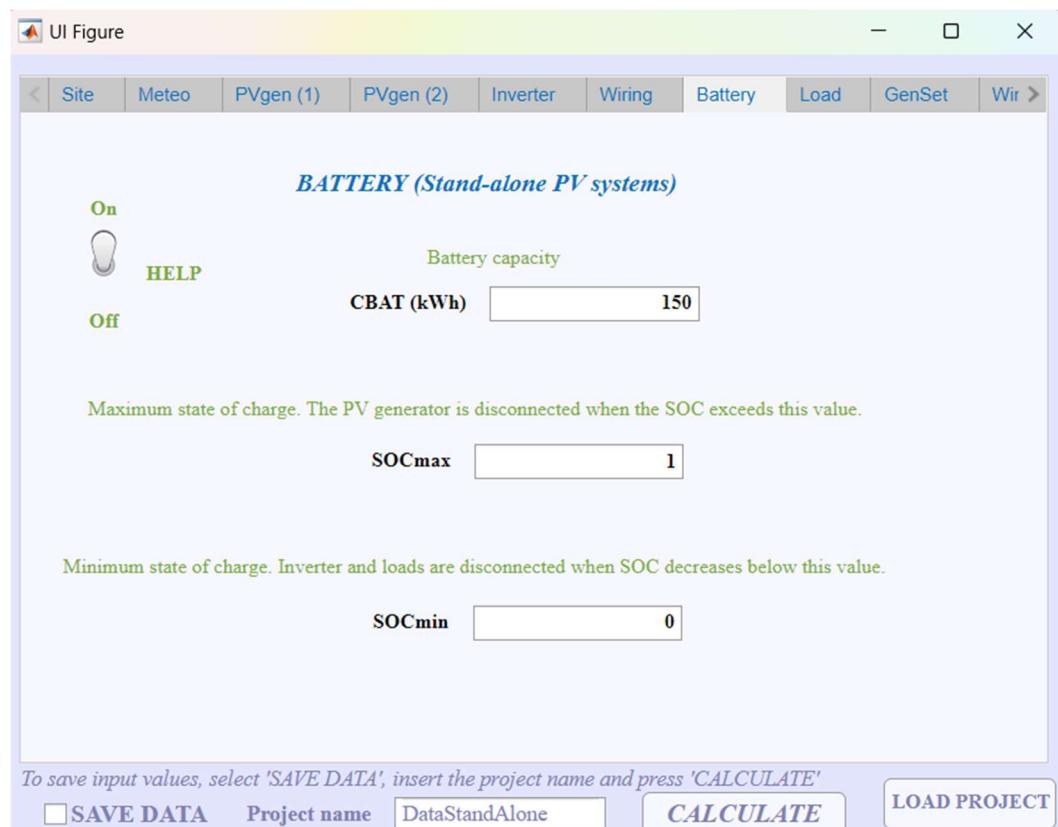


Ilustración 39. Interfaz en funcionamiento en la pestaña Battery

Capítulo 3

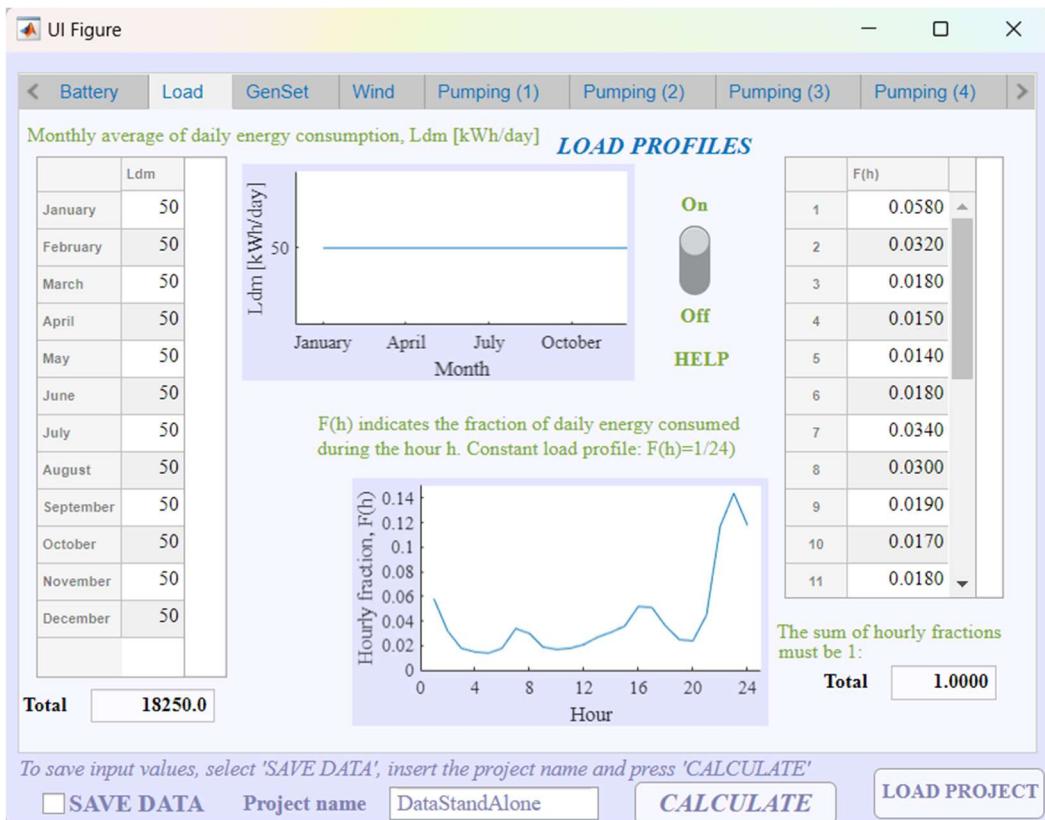


Ilustración 40. Interfaz en funcionamiento en la pestaña Load

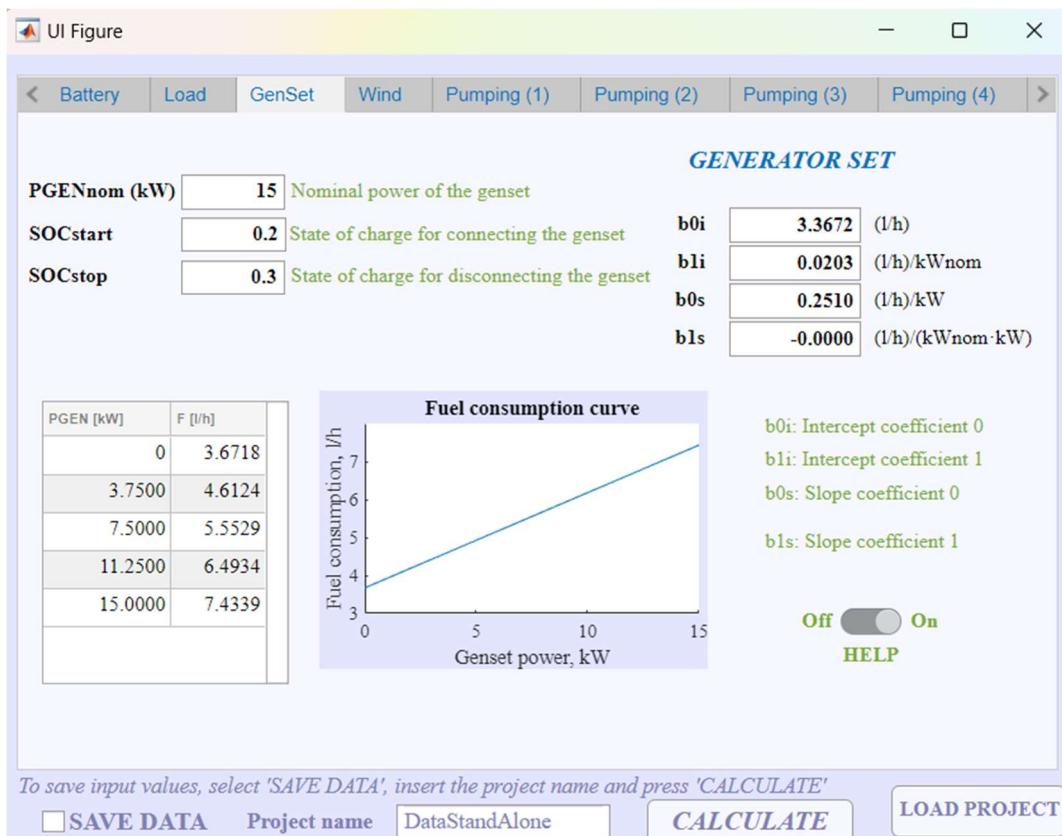


Ilustración 41. Interfaz en funcionamiento en la pestaña GenSet

Desarrollo de la interfaz

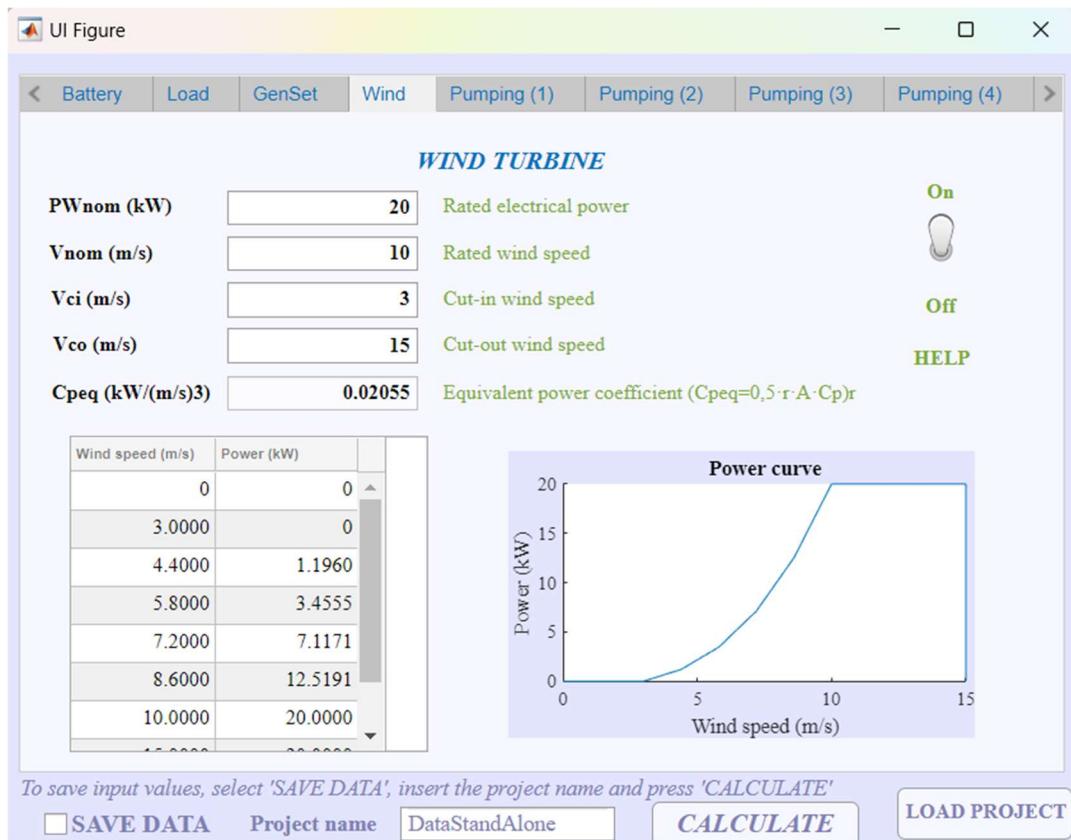


Ilustración 42. Interfaz en funcionamiento en la pestaña Wind

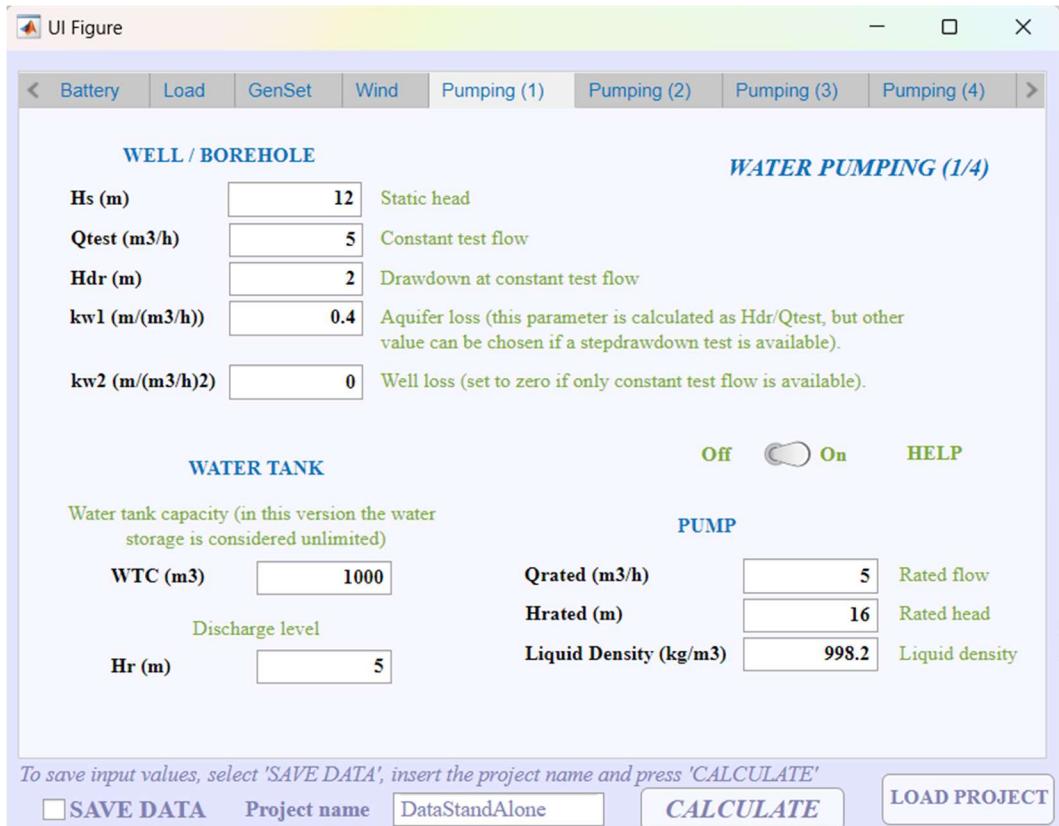


Ilustración 43. Interfaz en funcionamiento en la pestaña Pumping (1)

Capítulo 3

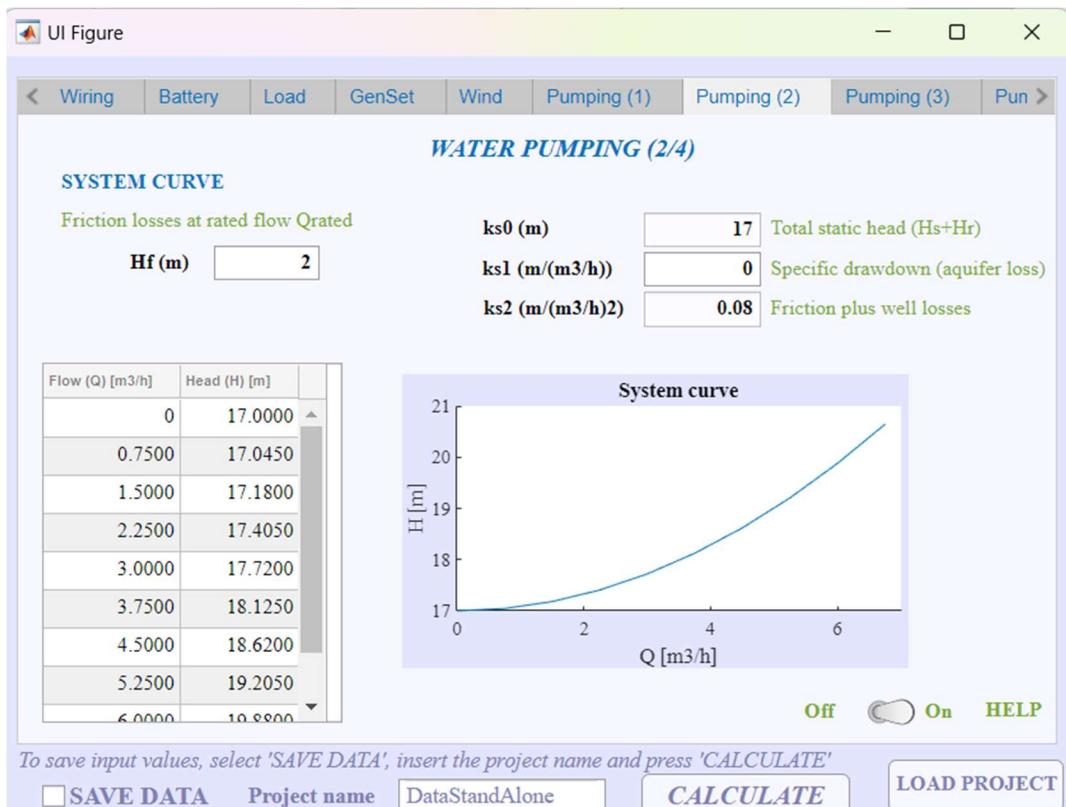


Ilustración 44. Interfaz en funcionamiento en la pestaña Pumping (2)

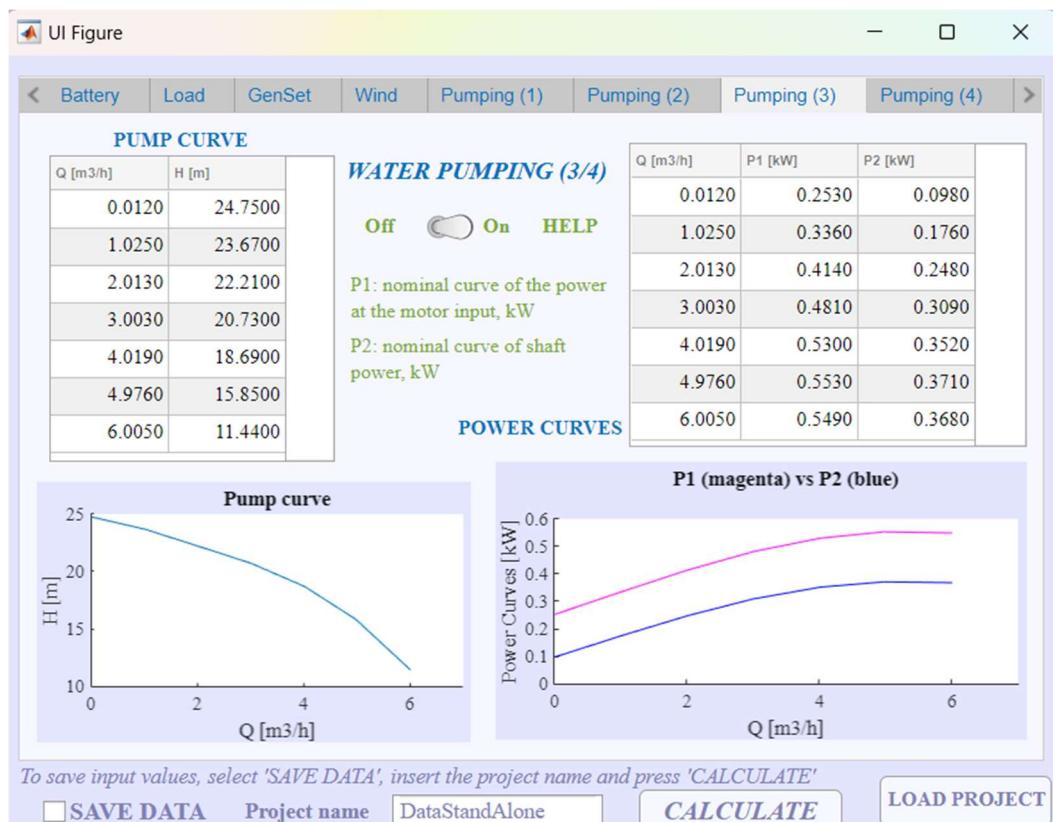


Ilustración 45. Interfaz en funcionamiento en la pestaña Pumping (3)

Desarrollo de la interfaz

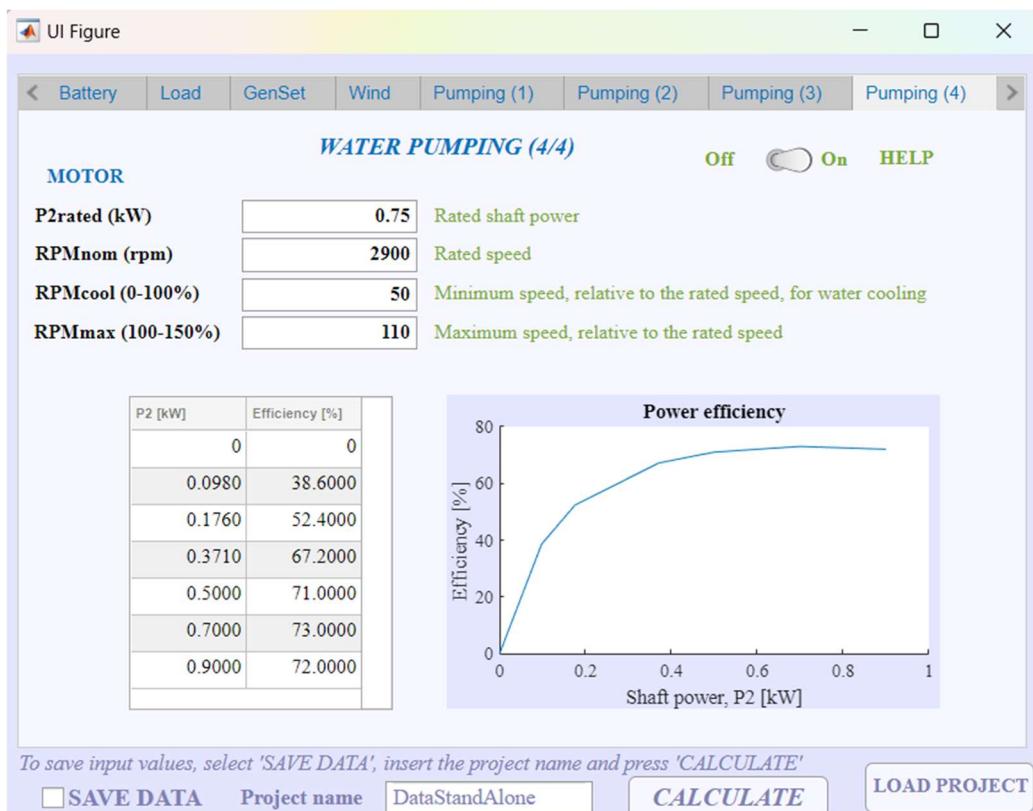


Ilustración 46. Interfaz en funcionamiento en la pestaña Pumping (4)

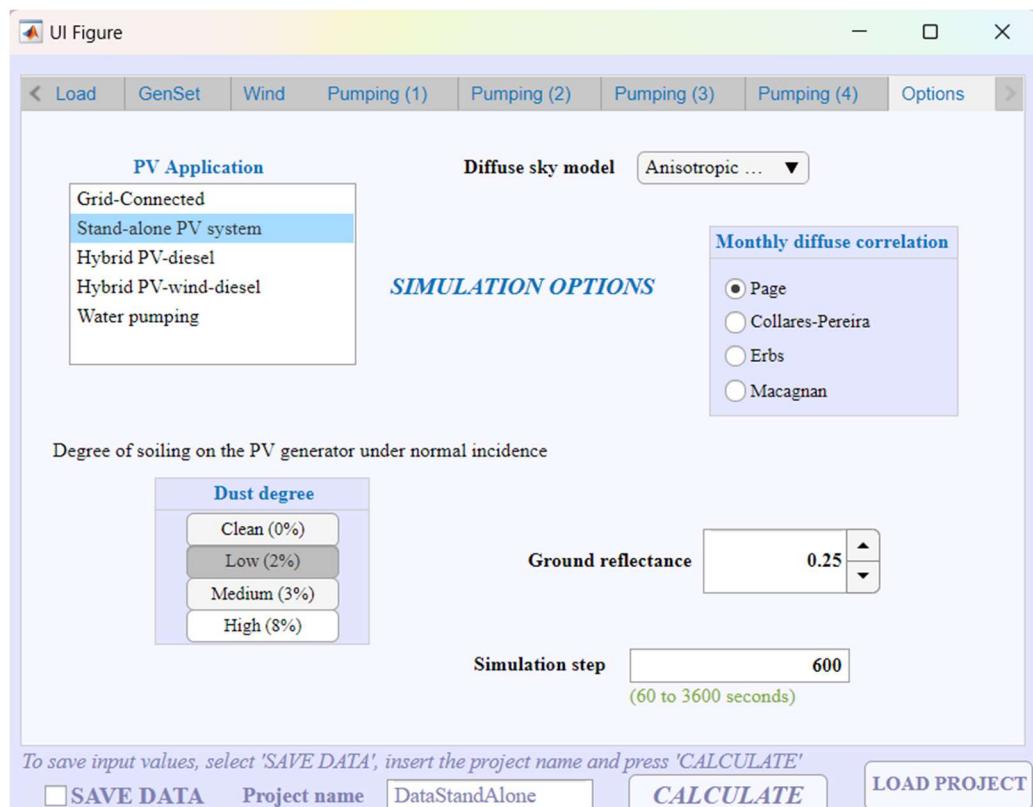


Ilustración 47. Interfaz en funcionamiento en la pestaña Options

Capítulo 3

También, se muestra en la Ilustración 48 un ejemplo de lo que ocurriría en caso de que se intente escribir un valor que esté fuera del rango permitido para una determinada variable. En este caso, la inclinación del generador debe ser entre 0 y 90°, se pretende poner 100°, y no solo no coge el valor (al pulsar intro, vuelve al valor anterior), si no que salta un mensaje de error en el que se muestra el rango permitido para esta variable.

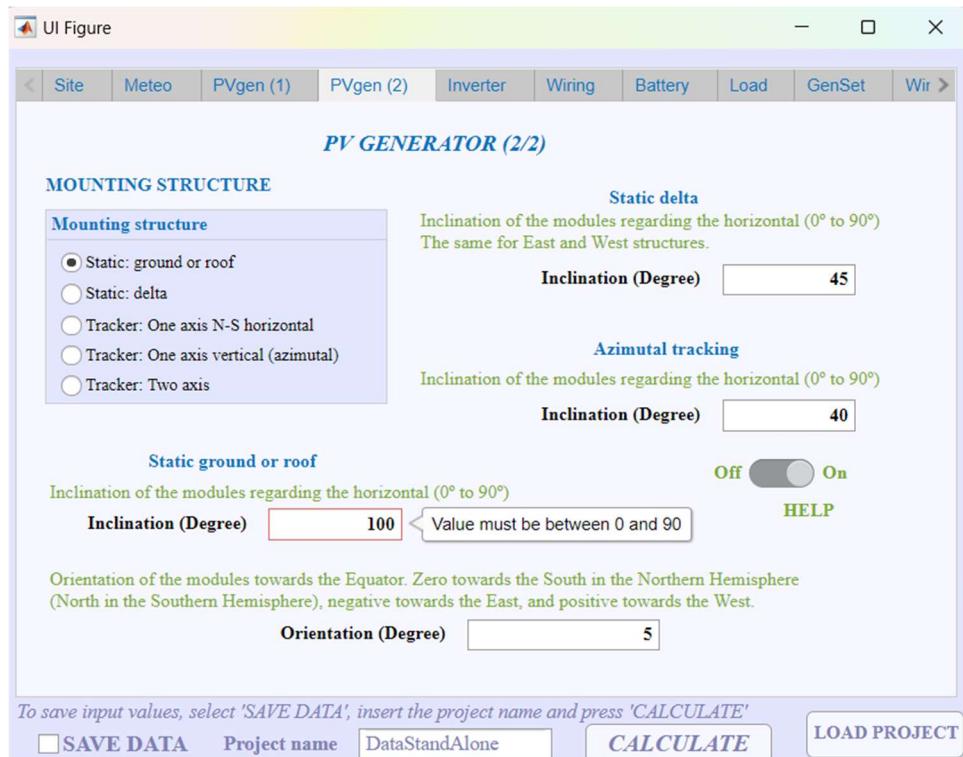


Ilustración 48. Error al introducir un valor fuera del rango permitido

Por último, para ver el funcionamiento completo del programa, se pulsa **CALCULATE**, y con los valores de entrada de las imágenes anteriores se obtienen los siguientes resultados (representados en la Ilustración 49). Estos resultados y gráficas que se obtienen serán distintos en función de las variables que se introduzcan en la interfaz.

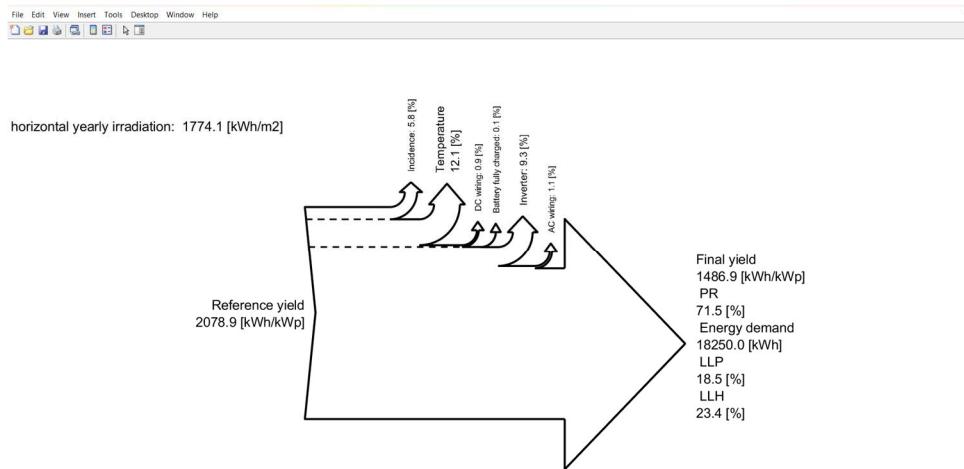


Ilustración 49. Resultados de la simulación para determinados parámetros, aplicación tipo Stand Alone

A continuación, para ver las distintas opciones, se ha modificado la variable relacionada con el tipo de aplicación a un tipo *Water Pumping* dejando todas las demás variables igual excepto las potencias del generador fotovoltaico y del inversor ya que serían muy grandes para este tipo de aplicación. Se pretende únicamente, ver como varían los resultados cambiando algunos de sus parámetros, principalmente el tipo de aplicación, siendo ahora uno de Bombeo de agua. Como es un nuevo proyecto y se desean guardar estos datos, se selecciona la casilla de *Save Data* y se introduce el nombre del nuevo proyecto con el que se pretende guardar, se pulsa *CALCULATE* y se obtienen los resultados. Se muestran imágenes de todo lo descrito en este párrafo en las siguientes Ilustraciones 50, 51, 52 y 53.

Capítulo 3

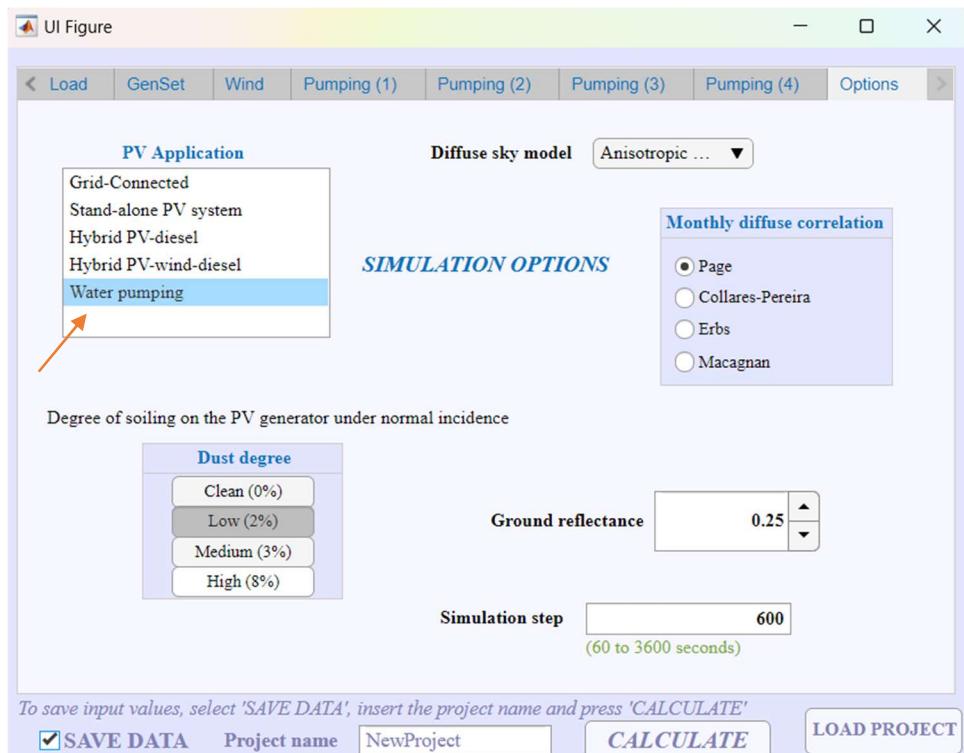


Ilustración 50. Modificaciones del nuevo Proyecto (1)

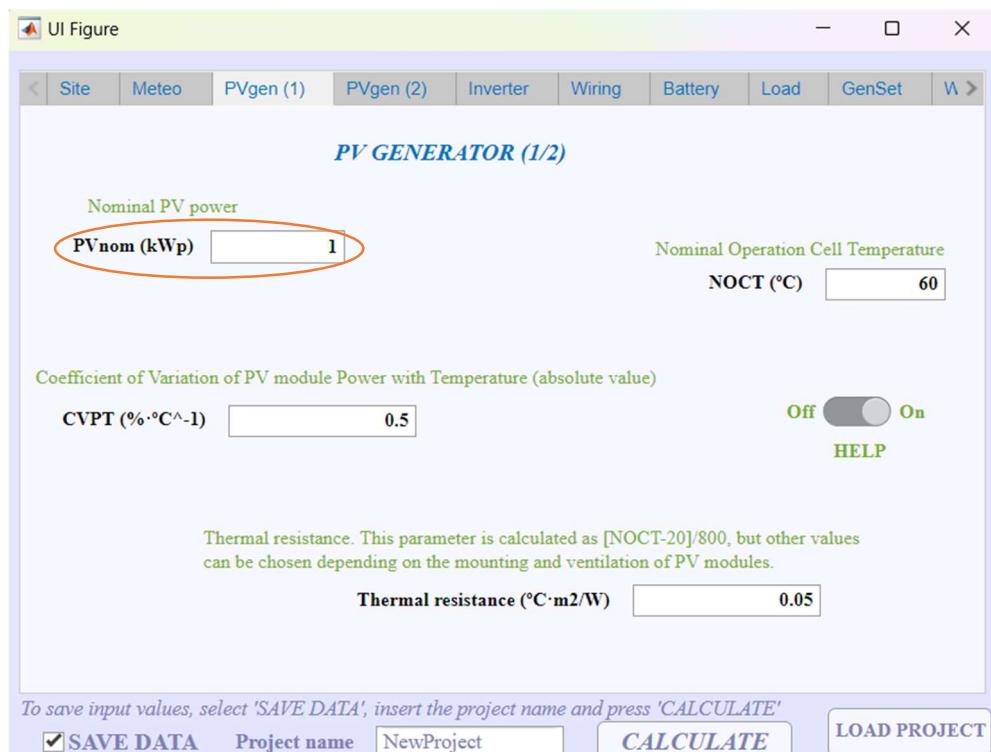


Ilustración 51. Modificaciones del nuevo Proyecto (2)

Desarrollo de la interfaz

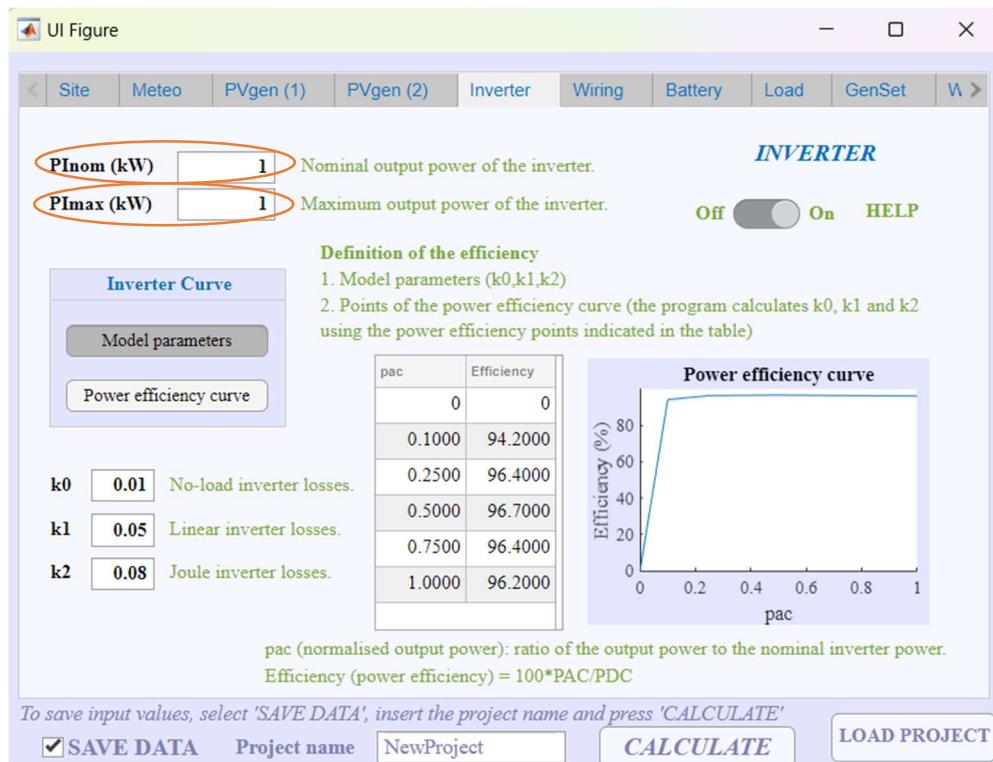


Ilustración 52. Modificaciones del nuevo Proyecto (3)

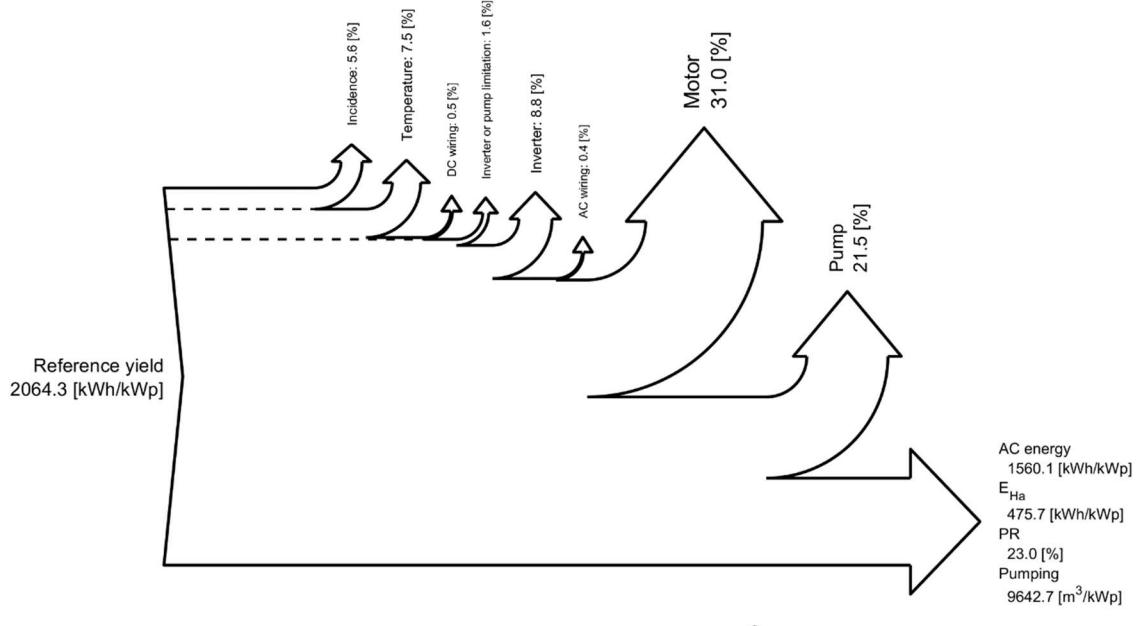


Ilustración 53. Resultados para el nuevo proyecto (Water Pumping)

Capítulo 4. CONCLUSIÓN Y LÍNEAS FUTURAS

4.1. CONCLUSIÓN

En este apartado se mencionan las principales conclusiones tras la finalización de este proyecto destinado al desarrollo de una interfaz gráfica de usuario para un simulador de sistemas fotovoltaicos.

App Designer es una herramienta que cumple con todas las utilidades necesarias para la creación de una interfaz gráfica intuitiva y sencilla con una amplia gama de posibilidades a la hora de diseñar dicha interfaz. Tiene la opción de comunicar el código implementado en App Designer (destinado a la creación de la interfaz) con *scripts* implementados en Matlab (destinados a cualquier otra función en el programa), lo que hace que incremente su productividad.

Una vez se tiene toda la información para empezar a programar en esta herramienta, resulta sencillo crear cada componente, acceder a él y definir sus características. También es cierto que, al tratarse de una plataforma relativamente nueva, la ayuda e información es escasa, lo que hace más difícil aprender a utilizarla de

manera fluida. En otras plataformas se pueden encontrar muchos artículos, tutoriales, etc. donde se ayuda con el aprendizaje de cada herramienta.

En cuanto a la interfaz, se puede decir que se han cumplido los objetivos propuestos al inicio del proyecto. Se ha obtenido una interfaz gráfica que permite comunicar al usuario con el programa de simulación de sistemas fotovoltaicos PVLite sin perder ningún tipo de información ni utilidad (propia del simulador) de una manera sencilla, intuitiva y rápida. Además de la posibilidad de modificar cada variable en el programa, actualizar las gráficas relacionadas con ellas al momento, crear mensajes instructivos con la explicación de cada parámetro y definición de rangos permitidos para las variables que así lo requieren, entre otros, se han conseguido otras funcionalidades muy útiles a la hora del estudio de proyectos relacionados con este tipo de sistemas. Estas funcionalidades son, principalmente, la posibilidad de cargar proyectos en cualquier momento con la interfaz ya abierta, crear nuevos y guardar tanto los nuevos como los ya existentes para su posterior consulta, continuación o simple depósito.

4.2. LÍNEAS FUTURAS

El programa de simulación PVLite podría evolucionar con el estudio de nuevos parámetros o la obtención de distintas soluciones, por lo que, si esto ocurre, la interfaz deberá evolucionar con él. En este documento, como anexo, se adjunta un breve tutorial por si esto pasara (Anexo C. Tutorial para agregar o eliminar componentes de la interfaz). En este tutorial se define de manera directa y sencilla qué pasos habría que seguir en caso de necesitar agregar o eliminar un componente de esta interfaz y que siga funcionando correctamente todo en su conjunto.

Como posibles mejoras del programa de simulación junto con la interfaz, se podría implementar la opción de guardar proyectos en un archivo más completo como un pdf, donde se especifique cada variable con su definición y su valor específico, la impresión de los resultados y una serie de datos como el autor del proyecto, fecha, etc. Esto sería de gran utilidad para realizar estudios de sistemas fotovoltaicos en ambientes más profesionales. Otra posible mejora podría ser la importación de datos meteorológicos de bases de datos de internet, como ya hacen otros simuladores de este tipo de sistemas mencionados en el Capítulo 1.

BIBLIOGRAFÍA

- [1] "PVsyst – Photovoltaic software". PVsyst – Photovoltaic software. Accedido en octubre de 2023. [En línea]. Disponible: <https://www.pvsyst.com/>
- [2] "PVsyst 7 _ Presentación del Software PVsyst". (4 de agosto de 2022). Accedido en octubre de 2023. [Vídeo en línea]. Disponible: <https://www.youtube.com/watch?v=J4dyNv-PApM>
- [3] "HOMER - Hybrid Renewable and Distributed Generation System Design Software". HOMER - Hybrid Renewable and Distributed Generation System Design Software. Accedido en octubre de 2023. [En línea]. Disponible: <https://www.homerenergy.com/index.html>
- [4] "Uniajc Virtual. HOMERO parte 1". (10 de junio de 2020). Accedido en octubre de 2023. [Vídeo en línea]. Disponible: <https://www.youtube.com/watch?v=YJEY9ycPhsE>
- [5] "Home - System Advisor Model - SAM". Home - System Advisor Model - SAM. Accedido en octubre de 2023. [En línea]. Disponible: <https://sam.nrel.gov/>
- [6] "Modelo de Asesor del Sistema. Modelado de sistemas fotovoltaicos a escala de servicios públicos en SAM." (23 de agosto de 2023). Accedido en octubre de 2023. [Vídeo en línea]. Disponible: https://www.youtube.com/watch?v=Z_2WuVkECjY
- [7] "PVWatts Calculator". PVWatts Calculator. Accedido en octubre de 2023. [En línea]. Disponible: <https://pvwatts.nrel.gov/>
- [8] "pvlib". PyPI. Accedido en octubre de 2023. [En línea]. Disponible: <https://pypi.org/project/pvlib/>
- [9] "Desarrollar apps mediante App Designer- MATLAB & Simulink- MathWorks España". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. Accedido en octubre de 2023. [En línea]. Disponible: <https://es.mathworks.com/help/matlab/app-designer.html>
- [10] "Foro de Matlab". La Web del Programador. Accedido en octubre de 2023. [En línea]. Disponible: <https://www.lawebdelprogramador.com/foros/Matlab/index1.html>

ANEXO A. CÓDIGO IMPLEMENTADO EN APP DESIGNER

```

% appInterface1
%
classdef appInterface1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                               matlab.ui.Figure
        TabGroup                                matlab.ui.container.TabGroup
        Site                                    matlab.ui.container.Tab
        GEOGRAPHICALDATALabel                  matlab.ui.control.Label
        LatitudeDegreeLabel                   matlab.ui.control.Label
        LatitudeI                             matlab.ui.control.NumericEditField
        LongitudeDegreeLabel                 matlab.ui.control.Label
        LongitudeI                           matlab.ui.control.NumericEditField
        AltitudemeterLabel                   matlab.ui.control.Label
        AltitudeI                            matlab.ui.control.NumericEditField
        StandardLongitudeDegreeLabel         matlab.ui.control.Label
        StandardLongitudeI                  matlab.ui.control.Spinner
        LatitudeLabel                         matlab.ui.control.Label
        StLongitudeLabel                     matlab.ui.control.Label
        LongitudeLabel                        matlab.ui.control.Label
        AltitudeLabel                         matlab.ui.control.Label
        HELPSwitchLabel_2                    matlab.ui.control.Label
        HELPSite                              matlab.ui.control.Switch
        Meteo                                  matlab.ui.container.Tab
        MeteorologicalinputdataListBoxLabel matlab.ui.control.Label
        InputDataI                            matlab.ui.control.ListBox
        TimeSeriesDropDownLabel              matlab.ui.control.Label
        TimeSeriesI                           matlab.ui.control.DropDown
        METEOROLOGICALINPUTDATALabel        matlab.ui.control.Label
        MetDataI                             matlab.ui.control.Table
        Gdm0Label1                           matlab.ui.control.Label
        Gdm0Label2                           matlab.ui.control.Label
        TmmLabel                             matlab.ui.control.Label
        TMmLabel                            matlab.ui.control.Label
        HELPSwitchLabel_3                    matlab.ui.control.Label
        HELPMeteo                            matlab.ui.control.Switch
        PVgenerator                          matlab.ui.container.Tab
        PVnomkWpEditFieldLabel             matlab.ui.control.Label
        PVnomI                               matlab.ui.control.NumericEditField
        CVPTC1Label                         matlab.ui.control.Label
        CVPTI                                matlab.ui.control.NumericEditField
        NOCTCLabel                          matlab.ui.control.Label
        NOCTI                                matlab.ui.control.NumericEditField
        NOCTLabel                           matlab.ui.control.Label
        ThermalresistanceCm2WLabel        matlab.ui.control.Label
        RthI                                 matlab.ui.control.NumericEditField
        PVnomLabel                          matlab.ui.control.Label
        CVPTLabel                           matlab.ui.control.Label
        RthLabel1                           matlab.ui.control.Label
        RthLabel2                           matlab.ui.control.Label
        PVGENERATOR12Label                 matlab.ui.control.Label
        HELPSwitchLabel_4                    matlab.ui.control.Label
        HELPPVgen                            matlab.ui.control.Switch
        PVqenenerator2                      matlab.ui.container.Tab

```

```
InclinationDegreeLabel matlab.ui.control.Label
inclineGround matlab.ui.control.NumericEditField
InclinationStaticLabel matlab.ui.control.Label
OrientationDegreeLabel matlab.ui.control.Label
orientationI matlab.ui.control.NumericEditField
OrientationLabel1 matlab.ui.control.Label
OrientationLabel2 matlab.ui.control.Label
PVGENERATOR22Label matlab.ui.control.Label
MOUNTINGSTRUCTURELabel matlab.ui.control.Label
MOUNTING matlab.ui.container.ButtonGroup
Staticground matlab.ui.control.RadioButton
Staticdelta matlab.ui.control.RadioButton
TrackerHorizontal matlab.ui.control.RadioButton
TrackerVertical matlab.ui.control.RadioButton
TrackerTwo matlab.ui.control.RadioButton
StaticgroundorroofLabel matlab.ui.control.RadioButton
StaticdeltaLabel matlab.ui.control.RadioButton
AzimutaltrackingLabel matlab.ui.control.RadioButton
InclinationDegreeLabel_2 matlab.ui.control.RadioButton
inclineI_delta matlab.ui.control.NumericEditField
InclinationDegreeLabel_3 matlab.ui.control.RadioButton
inclineI_tracking matlab.ui.control.NumericEditField
InclinationDeltaLabel1 matlab.ui.control.RadioButton
InclinationDeltaLabel2 matlab.ui.control.RadioButton
InclinationTrackingLabel matlab.ui.control.RadioButton
HELPSwitchLabel_12 matlab.ui.control.RadioButton
HELPPVgen_2 matlab.ui.control.Switch
Inverter matlab.ui.container.Tab
PINomkWEeditFieldLabel matlab.ui.control.Label
PINomI matlab.ui.control.NumericEditField
PImaxkWEeditFieldLabel matlab.ui.control.Label
PImaxI matlab.ui.control.NumericEditField
NominaloutputLabel matlab.ui.control.Label
MaximumoutputLabel matlab.ui.control.Label
InverterCurveI matlab.ui.container.ButtonGroup
ModelparametersButton matlab.ui.control.ToggleButton
PowerEfficiencycurveButton matlab.ui.control.ToggleButton
k0EditFieldLabel matlab.ui.control.Label
k0I matlab.ui.control.NumericEditField
k1EditFieldLabel matlab.ui.control.Label
k1I matlab.ui.control.NumericEditField
k2EditFieldLabel matlab.ui.control.Label
k2I matlab.ui.control.NumericEditField
pointsTable matlab.ui.control.Table
PowerEfgraphic matlab.ui.control.UIAxes
INVERTERLabel matlab.ui.control.Label
efficiencyLabel1 matlab.ui.control.Label
efficiencyLabel2 matlab.ui.control.Label
efficiencyLabel3 matlab.ui.control.Label
efficiencyLabel4 matlab.ui.control.Label
k0Label matlab.ui.control.Label
k1Label matlab.ui.control.Label
k2Label matlab.ui.control.Label
pacLabel matlab.ui.control.Label
EfficiencyLabel matlab.ui.control.Label
HELPSwitchLabel_8 matlab.ui.control.Label
HELPInverter matlab.ui.control.Switch
Wiring matlab.ui.container.Tab
```

WDCLabel	matlab.ui.control.Label
WDCI	matlab.ui.control.NumericEditField
WACLabel	matlab.ui.control.Label
WACI	matlab.ui.control.NumericEditField
DCLabel	matlab.ui.control.Label
ACLabel	matlab.ui.control.Label
WIRINGLabel	matlab.ui.control.Label
Bat	matlab.ui.container.Tab
CBATkWhLabel	matlab.ui.control.Label
CBATI	matlab.ui.control.NumericEditField
SOCmaxEditFieldLabel	matlab.ui.control.Label
SOCmaxI	matlab.ui.control.NumericEditField
SOCminEditFieldLabel	matlab.ui.control.Label
SOCmini	matlab.ui.control.NumericEditField
SOCmaxLabel	matlab.ui.control.Label
SOCminLabel	matlab.ui.control.Label
CBATLabel	matlab.ui.control.Label
BATTERYStandalonePVsystemsLabel	matlab.ui.control.Label
HELPswitchLabel_5	matlab.ui.control.Label
HELPBattery	matlab.ui.control.ToggleSwitch
Load	matlab.ui.container.Tab
LdmGraphic	matlab.ui.control.UIAxes
LdmI	matlab.ui.control.Table
TotalEditFieldLabel	matlab.ui.control.Label
TotalLdmI	matlab.ui.control.NumericEditField
LdmLabel	matlab.ui.control.Label
FhI	matlab.ui.control.Table
FhGraphic	matlab.ui.control.UIAxes
FhLabel1	matlab.ui.control.Label
FhLabel2	matlab.ui.control.Label
FhLabel3	matlab.ui.control.Label
FhLabel4	matlab.ui.control.Label
TotalEditFieldLabel_2	matlab.ui.control.Label
TotalFh	matlab.ui.control.NumericEditField
HELPswitchLabel_6	matlab.ui.control.Label
HELPload	matlab.ui.control.Switch
LOADPROFILESLlabel	matlab.ui.control.Label
GenSet	matlab.ui.container.Tab
PGENnomkWLabel	matlab.ui.control.Label
PGENnomI	matlab.ui.control.NumericEditField
SOCstartEditFieldLabel	matlab.ui.control.Label
SOCstartI	matlab.ui.control.NumericEditField
SOCstopEditFieldLabel	matlab.ui.control.Label
SOCstopI	matlab.ui.control.NumericEditField
b0iEditFieldLabel	matlab.ui.control.Label
b0iI	matlab.ui.control.NumericEditField
b1iEditFieldLabel	matlab.ui.control.Label
b1iI	matlab.ui.control.NumericEditField
b0sEditFieldLabel	matlab.ui.control.Label
b0sI	matlab.ui.control.NumericEditField
b1sEditFieldLabel	matlab.ui.control.Label
b1sI	matlab.ui.control.NumericEditField
b0iUnitsLabel	matlab.ui.control.Label
b1iUnitsLabel	matlab.ui.control.Label
b0sUnitsLabel	matlab.ui.control.Label
b1sUnitsLabel	matlab.ui.control.Label
PGENnomLabel	matlab.ui.control.Label
SOCstartLabel	matlab.ui.control.Label

SOCstopLabel	matlab.ui.control.Label
FuelCons	matlab.ui.control.Table
FuelConsGraph	matlab.ui.control.UIAxes
b0iLabel	matlab.ui.control.Label
b1iLabel	matlab.ui.control.Label
b0sLabel	matlab.ui.control.Label
b1sLabel	matlab.ui.control.Label
HELPswitchLabel	matlab.ui.control.Label
HELPGenSet	matlab.ui.control.Switch
GENERATORSET	matlab.ui.control.Label
Wind	matlab.ui.container.Tab
WIND	matlab.ui.control.Label
PWnomkWEeditFieldLabel	matlab.ui.control.Label
PWnomI	matlab.ui.control.NumericEditField
VnommsEditFieldLabel	matlab.ui.control.Label
VnomI	matlab.ui.control.NumericEditField
VcimsEditFieldLabel	matlab.ui.control.Label
VciI	matlab.ui.control.NumericEditField
VcomsEditFieldLabel	matlab.ui.control.Label
VcoI	matlab.ui.control.NumericEditField
CpeqkWms3EditFieldLabel	matlab.ui.control.Label
CpeqI	matlab.ui.control.NumericEditField
PWnomLabel	matlab.ui.control.Label
VnomLabel	matlab.ui.control.Label
VciLabel	matlab.ui.control.Label
VcoLabel	matlab.ui.control.Label
CpeqLabel	matlab.ui.control.Label
HELPswitchLabel_7	matlab.ui.control.Label
HELPWind	matlab.ui.control.ToggleSwitch
PowerCurve	matlab.ui.control.Table
PowerCurveGraph	matlab.ui.control.UIAxes
Pumping1	matlab.ui.container.Tab
PUMPLabel	matlab.ui.control.Label
Qratedm3hEditFieldLabel	matlab.ui.control.Label
QratedI	matlab.ui.control.NumericEditField
HratedmEditFieldLabel	matlab.ui.control.Label
HratedI	matlab.ui.control.NumericEditField
LiquidDensitykgm3EditFieldLabel	matlab.ui.control.Label
LiquidDensityI	matlab.ui.control.NumericEditField
QratedLabel	matlab.ui.control.Label
HratedLabel	matlab.ui.control.Label
densityLabel	matlab.ui.control.Label
WELLBOREHOLELabel	matlab.ui.control.Label
HsmLabel	matlab.ui.control.Label
HsI	matlab.ui.control.NumericEditField
Qtestm3hLabel	matlab.ui.control.Label
QtestI	matlab.ui.control.NumericEditField
HdrmLabel	matlab.ui.control.Label
HdrI	matlab.ui.control.NumericEditField
kw1mm3hLabel	matlab.ui.control.Label
kw1I	matlab.ui.control.NumericEditField
kw2mm3h2Label	matlab.ui.control.Label
kw2I	matlab.ui.control.NumericEditField
HsLabel	matlab.ui.control.Label
QtestLabel	matlab.ui.control.Label
HdrLabel	matlab.ui.control.Label
kw1Label1	matlab.ui.control.Label
kw1Label2	matlab.ui.control.Label

kw2Label	matlab.ui.control.Label
WATERTANKLabel	matlab.ui.control.Label
WTClm3Label	matlab.ui.control.Label
WTCl	matlab.ui.control.NumericEditField
HrmLabel	matlab.ui.control.Label
HrI	matlab.ui.control.NumericEditField
WTCLabel	matlab.ui.control.Label
WTCLabel2	matlab.ui.control.Label
HrLabel	matlab.ui.control.Label
PUMPING1	matlab.ui.control.Label
HELPswitchLabel_13	matlab.ui.control.Label
HELPPump1	matlab.ui.control.ToggleSwitch
Pumping2	matlab.ui.container.Tab
PUMPING2	matlab.ui.control.Label
ks0mLabel	matlab.ui.control.Label
ks0I	matlab.ui.control.NumericEditField
ks1mm3hLabel	matlab.ui.control.Label
ks1I	matlab.ui.control.NumericEditField
HfmLabel	matlab.ui.control.Label
HfI	matlab.ui.control.NumericEditField
ks0Label	matlab.ui.control.Label
ks1Label	matlab.ui.control.Label
HfLabel	matlab.ui.control.Label
SYSTEMCURVELabel	matlab.ui.control.Label
systemCurveTable	matlab.ui.control.Table
systemCurveGraph	matlab.ui.control.UIAxes
HELPswitchLabel_9	matlab.ui.control.Label
HELPPump2	matlab.ui.control.ToggleSwitch
ks2mm3h2Label	matlab.ui.control.Label
ks2I	matlab.ui.control.NumericEditField
ks2Label	matlab.ui.control.Label
Pumping3	matlab.ui.container.Tab
pumpCurveTable	matlab.ui.control.Table
pumpCurveGraph	matlab.ui.control.UIAxes
powerCurvesTable	matlab.ui.control.Table
powerCurvesGraph	matlab.ui.control.UIAxes
P1Label1	matlab.ui.control.Label
P1Label2	matlab.ui.control.Label
P2Label1	matlab.ui.control.Label
P2Label2	matlab.ui.control.Label
PUMPING3	matlab.ui.control.Label
PUMPCURVELabel	matlab.ui.control.Label
POWERCURVESLabel	matlab.ui.control.Label
HELPswitchLabel_10	matlab.ui.control.Label
HELPPump3	matlab.ui.control.ToggleSwitch
Pumping4	matlab.ui.container.Tab
PUMPING4	matlab.ui.control.Label
MOTORLabel	matlab.ui.control.Label
P2ratedkWEeditFieldLabel	matlab.ui.control.Label
P2ratedI	matlab.ui.control.NumericEditField
RPNomrpmLabel	matlab.ui.control.Label
RPNomI	matlab.ui.control.NumericEditField
RPMax10100Label	matlab.ui.control.Label
RPMax1I	matlab.ui.control.NumericEditField
RPMax100150Label	matlab.ui.control.Label
RPMax1I	matlab.ui.control.NumericEditField
P2RatedLabel	matlab.ui.control.Label
RPNomLabel	matlab.ui.control.Label

```

RPMcoolLabel matlab.ui.control.Label
RPMmaxLabel matlab.ui.control.Label
PowerEfficTable matlab.ui.control.Table
PowerEfficGraph matlab.ui.control.UIAxes
HELPSwitchLabel_11 matlab.ui.control.Label
HELPPump4 matlab.ui.control.ToggleSwitch
Options matlab.ui.container.Tab
PVApplicationLabel matlab.ui.control.Label
ApplicationI matlab.ui.control.ListBox
DiffuseskymodelDropDownLabel matlab.ui.control.Label
DiffuseModel matlab.ui.control.DropDown
DustLabel matlab.ui.control.Label
DustDegreeI matlab.ui.container.ButtonGroup
Clean matlab.ui.control.ToggleButton
Low matlab.ui.control.ToggleButton
Medium matlab.ui.control.ToggleButton
High matlab.ui.control.ToggleButton
GroundreflectanceSpinnerLabel matlab.ui.control.Label
GroundReflectanceI matlab.ui.control.Spinner
DiffuseFractionI matlab.ui.container.ButtonGroup
Page matlab.ui.control.RadioButton
CollaresPereira matlab.ui.control.RadioButton
Erbs matlab.ui.control.RadioButton
Macagnan matlab.ui.control.RadioButton
SimulationstepEditFieldLabel matlab.ui.control.Label
SimulationstepI matlab.ui.control.NumericEditField
to3600secondsLabel matlab.ui.control.Label
SIMULATIONOPTIONSLabel matlab.ui.control.Label
calculate matlab.ui.control.Button
SaveData matlab.ui.control.CheckBox
instructions matlab.ui.control.Label
ProjectnameLabel matlab.ui.control.Label
nameI matlab.ui.control.EditField
loadProject matlab.ui.control.Button
end

properties (Access = private)
    %coefficients for the calculation of F(l/h) depending on b0i, bli, PGENnom
    coef1;
    coef2;

end

% Callbacks that handle component events
methods (Access = private)

    % Code that executes after component creation
    function startupFcn(app)
        %Open a window to select the input data .mat file (Workspace in
        %Matlab)
        file1=uigetfile('*.*mat','Select the input data file');
        % If the file is not selected, or the window is closed, the program ↵
        stops.
        if isequal(file1,0)
            error('The file has not been selected');
        end

```

```

%Load the workspace data
load(file1);

%Take each value from the workspace and assign it to each component
app.PInomI.Value=PInom;
app.PImaxI.Value=PImax;
app.k0I.Value=k0;
app.k1I.Value=k1;
app.k2I.Value=k2;
app.pointsTable.ColumnFormat={ 'numeric', 'numeric' };
app.pointsTable.Data=points;
app.GroundReflectanceI.Value=GroundReflectance;
app.SimulationstepI.Value=SimulationStep;
app.LatitudeI.Value=Latitude;
app.LongitudeI.Value=Longitude;
app.AltitudeI.Value=Altitude;
app.StandardLongitudeI.Value=StandardLongitude;
app.MetDataI.ColumnFormat={ 'numeric', 'numeric', 'numeric' };
app.MetDataI.Data=months_Data;
app.PVnomI.Value=PVnom;
app.CVPTI.Value=CVPT;
app.NOCTI.Value=NOCT;
app.inclinationGround.Value=InclinationGround;
app.orientationI.Value=Orientation;
app.WACI.Value=WAC;
app.WDCI.Value=WDC;
app.CBATI.Value=CBAT;
app.SOCmaxI.Value=SOCmax;
app.SOCminI.Value=SOCmin;
app.LdmI.ColumnFormat={ 'numeric' };
app.LdmI.Data=Ldm_Data;
app.TotalLdmI.Value=Edemanda;
app.FhI.ColumnFormat={ 'numeric' };
app.FhI.Data=F_Data;
app.PGENnomI.Value=PGENnom;
app.SOCstartI.Value=SOCstart;
app.SOCstopI.Value=SOCstop;
app.b0iI.Value=b0i;
app.b1iI.Value=b1i;
app.b0sI.Value=b0s;
app.b1sI.Value=b1s;
app.PWnomI.Value=PWnom;
app.VnomI.Value=Vnom;
app.VciI.Value=Vci;
app.VcoI.Value=Vco;
app.QratedI.Value=Qrated;
app.HratedI.Value=Hrated;
app.LiquidDensityI.Value=Density;
app.HfI.Value=Hf;
app.pumpCurveTable.ColumnFormat={ 'numeric', 'numeric' };
app.pumpCurveTable.Data=PumpCurve;
app.powerCurvesTable.ColumnFormat={ 'numeric', 'numeric', 'numeric' };
app.powerCurvesTable.Data=PowerCurves;
app.P2ratedI.Value=P2rated;
app.RPMnomI.Value=RPMnom;
app.RPMcoolI.Value=RPMcoolM;
app.RPMmaxI.Value=RPMmaxM;

```

```

app.PowerEfficTable.ColumnFormat={ 'numeric', 'numeric'};
app.PowerEfficTable.Data=PowerEffic;
app.nameI.Value=name;
app.inclinationI_delta.Value=InclinationDelta;
app.inclinationI_tracking.Value=InclinationTracking;
app.HsI.Value=Hs;
app.QtestI.Value=Qtest;
app.HdrI.Value=Hdr;
app.kw1I.Value=kw1;
app.kw2I.Value=kw2;
app.WTCI.Value=WTC;
app.HrI.Value=Hr;

if(InverterCurve==1)
    app.ModelparametersButton.Value=true;
    app.PowerefficiencycurveButton.Value=false;
    assignin("base","InverterCurve",1);
else
    app.PowerefficiencycurveButton.Value=true;
    app.ModelparametersButton.Value=false;
    assignin("base","InverterCurve",2);
end

switch Application
case 1
    app.ApplicationI.Value='Grid-Connected';
    assignin("base","Application",1);
case 2
    app.ApplicationI.Value='Stand-alone PV system';
    assignin("base","Application",2);
case 3
    app.ApplicationI.Value='Hybrid PV-diesel';
    assignin("base","Application",3);
case 4
    app.ApplicationI.Value='Hybrid PV-wind-diesel';
    assignin("base","Application",4);
case 5
    app.ApplicationI.Value='Water pumping';
    assignin("base","Application",5);
end

if(DustDegree==1)
    app.Clean.Value=true;
    app.Low.Value=false;
    app.Medium.Value=false;
    app.High.Value=false;
    assignin("base","DustDegree",1);
elseif(DustDegree==2)
    app.Low.Value=true;
    app.Medium.Value=false;
    app.High.Value=false;
    app.Clean.Value=false;
    assignin("base","DustDegree",2);
elseif(DustDegree==3)
    app.Medium.Value=true;
    app.High.Value=false;
    app.Clean.Value=false;
    app.Low.Value=false;

```

```

        assignin("base", "DustDegree", 3);
    elseif (DustDegree==4)
        app.High.Value=true;
        app.Clean.Value=false;
        app.Low.Value=false;
        app.Medium.Value=false;
        assignin("base", "DustDegree", 4);
    end

    switch Diffuse_model
    case 1
        app.DiffuseModel.Value='Isotropic';
        assignin("base", "Diffuse_model",1);
    case 2
        app.DiffuseModel.Value='Anisotropic (Hay)';
        assignin("base", "Diffuse_model",2);
    case 3
        app.DiffuseModel.Value='Anisotropic (Perez)';
        assignin("base", "Diffuse_model",3);
    end

    if(Diffuse_fraction==1)
        app.Page.Value=true;
        assignin("base", "Diffuse_fraction",1);
    elseif(Diffuse_fraction==2)
        app.CollaresPereira.Value=true;
        assignin("base", "Diffuse_fraction",2);
    elseif(Diffuse_fraction==3)
        app.Erbs.Value=true;
        assignin("base", "Diffuse_fraction",3);
    else
        app.Macagnan.Value=true;
        assignin("base", "Diffuse_fraction",4);
    end

    if(Mounting==1)
        app.Staticground.Value=true;
        assignin("base", "Mounting",1);
    elseif(Mounting==2)
        app.Staticdelta.Value=true;
        assignin("base", "Mounting",2);
    elseif(Mounting==3)
        app.TrackerHorizontal.Value=true;
        assignin("base", "Mounting",3);
    elseif(Mounting==4)
        app.TrackerVertical.Value=true;
        assignin("base", "Mounting",4);
    else
        app.TrackerTwo.Value=true;
        assignin("base", "Mounting",5);
    end

    switch InputData
    case 1
        app.InputDataI.Value='Monthly averages from the table';
        assignin("base", "InputData",1);
    case 2

```

```

        app.InputDataI.Value='Monthly averages obtained from PVGIS TMY';
        assignin("base","InputData",2);
case 3
        app.InputDataI.Value='Hourly values from PVGIS TMY';
        assignin("base","InputData",3);
case 4
        app.InputDataI.Value='Monthly averages obtained from USA TMY3';
        assignin("base","InputData",4);
case 5
        app.InputDataI.Value='Hourly values from USA TMY3';
        assignin("base","InputData",5);
end

switch TimeSeries
case 1
        app.TimeSeriesI.Value='Mean Days';
        assignin("base","TimeSeries",1);
case 2
        app.TimeSeriesI.Value='Aguiar';
        assignin("base","TimeSeries",2);
end

app.RthI.Value=(app.NOCTI.Value-20)/800;

app.ks0I.Value = app.HrI.Value + app.HsI.Value;

if(app.HfI.Value==0 || app.QratedI.Value==0)
    app.ks2I.Value=app.kw2I.Value;
else
    app.ks2I.Value=(app.HfI.Value/(app.QratedI.Value^2))+app.kw2I. ↵
Value;
end

if(app.PWnomI.Value==0 || app.VnomI.Value==0 || app.VciI.Value==0)
    app.CpeqI.Value = 0;
else
    app.CpeqI.Value = app.PWnomI.Value/(app.VnomI.Value^3-app.VciI. ↵
Value^3);
end

app.coef1=(app.b0iI.Value+app.b1iI.Value*app.PGENnomI.Value);
app.coef2=(app.b0sI.Value+app.b1sI.Value*app.PGENnomI.Value);
app.FuelCons.Data=[0 app.coef1+app.coef2*0; ... ↵
0.25*app.PGENnomI.Value app.coef1+app.coef2*0.25*app.PGENnomI. ↵
Value; ... ↵
0.5*app.PGENnomI.Value app.coef1+app.coef2*0.5*app.PGENnomI.Value; ↵
...
0.75*app.PGENnomI.Value app.coef1+app.coef2*0.75*app.PGENnomI. ↵
Value; ... ↵
app.PGENnomI.Value app.coef1+app.coef2*app.PGENnomI.Value];

app.PowerCurve.Data=[0 0; app.VciI.Value 0; ... ↵
app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);

```

```

Value^3); ...
    app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ...
    app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ...
    app.VnomI.Value app.PWnomI.Value; app.VcoI.Value app.PWnomI.Value; ↵
app.VcoI.Value 0];

app.TotalFh.Value=sum(app.FhI.Data(1:24,1));

app.systemCurveTable.Data=[0 app.ks0I.Value; ...
    0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
    0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
    0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
    0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
    0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
    0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
    1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
    1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...
    1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];

%Initial graphics
plot(app.LdmGraphic,app.LdmI.Data(1:12,1));
plot(app.FhGraphic,app.FhI.Data(1:24,1));
plot(app.PowerEfgraphic,app.pointsTable.Data(1:6,1),app.pointsTable. ↵
Data(1:6,2));
plot(app.FuelConsGraph,app.FuelCons.Data(1:5,1),app.FuelCons.Data(1: ↵
5,2));
plot(app.PowerCurveGraph,app.PowerCurve.Data(1:9,1),app.PowerCurve.Data ↵
(1:9,2));
plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
systemCurveTable.Data(1:10,2));
plot(app.pumpCurveGraph,app.pumpCurveTable.Data(1:7,1),app. ↵
pumpCurveTable.Data(1:7,2));
plot(app.powerCurvesGraph,app.powerCurvesTable.Data(1:7,1),app. ↵
powerCurvesTable.Data(1:7,2), "m");
plot(app.powerCurvesGraph,app.powerCurvesTable.Data(1:7,1),app. ↵
powerCurvesTable.Data(1:7,3), "b");
plot(app.PowerEfficGraph,app.PowerEfficTable.Data(1:7,1),app. ↵
PowerEfficTable.Data(1:7,2));

%Assign values to the workspace
assignin("base","PInom",app.PInomI.Value);
assignin("base","PImax",app.PImaxI.Value);
assignin("base","k0",app.k0I.Value);

```

```

assignin("base","k1",app.k1I.Value);
assignin("base","k2",app.k2I.Value);
assignin("base","points",app.pointsTable.Data);
assignin("base","GroundReflectance",app.GroundReflectanceI.Value);
assignin("base","SimulationStep",app.SimulationstepI.Value);
assignin("base","Latitude",app.LatitudeI.Value);
assignin("base","Longitude",app.LongitudeI.Value);
assignin("base","Altitude",app.AltitudeI.Value);
assignin("base","StandardLongitude",app.StandardLongitudeI.Value);
assignin("base","months_Data",app.MetDataI.Data);
assignin("base","PVnom",app.PVnomI.Value);
assignin("base","CVPT",app.CVPTI.Value);
assignin("base","NOCT",app.NOCTI.Value);
assignin("base","InclinationGround",app.inclinationGround.Value);
assignin("base","Orientation",app.orientationI.Value);
assignin("base","Rth",app.RthI.Value);
assignin("base","WAC",app.WACI.Value);
assignin("base","WDC",app.WDCI.Value);
assignin("base","CBAT",app.CBATI.Value);
assignin("base","SOCmax",app.SOCmaxI.Value);
assignin("base","SOCmin",app.SOCminI.Value);
assignin("base","Ldm_Data",app.LdmI.Data);
assignin("base","Edemanda",app.TotalLdmI.Value);
assignin("base","F_Data",app.FhI.Data);
assignin("base","PGENnom",app.PGENnomI.Value);
assignin("base","SOCstart",app.SOCstartI.Value);
assignin("base","SOCstop",app.SOCstopI.Value);
assignin("base","b0i",app.b0II.Value);
assignin("base","bli",app.bIII.Value);
assignin("base","b0s",app.b0sI.Value);
assignin("base","b1s",app.b1sI.Value);
assignin("base","PWnom",app.PWnomI.Value);
assignin("base","Vnom",app.VnomI.Value);
assignin("base","Vci",app.VciI.Value);
assignin("base","Vco",app.VcoI.Value);
assignin("base","Qrated",app.QratedI.Value);
assignin("base","Hrated",app.HratedI.Value);
assignin("base","Density",app.LiquidDensityI.Value);
assignin("base","Hf",app.HfI.Value);
assignin("base","PumpCurve",app.pumpCurveTable.Data);
assignin("base","PowerCurves",app.powerCurvesTable.Data);
assignin("base","P2rated",app.P2ratedI.Value);
assignin("base","RPMnom",app.RPMnomI.Value);
assignin("base","RPMcoolM",app.RPMcoolI.Value);
assignin("base","RPMmaxM",app.RPMmaxI.Value);
assignin("base","PowerEffic",app.PowerEfficTable.Data);
assignin("base","saveDataApp",0);
assignin("base","name",app.nameI.Value);
assignin("base","InclinationTracking",app.inclinationI_tracking.Value);
assignin("base","InclinationDelta",app.inclinationI_delta.Value);
assignin("base","Hs",app.HsI.Value);
assignin("base","Qtest",app.QtestI.Value);
assignin("base","Hdr",app.HdrI.Value);
assignin("base","kw1",app.kw1I.Value);
assignin("base","kw2",app.kw2I.Value);
assignin("base","WTC",app.WTCI.Value);
assignin("base","Hr",app.HrI.Value);

```

```

end

% Value changed function: PInomI
function PInomIValueChanged(app, event)
    assignin("base","PInom",app.PInomI.Value);

end

% Value changed function: PImaxI
function PImaxIValueChanged(app, event)
    assignin("base","PImax",app.PImaxI.Value);

end

% Selection changed function: InverterCurveI
function InverterCurveISelectionChanged(app, event)
    if(app.ModelparametersButton.Value==true)
        assignin("base","InverterCurve",1);
    else
        assignin("base","InverterCurve",2);
    end

end

% Cell edit callback: pointsTable
function pointsTableCellEdit(app, event)
    assignin("base","points",app.pointsTable.Data);
    plot(app.PowerEfgraphic,app.pointsTable.Data(1:6,1),app.pointsTable.↙
Data(1:6,2));
end

% Button pushed function: calculate
function calculateButtonPushed(app, event)
    pvlite;
end

% Selection change function: TabGroup
function TabGroupSelectionChanged(app, event)
    selectedTab = app.TabGroup.SelectedTab;

end

% Value changed function: k0I
function k0IValueChanged(app, event)
    assignin("base","k0",app.k0I.Value);

end

% Value changed function: k1I
function k1IValueChanged(app, event)
    assignin("base","k1",app.k1I.Value);

end

% Value changed function: k2I
function k2IValueChanged(app, event)
    assignin("base","k2",app.k2I.Value);

end

```

```

end

% Value changed function: ApplicationI
function ApplicationIValueChanged(app, event)

    switch app.ApplicationI.Value
        case 'Grid-Connected'
            assignin("base", "Application", 1);
        case 'Stand-alone PV system'
            assignin("base", "Application", 2);
        case 'Hybrid PV-diesel'
            assignin("base", "Application", 3);
        case 'Hybrid PV-wind-diesel'
            assignin("base", "Application", 4);
        case 'Water pumping'
            assignin("base", "Application", 5);

    end
end

% Selection changed function: DustDegreeI
function DustDegreeISelectionChanged(app, event)
    if(app.Clean.Value==true)
        assignin("base", "DustDegree", 1);
    elseif(app.Low.Value==true)
        assignin("base", "DustDegree", 2);
    elseif(app.Medium.Value==true)
        assignin("base", "DustDegree", 3);
    elseif (app.High.Value==true)
        assignin("base", "DustDegree", 4);
    end
end

% Value changed function: DiffuseModel
function DiffuseModelValueChanged(app, event)
    switch app.DiffuseModel.Value
        case 'Isotropic'
            assignin("base", "Diffuse_model", 1);
        case 'Anisotropic (Hay)'
            assignin("base", "Diffuse_model", 2);
        case 'Anisotropic (Perez)'
            assignin("base", "Diffuse_model", 3);
    end
end

% Value changed function: GroundReflectanceI
function GroundReflectanceIValueChanged(app, event)
    assignin("base", "GroundReflectance", app.GroundReflectanceI.Value);

end

% Selection changed function: DiffuseFractionI
function DiffuseFractionISelectionChanged(app, event)
    if(app.Page.Value==true)
        assignin("base", "Diffuse fraction", 1);

```

```

elseif(app.CollaresPereira.Value==true)
    assignin("base","Diffuse_fraction",2);
elseif(app.Erbs.Value==true)
    assignin("base","Diffuse_fraction",3);
else
    assignin("base","Diffuse_fraction",4);
end
end

% Value changed function: SimulationstepI
function SimulationstepIValueChanged(app, event)
    assignin("base","SimulationStep",app.SimulationstepI.Value);

end

% Value changed function: LatitudeI
function LatitudeIValueChanged(app, event)
    assignin("base","Latitude",app.LatitudeI.Value);
end

% Value changed function: LongitudeI
function LongitudeIValueChanged(app, event)
    assignin("base","Longitude",app.LongitudeI.Value);
end

% Value changed function: AltitudeI
function AltitudeIValueChanged(app, event)
    assignin("base","Altitude",app.AltitudeI.Value);
end

% Value changed function: StandardLongitudeI
function StandardLongitudeIValueChanged(app, event)
    assignin("base","StandardLongitude",app.StandardLongitudeI.Value);
end

% Value changed function: InputDataI
function InputDataIValueChanged(app, event)
    switch app.InputDataI.Value
        case 'Monthly averages from the table'
            assignin("base","InputData",1);
        case 'Monthly averages obtained from PVGIS TMY'
            assignin("base","InputData",2);
        case 'Hourly values from PVGIS TMY'
            assignin("base","InputData",3);
        case 'Monthly averages obtained from USA TMY3'
            assignin("base","InputData",4);
        case 'Hourly values from USA TMY3'
            assignin("base","InputData",5);
    end
end

% Callback function
function MeteoDataICellEdit(app, event)
    assignin("base","months",app.MeteoDataI.Data);

end

% Callback function

```

```

function MeteoDataICellEdit2(app, event)
    assignin("base", "months", app.MeteoDataI.Data);
end

% Value changed function: TimeSeriesI
function TimeSeriesIValueChanged(app, event)
    switch app.TimeSeriesI.Value
        case 'Mean Days'
            assignin("base", "TimeSeries", 1);
        case 'Aguiar'
            assignin("base", "TimeSeries", 2);
    end
end

% Value changed function: PVnomI
function PVnomIValueChanged(app, event)
    assignin("base", "PVnom", app.PVnomI.Value);
end

% Value changed function: CVPTI
function CVPTIValueChanged(app, event)
    assignin("base", "CVPT", app.CVPTI.Value);
end

% Value changed function: NOCTI
function NOCTIValueChanged(app, event)
    assignin("base", "NOCT", app.NOCTI.Value);
    app.RthI.Value=(app.NOCTI.Value-20)/800;
    assignin("base", "Rth", app.RthI.Value);
end

% Value changed function: orientationI
function orientationIValueChanged(app, event)
    assignin("base", "Orientation", app.orientationI.Value);
end

% Value changed function: RthI
function RthIValueChanged(app, event)
    assignin("base", "Rth", app.RthI.Value);
end

% Value changed function: WDCI
function WDCIValueChanged(app, event)
    assignin("base", "WDC", app.WDCI.Value);
end

% Value changed function: WACI
function WACIValueChanged(app, event)
    assignin("base", "WAC", app.WACI.Value);
end

% Value changed function: CBATI
function CBATIValueChanged(app, event)
    assignin("base", "CBAT", app.CBATI.Value);
end

% Value changed function: SOCmaxI

```

```

function SOCmaxIValueChanged(app, event)
    assignin("base", "SOCmax", app.SOCmaxI.Value);
end

% Value changed function: SOCminI
function SOCminIValueChanged(app, event)
    assignin("base", "SOCmin", app.SOCminI.Value);
end

% Callback function
function LdmICellEdit(app, event)
    assignin("base", "Ldm_Data", app.LdmI.Data);
end

% Cell edit callback: LdmI
function LdmICellEdit2(app, event)
    assignin("base", "Ldm_Data", app.LdmI.Data);
    app.TotalLdmI.Value=31*app.LdmI.Data(1,1)+28*app.LdmI.Data(2,1)+31*app. ↵
LdmI.Data(3,1)+...
            30*app.LdmI.Data(4,1)+31*app.LdmI.Data(5,1)+30*app.LdmI.Data(6,1) ↵
+31*app.LdmI.Data(7,1)+...
            31*app.LdmI.Data(8,1)+30*app.LdmI.Data(9,1)+31*app.LdmI.Data(10,1) ↵
+...
            30*app.LdmI.Data(11,1)+31*app.LdmI.Data(12,1);
    assignin("base", "Edemanda", app.TotalLdmI.Value);

    plot(app.LdmGraphic,app.LdmI.Data(1:12,1));
end

% Cell edit callback: FhI
function FhICellEdit(app, event)
    assignin("base", "F_Data", app.FhI.Data);
    plot(app.FhGraphic,app.FhI.Data(1:24,1));
    app.TotalFh.Value=sum(app.FhI.Data(1:24,1));

end

% Cell edit callback: MetDataI
function MetDataICellEdit(app, event)
    assignin("base", "months_Data", app.MetDataI.Data);
end

% Value changed function: PGENnomI
function PGENnomIValueChanged(app, event)
    assignin("base", "PGENnom", app.PGENnomI.Value);
    app.coef1=(app.b0iI.Value+app.bliI.Value*app.PGENnomI.Value);
    app.coef2=(app.b0sI.Value+app.blsI.Value*app.PGENnomI.Value);
    app.FuelCons.Data=[0 app.coef1+app.coef2*0; ...
            0.25*app.PGENnomI.Value app.coef1+app.coef2*0.25*app.PGENnomI. ↵
Value; ...
            0.5*app.PGENnomI.Value app.coef1+app.coef2*0.5*app.PGENnomI. ↵
...
            0.75*app.PGENnomI.Value app.coef1+app.coef2*0.75*app.PGENnomI. ↵
Value; ...
            app.PGENnomI.Value app.coef1+app.coef2*app.PGENnomI.Value];
    plot(app.FuelConsGraph,app.FuelCons.Data(1:5,1),app.FuelCons.Data(1: ↵
5,2));

```

```

end

% Value changed function: SOCstartI
function SOCstartIValueChanged(app, event)
    assignin("base", "SOCstart", app.SOCstartI.Value);
end

% Value changed function: SOCstopI
function SOCstopIValueChanged(app, event)
    assignin("base", "SOCstop", app.SOCstopI.Value);
end

% Value changed function: b0ii
function b0iiValueChanged(app, event)
    assignin("base", "b0i", app.b0ii.Value);
    app.coef1=(app.b0ii.Value+app.blii.Value*app.PGENnomI.Value);
    app.FuelCons.Data=[0 app.coef1+app.coef2*0; ...
        0.25*app.PGENnomI.Value app.coef1+app.coef2*0.25*app.PGENnomI. ↵
Value; ...
        0.5*app.PGENnomI.Value app.coef1+app.coef2*0.5*app.PGENnomI. ↵
...
        0.75*app.PGENnomI.Value app.coef1+app.coef2*0.75*app.PGENnomI. ↵
Value; ...
        app.PGENnomI.Value app.coef1+app.coef2*app.PGENnomI.Value];
    plot(app.FuelConsGraph, app.FuelCons.Data(1:5,1),app.FuelCons.Data(1: ↵
5,2));
end

% Value changed function: blii
function bliiValueChanged(app, event)
    assignin("base", "bli", app.blii.Value);
    app.coef1=(app.b0ii.Value+app.blii.Value*app.PGENnomI.Value);
    app.FuelCons.Data=[0 app.coef1+app.coef2*0; ...
        0.25*app.PGENnomI.Value app.coef1+app.coef2*0.25*app.PGENnomI. ↵
Value; ...
        0.5*app.PGENnomI.Value app.coef1+app.coef2*0.5*app.PGENnomI. ↵
...
        0.75*app.PGENnomI.Value app.coef1+app.coef2*0.75*app.PGENnomI. ↵
Value; ...
        app.PGENnomI.Value app.coef1+app.coef2*app.PGENnomI.Value];
    plot(app.FuelConsGraph, app.FuelCons.Data(1:5,1),app.FuelCons.Data(1: ↵
5,2));
end

% Value changed function: b0si
function b0siValueChanged(app, event)
    assignin("base", "b0s", app.b0si.Value);
    app.coef2=(app.b0si.Value+app.blsi.Value*app.PGENnomI.Value);
    app.FuelCons.Data=[0 app.coef1+app.coef2*0; ...
        0.25*app.PGENnomI.Value app.coef1+app.coef2*0.25*app.PGENnomI. ↵
Value; ...
        0.5*app.PGENnomI.Value app.coef1+app.coef2*0.5*app.PGENnomI. ↵
...
        0.75*app.PGENnomI.Value app.coef1+app.coef2*0.75*app.PGENnomI. ↵
Value; ...
        app.PGENnomI.Value app.coef1+app.coef2*app.PGENnomI.Value];
end

```

```

    plot(app.FuelConsGraph,app.FuelCons.Data(1:5,1),app.FuelCons.Data(1:5,2));
end

% Value changed function: b1sI
function b1sIValueChanged(app, event)
    assignin("base","b1s",app.b1sI.Value);
    app.coef2=(app.b0sI.Value+app.b1sI.Value*app.PGENnomI.Value);
    app.FuelCons.Data=[0 app.coef1+app.coef2*0; ...
        0.25*app.PGENnomI.Value app.coef1+app.coef2*0.25*app.PGENnomI. ...
    Value; ...
        0.5*app.PGENnomI.Value app.coef1+app.coef2*0.5*app.PGENnomI. ...
    ...
        0.75*app.PGENnomI.Value app.coef1+app.coef2*0.75*app.PGENnomI. ...
    Value; ...
        app.PGENnomI.Value app.coef1+app.coef2*app.PGENnomI.Value];
    plot(app.FuelConsGraph,app.FuelCons.Data(1:5,1),app.FuelCons.Data(1:5,2));
end

% Value changed function: HELPGenSet
function HELPGenSetValueChanged(app, event)
    if (app.HELPGenSet.Value=="On")
        app.b0iLabel.Visible = 'on';
        app.b1iLabel.Visible = 'on';
        app.b0sLabel.Visible = 'on';
        app.b1sLabel.Visible = 'on';
        app.PGENnomLabel.Visible = 'on';
        app.SOCstartLabel.Visible = 'on';
        app.SOCstopLabel.Visible = 'on';
    else
        app.b0iLabel.Visible = 'off';
        app.b1iLabel.Visible = 'off';
        app.b0sLabel.Visible = 'off';
        app.b1sLabel.Visible = 'off';
        app.PGENnomLabel.Visible = 'off';
        app.SOCstartLabel.Visible = 'off';
        app.SOCstopLabel.Visible = 'off';
    end
end

% Value changed function: HELPSite
function HELPSiteValueChanged(app, event)
    if (app.HELPsite.Value=="On")
        app.LatitudeLabel.Visible = 'on';
        app.StLongitudeLabel.Visible = 'on';
        app.LongitudeLabel.Visible = 'on';
        app.AltitudeLabel.Visible = 'on';
    else
        app.LatitudeLabel.Visible = 'off';
        app.StLongitudeLabel.Visible = 'off';
        app.LongitudeLabel.Visible = 'off';
        app.AltitudeLabel.Visible = 'off';
    end
end

```

```

end

% Value changed function: HELPMeteo
function HELPMeteoValueChanged(app, event)
    if (app.HELPMeteo.Value=="On")
        app.Gdm0Label1.Visible = 'on';
        app.Gdm0Label2.Visible = 'on';
        app.TmmLabel.Visible = 'on';
        app.TMmLabel.Visible = 'on';
    else
        app.Gdm0Label1.Visible = 'off';
        app.Gdm0Label2.Visible = 'off';
        app.TmmLabel.Visible = 'off';
        app.TMmLabel.Visible = 'off';
    end
end

% Value changed function: HELPPVgen
function HELPPVgenValueChanged(app, event)
    if (app.HELPPVgen.Value=="On")
        app.RthLabel1.Visible = 'on';
        app.RthLabel2.Visible = 'on';
        app.NOCTLabel.Visible = 'on';
        app.PVnomLabel.Visible = 'on';
        app.CVPTLabel.Visible = 'on';
    else
        app.RthLabel1.Visible = 'off';
        app.RthLabel2.Visible = 'off';
        app.NOCTLabel.Visible = 'off';
        app.PVnomLabel.Visible = 'off';
        app.CVPTLabel.Visible = 'off';
    end
end

% Value changed function: HELPBattery
function HELPBatteryValueChanged(app, event)
    if (app.HELPBattery.Value=="On")
        app.CBATLabel.Visible = 'on';
        app.SOCmaxLabel.Visible = 'on';
        app.SOCminLabel.Visible = 'on';
    else
        app.CBATLabel.Visible = 'off';
        app.SOCmaxLabel.Visible = 'off';
        app.SOCminLabel.Visible = 'off';
    end
end

% Value changed function: HELPLoad
function HELPLoadValueChanged(app, event)
    if (app.HELPLoad.Value=="On")
        app.LdmLabel.Visible = 'on';
        app.FhLabel1.Visible = 'on';
        app.FhLabel2.Visible = 'on';
    else
        app.LdmLabel.Visible = 'off';
        app.FhLabel1.Visible = 'off';

```

```

        app.FhLabel2.Visible = 'off';
    end

end

% Value changed function: PWnomI
function PWnomIValueChanged(app, event)
    assignin("base", "PWnom", app.PWnomI.Value);
    if(app.PWnomI.Value==0)
        app.CpeqI.Value = 0;
    else
        app.CpeqI.Value = app.PWnomI.Value/(app.VnomI.Value^3-app.VciI. ↵
Value^3);
    end
    app.PowerCurve.Data=[0 0; app.VciI.Value 0; ... ↵
        app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VnomI.Value app.PWnomI.Value; app.VcoI.Value app.PWnomI.Value; ↵
app.VcoI.Value 0];
    plot(app.PowerCurveGraph,app.PowerCurve.Data(1:9,1),app.PowerCurve.Data ↵
(1:9,2));
end

% Value changed function: VnomI
function VnomIValueChanged(app, event)
    assignin("base", "Vnom", app.VnomI.Value);
    if(app.VnomI.Value==0)
        app.CpeqI.Value = 0;
    else
        app.CpeqI.Value = app.PWnomI.Value/(app.VnomI.Value^3-app.VciI. ↵
Value^3);
    end
    app.PowerCurve.Data=[0 0; app.VciI.Value 0; ... ↵
        app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3); ... ↵
        app.VnomI.Value app.PWnomI.Value; app.VcoI.Value app.PWnomI.Value; ↵
app.VcoI.Value 0];
    plot(app.PowerCurveGraph,app.PowerCurve.Data(1:9,1),app.PowerCurve.Data ↵

```

```

(1:9,2));
end

% Value changed function: VciI
function VciIValueChanged(app, event)
    assignin("base", "Vci", app.VciI.Value);
    if(app.VciI.Value==0)
        app.CpeqI.Value = 0;
    else
        app.CpeqI.Value = app.PWnomI.Value/(app.VnomI.Value^3-app.VciI. ↵
Value^3);
    end
    app.PowerCurve.Data=[0 0; app.VciI.Value 0;... ↵
        app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VnomI.Value app.PWnomI.Value; app.VcoI.Value app.PWnomI.Value; ↵
app.VcoI.Value 0];
    plot(app.PowerCurveGraph,app.PowerCurve.Data(1:9,1),app.PowerCurve.Data ↵
(1:9,2));
end

% Value changed function: VcoI
function VcoIValueChanged(app, event)
    assignin("base", "Vco", app.VcoI.Value);
    app.PowerCurve.Data=[0 0; app.VciI.Value 0;... ↵
        app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.2*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.4*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.6*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value) app.CpeqI. ↵
Value*((app.VciI.Value+0.8*(app.VnomI.Value-app.VciI.Value))^3-app.VciI. ↵
Value^3);... ↵
        app.VnomI.Value app.PWnomI.Value; app.VcoI.Value app.PWnomI.Value; ↵
app.VcoI.Value 0];
    plot(app.PowerCurveGraph,app.PowerCurve.Data(1:9,1),app.PowerCurve.Data ↵
(1:9,2));
end

% Value changed function: HELPWind
function HELPWindValueChanged(app, event)
    if(app.HELPWind.Value=="On")
        app.PWnomLabel.Visible = 'on';
        app.VnomLabel.Visible = 'on';

```

```

        app.VciLabel.Visible = 'on';
        app.VcoLabel.Visible = 'on';
        app.CpeqLabel.Visible = 'on';
    else
        app.PWnomLabel.Visible = 'off';
        app.VnomLabel.Visible = 'off';
        app.VciLabel.Visible = 'off';
        app.VcoLabel.Visible = 'off';
        app.CpeqLabel.Visible = 'off';
    end
end

% Value changed function: QratedI
function QratedIValueChanged(app, event)
    assignin("base", "Qrated", app.QratedI.Value);

    if(app.HfI.Value==0 || app.QratedI.Value==0)
        app.ks2I.Value=app.kw2I.Value;
    else
        app.ks2I.Value=(app.HfI.Value/(app.QratedI.Value^2))+app.kw2I. ↵
Value;
    end

    app.systemCurveTable.Data=[0 app.ks0I.Value; ...
        0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
        0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
        0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
        0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
        0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
        0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
        1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
        1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...
        1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];
    plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
systemCurveTable.Data(1:10,2));
end

% Value changed function: HratedI
function HratedIValueChanged(app, event)
    assignin("base", "Hrated", app.HratedI.Value);
end

% Value changed function: LiquidDensityI
function LiquidDensityIValueChanged(app, event)
    assignin("base", "Density", app.LiquidDensityI.Value);
end

% Value changed function: ks1I

```

```

function ks1IValueChanged(app, event)
    app.systemCurveTable.Data=[0 app.ks0I.Value; ...
        0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
    QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
        0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
    QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
        0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
    QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
        0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
    QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
        0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
    QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
        0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
    QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
        1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
    QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
        1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
    QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...
        1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
    QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];
        plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
    systemCurveTable.Data(1:10,2));

end

% Value changed function: HfI
function HfIValueChanged(app, event)
    assignin("base","Hf",app.HfI.Value);

    if(app.HfI.Value==0 || app.QratedI.Value==0)
        app.ks2I.Value=app.kw2I.Value;
    else
        app.ks2I.Value=(app.HfI.Value/(app.QratedI.Value^2))+app.kw2I. ↵
Value;
    end

    app.systemCurveTable.Data=[0 app.ks0I.Value; ...
        0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
    QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
        0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
    QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
        0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
    QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
        0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
    QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
        0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
    QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
        0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
    QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
        1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
    QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
        1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
    QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...
        1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
    QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];
        plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
    systemCurveTable.Data(1:10,2));

```

```

end

% Cell edit callback: pumpCurveTable
function pumpCurveTableCellEdit(app, event)
    assignin("base", "PumpCurve", app.pumpCurveTable.Data);
    plot(app.pumpCurveGraph, app.pumpCurveTable.Data(1:7,1), app. ↵
pumpCurveTable.Data(1:7,2));
end

% Cell edit callback: powerCurvesTable
function powerCurvesTableCellEdit(app, event)
    assignin("base", "PowerCurves", app.powerCurvesTable.Data);
    plot(app.powerCurvesGraph, app.powerCurvesTable.Data(1:7,1), app. ↵
powerCurvesTable.Data(1:7,2), "m");
    plot(app.powerCurvesGraph, app.powerCurvesTable.Data(1:7,1), app. ↵
powerCurvesTable.Data(1:7,3), "b");

end

% Value changed function: P2ratedI
function P2ratedIValueChanged(app, event)
    assignin("base", "P2rated", app.P2ratedI.Value);
end

% Value changed function: RPMnomI
function RPMnomIValueChanged(app, event)
    assignin("base", "RPMnom", app.RPMnomI.Value);
end

% Value changed function: RPMcoolI
function RPMcoolIValueChanged(app, event)
    assignin("base", "RPMcoolM", app.RPMcoolI.Value);
end

% Value changed function: RPMmaxI
function RPMmaxIValueChanged(app, event)
    assignin("base", "RPMmaxM", app.RPMmaxI.Value);
end

% Cell edit callback: PowerEfficTable
function PowerEfficTableCellEdit(app, event)
    assignin("base", "PowerEffic", app.PowerEfficTable.Data);
    plot(app.PowerEfficGraph, app.PowerEfficTable.Data(1:7,1), app. ↵
PowerEfficTable.Data(1:7,2));
end

% Value changed function: HELPIInverter
function HELPIInverterValueChanged(app, event)
    if(app.HELPInverter.Value=="On")
        app.NominaloutputLabel.Visible = 'on';
        app.MaximumoutputLabel.Visible = 'on';
        app.efficiencyLabel1.Visible = 'on';
        app.efficiencyLabel2.Visible = 'on';
        app.efficiencyLabel3.Visible = 'on';
        app.efficiencyLabel4.Visible = 'on';
        app.k0Label.Visible = 'on';
        app.k1Label.Visible = 'on';

```

```

        app.k2Label.Visible = 'on';
        app.pacLabel.Visible = 'on';
        app.EfficiencyLabel.Visible = 'on';
    else
        app.NominaloutputLabel.Visible = 'off';
        app.MaximumoutputLabel.Visible = 'off';
        app.technologyLabel1.Visible = 'off';
        app.technologyLabel2.Visible = 'off';
        app.technologyLabel3.Visible = 'off';
        app.technologyLabel4.Visible = 'off';
        app.k0Label.Visible = 'off';
        app.k1Label.Visible = 'off';
        app.k2Label.Visible = 'off';
        app.pacLabel.Visible = 'off';
        app.EfficiencyLabel.Visible = 'off';
    end
end

% Value changed function: HELPPump2
function HELPPump2ValueChanged(app, event)
    if(app.HELPPump2.Value=="On")
        app.ks0Label.Visible='on';
        app.ks1Label.Visible='on';
        app.ks2Label.Visible='on';
        app.HfLabel.Visible='on';
    else
        app.ks0Label.Visible='off';
        app.ks1Label.Visible='off';
        app.ks2Label.Visible='off';
        app.HfLabel.Visible='off';
    end
end

% Value changed function: HELPPump3
function HELPPump3ValueChanged(app, event)
    if(app.HELPPump3.Value=="On")
        app.P1Label1.Visible = 'on';
        app.P1Label2.Visible = 'on';
        app.P2Label1.Visible = 'on';
        app.P2Label2.Visible = 'on';
    else
        app.P1Label1.Visible = 'off';
        app.P1Label2.Visible = 'off';
        app.P2Label1.Visible = 'off';
        app.P2Label2.Visible = 'off';
    end
end

% Value changed function: HELPPump4
function HELPPump4ValueChanged(app, event)
    if(app.HELPPump4.Value=="On")
        app.P2RatedLabel.Visible = 'on';
        app.RPMnomLabel.Visible = 'on';
        app.RPMcoolLabel.Visible = 'on';
        app.RPMmaxLabel.Visible = 'on';
    else
        app.P2RatedLabel.Visible = 'off';

```

```

        app.RPMnomLabel.Visible = 'off';
        app.RPMcoolLabel.Visible = 'off';
        app.RPMmaxLabel.Visible = 'off';
    end
end

% Value changed function: SaveData
function SaveDataValueChanged(app, event)
    if(app.SaveData.Value==0)
        assignin("base","saveDataApp",0);
    else
        assignin("base","saveDataApp",1);
    end
end

% Value changed function: nameI
function nameIValueChanged(app, event)
    assignin("base","name", app.nameI.Value);
end

% Selection changed function: MOUNTING
function MOUNTINGSelectionChanged(app, event)
    if(app.Staticground.Value==true)
        assignin("base","Mounting",1);
    elseif(app.Staticdelta.Value==true)
        assignin("base","Mounting",2);
    elseif(app.TrackerHorizontal.Value==true)
        assignin("base","Mounting",3);
    elseif(app.TrackerVertical.Value==true)
        assignin("base","Mounting",4);
    else
        assignin("base","Mounting",5);
    end
end

% Value changed function: inclinationI_delta
function inclinationI_deltaValueChanged(app, event)
    assignin("base","InclinationDelta",app.inclinationI_delta.Value);
end

% Value changed function: inclinationI_tracking
function inclinationI_trackingValueChanged(app, event)
    assignin("base","InclinationTracking",app.inclinationI_tracking.Value);
end

% Value changed function: HELPPVgen_2
function HELPPVgen_2ValueChanged(app, event)
    if(app.HELPPVgen_2.Value=="On")
        app.InclinationStaticLabel.Visible = 'on';
        app.OrientationLabel1.Visible = 'on';
        app.OrientationLabel2.Visible = 'on';
        app.InclinationDeltaLabel1.Visible = 'on';
        app.InclinationDeltaLabel2.Visible = 'on';
        app.InclinationTrackingLabel.Visible = 'on';
    else

```

```

        app.InclinationStaticLabel.Visible = 'off';
        app.OrientationLabel1.Visible = 'off';
        app.OrientationLabel2.Visible = 'off';
        app.InclinationDeltaLabel1.Visible = 'off';
        app.InclinationDeltaLabel2.Visible = 'off';
        app.InclinationTrackingLabel.Visible = 'off';
    end

end

% Value changed function: HsI
function HsIValueChanged(app, event)
    assignin("base","Hs",app.HsI.Value);
    app.ks0I.Value = app.HrI.Value + app.HsI.Value;
    app.systemCurveTable.Data=[0 app.ks0I.Value; ...
        0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
    QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
        0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
    QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
        0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
    QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
        0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
    QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
        0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
    QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
        0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
    QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
        1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
    QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
        1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
    QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...
        1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
    QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];
    plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
    systemCurveTable.Data(1:10,2));

end

% Value changed function: QTestI
function QTestIValueChanged(app, event)
    assignin("base","Qtest",app.QtestI.Value);
    app.kw1I.Value=app.HdrI.Value/app.QtestI.Value;
    assignin("base","kw1",app.kw1I.Value);
end

% Value changed function: HdrI
function HdrIValueChanged(app, event)
    assignin("base","Hdr",app.HdrI.Value);
    app.kw1I.Value=app.HdrI.Value/app.QtestI.Value;
    assignin("base","kw1",app.kw1I.Value);
end

% Value changed function: kw1I
function kw1IValueChanged(app, event)
    assignin("base","kw1",app.kw1I.Value);
end

% Value changed function: kw2I

```

```

function kw2IValueChanged(app, event)
    assignin("base", "kw2", app.kw2I.Value);
    if(app.HfI.Value==0 || app.QratedI.Value==0)
        app.ks2I.Value=app.kw2I.Value;
    else
        app.ks2I.Value=(app.HfI.Value/(app.QratedI.Value^2))+app.kw2I. ↵
Value;
    end
    app.systemCurveTable.Data=[0 app.ks0I.Value; ...
        0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
        0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
        0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
        0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
        0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
        0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
        1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
        1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...
        1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];
    plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
systemCurveTable.Data(1:10,2));
end

% Value changed function: WTCI
function WTCIValueChanged(app, event)
    assignin("base", "WTC", app.WTCI.Value);
end

% Value changed function: HrI
function HrIValueChanged(app, event)
    assignin("base", "Hr", app.HrI.Value);
    app.ks0I.Value = app.HrI.Value + app.HsI.Value;
    app.systemCurveTable.Data=[0 app.ks0I.Value; ...
        0.15*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.15*app. ↵
QratedI.Value+app.ks2I.Value*(0.15*app.QratedI.Value)^2; ...
        0.3*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.3*app. ↵
QratedI.Value+app.ks2I.Value*(0.3*app.QratedI.Value)^2; ...
        0.45*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.45*app. ↵
QratedI.Value+app.ks2I.Value*(0.45*app.QratedI.Value)^2; ...
        0.6*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.6*app. ↵
QratedI.Value+app.ks2I.Value*(0.6*app.QratedI.Value)^2; ...
        0.75*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.75*app. ↵
QratedI.Value+app.ks2I.Value*(0.75*app.QratedI.Value)^2; ...
        0.9*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*0.9*app. ↵
QratedI.Value+app.ks2I.Value*(0.9*app.QratedI.Value)^2; ...
        1.05*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.05*app. ↵
QratedI.Value+app.ks2I.Value*(1.05*app.QratedI.Value)^2; ...
        1.2*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.2*app. ↵
QratedI.Value+app.ks2I.Value*(1.2*app.QratedI.Value)^2; ...

```

```

    1.35*app.QratedI.Value app.ks0I.Value+app.ks1I.Value*1.35*app. ↵
QratedI.Value+app.ks2I.Value*(1.35*app.QratedI.Value)^2];
    plot(app.systemCurveGraph,app.systemCurveTable.Data(1:10,1),app. ↵
systemCurveTable.Data(1:10,2));

end

% Value changed function: HELPPump1
function HELPPump1ValueChanged(app, event)
    if(app.HELPPump1.Value=="On")
        app.HsLabel.Visible = 'on';
        app.QtestLabel.Visible = 'on';
        app.HdrLabel.Visible = 'on';
        app.kw1Label1.Visible = 'on';
        app.kw1Label2.Visible = 'on';
        app.kw2Label.Visible = 'on';
        app.WTCLabel.Visible = 'on';
        app.WTCLabel2.Visible = 'on';
        app.HrLabel.Visible = 'on';
        app.QratedLabel.Visible = 'on';
        app.HratedLabel.Visible = 'on';
        app.densityLabel.Visible = 'on';
    else
        app.HsLabel.Visible = 'off';
        app.QtestLabel.Visible = 'off';
        app.HdrLabel.Visible = 'off';
        app.kw1Label1.Visible = 'off';
        app.kw1Label2.Visible = 'off';
        app.kw2Label.Visible = 'off';
        app.WTCLabel.Visible = 'off';
        app.WTCLabel2.Visible = 'off';
        app.HrLabel.Visible = 'off';
        app.QratedLabel.Visible = 'off';
        app.HratedLabel.Visible = 'off';
        app.densityLabel.Visible = 'off';
    end
end

% Value changed function: inclinationGround
function inclinationGroundValueChanged(app, event)
    assignin("base","InclinationGround",app.inclinationGround.Value);
end

% Button pushed function: loadProject
function loadProjectButtonPushed(app, event)
    startupFcn(app);
end
end

% Component initialization
methods (Access = private)

% Create UIFigure and components
function createComponents(app)

    % Create UIFigure and hide until all components are created
    app.UIFigure = uifigure('Visible', 'off');

```

```

app.UIFigure.Color = [0.8902 0.898 1];
app.UIFigure.Position = [100 100 640 480];
app.UIFigure.Name = 'UI Figure';

% Create TabGroup
app.TabGroup = uitabgroup(app.UIFigure);
app.TabGroup.SelectionChangedFcn = createCallbackFcn(app, ↵
@TabGroupSelectionChanged, true);
app.TabGroup.Position = [9 48 624 420];

% Create Site
app.Site = uitab(app.TabGroup);
app.Site.Title = 'Site';
app.Site.BackgroundColor = [0.9686 0.9686 1];
app.Site.ForegroundColor = [0 0.4471 0.7412];

% Create GEOGRAPHICALDATALabel
app.GEOGRAPHICALDATALabel = uilabel(app.Site);
app.GEOGRAPHICALDATALabel.FontName = 'Times New Roman';
app.GEOGRAPHICALDATALabel.FontSize = 16;
app.GEOGRAPHICALDATALabel.FontWeight = 'bold';
app.GEOGRAPHICALDATALabel.FontAngle = 'italic';
app.GEOGRAPHICALDATALabel.FontColor = [0 0.4471 0.7412];
app.GEOGRAPHICALDATALabel.Position = [215 354 176 22];
app.GEOGRAPHICALDATALabel.Text = 'GEOGRAPHICAL DATA';

% Create LatitudeDegreeLabel
app.LatitudeDegreeLabel = uilabel(app.Site);
app.LatitudeDegreeLabel.HorizontalAlignment = 'right';
app.LatitudeDegreeLabel.FontName = 'Times New Roman';
app.LatitudeDegreeLabel.FontSize = 14;
app.LatitudeDegreeLabel.FontWeight = 'bold';
app.LatitudeDegreeLabel.Position = [184 280 111 22];
app.LatitudeDegreeLabel.Text = 'Latitude (Degree)';

% Create LatitudeI
app.LatitudeI = uieditfield(app.Site, 'numeric');
app.LatitudeI.ValueChangedFcn = createCallbackFcn(app, ↵
@LatitudeIValueChanged, true);
app.LatitudeI.FontName = 'Times New Roman';
app.LatitudeI.FontSize = 14;
app.LatitudeI.FontWeight = 'bold';
app.LatitudeI.Position = [310 280 100 22];

% Create LongitudeDegreeLabel
app.LongitudeDegreeLabel = uilabel(app.Site);
app.LongitudeDegreeLabel.HorizontalAlignment = 'right';
app.LongitudeDegreeLabel.FontName = 'Times New Roman';
app.LongitudeDegreeLabel.FontSize = 14;
app.LongitudeDegreeLabel.FontWeight = 'bold';
app.LongitudeDegreeLabel.Position = [379 94 121 22];
app.LongitudeDegreeLabel.Text = 'Longitude (Degree)';

% Create LongitudeI
app.LongitudeI = uieditfield(app.Site, 'numeric');
app.LongitudeI.ValueChangedFcn = createCallbackFcn(app, ↵
@LongitudeIValueChanged, true);
app.LongitudeI.FontName = 'Times New Roman';

```

```

app.LongitudeI.FontSize = 14;
app.LongitudeI.FontWeight = 'bold';
app.LongitudeI.Position = [515 94 83 22];

% Create AltitudemeterLabel
app.AltitudemeterLabel = uilabel(app.Site);
app.AltitudemeterLabel.HorizontalAlignment = 'right';
app.AltitudemeterLabel.FontName = 'Times New Roman';
app.AltitudemeterLabel.FontSize = 14;
app.AltitudemeterLabel.FontWeight = 'bold';
app.AltitudemeterLabel.Position = [5 44 102 22];
app.AltitudemeterLabel.Text = 'Altitude (meter)';

% Create AltitudeI
app.AltitudeI = uieditfield(app.Site, 'numeric');
app.AltitudeI.ValueChangedFcn = createCallbackFcn(app, ↵
@AltitudeIValueChanged, true);
app.AltitudeI.FontName = 'Times New Roman';
app.AltitudeI.FontSize = 14;
app.AltitudeI.FontWeight = 'bold';
app.AltitudeI.Position = [122 44 71 22];

% Create StandardLongitudeDegreeLabel
app.StandardLongitudeDegreeLabel = uilabel(app.Site);
app.StandardLongitudeDegreeLabel.HorizontalAlignment = 'right';
app.StandardLongitudeDegreeLabel.FontName = 'Times New Roman';
app.StandardLongitudeDegreeLabel.FontSize = 14;
app.StandardLongitudeDegreeLabel.FontWeight = 'bold';
app.StandardLongitudeDegreeLabel.Position = [8 191 181 22];
app.StandardLongitudeDegreeLabel.Text = 'Standard Longitude (Degree)';

% Create StandardLongitudeI
app.StandardLongitudeI = uislider(app.Site);
app.StandardLongitudeI.Step = 15;
app.StandardLongitudeI.ValueChangedFcn = createCallbackFcn(app, ↵
@StandardLongitudeIValueChanged, true);
app.StandardLongitudeI.FontName = 'Times New Roman';
app.StandardLongitudeI.FontSize = 14;
app.StandardLongitudeI.FontWeight = 'bold';
app.StandardLongitudeI.Position = [204 191 100 22];

% Create LatitudeLabel
app.LatitudeLabel = uilabel(app.Site);
app.LatitudeLabel.FontName = 'Times New Roman';
app.LatitudeLabel.FontSize = 14;
app.LatitudeLabel.FontColor = [0.4667 0.6745 0.1882];
app.LatitudeLabel.Visible = 'off';
app.LatitudeLabel.Position = [13 309 599 22];
app.LatitudeLabel.Text = 'Latitude of the location, positive in the ↵
Northern Hemisphere and negative in the Southern Hemisphere.';

% Create StLongitudeLabel
app.StLongitudeLabel = uilabel(app.Site);
app.StLongitudeLabel.FontName = 'Times New Roman';
app.StLongitudeLabel.FontSize = 14;
app.StLongitudeLabel.FontColor = [0.4667 0.6745 0.1882];
app.StLongitudeLabel.Visible = 'off';
app.StLongitudeLabel.Position = [13 221 505 22];

```

```

    app.StLongitudeLabel.Text = 'Standard longitude (multiple of 15°). ↵
Negative towards West and positive towards East.';

    % Create LongitudeLabel
    app.LongitudeLabel = uilabel(app.Site);
    app.LongitudeLabel.FontName = 'Times New Roman';
    app.LongitudeLabel.FontSize = 14;
    app.LongitudeLabel.FontColor = [0.4667 0.6745 0.1882];
    app.LongitudeLabel.Visible = 'off';
    app.LongitudeLabel.Position = [173 123 433 22];
    app.LongitudeLabel.Text = 'Longitude of the location, negative towards ↵
West and positive towards Est.';

    % Create AltitudeLabel
    app.AltitudeLabel = uilabel(app.Site);
    app.AltitudeLabel.FontName = 'Times New Roman';
    app.AltitudeLabel.FontSize = 14;
    app.AltitudeLabel.FontColor = [0.4667 0.6745 0.1882];
    app.AltitudeLabel.Visible = 'off';
    app.AltitudeLabel.Position = [13 73 220 22];
    app.AltitudeLabel.Text = 'Altitude of the location over sea level.';

    % Create HELPSwitchLabel_2
    app.HELPSSwitchLabel_2 = uilabel(app.Site);
    app.HELPSSwitchLabel_2.HorizontalAlignment = 'center';
    app.HELPSSwitchLabel_2.FontName = 'Times New Roman';
    app.HELPSSwitchLabel_2.FontSize = 14;
    app.HELPSSwitchLabel_2.FontWeight = 'bold';
    app.HELPSSwitchLabel_2.FontColor = [0.4667 0.6745 0.1882];
    app.HELPSSwitchLabel_2.Position = [555.5 182 43 22];
    app.HELPSSwitchLabel_2.Text = 'HELP';

    % Create HELPSite
    app.HELPSSite = uiswitch(app.Site, 'slider');
    app.HELPSSite.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPSSiteValueChanged, true);
    app.HELPSSite.FontName = 'Times New Roman';
    app.HELPSSite.FontSize = 14;
    app.HELPSSite.FontWeight = 'bold';
    app.HELPSSite.FontColor = [0.4667 0.6745 0.1882];
    app.HELPSSite.Position = [466 184 45 20];

    % Create Meteo
    app.Meteo = uitab(app.TabGroup);
    app.Meteo.Title = 'Meteo';
    app.Meteo.BackgroundColor = [0.9686 0.9686 1];
    app.Meteo.ForegroundColor = [0 0.4471 0.7412];

    % Create MeteorologicalInputLabel
    app.MeteorologicalInputLabel = uilabel(app.Meteo);
    app.MeteorologicalInputLabel.HorizontalAlignment = 'right';
    app.MeteorologicalInputLabel.FontName = 'Times New Roman';
    app.MeteorologicalInputLabel.FontSize = 13;
    app.MeteorologicalInputLabel.FontWeight = 'bold';
    app.MeteorologicalInputLabel.Position = [73 301 149 22];
    app.MeteorologicalInputLabel.Text = 'Meteorological input ↵
data';

```

```

% Create InputDataI
app.InputDataI = uilistbox(app.Meteo);
app.InputDataI.Items = {'Monthly averages from the table', 'Monthly averages obtained from PVGIS TMY', 'Hourly values from PVGIS TMY', 'Monthly averages obtained from USA TMY3', 'Hourly values from USA TMY3'};
app.InputDataI.ValueChangedFcn = createCallbackFcn(app, %
@InputDataIValueChanged, true);
app.InputDataI.FontName = 'Times New Roman';
app.InputDataI.FontSize = 13;
app.InputDataI.Position = [27 198 268 98];
app.InputDataI.Value = 'Monthly averages from the table';

% Create TimeSeriesDropDownLabel
app.TimeSeriesDropDownLabel = uilabel(app.Meteo);
app.TimeSeriesDropDownLabel.HorizontalAlignment = 'right';
app.TimeSeriesDropDownLabel.FontName = 'Times New Roman';
app.TimeSeriesDropDownLabel.FontSize = 13;
app.TimeSeriesDropDownLabel.FontWeight = 'bold';
app.TimeSeriesDropDownLabel.Position = [64 140 71 22];
app.TimeSeriesDropDownLabel.Text = 'Time Series';

% Create TimeSeriesI
app.TimeSeriesI = uidropdown(app.Meteo);
app.TimeSeriesI.Items = {'Mean Days', 'Aguiar'};
app.TimeSeriesI.ValueChangedFcn = createCallbackFcn(app, %
@TimeSeriesIValueChanged, true);
app.TimeSeriesI.FontName = 'Times New Roman';
app.TimeSeriesI.FontSize = 13;
app.TimeSeriesI.Position = [150 140 100 22];
app.TimeSeriesI.Value = 'Mean Days';

% Create METEOROLOGICALINPUTDATALabel
app.METEOROLOGICALINPUTDATALabel = uilabel(app.Meteo);
app.METEOROLOGICALINPUTDATALabel.FontName = 'Times New Roman';
app.METEOROLOGICALINPUTDATALabel.FontSize = 16;
app.METEOROLOGICALINPUTDATALabel.FontWeight = 'bold';
app.METEOROLOGICALINPUTDATALabel.FontAngle = 'italic';
app.METEOROLOGICALINPUTDATALabel.FontColor = [0 0.4471 0.7412];
app.METEOROLOGICALINPUTDATALabel.Position = [27 344 254 22];
app.METEOROLOGICALINPUTDATALabel.Text = 'METEOROLOGICAL INPUT DATA';

% Create MetDataI
app.MetDataI = uitable(app.Meteo);
app.MetDataI.ColumnName = {'Gdm0 [Wh/m2]', 'Tmm [°C]', 'TMm [°C]'};
app.MetDataI.ColumnWidth = {80, 60, 60};
app.MetDataI.RowName = {'January', 'February', 'March', 'April', 'May', %
'June', 'July', 'August', 'September', 'October', 'November', 'December'};
app.MetDataI.ColumnEditable = true;
app.MetDataI.CellEditCallback = createCallbackFcn(app, %
@MetDataICellEdit, true);
app.MetDataI.FontName = 'Times New Roman';
app.MetDataI.FontSize = 13;
app.MetDataI.Position = [330 36 272 309];

% Create Gdm0Label1
app.Gdm0Label1 = uilabel(app.Meteo);
app.Gdm0Label1.FontName = 'Times New Roman';
app.Gdm0Label1.FontSize = 13;

```

```

app.Gdm0Label1.FontColor = [0.4667 0.6745 0.1882];
app.Gdm0Label1.Visible = 'off';
app.Gdm0Label1.Position = [22 84 274 22];
app.Gdm0Label1.Text = 'Gdm0: Monthly average of daily global ↵
horizontal';

% Create Gdm0Label2
app.Gdm0Label2 = uilabel(app.Meteo);
app.Gdm0Label2.FontName = 'Times New Roman';
app.Gdm0Label2.FontSize = 13;
app.Gdm0Label2.FontColor = [0.4667 0.6745 0.1882];
app.Gdm0Label2.Visible = 'off';
app.Gdm0Label2.Position = [20 70 66 22];
app.Gdm0Label2.Text = ' irradiation.';

% Create TmmLabel
app.TmmLabel = uilabel(app.Meteo);
app.TmmLabel.FontName = 'Times New Roman';
app.TmmLabel.FontSize = 13;
app.TmmLabel.FontColor = [0.4667 0.6745 0.1882];
app.TmmLabel.Visible = 'off';
app.TmmLabel.Position = [22 53 295 22];
app.TmmLabel.Text = 'Tmm: Monthly average of ninimum daily ↵
temperature.';

% Create TMmLabel
app.TMmLabel = uilabel(app.Meteo);
app.TMmLabel.FontName = 'Times New Roman';
app.TMmLabel.FontSize = 13;
app.TMmLabel.FontColor = [0.4667 0.6745 0.1882];
app.TMmLabel.Visible = 'off';
app.TMmLabel.Position = [22 36 302 22];
app.TMmLabel.Text = 'TMm: Monthly average of maximum daily ↵
temperature.';

% Create HELPSwitchLabel_3
app.HELPswitchLabel_3 = uilabel(app.Meteo);
app.HELPswitchLabel_3.HorizontalAlignment = 'center';
app.HELPswitchLabel_3.FontName = 'Times New Roman';
app.HELPswitchLabel_3.FontSize = 13;
app.HELPswitchLabel_3.FontWeight = 'bold';
app.HELPswitchLabel_3.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_3.Position = [496.5 357 41 22];
app.HELPswitchLabel_3.Text = 'HELP';

% Create HELPMeteo
app.HELPMeteo = uiswitch(app.Meteo, 'slider');
app.HELPMeteo.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPMeteoValueChanged, true);
app.HELPMeteo.FontName = 'Times New Roman';
app.HELPMeteo.FontSize = 13;
app.HELPMeteo.FontWeight = 'bold';
app.HELPMeteo.FontColor = [0.4667 0.6745 0.1882];
app.HELPMeteo.Position = [409 359 45 20];

% Create PVgenerator
app.PVgenerator = uitab(app.TabGroup);
app.PVgenerator.Title = 'PVgen (1)';

```

```

app.PVgenerator.BackgroundColor = [0.9686 0.9686 1];
app.PVgenerator.ForegroundColor = [0 0.4471 0.7412];

% Create PVnomkWpEditFieldLabel
app.PVnomkWpEditFieldLabel = uilabel(app.PVgenerator);
app.PVnomkWpEditFieldLabel.HorizontalAlignment = 'right';
app.PVnomkWpEditFieldLabel.FontName = 'Times New Roman';
app.PVnomkWpEditFieldLabel.FontSize = 13;
app.PVnomkWpEditFieldLabel.FontWeight = 'bold';
app.PVnomkWpEditFieldLabel.Position = [31 290 87 22];
app.PVnomkWpEditFieldLabel.Text = 'PVnom (kWp)';

% Create PVnomI
app.PVnomI = uieditfield(app.PVgenerator, 'numeric');
app.PVnomI.ValueChangedFcn = createCallbackFcn(app, ↵
@PVnomIValueChanged, true);
app.PVnomI.FontName = 'Times New Roman';
app.PVnomI.FontSize = 13;
app.PVnomI.FontWeight = 'bold';
app.PVnomI.Position = [129 290 64 22];

% Create CVPTC1Label
app.CVPTC1Label = uilabel(app.PVgenerator);
app.CVPTC1Label.HorizontalAlignment = 'right';
app.CVPTC1Label.FontName = 'Times New Roman';
app.CVPTC1Label.FontSize = 13;
app.CVPTC1Label.FontWeight = 'bold';
app.CVPTC1Label.Position = [24 173 102 22];
app.CVPTC1Label.Text = 'CVPT (%·°C^-1)';

% Create CVPTI
app.CVPTI = uieditfield(app.PVgenerator, 'numeric');
app.CVPTI.Limits = [0 Inf];
app.CVPTI.ValueChangedFcn = createCallbackFcn(app, @CVPTIValueChanged, ↵
true);
app.CVPTI.FontName = 'Times New Roman';
app.CVPTI.FontSize = 13;
app.CVPTI.FontWeight = 'bold';
app.CVPTI.Position = [141 173 100 22];

% Create NOCTCLabel
app.NOCTCLabel = uilabel(app.PVgenerator);
app.NOCTCLabel.HorizontalAlignment = 'right';
app.NOCTCLabel.FontName = 'Times New Roman';
app.NOCTCLabel.FontSize = 13;
app.NOCTCLabel.FontWeight = 'bold';
app.NOCTCLabel.Position = [459 265 68 22];
app.NOCTCLabel.Text = 'NOCT (°C)';

% Create NOCTI
app.NOCTI = uieditfield(app.PVgenerator, 'numeric');
app.NOCTI.ValueChangedFcn = createCallbackFcn(app, @NOCTIValueChanged, ↵
true);
app.NOCTI.FontName = 'Times New Roman';
app.NOCTI.FontSize = 13;
app.NOCTI.FontWeight = 'bold';
app.NOCTI.Position = [542 265 55 22];

```

```

% Create NOCTLabel
app.NOCTLabel = uilabel(app.PVgenerator);
app.NOCTLabel.FontName = 'Times New Roman';
app.NOCTLabel.FontSize = 13;
app.NOCTLabel.FontColor = [0.4667 0.6745 0.1882];
app.NOCTLabel.Visible = 'off';
app.NOCTLabel.Position = [402 288 200 22];
app.NOCTLabel.Text = 'Nominal Operation Cell Temperature';

% Create ThermalresistanceCm2WLabel
app.ThermalresistanceCm2WLabel = uilabel(app.PVgenerator);
app.ThermalresistanceCm2WLabel.HorizontalAlignment = 'right';
app.ThermalresistanceCm2WLabel.FontName = 'Times New Roman';
app.ThermalresistanceCm2WLabel.FontSize = 13;
app.ThermalresistanceCm2WLabel.FontWeight = 'bold';
app.ThermalresistanceCm2WLabel.Position = [223 52 175 22];
app.ThermalresistanceCm2WLabel.Text = 'Thermal resistance (°C·m2/W)';

% Create RthI
app.RthI = uieditfield(app.PVgenerator, 'numeric');
app.RthI.ValueChangedFcn = createCallbackFcn(app, @RthIValueChanged, ↵
true);
app.RthI.FontName = 'Times New Roman';
app.RthI.FontSize = 13;
app.RthI.FontWeight = 'bold';
app.RthI.Position = [413 52 100 22];

% Create PVnomLabel
app.PVnomLabel = uilabel(app.PVgenerator);
app.PVnomLabel.FontName = 'Times New Roman';
app.PVnomLabel.FontSize = 13;
app.PVnomLabel.FontColor = [0.4667 0.6745 0.1882];
app.PVnomLabel.Visible = 'off';
app.PVnomLabel.Position = [46 314 107 27];
app.PVnomLabel.Text = 'Nominal PV power';

% Create CVPTLabel
app.CVPTLabel = uilabel(app.PVgenerator);
app.CVPTLabel.FontName = 'Times New Roman';
app.CVPTLabel.FontSize = 13;
app.CVPTLabel.FontColor = [0.4667 0.6745 0.1882];
app.CVPTLabel.Visible = 'off';
app.CVPTLabel.Position = [11 196 431 33];
app.CVPTLabel.Text = 'Coefficient of Variation of PV module Power with ↵
Temperature (absolute value)';

% Create RthLabel1
app.RthLabel1 = uilabel(app.PVgenerator);
app.RthLabel1.FontName = 'Times New Roman';
app.RthLabel1.FontSize = 13;
app.RthLabel1.FontColor = [0.4667 0.6745 0.1882];
app.RthLabel1.Visible = 'off';
app.RthLabel1.Position = [124 93 472 22];
app.RthLabel1.Text = 'Thermal resistance. This parameter is calculated ↵
as [NOCT-20]/800, but other values ';

% Create RthLabel2
app.RthLabel2 = uilabel(app.PVgenerator);

```

```

app.RthLabel2.FontName = 'Times New Roman';
app.RthLabel2.FontSize = 13;
app.RthLabel2.FontColor = [0.4667 0.6745 0.1882];
app.RthLabel2.Visible = 'off';
app.RthLabel2.Position = [124 77 405 22];
app.RthLabel2.Text = 'can be chosen depending on the mounting and ↵
ventilation of PV modules.';

% Create PVGENERATOR12Label
app.PVGENERATOR12Label = uilabel(app.PVgenerator);
app.PVGENERATOR12Label.FontName = 'Times New Roman';
app.PVGENERATOR12Label.FontSize = 16;
app.PVGENERATOR12Label.FontWeight = 'bold';
app.PVGENERATOR12Label.FontAngle = 'italic';
app.PVGENERATOR12Label.FontColor = [0 0.4471 0.7412];
app.PVGENERATOR12Label.Position = [212 354 162 22];
app.PVGENERATOR12Label.Text = 'PV GENERATOR (1/2)';

% Create HELPSwitchLabel_4
app.HELPswitchLabel_4 = uilabel(app.PVgenerator);
app.HELPswitchLabel_4.HorizontalAlignment = 'center';
app.HELPswitchLabel_4.FontName = 'Times New Roman';
app.HELPswitchLabel_4.FontSize = 13;
app.HELPswitchLabel_4.FontWeight = 'bold';
app.HELPswitchLabel_4.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_4.Position = [519 152 41 22];
app.HELPswitchLabel_4.Text = 'HELP';

% Create HELPPVgen
app.HELPPVgen = uiswitch(app.PVgenerator, 'slider');
app.HELPPVgen.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPPVgenValueChanged, true);
app.HELPPVgen.FontName = 'Times New Roman';
app.HELPPVgen.FontSize = 13;
app.HELPPVgen.FontWeight = 'bold';
app.HELPPVgen.FontColor = [0.4667 0.6745 0.1882];
app.HELPPVgen.Position = [515 180 45 20];

% Create PVgenenerator2
app.PVgenenerator2 = uitab(app.TabGroup);
app.PVgenenerator2.Title = 'PVgen (2)';
app.PVgenenerator2.BackgroundColor = [0.9686 0.9686 1];
app.PVgenenerator2.ForegroundColor = [0 0.451 0.7412];

% Create InclinationDegreeLabel
app.InclinationDegreeLabel = uilabel(app.PVgenenerator2);
app.InclinationDegreeLabel.HorizontalAlignment = 'right';
app.InclinationDegreeLabel.FontName = 'Times New Roman';
app.InclinationDegreeLabel.FontSize = 13;
app.InclinationDegreeLabel.FontWeight = 'bold';
app.InclinationDegreeLabel.Position = [45 90 117 22];
app.InclinationDegreeLabel.Text = 'Inclination (Degree)';

% Create inclinationGround
app.inclinationGround = uieditfield(app.PVgenenerator2, 'numeric');
app.inclinationGround.Limits = [0 90];
app.inclinationGround.ValueChangedFcn = createCallbackFcn(app, ↵
@inclinationGroundValueChanged, true);

```

```

app.inclinationGround.FontName = 'Times New Roman';
app.inclinationGround.FontSize = 13;
app.inclinationGround.FontWeight = 'bold';
app.inclinationGround.Position = [177 90 59 22];

% Create InclinationStaticLabel
app.InclinationStaticLabel = uilabel(app.PVgenenerator2);
app.InclinationStaticLabel.FontName = 'Times New Roman';
app.InclinationStaticLabel.FontSize = 13;
app.InclinationStaticLabel.FontColor = [0.4667 0.6745 0.1882];
app.InclinationStaticLabel.Visible = 'off';
app.InclinationStaticLabel.Position = [24 112 332 22];
app.InclinationStaticLabel.Text = 'Inclination of the modules regarding ↵
the horizontal (0° to 90°)';

% Create OrientationDegreeLabel
app.OrientationDegreeLabel = uilabel(app.PVgenenerator2);
app.OrientationDegreeLabel.HorizontalAlignment = 'right';
app.OrientationDegreeLabel.FontName = 'Times New Roman';
app.OrientationDegreeLabel.FontSize = 13;
app.OrientationDegreeLabel.FontWeight = 'bold';
app.OrientationDegreeLabel.Position = [180 13 121 22];
app.OrientationDegreeLabel.Text = 'Orientation (Degree)';

% Create orientationI
app.orientationI = uieditfield(app.PVgenenerator2, 'numeric');
app.orientationI.ValueChangedFcn = createCallbackFcn(app, ↵
@orientationIValueChanged, true);
app.orientationI.FontName = 'Times New Roman';
app.orientationI.FontSize = 13;
app.orientationI.FontWeight = 'bold';
app.orientationI.Position = [316 13 100 22];

% Create OrientationLabel1
app.OrientationLabel1 = uilabel(app.PVgenenerator2);
app.OrientationLabel1.FontName = 'Times New Roman';
app.OrientationLabel1.FontSize = 13;
app.OrientationLabel1.FontColor = [0.4667 0.6745 0.1882];
app.OrientationLabel1.Visible = 'off';
app.OrientationLabel1.Position = [24 51 543 22];
app.OrientationLabel1.Text = 'Orientation of the modules towards the ↵
Equator. Zero towards the South in the Northern Hemisphere';

% Create OrientationLabel2
app.OrientationLabel2 = uilabel(app.PVgenenerator2);
app.OrientationLabel2.FontName = 'Times New Roman';
app.OrientationLabel2.FontSize = 13;
app.OrientationLabel2.FontColor = [0.4667 0.6745 0.1882];
app.OrientationLabel2.Visible = 'off';
app.OrientationLabel2.Position = [24 36 509 22];
app.OrientationLabel2.Text = '(North in the Southern Hemisphere), ↵
negative towards the East, and positive towards the West.';

% Create PVGENERATOR22Label
app.PVGENERATOR22Label = uilabel(app.PVgenenerator2);
app.PVGENERATOR22Label.FontName = 'Times New Roman';
app.PVGENERATOR22Label.FontSize = 16;
app.PVGENERATOR22Label.FontWeight = 'bold';

```

```

app.PVGENERATOR22Label.FontAngle = 'italic';
app.PVGENERATOR22Label.FontColor = [0 0.451 0.7412];
app.PVGENERATOR22Label.Position = [215 357 162 22];
app.PVGENERATOR22Label.Text = 'PV GENERATOR (2/2)';

% Create MOUNTINGSTRUCTURELabel
app.MOUNTINGSTRUCTURELabel = uilabel(app.PVgenenerator2);
app.MOUNTINGSTRUCTURELabel.FontName = 'Times New Roman';
app.MOUNTINGSTRUCTURELabel.FontSize = 13;
app.MOUNTINGSTRUCTURELabel.FontWeight = 'bold';
app.MOUNTINGSTRUCTURELabel.FontColor = [0 0.451 0.7412];
app.MOUNTINGSTRUCTURELabel.Position = [21 326 163 22];
app.MOUNTINGSTRUCTURELabel.Text = 'MOUNTING STRUCTURE';

% Create MOUNTING
app.MOUNTING = uibuttongroup(app.PVgenenerator2);
app.MOUNTING.SelectionChangedFcn = createCallbackFcn(app, ↵
@MOUNTINGSelectionChanged, true);
app.MOUNTING.ForegroundColor = [0 0.451 0.7412];
app.MOUNTING.Title = 'Mounting structure';
app.MOUNTING.BackgroundColor = [0.8902 0.902 1];
app.MOUNTING.FontName = 'Times New Roman';
app.MOUNTING.FontWeight = 'bold';
app.MOUNTING.FontSize = 13;
app.MOUNTING.Position = [21 185 239 136];

% Create Staticground
app.Staticground = uiradiobutton(app.MOUNTING);
app.Staticground.Text = 'Static: ground or roof';
app.Staticground.FontName = 'Times New Roman';
app.Staticground.Position = [11 88 126 22];
app.Staticground.Value = true;

% Create Staticdelta
app.Staticdelta = uiradiobutton(app.MOUNTING);
app.Staticdelta.Text = 'Static: delta';
app.Staticdelta.FontName = 'Times New Roman';
app.Staticdelta.Position = [11 66 79 22];

% Create TrackerHorizontal
app.TrackerHorizontal = uiradiobutton(app.MOUNTING);
app.TrackerHorizontal.Text = 'Tracker: One axis N-S horizontal';
app.TrackerHorizontal.FontName = 'Times New Roman';
app.TrackerHorizontal.Position = [11 44 182 22];

% Create TrackerVertical
app.TrackerVertical = uiradiobutton(app.MOUNTING);
app.TrackerVertical.Text = 'Tracker: One axis vertical (azimuthal)';
app.TrackerVertical.FontName = 'Times New Roman';
app.TrackerVertical.Position = [11 23 200 22];

% Create TrackerTwo
app.TrackerTwo = uiradiobutton(app.MOUNTING);
app.TrackerTwo.Text = 'Tracker: Two axis';
app.TrackerTwo.FontName = 'Times New Roman';
app.TrackerTwo.Position = [11 2 109 22];

% Create StaticgroundorroofLabel

```

```

app.StaticgroundorroofLabel = uilabel(app.PVgenenerator2);
app.StaticgroundorroofLabel.FontName = 'Times New Roman';
app.StaticgroundorroofLabel.FontSize = 13;
app.StaticgroundorroofLabel.FontWeight = 'bold';
app.StaticgroundorroofLabel.FontColor = [0 0.451 0.7412];
app.StaticgroundorroofLabel.Position = [93 133 123 22];
app.StaticgroundorroofLabel.Text = 'Static ground or roof';

% Create StaticdeltaLabel
app.StaticdeltaLabel = uilabel(app.PVgenenerator2);
app.StaticdeltaLabel.FontName = 'Times New Roman';
app.StaticdeltaLabel.FontSize = 13;
app.StaticdeltaLabel.FontWeight = 'bold';
app.StaticdeltaLabel.FontColor = [0 0.451 0.7412];
app.StaticdeltaLabel.Position = [415 317 68 22];
app.StaticdeltaLabel.Text = 'Static delta';

% Create AzimutaltrackingLabel
app.AzimutaltrackingLabel = uilabel(app.PVgenenerator2);
app.AzimutaltrackingLabel.FontName = 'Times New Roman';
app.AzimutaltrackingLabel.FontSize = 13;
app.AzimutaltrackingLabel.FontWeight = 'bold';
app.AzimutaltrackingLabel.FontColor = [0 0.451 0.7412];
app.AzimutaltrackingLabel.Position = [404 212 107 22];
app.AzimutaltrackingLabel.Text = 'Azimutal tracking';

% Create InclinationDegreeLabel_2
app.InclinationDegreeLabel_2 = uilabel(app.PVgenenerator2);
app.InclinationDegreeLabel_2.HorizontalAlignment = 'right';
app.InclinationDegreeLabel_2.FontName = 'Times New Roman';
app.InclinationDegreeLabel_2.FontSize = 13;
app.InclinationDegreeLabel_2.FontWeight = 'bold';
app.InclinationDegreeLabel_2.Position = [362 261 117 22];
app.InclinationDegreeLabel_2.Text = 'Inclination (Degree)';

% Create inclinationI_delta
app.inclinationI_delta = uieditfield(app.PVgenenerator2, 'numeric');
app.inclinationI_delta.Limits = [0 90];
app.inclinationI_delta.ValueChangedFcn = createCallbackFcn(app, ↵
@inclinationI_deltaValueChanged, true);
app.inclinationI_delta.FontName = 'Times New Roman';
app.inclinationI_delta.FontSize = 13;
app.inclinationI_delta.FontWeight = 'bold';
app.inclinationI_delta.Position = [494 261 59 22];

% Create InclinationDegreeLabel_3
app.InclinationDegreeLabel_3 = uilabel(app.PVgenenerator2);
app.InclinationDegreeLabel_3.HorizontalAlignment = 'right';
app.InclinationDegreeLabel_3.FontName = 'Times New Roman';
app.InclinationDegreeLabel_3.FontSize = 13;
app.InclinationDegreeLabel_3.FontWeight = 'bold';
app.InclinationDegreeLabel_3.Position = [362 166 117 22];
app.InclinationDegreeLabel_3.Text = 'Inclination (Degree)';

% Create inclinationI_tracking
app.inclinationI_tracking = uieditfield(app.PVgenenerator2, 'numeric');
app.inclinationI_tracking.Limits = [0 90];
app.inclinationI_tracking.ValueChangedFcn = createCallbackFcn(app, ↵

```

```

@inclineI_trackingValueChanged, true);
    app.inclineI_tracking.FontName = 'Times New Roman';
    app.inclineI_tracking.FontSize = 13;
    app.inclineI_tracking.FontWeight = 'bold';
    app.inclineI_tracking.Position = [494 166 59 22];

    % Create InclineDeltaLabel1
    app.InclineDeltaLabel1 = uilabel(app.PVgenenerator2);
    app.InclineDeltaLabel1.FontName = 'Times New Roman';
    app.InclineDeltaLabel1.FontSize = 13;
    app.InclineDeltaLabel1.FontColor = [0.4667 0.6745 0.1882];
    app.InclineDeltaLabel1.Visible = 'off';
    app.InclineDeltaLabel1.Position = [283 301 332 22];
    app.InclineDeltaLabel1.Text = 'Incline of the modules regarding ↵
the horizontal (0° to 90°)';

    % Create InclineDeltaLabel2
    app.InclineDeltaLabel2 = uilabel(app.PVgenenerator2);
    app.InclineDeltaLabel2.FontName = 'Times New Roman';
    app.InclineDeltaLabel2.FontSize = 13;
    app.InclineDeltaLabel2.FontColor = [0.4667 0.6745 0.1882];
    app.InclineDeltaLabel2.Visible = 'off';
    app.InclineDeltaLabel2.Position = [283 286 212 22];
    app.InclineDeltaLabel2.Text = 'The same for East and West ↵
structures.';

    % Create InclineTrackingLabel
    app.InclineTrackingLabel = uilabel(app.PVgenenerator2);
    app.InclineTrackingLabel.FontName = 'Times New Roman';
    app.InclineTrackingLabel.FontSize = 13;
    app.InclineTrackingLabel.FontColor = [0.4667 0.6745 0.1882];
    app.InclineTrackingLabel.Visible = 'off';
    app.InclineTrackingLabel.Position = [283 191 332 22];
    app.InclineTrackingLabel.Text = 'Incline of the modules ↵
regarding the horizontal (0° to 90°)';

    % Create HELPSwitchLabel_12
    app.HELPswitchLabel_12 = uilabel(app.PVgenenerator2);
    app.HELPswitchLabel_12.HorizontalAlignment = 'center';
    app.HELPswitchLabel_12.FontName = 'Times New Roman';
    app.HELPswitchLabel_12.FontSize = 13;
    app.HELPswitchLabel_12.FontWeight = 'bold';
    app.HELPswitchLabel_12.FontColor = [0.4667 0.6745 0.1882];
    app.HELPswitchLabel_12.Position = [500 99 41 22];
    app.HELPswitchLabel_12.Text = 'HELP';

    % Create HELPPVgen_2
    app.HELPPVgen_2 = uiswitch(app.PVgenenerator2, 'slider');
    app.HELPPVgen_2.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPPVgen_2ValueChanged, true);
    app.HELPPVgen_2.FontName = 'Times New Roman';
    app.HELPPVgen_2.FontSize = 13;
    app.HELPPVgen_2.FontWeight = 'bold';
    app.HELPPVgen_2.FontColor = [0.4667 0.6745 0.1882];
    app.HELPPVgen_2.Position = [496 127 45 20];

    % Create Inverter
    app.Inverter = uitab(app.TabGroup);

```

```

app.Inverter.Title = 'Inverter';
app.Inverter.BackgroundColor = [0.9686 0.9686 1];
app.Inverter.ForegroundColor = [0 0.4471 0.7412];

% Create PInomkWEeditFieldLabel
app.PInomkWEeditFieldLabel = uilabel(app.Inverter);
app.PInomkWEeditFieldLabel.HorizontalAlignment = 'right';
app.PInomkWEeditFieldLabel.FontName = 'Times New Roman';
app.PInomkWEeditFieldLabel.FontSize = 13;
app.PInomkWEeditFieldLabel.FontWeight = 'bold';
app.PInomkWEeditFieldLabel.Position = [15 345 75 22];
app.PInomkWEeditFieldLabel.Text = 'PInom (kW)';

% Create PInomI
app.PInomI = uieditfield(app.Inverter, 'numeric');
app.PInomI.Limits = [0 Inf];
app.PInomI.ValueChangedFcn = createCallbackFcn(app, ↵
@PInomIValueChanged, true);
app.PInomI.FontName = 'Times New Roman';
app.PInomI.FontSize = 13;
app.PInomI.FontWeight = 'bold';
app.PInomI.Position = [105 345 56 22];

% Create PImaxkWEeditFieldLabel
app.PImaxkWEeditFieldLabel = uilabel(app.Inverter);
app.PImaxkWEeditFieldLabel.HorizontalAlignment = 'right';
app.PImaxkWEeditFieldLabel.FontName = 'Times New Roman';
app.PImaxkWEeditFieldLabel.FontSize = 13;
app.PImaxkWEeditFieldLabel.FontWeight = 'bold';
app.PImaxkWEeditFieldLabel.Position = [14 320 75 22];
app.PImaxkWEeditFieldLabel.Text = 'PImax (kW)';

% Create PImaxI
app.PImaxI = uieditfield(app.Inverter, 'numeric');
app.PImaxI.Limits = [0 Inf];
app.PImaxI.ValueChangedFcn = createCallbackFcn(app, ↵
@PImaxIValueChanged, true);
app.PImaxI.FontName = 'Times New Roman';
app.PImaxI.FontSize = 13;
app.PImaxI.FontWeight = 'bold';
app.PImaxI.Position = [105 320 56 22];

% Create NominaloutputLabel
app.NominaloutputLabel = uilabel(app.Inverter);
app.NominaloutputLabel.FontName = 'Times New Roman';
app.NominaloutputLabel.FontSize = 13;
app.NominaloutputLabel.FontColor = [0.4667 0.6745 0.1882];
app.NominaloutputLabel.Visible = 'off';
app.NominaloutputLabel.Position = [178 345 204 22];
app.NominaloutputLabel.Text = 'Nominal output power of the inverter.';

% Create MaximumoutputLabel
app.MaximumoutputLabel = uilabel(app.Inverter);
app.MaximumoutputLabel.FontName = 'Times New Roman';
app.MaximumoutputLabel.FontSize = 13;
app.MaximumoutputLabel.FontColor = [0.4667 0.6745 0.1882];
app.MaximumoutputLabel.Visible = 'off';
app.MaximumoutputLabel.Position = [178 320 212 22];

```

```

app.MaximumoutputLabel.Text = 'Maximum output power of the inverter.';

% Create InverterCurveI
app.InverterCurveI = uibuttongroup(app.Inverter);
app.InverterCurveI.SelectionChangedFcn = createCallbackFcn(app, ↵
@InverterCurveISelectionChanged, true);
app.InverterCurveI.ForegroundColor = [0 0.4471 0.7412];
app.InverterCurveI.TitlePosition = 'centertop';
app.InverterCurveI.Title = 'Inverter Curve';
app.InverterCurveI.BackgroundColor = [0.8902 0.898 1];
app.InverterCurveI.FontName = 'Times New Roman';
app.InverterCurveI.FontWeight = 'bold';
app.InverterCurveI.FontSize = 13;
app.InverterCurveI.Position = [21 182 158 106];

% Create ModelparametersButton
app.ModelparametersButton = uitogglebutton(app.InverterCurveI);
app.ModelparametersButton.IconAlignment = 'center';
app.ModelparametersButton.Text = 'Model parameters';
app.ModelparametersButton.FontName = 'Times New Roman';
app.ModelparametersButton.Position = [11 47 135 22];
app.ModelparametersButton.Value = true;

% Create PowerefficiencycurveButton
app.PowerefficiencycurveButton = uitogglebutton(app.InverterCurveI);
app.PowerefficiencycurveButton.IconAlignment = 'center';
app.PowerefficiencycurveButton.Text = 'Power efficiency curve';
app.PowerefficiencycurveButton.FontName = 'Times New Roman';
app.PowerefficiencycurveButton.Position = [11 10 135 22];

% Create k0EditFieldLabel
app.k0EditFieldLabel = uilabel(app.Inverter);
app.k0EditFieldLabel.HorizontalAlignment = 'right';
app.k0EditFieldLabel.FontName = 'Times New Roman';
app.k0EditFieldLabel.FontSize = 13;
app.k0EditFieldLabel.FontWeight = 'bold';
app.k0EditFieldLabel.Position = [9 132 27 22];
app.k0EditFieldLabel.Text = 'k0';

% Create k0I
app.k0I = uieditfield(app.Inverter, 'numeric');
app.k0I.Limits = [0 Inf];
app.k0I.ValueChangedFcn = createCallbackFcn(app, @k0IValueChanged, ↵
true);
app.k0I.FontName = 'Times New Roman';
app.k0I.FontSize = 13;
app.k0I.FontWeight = 'bold';
app.k0I.Position = [49 132 41 22];

% Create k1EditFieldLabel
app.k1EditFieldLabel = uilabel(app.Inverter);
app.k1EditFieldLabel.HorizontalAlignment = 'right';
app.k1EditFieldLabel.FontName = 'Times New Roman';
app.k1EditFieldLabel.FontSize = 13;
app.k1EditFieldLabel.FontWeight = 'bold';
app.k1EditFieldLabel.Position = [9 102 27 22];
app.k1EditFieldLabel.Text = 'k1';

```

```

% Create k1I
app.k1I = uieditfield(app.Inverter, 'numeric');
app.k1I.Limits = [0 Inf];
app.k1I.ValueChangedFcn = createCallbackFcn(app, @k1IValueChanged, ↵
true);
app.k1I.FontName = 'Times New Roman';
app.k1I.FontSize = 13;
app.k1I.FontWeight = 'bold';
app.k1I.Position = [49 102 41 22];

% Create k2EditFieldLabel
app.k2EditFieldLabel = uilabel(app.Inverter);
app.k2EditFieldLabel.HorizontalAlignment = 'right';
app.k2EditFieldLabel.FontName = 'Times New Roman';
app.k2EditFieldLabel.FontSize = 13;
app.k2EditFieldLabel.FontWeight = 'bold';
app.k2EditFieldLabel.Position = [9 73 27 22];
app.k2EditFieldLabel.Text = 'k2';

% Create k2I
app.k2I = uieditfield(app.Inverter, 'numeric');
app.k2I.Limits = [0 Inf];
app.k2I.ValueChangedFcn = createCallbackFcn(app, @k2IValueChanged, ↵
true);
app.k2I.FontName = 'Times New Roman';
app.k2I.FontSize = 13;
app.k2I.FontWeight = 'bold';
app.k2I.Position = [49 73 41 22];

% Create pointsTable
app.pointsTable = uitable(app.Inverter);
app.pointsTable.ColumnName = {'pac'; 'Efficiency'};
app.pointsTable.ColumnWidth = {60, 60};
app.pointsTable.RowName = {};
app.pointsTable.ColumnEditable = true;
app.pointsTable.CellEditCallback = createCallbackFcn(app, ↵
@pointsTableCellEdit, true);
app.pointsTable.FontName = 'Times New Roman';
app.pointsTable.FontSize = 13;
app.pointsTable.Position = [237 46 123 185];

% Create PowerEfgraphic
app.PowerEfgraphic = uiaxes(app.Inverter);
title(app.PowerEfgraphic, 'Power efficiency curve')
xlabel(app.PowerEfgraphic, 'pac')
ylabel(app.PowerEfgraphic, 'Efficiency (%)')
app.PowerEfgraphic.FontName = 'Times New Roman';
app.PowerEfgraphic.XTick = [0 0.2 0.4 0.6 0.8 1];
app.PowerEfgraphic.YTick = [0 20 40 60 80];
app.PowerEfgraphic.YTickLabel = {'0'; '20'; '40'; '60'; '80'};
app.PowerEfgraphic.BackgroundColor = [0.8902 0.902 1];
app.PowerEfgraphic.Position = [376 50 223 178];

% Create INVERTERLabel
app.INVERTERLabel = uilabel(app.Inverter);
app.INVERTERLabel.FontName = 'Times New Roman';
app.INVERTERLabel.FontSize = 16;
app.INVERTERLabel.FontWeight = 'bold';

```

```

app.INVERTERLabel.FontAngle = 'italic';
app.INVERTERLabel.FontColor = [0 0.4471 0.7412];
app.INVERTERLabel.Position = [482 355 87 22];
app.INVERTERLabel.Text = 'INVERTER';

% Create efficiencyLabel1
app.efficiencyLabel1 = uilabel(app.Inverter);
app.efficiencyLabel1.FontName = 'Times New Roman';
app.efficiencyLabel1.FontSize = 13;
app.efficiencyLabel1.FontWeight = 'bold';
app.efficiencyLabel1.FontColor = [0.4667 0.6745 0.1882];
app.efficiencyLabel1.Visible = 'off';
app.efficiencyLabel1.Position = [202 287 151 22];
app.efficiencyLabel1.Text = 'Definition of the efficiency';

% Create efficiencyLabel2
app.efficiencyLabel2 = uilabel(app.Inverter);
app.efficiencyLabel2.FontName = 'Times New Roman';
app.efficiencyLabel2.FontSize = 13;
app.efficiencyLabel2.FontColor = [0.4667 0.6745 0.1882];
app.efficiencyLabel2.Visible = 'off';
app.efficiencyLabel2.Position = [202 269 173 22];
app.efficiencyLabel2.Text = '1. Model parameters (k0,k1,k2) ';

% Create efficiencyLabel3
app.efficiencyLabel3 = uilabel(app.Inverter);
app.efficiencyLabel3.FontName = 'Times New Roman';
app.efficiencyLabel3.FontSize = 13;
app.efficiencyLabel3.FontColor = [0.4667 0.6745 0.1882];
app.efficiencyLabel3.Visible = 'off';
app.efficiencyLabel3.Position = [202 252 417 22];
app.efficiencyLabel3.Text = '2. Points of the power efficiency curve ↵
(the program calculates k0, k1 and k2 )';

% Create efficiencyLabel4
app.efficiencyLabel4 = uilabel(app.Inverter);
app.efficiencyLabel4.FontName = 'Times New Roman';
app.efficiencyLabel4.FontSize = 13;
app.efficiencyLabel4.FontColor = [0.4667 0.6745 0.1882];
app.efficiencyLabel4.Visible = 'off';
app.efficiencyLabel4.Position = [202 235 305 22];
app.efficiencyLabel4.Text = 'using the power efficiency points ↵
indicated in the table) ';

% Create k0Label
app.k0Label = uilabel(app.Inverter);
app.k0Label.FontName = 'Times New Roman';
app.k0Label.FontSize = 13;
app.k0Label.FontColor = [0.4667 0.6745 0.1882];
app.k0Label.Visible = 'off';
app.k0Label.Position = [100 132 129 22];
app.k0Label.Text = 'No-load inverter losses.';

% Create k1Label
app.k1Label = uilabel(app.Inverter);
app.k1Label.FontName = 'Times New Roman';
app.k1Label.FontSize = 13;
app.k1Label.FontColor = [0.4667 0.6745 0.1882];

```

```

app.k1Label.Visible = 'off';
app.k1Label.Position = [100 102 120 22];
app.k1Label.Text = 'Linear inverter losses.';

% Create k2Label
app.k2Label = uilabel(app.Inverter);
app.k2Label.FontName = 'Times New Roman';
app.k2Label.FontSize = 13;
app.k2Label.FontColor = [0.4667 0.6745 0.1882];
app.k2Label.Visible = 'off';
app.k2Label.Position = [100 73 114 22];
app.k2Label.Text = 'Joule inverter losses.';

% Create pacLabel
app.pacLabel = uilabel(app.Inverter);
app.pacLabel.FontName = 'Times New Roman';
app.pacLabel.FontSize = 13;
app.pacLabel.FontColor = [0.4667 0.6745 0.1882];
app.pacLabel.Visible = 'off';
app.pacLabel.Position = [154 22 474 22];
app.pacLabel.Text = 'pac (normalised output power): ratio of the output power to the nominal inverter power.';

% Create EfficiencyLabel
app.EfficiencyLabel = uilabel(app.Inverter);
app.EfficiencyLabel.FontName = 'Times New Roman';
app.EfficiencyLabel.FontSize = 13;
app.EfficiencyLabel.FontColor = [0.4667 0.6745 0.1882];
app.EfficiencyLabel.Visible = 'off';
app.EfficiencyLabel.Position = [154 4 253 22];
app.EfficiencyLabel.Text = 'Efficiency (power efficiency) = 100*PAC/PDC';

% Create HELPSwitchLabel_8
app.HELPswitchLabel_8 = uilabel(app.Inverter);
app.HELPswitchLabel_8.HorizontalAlignment = 'center';
app.HELPswitchLabel_8.FontName = 'Times New Roman';
app.HELPswitchLabel_8.FontSize = 13;
app.HELPswitchLabel_8.FontWeight = 'bold';
app.HELPswitchLabel_8.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_8.Position = [553.5 315 41 22];
app.HELPswitchLabel_8.Text = 'HELP';

% Create HELPINverter
app.HELPInverter = uiswitch(app.Inverter, 'slider');
app.HELPInverter.ValueChangedFcn = createCallbackFcn(app, @HELPInverterValueChanged, true);
app.HELPInverter.FontName = 'Times New Roman';
app.HELPInverter.FontSize = 13;
app.HELPInverter.FontWeight = 'bold';
app.HELPInverter.FontColor = [0.4667 0.6745 0.1882];
app.HELPInverter.Position = [467 315 45 20];

% Create Wiring
app.Wiring = uitab(app.TabGroup);
app.Wiring.Title = 'Wiring';
app.Wiring.BackgroundColor = [0.9686 0.9686 1];
app.Wiring.ForegroundColor = [0 0.4471 0.7412];

```

```

% Create WDCLabel
app.WDCLabel = uilabel(app.Wiring);
app.WDCLabel.HorizontalAlignment = 'right';
app.WDCLabel.FontName = 'Times New Roman';
app.WDCLabel.FontSize = 15;
app.WDCLabel.FontWeight = 'bold';
app.WDCLabel.Position = [105 240 71 22];
app.WDCLabel.Text = 'WDC (%)';

% Create WDCI
app.WDCI = uieditfield(app.Wiring, 'numeric');
app.WDCI.Limits = [0 10];
app.WDCI.ValueChangedFcn = createCallbackFcn(app, @WDCIValueChanged, ↵
true);
app.WDCI.FontName = 'Times New Roman';
app.WDCI.FontSize = 15;
app.WDCI.FontWeight = 'bold';
app.WDCI.Position = [191 240 67 22];

% Create WACLabel
app.WACLabel = uilabel(app.Wiring);
app.WACLabel.HorizontalAlignment = 'right';
app.WACLabel.FontName = 'Times New Roman';
app.WACLabel.FontSize = 15;
app.WACLabel.FontWeight = 'bold';
app.WACLabel.Position = [334 123 69 22];
app.WACLabel.Text = 'WAC (%)';

% Create WACI
app.WACI = uieditfield(app.Wiring, 'numeric');
app.WACI.Limits = [0 10];
app.WACI.ValueChangedFcn = createCallbackFcn(app, @WACIValueChanged, ↵
true);
app.WACI.FontName = 'Times New Roman';
app.WACI.FontSize = 15;
app.WACI.FontWeight = 'bold';
app.WACI.Position = [418 123 67 22];

% Create DCLabel
app.DCLabel = uilabel(app.Wiring);
app.DCLabel.FontName = 'Times New Roman';
app.DCLabel.FontSize = 15;
app.DCLabel.FontColor = [0.4667 0.6745 0.1882];
app.DCLabel.Position = [66 279 177 22];
app.DCLabel.Text = 'DC wiring losses (0 to 10%)';

% Create ACLabel
app.ACLabel = uilabel(app.Wiring);
app.ACLabel.FontName = 'Times New Roman';
app.ACLabel.FontSize = 15;
app.ACLabel.FontColor = [0.4667 0.6745 0.1882];
app.ACLabel.Position = [336 163 177 22];
app.ACLabel.Text = 'AC wiring losses (0 to 10%)';

% Create WIRINGLabel
app.WIRINGLabel = uilabel(app.Wiring);
app.WIRINGLabel.FontName = 'Times New Roman';

```

```

app.WIRINGLabel.FontSize = 16;
app.WIRINGLabel.FontWeight = 'bold';
app.WIRINGLabel.FontAngle = 'italic';
app.WIRINGLabel.FontColor = [0 0.4471 0.7412];
app.WIRINGLabel.Position = [488 329 66 22];
app.WIRINGLabel.Text = 'WIRING';

% Create Bat
app.Bat = uitab(app.TabGroup);
app.Bat.Title = 'Battery';
app.Bat.BackgroundColor = [0.9686 0.9686 1];
app.Bat.ForegroundColor = [0 0.4471 0.7412];

% Create CBATkWhLabel
app.CBATkWhLabel = uilabel(app.Bat);
app.CBATkWhLabel.HorizontalAlignment = 'right';
app.CBATkWhLabel.FontName = 'Times New Roman';
app.CBATkWhLabel.FontSize = 13;
app.CBATkWhLabel.FontWeight = 'bold';
app.CBATkWhLabel.Position = [206 263 80 22];
app.CBATkWhLabel.Text = 'CBAT (kWh)';

% Create CBATI
app.CBATI = uieditfield(app.Bat, 'numeric');
app.CBATI.ValueChangedFcn = createCallbackFcn(app, @CBATIValueChanged, ↵
true);
app.CBATI.FontName = 'Times New Roman';
app.CBATI.FontSize = 13;
app.CBATI.FontWeight = 'bold';
app.CBATI.Position = [301 263 100 22];

% Create SOCmaxEditFieldLabel
app.SOCmaxEditFieldLabel = uilabel(app.Bat);
app.SOCmaxEditFieldLabel.HorizontalAlignment = 'right';
app.SOCmaxEditFieldLabel.FontName = 'Times New Roman';
app.SOCmaxEditFieldLabel.FontSize = 13;
app.SOCmaxEditFieldLabel.FontWeight = 'bold';
app.SOCmaxEditFieldLabel.Position = [220 162 56 22];
app.SOCmaxEditFieldLabel.Text = 'SOCmax';

% Create SOCmaxI
app.SOCmaxI = uieditfield(app.Bat, 'numeric');
app.SOCmaxI.ValueChangedFcn = createCallbackFcn(app, ↵
@SOCmaxIValueChanged, true);
app.SOCmaxI.FontName = 'Times New Roman';
app.SOCmaxI.FontSize = 13;
app.SOCmaxI.FontWeight = 'bold';
app.SOCmaxI.Position = [291 162 100 22];

% Create SOCminEditFieldLabel
app.SOCminEditFieldLabel = uilabel(app.Bat);
app.SOCminEditFieldLabel.HorizontalAlignment = 'right';
app.SOCminEditFieldLabel.FontName = 'Times New Roman';
app.SOCminEditFieldLabel.FontSize = 13;
app.SOCminEditFieldLabel.FontWeight = 'bold';
app.SOCminEditFieldLabel.Position = [221 59 54 22];
app.SOCminEditFieldLabel.Text = 'SOCmin';

```

```

% Create SOCminI
app.SOCminI = uieditfield(app.Bat, 'numeric');
app.SOCminI.ValueChangedFcn = createCallbackFcn(app, ↵
@SOCminIValueChanged, true);
    app.SOCminI.FontName = 'Times New Roman';
    app.SOCminI.FontSize = 13;
    app.SOCminI.FontWeight = 'bold';
    app.SOCminI.Position = [290 59 100 22];

% Create SOCmaxLabel
app.SOCmaxLabel = uilabel(app.Bat);
app.SOCmaxLabel.FontName = 'Times New Roman';
app.SOCmaxLabel.FontSize = 13;
app.SOCmaxLabel.FontColor = [0.4667 0.6745 0.1882];
app.SOCmaxLabel.Visible = 'off';
app.SOCmaxLabel.Position = [44 195 510 22];
app.SOCmaxLabel.Text = 'Maximum state of charge. The PV generator is ↵
disconnected when the SOC exceeds this value.';

% Create SOCminLabel
app.SOCminLabel = uilabel(app.Bat);
app.SOCminLabel.FontName = 'Times New Roman';
app.SOCminLabel.FontSize = 13;
app.SOCminLabel.FontColor = [0.4667 0.6745 0.1882];
app.SOCminLabel.Visible = 'off';
app.SOCminLabel.Position = [28 94 542 22];
app.SOCminLabel.Text = 'Minimum state of charge. Inverter and loads are ↵
disconnected when SOC decreases below this value.';

% Create CBATLabel
app.CBATLabel = uilabel(app.Bat);
app.CBATLabel.FontName = 'Times New Roman';
app.CBATLabel.FontSize = 13;
app.CBATLabel.FontColor = [0.4667 0.6745 0.1882];
app.CBATLabel.Visible = 'off';
app.CBATLabel.Position = [261 292 91 22];
app.CBATLabel.Text = 'Battery capacity';

% Create BATTERYStandalonePVsystemsLabel
app.BATTERYStandalonePVsystemsLabel = uilabel(app.Bat);
app.BATTERYStandalonePVsystemsLabel.FontName = 'Times New Roman';
app.BATTERYStandalonePVsystemsLabel.FontSize = 16;
app.BATTERYStandalonePVsystemsLabel.FontWeight = 'bold';
app.BATTERYStandalonePVsystemsLabel.FontAngle = 'italic';
app.BATTERYStandalonePVsystemsLabel.FontColor = [0 0.4471 0.7412];
app.BATTERYStandalonePVsystemsLabel.Position = [177 341 251 22];
app.BATTERYStandalonePVsystemsLabel.Text = 'BATTERY (Stand-alone PV ↵
systems)';

% Create HELPSwitchLabel_5
app.HELPswitchLabel_5 = uilabel(app.Bat);
app.HELPswitchLabel_5.HorizontalAlignment = 'center';
app.HELPswitchLabel_5.FontName = 'Times New Roman';
app.HELPswitchLabel_5.FontSize = 13;
app.HELPswitchLabel_5.FontWeight = 'bold';
app.HELPswitchLabel_5.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_5.Position = [78.5 281 41 22];
app.HELPswitchLabel_5.Text = 'HELP';

```

```

% Create HELPBattery
app.HELPBattery = uiswitch(app.Bat, 'toggle');
app.HELPBattery.ValueChangedFcn = createCallbackFcn(app, @(HELBatteryValueChanged, true));
app.HELPBattery.FontName = 'Times New Roman';
app.HELPBattery.FontSize = 13;
app.HELPBattery.FontWeight = 'bold';
app.HELPBattery.FontColor = [0.4667 0.6745 0.1882];
app.HELPBattery.Position = [44 276 20 45];

% Create Load
app.Load = uitab(app.TabGroup);
app.Load.Title = 'Load';
app.Load.BackgroundColor = [0.9686 0.9686 1];
app.Load.ForegroundColor = [0 0.4471 0.7412];

% Create LdmGraphic
app.LdmGraphic = uiaxes(app.Load);
title(app.LdmGraphic, '')
xlabel(app.LdmGraphic, 'Month')
ylabel(app.LdmGraphic, 'Ldm [kWh/day]')
app.LdmGraphic.FontName = 'Times New Roman';
app.LdmGraphic.XTick = [1 4 7 10];
app.LdmGraphic.XTickLabel = {'January'; 'April'; 'July'; 'October'};
app.LdmGraphic.YTick = [0 50 100];
app.LdmGraphic.YTickLabel = {'0'; '50'; '100'};
app.LdmGraphic.BackgroundColor = [0.8902 0.898 1];
app.LdmGraphic.Position = [140 230 242 131];

% Create LdmI
app.LdmI = uitable(app.Load);
app.LdmI.ColumnName = {'Ldm'};
app.LdmI.ColumnWidth = {40};
app.LdmI.RowName = {'January'; 'February'; 'March'; 'April'; 'May'; ...
'June'; 'July'; 'August'; 'September'; 'October'; 'November'; 'December'};
app.LdmI.ColumnEditable = true;
app.LdmI.CellEditCallback = createCallbackFcn(app, @LdmICellEdit2, true);
app.LdmI.FontName = 'Times New Roman';
app.LdmI.FontSize = 13;
app.LdmI.Position = [12 49 118 317];

% Create TotalEditFieldLabel
app.TotalEditFieldLabel = uilabel(app.Load);
app.TotalEditFieldLabel.HorizontalAlignment = 'right';
app.TotalEditFieldLabel.FontName = 'Times New Roman';
app.TotalEditFieldLabel.FontSize = 13;
app.TotalEditFieldLabel.FontWeight = 'bold';
app.TotalEditFieldLabel.Position = [-2 20 34 22];
app.TotalEditFieldLabel.Text = 'Total';

% Create TotalLdmI
app.TotalLdmI = uieditfield(app.Load, 'numeric');
app.TotalLdmI.ValueDisplayFormat = '%.1f';
app.TotalLdmI.Editable = 'off';
app.TotalLdmI.FontName = 'Times New Roman';
app.TotalLdmI.FontSize = 13;

```

```

app.TotalLdmI.FontWeight = 'bold';
app.TotalLdmI.Position = [47 20 75 22];

% Create LdmLabel
app.LdmLabel = uilabel(app.Load);
app.LdmLabel.FontName = 'Times New Roman';
app.LdmLabel.FontSize = 13;
app.LdmLabel.FontColor = [0.4667 0.6745 0.1882];
app.LdmLabel.Visible = 'off';
app.LdmLabel.Position = [6 368 341 22];
app.LdmLabel.Text = 'Monthly average of daily energy consumption, Ldm ↵
[kWh/day]';

% Create FhI
app.FhI = uitable(app.Load);
app.FhI.ColumnName = {'F(h)'};
app.FhI.ColumnWidth = {65};
app.FhI.RowName = {'1'; '2'; '3'; '4'; '5'; '6'; '7'; '8'; '9'; '10'; ↵
'11'; '12'; '13'; '14'; '15'; '16'; '17'; '18'; '19'; '20'; '21'; '22'; '23'; ↵
'24'};
app.FhI.ColumnEditable = true;
app.FhI.CellEditCallback = createCallbackFcn(app, @FhICellEdit, true);
app.FhI.FontName = 'Times New Roman';
app.FhI.FontSize = 13;
app.FhI.Position = [479 100 136 266];

% Create FhGraphic
app.FhGraphic = uiaxes(app.Load);
title(app.FhGraphic, '')
xlabel(app.FhGraphic, 'Hour')
ylabel(app.FhGraphic, 'Hourly fraction, F(h)')
app.FhGraphic.FontName = 'Times New Roman';
app.FhGraphic.XTick = [0 4 8 12 16 20 24];
app.FhGraphic.XTickLabel = {'0'; '4'; '8'; '12'; '16'; '20'; '24'};
app.FhGraphic.YTick = [0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16];
app.FhGraphic.YTickLabel = {'0'; '0.02'; '0.04'; '0.06'; '0.08'; '0.1'; ↵
'0.12'; '0.14'; '0.16'};
app.FhGraphic.BackgroundColor = [0.8902 0.898 1];
app.FhGraphic.Position = [211 18 258 149];

% Create FhLabel1
app.FhLabel1 = uilabel(app.Load);
app.FhLabel1.FontName = 'Times New Roman';
app.FhLabel1.FontSize = 13;
app.FhLabel1.FontColor = [0.4667 0.6745 0.1882];
app.FhLabel1.Visible = 'off';
app.FhLabel1.Position = [189 191 281 22];
app.FhLabel1.Text = 'F(h) indicates the fraction of daily energy ↵
consumed';

% Create FhLabel2
app.FhLabel2 = uilabel(app.Load);
app.FhLabel2.FontName = 'Times New Roman';
app.FhLabel2.FontSize = 13;
app.FhLabel2.FontColor = [0.4667 0.6745 0.1882];
app.FhLabel2.Visible = 'off';
app.FhLabel2.Position = [185 175 282 22];
app.FhLabel2.Text = ' during the hour h. Constant load profile: F(h) ↵

```

```

=1/24)';

% Create FhLabel3
app.FhLabel3 = uilabel(app.Load);
app.FhLabel3.FontName = 'Times New Roman';
app.FhLabel3.FontSize = 13;
app.FhLabel3.FontColor = [0.4667 0.6745 0.1882];
app.FhLabel3.Position = [474 67 151 22];
app.FhLabel3.Text = 'The sum of hourly fractions';

% Create FhLabel4
app.FhLabel4 = uilabel(app.Load);
app.FhLabel4.FontName = 'Times New Roman';
app.FhLabel4.FontSize = 13;
app.FhLabel4.FontColor = [0.4667 0.6745 0.1882];
app.FhLabel4.Position = [472 54 63 22];
app.FhLabel4.Text = ' must be 1:';

% Create TotalEditFieldLabel_2
app.TotalEditFieldLabel_2 = uilabel(app.Load);
app.TotalEditFieldLabel_2.HorizontalAlignment = 'right';
app.TotalEditFieldLabel_2.FontName = 'Times New Roman';
app.TotalEditFieldLabel_2.FontSize = 13;
app.TotalEditFieldLabel_2.FontWeight = 'bold';
app.TotalEditFieldLabel_2.Position = [499 35 34 22];
app.TotalEditFieldLabel_2.Text = 'Total';

% Create TotalFh
app.TotalFh = uieditfield(app.Load, 'numeric');
app.TotalFh.ValueDisplayFormat = '%.4f';
app.TotalFh.Editable = 'off';
app.TotalFh.FontName = 'Times New Roman';
app.TotalFh.FontSize = 13;
app.TotalFh.FontWeight = 'bold';
app.TotalFh.Position = [548 35 62 22];
app.TotalFh.Value = 1;

% Create HELPSwitchLabel_6
app.HELPSwitchLabel_6 = uilabel(app.Load);
app.HELPSwitchLabel_6.HorizontalAlignment = 'center';
app.HELPSwitchLabel_6.FontName = 'Times New Roman';
app.HELPSwitchLabel_6.FontSize = 13;
app.HELPSwitchLabel_6.FontWeight = 'bold';
app.HELPSwitchLabel_6.FontColor = [0.4667 0.6745 0.1882];
app.HELPSwitchLabel_6.Position = [405 232 41 22];
app.HELPSwitchLabel_6.Text = 'HELP';

% Create HELPLoad
app.HELPLoad = uiswitch(app.Load, 'slider');
app.HELPLoad.Orientation = 'vertical';
app.HELPLoad.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPLoadValueChanged, true);
app.HELPLoad.FontName = 'Times New Roman';
app.HELPLoad.FontSize = 13;
app.HELPLoad.FontWeight = 'bold';
app.HELPLoad.FontColor = [0.4667 0.6745 0.1882];
app.HELPLoad.Position = [411 276 20 45];

```

```

% Create LOADPROFILESLabel
app.LOADPROFILESLabel = uilabel(app.Load);
app.LOADPROFILESLabel.FontName = 'Times New Roman';
app.LOADPROFILESLabel.FontSize = 16;
app.LOADPROFILESLabel.FontWeight = 'bold';
app.LOADPROFILESLabel.FontAngle = 'italic';
app.LOADPROFILESLabel.FontColor = [0 0.4471 0.7412];
app.LOADPROFILESLabel.Position = [348 362 131 22];
app.LOADPROFILESLabel.Text = 'LOAD PROFILES';

% Create GenSet
app.GenSet = uitab(app.TabGroup);
app.GenSet.Title = 'GenSet';
app.GenSet.BackgroundColor = [0.9686 0.9686 1];
app.GenSet.ForegroundColor = [0 0.4471 0.7412];

% Create PGENnomkWLabel
app.PGENnomkWLabel = uilabel(app.GenSet);
app.PGENnomkWLabel.HorizontalAlignment = 'right';
app.PGENnomkWLabel.FontName = 'Times New Roman';
app.PGENnomkWLabel.FontSize = 13;
app.PGENnomkWLabel.FontWeight = 'bold';
app.PGENnomkWLabel.Position = [2 335 99 22];
app.PGENnomkWLabel.Text = 'PGENnom (kW)';

% Create PGENnomI
app.PGENnomI = uieditfield(app.GenSet, 'numeric');
app.PGENnomI.Limits = [0 Inf];
app.PGENnomI.ValueChangedFcn = createCallbackFcn(app, ↵
@PGENnomIValueChanged, true);
app.PGENnomI.FontName = 'Times New Roman';
app.PGENnomI.FontSize = 13;
app.PGENnomI.FontWeight = 'bold';
app.PGENnomI.Position = [104 335 56 22];

% Create SOCstartEditFieldLabel
app.SOCstartEditFieldLabel = uilabel(app.GenSet);
app.SOCstartEditFieldLabel.HorizontalAlignment = 'right';
app.SOCstartEditFieldLabel.FontName = 'Times New Roman';
app.SOCstartEditFieldLabel.FontSize = 13;
app.SOCstartEditFieldLabel.FontWeight = 'bold';
app.SOCstartEditFieldLabel.Position = [3 308 58 22];
app.SOCstartEditFieldLabel.Text = 'SOCstart';

% Create SOCstartI
app.SOCstartI = uieditfield(app.GenSet, 'numeric');
app.SOCstartI.ValueChangedFcn = createCallbackFcn(app, ↵
@SOCstartIValueChanged, true);
app.SOCstartI.FontName = 'Times New Roman';
app.SOCstartI.FontSize = 13;
app.SOCstartI.FontWeight = 'bold';
app.SOCstartI.Position = [104 308 57 22];

% Create SOCstopEditFieldLabel
app.SOCstopEditFieldLabel = uilabel(app.GenSet);
app.SOCstopEditFieldLabel.HorizontalAlignment = 'right';
app.SOCstopEditFieldLabel.FontName = 'Times New Roman';
app.SOCstopEditFieldLabel.FontSize = 13;

```

```

app.SOCstopEditFieldLabel.FontWeight = 'bold';
app.SOCstopEditFieldLabel.Position = [3 281 55 22];
app.SOCstopEditFieldLabel.Text = 'SOCstop';

% Create SOCstopI
app.SOCstopI = uieditfield(app.GenSet, 'numeric');
app.SOCstopI.ValueChangedFcn = createCallbackFcn(app, ↵
@SOCstopIValueChanged, true);
app.SOCstopI.FontName = 'Times New Roman';
app.SOCstopI.FontSize = 13;
app.SOCstopI.FontWeight = 'bold';
app.SOCstopI.Position = [104 281 56 22];

% Create b0iEditFieldLabel
app.b0iEditFieldLabel = uilabel(app.GenSet);
app.b0iEditFieldLabel.HorizontalAlignment = 'right';
app.b0iEditFieldLabel.FontName = 'Times New Roman';
app.b0iEditFieldLabel.FontSize = 13;
app.b0iEditFieldLabel.FontWeight = 'bold';
app.b0iEditFieldLabel.Position = [401 314 25 22];
app.b0iEditFieldLabel.Text = 'b0i';

% Create b0ii
app.b0ii = uieditfield(app.GenSet, 'numeric');
app.b0ii.ValueDisplayFormat = '%.4f';
app.b0ii.ValueChangedFcn = createCallbackFcn(app, @b0iiValueChanged, ↵
true);
app.b0ii.FontName = 'Times New Roman';
app.b0ii.FontSize = 13;
app.b0ii.FontWeight = 'bold';
app.b0ii.Position = [441 314 72 22];

% Create bliEditFieldLabel
app.bliEditFieldLabel = uilabel(app.GenSet);
app.bliEditFieldLabel.HorizontalAlignment = 'right';
app.bliEditFieldLabel.FontName = 'Times New Roman';
app.bliEditFieldLabel.FontSize = 13;
app.bliEditFieldLabel.FontWeight = 'bold';
app.bliEditFieldLabel.Position = [401 290 25 22];
app.bliEditFieldLabel.Text = 'bli';

% Create blii
app.blii = uieditfield(app.GenSet, 'numeric');
app.blii.ValueDisplayFormat = '%.4f';
app.blii.ValueChangedFcn = createCallbackFcn(app, @bliiValueChanged, ↵
true);
app.blii.FontName = 'Times New Roman';
app.blii.FontSize = 13;
app.blii.FontWeight = 'bold';
app.blii.Position = [441 290 72 22];

% Create b0sEditFieldLabel
app.b0sEditFieldLabel = uilabel(app.GenSet);
app.b0sEditFieldLabel.HorizontalAlignment = 'right';
app.b0sEditFieldLabel.FontName = 'Times New Roman';
app.b0sEditFieldLabel.FontSize = 13;
app.b0sEditFieldLabel.FontWeight = 'bold';
app.b0sEditFieldLabel.Position = [401 266 25 22];

```

```

app.b0sEditFieldLabel.Text = 'b0s';

% Create b0sI
app.b0sI = uieditfield(app.GenSet, 'numeric');
app.b0sI.ValueDisplayFormat = '%.4f';
app.b0sI.ValueChangedFcn = createCallbackFcn(app, @b0sIValueChanged, ↵
true);
app.b0sI.FontName = 'Times New Roman';
app.b0sI.FontSize = 13;
app.b0sI.FontWeight = 'bold';
app.b0sI.Position = [441 266 72 22];

% Create b1sEditFieldLabel
app.b1sEditFieldLabel = uilabel(app.GenSet);
app.b1sEditFieldLabel.HorizontalAlignment = 'right';
app.b1sEditFieldLabel.FontName = 'Times New Roman';
app.b1sEditFieldLabel.FontSize = 13;
app.b1sEditFieldLabel.FontWeight = 'bold';
app.b1sEditFieldLabel.Position = [401 242 25 22];
app.b1sEditFieldLabel.Text = 'b1s';

% Create b1sI
app.b1sI = uieditfield(app.GenSet, 'numeric');
app.b1sI.ValueDisplayFormat = '%.4f';
app.b1sI.ValueChangedFcn = createCallbackFcn(app, @b1sIValueChanged, ↵
true);
app.b1sI.FontName = 'Times New Roman';
app.b1sI.FontSize = 13;
app.b1sI.FontWeight = 'bold';
app.b1sI.Position = [441 242 72 22];

% Create b0iUnitsLabel
app.b0iUnitsLabel = uilabel(app.GenSet);
app.b0iUnitsLabel.FontName = 'Times New Roman';
app.b0iUnitsLabel.FontSize = 13;
app.b0iUnitsLabel.Position = [521 314 27 22];
app.b0iUnitsLabel.Text = '(l/h)';

% Create bliUnitsLabel
app.bliUnitsLabel = uilabel(app.GenSet);
app.bliUnitsLabel.FontName = 'Times New Roman';
app.bliUnitsLabel.FontSize = 13;
app.bliUnitsLabel.Position = [521 290 73 22];
app.bliUnitsLabel.Text = '(l/h)/kWnom';

% Create b0sUnitsLabel
app.b0sUnitsLabel = uilabel(app.GenSet);
app.b0sUnitsLabel.FontName = 'Times New Roman';
app.b0sUnitsLabel.FontSize = 13;
app.b0sUnitsLabel.Position = [521 266 50 22];
app.b0sUnitsLabel.Text = '(l/h)/kW';

% Create b1sUnitsLabel
app.b1sUnitsLabel = uilabel(app.GenSet);
app.b1sUnitsLabel.FontName = 'Times New Roman';
app.b1sUnitsLabel.FontSize = 13;
app.b1sUnitsLabel.Position = [521 242 105 22];
app.b1sUnitsLabel.Text = '(l/h)/(kWnom·kW)';

```

```

% Create PGENnomLabel
app.PGENnomLabel = uilabel(app.GenSet);
app.PGENnomLabel.FontName = 'Times New Roman';
app.PGENnomLabel.FontSize = 13;
app.PGENnomLabel.FontColor = [0.4667 0.6745 0.1882];
app.PGENnomLabel.Visible = 'off';
app.PGENnomLabel.Position = [164 335 156 22];
app.PGENnomLabel.Text = 'Nominal power of the genset';

% Create SOCstartLabel
app.SOCstartLabel = uilabel(app.GenSet);
app.SOCstartLabel.FontName = 'Times New Roman';
app.SOCstartLabel.FontSize = 13;
app.SOCstartLabel.FontColor = [0.4667 0.6745 0.1882];
app.SOCstartLabel.Visible = 'off';
app.SOCstartLabel.Position = [163 308 219 22];
app.SOCstartLabel.Text = 'State of charge for connecting the genset';

% Create SOCstopLabel
app.SOCstopLabel = uilabel(app.GenSet);
app.SOCstopLabel.FontName = 'Times New Roman';
app.SOCstopLabel.FontSize = 13;
app.SOCstopLabel.FontColor = [0.4667 0.6745 0.1882];
app.SOCstopLabel.Visible = 'off';
app.SOCstopLabel.Position = [164 281 234 22];
app.SOCstopLabel.Text = 'State of charge for disconnecting the genset';

% Create FuelCons
app.FuelCons = uitable(app.GenSet);
app.FuelCons.ColumnName = {'PGEN [kW]'; 'F [l/h]'};
app.FuelCons.ColumnWidth = {80, 60};
app.FuelCons.RowName = {};
app.FuelCons.FontName = 'Times New Roman';
app.FuelCons.FontSize = 13;
app.FuelCons.Position = [16 55 148 159];

% Create FuelConsGraph
app.FuelConsGraph = uiaxes(app.GenSet);
title(app.FuelConsGraph, 'Fuel consumption curve')
xlabel(app.FuelConsGraph, 'Genset power, kW')
ylabel(app.FuelConsGraph, 'Fuel consumption, l/h')
app.FuelConsGraph.FontName = 'Times New Roman';
app.FuelConsGraph.XTick = [0 5 10 15];
app.FuelConsGraph.XTickLabel = {'0'; '5'; '10'; '15'};
app.FuelConsGraph.YTick = [0 1 2 3 4 5 6 7];
app.FuelConsGraph.YTickLabel = {'0'; '1'; '2'; '3'; '4'; '5'; '6'; ↵
'7'};
app.FuelConsGraph.BackgroundColor = [0.8902 0.898 1];
app.FuelConsGraph.Position = [183 64 233 156];

% Create b0iLabel
app.b0iLabel = uilabel(app.GenSet);
app.b0iLabel.FontName = 'Times New Roman';
app.b0iLabel.FontSize = 13;
app.b0iLabel.FontColor = [0.4667 0.6745 0.1882];
app.b0iLabel.Visible = 'off';
app.b0iLabel.Position = [452 186 143 22];

```

```

app.b0iLabel.Text = 'b0i: Intercept coefficient 0';

% Create bliLabel
app.bliLabel = uilabel(app.GenSet);
app.bliLabel.FontName = 'Times New Roman';
app.bliLabel.FontSize = 13;
app.bliLabel.FontColor = [0.4667 0.6745 0.1882];
app.bliLabel.Visible = 'off';
app.bliLabel.Position = [452 165 143 22];
app.bliLabel.Text = 'bli: Intercept coefficient 1';

% Create b0sLabel
app.b0sLabel = uilabel(app.GenSet);
app.b0sLabel.FontName = 'Times New Roman';
app.b0sLabel.FontSize = 13;
app.b0sLabel.FontColor = [0.4667 0.6745 0.1882];
app.b0sLabel.Visible = 'off';
app.b0sLabel.Position = [452 144 128 22];
app.b0sLabel.Text = 'b0s: Slope coefficient 0';

% Create b1sLabel
app.b1sLabel = uilabel(app.GenSet);
app.b1sLabel.FontName = 'Times New Roman';
app.b1sLabel.FontSize = 13;
app.b1sLabel.FontColor = [0.4667 0.6745 0.1882];
app.b1sLabel.Visible = 'off';
app.b1sLabel.Position = [452 123 128 22];
app.b1sLabel.Text = 'b1s: Slope coefficient 1';

% Create HELPSwitchLabel
app.HELPSSwitchLabel = uilabel(app.GenSet);
app.HELPSSwitchLabel.HorizontalAlignment = 'center';
app.HELPSSwitchLabel.FontName = 'Times New Roman';
app.HELPSSwitchLabel.FontSize = 13;
app.HELPSSwitchLabel.FontWeight = 'bold';
app.HELPSSwitchLabel.FontColor = [0.4667 0.6745 0.1882];
app.HELPSSwitchLabel.Position = [498.5 51 41 22];
app.HELPSSwitchLabel.Text = 'HELP';

% Create HELPGenSet
app.HELPGenSet = uiswitch(app.GenSet, 'slider');
app.HELPGenSet.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPGenSetValueChanged, true);
app.HELPGenSet.FontName = 'Times New Roman';
app.HELPGenSet.FontSize = 13;
app.HELPGenSet.FontWeight = 'bold';
app.HELPGenSet.FontColor = [0.4667 0.6745 0.1882];
app.HELPGenSet.Position = [500 76 38 17];

% Create GENERATORSET
app.GENERATORSET = uilabel(app.GenSet);
app.GENERATORSET.FontName = 'Times New Roman';
app.GENERATORSET.FontSize = 16;
app.GENERATORSET.FontWeight = 'bold';
app.GENERATORSET.FontAngle = 'italic';
app.GENERATORSET.FontColor = [0 0.4471 0.7412];
app.GENERATORSET.Position = [415 356 135 22];
app.GENERATORSET.Text = 'GENERATOR SET';

```

```

% Create Wind
app.Wind = uitab(app.TabGroup);
app.Wind.Title = 'Wind';
app.Wind.BackgroundColor = [0.9686 0.9686 1];
app.Wind.ForegroundColor = [0 0.4471 0.7412];

% Create WIND
app.WIND = uilabel(app.Wind);
app.WIND.FontName = 'Times New Roman';
app.WIND.FontSize = 16;
app.WIND.FontWeight = 'bold';
app.WIND.FontAngle = 'italic';
app.WIND.FontColor = [0 0.4471 0.7412];
app.WIND.Position = [250 355 124 22];
app.WIND.Text = 'WIND TURBINE';

% Create PWnomkWEEditFieldLabel
app.PWnomkWEEditFieldLabel = uilabel(app.Wind);
app.PWnomkWEEditFieldLabel.HorizontalAlignment = 'right';
app.PWnomkWEEditFieldLabel.FontName = 'Times New Roman';
app.PWnomkWEEditFieldLabel.FontSize = 13;
app.PWnomkWEEditFieldLabel.FontWeight = 'bold';
app.PWnomkWEEditFieldLabel.Position = [14 325 83 22];
app.PWnomkWEEditFieldLabel.Text = 'PWnom (kW)';

% Create PWnomI
app.PWnomI = uieditfield(app.Wind, 'numeric');
app.PWnomI.ValueChangedFcn = createCallbackFcn(app, ↵
@PWnomIValueChanged, true);
app.PWnomI.FontName = 'Times New Roman';
app.PWnomI.FontSize = 13;
app.PWnomI.FontWeight = 'bold';
app.PWnomI.Position = [131 325 86 22];

% Create VnommsEditFieldLabel
app.VnommsEditFieldLabel = uilabel(app.Wind);
app.VnommsEditFieldLabel.HorizontalAlignment = 'right';
app.VnommsEditFieldLabel.FontName = 'Times New Roman';
app.VnommsEditFieldLabel.FontSize = 13;
app.VnommsEditFieldLabel.FontWeight = 'bold';
app.VnommsEditFieldLabel.Position = [14 296 71 22];
app.VnommsEditFieldLabel.Text = 'Vnom (m/s)';

% Create VnomI
app.VnomI = uieditfield(app.Wind, 'numeric');
app.VnomI.ValueChangedFcn = createCallbackFcn(app, @VnomIValueChanged, ↵
true);
app.VnomI.FontName = 'Times New Roman';
app.VnomI.FontSize = 13;
app.VnomI.FontWeight = 'bold';
app.VnomI.Position = [131 296 86 22];

% Create VcimsEditFieldLabel
app.VcimsEditFieldLabel = uilabel(app.Wind);
app.VcimsEditFieldLabel.HorizontalAlignment = 'right';
app.VcimsEditFieldLabel.FontName = 'Times New Roman';
app.VcimsEditFieldLabel.FontSize = 13;

```

```

app.VcimsEditFieldLabel.FontWeight = 'bold';
app.VcimsEditFieldLabel.Position = [16 267 55 22];
app.VcimsEditFieldLabel.Text = 'Vci (m/s)';

% Create VciI
app.VciI = uieditfield(app.Wind, 'numeric');
app.VciI.ValueChangedFcn = createCallbackFcn(app, @VciIValueChanged, ↵
true);
app.VciI.FontName = 'Times New Roman';
app.VciI.FontSize = 13;
app.VciI.FontWeight = 'bold';
app.VciI.Position = [131 267 86 22];

% Create VcomsEditFieldLabel
app.VcomsEditFieldLabel = uilabel(app.Wind);
app.VcomsEditFieldLabel.HorizontalAlignment = 'right';
app.VcomsEditFieldLabel.FontName = 'Times New Roman';
app.VcomsEditFieldLabel.FontSize = 13;
app.VcomsEditFieldLabel.FontWeight = 'bold';
app.VcomsEditFieldLabel.Position = [16 238 59 22];
app.VcomsEditFieldLabel.Text = 'Vco (m/s)';

% Create VcoI
app.VcoI = uieditfield(app.Wind, 'numeric');
app.VcoI.ValueChangedFcn = createCallbackFcn(app, @VcoIValueChanged, ↵
true);
app.VcoI.FontName = 'Times New Roman';
app.VcoI.FontSize = 13;
app.VcoI.FontWeight = 'bold';
app.VcoI.Position = [131 238 86 22];

% Create CpeqkWms3EditFieldLabel
app.CpeqkWms3EditFieldLabel = uilabel(app.Wind);
app.CpeqkWms3EditFieldLabel.HorizontalAlignment = 'right';
app.CpeqkWms3EditFieldLabel.FontName = 'Times New Roman';
app.CpeqkWms3EditFieldLabel.FontSize = 13;
app.CpeqkWms3EditFieldLabel.FontWeight = 'bold';
app.CpeqkWms3EditFieldLabel.Position = [15 208 106 22];
app.CpeqkWms3EditFieldLabel.Text = 'Cpeq (kW/(m/s)^3)';

% Create CpeqI
app.CpeqI = uieditfield(app.Wind, 'numeric');
app.CpeqI.Editable = 'off';
app.CpeqI.FontName = 'Times New Roman';
app.CpeqI.FontSize = 13;
app.CpeqI.FontWeight = 'bold';
app.CpeqI.Position = [131 208 86 22];

% Create PWnomLabel
app.PWnomLabel = uilabel(app.Wind);
app.PWnomLabel.FontName = 'Times New Roman';
app.PWnomLabel.FontSize = 13;
app.PWnomLabel.FontColor = [0.4667 0.6745 0.1882];
app.PWnomLabel.Visible = 'off';
app.PWnomLabel.Position = [233 325 123 22];
app.PWnomLabel.Text = 'Rated electrical power';

% Create VnomLabel

```

```

app.VnomLabel = uilabel(app.Wind);
app.VnomLabel.FontName = 'Times New Roman';
app.VnomLabel.FontSize = 13;
app.VnomLabel.FontColor = [0.4667 0.6745 0.1882];
app.VnomLabel.Visible = 'off';
app.VnomLabel.Position = [233 296 98 22];
app.VnomLabel.Text = 'Rated wind speed';

% Create VciLabel
app.VciLabel = uilabel(app.Wind);
app.VciLabel.FontName = 'Times New Roman';
app.VciLabel.FontSize = 13;
app.VciLabel.FontColor = [0.4667 0.6745 0.1882];
app.VciLabel.Visible = 'off';
app.VciLabel.Position = [233 267 101 22];
app.VciLabel.Text = 'Cut-in wind speed';

% Create VcoLabel
app.VcoLabel = uilabel(app.Wind);
app.VcoLabel.FontName = 'Times New Roman';
app.VcoLabel.FontSize = 13;
app.VcoLabel.FontColor = [0.4667 0.6745 0.1882];
app.VcoLabel.Visible = 'off';
app.VcoLabel.Position = [233 238 107 22];
app.VcoLabel.Text = 'Cut-out wind speed';

% Create CpeqLabel
app.CpeqLabel = uilabel(app.Wind);
app.CpeqLabel.FontName = 'Times New Roman';
app.CpeqLabel.FontSize = 13;
app.CpeqLabel.FontColor = [0.4667 0.6745 0.1882];
app.CpeqLabel.Visible = 'off';
app.CpeqLabel.Position = [233 208 268 22];
app.CpeqLabel.Text = 'Equivalent power coefficient (Cpeq=0,5·r·A·Cp)r';

% Create HELPSwitchLabel_7
app.HELPswitchLabel_7 = uilabel(app.Wind);
app.HELPswitchLabel_7.HorizontalAlignment = 'center';
app.HELPswitchLabel_7.FontName = 'Times New Roman';
app.HELPswitchLabel_7.FontSize = 13;
app.HELPswitchLabel_7.FontWeight = 'bold';
app.HELPswitchLabel_7.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_7.Position = [526.5 229 41 22];
app.HELPswitchLabel_7.Text = 'HELP';

% Create HELPWind
app.HELPWind = uiswitch(app.Wind, 'toggle');
app.HELPWind.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPWindValueChanged, true);
app.HELPWind.FontName = 'Times New Roman';
app.HELPWind.FontSize = 13;
app.HELPWind.FontWeight = 'bold';
app.HELPWind.FontColor = [0.4667 0.6745 0.1882];
app.HELPWind.Position = [536 287 20 45];

% Create PowerCurve
app.PowerCurve = uitable(app.Wind);
app.PowerCurve.ColumnName = {'Wind speed (m/s)'; 'Power (kW)'};

```

```

app.PowerCurve.ColumnWidth = {90, 90};
app.PowerCurve.RowName = {};
app.PowerCurve.FontName = 'Times New Roman';
app.PowerCurve.FontSize = 13;
app.PowerCurve.Position = [32 13 211 179];

% Create PowerCurveGraph
app.PowerCurveGraph = uiaxes(app.Wind);
title(app.PowerCurveGraph, 'Power curve')
xlabel(app.PowerCurveGraph, 'Wind speed (m/s)')
ylabel(app.PowerCurveGraph, 'Power (kW)')
app.PowerCurveGraph.FontName = 'Times New Roman';
app.PowerCurveGraph.XTick = [0 5 10 15 20];
app.PowerCurveGraph.XTickLabel = {'0'; '5'; '10'; '15'; '20'};
app.PowerCurveGraph.YTick = [0 5 10 15 20 25];
app.PowerCurveGraph.YTickLabel = {'0'; '5'; '10'; '15'; '20'; '25'};
app.PowerCurveGraph.BackgroundColor = [0.8902 0.898 1];
app.PowerCurveGraph.Position = [293 22 275 161];

% Create Pumping1
app.Pumping1 = uitab(app.TabGroup);
app.Pumping1.Title = 'Pumping (1)';
app.Pumping1.BackgroundColor = [0.9686 0.9686 1];
app.Pumping1.ForegroundColor = [0 0.451 0.7412];

% Create PUMPLabel
app.PUMPLabel = uilabel(app.Pumping1);
app.PUMPLabel.FontName = 'Times New Roman';
app.PUMPLabel.FontSize = 13;
app.PUMPLabel.FontWeight = 'bold';
app.PUMPLabel.FontColor = [0 0.4471 0.7412];
app.PUMPLabel.Position = [405 117 43 22];
app.PUMPLabel.Text = 'PUMP';

% Create Qratedm3hEditFieldLabel
app.Qratedm3hEditFieldLabel = uilabel(app.Pumping1);
app.Qratedm3hEditFieldLabel.HorizontalAlignment = 'right';
app.Qratedm3hEditFieldLabel.FontName = 'Times New Roman';
app.Qratedm3hEditFieldLabel.FontSize = 13;
app.Qratedm3hEditFieldLabel.FontWeight = 'bold';
app.Qratedm3hEditFieldLabel.Position = [302 86 85 22];
app.Qratedm3hEditFieldLabel.Text = 'Qrated (m3/h)';

% Create QratedI
app.QratedI = uieditfield(app.Pumping1, 'numeric');
app.QratedI.ValueChangedFcn = createCallbackFcn(app, ↵
@QratedIValueChanged, true);
app.QratedI.FontName = 'Times New Roman';
app.QratedI.FontSize = 13;
app.QratedI.FontWeight = 'bold';
app.QratedI.Position = [447 86 69 22];

% Create HratedmEditFieldLabel
app.HratedmEditFieldLabel = uilabel(app.Pumping1);
app.HratedmEditFieldLabel.HorizontalAlignment = 'right';
app.HratedmEditFieldLabel.FontName = 'Times New Roman';
app.HratedmEditFieldLabel.FontSize = 13;
app.HratedmEditFieldLabel.FontWeight = 'bold';

```

```

app.HratedmEditFieldLabel.Position = [302 61 68 22];
app.HratedmEditFieldLabel.Text = 'Hrated (m)';

% Create HratedI
app.HratedI = uieditfield(app.Pumping1, 'numeric');
app.HratedI.ValueChangedFcn = createCallbackFcn(app, ↵
@HratedIValueChanged, true);
app.HratedI.FontName = 'Times New Roman';
app.HratedI.FontSize = 13;
app.HratedI.FontWeight = 'bold';
app.HratedI.Position = [447 61 69 22];

% Create LiquidDensitykgm3EditFieldLabel
app.LiquidDensitykgm3EditFieldLabel = uilabel(app.Pumping1);
app.LiquidDensitykgm3EditFieldLabel.HorizontalAlignment = 'right';
app.LiquidDensitykgm3EditFieldLabel.FontName = 'Times New Roman';
app.LiquidDensitykgm3EditFieldLabel.FontSize = 13;
app.LiquidDensitykgm3EditFieldLabel.FontWeight = 'bold';
app.LiquidDensitykgm3EditFieldLabel.Position = [302 36 135 22];
app.LiquidDensitykgm3EditFieldLabel.Text = 'Liquid Density (kg/m3)';

% Create LiquidDensityI
app.LiquidDensityI = uieditfield(app.Pumping1, 'numeric');
app.LiquidDensityI.ValueChangedFcn = createCallbackFcn(app, ↵
@LiquidDensityIValueChanged, true);
app.LiquidDensityI.FontName = 'Times New Roman';
app.LiquidDensityI.FontSize = 13;
app.LiquidDensityI.FontWeight = 'bold';
app.LiquidDensityI.Position = [447 36 69 22];

% Create QratedLabel
app.QratedLabel = uilabel(app.Pumping1);
app.QratedLabel.FontName = 'Times New Roman';
app.QratedLabel.FontSize = 13;
app.QratedLabel.FontColor = [0.4706 0.6706 0.1882];
app.QratedLabel.Visible = 'off';
app.QratedLabel.Position = [530 86 63 22];
app.QratedLabel.Text = 'Rated flow';

% Create HratedLabel
app.HratedLabel = uilabel(app.Pumping1);
app.HratedLabel.FontName = 'Times New Roman';
app.HratedLabel.FontSize = 13;
app.HratedLabel.FontColor = [0.4706 0.6706 0.1882];
app.HratedLabel.Visible = 'off';
app.HratedLabel.Position = [530 61 63 22];
app.HratedLabel.Text = 'Rated head';

% Create densityLabel
app.densityLabel = uilabel(app.Pumping1);
app.densityLabel.FontName = 'Times New Roman';
app.densityLabel.FontSize = 13;
app.densityLabel.FontColor = [0.4706 0.6706 0.1882];
app.densityLabel.Visible = 'off';
app.densityLabel.Position = [530 36 81 22];
app.densityLabel.Text = 'Liquid density';

% Create WELLBOREHOLELabel

```

```

app.WELBOREHOLELabel = uilabel(app.Pumping1);
app.WELBOREHOLELabel.FontName = 'Times New Roman';
app.WELBOREHOLELabel.FontSize = 13;
app.WELBOREHOLELabel.FontWeight = 'bold';
app.WELBOREHOLELabel.FontColor = [0 0.4471 0.7412];
app.WELBOREHOLELabel.Position = [50 354 129 22];
app.WELBOREHOLELabel.Text = 'WELL / BOREHOLE';

% Create HsmLabel
app.HsmLabel = uilabel(app.Pumping1);
app.HsmLabel.HorizontalAlignment = 'right';
app.HsmLabel.FontName = 'Times New Roman';
app.HsmLabel.FontSize = 13;
app.HsmLabel.FontWeight = 'bold';
app.HsmLabel.Position = [29 327 43 22];
app.HsmLabel.Text = 'Hs (m)';

% Create HsI
app.HsI = uieditfield(app.Pumping1, 'numeric');
app.HsI.ValueChangedFcn = createCallbackFcn(app, @HsIValueChanged, ↵
true);
app.HsI.FontName = 'Times New Roman';
app.HsI.FontSize = 13;
app.HsI.FontWeight = 'bold';
app.HsI.Position = [134 327 69 22];

% Create Qtestm3hLabel
app.Qtestm3hLabel = uilabel(app.Pumping1);
app.Qtestm3hLabel.HorizontalAlignment = 'right';
app.Qtestm3hLabel.FontName = 'Times New Roman';
app.Qtestm3hLabel.FontSize = 13;
app.Qtestm3hLabel.FontWeight = 'bold';
app.Qtestm3hLabel.Position = [29 301 75 22];
app.Qtestm3hLabel.Text = 'Qtest (m3/h)',

% Create QtestI
app.QtestI = uieditfield(app.Pumping1, 'numeric');
app.QtestI.ValueChangedFcn = createCallbackFcn(app, ↵
@QtestIValueChanged, true);
app.QtestI.FontName = 'Times New Roman';
app.QtestI.FontSize = 13;
app.QtestI.FontWeight = 'bold';
app.QtestI.Position = [135 301 69 22];

% Create HdrmLabel
app.HdrmLabel = uilabel(app.Pumping1);
app.HdrmLabel.HorizontalAlignment = 'right';
app.HdrmLabel.FontName = 'Times New Roman';
app.HdrmLabel.FontSize = 13;
app.HdrmLabel.FontWeight = 'bold';
app.HdrmLabel.Position = [29 276 51 22];
app.HdrmLabel.Text = 'Hdr (m)';

% Create HdrI
app.HdrI = uieditfield(app.Pumping1, 'numeric');
app.HdrI.ValueChangedFcn = createCallbackFcn(app, @HdrIValueChanged, ↵
true);
app.HdrI.FontName = 'Times New Roman';

```

```

app.HdrI.FontSize = 13;
app.HdrI.FontWeight = 'bold';
app.HdrI.Position = [135 276 69 22];

% Create kw1mm3hLabel
app.kw1mm3hLabel = uilabel(app.Pumping1);
app.kw1mm3hLabel.HorizontalAlignment = 'right';
app.kw1mm3hLabel.FontName = 'Times New Roman';
app.kw1mm3hLabel.FontSize = 13;
app.kw1mm3hLabel.FontWeight = 'bold';
app.kw1mm3hLabel.Position = [29 251 91 22];
app.kw1mm3hLabel.Text = 'kw1 (m/(m3/h))';

% Create kw1I
app.kw1I = uieditfield(app.Pumping1, 'numeric');
app.kw1I.ValueChangedFcn = createCallbackFcn(app, @kw1IValueChanged, ↵
true);
app.kw1I.FontName = 'Times New Roman';
app.kw1I.FontSize = 13;
app.kw1I.FontWeight = 'bold';
app.kw1I.Position = [135 251 69 22];

% Create kw2mm3h2Label
app.kw2mm3h2Label = uilabel(app.Pumping1);
app.kw2mm3h2Label.HorizontalAlignment = 'right';
app.kw2mm3h2Label.FontName = 'Times New Roman';
app.kw2mm3h2Label.FontSize = 13;
app.kw2mm3h2Label.FontWeight = 'bold';
app.kw2mm3h2Label.Position = [29 210 98 22];
app.kw2mm3h2Label.Text = 'kw2 (m/(m3/h)2)';

% Create kw2I
app.kw2I = uieditfield(app.Pumping1, 'numeric');
app.kw2I.ValueChangedFcn = createCallbackFcn(app, @kw2IValueChanged, ↵
true);
app.kw2I.FontName = 'Times New Roman';
app.kw2I.FontSize = 13;
app.kw2I.FontWeight = 'bold';
app.kw2I.Position = [135 210 69 22];

% Create HsLabel
app.HsLabel = uilabel(app.Pumping1);
app.HsLabel.FontName = 'Times New Roman';
app.HsLabel.FontSize = 13;
app.HsLabel.FontColor = [0.4706 0.6706 0.1882];
app.HsLabel.Visible = 'off';
app.HsLabel.Position = [215 327 63 22];
app.HsLabel.Text = 'Static head';

% Create QTestLabel
app.QtestLabel = uilabel(app.Pumping1);
app.QtestLabel.FontName = 'Times New Roman';
app.QtestLabel.FontSize = 13;
app.QtestLabel.FontColor = [0.4706 0.6706 0.1882];
app.QtestLabel.Visible = 'off';
app.QtestLabel.Position = [215 301 100 22];
app.QtestLabel.Text = 'Constant test flow';

```

```

% Create HdrLabel
app.HdrLabel = uilabel(app.Pumping1);
app.HdrLabel.FontName = 'Times New Roman';
app.HdrLabel.FontSize = 13;
app.HdrLabel.FontColor = [0.4706 0.6706 0.1882];
app.HdrLabel.Visible = 'off';
app.HdrLabel.Position = [215 276 171 22];
app.HdrLabel.Text = 'Drawdown at constant test flow';

% Create kw1Label1
app.kw1Label1 = uilabel(app.Pumping1);
app.kw1Label1.FontName = 'Times New Roman';
app.kw1Label1.FontSize = 13;
app.kw1Label1.FontColor = [0.4706 0.6706 0.1882];
app.kw1Label1.Visible = 'off';
app.kw1Label1.Position = [215 251 345 22];
app.kw1Label1.Text = 'Aquifer loss (this parameter is calculated as ↵
Hdr/Qtest, but other)';

% Create kw1Label2
app.kw1Label2 = uilabel(app.Pumping1);
app.kw1Label2.FontName = 'Times New Roman';
app.kw1Label2.FontSize = 13;
app.kw1Label2.FontColor = [0.4706 0.6706 0.1882];
app.kw1Label2.Visible = 'off';
app.kw1Label2.Position = [215 235 304 22];
app.kw1Label2.Text = 'value can be chosen if a stepdrawdown test is ↵
available).';

% Create kw2Label
app.kw2Label = uilabel(app.Pumping1);
app.kw2Label.FontName = 'Times New Roman';
app.kw2Label.FontSize = 13;
app.kw2Label.FontColor = [0.4706 0.6706 0.1882];
app.kw2Label.Visible = 'off';
app.kw2Label.Position = [215 210 320 22];
app.kw2Label.Text = 'Well loss (set to zero if only constant test flow ↵
is available).';

% Create WATERTANKLabel
app.WATERTANKLabel = uilabel(app.Pumping1);
app.WATERTANKLabel.FontName = 'Times New Roman';
app.WATERTANKLabel.FontSize = 13;
app.WATERTANKLabel.FontWeight = 'bold';
app.WATERTANKLabel.FontColor = [0 0.4471 0.7412];
app.WATERTANKLabel.Position = [92 154 92 22];
app.WATERTANKLabel.Text = 'WATER TANK';

% Create WTCm3Label
app.WTCm3Label = uilabel(app.Pumping1);
app.WTCm3Label.HorizontalAlignment = 'right';
app.WTCm3Label.FontName = 'Times New Roman';
app.WTCm3Label.FontSize = 13;
app.WTCm3Label.FontWeight = 'bold';
app.WTCm3Label.Position = [54 85 66 22];
app.WTCm3Label.Text = 'WTC (m3)';

% Create WTCI

```

```

app.WTCI = uieditfield(app.Pumping1, 'numeric');
app.WTCI.ValueChangedFcn = createCallbackFcn(app, @WTCIValueChanged, ↵
true);
app.WTCI.FontName = 'Times New Roman';
app.WTCI.FontSize = 13;
app.WTCI.FontWeight = 'bold';
app.WTCI.Position = [153 85 69 22];

% Create HrmLabel
app.HrmLabel = uilabel(app.Pumping1);
app.HrmLabel.HorizontalAlignment = 'right';
app.HrmLabel.FontName = 'Times New Roman';
app.HrmLabel.FontSize = 13;
app.HrmLabel.FontWeight = 'bold';
app.HrmLabel.Position = [54 28 44 22];
app.HrmLabel.Text = 'Hr (m)';

% Create HrI
app.HrI = uieditfield(app.Pumping1, 'numeric');
app.HrI.ValueChangedFcn = createCallbackFcn(app, @HrIValueChanged, ↵
true);
app.HrI.FontName = 'Times New Roman';
app.HrI.FontSize = 13;
app.HrI.FontWeight = 'bold';
app.HrI.Position = [153 28 69 22];

% Create WTCLabel
app.WTCLabel = uilabel(app.Pumping1);
app.WTCLabel.FontName = 'Times New Roman';
app.WTCLabel.FontSize = 13;
app.WTCLabel.FontColor = [0.4706 0.6706 0.1882];
app.WTCLabel.Visible = 'off';
app.WTCLabel.Position = [15 125 246 22];
app.WTCLabel.Text = 'Water tank capacity (in this version the water ';

% Create WTCLabel2
app.WTCLabel2 = uilabel(app.Pumping1);
app.WTCLabel2.FontName = 'Times New Roman';
app.WTCLabel2.FontSize = 13;
app.WTCLabel2.FontColor = [0.4706 0.6706 0.1882];
app.WTCLabel2.Visible = 'off';
app.WTCLabel2.Position = [52 109 172 22];
app.WTCLabel2.Text = 'storage is considered unlimited)';

% Create HrLabel
app.HrLabel = uilabel(app.Pumping1);
app.HrLabel.FontName = 'Times New Roman';
app.HrLabel.FontSize = 13;
app.HrLabel.FontColor = [0.4706 0.6706 0.1882];
app.HrLabel.Visible = 'off';
app.HrLabel.Position = [95 52 87 22];
app.HrLabel.Text = 'Discharge level';

% Create PUMPING1
app.PUMPING1 = uilabel(app.Pumping1);
app.PUMPING1.FontName = 'Times New Roman';
app.PUMPING1.FontSize = 16;
app.PUMPING1.FontWeight = 'bold';

```

```

app.PUMPING1.FontAngle = 'italic';
app.PUMPING1.FontColor = [0 0.4471 0.7412];
app.PUMPING1.Position = [420 348 173 22];
app.PUMPING1.Text = 'WATER PUMPING (1/4)';

% Create HELPSwitchLabel_13
app.HELPSSwitchLabel_13 = uilabel(app.Pumping1);
app.HELPSSwitchLabel_13.HorizontalAlignment = 'center';
app.HELPSSwitchLabel_13.FontName = 'Times New Roman';
app.HELPSSwitchLabel_13.FontSize = 13;
app.HELPSSwitchLabel_13.FontWeight = 'bold';
app.HELPSSwitchLabel_13.FontColor = [0.4667 0.6745 0.1882];
app.HELPSSwitchLabel_13.Position = [532 164 41 22];
app.HELPSSwitchLabel_13.Text = 'HELP';

% Create HELPPump1
app.HELPPump1 = uiswitch(app.Pumping1, 'toggle');
app.HELPPump1.Orientation = 'horizontal';
app.HELPPump1.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPPump1ValueChanged, true);
app.HELPPump1.FontName = 'Times New Roman';
app.HELPPump1.FontSize = 13;
app.HELPPump1.FontWeight = 'bold';
app.HELPPump1.FontColor = [0.4667 0.6745 0.1882];
app.HELPPump1.Position = [445 165 45 20];

% Create Pumping2
app.Pumping2 = uitab(app.TabGroup);
app.Pumping2.Title = 'Pumping (2)';
app.Pumping2.BackgroundColor = [0.9686 0.9686 1];
app.Pumping2.ForegroundColor = [0 0.4471 0.7412];

% Create PUMPING2
app.PUMPING2 = uilabel(app.Pumping2);
app.PUMPING2.FontName = 'Times New Roman';
app.PUMPING2.FontSize = 16;
app.PUMPING2.FontWeight = 'bold';
app.PUMPING2.FontAngle = 'italic';
app.PUMPING2.FontColor = [0 0.4471 0.7412];
app.PUMPING2.Position = [248 365 173 22];
app.PUMPING2.Text = 'WATER PUMPING (2/4)';

% Create ks0mLabel
app.ks0mLabel = uilabel(app.Pumping2);
app.ks0mLabel.HorizontalAlignment = 'right';
app.ks0mLabel.FontName = 'Times New Roman';
app.ks0mLabel.FontSize = 13;
app.ks0mLabel.FontWeight = 'bold';
app.ks0mLabel.Position = [277 313 47 22];
app.ks0mLabel.Text = 'ks0 (m)';

% Create ks0I
app.ks0I = uieditfield(app.Pumping2, 'numeric');
app.ks0I.Editable = 'off';
app.ks0I.FontName = 'Times New Roman';
app.ks0I.FontSize = 13;
app.ks0I.FontWeight = 'bold';
app.ks0I.Position = [383 313 55 22];

```

```

% Create ks1mm3hLabel
app.ks1mm3hLabel = uilabel(app.Pumping2);
app.ks1mm3hLabel.HorizontalAlignment = 'right';
app.ks1mm3hLabel.FontName = 'Times New Roman';
app.ks1mm3hLabel.FontSize = 13;
app.ks1mm3hLabel.FontWeight = 'bold';
app.ks1mm3hLabel.Position = [277 288 87 22];
app.ks1mm3hLabel.Text = 'ks1 (m/(m3/h))';

% Create ks1I
app.ks1I = uieditfield(app.Pumping2, 'numeric');
app.ks1I.ValueChangedFcn = createCallbackFcn(app, @ks1IValueChanged, ↵
true);
app.ks1I.FontName = 'Times New Roman';
app.ks1I.FontSize = 13;
app.ks1I.FontWeight = 'bold';
app.ks1I.Position = [383 288 55 22];

% Create HfmLabel
app.HfmLabel = uilabel(app.Pumping2);
app.HfmLabel.HorizontalAlignment = 'right';
app.HfmLabel.FontName = 'Times New Roman';
app.HfmLabel.FontSize = 13;
app.HfmLabel.FontWeight = 'bold';
app.HfmLabel.Position = [66 292 43 22];
app.HfmLabel.Text = 'Hf (m)';

% Create HfI
app.HfI = uieditfield(app.Pumping2, 'numeric');
app.HfI.ValueChangedFcn = createCallbackFcn(app, @HfIValueChanged, ↵
true);
app.HfI.FontName = 'Times New Roman';
app.HfI.FontSize = 13;
app.HfI.FontWeight = 'bold';
app.HfI.Position = [124 292 55 22];

% Create ks0Label
app.ks0Label = uilabel(app.Pumping2);
app.ks0Label.FontName = 'Times New Roman';
app.ks0Label.FontSize = 13;
app.ks0Label.FontColor = [0.4667 0.6745 0.1882];
app.ks0Label.Visible = 'off';
app.ks0Label.Position = [444 313 138 22];
app.ks0Label.Text = 'Total static head (Hs+Hr)';

% Create ks1Label
app.ks1Label = uilabel(app.Pumping2);
app.ks1Label.FontName = 'Times New Roman';
app.ks1Label.FontSize = 13;
app.ks1Label.FontColor = [0.4667 0.6745 0.1882];
app.ks1Label.Visible = 'off';
app.ks1Label.Position = [444 288 179 22];
app.ks1Label.Text = 'Specific drawdown (aquifer loss)';

% Create HfLabel
app.HfLabel = uilabel(app.Pumping2);
app.HfLabel.FontName = 'Times New Roman';

```

```

app.HfLabel.FontSize = 13;
app.HfLabel.FontColor = [0.4667 0.6745 0.1882];
app.HfLabel.Visible = 'off';
app.HfLabel.Position = [28 318 188 22];
app.HfLabel.Text = 'Friction losses at rated flow Qrated';

% Create SYSTEMCURVELabel
app.SYSTEMCURVELabel = uilabel(app.Pumping2);
app.SYSTEMCURVELabel.FontName = 'Times New Roman';
app.SYSTEMCURVELabel.FontSize = 13;
app.SYSTEMCURVELabel.FontWeight = 'bold';
app.SYSTEMCURVELabel.FontColor = [0 0.4471 0.7412];
app.SYSTEMCURVELabel.Position = [28 342 108 22];
app.SYSTEMCURVELabel.Text = 'SYSTEM CURVE';

% Create systemCurveTable
app.systemCurveTable = uitable(app.Pumping2);
app.systemCurveTable.ColumnName = {'Flow (Q) [m3/h]'; 'Head (H) [m]'};
app.systemCurveTable.ColumnWidth = {85, 75};
app.systemCurveTable.RowName = {};
app.systemCurveTable.FontName = 'Times New Roman';
app.systemCurveTable.FontSize = 13;
app.systemCurveTable.Position = [16 14 178 226];

% Create systemCurveGraph
app.systemCurveGraph = uiaxes(app.Pumping2);
title(app.systemCurveGraph, 'System curve')
xlabel(app.systemCurveGraph, 'Q [m3/h]')
ylabel(app.systemCurveGraph, 'H [m]')
app.systemCurveGraph.FontName = 'Times New Roman';
app.systemCurveGraph.XTick = [0 2 4 6 8];
app.systemCurveGraph.XTickLabel = {'0'; '2'; '4'; '6'; '8'};
app.systemCurveGraph.BackgroundColor = [0.8902 0.898 1];
app.systemCurveGraph.Position = [231 48 300 185];

% Create HELPSwitchLabel_9
app.HELPswitchLabel_9 = uilabel(app.Pumping2);
app.HELPswitchLabel_9.HorizontalAlignment = 'center';
app.HELPswitchLabel_9.FontName = 'Times New Roman';
app.HELPswitchLabel_9.FontSize = 13;
app.HELPswitchLabel_9.FontWeight = 'bold';
app.HELPswitchLabel_9.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_9.Position = [576.5 10 41 22];
app.HELPswitchLabel_9.Text = 'HELP';

% Create HELPPump2
app.HELPPump2 = uiswitch(app.Pumping2, 'toggle');
app.HELPPump2.Orientation = 'horizontal';
app.HELPPump2.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPPump2ValueChanged, true);
app.HELPPump2.FontName = 'Times New Roman';
app.HELPPump2.FontSize = 13;
app.HELPPump2.FontWeight = 'bold';
app.HELPPump2.FontColor = [0.4667 0.6745 0.1882];
app.HELPPump2.Position = [490 11 45 20];

% Create ks2mm3h2Label
app.ks2mm3h2Label = uilabel(app.Pumping2);

```

```

app.ks2mm3h2Label.HorizontalAlignment = 'right';
app.ks2mm3h2Label.FontName = 'Times New Roman';
app.ks2mm3h2Label.FontSize = 13;
app.ks2mm3h2Label.FontWeight = 'bold';
app.ks2mm3h2Label.Position = [277 263 94 22];
app.ks2mm3h2Label.Text = 'ks2 (m/(m3/h)2)';

% Create ks2I
app.ks2I = uieditfield(app.Pumping2, 'numeric');
app.ks2I.Editable = 'off';
app.ks2I.FontName = 'Times New Roman';
app.ks2I.FontSize = 13;
app.ks2I.FontWeight = 'bold';
app.ks2I.Position = [383 263 55 22];

% Create ks2Label
app.ks2Label = uilabel(app.Pumping2);
app.ks2Label.FontName = 'Times New Roman';
app.ks2Label.FontSize = 13;
app.ks2Label.FontColor = [0.4667 0.6745 0.1882];
app.ks2Label.Visible = 'off';
app.ks2Label.Position = [444 263 131 22];
app.ks2Label.Text = 'Friction plus well losses';

% Create Pumping3
app.Pumping3 = uitab(app.TabGroup);
app.Pumping3.Title = 'Pumping (3)';
app.Pumping3.BackgroundColor = [0.9686 0.9686 1];
app.Pumping3.ForegroundColor = [0 0.4471 0.7412];

% Create pumpCurveTable
app.pumpCurveTable = uitable(app.Pumping3);
app.pumpCurveTable.ColumnName = {'Q [m3/h]'; 'H [m]'};
app.pumpCurveTable.ColumnWidth = {75, 75};
app.pumpCurveTable.RowName = {};
app.pumpCurveTable.ColumnEditable = true;
app.pumpCurveTable.CellEditCallback = createCallbackFcn(app, ↵
@pumpCurveTableCellEdit, true);
app.pumpCurveTable.FontName = 'Times New Roman';
app.pumpCurveTable.FontSize = 13;
app.pumpCurveTable.Position = [20 183 169 186];

% Create pumpCurveGraph
app.pumpCurveGraph = uiaxes(app.Pumping3);
title(app.pumpCurveGraph, 'Pump curve')
xlabel(app.pumpCurveGraph, 'Q [m3/h]')
ylabel(app.pumpCurveGraph, 'H [m]')
app.pumpCurveGraph.FontName = 'Times New Roman';
app.pumpCurveGraph.XTick = [0 2 4 6 8];
app.pumpCurveGraph.XTickLabel = {'0'; '2'; '4'; '6'; '8'};
app.pumpCurveGraph.YTick = [0 5 10 15 20 25 30];
app.pumpCurveGraph.YTickLabel = {'0'; '5'; '10'; '15'; '20'; '25'; ↵
'30'};
app.pumpCurveGraph.BackgroundColor = [0.8902 0.902 1];
app.pumpCurveGraph.Position = [12 12 263 158];

% Create powerCurvesTable
app.powerCurvesTable = uitable(app.Pumping3);

```

```

app.powerCurvesTable.ColumnName = {'Q [m3/h]'; 'P1 [kW]'; 'P2 [kW]'};
app.powerCurvesTable.ColumnWidth = {70, 75, 75};
app.powerCurvesTable.RowName = {};
app.powerCurvesTable.ColumnEditable = true;
app.powerCurvesTable.CellEditCallback = createCallbackFcn(app, ↵
@powerCurvesTableCellEdit, true);
app.powerCurvesTable.FontName = 'Times New Roman';
app.powerCurvesTable.FontSize = 13;
app.powerCurvesTable.Position = [378 192 235 185];

% Create powerCurvesGraph
app.powerCurvesGraph = uiaxes(app.Pumping3);
title(app.powerCurvesGraph, {'P1 (magenta) vs P2 (blue)'; ''})
xlabel(app.powerCurvesGraph, 'Q [m3/h]')
ylabel(app.powerCurvesGraph, 'Power Curves [kW]')
app.powerCurvesGraph.FontName = 'Times New Roman';
app.powerCurvesGraph.XTick = [0 2 4 6 8];
app.powerCurvesGraph.XTickLabel = {'0'; '2'; '4'; '6'; '8'};
app.powerCurvesGraph.YTick = [0 0.1 0.2 0.3 0.4 0.5 0.6];
app.powerCurvesGraph.YTickLabel = {'0'; '0.1'; '0.2'; '0.3'; '0.4'; ↵
'0.5'; '0.6'};
app.powerCurvesGraph.NextPlot = 'add';
app.powerCurvesGraph.BackgroundColor = [0.8902 0.898 1];
app.powerCurvesGraph.Position = [291 14 322 169];

% Create P1Label1
app.P1Label1 = uilabel(app.Pumping3);
app.P1Label1.FontName = 'Times New Roman';
app.P1Label1.FontSize = 13;
app.P1Label1.FontColor = [0.4667 0.6745 0.1882];
app.P1Label1.Visible = 'off';
app.P1Label1.Position = [199 274 170 22];
app.P1Label1.Text = 'P1: nominal curve of the power';

% Create P1Label2
app.P1Label2 = uilabel(app.Pumping3);
app.P1Label2.FontName = 'Times New Roman';
app.P1Label2.FontSize = 13;
app.P1Label2.FontColor = [0.4667 0.6745 0.1882];
app.P1Label2.Visible = 'off';
app.P1Label2.Position = [196 257 127 22];
app.P1Label2.Text = ' at the motor input, kw';

% Create P2Label1
app.P2Label1 = uilabel(app.Pumping3);
app.P2Label1.FontName = 'Times New Roman';
app.P2Label1.FontSize = 13;
app.P2Label1.FontColor = [0.4667 0.6745 0.1882];
app.P2Label1.Visible = 'off';
app.P2Label1.Position = [199 235 143 22];
app.P2Label1.Text = 'P2: nominal curve of shaft';

% Create P2Label2
app.P2Label2 = uilabel(app.Pumping3);
app.P2Label2.FontName = 'Times New Roman';
app.P2Label2.FontSize = 13;
app.P2Label2.FontColor = [0.4667 0.6745 0.1882];
app.P2Label2.Visible = 'off';

```

```

app.P2Label2.Position = [199 218 63 22];
app.P2Label2.Text = 'power, kW';

% Create PUMPING3
app.PUMPING3 = uilabel(app.Pumping3);
app.PUMPING3.FontName = 'Times New Roman';
app.PUMPING3.FontSize = 16;
app.PUMPING3.FontWeight = 'bold';
app.PUMPING3.FontAngle = 'italic';
app.PUMPING3.FontColor = [0 0.4471 0.7412];
app.PUMPING3.Position = [196 349 173 22];
app.PUMPING3.Text = 'WATER PUMPING (3/4)';

% Create PUMPCURVELabel
app.PUMPCURVELabel = uilabel(app.Pumping3);
app.PUMPCURVELabel.FontName = 'Times New Roman';
app.PUMPCURVELabel.FontSize = 13;
app.PUMPCURVELabel.FontWeight = 'bold';
app.PUMPCURVELabel.FontColor = [0 0.4471 0.7412];
app.PUMPCURVELabel.Position = [61 367 92 22];
app.PUMPCURVELabel.Text = 'PUMP CURVE';

% Create POWERCURVESLabel
app.POWERCURVESLabel = uilabel(app.Pumping3);
app.POWERCURVESLabel.FontName = 'Times New Roman';
app.POWERCURVESLabel.FontSize = 13;
app.POWERCURVESLabel.FontWeight = 'bold';
app.POWERCURVESLabel.FontColor = [0 0.4471 0.7412];
app.POWERCURVESLabel.Position = [268 182 111 22];
app.POWERCURVESLabel.Text = 'POWER CURVES';

% Create HELPSwitchLabel_10
app.HELPswitchLabel_10 = uilabel(app.Pumping3);
app.HELPswitchLabel_10.HorizontalAlignment = 'center';
app.HELPswitchLabel_10.FontName = 'Times New Roman';
app.HELPswitchLabel_10.FontSize = 13;
app.HELPswitchLabel_10.FontWeight = 'bold';
app.HELPswitchLabel_10.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_10.Position = [319.5 312 41 22];
app.HELPswitchLabel_10.Text = 'HELP';

% Create HELPPump3
app.HELPPump3 = uiswitch(app.Pumping3, 'toggle');
app.HELPPump3.Orientation = 'horizontal';
app.HELPPump3.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPPump3ValueChanged, true);
app.HELPPump3.FontName = 'Times New Roman';
app.HELPPump3.FontSize = 13;
app.HELPPump3.FontWeight = 'bold';
app.HELPPump3.FontColor = [0.4667 0.6745 0.1882];
app.HELPPump3.Position = [233 313 45 20];

% Create Pumping4
app.Pumping4 = uitab(app.TabGroup);
app.Pumping4.Title = 'Pumping (4)';
app.Pumping4.BackgroundColor = [0.9686 0.9686 1];
app.Pumping4.ForegroundColor = [0 0.4471 0.7412];

```

```

% Create PUMPING4
app.PUMPING4 = uilabel(app.Pumping4);
app.PUMPING4.FontName = 'Times New Roman';
app.PUMPING4.FontSize = 16;
app.PUMPING4.FontWeight = 'bold';
app.PUMPING4.FontAngle = 'italic';
app.PUMPING4.FontColor = [0 0.4471 0.7412];
app.PUMPING4.Position = [196 362 173 22];
app.PUMPING4.Text = 'WATER PUMPING (4/4)';

% Create MOTORLabel
app.MOTORLabel = uilabel(app.Pumping4);
app.MOTORLabel.FontName = 'Times New Roman';
app.MOTORLabel.FontSize = 13;
app.MOTORLabel.FontWeight = 'bold';
app.MOTORLabel.FontColor = [0 0.4471 0.7412];
app.MOTORLabel.Position = [19 341 55 22];
app.MOTORLabel.Text = 'MOTOR';

% Create P2ratedkWEeditFieldLabel
app.P2ratedkWEeditFieldLabel = uilabel(app.Pumping4);
app.P2ratedkWEeditFieldLabel.HorizontalAlignment = 'right';
app.P2ratedkWEeditFieldLabel.FontName = 'Times New Roman';
app.P2ratedkWEeditFieldLabel.FontSize = 13;
app.P2ratedkWEeditFieldLabel.FontWeight = 'bold';
app.P2ratedkWEeditFieldLabel.Position = [6 316 82 22];
app.P2ratedkWEeditFieldLabel.Text = 'P2rated (kW)';

% Create P2ratedI
app.P2ratedI = uieditfield(app.Pumping4, 'numeric');
app.P2ratedI.ValueChangedFcn = createCallbackFcn(app, ↵
@P2ratedIValueChanged, true);
app.P2ratedI.FontName = 'Times New Roman';
app.P2ratedI.FontSize = 13;
app.P2ratedI.FontWeight = 'bold';
app.P2ratedI.Position = [146 316 81 22];

% Create RPMnomrpmLabel
app.RPMnomrpmLabel = uilabel(app.Pumping4);
app.RPMnomrpmLabel.HorizontalAlignment = 'right';
app.RPMnomrpmLabel.FontName = 'Times New Roman';
app.RPMnomrpmLabel.FontSize = 13;
app.RPMnomrpmLabel.FontWeight = 'bold';
app.RPMnomrpmLabel.Position = [7 291 95 22];
app.RPMnomrpmLabel.Text = 'RPMnom (rpm)';

% Create RPMnomI
app.RPMnomI = uieditfield(app.Pumping4, 'numeric');
app.RPMnomI.ValueChangedFcn = createCallbackFcn(app, ↵
@RPMnomIValueChanged, true);
app.RPMnomI.FontName = 'Times New Roman';
app.RPMnomI.FontSize = 13;
app.RPMnomI.FontWeight = 'bold';
app.RPMnomI.Position = [146 291 81 22];

% Create RPMcool0100Label
app.RPMcool0100Label = uilabel(app.Pumping4);
app.RPMcool0100Label.HorizontalAlignment = 'right';

```

```

app.RPMcool100Label.FontName = 'Times New Roman';
app.RPMcool100Label.FontSize = 13;
app.RPMcool100Label.FontWeight = 'bold';
app.RPMcool100Label.Position = [6 265 113 22];
app.RPMcool100Label.Text = 'RPMcool (0-100%)';

% Create RPMcoolI
app.RPMcoolI = uieditfield(app.Pumping4, 'numeric');
app.RPMcoolI.Limits = [0 100];
app.RPMcoolI.ValueChangedFcn = createCallbackFcn(app, ↵
@RPMcoolIValueChanged, true);
app.RPMcoolI.FontName = 'Times New Roman';
app.RPMcoolI.FontSize = 13;
app.RPMcoolI.FontWeight = 'bold';
app.RPMcoolI.Position = [146 265 81 22];

% Create RPMmax100150Label
app.RPMmax100150Label = uilabel(app.Pumping4);
app.RPMmax100150Label.HorizontalAlignment = 'right';
app.RPMmax100150Label.FontName = 'Times New Roman';
app.RPMmax100150Label.FontSize = 13;
app.RPMmax100150Label.FontWeight = 'bold';
app.RPMmax100150Label.Position = [6 240 127 22];
app.RPMmax100150Label.Text = 'RPMmax (100-150%)';

% Create RPMmaxI
app.RPMmaxI = uieditfield(app.Pumping4, 'numeric');
app.RPMmaxI.Limits = [100 150];
app.RPMmaxI.ValueChangedFcn = createCallbackFcn(app, ↵
@RPMmaxIValueChanged, true);
app.RPMmaxI.FontName = 'Times New Roman';
app.RPMmaxI.FontSize = 13;
app.RPMmaxI.FontWeight = 'bold';
app.RPMmaxI.Position = [146 240 81 22];
app.RPMmaxI.Value = 100;

% Create P2RatedLabel
app.P2RatedLabel = uilabel(app.Pumping4);
app.P2RatedLabel.FontName = 'Times New Roman';
app.P2RatedLabel.FontSize = 13;
app.P2RatedLabel.FontColor = [0.4667 0.6745 0.1882];
app.P2RatedLabel.Visible = 'off';
app.P2RatedLabel.Position = [238 316 100 22];
app.P2RatedLabel.Text = 'Rated shaft power';

% Create RPMnomLabel
app.RPMnomLabel = uilabel(app.Pumping4);
app.RPMnomLabel.FontName = 'Times New Roman';
app.RPMnomLabel.FontSize = 13;
app.RPMnomLabel.FontColor = [0.4667 0.6745 0.1882];
app.RPMnomLabel.Visible = 'off';
app.RPMnomLabel.Position = [238 291 68 22];
app.RPMnomLabel.Text = 'Rated speed';

% Create RPMcoolLabel
app.RPMcoolLabel = uilabel(app.Pumping4);
app.RPMcoolLabel.FontName = 'Times New Roman';
app.RPMcoolLabel.FontSize = 13;

```

```

app.RPMcoolLabel.FontColor = [0.4667 0.6745 0.1882];
app.RPMcoolLabel.Visible = 'off';
app.RPMcoolLabel.Position = [238 265 332 22];
app.RPMcoolLabel.Text = 'Minimum speed, relative to the rated speed, ↵
for water cooling';

% Create RPMmaxLabel
app.RPMmaxLabel = uilabel(app.Pumping4);
app.RPMmaxLabel.FontName = 'Times New Roman';
app.RPMmaxLabel.FontSize = 13;
app.RPMmaxLabel.FontColor = [0.4667 0.6745 0.1882];
app.RPMmaxLabel.Visible = 'off';
app.RPMmaxLabel.Position = [238 240 234 22];
app.RPMmaxLabel.Text = 'Maximum speed, relative to the rated speed';

% Create PowerEfficTable
app.PowerEfficTable = uitable(app.Pumping4);
app.PowerEfficTable.ColumnName = {'P2 [kW]'; 'Efficiency [%]'};
app.PowerEfficTable.ColumnWidth = {75, 75};
app.PowerEfficTable.RowName = {};
app.PowerEfficTable.ColumnEditable = true;
app.PowerEfficTable.CellEditCallback = createCallbackFcn(app, ↵
@PowerEfficTableCellEdit, true);
app.PowerEfficTable.FontName = 'Times New Roman';
app.PowerEfficTable.FontSize = 13;
app.PowerEfficTable.Position = [73 25 160 185];

% Create PowerEfficGraph
app.PowerEfficGraph = uiaxes(app.Pumping4);
title(app.PowerEfficGraph, 'Power efficiency')
xlabel(app.PowerEfficGraph, 'Shaft power, P2 [kW]')
ylabel(app.PowerEfficGraph, 'Efficiency [%]')
app.PowerEfficGraph.FontName = 'Times New Roman';
app.PowerEfficGraph.XTick = [0 0.2 0.4 0.6 0.8 1];
app.PowerEfficGraph.XTickLabel = {'0'; '0.2'; '0.4'; '0.6'; '0.8'; ↵
'1'};
app.PowerEfficGraph.YTick = [0 20 40 60 80 100];
app.PowerEfficGraph.YTickLabel = {'0'; '20'; '40'; '60'; '80'; '100'};
app.PowerEfficGraph.BackgroundColor = [0.8902 0.902 1];
app.PowerEfficGraph.Position = [268 26 300 185];

% Create HELPSwitchLabel_11
app.HELPswitchLabel_11 = uilabel(app.Pumping4);
app.HELPswitchLabel_11.HorizontalAlignment = 'center';
app.HELPswitchLabel_11.FontName = 'Times New Roman';
app.HELPswitchLabel_11.FontSize = 13;
app.HELPswitchLabel_11.FontWeight = 'bold';
app.HELPswitchLabel_11.FontColor = [0.4667 0.6745 0.1882];
app.HELPswitchLabel_11.Position = [559.5 353 41 22];
app.HELPswitchLabel_11.Text = 'HELP';

% Create HELPPump4
app.HELPPump4 = uiswitch(app.Pumping4, 'toggle');
app.HELPPump4.Orientation = 'horizontal';
app.HELPPump4.ValueChangedFcn = createCallbackFcn(app, ↵
@HELPPump4ValueChanged, true);
app.HELPPump4.FontName = 'Times New Roman';
app.HELPPump4.FontSize = 13;

```

```

app.HELPPump4.FontWeight = 'bold';
app.HELPPump4.FontColor = [0.4667 0.6745 0.1882];
app.HELPPump4.Position = [473 354 45 20];

% Create Options
app.Options = uitab(app.TabGroup);
app.Options.Title = 'Options';
app.Options.BackgroundColor = [0.9686 0.9686 1];
app.Options.ForegroundColor = [0 0.4471 0.7412];

% Create PVApplicationLabel
app.PVApplicationLabel = uilabel(app.Options);
app.PVApplicationLabel.HorizontalAlignment = 'center';
app.PVApplicationLabel.FontName = 'Times New Roman';
app.PVApplicationLabel.FontSize = 13;
app.PVApplicationLabel.FontWeight = 'bold';
app.PVApplicationLabel.FontColor = [0 0.4471 0.7412];
app.PVApplicationLabel.Position = [71.5 348 91 22];
app.PVApplicationLabel.Text = 'PV Application';

% Create ApplicationI
app.ApplicationI = uilistbox(app.Options);
app.ApplicationI.Items = {'Grid-Connected', 'Stand-alone PV system', ↵
'Hybrid PV-diesel', 'Hybrid PV-wind-diesel', 'Water pumping'};
app.ApplicationI.ValueChangedFcn = createCallbackFcn(app, ↵
@ApplicationIValueChanged, true);
app.ApplicationI.FontName = 'Times New Roman';
app.ApplicationI.FontSize = 13;
app.ApplicationI.Position = [32 250 167 99];
app.ApplicationI.Value = 'Grid-Connected';

% Create DiffuseskymodelDropDownLabel
app.DiffuseskymodelDropDownLabel = uilabel(app.Options);
app.DiffuseskymodelDropDownLabel.HorizontalAlignment = 'right';
app.DiffuseskymodelDropDownLabel.FontName = 'Times New Roman';
app.DiffuseskymodelDropDownLabel.FontSize = 13;
app.DiffuseskymodelDropDownLabel.FontWeight = 'bold';
app.DiffuseskymodelDropDownLabel.Position = [264 348 104 22];
app.DiffuseskymodelDropDownLabel.Text = 'Diffuse sky model';

% Create DiffuseModel
app.DiffuseModel = uidropdown(app.Options);
app.DiffuseModel.Items = {'Isotropic', 'Anisotropic (Hay)', ↵
'Anisotropic (Perez)'};
app.DiffuseModel.ValueChangedFcn = createCallbackFcn(app, ↵
@DiffuseModelValueChanged, true);
app.DiffuseModel.FontName = 'Times New Roman';
app.DiffuseModel.FontSize = 13;
app.DiffuseModel.Position = [383 348 100 22];
app.DiffuseModel.Value = 'Isotropic';

% Create DustLabel
app.DustLabel = uilabel(app.Options);
app.DustLabel.HorizontalAlignment = 'center';
app.DustLabel.FontName = 'Times New Roman';
app.DustLabel.FontSize = 13;
app.DustLabel.Position = [16.5 178 334 32];
app.DustLabel.Text = 'Degree of soiling on the PV generator under' ↵

```

```

normal incidence';

% Create DustDegreeI
app.DustDegreeI = uibuttongroup(app.Options);
app.DustDegreeI.SelectionChangedFcn = createCallbackFcn(app, ↵
@DustDegreeISelectionChanged, true);
app.DustDegreeI.ForegroundColor = [0 0.4471 0.7412];
app.DustDegreeI.TitlePosition = 'centertop';
app.DustDegreeI.Title = 'Dust degree';
app.DustDegreeI.BackgroundColor = [0.8902 0.898 1];
app.DustDegreeI.FontName = 'Times New Roman';
app.DustDegreeI.FontWeight = 'bold';
app.DustDegreeI.FontSize = 13;
app.DustDegreeI.Position = [88 67 141 109];

% Create Clean
app.Clean = uitogglebutton(app.DustDegreeI);
app.Clean.Text = 'Clean (0%)';
app.Clean.FontName = 'Times New Roman';
app.Clean.Position = [21 64 100 22];
app.Clean.Value = true;

% Create Low
app.Low = uitogglebutton(app.DustDegreeI);
app.Low.Text = 'Low (2%)';
app.Low.BackgroundColor = [1 1 1];
app.Low.FontName = 'Times New Roman';
app.Low.FontColor = [0.149 0.149 0.149];
app.Low.Position = [21 43 100 22];

% Create Medium
app.Medium = uitogglebutton(app.DustDegreeI);
app.Medium.Text = 'Medium (3%)';
app.Medium.FontName = 'Times New Roman';
app.Medium.Position = [21 22 100 22];

% Create High
app.High = uitogglebutton(app.DustDegreeI);
app.High.Text = 'High (8%)';
app.High.BackgroundColor = [1 1 1];
app.High.FontName = 'Times New Roman';
app.High.Position = [21 1 100 22];

% Create GroundreflectanceSpinnerLabel
app.GroundreflectanceSpinnerLabel = uilabel(app.Options);
app.GroundreflectanceSpinnerLabel.HorizontalAlignment = 'right';
app.GroundreflectanceSpinnerLabel.FontName = 'Times New Roman';
app.GroundreflectanceSpinnerLabel.FontSize = 13;
app.GroundreflectanceSpinnerLabel.FontWeight = 'bold';
app.GroundreflectanceSpinnerLabel.Position = [327 110 113 22];
app.GroundreflectanceSpinnerLabel.Text = 'Ground reflectance';

% Create GroundReflectanceI
app.GroundReflectanceI = uispanner(app.Options);
app.GroundReflectanceI.Step = 0.05;
app.GroundReflectanceI.Limits = [0 0.5];
app.GroundReflectanceI.ValueChangedFcn = createCallbackFcn(app, ↵
@GroundReflectanceIValueChanged, true);

```

```

app.GroundReflectanceI.FontName = 'Times New Roman';
app.GroundReflectanceI.FontSize = 13;
app.GroundReflectanceI.FontWeight = 'bold';
app.GroundReflectanceI.Position = [447 100 84 43];

% Create DiffuseFractionI
app.DiffuseFractionI = uibuttongroup(app.Options);
app.DiffuseFractionI.SelectionChangedFcn = createCallbackFcn(app, ↵
@DiffuseFractionISelectionChanged, true);
app.DiffuseFractionI.ForegroundColor = [0 0.451 0.7412];
app.DiffuseFractionI.Title = 'Monthly diffuse correlation';
app.DiffuseFractionI.BackgroundColor = [0.8902 0.902 1];
app.DiffuseFractionI.FontName = 'Times New Roman';
app.DiffuseFractionI.FontWeight = 'bold';
app.DiffuseFractionI.FontSize = 13;
app.DiffuseFractionI.Position = [430 196 163 125];

% Create Page
app.Page = uiradiobutton(app.DiffuseFractionI);
app.Page.Text = 'Page';
app.Page.FontName = 'Times New Roman';
app.Page.Position = [10 73 58 22];
app.Page.Value = true;

% Create CollaresPereira
app.CollaresPereira = uiradiobutton(app.DiffuseFractionI);
app.CollaresPereira.Text = 'Collares-Pereira';
app.CollaresPereira.FontName = 'Times New Roman';
app.CollaresPereira.Position = [10 51 100 22];

% Create Erbs
app.Erbs = uiradiobutton(app.DiffuseFractionI);
app.Erbs.Text = 'Erbs';
app.Erbs.FontName = 'Times New Roman';
app.Erbs.Position = [10 29 65 22];

% Create Macagnan
app.Macagnan = uiradiobutton(app.DiffuseFractionI);
app.Macagnan.Text = 'Macagnan';
app.Macagnan.FontName = 'Times New Roman';
app.Macagnan.Position = [10 6 72 22];

% Create SimulationstepEditFieldLabel
app.SimulationstepEditFieldLabel = uilabel(app.Options);
app.SimulationstepEditFieldLabel.HorizontalAlignment = 'right';
app.SimulationstepEditFieldLabel.FontName = 'Times New Roman';
app.SimulationstepEditFieldLabel.FontSize = 13;
app.SimulationstepEditFieldLabel.FontWeight = 'bold';
app.SimulationstepEditFieldLabel.Position = [293 42 91 22];
app.SimulationstepEditFieldLabel.Text = 'Simulation step';

% Create SimulationstepI
app.SimulationstepI = uieditfield(app.Options, 'numeric');
app.SimulationstepI.Limits = [60 3600];
app.SimulationstepI.RoundFractionalValues = 'on';
app.SimulationstepI.ValueChangedFcn = createCallbackFcn(app, ↵
@SimulationstepIValueChanged, true);
app.SimulationstepI.FontName = 'Times New Roman';

```

```

app.SimulationstepI.FontSize = 13;
app.SimulationstepI.FontWeight = 'bold';
app.SimulationstepI.Position = [399 42 110 22];
app.SimulationstepI.Value = 60;

% Create to3600secondsLabel
app.to3600secondsLabel = uilabel(app.Options);
app.to3600secondsLabel.FontName = 'Times New Roman';
app.to3600secondsLabel.FontSize = 13;
app.to3600secondsLabel.FontColor = [0.4667 0.6745 0.1882];
app.to3600secondsLabel.Position = [399 21 114 22];
app.to3600secondsLabel.Text = '(60 to 3600 seconds)';

% Create SIMULATIONOPTIONSLabel
app.SIMULATIONOPTIONSLabel = uilabel(app.Options);
app.SIMULATIONOPTIONSLabel.FontName = 'Times New Roman';
app.SIMULATIONOPTIONSLabel.FontSize = 16;
app.SIMULATIONOPTIONSLabel.FontWeight = 'bold';
app.SIMULATIONOPTIONSLabel.FontAngle = 'italic';
app.SIMULATIONOPTIONSLabel.FontColor = [0 0.4471 0.7412];
app.SIMULATIONOPTIONSLabel.Position = [221 271 179 22];
app.SIMULATIONOPTIONSLabel.Text = 'SIMULATION OPTIONS';

% Create calculate
app.calculate = uibutton(app.UIFigure, 'push');
app.calculate.ButtonPushedFcn = createCallbackFcn(app, ↵
@calculateButtonPushed, true);
app.calculate.BackgroundColor = [0.9686 0.9686 1];
app.calculate.FontName = 'Times New Roman';
app.calculate.FontSize = 17;
app.calculate.FontWeight = 'bold';
app.calculate.FontAngle = 'italic';
app.calculate.FontColor = [0.502 0.502 0.702];
app.calculate.Position = [372 2 131 28];
app.calculate.Text = 'CALCULATE';

% Create SaveData
app.SaveData = uicheckbox(app.UIFigure);
app.SaveData.ValueChangedFcn = createCallbackFcn(app, ↵
@SaveDataValueChanged, true);
app.SaveData.Text = 'SAVE DATA';
app.SaveData.FontName = 'Times New Roman';
app.SaveData.FontSize = 16;
app.SaveData.FontWeight = 'bold';
app.SaveData.FontColor = [0.502 0.502 0.702];
app.SaveData.Position = [25 5 110 22];

% Create instructions
app.instructions = uilabel(app.UIFigure);
app.instructions.FontName = 'Times New Roman';
app.instructions.FontSize = 14;
app.instructions.FontAngle = 'italic';
app.instructions.FontColor = [0.502 0.502 0.702];
app.instructions.Position = [10 28 526 22];
app.instructions.Text = 'To save input values, select ''SAVE DATA'', ↵
insert the project name and press ''CALCULATE''';

% Create ProjectnameLabel

```

```

app.ProjectnameLabel = uilabel(app.UIFigure);
app.ProjectnameLabel.HorizontalAlignment = 'right';
app.ProjectnameLabel.FontName = 'Times New Roman';
app.ProjectnameLabel.FontSize = 14;
app.ProjectnameLabel.FontWeight = 'bold';
app.ProjectnameLabel.FontColor = [0.502 0.502 0.702];
app.ProjectnameLabel.Position = [149 5 85 22];
app.ProjectnameLabel.Text = 'Project name';

% Create nameI
app.nameI = uieditfield(app.UIFigure, 'text');
app.nameI.ValueChangedFcn = createCallbackFcn(app, @nameIValueChanged, ↵
true);
app.nameI.FontName = 'Times New Roman';
app.nameI.FontSize = 14;
app.nameI.FontColor = [0.502 0.502 0.702];
app.nameI.Position = [249 5 100 22];

% Create loadProject
app.loadProject = uibutton(app.UIFigure, 'push');
app.loadProject.ButtonPushedFcn = createCallbackFcn(app, ↵
@loadProjectButtonPushed, true);
app.loadProject.BackgroundColor = [0.9686 0.9686 1];
app.loadProject.FontName = 'Times New Roman';
app.loadProject.FontSize = 13;
app.loadProject.FontWeight = 'bold';
app.loadProject.FontColor = [0.502 0.502 0.702];
app.loadProject.Position = [527 6 110 33];
app.loadProject.Text = 'LOAD PROJECT';

% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end

% App creation and deletion
methods (Access = public)

    % Construct app
    function app = appInterface1

        % Create UIFigure and components
        createComponents(app)

        % Register the app with App Designer
        registerApp(app, app.UIFigure)

        % Execute the startup function
        runStartupFcn(app, @startupFcn)

        if nargout == 0
            clear app
        end
    end

    % Code that executes before app deletion
    function delete(app)

```

```
% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end
end
```

ANEXO B. CÓDIGO IMPLEMENTADO EN MATLAB

B.1. SCRIPT NEWREADINPUTDATA

```

% newReadInputData

% Site.
%Latitude of the location, positive in the Northern Hemisphere and negative
%in the Southern Hemisphere.
Latitude=evalin("base","Latitude");

%Latitude, radians. Internal calculation.
lat=Latitude*pi/180;

%Longitude of the location, negative towards West and positive towards Est.
Longitude=evalin("base","Longitude");

%Altitude of the location over sea level.
Altitude=evalin("base","Altitude");

%Standard longitude of the local meridian (multiple of 15), negative towards
%West and positive towards East.
StandardLongitude=evalin("base","StandardLongitude");
TimeZone=StandardLongitude/15;

%End Site

%%%%%%%%%%%%%%%
% Meteorological data.
%%%%%%%%%%%%%%%
%Input data
% 1-Monthly averages from the excel sheet
% 2-Monthly averages obtained from PVGIS TMY
% 3-Hourly values from PVGIS TMY
InputDialog=evalin("base","InputDialog");
if(InputData==1)
    months_Data=evalin("base","months_Data");
    %Read monthly averages from the excel sheet
    %Mean daily global horizontal irradiation, monthly average, Wh/m2.
    Gdm0=(months_Data(1:12,1)');
    %Minimum daily temperature, monthly average, °C.
    Tmm=(months_Data(1:12,2)');
    %Maximum daily temperature, monthly average, °C.
    TMm=(months_Data(1:12,3)');
elseif (InputDialog==2 || InputData==3)
    %Read TMY PGIS *.csv file
    ReadTMYPVGIS;
elseif (InputDialog==4 || InputData==5)
    %Read USA TMY3 *.csv file
    ReadTMY3;
end

%Generation of time series
TimeSeries=evalin("base","TimeSeries");
%1. Mean days
%2. ...

```

```

% End Meteo.

%%%%%%%%%%%%%%%
% PV generator
%%%%%%%%%%%%%%%
%Nominal PV power
PVnom=evalin("base", "PVnom");
%Coefficient of Variation of module Power with Temperature (absolute value), %
CVPT=evalin("base", "CVPT");
%Nominal Operation Cell Temperature, °C
NOCT=evalin("base", "NOCT");
%Thermal resistance, °C·m^2/W
Rth=evalin("base", "Rth");
%Inclination
InclinationGround=evalin("base", "InclinationGround");
InclinationDelta=evalin("base", "InclinationDelta");
InclinationTracking=evalin("base", "InclinationTracking");
%Orientation of the modules towards the Equator.
%Zero towards the South in the Northern Hemisphere (North in the Southern Hemisphere),
%negative towards the East, and positive towards the West.
Orientation=evalin("base", "Orientation");
%Mounting structure
Mounting=evalin("base", "Mounting");
if (Mounting==1)
    %Static ground or roof
    %Inclination of the modules regarding the horizontal, from 0° to 90°.
    Inclination=InclinationGround;
elseif (Mounting==2)
    %Static delta
    %Inclination of the modules regarding the horizontal, from 0° to 90°.
    %The same for East and West structures.
    Inclination=InclinationDelta;
elseif (Mounting==4)
    %Azimutal tracker
    %Inclination of the modules regarding the horizontal, from 0° to 90°.
    Inclination=InclinationTracking;
end
%%%%%%%%%%%%%%%
%End PVgen
%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%
% Inverter
%%%%%%%%%%%%%%%
%Nominal output power, kW
PINom=evalin("base", "PINom");
%Maximum output power, kW
PImax=evalin("base", "PImax");
%Power efficiency curve
InverterCurve=evalin("base", "InverterCurve");

%points: matrix composed of pac and efficiency

```

```

points=evalin("base", "points");
pac=points(1:6,1);
Efficiency=points(1:6,2);

if(InverterCurve==1)
    %Power efficiency curve parameters
    k0=evalin("base", "k0");
    k1=evalin("base", "k1");
    k2=evalin("base", "k2");
else
    %Calculation of the previous parameters starting power efficiency
    %points
    [k0, k1, k2]=InverterParameters(pac, Efficiency);
end
%%%%%%%%%%%%%%%
%End Inverter
%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%
% Wiring.
%%%%%%%%%%%%%%%
%DC losses, %
WDC=evalin("base", "WDC");
%LV losses, %
WAC=evalin("base", "WAC");
%%%%%%%%%%%%%%%
%End Wiring.
%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%
% Battery
%%%%%%%%%%%%%%%
%Battery. For stand-alone PV systems
%Battery capacity, kWh
CBAT=evalin("base", "CBAT");
%Maximum SOC
SOCmax=evalin("base", "SOCmax");
%Minimum SOC
SOCmin=evalin("base", "SOCmin");
%%%%%%%%%%%%%%%
%End Battery
%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%
% Load profiles
%%%%%%%%%%%%%%%
%Monthly average of daily energy consumption, kWh/day
Ldm_Data=evalin("base", "Ldm_Data");
Ldm=(Ldm_Data');
%Yearly energy demand, kWh
Edemanda=evalin("base", "Edemanda");
%Normalised daily load profiles
F_Data=evalin("base", "F_Data");
F=(F_Data');
%%%%%%%%%%%%%%

```

```

%End Load
%%%%%%%%%%%%%%%
% Options
%%%%%%%%%%%%%%%
%PV application
Application=evalin("base","Application");
if(Application==4 && InputData~=3 && InputData~=5)
    error('This simulation requires hourly wind data. Select InputData = ↵
Monthly averages obtained from USA TMY3')
end

%Degree of dust/soiling
DustDegree=evalin("base","DustDegree");
%Model of diffuse radiation
Diffuse_model=evalin("base","Diffuse_model");
%Monthly correlation between the fraction of diffuse and clearness index
Diffuse_fraction=evalin("base","Diffuse_fraction");
%Ground reflectance
GroundReflectance=evalin("base","GroundReflectance");

%Other parameters
%Minimum irradiance required to injecting power in the grid, W/m2
Gth=0;
%Number of simulated days.
Ndays=365;
%Simulation step, seconds. Up to one hour maximum.
SimulationStep=evalin("base","SimulationStep");
%For TMY data the simulation step must be one hour
if(InputData==3 || InputData==5 )
    SimulationStep=3600;
end
%Number of simulation points per day
Nsteps=floor((24*60*60)/SimulationStep);
%Number of simulation points per hour
Stepph=3600/SimulationStep;
%%%%%%%%%%%%%%%
%End Options
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%PUMPING DATA
%%%%%%%%%%%%%%%
%Well/borehole
%Static head, m
Hs=evalin("base","Hs");
%Constant test flow, m3/h
Qtest=evalin("base","Qtest");
%Drawdown at constant test flow
Hdr=evalin("base","Hdr");
%Aquifer loss
kw1=evalin("base","kw1");
%Well loss
kw2=evalin("base","kw2");
%%%%%%%%%%%%%%%

```

```

%Water tank
%Water tank capacity
WTC=evalin("base", "WTC");
%Discharge level
Hr=evalin("base", "Hr");
%%%%%%%%%%%%%
%Pump
%Rated flow, m3/h
Qrated=evalin("base", "Qrated");
%Rated head, m
Hrated=evalin("base", "Hrated");
%Liquid density, kg/m3
Density=evalin("base", "Density");
%%%%%%%%%%%%%
%System curve
%Friction losses, m
Hf=evalin("base", "Hf");
%Pump curve
PumpCurve=evalin("base", "PumpCurve");
%Powers
PowerCurves=evalin("base", "PowerCurves");
%Motor
%Rated shaft power, kW
P2rated=evalin("base", "P2rated");
%Rated speed
RPMnom=evalin("base", "RPMnom");
% Minimum speed
RPMcoolM=evalin("base", "RPMcoolM");
RPMcool=RPMnom*RPMcoolM/100;
% Maximum speed
RPMmaxM=evalin("base", "RPMmaxM");
RPMmax=RPMnom*RPMmaxM/100;
%Motor power efficiency
PowerEffic=evalin("base", "PowerEffic");

%%%%%%%%%%%%%
%End Pumping
%%%%%%%%%%%%%

```

```

%%%%%%%%%%%%%
% Generator set
%%%%%%%%%%%%%
%Nominal power of the genset
PGENnom=evalin("base", "PGENnom");
%State of charge for connecting the genset
SOCstart=evalin("base", "SOCstart");
%State of charge for disconnecting the genset
SOCstop=evalin("base", "SOCstop");
%Fuel consumption model
%Intercept coefficient 0
b0i=evalin("base", "b0i");
%Intercept coefficient 1
b1i=evalin("base", "b1i");
%Slope coefficient 0
b0s=evalin("base", "b0s");
%Slope coefficient 1
b1s=evalin("base", "b1s");

```

```
%End Genset
```

```
% Wind generator
```

```
% Rated electrical power
```

```
PWnom=evalin("base", "PWnom");
```

```
% Rated wind speed
```

```
Vnom=evalin("base", "Vnom");
```

```
% Cut-in wind speed
```

```
Vci=evalin("base", "Vci");
```

```
% Cut-out wind speed
```

```
Vco=evalin("base", "Vco");
```

```
% Equivalent power coefficient ( $C_{peq}=0,5 \cdot r \cdot A \cdot Cp$ ), kW / (m/s) 3
```

```
Cpeq=PWnom/ (Vnom^3-Vci^3);
```

```
% End Wind
```

```
% Save data in the same or a different project
```

```
saveDataApp=evalin("base", "saveDataApp");
```

```
name=evalin("base", "name");
```

B.2. SCRIPT PUMPING MODEL

```

% PumpingModel

%Curve parameters
Ks=[kw2+Hf/ (Qrated^2)    kw1    (Hs+Hr) ];
Ks0=Ks (3);
Ks1=Ks (2);
Ks2=Ks (1);

%Flow nominal curve, m3/h
Qnom=PumpCurve(1:7,1)';
%Removes NaN
Qnom(isnan(Qnom))=[];
%Head nominal curve, m
Hnom=PumpCurve(1:7,2)';
%Removes NaN
Hnom(isnan(Hnom))=[];
%Fit pump curve with a second-degree polynomial (H=Kh0 + Kh1*Q + Kh2*Q*Q)
FitResult1=fit(Qnom', Hnom', 'poly2');
Kh=coeffvalues(FitResult1);
Kh0=Kh(3);
Kh1=Kh(2);
Kh2=Kh(1);
%Pump curve
%plot(FitResult1, '-k', Qnom, Hnom, 'ko');
%xlabel('Flow [m^3/h]');
%ylabel('Head [m]');
%hold on;
%System curve
%x=0:.1:6;
%plot(x,polyval(Ks,x), '-k')

%Nominal curve of power at the motor input P1, kW
QnomP1=PowerCurves(1:7,1)';
%Removes NaN
QnomP1(isnan(QnomP1))=[];
%P1 power
P1nom=PowerCurves(1:7,2)';
%Removes NaN
P1nom(isnan(P1nom))=[];
%Fit pump curve with a second-degree polynomial
FitResult3=fit(QnomP1', P1nom', 'poly2');
Kp1=coeffvalues(FitResult3);
Kp10=Kp1(3);
Kp11=Kp1(2);
Kp12=Kp1(1);
%Nominal curve of shaft power P2, kW
QnomP2=PowerCurves(1:7,1)';
%Removes NaN
QnomP2(isnan(QnomP2))=[];
%P2 power
P2nom=PowerCurves(1:7,3)';
%Removes NaN
P2nom(isnan(P2nom))=[];
%Fit pump curve with a second-degree polynomial
FitResult4=fit(QnomP2', P2nom', 'poly2');


```

```

Kp2=coeffvalues(FitResult4);
Kp20=Kp2(3);
Kp21=Kp2(2);
Kp22=Kp2(1);
%Plot input data and fit results
%figure
%plot(FitResult3, '-k', Qnom, P1nom, 'ko');
%hold on
%plot(FitResult4, '-k', Qnom, P2nom, 'ks');
%ylabel('Power [kW]');
%xlabel('Flow [m^3/h]');
%%%%%%%%%%%%%
%Motor
%Power output P2, kW
P2motor=PowerEffic(1:7,1)';
%Removes NaN
P2motor(isnan(P2motor))=[];
%Efficiency, %
eta_motor=PowerEffic(1:7,2)';
%Removes NaN
eta_motor(isnan(eta_motor))=[];
%Fitting
warning('off','all');
model = fittype('100*p2motor/(p2motor+  

(Km0+Km1*p2motor+Km2*p2motor*p2motor))','ind','p2motor','dep','eta_m');
%Fitted coefficients
[Fitresult5, goodness, output] = fit(P2motor', eta_motor', model);
coefficients=coeffvalues(Fitresult5);
Km0=coefficients(1);
Km1=coefficients(2);
Km2=coefficients(3);
warning('on','all');

%Draw a plot with data and model
%plot(Fitresult5, '-k', P2motor, eta_motor, 'ko');
%ylabel('Motor power efficiency [%]');
%xlabel('P_2 [kW]');
%%%%%%%%%%%%%
% Curve Q-P2 at variable speed
%Flow sweep

Q2=Qrated/100:Qrated/100:1.5*Qrated;
for i=1:length(Q2)
    %Head H2 corresponding to Q2 on the system curve
    %H2(i)=Ks(3)+Ks(1)*Q2(i)^2;
    H2(i)=Ks(3)+Ks(2)*Q2(i)+Ks(1)*Q2(i)^2;

    %Flow Q1 at point 1 on the nominal pump curve
    a=Kh(1)-H2(i)/(Q2(i)^2);
    b=Kh(2);
    c=Kh(3);
    Q1(i)=(-b - sqrt(b^2-4*a*c))/(2*a);
    %Head at point 1 on the pump curve
    H1(i)=Kh0 + Kh1*Q1(i) + Kh2*Q1(i)*Q1(i);
    %Hidraulic power at point 1, kW
    PH1(i)=1e-3*(Density*9.81/3600)*H1(i)*Q1(i);
    %Shaft power at point 1, kW
    P21(i)=Kp20 + Kp21*Q1(i) + Kp22*Q1(i)*Q1(i);

```

```

%Pump efficiency at point 1
eta_p1(i)=PH1(i)/P21(i);
%Which is equal to pump efficiency in point 2
eta_p2(i)=eta_p1(i);
%Hidraulic power at point 2, kW
PH2(i)=1e-3*(Density*9.81/3600)*H2(i)*Q2(i);
%Shaft power at point 2;
P22(i)=PH2(i)/eta_p2(i);
rpm2(i)=Q2(i)/Q1(i);
end
%Fit P22-Q22 curve with a second orden polynominal P22=Kq0 + Kq1*Q + Kq2*Q*Q
FitResult6=fit(Q2', P22', 'poly2');
Kq=coeffvalues(FitResult6);
Kq0=Kq(3);
Kq1=Kq(2);
Kq2=Kq(1);

%plot(FitResult6, '-k', Q2, P22, '--k');
P22fit=Kq0+Kq1*Q2+Kq2*Q2.*Q2;
%plot(rpm2, P22)
%plot(rpm2, P22fit)
%plot(rpm2, Q2)
%plot(P22, Q2, '-k', P22fit, Q2, '--k');

%RPMmin, rpm
rpmmmin=sqrt(Ks0/Kh0);
RPMmin=RPMnom*rpmmmin;

%Flow at the intersection point, Qn, of the system curve and the nominal pump curve
a=Kh2-Ks2;
b=Kh1;
c=Kh0-Ks0;
Qn=(-b - sqrt(b^2-4*a*c)) / (2*a);
if (Qn<0)
    Qn=(-b + sqrt(b^2-4*a*c)) / (2*a);
end
%Maximum flow, m3/h, and head, m
Qmax=Qn*(RPMmax/RPMnom);
Hmax=Ks0+Ks2*(Qmax^2);
%Maximum hidraulic power, kW
PHmax=1e-3*(Density*9.81/3600)*Hmax*Qmax;
%Maximum shaft power, kW
P2max=Kq0 + Kq1*Qmax + Kq2*Qmax*Qmax;
%Maximum P1 power, kW
P1max=P2max + Km0 + Km1*P2max + Km2*P2max*P2max;
%New maximum inverter power
if (P1max<PImax)
    disp('Warning: the maximum pump speed will limit the maximum inverter power');
    to P1max [kW];
    %P1max
end
PImax=P1max;

```

B.3. SCRIPT PVLITE

```

% pvlite
%Clear workspace
clear;

%Input data
disp('Reading inputs ...');
newReadInputData;
disp('Inputs Ok');

%%%%%%%%%%%%%
%Start
%Generation of irradiance time series.
Irradiances;
%Generation of temperature time series.
Temperatures;
%PV power
if(Mounting==2)
    PVpower_delta;
else
    PVpower;
end

%Applications
if (Application==1)
    %Grid-connected PV system
    mainGrid;
elseif (Application==2)
    %Stand-alone PV system
    mainSA;
elseif (Application==3 || Application==4)
    %Stand-alone hybrid PV system
    mainHybrid;
elseif (Application==5)
    %PV pumping
    mainPump;
end

%save project
if(saveDataApp==1)
    save([name, '.mat']);
end
%%%%%%%%%%%%%

```

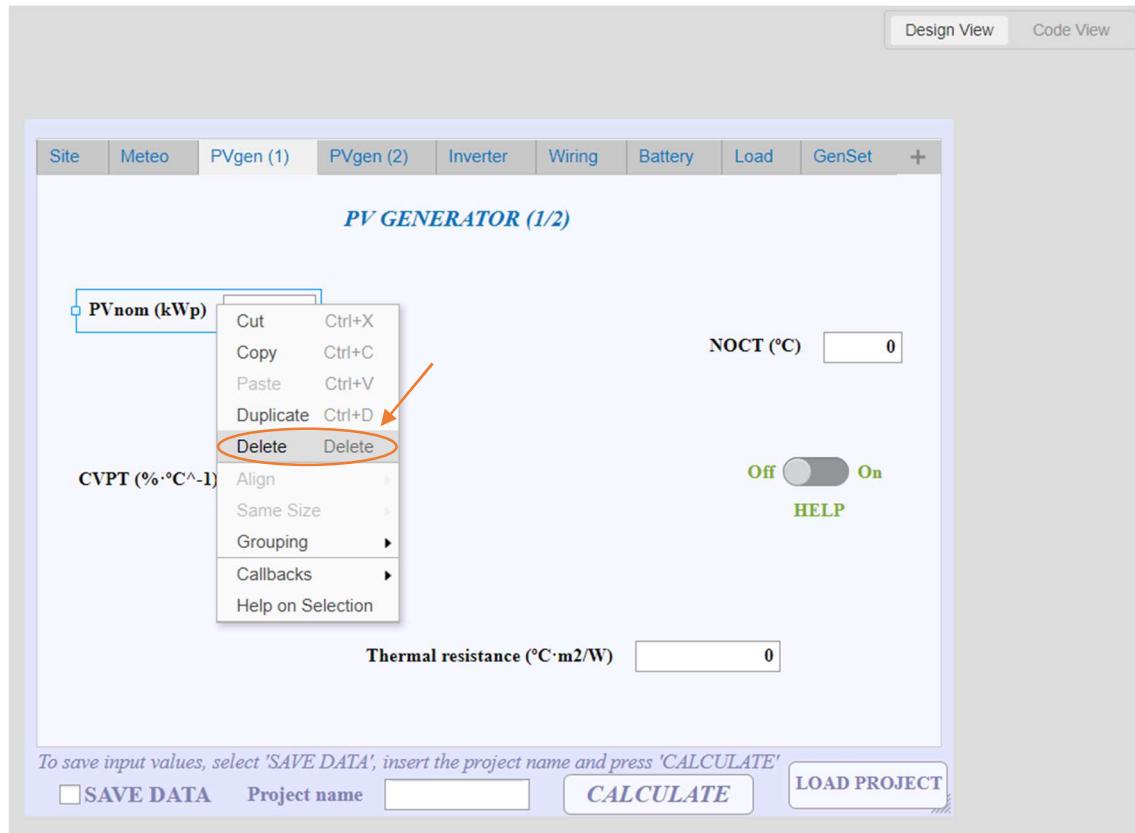
ANEXO C. TUTORIAL PARA AGREGAR O ELIMINAR COMPONENTES DE LA INTERFAZ

En este anexo se explica un breve tutorial por si en un futuro el programa se desea actualizar de alguna manera implementando nuevos componentes o eliminando algunos de los ya existentes. Para que siga funcionando la interfaz en conjunto con el programa de simulación debe haber una coherencia entre ambos y, por lo tanto, si se crea o elimina algún aspecto del programa, de la misma manera se deberá hacer en la interfaz de App Designer.

C.1. ELIMINACIÓN DE COMPONENTES

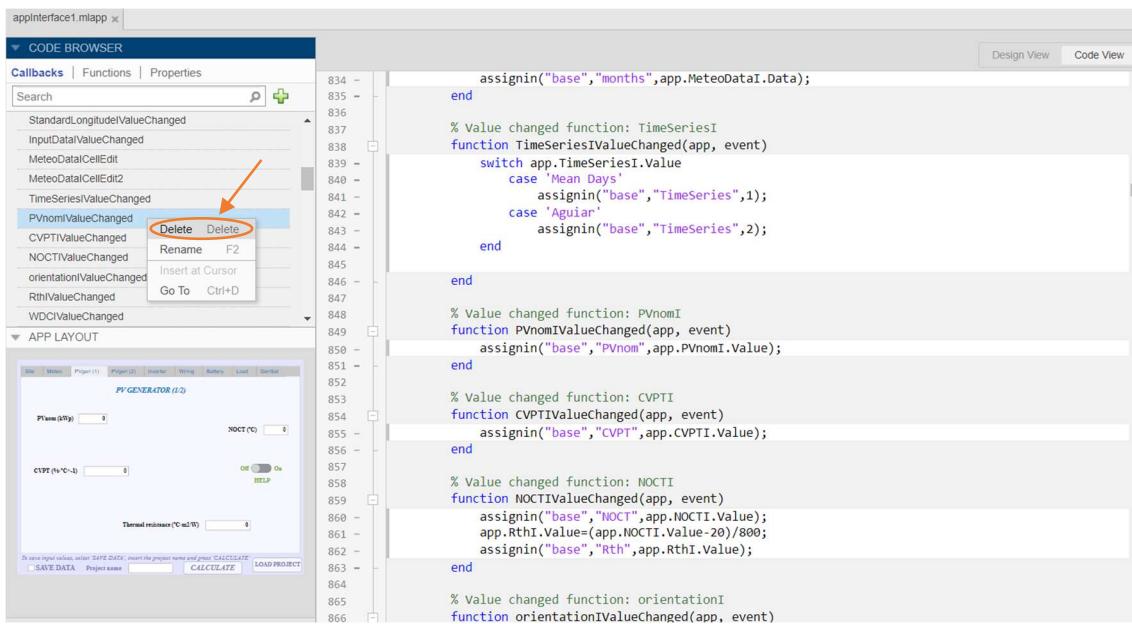
En caso de que en el programa se deje de evaluar alguno de los componentes existentes, habrá que eliminarlo también de la interfaz. Para eliminar un componente de la interfaz se deberá eliminar de todas las partes del código en las que aparezca.

El primer paso será eliminarlo en la *Design View*, esto se hará pulsando sobre el elemento que se desee eliminar, botón derecho y *Delete*, como se indica en la imagen mostrada a continuación.



Eliminación de un componente desde la Design View

Una vez eliminado de la *Design View*, automáticamente se eliminarán todas las funciones que se hayan creado relacionadas con la creación y definición de parámetros iniciales de dicho componente. Esto no ocurre con las *Callbacks* que se hayan creado manualmente, por lo que se deberá ir a la *Code View* y en la parte de la izquierda, donde se listan todas las *Callbacks* se deberá eliminar también (pulsar sobre la *Callback*, botón derecho, *Delete*).



Eliminación de la *Callback* del componente eliminado

Una vez hecho esto, lo último que habría que hacer en App Designer es revisar todo el código (solo el código editable por el programador) y eliminar todos los comandos donde se nombre alguna de las características de este componente, ya sea en otras *Callbacks* de otros componentes, en la función de *startupFcn*, etc.

Con estos pasos ya se habría eliminado el componente en App Designer, por lo que lo único que faltaría por hacer es eliminar todos los comandos relacionados con este componente en el código de Matlab. En el script de *newReadInputData*, el encargado de leer todos los datos, se deberá eliminar la línea de código que lea esta variable, y en caso de haberlos, los cálculos que se realicen los ella.

Por último, en el *Workspace* de Matlab también se recomienda eliminar el espacio reservado para dicha variable, ya que, aunque no daría error si no se usa, ocuparía espacio innecesariamente.

C.2. CREACIÓN DE COMPONENTES

Para la creación de un nuevo componente, se deberá añadir tanto en el código de Matlab como en el de App Designer, al igual que en el caso de la eliminación de componentes.

El primer paso será crear en el *Workspace* en el que están todas las variables, una nueva variable para este nuevo componente con su valor inicial deseado. Se puede crear una matriz, un valor numérico o lo que se quiera, en función del componente que se vaya a añadir en App Designer. En este caso se ha optado por una matriz para la creación de Tablas para poder almacenar varios valores a la vez y números únicos para *Edit Fields*, *Spinner*, etc., pero esto es elección del programador y hay diversas opciones, todas ellas válidas.

Una vez creada esta variable en el *Workspace*, se deberá añadir el componente en la *Design View* de App Designer. La manera de añadirlo, descrita más detalladamente en apartados anteriores, es simplemente escogiendo el componente desde la *Component Library* y arrastrándolo hasta la ventana central. Se colocará donde se desea y pulsando sobre él se podrán cambiar algunas de sus características desde la ventana del *Component Browser*.

Una vez añadido a la *Design View* y definidas sus características *iniciales*, el programa automáticamente añade el código necesario para su creación en la *Code View*. El código que se debe añadir en esta vista para que funcione con el programa completo es el siguiente:

1. En la función *startupFcn*, asignar el valor inicial almacenado en el *Workspace* al componente. Por ejemplo, en caso de ser un *Edit Field* el comando sería:

```
app.NombreComponente.Value=NombreVarWorkspace;
```

2. En el final de la función *startupFcn*, asignar a la variable del *Workspace* el valor del componente (para evitar incoherencias).

```
assignin("base","NombreVarWorkspace", app.NombreComponente.Value);
```

3. Creación de la *Callback* que ejecutará el programa cuando el usuario interaccione con el componente. Para crear una *Callback*, una de las opciones es pulsar sobre el nuevo elemento, botón derecho, 'Callbacks', 'add NombreComponenteValueChanged callback'. Al pulsar ahí, se crea la *Callback* directamente en la *Code View* y únicamente hay que rellenarla con lo que se quiera que haga el programa cuando se modifique el valor del componente.

Se debe asignar el nuevo valor a la variable del *Workspace* cuando el usuario modifique el componente de la interfaz, por lo que dentro de la *Callback* se escribirá el siguiente comando:

```
assignin("base","NombreVarWorkspace", app.NombreComponente.Value);
```

Si se desea que el programa haga algo más cuando se cambie el valor del componente, es aquí donde se deberá escribir el respectivo código.

Tras la realización de estos pasos ya estaría actualizada la interfaz de App Designer, solo faltaría actualizar también el código de Matlab. Para ello, se debe implementar en el script *newReadInputData* el comando necesario para la lectura de la variable del *Workspace*.

```
NombreVar=evalin("base","NombreVarWorkspace");
```

Este comando almacenará en una variable local (llamada NombreVar, que puede ser el mismo nombre que la variable del *Workspace*) el valor introducido por la interfaz. Una vez almacenado, se implementarán los cálculos internos que se deseen, a partir de ahora, llamando únicamente a la variable NombreVar.

Por último, se deberá introducir el código necesario en el programa completo de la simulación del sistema para la obtención de resultados.

Estos pasos descritos están centrados en la creación de un componente de tipo *Edit Field*, que contiene un valor numérico. Para cada tipo de componente los comandos cambiarán y la manera de enfocarlos a la hora de la programación también. Por ejemplo, una tabla es una matriz compuesta de x filas y columnas y se deben definir todas las casillas. Por otro lado, en componentes en los que se elige entre varias opciones se deben realizar estudios con 'switch case' o 'if' para asignar un valor numérico a cada opción y así poder evaluarlas en otras partes del código. Por lo tanto, cada enfoque y comandos tendrán ciertas variaciones en función del tipo de componente, pero el procedimiento seguido descrito anteriormente para la lectura y escritura de datos entre ambas plataformas (App Designer y Matlab) es el mismo. Se pueden coger como ejemplos otros componentes ya creados, ya que en la interfaz se muestra gran variedad de ellos, con distintos enfoques y planteamientos.