Máster en Ingeniería de los Sistemas Fotovoltaicos

Centro de Estudios de Postgrado

Universidad de Jaén

Centro de Estudios de Postgrado

DESARROLLO DE SOFTWARE PARA EL MODELADO Y SIMULACIÓN DE IFV BAJO AFECCIONES POR SOMBREADO

Alumno: **Hierro Roberto, Fabio**

Tutor: Prof. D. Andrés Firman

Dpto: Grupo en Energías Renovables

Diciembre, 2023

A mi madre Meme, a mi hermana Sara y a mi pareja Naomi

A quienes agradeceré y dedicaré cada logro

RESUMEN

El presente Trabajo de Fin de Máster versa sobre el desarrollo de un software para el modelado y simulación de instalaciones fotovoltaicas conectadas a red, con capacidad para la cuantificación de las pérdidas energéticas por sombreados parciales.

A lo largo de este documento, se expondrán las metodologías empleadas para modelar y simular el desempeño energético del sistema, de manera en la que puedan reflejarse las principales variables de las que éste depende: recurso solar, disposición de los módulos fotovoltaicos en el espacio, presencia de obstáculos cercanos, topología eléctrica del generador fotovoltaico y las características de los componentes que integran al sistema.

Este documento finalizará con la exposición de varios casos de estudio, abordados a través del software desarrollado, y con una declaración de trabajos futuros relacionados con él.

PALABRAS CLAVE

Instalaciones fotovoltaicas, sombreado parcial, modelado, simulación, *mismatch*, célula fotovoltaica, *ray casting.*

1 INTRODUCCIÓN

1.1 Antecedentes

El diseño de una instalación fotovoltaica encuentra su origen en el proceso de modelado y simulación, mediante los que se trata de evaluar las métricas que cuantifican su idoneidad para cumplir unos objetivos concretos, generalmente energéticos y/o económicos.

Existen modelos simplificados que permiten evaluar las propiedades de los diseños con bastante exactitud, basados en la suposición de un funcionamiento de la instalación en su punto de máxima potencia, y en los que las pérdidas más habituales suelen considerarse a través de unos coeficientes aplicados sobre los datos de producción. Además, los preceptos básicos del diseño de instalaciones fotovoltaicas orientan a éstos a reducir al máximo las ventanas de tiempo en las que puedan producirse sombreados, y a emplear idénticos modelos y condiciones de instalación (inclinación y orientación) para todos los módulos que estén interconectados, tanto en serie como en paralelo, por lo que generalmente se cumplen las hipótesis simplificadoras asumidas, y se evita la necesidad de modelar la instalación en términos eléctricos, y a nivel de célula fotovoltaica.

Sin embargo, existen una multitud de escenarios en los que se hace necesario estudiar en qué medida la gradual desviación con respecto a las condiciones de operación descritas en el párrafo anterior afectan al desempeño del sistema. Un ejemplo de ello puede ser la situación que enfrentan la *BAPV* (*Building Applied Photovoltaics*) y la *BIPV* (*Building Integrated Photovoltaics*) en contextos urbanos, caracterizados por elevados consumos eléctricos, limitaciones de espacio útil, y presencia de multitud de obstáculos. En estos casos, según el incentivo económico que suponga la instalación fotovoltaica, podrá quedar justificado un diseño orientado a la maximización de la energía generada por la misma, a través del máximo aprovechamiento posible de la superficie disponible, aunque ello implique desviarse de las condiciones ideales de instalación. Esta situación demanda, por tanto, poder cuantificar cómo la proximidad de los obstáculos afecta al rendimiento de la planta, en aras de determinar el diseño que resulte más apropiado, bajo algún criterio preestablecido.

Cabe señalar que la confección de modelos capaces de sintetizar todos los aspectos señalados hasta ahora, puede encontrar serias dificultades: en especial, si se quiere dotar a éstos de la capacidad de tener en cuenta el efecto del sombreado parcial de la instalación, y su dependencia con la topología eléctrica, ya que se hacen necesarios tanto el modelado de la obstrucción a la radiación, como el modelado eléctrico de la instalación, ambos a nivel de célula. En concreto, la dificultad radica en la necesidad de realizar los cálculos con una resolución espacial y temporal

lo suficientemente pequeña como para reflejar de forma fidedigna el efecto del sombreado parcial, que, junto con el carácter no lineal e implícito de las ecuaciones que definen el comportamiento eléctrico de la célula fotovoltaica, hacen inabordables tanto la confección como la simulación de estos modelos si no es a través de herramientas informáticas.

En la actualidad, existen numerosos programas con los que se puede abordar el modelado y la simulación del fotovoltaico. Sin embargo, la mayor parte de ellos son programas comerciales, presentando así las siguientes limitaciones para el usuario:

- Distribuidos bajo licencias de pago, normalmente con caducidad.
- No se tiene acceso al código fuente, lo que implica falta de transparencia en lo relativo a su algoritmia y métodos de cálculo, e incapacidad para ser modificados.
- No pueden ser integrados como parte de algún programa externo para ampliar sus funcionalidades.

Estas limitaciones se suman a las múltiples razones por las que puede resultar de interés, tanto en el ámbito profesional como en el académico, abordar el problema del diseño y simulación a través de lenguajes de programación de propósito general:

- Se alivia la dependencia con respecto al software comercial especializado.
- Sus funcionalidades pueden ser extendidas por el usuario.
- Permite automatizar la parametrización de los diseños.

A su vez, si el código es publicado de forma abierta:

- Se tiene plena transparencia sobre la algoritmia y metodologías de cálculo empleadas.
- Acceso universal.

Otra ventaja añadida del uso de lenguajes de programación de propósito general, tales como *Python* o *C++*, es que cuentan con una extensa cantidad de librerías y de recursos de libre acceso, gratuitos y en continuo desarrollo, en diversos ámbitos de la ciencia y de la ingeniería, que facilitan el ejercicio de programación.

En el contexto de fotovoltaico, una de las librerías más completas de código abierto disponible en la actualidad es *pvlib* [1], escrita en el lenguaje de programación *Python*. Esta librería incorpora una amplia batería de funciones para la simulación de la producción eléctrica de instalaciones conectadas a red, con posibilidad de simular estrategias de seguimiento en uno o varios ejes, o bifacialidad. Además, incorpora funciones para la evaluación del recurso solar, así como para el modelado de módulos fotovoltaicos y el trazado de sus curvas I-V para distintas

condiciones de operación. Sin embargo, esta librería no implementa funciones para modelar con facilidad el efecto del *mismatch* por distintas condiciones de operación entre las células, módulos o cadenas de la planta fotovoltaica. Tampoco se pueden abordar a través de ella el cálculo de los factores de obstrucción a la radiación directa y difusa, provocados por obstáculos cercanos, ni las pérdidas asociadas al cableado eléctrico.

1.2 Objetivos

El objetivo principal del presente Trabajo de Fin de Máster es el desarrollo de una librería, escrita en *Python*, para el modelado y simulación de instalaciones fotovoltaicas conectadas a red, cuyas funcionalidades básicas sean las siguientes:

- Importación de datos meteorológicos y de las componentes de la radiación solar.
- Modelado geométrico y representación de obstáculos y receptores en el espacio.
- Cálculo y representación gráfica de la distribución de la irradiancia e irradiación sobre un receptor, monofacial o bifacial.
- Modelado e implementación de los componentes del sistema fotovoltaico.
- Especificación de los parámetros de los modelos de los componentes del sistema, y modificación de los propios de la simulación.
- Simulación, con resolución horaria, de la producción de la instalación fotovoltaica.
- Exportación y representación gráfica de los resultados, así como de las curvas *I-V* de las distintas partes del campo fotovoltaico (módulos, cadenas y cajas de conexiones).

En lo sucesivo, se hará referencia a esta librería como *PV-AMP* (*Photovoltaic Analysis and Modeling Program*).

1.3 Herramientas

La librería *PV-AMP* está escrita en el lenguaje de programación *Python*, versión 3.11.4. Para su desarrollo, se ha hecho uso tanto de las librerías estándar integradas en esta versión, como de las siguientes librerías externas:

- *matplotlib* (v. 3.7.2)	- *pandas* (v. 2.0.3)	- *scipy* (v. 1.11.1)
- *numpy* (v. 1.25.2)	- *pvlib* (v. 0.10.1)	

1.4 Estructura del TFM

- En el Capítulo 2, se hará una presentación de la librería *PV-AMP*, y se detallarán las metodologías empleadas para modelar, implementar y simular los diseños fotovoltaicos:
  - Obtención de datos meteorológicos
  - Modelado de las superficies y su implementación.
  - Modelado de la distribución heterogénea de la radiación sobre las superficies.
  - Modelado de los principales componentes del sistema fotovoltaico y su implementación.
  - Modelado de las pérdidas y simulación del sistema

- En el Capítulo 3, se presentarán una serie de casos de estudio, a través de los cuales se pretende poner de manifiesto las posibilidades utilidades de la librería.

- En el Capítulo 4, se expondrá, a modo de conclusión, una reflexión sobre los posibles trabajos futuros que pueden abordarse a través de la librería, y sobre las funcionalidades con las que se puede seguir expandiendo.

- En el Anexo A, se exponen los algoritmos empleados para cuantificar la atenuación de la radiación que experimentan las superficies, así como los métodos numéricos principales empleados en el cálculo de la curva *I-V* de la célula fotovoltaica.

- En el Anexo B, se describen los comandos principales de uso de la librería.

2 ESPECIFICACIONES DE LA LIBRERÍA

2.1 Descripción

La librería que se desarrolla en este TFM, denominada *PV-AMP*, está escrita en lenguaje de programación *Python*, y tiene por objetivo el análisis y simulación de sistemas fotovoltaicos conectados a red. Esta librería incorpora las herramientas necesarias para la implementación de los diseños, a través de las siguientes especificaciones:

- Datos meteorológicos del enclave.
- Disposición de las superficies en el espacio.
- Componentes y topología eléctrica del campo fotovoltaico.

En lo referente a la simulación de los sistemas implementados, *PV-AMP* incorpora las herramientas necesarias para la realización de dos tipos de cálculos:

- **Balances de energía radiante**: determinan la distribución de la irradiación a lo largo de las superficies receptoras, con resolución horaria. Estos cálculos tienen en cuenta la radiación reflejada y la atenuación de la radiación debida a la presencia de obstáculos.
- **Cálculos eléctricos**: En función de las condiciones de temperatura ambiente y de la distribución de la irradiancia sobre cada módulo, junto con topología eléctrica del sistema y las propiedades de sus componentes, determinan la producción eléctrica del sistema, así como las curvas *I-V* de las distintas partes que integran al generador fotovoltaico.

Para la gestión de los resultados de la simulación, *PV-AMP* posee las siguientes funcionalidades:

- Confección de gráficas y animaciones de la distribución de la irradiación en las superficies.
- Representación gráfica de las curvas *I-V* y *P-V* de las distintas partes del generador fotovoltaico.
- Exportación y representación gráfica de los resultados del balance de energía, en cada componente del sistema.

En los siguientes apartados de este capítulo, se describirán los *objetos y métodos*[1] que incorpora la librería para implementar, simular y analizar los diseños fotovoltaicos.

[1] En la *programación orientada a objetos*, un objeto es un tipo de dato estructurado que posee una serie de variables características (*atributos*), y unas funciones que permiten operar sobre ellos (*métodos*).

2.2 Nociones básicas de uso

La implementación de los diseños a través de la librería se realizará a través de la siguiente secuencia de pasos:

1) Importación de los datos meteorológicos del enclave de la instalación.
2) Especificación de la geometría y distribución espacial de los receptores y obstáculos.
3) Especificación de la topología y parámetros eléctricos del resto de componentes del sistema.

Los dos primeros pasos habilitan el cálculo del balance de energía radiante entre las superficies, que ofrecerá como principal resultado la distribución de la radiación que incide sobre cada módulo fotovoltaico. El tercero permitirá determinar la producción energética del sistema, así como el desglose de pérdidas en los distintos componentes del mismo.

Para cada uno de los pasos anteriores, la librería implementa una serie de *objetos*, mediante los cuales se representarán los distintos elementos que definen tanto al sistema como a su entorno. En concreto, para los pasos uno y dos, se representarán a través de objetos los siguientes agentes:

- Recurso solar.
- Superficies receptoras y obstáculos.
- Escenario en el que se distribuyen las superficies.

Para el tercer paso, se cuentan con objetos que representarán los siguientes componentes de la instalación:

- Módulos fotovoltaicos.
- Cadenas de módulos o *strings*.
- Cajas de conexiones.
- Seguidores del punto de máxima potencia.
- Inversores de red.

La confección de un determinado diseño se realizará a través de la especificación de los distintos componentes de la instalación, en el siguiente orden:

Módulos → Strings → Cajas de conexiones → MPPT → Inversores

En cada etapa, se definen únicamente aquellos elementos que sean representativos de un determinado conjunto. Posteriormente, un componente de una etapa superior se confeccionará a través de la especificación de los distintos tipos de objetos representativos que lo integran, y sus respectivas cantidades. Por ejemplo, si se prevé que en un determinado diseño existen dos módulos de tipos *A* y *B* que, por sus condiciones de irradiancia y temperatura en cualquier instante

de tiempo, son representativos del resto, sólo se implementarán éstos. Los *strings*, por tanto, se confeccionarán como combinaciones de las cantidades de módulos de tipos *A* y *B* que los conforman. Del mismo modo, la definición de estos *strings* se puede restringir a aquellos que se consideren representativos de los restantes.

En última instancia, se podrán definir todos los elementos de forma individual. Sin embargo, esta estrategia de implementación permitirá acotar los cálculos a los elementos representativos, cuyos resultados serán posteriormente escalados.

En la Figura 1 se muestra un diagrama de bloques en el que se ilustra las distintas etapas del proceso de implementación del diseño, los objetos intervinientes y sus interdependencias:

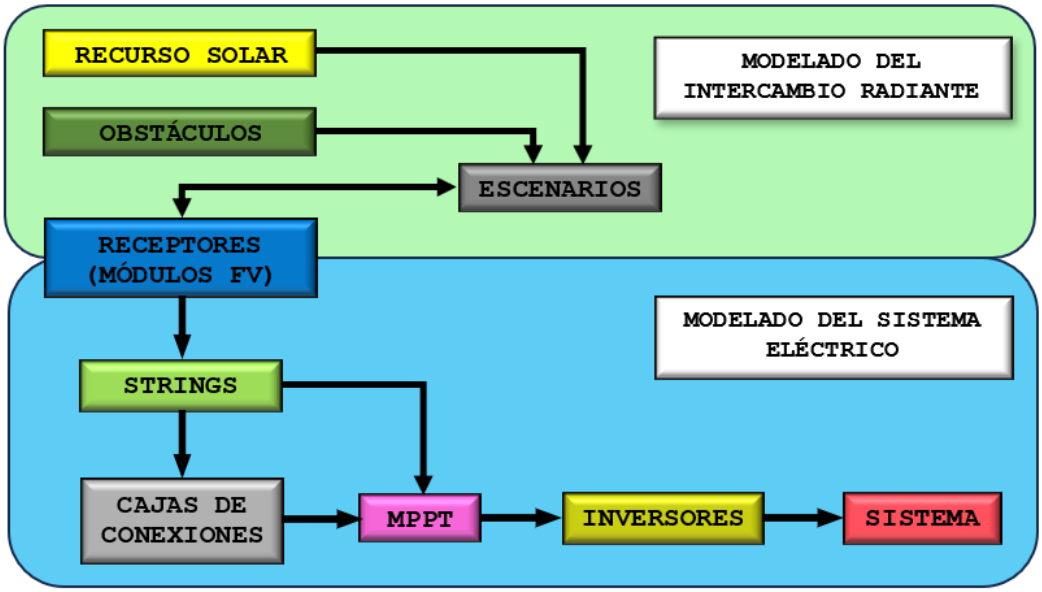

Figura 1. Diagrama de bloques de la librería

2.3 Modelado e implementación del balance radiante

Para modelar la distribución de los módulos fotovoltaicos y el entorno de la instalación, se hace uso de dos tipos de objetos: *obstáculos* y *receptores*. La distribución particular de estos objetos en el espacio, junto con el *recurso solar*, definen el *escenario* sobre el que se realiza el cálculo del balance radiante. En los siguientes apartados se describirán las metodologías empleadas para modelar e implementar de dicho escenario.

2.3.1 Importación de los datos meteorológicos y de las coordenadas solares

La obtención de los datos meteorológicos y de las coordenadas solares del emplazamiento se realizará a través del objeto *solarResource*. Esta obtención podrá realizarse de dos formas

distintas: a través de la especificación de las coordenadas geográficas del emplazamiento de la instalación, o bien a partir de la lectura de un fichero *.csv*. Los comandos de este objeto pueden consultarse en el Recurso solar.

Cuando se utiliza como fuente de los datos un fichero *.csv*, éste deberá adecuarse a lo especificado en el B.2.1 Inicialización ***a través de un fichero externo***. En caso de especificarse las coordenadas geográficas, *PV-AMP* el procede como sigue:

- Para la adquisición de los datos meteorológicos horarios, se realiza una consulta a la base de datos *PVGIS-SARAH2*, a través de la *API* de *PVGIS*. Los datos que devuelve son los propios de un *año típico meteorológico*, calculado a través de los datos horarios del periodo comprendido entre 2005 y 2020 [2].
- Las coordenadas solares se calcularán a través del módulo *solpos* de la librería *pvlib* [1]. En concreto, este módulo emplea el algoritmo *SPA* (*Solar Position Algorithm*) desarrollado por el *NREL* (*National Renewable Energy Laboratory*) [3].

En la Figura 2 se muestra el diagrama de flujo del proceso de inicialización del objeto *solarResource*, que finaliza con la confección de una tabla interna en la que se copiarán los datos horarios meteorológicos y de las coordenadas solares.

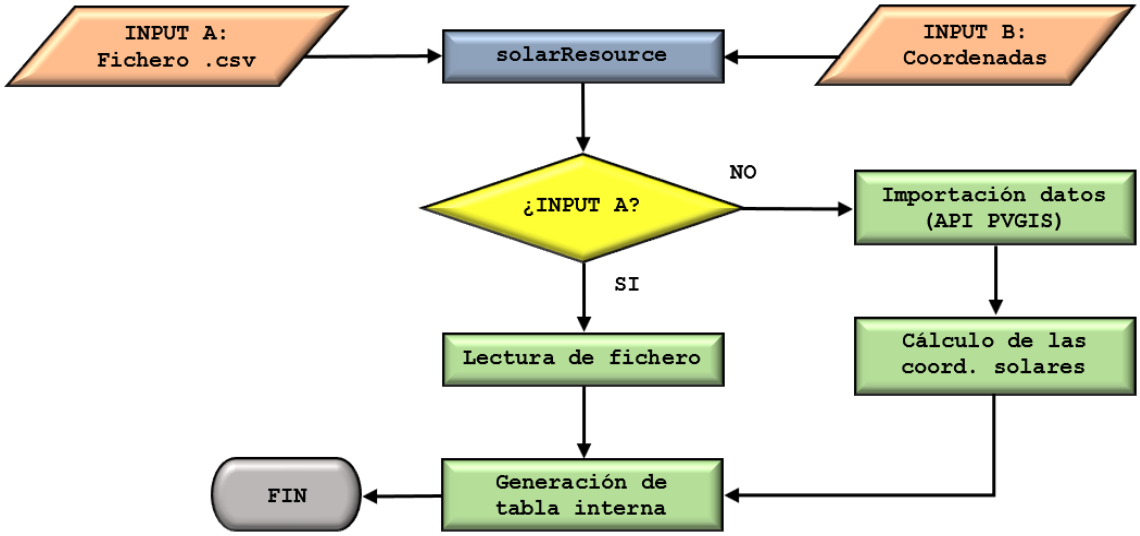

Figura 2. Inicialización del objeto *solarResource*

2.3.2 Modelado de las superficies

2.3.2.1 TIPOS DE SUPERFICIES

Las distintas superficies presentes en un determinado escenario se pueden clasificar atendiendo a la forma en la que participan en el balance radiante:

- ***Obstáculos***: su función será simular aquellos obstáculos cercanos de los que no resulta de interés adquirir datos de la irradiación que reciben, ni de la que reflejan, por considerarse esta última despreciable. A efectos de cálculo, sólo serán utilizados para cuantificar la atenuación de las distintas componentes de la radiación que llegan a los receptores.
- ***Receptores***: superficies bifaciales que participan en el balance radiante, reflejando la radiación incidente sobre ellas y obstaculizando la propia que llega al resto de receptores.

Estas superficies se modelan como un conjunto de polígonos que comparten un mismo plano. En este aspecto, las superficies pueden entenderse como *mosaicos*, y los polígonos asociados como *baldosas*. En general, las baldosas de un mosaico no tienen por qué ser iguales, ni tampoco contiguas, dando así lugar a las siguientes posibilidades:

- ***Mosaico genérico***: aquel cuyas baldosas pueden responder a cualquier forma poligonal y distribución en el plano.
- ***Mosaico convexo***: aquel cuyas baldosas lo adoquinan de manera en la que el mosaico adquiere una forma de polígono convexo, caracterizado por ser todos sus ángulos interiores inferior a 180 grados, y con un único contorno.
- ***Mosaico rectangular***: caso particular del mosaico convexo, en el que sus baldosas son rectangulares, generándose así una cuadrícula rectangular.

En la Figura 3, los dos primeros mosaicos de la parte superior izquierda poseen dos contornos, mientras que el tercero posee un ángulo interior superior a 180º. Los tres mosaicos inferiores cumplen con el criterio de convexidad, siendo el que se sitúa más a la derecha el único que puede considerarse rectangular.

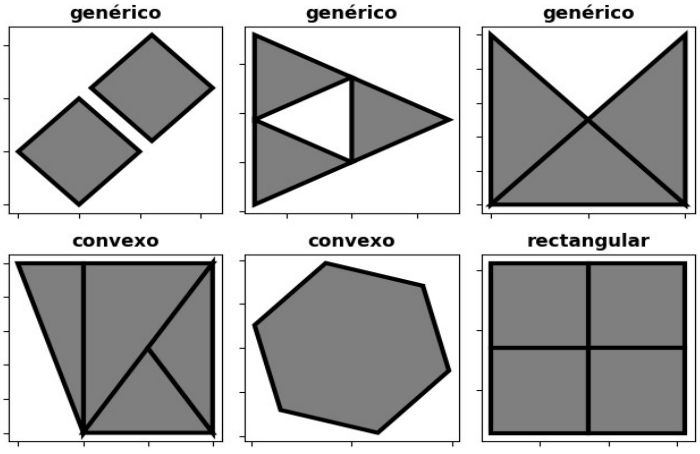

Figura 2. Ejemplos de cada tipo de mosaico

Como se verá en los siguientes apartados, la finalidad principal del modelado de las superficies en términos de mosaicos de distintos tipos es múltiple:

- Discretizar las superficies receptoras (mosaicos) en elementos discretos (baldosas), que se considerarán uniformemente irradiados. En este aspecto, las baldosas constituyen las unidades básicas sobre las que se realiza el balance de energía radiante.
- Flexibilizar la implementación de superficies.
- Simplificación de los algoritmos empleados para calcular la atenuación de la radiación sobre las superficies por la presencia de obstáculos.

2.3.2.2 GENERACIÓN DE PUNTOS INTERIORES A UNA SUPERFICIE

El siguiente nivel de modelado de las superficies consiste en definir una serie de parámetros que permitan generar una serie de puntos interiores a ellas. Dicha generación de puntos será empleada para los siguientes propósitos:

- Para el cálculo de los factores de visión de una baldosa con respecto a los elementos de su entorno (cielo, terreno y otras baldosas), se generará una nube de puntos interiores, que se designará en lo sucesivo como *"mallado de nodos difusos"*.
- Para el cálculo del factor de obstrucción a la radiación directa que experimenta una baldosa, se generará un *"mallado de nodos directos"*.
- En el caso de los mosaicos convexos no rectangulares, la generación de los puntos interiores será utilizada para dividirlo de forma automática en baldosas.

Para generar el conjunto de puntos interiores a un polígono, se parte del Algoritmo A.1. Este algoritmo determina si un punto *P* es interior a un polígono *S*, a través del empleo de un punto exterior *E* al mismo. Si el segmento *PE* es secante un número impar de veces al polígono *S*, entonces dicho punto *P* es interior a *S*.

Posteriormente, se determinan los vértices del rectángulo de menor tamaño que contiene al polígono *S*, cuyos lados son paralelos a los ejes *X* e *Y* del plano en el que se definen sus vértices. Este rectángulo se dividirá en *nx* columnas y *ny* filas, formando un entramado rectangular. Si el centroide *C* de una celda es interior al polígono (Algoritmo A.1), entonces *C* pasa a formar parte del conjunto de puntos interiores de *S*. Este mecanismo se implementa en el Algoritmo A.2.

En la Figura 4 se ejemplifica la generación de puntos interiores a un polígono, en gris oscuro, a través del Algoritmo A.2, con $\{nx = 6, ny = 8\}$. En color gris claro, se representa el rectángulo mínimo que contiene al polígono, dividido en celdas a través de las líneas discontinuas. Los centroides exteriores se han representado en color negro, y los interiores en color verde.

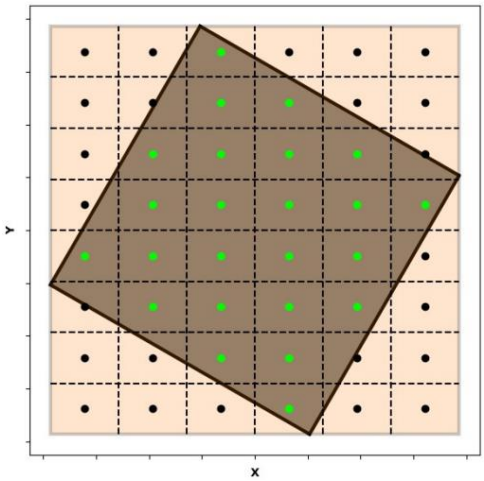

Figura 4. Ejemplo de generación de puntos interiores a un polígono mediante el Algoritmo A.2

A su vez, como alternativa a la especificación de cada uno de los vértices de las baldosas en las que se desea dividir un mosaico convexo no rectangular, el conjunto de puntos interiores determinados por el Algoritmo A.2, incrementado con la adición de los vértices del mosaico, pueden ser empleados para realizar una *triangulación de Delaunay* [4], en la que se reagrupan los puntos del conjunto de entrada para dar lugar a un listado con los vértices de los triángulos con los cuales se puede adoquinar el mosaico. Esta triangulación queda implementada en *PV-AMP* a través de la librería embebida *scipy* [5].

En la Figura 5 se muestra un ejemplo de generación de baldosas para dos mosaicos diferentes (polígono de cuatro lados en la parte superior, y de cuarenta lados en la inferior), por medio del mecanismo descrito en el párrafo anterior. De izquierda a derecha, puede observarse cómo evolucionan las baldosas generadas según se incrementan los parámetros *nx* y *ny*.

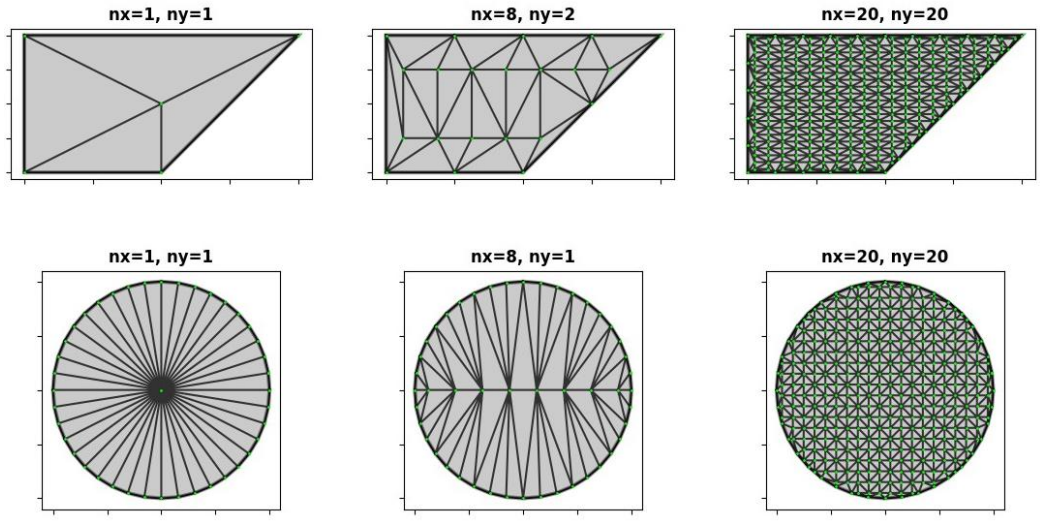

Figura 5. Baldosas generadas a través de la triangulación de Delaunay

En el caso de los mosaicos rectangulares, la división en baldosas se realiza aprovechando las divisiones en *nx* columnas y *ny* filas que realiza el Algoritmo A.2, ya que el rectángulo mínimo generado coincide con el mosaico.

Como corolario de lo expuesto, se destaca lo siguiente:

- La generación de puntos interiores a un polígono (mosaico convexo o baldosa) se abordará a través de la especificación de los parámetros *nx* y *ny*.
- En el caso de que la generación de estos puntos responda a un *mallado de nodos difusos* de las baldosas, los parámetros *nx* y *ny* serán designados, respectivamente, como *nxd* y *nyd*; en el caso del *mallado de nodos directos*, como *nxb* y *nyb*.
- Cuando se aborda la generación automática de las baldosas de un mosaico convexo a través de los parámetros *nx* y *ny*, estos se designarán como *nxt* y *nyt*.

En la Figura 6 se muestra un ejemplo de los sucesivos niveles de modelado de un mosaico rectangular, a través de la especificación de los parámetros {*nxt* = 2, *nyt* = 2}, {*nxb* = 4, *nyb* = 4} y {*nxd* = 3, *nyd* = 5}.

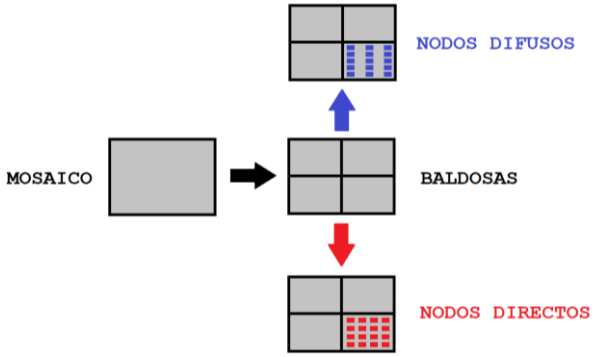

Figura 6. Ejemplo de los sucesivos niveles de modelado de un mosaico rectangular

2.3.2.3 SISTEMAS DE REFERENCIA Y MOVIMIENTOS

Cada mosaico S_i tiene su propio sistema de referencia *local* bidimensional, en el cual se definen inicialmente los vértices del mismo:

$$S_{\mathrm{i}}^{0} := \left\{ \left(v_{x1}, v_{y1}\right)^{0}, \ldots, \left(v_{xk}, v_{yk}\right)^{0}, \ldots, \left(v_{xN}, v_{yN}\right)^{0} \right\} \tag{2.1}$$

A su vez, cada sistema de referencia local queda definido en un sistema de referencia *global* único, a través de una base ortonormal $M_i = \{\overrightarrow{X_\iota}, \overrightarrow{Y_\iota}, \overrightarrow{Z_\iota}\}$ y un punto de origen $\overrightarrow{0_\iota}$. El mosaico queda contenido en el plano formado por las componentes $\{\overrightarrow{X_\iota}, \overrightarrow{Y_\iota}\}$, utilizándose la componente $\overrightarrow{Z_\iota}$ como la normal de la cara frontal del mosaico, y la de sus baldosas. En la Figura 7 se muestran dos mosaicos (verde y azul) y los elementos de sus respectivos sistemas de referencia.

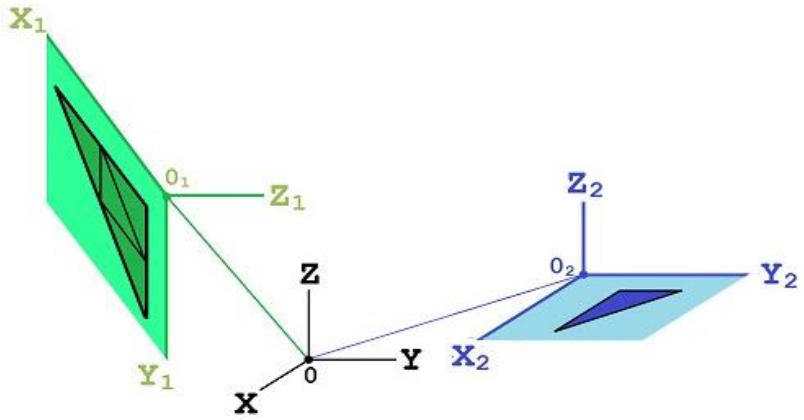

Figura 3. Ejemplo de dos sistemas de referencia locales, definidos por sus bases y puntos de origen

Al comienzo, todos los sistemas de referencia se inicializan como sigue:

$$\mathrm{M}_{\mathrm{i}}^{0} := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$\overrightarrow{O_l^0} := (0,0,0) \tag{2.3}$$

Los movimientos de traslación y rotación de las superficies se realizarán de forma secuencial, pudiendo ser definidos tanto en sus respectivas referencias locales como en la global. En particular, cuando se realiza una traslación en la referencia global, de cantidad $\overrightarrow{\Delta P_{3D}}$ en la etapa *k*, se refleja través de una modificación del punto de origen:

$$\overrightarrow{O_l^k} = \overrightarrow{O_l^{k-1}} + \overrightarrow{\Delta P_{3D}} \tag{2.4}$$

Los movimientos, si son especificados en el sistema de referencia local de la superficie, se reflejan a través de la modificación de la posición de sus vértices:

$$S_{\mathrm{i}}^{\mathrm{k}} = \left\{ \left(v_{x1}, v_{y1}\right)^{k-1} + \overrightarrow{\Delta P_{2D}}, \ldots, \left(v_{xN}, v_{yN}\right)^{k-1} + \overrightarrow{\Delta P_{2D}} \right\} \tag{2.5}$$

Las rotaciones en el sistema de referencia global se especificarán a través de un ángulo de giro θ, y de un eje unitario $\vec{u} = (u_x, u_y, u_z)$ que pasa por el origen, en torno al que se rota. Esta rotación se reflejará a través de la modificación de la base y del origen del sistema de referencia local, por medio de una matriz de rotación $R(\theta, \vec{u})$, bajo la regla de la mano derecha:

$$R(\theta,\vec{u}) := \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) - u_y\sin\theta \\ u_x u_y(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_z u_y(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \tag{2.6}$$

$$\overrightarrow{O_l^k} = \overrightarrow{O_l^{k-1}} \cdot (R(\theta,\vec{u}))^T \tag{2.7}$$

$$\mathrm{M}_{\mathrm{i}}^{k} = R(\theta, \vec{u}) \cdot \mathrm{M}_{\mathrm{i}}^{k-1} \tag{2.8}$$

Las rotaciones definidas en el sistema de referencia local se ejecutarán rotando las componentes $\overrightarrow{X_\iota}$ e $\overrightarrow{Y_\iota}$ de su base en torno a la componente $\overrightarrow{Z_\iota}$, en una cantidad θ:

$$\mathrm{M}_{\mathrm{i}}^{k} = R(\theta, \overrightarrow{Z_\iota}) \cdot \mathrm{M}_{\mathrm{i}}^{k-1} \tag{2.9}$$

Finalmente, para expresar un punto $\overrightarrow{P_{l,\iota}}$ dado en la referencia local de la superficie, en la referencia global, y viceversa, se hará uso de (2.10) y (2.11), respectivamente.

$$\overrightarrow{P_g} = \overrightarrow{P_{l,\iota}} \cdot \left(\mathrm{M}_{\mathrm{i}}^{k}\right)^{T} + \overrightarrow{O_\iota^k} \tag{2.10}$$

$$\overrightarrow{P_{l,\iota}} = \left(\overrightarrow{P_g} - \overrightarrow{O_\iota^k}\right) \cdot \left(\left(\mathrm{M}_{\mathrm{i}}^{k}\right)^{T}\right)^{-1} \tag{2.11}$$

2.3.3 Implementación de las superficies

En la Tabla 1 se resumen los objetos implementados en la librería para modelar las distintas superficies, clasificados según los aspectos comentados anteriormente:

Tabla 1. Resumen de los objetos implementados en PV-AMP para modelar las superficies

	Obstáculo	**Receptor**
Mosaico genérico	*shadingMosaic*	*receivingMosaic*
Mosaico convexo	-	*receivingConvex*
Mosaico rectangular	*shadingRectangle*	*receivingRectangle*

Se hace notar que no existe un objeto específico para el caso de un obstáculo convexo, ya que no se necesita subdividir en partes más pequeñas. Se implementa como un mosaico con una única baldosa, a través de la especificación de sus vértices.

Los datos de entrada para inicializar cada objeto son los siguientes:

- ***shadingMosaic***:
    - Listado con los vértices de las baldosas constituyentes.
- ***shadingRectangle***:
    - Límites inferiores y superiores de sus lados, paralelos a los ejes *X* e *Y* de su sistema de referencia local → [x_{min}, x_{max}], [y_{min}, y_{max}].
- ***receivingMosaic***:

- Listado con los vértices de las baldosas constituyentes, junto con la especificación, por cada baldosa, de los parámetros {*nxb*, *nyb*} y {*nxd*, *nyd*} para el mallado de los nodos directos y difusos, respectivamente.
- Reflectividades de la caras frontal y trasera.

- ***receivingConvex***:
  - Listado con vértices del mosaico.
  - Parámetros {*nxt*, *nyt*} para la generación de las baldosas, y {*nxb*, *nyb*} y {*nxd*, *nyd*} para el mallado de los nodos directos y difusos, respectivamente, que se aplicarán por igual a todas las baldosas generadas.
  - Reflectividades de la caras frontal y trasera.
- ***receivingRectangle***:
  - Límites inferiores y superiores de sus lados, paralelos a los ejes *X* e *Y* de su sistema de referencia local → [x_{min}, x_{max}], [y_{min}, y_{max}].
  - Parámetros {*nxt*, *nyt*} para la generación de las baldosas, y {*nxb*, *nyb*} y {*nxd*, *nyd*} para el mallado de los nodos directos y difusos, respectivamente, que se aplicarán por igual a todas las baldosas generadas.
  - Reflectividades de la caras frontal y trasera.

En la Figura 8 se muestra el diagrama de flujo del proceso de inicialización de los distintos mosaicos que implementa la librería. Los comandos específicos a emplear pueden consultarse en los anexos B.3.1 a B.3.3.

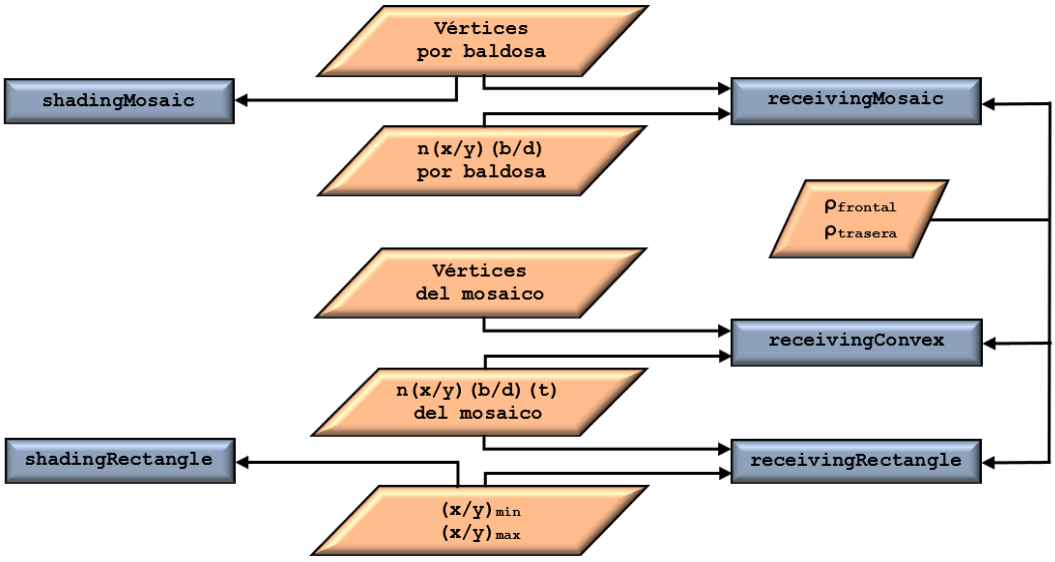

Figura 8. Diagrama de flujo de la inicialización de los mosaicos

Tras la inicialización de los mosaicos, se especificarán sus respectivas posiciones a través de una secuencia de movimientos, descritos por las ecuaciones del aptdo. 2.3.2.3, y ejecutados a través de los comandos especificados en el Anexo B.3.4. Una vez han alcanzado sus posiciones finales, se finaliza con la generación de las baldosas (en caso de receptores convexos) y del mallado de nodos. En la Figura 9 se muestra el diagrama de flujo asociado este proceso.

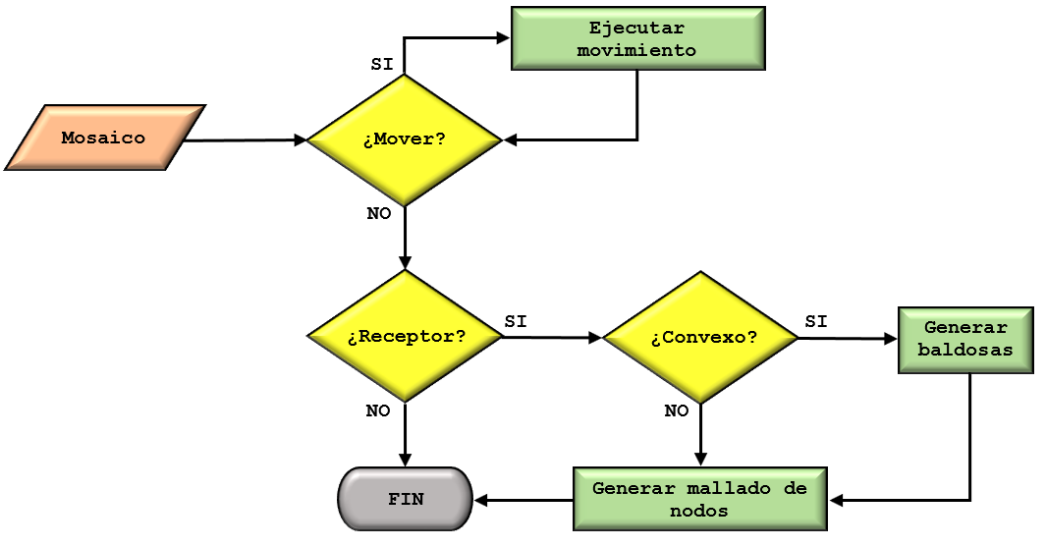

Figura 9. Diagrama de flujo de la implementación de los mosaicos

2.3.4 Modelado del balance radiante

El modelo de balance radiante implementado en la librería se caracteriza por la capacidad de reflejar la distribución heterogénea de la radiación que llega a cada cara de una superficie, conseguida a través de la división de dicha superficie en unidades más pequeñas, referenciadas en el apartado anterior como *baldosas*, cuyas caras se consideran uniformemente irradiadas.

En general, la irradiancia que recibe una superficie *i* (correspondiente con la frontal o trasera de una baldosa) posee tres componentes principales:

- ***Irradiancia directa*** (***$G_{b,i}$***): componente de la radiación solar que no ha sido dispersada a su paso por la atmósfera, con carácter direccional.
- ***Irradiancia difusa*** (***$G_{d,i}$***): componente de la radiación solar que proviene de todo el domo celeste, como resultado de la dispersión atmosférica.
- ***Irradiancia reflejada*** (***$G_{ref,i}$***): componente de la radiación que, siendo reflejada por el terreno y superficies cercanas, alcanza a la superficie objeto.

$$G_i = G_{b,i} + G_{d,i} + G_{ref,i} \tag{2.12}$$

Para cuantificar en qué medida estas componentes se ven afectadas por la presencia de obstáculos y reflectores cercanos, se hace uso de un *factor de obstrucción*, en el caso de la componente directa, y de *factores de visión*, para las componentes difusa y reflejada. Estos factores serán caracterizados mediante una técnica de *ray casting*, que consiste en la emisión de rayos desde una serie de puntos, cuyas intersecciones con las superficies serán empleadas para extraer la información requerida para evaluar estos factores. En los siguientes apartados se detallará cómo se modelan cada una de las componentes de la radiación y los factores mentados.

2.3.4.1 CÁLCULO DE LA IRRADIANCIA DIRECTA

La irradiancia directa que recibe una superficie *i* se modelará a través de (2.13), en la que la irradiancia directa en el plano normal a la dirección de los rayos solares y la posición solar se consideran datos de entrada:

$$G_{b,i} = \max\{0, \cos(\theta_i)\}(1 - \Gamma_i)G_{b,n} \tag{2.13}$$

$$\cos(\theta_i) = \begin{cases} \overrightarrow{Z_\iota} \cdot \overrightarrow{r_s} \rightarrow \text{cara frontal} \\ -\overrightarrow{Z_\iota} \cdot \overrightarrow{r_s} \rightarrow \text{cara trasera} \end{cases} ;\ \theta_i \in [0, \pi] \tag{2.13a}$$

Siendo:

- $\boldsymbol{G_{b,n}}$: irradiancia directa, proveniente del disco solar, por unidad de superficie normal a la dirección de los rayos solares.
- $\boldsymbol{\theta_i}$: ángulo formado entre la normal de la superficie *i* ($\overrightarrow{Z_\iota}$ si se trata de la cara frontal, y -$\overrightarrow{Z_\iota}$ si se trata de la trasera, siendo $\overrightarrow{Z_\iota}$ la normal de la baldosa *i*) y el vector unitario que apunta al disco solar ($\overrightarrow{r_s}$). Valores comprendidos en el intervalo [π/2, π] denotan que los rayos inciden en la cara opuesta.
- $\boldsymbol{\Gamma_i}$: factor de obstrucción de la radiación directa sobre la superficie *i*.

Para calcular la atenuación de esta componente por la presencia de obstáculos entre la superficie y el haz solar, se parte de un *mallado de nodos directos* preliminar (ver apdo. 2.3.2.2). El factor de obstrucción se corresponderá con la fracción de nodos que resulten sombreados con respecto al total:

$$\Gamma_i = \frac{\sum_{k=1}^{N_b} X_k}{N_b} \tag{2.14}$$

Donde:

- $\boldsymbol{X_k}$: si el nodo *k* queda sombreado, valdrá uno; en otro caso, valdrá cero.
- $\boldsymbol{N_b}$: Número de nodos del mallado.

La estrategia para determinar si un nodo *P* de una superficie queda sombreado, se basa en trazar una semirrecta que, con origen en *P*, se propague hacia el disco solar. Posteriormente, se evalúa si la semirrecta intercepta a alguna superficie del entorno, calculando el punto de intersección *I* con el plano que la contiene, y determinando a través del Algoritmo A.1, si *I* es interior a dicha superficie. Si se da algún caso positivo, entonces *P* queda sombreado. En el Algoritmo A.3 se expone el procedimiento para cuantificar el factor de obstrucción de una de las caras de la baldosa *i* debido a la presencia de un conjunto *S* de baldosas.

2.3.4.2 CÁLCULO DE LA IRRADIANCIA DIFUSA

Son múltiples los modelos propuestos para la caracterización de la irradiancia difusa en el plano inclinado, a partir de la propia sobre el plano horizontal. Estos modelos se diferencian por la forma en la que se caracteriza la distribución de la irradiancia difusa en el domo celeste.

El más básico de ellos es el modelo isótropo propuesto por Liu y Jordan [6], y es el que se implementa en la librería. Este modelo considera que la distribución de la intensidad de la radiación es uniforme, por lo que no depende de las coordenadas acimutal y cenital. De este modo, la irradiancia difusa que incide sobre un plano dependerá únicamente de la medida en la que el cielo ocupe su campo de visión:

$$G_{d,i} = F_{i \to s} G_{d,h} \tag{2.15}$$

Siendo:

- $\boldsymbol{G_{d,i}}$: irradiancia difusa del cielo sobre la superficie *i*.
- $\boldsymbol{F_{i \to s}}$: factor de visión de la superficie *i* con respecto al cielo.
- $\boldsymbol{G_{d,h}}$: irradiancia difusa del cielo sobre el plano horizontal.

A su vez, la hipótesis de isotropía permite tener en cuenta la eventual obstrucción de esta componente sin más que tener en cuenta la distribución de los obstáculos cercanos a la hora de computar el factor de visión del receptor con respecto al cielo.

No obstante, existen regiones en el cielo con mayor irradiancia difusa: en especial, la región circunsolar, por lo que la irradiancia difusa dependerá de en qué medida estas regiones ocupan el campo de visión del receptor [7]. Esta heterogeneidad se materializa en una subestimación de la irradiancia difusa sobre un receptor inclinado bajo la hipótesis isotrópica, sobre todo en días claros [8]. Es por ello por lo que la literatura especializada cuenta con una variedad de modelos que tratan de reflejar la anisotropía señalada [9] [10] [11].

Aunque existen modelos preferentes a la hora de abordar el cálculo de la irradiancia difusa [8] [12], como los propuestos por Pérez et al. [11] y Hay y Davies [9], cabe señalar que no son

universales, pues el desempeño de los distintos modelos depende en buena medida del entorno local y de las condiciones de instalación del receptor [8]. A su vez, la corrección de los modelos anisótropos para tener en cuenta la atenuación por la presencia de obstáculos amerita su estudio particular, que se extiende más allá del alcance del presente TFM.

2.3.4.3 CÁLCULO DE LA IRRADIANCIA REFLEJADA

La metodología habitual para caracterizar la componente de la irradiancia reflejada por el entorno consiste en suponer su origen en el plano horizontal, el cual actúa como un reflector difuso ideal, y cuya reflectividad depende del tipo de terreno y la época del año [8] [13].

No obstante, con objeto de considerar el efecto de las superficies cercanas al receptor, el modelado implementado distinguirá la irradiancia que procede del terreno de aquella que es reflejada por dichas superficies. Sin embargo, se mantendrá la hipótesis de que tanto el terreno como las superficies se comportan como reflectores difusos ideales, reflejando la irradiancia que sobre ellos inciden de manera uniforme en todo su campo de visión. De este modo, el efecto de las superficies como obstáculos podrá cuantificarse a través de los factores de visión del receptor con respecto al terreno y estas superficies. Por tanto, la componente de la irradiancia reflejada sobre una superficie *i* se modelará según (2.16):

$$G_{ref,i} = \rho_g F_{i\to g} G_{g,h} + \sum_{j=1}^{N_s} \rho_j F_{i\to j} G_j \qquad (2.16)$$

Siendo:

- $\boldsymbol{G_{ref,i}}$: componente de la radiación reflejada sobre la superficie *i*.
- $\boldsymbol{\rho_g}$: reflectividad del terreno.
- $\boldsymbol{F_{i\to g}}$: factor de visión de la superficie *i* respecto al terreno, con eventual obstrucción.
- $\boldsymbol{G_{g,h}}$: irradiancia global horizontal.
- $\boldsymbol{N_s}$: Número de superficies.
- $\boldsymbol{\rho_j}$: reflectividad de la superficie *j*.
- $\boldsymbol{F_{i\to j}}$: factor de visión de la superficie *i* respecto a la superficie *j*.
- $\boldsymbol{G_j}$: irradiancia sobre la superficie *j*.

2.3.4.4 CÁLCULO DE LOS FACTORES DE VISIÓN OBSTRUIDOS

Para el cálculo del factor de visión de una de las caras de una baldosa *i* con respecto a un elemento *j*, se parte de un *mallado de nodos difusos* preliminar (ver apdo. 2.3.2.2). El factor de visión de la cara estudiada con respecto al elemento *j* se calculará ponderando los propios de cada

nodo difuso *P* respecto a dicho elemento, a través de (2.17), en la que *N* representa el número de nodos difusos:

$$F_{i\to j} = \frac{\sum_{k=1}^{N_d} F_{P,k\to j}}{N_d} \tag{2.17}$$

Siendo:

- $\boldsymbol{F_{i\to j}}$: factor de visión de una de las caras de la baldosa *i* respecto al elemento *j*.
- $\boldsymbol{F_{P,k\to j}}$: factor de visión del nodo P_k respecto al elemento *j*.
- $\boldsymbol{N_d}$: Número de nodos difusos de la baldosa *i*.

El método empleado para calcular el factor de visión de un nodo *P* con respecto a su entorno se expone en el Algoritmo A.4, el cual está basado en el propuesto por Sönmez et al. [14], que parte de *lanzar* desde dicho nodo un conjunto de semirrectas con una distribución *cuasi-uniforme*[2] en la semiesfera vista por la cara de la baldosa *i* a la que pertenece. A cada semirrecta o *rayo* lanzado se le asigna un *peso* igual al coseno del ángulo formado entre éste y la normal de la cara desde la que se emite. Finalmente, el factor de visión del nodo *P* respecto a un elemento *j* se calcula como el cociente entre los pesos totales que llegan a *j* y todos los emitidos desde el nodo:

$$F_{P\to j} = \frac{\sum_{k=1}^{N_v} X_{k,j} \cdot \cos\theta_k}{\sum_{k=1}^{N_v} \cos\theta_k} \tag{2.18}$$

Siendo:

- $\boldsymbol{F_{P\to j}}$: factor de visión del nodo *P* respecto al elemento *j*.
- $\boldsymbol{X_{k,j}}$: si el rayo *k* intercepta a *j*, valdrá uno; en otro caso, valdrá cero.
- $\boldsymbol{\theta_k}$: Ángulo formado entre el rayo *k* y la normal de la superficie emisora.
- $\boldsymbol{N_v}$: Número de rayos lanzados desde el nodo *P*.

Para la generación de los vectores directores unitarios de los rayos lanzados, la librería emplea una distribución áurea de puntos sobre la semiesfera unitaria, centrada en el origen, para la generación de los vectores directores unitarios de cada semirrecta. El conjunto de N_v vectores bajo esta distribución se genera a como sigue:

$$\left\{\overrightarrow{v_{k,i}}\right\} = \{\, (r_k \cdot \cos\varphi_k \,, r_k \cdot \sin\varphi_k \,, z_k) : k = 1\,, 2\,, \dots\,, N_v \,\} \tag{2.19}$$

$$z_k = 1 - \frac{k}{N_v} \;;\; \varphi_k = \pi \cdot \left(1 + \sqrt{5}\right) \cdot k \;;\; r_k = \sqrt{1 - z_k^2} \tag{2.19a}$$

[2] La distribución uniforme de puntos sobre una esfera sólo es posible si éstos se corresponden con los vértices de algún sólido platónico, limitados a cinco.

El vector de una semirrecta *k*, cuando es emitido desde la cara frontal de una baldosa *i*, se expresa en el sistema de referencia global como sigue:

$$\overrightarrow{v_k} = \overrightarrow{v_{k,t}} \cdot (M_i)^T \tag{2.20}$$

Siendo M_i la base del sistema de referencia de la baldosa *i*. Si la semirrecta es emitida desde la cara posterior, bastará con cambiar de signo $\overrightarrow{v_k}$.

En la Figura 9 se muestra un ejemplo de distribución áurea de 2.000 puntos sobre una semiesfera, centrada en un nodo de una baldosa.

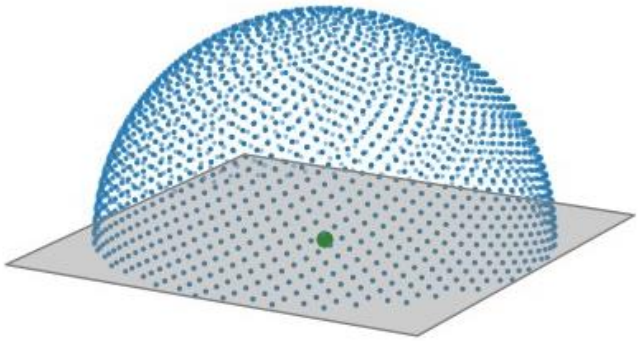

Figura 10. Distribución áurea de puntos sobre una semiesfera

2.3.4.5 RESOLUCIÓN DEL BALANCE RADIANTE

En virtud de lo expuesto hasta ahora, la irradiancia que recibe una superficie *i* puede expresarse según (2.21), en la que se han condensado todos los términos que no dependen de la irradiancia de otras superficies en el parámetro K_i:

$$G_i = K_i + \sum_{j=1}^{N_s} \rho_j F_{i\to j} G_j \tag{2.21}$$

$$K_i = \max\{0, \cos(\theta_i)\}(1 - \Gamma_i) G_{b,n} + F_{i\to s} G_{d,h} + \rho_g F_{i\to g} G_{g,h} \tag{2.21a}$$

Esta reagrupación de términos permite identificar con facilidad que la determinación de la irradiancia sobre cada superficie parte de la resolución de un sistema de N_s ecuaciones lineales, expresado en su forma matricial por (2.22), y que puede ser resuelto a través de (2.23):

$$\begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_{N_s} \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_{N_s} \end{bmatrix} + \begin{bmatrix} \rho_1 F_{1\to 1} & \rho_2 F_{1\to 2} & \cdots & \rho_{N_s} F_{1\to N_s} \\ \rho_1 F_{2\to 1} & \rho_2 F_{2\to 2} & & \rho_{N_s} F_{2\to N_s} \\ \vdots & & \ddots & \vdots \\ \rho_1 F_{N_s\to 1} & \rho_2 F_{N_s\to 2} & \cdots & \rho_{N_s} F_{N_s\to N_s} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_{N_s} \end{bmatrix} \to G = K + \rho F \cdot G \tag{2.22}$$

$$G = (I - \rho F)^{-1} \cdot K \tag{2.23}$$

No obstante, si el número de superficies N_s es elevado, la determinación de la irradiancia a través de (2.23) puede comprometer en exceso el tiempo y los recursos de memoria empleados, ya que sería necesario invertir una matriz de tamaño (N_s x N_s) en cada fecha y hora a simular. En su lugar, la resolución de este sistema se realizará a través del esquema iterativo (2.24):

$$G^{k)} = K + \rho F \cdot G^{k-1)} \;\; ; \;\; G^{0)} \coloneqq K \tag{2.24}$$

En general, un esquema iterativo $x^{k)} = K + T \cdot x^{k-1)}$ converge a la solución del sistema de ecuaciones lineales $x = K + T \cdot x$ para cualquier aproximación inicial $x^{0)}$, siempre y alguna norma matricial de T cumpla la inecuación $\|T\| < 1$ [15]. En el caso particular del esquema (2.24), la norma del máximo aplicada a la matriz ρF (2.25) puede emplearse para determinar las condiciones que han de cumplirse para satisfacer el criterio de convergencia anterior:

$$\|\rho F\|_\infty = \max_{1 \le i \le N_s} \left[\sum_{j=1}^{N_s} \left| \rho_j F_{i \to j} \right| \right] < 1 \tag{2.25}$$

Si se tienen en cuenta (2.26), el que la reflectividad máxima que presenta alguna superficie sea menor que la unidad, resulta condición suficiente para asegurar la convergencia (2.27):

$$\sum_{j=1}^{N_s} F_{i \to j} = 1 \to \sum_{j=1}^{N_s} \left| \rho_j F_{i \to j} \right| \le \rho_{max} \;\; ; \;\; \rho_{max} \coloneqq \max\{\rho_j : j = 1,2, \dots, N_s\} \tag{2.26}$$

$$\rho_{max} < 1 \to \sum_{j=1}^{N_s} \left| \rho_j F_{i \to j} \right| < 1 \tag{2.27}$$

2.3.5 Implementación del balance radiante

Para la implementación del escenario radiante, se hará uso del objeto *Scene*. A través del mismo, se realizarán, almacenarán y representarán los resultados de los cálculos de irradiancia e irradiación sobre las superficies que lo integran. Se inicializa mediante las siguientes entradas:

- Instancia del objeto *solarResource* que contiene los datos meteorológicos.
- Listado con los receptores de la escena, que participarán en el balance radiante como obstáculos y como reflectores. Se admiten únicamente instancias de los objetos *receivingMosaic*, *receivingConvex*, o *receivingRectangle*.

- Listado con las superficies que participarán únicamente en el balance radiante como obstáculos. Pueden ser instancias de cualquiera de los objetos descritos en la Tabla 1, del apartado 2.3.3.
- Valor de la reflectividad del terreno, comprendido entre 0 y 1.

En disposición de estos datos, el proceso de inicialización continúa con las siguientes operaciones:

- Cálculo de los factores de visión de cada cara de las baldosas de los receptores con respecto al cielo, terreno y al resto de caras de las baldosas de otros receptores.
- Generación de una tabla interna en la que se almacenarán los datos de irradiancia e irradiación acumulada sobre cada cara de las baldosas de los receptores, que se actualizará en cada simulación horaria realizada.
- Almacenar en cada receptor de la escena la dirección del espacio de memoria en la que se almacena el objeto *Scene* generado. De esta manera, se facilitará y automatizará el acceso a la información de la irradiancia sobre las baldosas, y de la temperatura ambiente, a través del receptor de interés.

El comando para la inicialización del objeto *Scene* puede consultarse en el Anexo B.4.1. En la Figura 11 se muestra el diagrama de flujo asociado a dicho proceso de inicialización:

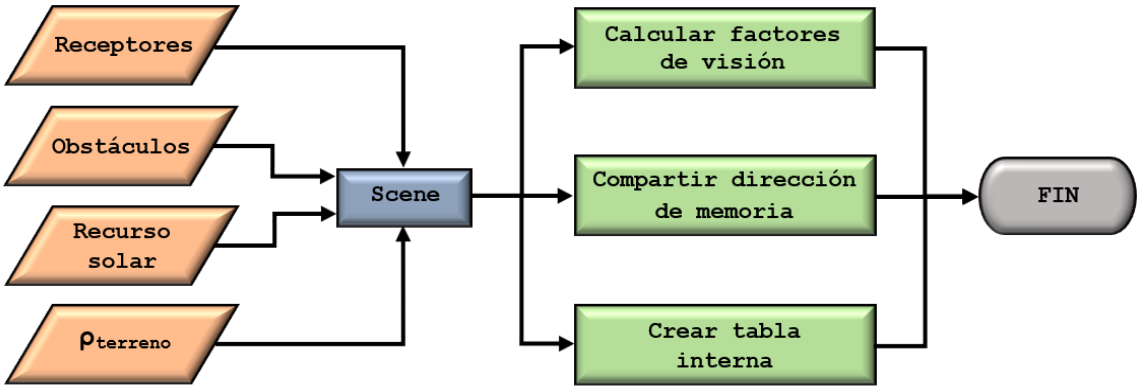

Figura 11. Diagrama de flujo de la inicialización del objeto *Scene*

2.4 Modelado e implementación de los componentes del sistema fotovoltaico

Para modelar e implementar los componentes de la instalación fotovoltaica, se hará uso de los siguientes objetos:

- Módulos fotovoltaicos.
- Cadenas de módulos conectados en serie.
- Cajas de conexiones.

- Seguidores del punto de máxima potencia.
- Inversores.

En los siguientes apartados se describirá cómo se modelan e implementan cada uno de estos objetos en la librería.

2.4.1 Modelado de la célula fotovoltaica

El punto de partida del modelado de la respuesta eléctrica del módulo fotovoltaico, en cualesquiera que sean sus condiciones de operación, parte del modelado eléctrico de la célula fotovoltaica. El modelo implementado en la librería se basa en el *circuito equivalente de un diodo*, el cual es uno de los más extendidos, debido a su simplicidad y versatilidad a la hora de simular las principales tecnologías de los módulos comerciales, desde un punto de vista práctico. Tales tecnologías son el silicio cristalino, policristalino y silicio amorfo [16] [17]. Además, este modelo permite ser particularizado a través de la información generalmente disponible en las fichas técnicas de los módulos. El circuito equivalente del modelo de un diodo, mostrado en la Figura 16, se caracteriza por los siguientes parámetros:

- $\boldsymbol{I_{ph}}$: fotocorriente.
- $\boldsymbol{I_s}$: corriente de saturación inversa del diodo.
- $\boldsymbol{a}$: producto del factor de idealidad del diodo por el potencial térmico $V_t = KT/q$, siendo K la constante de Boltzmann, T la temperatura absoluta de la célula, y q la carga del electrón.
- $\boldsymbol{R_s}$: resistencia en serie.
- $\boldsymbol{R_{sh}}$: resistencia en paralelo

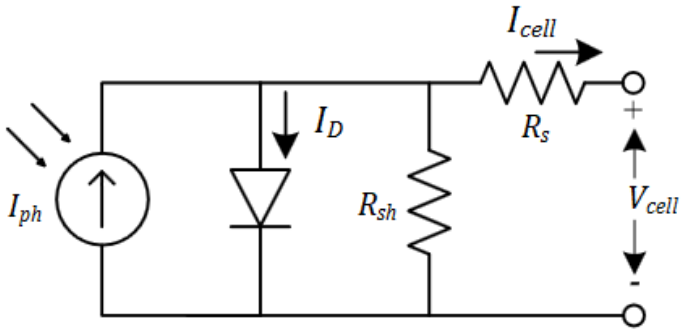

Figura 16. Circuito eléctrico equivalente de un diodo de la célula fotovoltaica

La curva *I-V* que resulta de este modelo viene dada por (2.27):

$$I_{cell} = I_{ph} - I_S\left(e^{\frac{V_{cell}+I_{cell}R_S}{a}} - 1\right) - \frac{V_{cell} + I_{cell}R_S}{R_{sh}} \qquad (2.27)$$

El carácter implícito de esta expresión impide su evaluación directa como función de la corriente o de la tensión. Aunque su resolución puede ser abordada mediante algún esquema iterativo, su empleo puede implicar problemas de convergencia y un excesivo tiempo de simulación si el número de células es elevado. Sin embargo, la aplicación de la *rama principal de la función W de Lambert* (W_0) sobre (2.27) permite expresar explícitamente la corriente como una función de la tensión y viceversa, a través de (2.28) y (2.29), respectivamente [17] [18]. El método para evaluar la función W_0 puede consultarse en el Anexo A.6.

$$I_{cell} = \frac{R_{sh}\left(I_{ph} + I_s\right) - V_{cell}}{R_s + R_{sh}} - \frac{a}{R_s} W_0\left(\frac{R_s R_{sh} I_s}{a(R_s + R_{sh})} e^{\frac{R_s R_{sh}\left(I_{ph}+I_s\right)+R_{sh}V_{cell}}{a}}\right) \tag{2.28}$$

$$V_{cell} = R_{sh}\left(I_{ph} + I_s\right) - (R_s + R_{sh})I_{cell} - aW_0\left(\frac{R_{sh} I_s}{a} e^{\frac{R_{sh}\left(I_{ph}+I_s-I\right)}{a}}\right) \tag{2.29}$$

Sin embargo, para tener en cuenta la zona de ruptura de la célula[3], este modelo será modificado según [19], en el que dicha zona es modelada mediante (2.30) a (2.33), donde V_{br}, b y m son constantes del modelo, y z_{Rmin} es la menor de las raíces reales del polinomio (2.31):

$$V_{cell} = V_{br} - R_s I_{cell} - z_{R,min} \tag{2.30}$$

$$\frac{1}{R_{sh}} z^4 + \left(I_{ph} - I - \frac{V_{br}}{R_{sh}}\right) z^3 + b V_{br}^3 z - b V_{br}^4 = 0 \tag{2.31}$$

$$z_{Rmin} = \min\left(\text{real}(z)\right) \tag{2.32}$$

$$V_{br} = -21.29\ \text{V};\ b = 0.002\ \Omega^{-1};\ m = 3 \tag{2.33}$$

Por tanto, la curva *I-V* de la célula fotovoltaica vendrá dada por (2.34):

$$V_{cell} = \begin{cases} R_{sh}\left(I_{ph} + I_s\right) - (R_s + R_{sh})I_{cell} - aW_0\left(\frac{R_{sh} I_s}{a} e^{\frac{R_{sh}\left(I_{ph}+I_s-I\right)}{a}}\right) & I \le I_{ph} \\ V_{br} - R_s I_{cell} - z_{R,min} & I > I_{ph} \end{cases} \tag{2.34}$$

[3] La operación en esta región suele ser el resultado de sombreados parciales del módulo, y está caracterizada por el paso de una elevada corriente. La poca capacidad de evacuar el calor generado se materializa en un incremento notable de la temperatura de la célula, generándose así los conocidos como "puntos calientes".

2.4.1.1 MÉTODOS DE EXTRACCIÓN DE LOS PARÁMETROS DEL MODELO DE UN DIODO

Los métodos empleados para caracterizar los cinco parámetros del modelo de célula de un diodo pueden clasificarse en los siguientes tipos [17]:

- ***Métodos iterativos***: basados en resolver el sistema de ecuaciones que resulta de particularizar la expresión (2.27) en una serie de puntos conocidos de la curva *I-V*, mediante algún método numérico iterativo, debido al carácter no lineal e implícito de las ecuaciones [20]. Aunque alcanzan una precisión elevada, tienen un elevado coste computacional y están sujetos a problemas de convergencia.
- ***Métodos no iterativos***: basados en el planteamiento de un sistema de ecuaciones que pueden resolverse explícitamente y de manera individual, en un orden concreto. Estas ecuaciones son el resultado de la aplicación de hipótesis simplificadoras y empíricas [18] [21] [22]. Presentan un menor coste computacional con respecto a los métodos iterativos, pero también una menor precisión.
- ***Métodos de ajuste de curva***: fijan los valores de los parámetros del modelo en aquellos que mejor reproducen un conjunto de puntos de la curva *I-V* [23]. Al igual que los métodos iterativos, muestran una elevada precisión, pero al precio de un elevado coste computacional.

Los métodos de extracción de parámetros implementados en la librería se limitan a los no iterativos, pues el elevado número de células que puede conllevar un diseño hacen del tiempo empleado por ellos un aspecto crucial que debe tenerse en cuenta. A su vez, estos métodos quedarán restringidos a aquellos que no precisen de más información que la ofrecida en la ficha técnica de los módulos fotovoltaicos.

En concreto, los métodos implementados son los propuestos por Cubas et al. [18], Saloux et al. [21] y Batzelis et al. [22], detallados en el Anexo A.7. La elección está basada en los resultados del estudio comparativo del desempeño de 17 métodos no iterativos frente a un millón de curvas *I-V* de módulos de distintas tecnologías, y en distintas condiciones de operación, realizado en [17], cuyas conclusiones, en relación a estos métodos, fueron las siguientes:

- ***Método de Saloux***: de los métodos analizados que no presentaron fallos por errores numéricos, es el que presentó un menor tiempo de ejecución, y menor valor de su error cuadrático medio (ECM) relativo medio.
- ***Método de Batzelis***: de los que no presentaron fallos por errores numéricos, es el que más precisión mostró en términos de ECM absoluto medio, ECM absoluto máximo y ECM relativo máximo.
- ***Método de Cubas***: de todos los métodos analizados, fue el que presentó un menor ECM absoluto máximo.

En la Tabla 2 se exponen los resultados obtenidos en [17] para los métodos implementados, en la que puede observarse que el más preciso es el de Cubas. No obstante, es el de mayor tiempo de ejecución y de posibilidad de fallo numérico. El método de Saloux es el que presenta un menor tiempo de ejecución, pero al precio de una peor precisión. El método de Batzelis encuentra una situación intermedia entre los dos anteriores, siendo el de mejor desempeño en términos globales.

Tabla 2. Métricas principales del desempeño de los métodos implementados

	ECM absoluto (A)		**ECM relativo (%)**			
Método	**Medio**	**Máximo**	**Medio**	**Máximo**	**Fallos**	**Tiempo (s)**
Saloux	0.029	1.271	0.79	25.5	0	1.2
Batzelis	0.028	0.932	0.87	18.7	0	1.5
Cubas	0.024	0.430	0.83	13	224	2.0

2.4.1.2 DETERMINACIÓN DE LA CURVA I-V EN SEGÚN LAS CONDICIONES AMBIENTALES

La metodología empleada para caracterizar la curva *I-V* de una célula fotovoltaica, en unas condiciones de operación de temperatura e irradiancia dadas, parte del siguiente conjunto de datos, generalmente disponibles en las fichas técnicas de los módulos fotovoltaicos:

- $\boldsymbol{I_{sc,STC}}$: corriente de cortocircuito, en condiciones *STC*[4].
- $\boldsymbol{V_{oc,STC}}$: tensión de circuito abierto, en condiciones *STC*.
- $\boldsymbol{I_{mp,STC}}$, $\boldsymbol{V_{mp,STC}}$, $\boldsymbol{P_{mp,STC}}$: intensidad, tensión y potencia en el punto de máxima potencia, en condiciones *STC*, respectivamente.
- $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$: coeficientes de variación con la temperatura de la corriente de cortocircuito, de la tensión de circuito abierto, y de la potencia máxima, respectivamente.
- ***NOCT***: temperatura de célula en condiciones normales de operación *NOCT*[5].
- $\boldsymbol{\varphi}$: coeficiente de bifacialidad de la potencia máxima, en el caso de que el módulo sea bifacial. Este coeficiente se define como el cociente entre la potencia máxima del módulo cuando se ilumina únicamente su cara trasera, sobre la propia cuando se ilumina únicamente su cara delantera, en condiciones *STC* en ambos casos.

A su vez, con objeto de modelar las pérdidas por suciedad (considerada uniformemente distribuida sobre el módulo, por simplicidad), se incorpora el parámetro L_{pol}, el cual incidirá en la

[4] *STC* (*Standard Test Conditions*): Temperatura de célula: 25ºC; irradiancia: 1000 W/m^2; espectro AM1.5.
[5] *NOCT* (*Normal Operating Cell Temperature*): Temperatura ambiente: 20ºC; irradiancia: 800 W/m^2; velocidad del viento: 1 m/s.

irradiancia recibida por la célula, según (2.35) para el caso de una célula monofacial, y según (2.36) para una célula bifacial:

$$G = \left(1 - L_{pol}\right) G_{front} \tag{2.35}$$

$$G = \left(1 - L_{pol}\right)\left(G_{front} + \varphi G_{rear}\right) \tag{2.36}$$

Siendo:

- ***G***: irradiancia efectiva sobre la célula.
- G_{front}: irradiancia sobre la cara frontal de la célula.
- G_{rear}: irradiancia sobre la cara trasera de la célula.

Las magnitudes de tensión, corriente y potencia de la célula serán deducidas de las propias del módulo fotovoltaico mediante (2.37). Los coeficientes de variación de los parámetros con la temperatura, la temperatura en condiciones NOCT y el coeficiente de bifacialidad se supondrán iguales a los del módulo.:

$$V_{cell} = \frac{V_{mod}}{N_S} \;\; ; \;\; I_{cell} = \frac{I_{mod}}{N_P} \;\; ; \;\; P_{cell} = \frac{P_{mod}}{N_S N_P} \tag{2.37}$$

Siendo:

- N_S: número de células conectadas en serie.
- N_P: número de cadenas de células conectadas en paralelo.

Una vez se hayan determinado los parámetros eléctricos de referencia de la célula a partir de los del módulo, se continuará evaluando éstos a las condiciones de irradiancia y temperatura ambiente bajo las que opera dicha célula, comenzando por la estimación de su temperatura mediante (2.38):

$$T_{cell}|_{G,T_{amb}} = T_{amb} + G\frac{NOCT - 20[°C]}{800[Wm^{-2}]} \tag{2.38}$$

Siendo:

- T_{amb}: temperatura ambiente [ºC].
- ***G***: irradiancia neta sobre la célula [Wm^{-2}].
- ***NOCT***: temperatura de operación de la célula en condiciones *NOCT* [ºC].

La corriente de cortocircuito y la tensión de circuito abierto se evaluarán, respectivamente, mediante (2.39) y (2.40), en la que los coeficientes de variación con la temperatura, α y β, están expresados en unidades de [1/ºC]:

$$I_{sc}|_{G,T_{cell}} = I_{sc,STC}\left(\frac{G}{1000[Wm^{-2}]}\right)\left(1+\alpha(T_{cell}-25[°C])\right) \quad (2.39)$$

$$V_{oc}|_{T_{cell}} = V_{oc,STC}\left(1+\beta(T_{cell}-25[°C])\right) \quad (2.40)$$

Puesto que los fabricantes no suelen especificar coeficientes de variación de la tensión y de la corriente en el punto de máxima potencia, estas magnitudes serán determinadas a través de (2.41), (2.42) y (2.43), que parten de las siguientes suposiciones:

- El coeficiente de variación de la tensión en el punto de máxima potencia coincide con el de circuito abierto [18].
- La potencia máxima de la célula es directamente proporcional a la irradiancia que recibe [24].

$$P_{mp}\big|_{G,T_{cell}} = P_{mp,STC}\left(\frac{G}{1000[Wm^{-2}]}\right)\left(1+\gamma(T_{cell}-25[°C])\right) \quad (2.41)$$

$$V_{mp}\big|_{T_{cell}} = V_{mp,STC}\left(1+\beta(T_{cell}-25[°C])\right) \quad (2.42)$$

$$I_{mp}\big|_{G,T_{cell}} = \frac{P_{mp}}{V_{mp}}\bigg|_{G,T_{cell}} \quad (2.43)$$

Finalmente, se evaluarán los parámetros del circuito equivalente de la célula a través de alguno de los métodos implementados, cuyos datos de partida se exponen en la Tabla 3. Se destaca que el método de Cubas precisa de un parámetro adicional: el factor de idealidad *n* de la célula, que se supondrá constante y de valor *n* = 1.1 [17] [18]. Aunque depende de la tecnología constructiva de las células [25] y de las condiciones de operación [26], su valor puede considerarse constante sin que ello incurra en pérdidas notables de precisión, pues afectará principalmente a la curvatura de la curva *I-V* [18].

Tabla 3. Datos de entrada de los métodos implementados

Método	$\mathbf{V_{oc,STC}}$ **(V)**	$\mathbf{V_{oc}}$ **(V)**	$\mathbf{I_{sc}}$ **(A)**	$\mathbf{V_{mp}}$ **(V)**	$\mathbf{I_{mp}}$ **(A)**	$\mathbf{T_{cell}}$ **(ºC)**	**α (1/ºC)**	**β (1/ºC)**	**n (-)**
Saloux		✓	✓	✓	✓				
Batzelis	✓	✓	✓	✓	✓	✓	✓	✓	
Cubas		✓	✓	✓	✓	✓			✓

2.4.2 Modelado del módulo fotovoltaico

El modelado de los módulos fotovoltaicos está orientado a poder ser realizado en base a los datos que suelen ofrecer los fabricantes en las fichas técnicas, tanto constructivos como eléctricos. En particular, los módulos se modelarán como mosaicos rectangulares (aptdo. 2.3.2.1), en cuyas baldosas se embeberán las células fotovoltaicas.

Los tipos de módulos que pueden modelarse a través de la librería son los siguientes:

- ***Módulos estándar***: basados en un conjunto de cadenas de células conectadas en serie, en las que cada cadena posee un diodo de *by-pass* en antiparalelo. Si los diodos se disponen en horizontal, sus respectivas cadenas se disponen en el espacio formando *columnas*. En la Figura 12 se muestra un ejemplo de esta tipología.

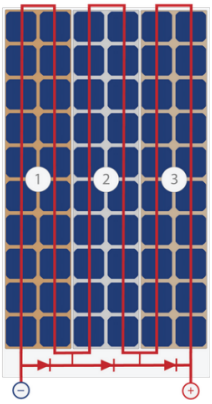

Figura 12. Ejemplo de módulo estándar. Fuente: Manual de usuario PVSyst 7

- ***Módulos de célula partida***: basados en un conjunto de cadenas de células que se conectan por pares a un mismo diodo de *by-pass* en antiparalelo, en la que los terminales de dichos conjuntos *cadena-diodo-cadena* se conectan en serie. Si los diodos se disponen horizontalmente, entonces cada par de cadenas asociadas a un mismo diodo se dispondrán en el espacio como *dos columnas apiladas*. En la Figura 13 se muestra un ejemplo de esta tipología.

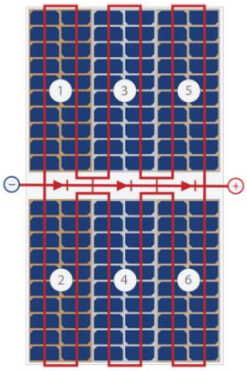

Figura 13. Ejemplo de módulo de célula partida. Fuente: Manual de usuario de PVSyst 7

2.4.2.1 MODELADO CONSTRUCTIVO

La topología eléctrica de un módulo estándar se definirá en la librería a través de la especificación de los siguientes parámetros:

- ***Número de diodos***: Este parámetro divide al módulo en tantas columnas como diodos tenga el módulo fotovoltaico, que se dispondrán de forma contigua a lo largo del eje *X* del sistema de referencia local. Se hace notar que todas las células de cada columna estarán conectadas en serie entre sí, y en antiparalelo con su diodo.
- ***Columnas por diodo***: Número de subcolumnas en las que se divide la columna asociada a un diodo.
- ***Baldosas por columna***: Número de baldosas en las que se subdivide cada una de las subcolumnas de un diodo. Las baldosas se dispondrán apiladas, es decir, contiguas a lo largo del eje *Y* del sistema de referencia local.
- ***Células por baldosa***: Número de células contenidas en cada baldosa, que pertenecerán a la misma cadena.

A la izquierda de la Figura 14 se muestra un ejemplo de cómo definir la topología un módulo estándar a través de estos cuatro parámetros, en la que la división del módulo en diodos (3) se representa en gris, las columnas por cada diodo (2) en azul, las baldosas por columna (2) en verde, y las células por baldosa (3) en rojo. A la derecha, se representa en un mismo color aquellas células que pertenecen a una misma cadena de conexiones en serie.

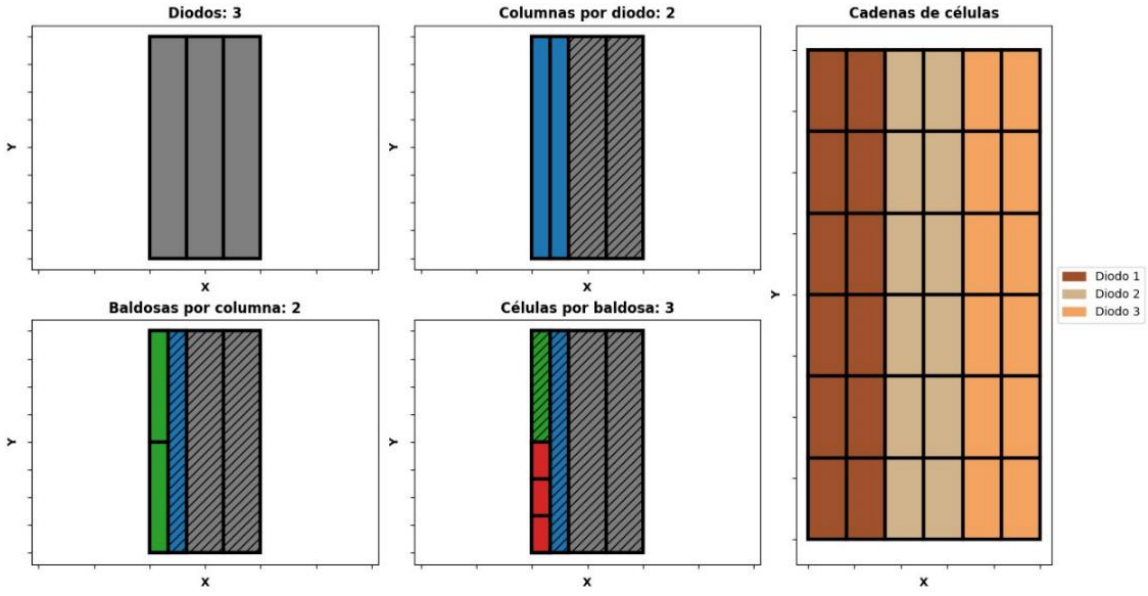

Figura 14. Ejemplo de modelado de la topología eléctrica de un módulo estándar

El modelado de la topología eléctrica de un módulo de célula partida se realizará de forma similar, con la salvedad de que los diodos de *by-pass* no dividen al módulo en columnas, sino en

en semicolumnas apiladas: una superior y otra inferior, por cada diodo. Cada semicolumna contendrá todas las células de una de las dos cadenas que se conectan en antiparalelo a su respectivo diodo.

Los parámetros a emplear para definir la topología son los siguientes:

- ***Número de diodos***: Este parámetro divide al módulo en tantas columnas como diodos tenga el módulo fotovoltaico, que se dispondrán de forma contigua a lo largo del eje *X* del sistema de referencia local. Todas las células de una de las cadenas de un diodo se ubicarán en la mitad superior de su columna, mientras que las propias de la segunda cadena lo harán en la otra mitad inferior.
- ***Columnas por diodo***: Número de subcolumnas en las que se divide la columna asociada a un diodo.
- ***Baldosas por semicolumna***: Número de baldosas en las que se subdivide cada una de las semicolumnas de un diodo. Las baldosas se dispondrán apiladas, es decir, contiguas a lo largo del eje *Y* del sistema de referencia local.
- ***Células por baldosa***: Número de células contenidas en cada baldosa, que pertenecerán a la misma cadena.

En la Figura 15 se muestra un ejemplo de cómo definir la topología de un módulo de célula partida a través de estos cuatro parámetros, en la que la división del módulo en diodos (3) se representa en gris, las columnas por cada diodo (2) en azul, las baldosas por semicolumna (2) en verde, y las células por baldosa (3) en rojo. A la derecha, se representa en un mismo color y sombreado aquellas células que pertenecen a una misma cadena de conexiones en serie, y en el mismo color las que están asociadas a un mismo diodo.

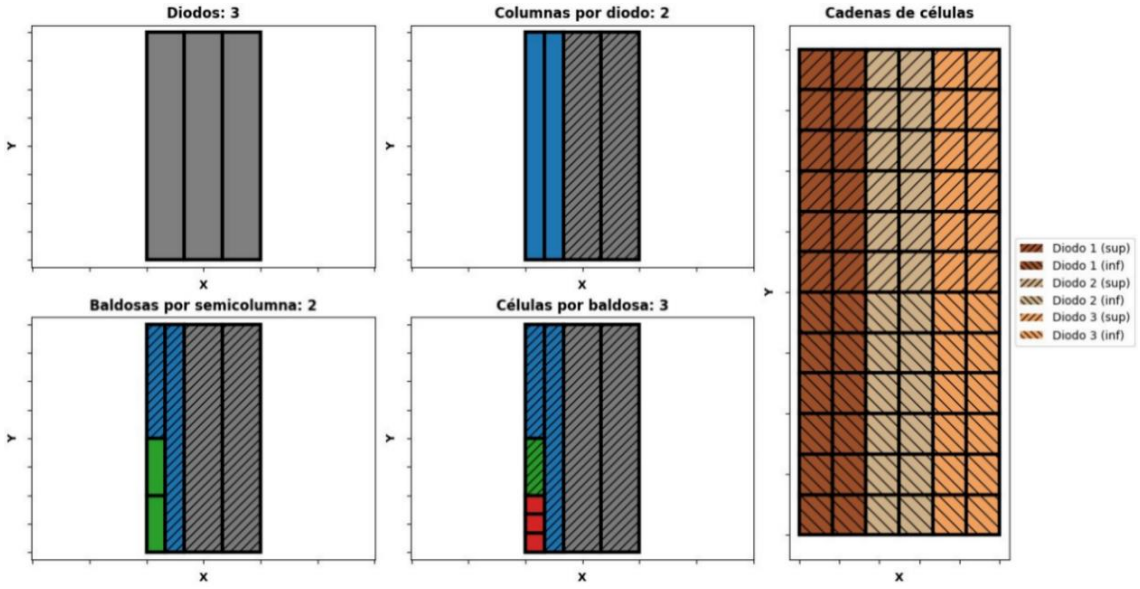

Figura 15. Ejemplo de modelado de la topología eléctrica de un módulo de célula partida

Tal y como se comentó en el aptdo. 2.3.4, las baldosas se consideran iluminadas uniformememtne. Por tanto, todas las células de una baldosa estarán igualmente iluminadas, y al quedar conectadas en serie en la misma cadena, dichas células serán indistinguibles. Este hecho permitirá controlar la cantidad de cálculos a realizar durante las simulaciones, ya que bastará con realizarlos sobre una única célula por baldosa y escalar los resultados.

2.4.2.2 MODELADO ELÉCTRICO

El modelado eléctrico del módulo fotovoltaico parte de una caracterización previa de las curvas *I-V* de sus células integrantes, cuya metodología fue explicada en el aptdo. 2.4.1. Puesto que las $N_{cell,i}$ células de una baldosa i están conectadas en serie y operan bajo las mismas condiciones de irradiancia y temperatura, la tensión $V_{tile,i}$ de este conjunto, como función de la corriente I_{cell} vendrá dada por (2.44):

$$V_{tile,i}(I_{cell}) = N_{cell,i} V_{cell,i}(I_{cell}) \tag{2.44}$$

A su vez, la tensión $V_{cs,i}$ de una cadena i que involucra a $N_{tile,i}$ baldosas vendrá dada por (2.45). En el caso de que el módulo sea estándar, estas baldosas pertenecerán a la columna del diodo de la cadena i. En el caso del módulo de célula partida, estas baldosas se corresponderán con alguna de las semicolumnas (inferior o superior) del diodo de la cadena i.

$$V_{cs,i}(I_{cell}) = \sum_{j}^{N_{tile,i}} V_{tile,j}(I_{cell}) \tag{2.45}$$

Se destaca que la evaluación de (2.45) resulta directa cuando I_{cell} actúa como parámetro. Sin embargo, la evaluación de la tensión $V_{csd,i}$ del conjunto formado por una cadena y un diodo de *by-pass* en antiparalelo, como una función de la corriente de entrada a dicho conjunto, no es directa, pues deben determinarse las corrientes que circulan por cada uno de estos elementos, satisfaciendo (2.46):

$$V_{dcs,i}(I_{dcs}) = V_{cs,i}(I_{cell}) = -V_d(I_d) \;\; ; \;\; I_{dcs} = I_{cell} + I_d \tag{2.46}$$

La corriente del diodo de *by-pass* como función de la tensión y viceversa vienen dadas, respectivamente, por (2.47) y (2.48), en la que la los valores implementados para la corriente de saturación I_{sd} y el voltaje térmico v_t son $\{I_{sd} = 1.6 \cdot 10^{-7}$ A, $v_t = 0.05$ V$\}$ [19]:

$$I_d(V_d) = I_{sd}\left(e^{\frac{V_d}{V_t}} - 1\right) \tag{2.47}$$

$$V_d(I_d) = v_t \ln\left(\frac{I_d}{I_{sd}} + 1\right) \tag{2.48}$$

Sin embargo, según se argumenta en [19], al ser despreciable la corriente del diodo cuando queda polarizado en inversa, y al ser la corriente de la cadena prácticamente constante e igual que la de cortocircuito a la tensión en la que el diodo de *by-pass* comienza a conducir la electricidad (-0.7 V, aprox.), la tensión del conjunto cadena-diodo puede evaluarse según (2.49):

$$V_{dcs,i}(I_{dcs}) = \begin{cases} V_{cs,i}(I_{dcs}) & I_{dcs} \leq I_{sc,cs,i} \\ -V_d(I_{dcs}) & I_{dcs} > I_{sc,cs,i} \end{cases} \tag{2.49}$$

La corriente de cortocircuito $I_{sc,cs,i}$ de la cadena *i* se calcula como la mínima de las células que la integran:

$$I_{sc,cs,i} = \min\{I_{sc,cell,j} : j = 1,2,\dots,N_{tile,i}\} \tag{2.50}$$

Puesto que el módulo estándar consiste en conexiones en serie de conjuntos cadena-diodo, la tensión $V_{mod,st}$ de éste se calculará según (2.51), donde N_d representa el número de diodos:

$$V_{mod,st}(I_{mod}) = \sum_{i}^{N_d} V_{dcs,i}(I_{mod}) \tag{2.51}$$

Sin embargo, en el caso del módulo de célula partida, las conexiones en paralelo de las cadenas inferiores y superiores de cada diodo fuerza a que la evaluación explícita de la curva *I-V* de este conjunto sólo pueda realizarse calculando la corriente $I_{cdc,i}$ como función de la tensión, a través de (2.52):

$$I_{cdc,i}(V_{cdc,i}) = I_{cs,inf,i}(V_{cdc,i}) + I_{cs,sup,i}(V_{cdc,i}) + I_d(-V_{cdc,i}) \tag{2.52}$$

Siendo:

- $\boldsymbol{I_{cdc,i}}$: corriente de entrada al conjunto *i* formado por dos cadenas de células en paralelo (inferior y superior), que se conectan a su vez en antiparalelo con un diodo de by-pass.
- $I_{cs,inf,i}$: corriente que circula por la cadena inferior del conjunto *i*.
- $I_{cs,sup,i}$: corriente que circula por la cadena superior del conjunto *i*.
- $V_{cdc,i}$: tensión aplicada al conjunto *i*.

Aunque la corriente $I_{cs,i}$ que circula por una cadena puede calcularse como aquella que minimiza la función residuo $r = |V_{cdc,i} - V_{cs,i}(I_{cs,i})|$, a través de algún método numérico, se opta por generar una curva interpolada $I_{cs,i}(V)$ a través de la evaluación de la curva $V_{cs,i}(I)$ en N_{cs} puntos,

distribuidos uniformemente en el intervalo [$I_{cs,min}$, $I_{cs,max}$], cuyos extremos se calcularán multiplicando la corriente de cortocircuito *STC* de la célula por sendos coeficientes $\delta_{min,cs}$ y $\delta_{max,cs}$. Los valores por defecto en *PV-AMP* son {$\delta_{min,cs}$ = -0.1, $\delta_{max,cs}$ = 1.2, N_{cs} = 100}. La interpolación, de tipo lineal, se realiza a través de la librería *numpy* [27].

Puesto que el módulo de célula partida consiste en conexiones en serie de grupos cadena-diodo-cadena, su tensión en función de la corriente que circula por él se calculará según (2.53):

$$V_{mod,hc}(I_{mod}) = \sum_{i}^{N_d} V_{cdc,i}(I_{mod}) \tag{2.53}$$

Al igual que sucede con la corriente de una cadena de células, la tensión de un grupo $V_{cdc,i}$ no puede evaluarse de forma explícita en función de la corriente de entrada. Por ello, se emplea la misma estrategia de interpolación lineal, mediante la evaluación de la corriente $I_{cdc}(V)$, en N_{cdc} puntos, distribuidos uniformemente en el intervalo [$V_{cdc,min}$, $V_{cdc,max}$], cuyos extremos se calcularán multiplicando la tensión de circuito abierto *STC* de una cadena por sendos coeficientes $\delta_{min,cdc}$ y $\delta_{max,cdc}$. Los valores por defecto en *PV-AMP* son {$\delta_{min,cdc}$ = -0.1, $\delta_{max,cdc}$ = 1.2, N_{cdc} = 150}.

Como corolario, se citan los pasos a seguir para generar las curvas *I-V* de cada módulo:

- ***Módulo estándar***:
  1) Evaluar la curva $V(I)$ mediante (2.49) directamente.
- ***Módulo de célula partida***:
  1) Generar la curva interpolada I_{cs} de cada cadena del módulo mediante (2.45).
  2) Generar la curva interpolada V_{cdc} de cada grupo cadena-diodo-cadena del módulo mediante (2.52).
  3) Evaluar la curva $V(I)$ mediante (2.53)

2.4.3 Implementación de los módulos

Para la implementación de los módulos estándar y de célula partida, se hará de los objetos *standardModule* y *halfCellModule*, respectivamente. A través de estos objetos, se simularán sus curvas *I-V*, y se construirán las cadenas de módulos conectados en serie. Los datos de entrada para inicializar estos objetos se dividen en tres tipos: datos constructivos (aptdo. 2.4.2.1), datos eléctricos (aptdo. 2.4.1.2) y datos de la simulación (apartados 2.3.2 y 2.4.1.1):

- ***Datos constructivos***:
  - Límites inferiores y superiores de los lados del módulo, paralelos a los ejes *X* e *Y* de su sistema de referencia local → [x_{min}, x_{max}], [y_{min}, y_{max}] (aptdo. 2.3.2.3).

- Número de diodos del módulo fotovoltaico.
- Columnas en las que se divide el módulo por cada diodo.
- En caso de que el módulo sea estándar, se especificará el número de baldosas en las que se dividen las columnas asociadas a un mismo diodo; en caso de que sea de célula partida, se especificará el número de baldosas por semicolumna.
- Número de células por baldosa.

- ***Datos eléctricos:***
  - Corriente de cortocircuito, en condiciones *STC*.
  - Tensión de circuito abierto, en condiciones *STC*.
  - Intensidad, tensión y potencia en el punto de máxima potencia, en condiciones *STC*.
  - Coeficientes de variación con la temperatura de la corriente de cortocircuito, de la tensión de circuito abierto, y de la potencia máxima.
  - Temperatura de célula en condiciones normales de operación *NOCT*.
  - Coeficiente de bifacialidad de la potencia máxima, si el módulo es bifacial.
  - Factor de minoración de la irradiancia recibida por el módulo.
- ***Datos de la simulación:***
  - Método a emplear para extraer los parámetros de la célula fotovoltaica.
  - Reflectividades de la caras frontal y trasera.
  - Parámetros {*nxb*, *nyb*} y {*nxd*, *nyd*} para el mallado de los nodos directos y difusos, respectivamente, que se aplicarán por igual a todas las baldosas generadas.

El proceso de inicialización e implementación de los módulos, mostrado en la Figura 16, es similar al proceso explicado en el aptdo. 2.3.3, particularizado al caso de un mosaico de tipo *receivingRectangle*. Los comandos asociados a estos objetos pueden consultarse en el Anexo B.5

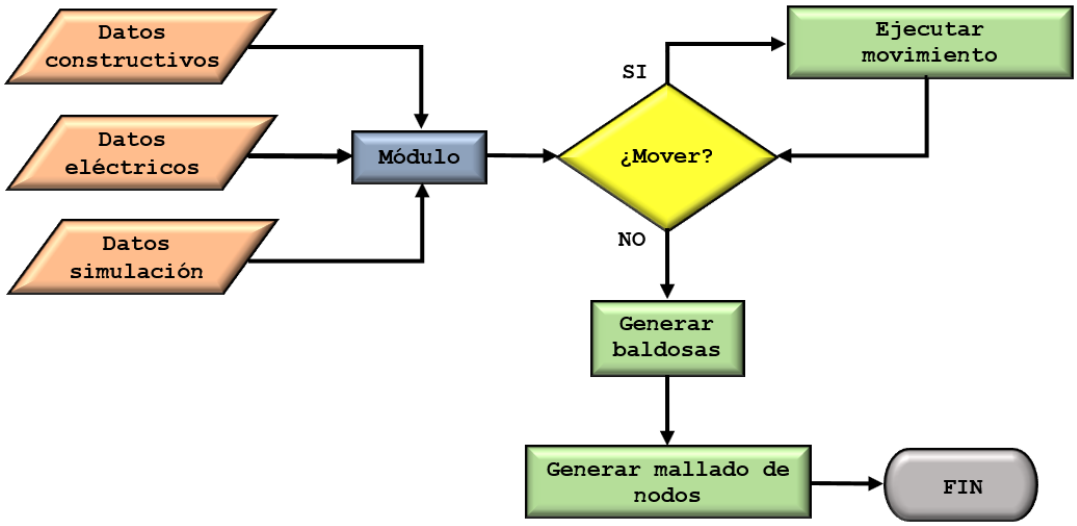

Figura 16. Inicialización e implementación de los módulos

A su vez, cuando se requiera la evaluación de su curva *I-V* para unas condiciones de temperatura ambiente T_{amb} e irradiancia *G* dadas, almacenadas en su objeto *Scene* asociado (aptdo. 2.3.5), deberá realizarse una secuencia de pasos previos para actualizar los datos a dichas condiciones ambientales. En este proceso se extraerán los parámetros del modelo de un diodo para cada una de sus células (aptdo. 2.4.1.2). En caso de que el módulo sea de tipo estándar, la evaluación de su curva ya es posible; si el módulo es de célula partida, deberán generarse las curvas interpoladas de cada cadena de células y de cada grupo cadena-diodo-cadena (aptdo. 2.4.2.2). El diagrama de flujo de este proceso de acondicionamiento se muestra en la Figura 17:

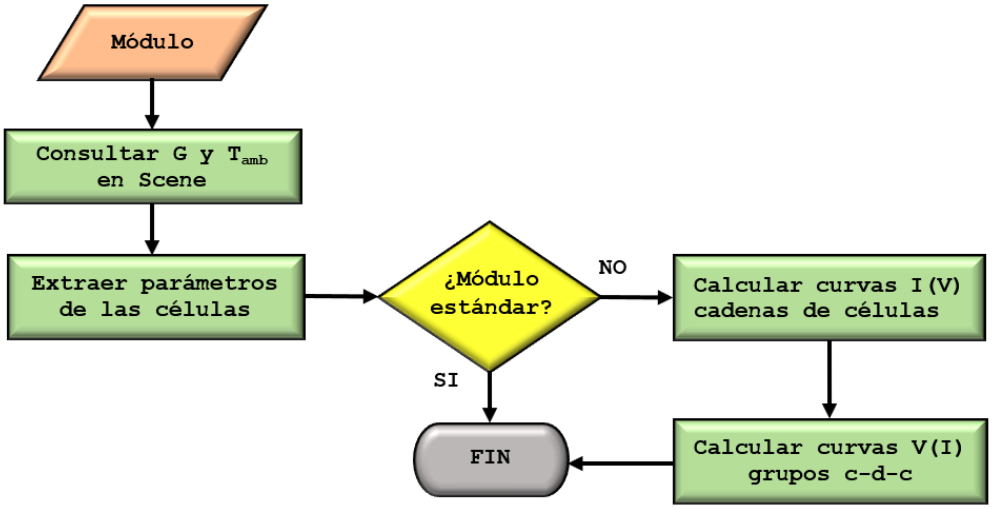

Figura 17. Fase previa a la evaluación de la curva *I-V* de un módulo

2.4.4 Modelado de la cadena de módulos fotovoltaicos

Una cadena de módulos fotovoltaicos o *string* consiste en conexiones en serie de módulos fotovoltaicos, y se modelará partiendo de las siguientes especificaciones:

- Curvas *V*(*I*) de los módulos representativos y sus respectivas cantidades, entendiéndose por módulo representativo aquel que, por sus condiciones de irradiancia y temperatura, se considera representativo de un determinado conjunto de módulos del *string*, en cualquier instante de tiempo en el que se evalúen.
- Resistencia eléctrica total del cableado (R_{total}). Esta resistencia integra las propias del conexionado entre los módulos, de las protecciones y de los polos de salida del *string*.
- Diodo de bloqueo. Su incorporación es opcional.

En la Figura 18 se muestran los componentes empleados para caracterizar un *string*, en la que cada módulo representativo posee un color distinto.

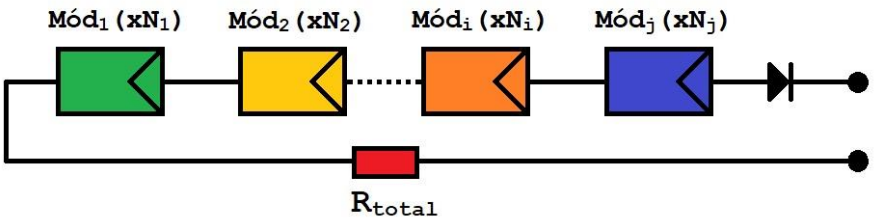

Figura 18. Elementos que conforman un *string*

En lo que a la curva *I-V* del *string* respecta, el caso más general se da cuando se considera la presencia de resistencias y del diodo de bloqueo. En este caso, su tensión viene dada por (2.54), la cual se puede evaluar de forma explícita empleando como parámetro la corriente:

$$V_{string}(I_{string}) = \sum_{i}^{N_{mr}} N_{mod,i} V_{mod,i}(I_{string}) - I_{string} R_{total} - V_d(I_{string}) \qquad (2.54)$$

Siendo:

- $\boldsymbol{V_{string}}$: función de tensión del *string*.
- $\boldsymbol{I_{string}}$: corriente del *string*.
- $\boldsymbol{N_{mr}}$: número de módulos representativos del *string*.
- $\boldsymbol{V_{mod,i}}$: función de tensión del módulo *i*, dada por (2.49) si el módulo es estándar, y por (2.53) si el módulo es de célula partida.
- $\boldsymbol{N_{mod,i}}$: cantidad de módulos iguales al módulo *i* en el *string*.
- $\boldsymbol{R_{total}}$: resistencia total del *string*.
- $\boldsymbol{V_d}$: función de tensión del *string*, dada por (2.48).

En el caso de que el *string* se conecte en paralelo con otros elementos, se precisará del conocimiento de su curva *I*(*V*). Puesto que las conexiones se realizan en serie, dicha curva *I*(*V*) del *string* no puede evaluarse de forma explícita, por lo que se aproximará mediante una interpolación lineal basada en la evaluación de la tensión $V_{string}(I)$ en N_{string} puntos, distribuidos uniformemente en el intervalo $[I_{string,min}, I_{string,max}]$, cuyos extremos se calcularán multiplicando la corriente de circuito *STC* del *string*[6] por sendos coeficientes $\delta_{min,string}$ y $\delta_{max,string}$. Los valores por defecto en *PV-AMP* son $\{\delta_{min,string} = -0.1, \delta_{max,string} = 1.2, N_{string} = 150\}$.

[6] En el caso de que el *string* esté formado por módulos con distintas corrientes de cortocircuito *STC*, se asumirá que la del *string* es la mayor de ellas.

2.4.5 Implementación de las cadenas de módulos fotovoltaicos

Para la implementación de una cadena de módulos fotovoltaicos o *strings*, se hará uso del objeto *String*. A través de él, se simularán sus curvas *I-V*, y se construirán objetos de niveles superiores, como cajas de conexiones o *MPPT*. Los datos de entrada para inicializar e implementar este objeto son los siguientes:

- Etiqueta o alias con el que se identificará al *string*.
- Módulos fotovoltaicos representativos y sus respectivas cantidades. Estos módulos serán instancias de los objetos *standardModule* o *halfCellModule*.
- Resistencias eléctricas que se conectan en serie.
- Diodo de bloqueo.

Las resistencias y los distintos módulos representativos, junto con sus respectivas cantidades, se añadirán de forma secuencial. El diodo de bloqueo será añadido una única vez, resultando baldío repetir su incorporación. El diagrama de flujo del proceso de inicialización e implementación se muestra en la Figura 19. Los comandos asociados a dicho proceso pueden consultarse en el Anexo B.6

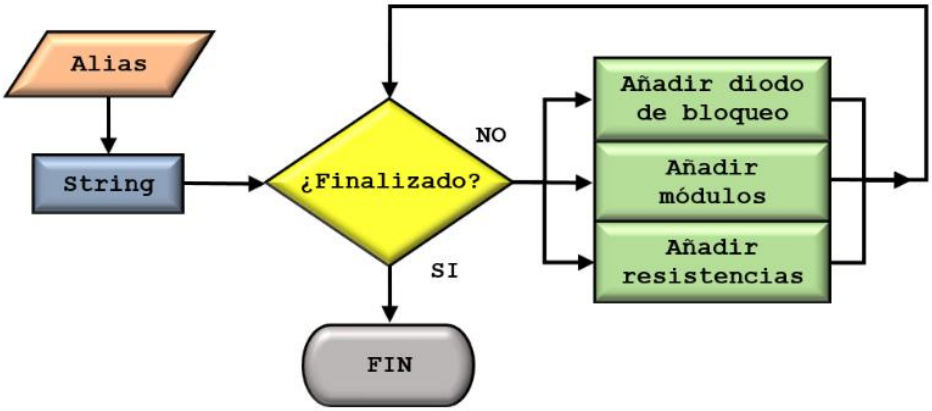

Figura 19. Inicialización e implementación del objeto *String*

Cuando se precise la evaluación de su curva *I-V* para unas condiciones de temperatura ambiente T_{amb} e irradiancia *G* dadas, deberán realizarse unos pasos previos para poder evaluar las curvas *I-V* de los módulos en tales condiciones, tal y como se explicó en el aptdo. 2.4.3. Una vez actualizados los módulos, la curva *V*(*I*) del *string* puede ser evaluada directamente. En caso de que se necesite evaluar *I*(*V*), se deberá generar de forma previa su curva interpolada (aptdo. 2.4.4). El diagrama de flujo de este proceso de acondicionamiento se muestra en la Figura 20:

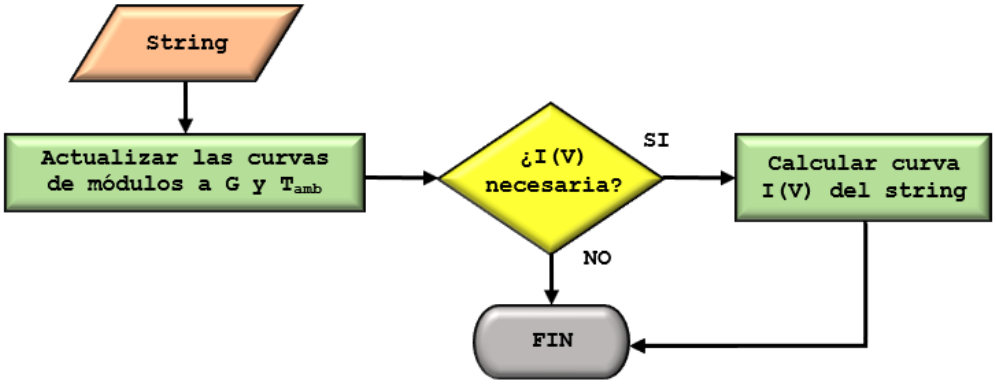

Figura 20. Fase previa a la evaluación de la curva *I-V* de un *string*

2.4.6 Modelado de la caja de conexiones

Una caja de conexiones representará cualquier conexión en paralelo de una serie de pares de polos de entrada, de la que resultan un único par de polos de salida. Estos polos podrán proceder de *strings* y otras cajas de conexiones. Se modelará partiendo de las siguientes especificaciones:

- Curvas *I*(*V*) de los elementos representativos de entrada (strings u otras cajas de conexiones) y sus respectivas cantidades.
- Resistencia eléctrica total del par de polos que abandona la caja de conexiones (R_{total}).

En la Figura 21 se muestra un ejemplo de cómo reflejar un arreglo de dos variedades de *strings* (*SA* y *SB*) con sendas resistencias (R_{SA} y R_{SB}), y las propias de su interconexionado (R_{CA} y R_{CB}), a través de dos cajas de conexiones (*CA* y *CB*).

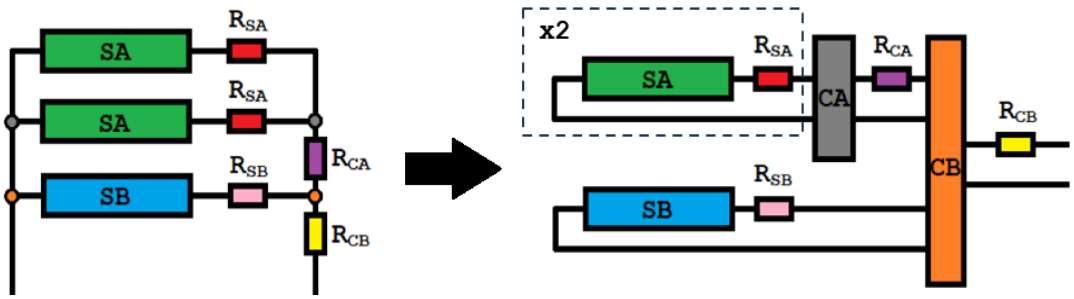

Figura 21. Ejemplo de uso de las cajas de conexiones

En el caso más general, los polos de salida de la caja de conexiones generarán una caída de tensión, por lo que se conviene definir una tensión V_{in} en los bornes de conexión de la caja, y otra tensión V_{box} en los extremos de los polos de salida de ésta, relacionadas mediante (2.55), en la que I_{box} representa la corriente por dichos polos de salida:

$$V_{box}(I_{box}) = V_{in} - I_{box}R_{total} \tag{2.55}$$

Puesto que todas las entradas se conectan en paralelo, la corriente de salida I_{box} puede evaluarse de forma explícita a través de (2.56):

$$I_{box}(V_{in}) = \sum_{i}^{N_{sr}} N_{string,i} I_{string,i}(V_{in}) + \sum_{i}^{N_{br}} N_{box,i} I_{box,i}(V_{in}) \qquad (2.56)$$

Siendo:

- $\boldsymbol{I_{box}}$: función de corriente de la caja de conexiones.
- $\boldsymbol{V_{in}}$: tensión en los bornes de la caja de conexiones.
- $\boldsymbol{N_{sr}}$: número de *strings* representativos que se conectan.
- $\boldsymbol{I_{string,i}}$: función de corriente del *string i*.
- $\boldsymbol{N_{string,i}}$: cantidad de *strings* similares al *string i*.
- $\boldsymbol{N_{br}}$: número de cajas de conexiones representativas que se conectan.
- $\boldsymbol{I_{box,i}}$: función de corriente de la caja de conexiones *i*.
- $\boldsymbol{N_{box,i}}$: cantidad de cajas de conexiones similares a la caja de conexiones *i*.

En síntesis, se tiene que la curva *I-V* de la caja de conexiones puede evaluarse de forma explícita mediante (2.55) y (2.56) si se emplea como parámetro de entrada la tensión en sus terminales. No obstante, al poder participar también la caja de conexiones como un elemento de entrada, se necesitará obtener su curva de corriente I_{box} como función de la tensión en los extremos de los polos de salida V_{box}. Dicha curva se obtendrá mediante una interpolación lineal basada en la evaluación de la tensión $V_{box}(V_{in})$ y de la corriente $I_{box}(V_{in})$ en N_{box} puntos, distribuidos uniformemente en el intervalo $[V_{in,min}, V_{in,max}]$, cuyos extremos se calcularán multiplicando la tensión de circuito abierto *STC* de la caja de conexiones[7] por sendos coeficientes $\delta_{min,box}$ y $\delta_{max,box}$. Los valores por defecto en *PV-AMP* son $\{\delta_{min,box} = -0.1, \delta_{max,box} = 1.2, N_{box} = 150\}$.

2.4.7 Implementación de las cajas de conexiones

Para la implementación de una caja de conexiones, se hará uso del objeto *junctionBox*. A través de este objeto, se simularán sus curvas *I-V*, y se construirán objetos de niveles superiores, como otras cajas de conexiones o *MPPT*. Los datos de entrada para inicializar e implementar este objeto son los siguientes:

- Etiqueta o alias con el que se identificará a la caja de conexiones.

[7] En el caso de que en la entrada de la caja de conexiones figuren elementos con distintas tensiones de circuito abierto *STC*, se asumirá que la de la caja es la mayor de ellas.

- Cajas de conexiones representativas y sus respectivas cantidades. Serán instancias del objeto *junctionBox*.
- *Strings* representativos y sus respectivas cantidades. Serán instancias del objeto *String*.
- Resistencias eléctricas que se conectan en serie, en alguno de los polos de salida de la caja de conexiones.

Las resistencias y los elementos representativos, junto con sus respectivas cantidades, se añadirán de forma secuencial. El diagrama de flujo del proceso de inicialización e implementación se muestra en la Figura 22. Los comandos asociados a dicho proceso pueden consultarse en el AnexoB.6

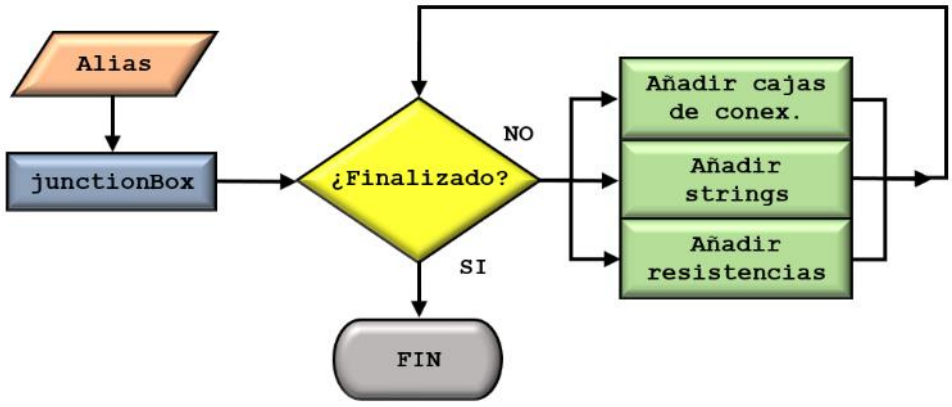

Figura 22. Inicialización e implementación del objeto *junctionBox*

Para poder evaluar su curva *I-V* en unas condiciones operación dadas, se deberá estar primero en disposición de poder evaluar las curvas *I*(*V*) de cada elemento de la caja de conexiones. Tal y como se explicó en el apartado anterior, la curva *I-V* se puede determinar explícitamente utilizando como parámetro la tensión V_{in} en los bornes de conexión. En caso de que se necesite evaluar *I*(*V*), se deberá generar de forma previa su curva interpolada (aptdo. 2.4.6). El diagrama de flujo de este proceso de acondicionamiento se muestra en la Figura 23:

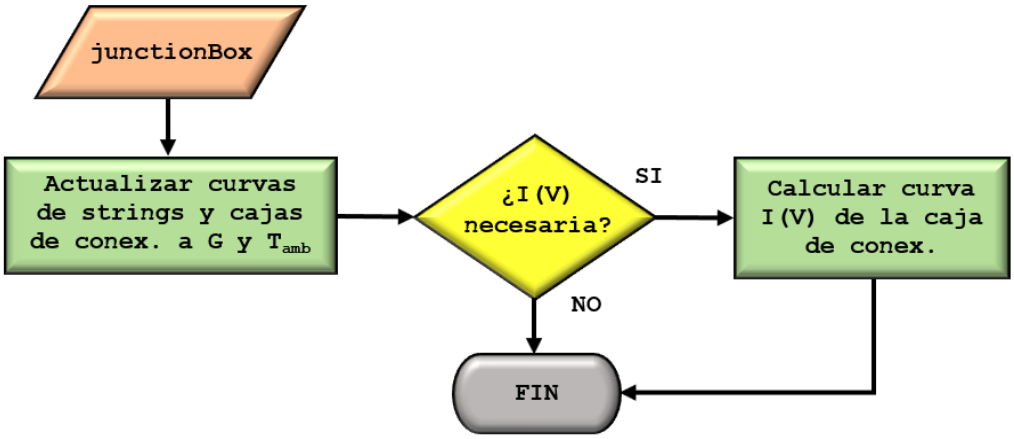

Figura 23. Fase previa a la evaluación de la curva *I-V* de una caja de conexiones

2.4.8 Modelado del MPPT

El *MPPT* (*Maximum Power Point Tracker*) es el sistema encargado de fijar la tensión que maximiza la potencia de salida conjunta de los elementos que se conecten a él, dentro de una determinada ventana de operación [V_{min}, V_{max}]. Se modelará partiendo de las siguientes especificaciones:

- Curvas *I*(*V*) de los elementos representativos de entrada y sus respectivas cantidades.
- Ventana de operación en la que el *MPPT* busca la tensión que maximiza la potencia del conjunto de entrada.

Se hace notar que este modelado, por simplicidad, asume las siguientes hipótesis:

- El algoritmo de búsqueda del punto de máxima potencia es ideal.
- No existen pérdidas por autoconsumo.
- Admite infinitas entradas.
- Pueden soportar una corriente infinita.

Puesto que todas las entradas se conectan en paralelo, el *MPPT* puede entenderse como un caso particular de una caja de conexiones en la que no se tienen caídas de tensión en los polos de salida. Por tanto, la curva *I*-*V* conjunta de los elementos que se conectan a su entrada puede evaluarse explícitamente empleando como parámetro la tensión V_{mppt}, a través de (2.56). Para determinar el punto de máxima potencia, se realizará una maximización de la función $P_{mppt}(V_{mppt}) = I_{mppt}(V_{mppt}) \cdot V_{mppt}$, restringida al intervalo $[V_{min}, V_{max}]$. En particular, la optimización se realizará a través del método de Brent [28], que se implementa en *PV-AMP* a través de la librería *scipy* [5]. La potencia de salida del *MPPT* vendrá expresada por (2.57):

$$P_{mppt,out} = \max\{I_{mppt}(V_{mppt}) \cdot V_{mppt} : V_{mppt} \in [V_{min}, V_{max}]\} \tag{2.57}$$

2.4.9 Implementación del MPPT

Para la implementación de un *MPPT*, se hará del objeto *mppTracker*. A través de este objeto, se simularán las curvas *I*-*V* conjuntas de los elementos que se conecten (en paralelo) a su entrada, y se determinará el punto de máxima potencia de este sistema conjunto. Los datos de entrada para inicializar e implementar este objeto son los siguientes:

- Etiqueta o alias con el que se identificará al *MPPT*.
- Cajas de conexiones representativas y sus respectivas cantidades. Serán instancias del objeto *junctionBox*.

- *Strings* representativos y sus respectivas cantidades. Serán instancias del objeto *String*.
- Rango de tensión en el que el *MPPT* busca el punto de máxima potencia.

Al igual que en el caso de las cajas de conexiones, los elementos representativos, junto con sus respectivas cantidades, se añadirán de forma secuencial. El diagrama de flujo del proceso de inicialización e implementación se muestra en la Figura 24. Los comandos asociados a dicho proceso pueden consultarse en el Anexo B.8

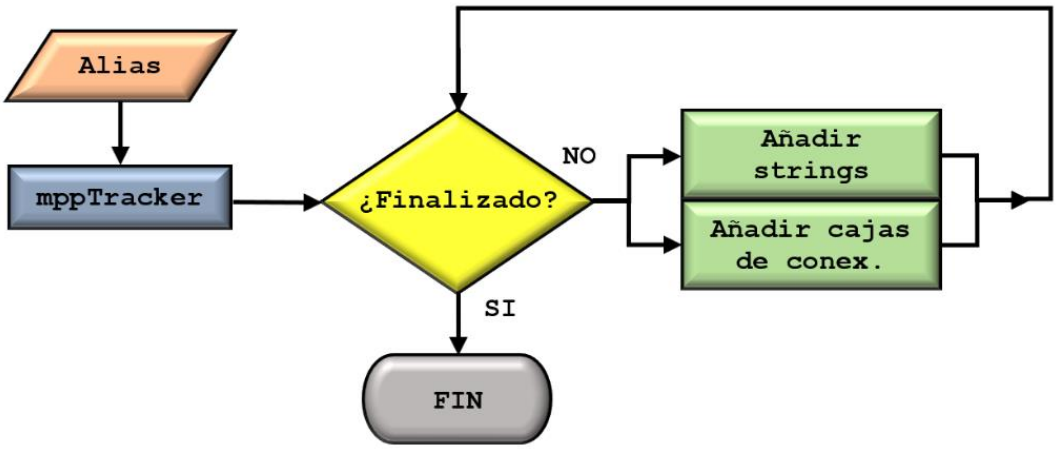

Figura 24. Inicialización e implementación del objeto *mppTracker*

Cuando se precise la determinación de la potencia máxima que el *MPPT* es capaz de extraer del campo fotovoltaico que gestiona, para unas condiciones de operación dadas, se deberá estar primero en disposición de poder evaluar las curvas $I(V)$ de cada elemento conectado al *MPPT*. El diagrama de flujo de este proceso de acondicionamiento se muestra en la Figura 25:

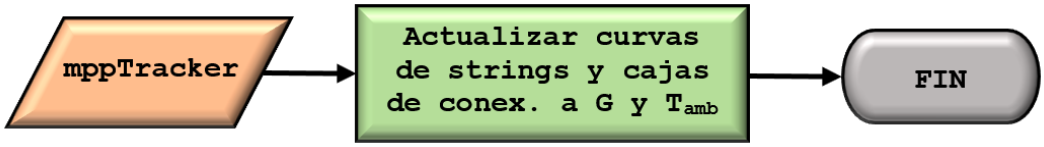

Figura 25. Fase previa a la obtención de la potencia máxima de un *MPPT*

2.4.10 Modelado del inversor

El inversor será el sistema encargado de transformar la corriente continua de salida de los *MPPT* que lo integran en corriente alterna. Como hipótesis, se supondrá que no existen pérdidas por autoconsumo. Los datos de partida para modelarlo son los siguientes:

- Potencias de salida de los *MPPT* representativos y sus respectivas cantidades.
- Potencia nominal del inversor.
- Tensión nominal del circuito de corriente alterna.

- Número de fases del circuito de corriente alterna.
- Resistencia eléctrica de los conductores del circuito de corriente de alterna.

Por simplicidad, se asumen las siguientes hipótesis:

- No existen pérdidas por autoconsumo.
- Toda la potencia cedida por el inversor es activa.
- La tensión del circuito de alterna es constante.

El modelo de la eficiencia del inversor empleado está basado en una modificación del modelo polinomial de eficiencia de Schmid [29], que determina la potencia de salida en función de la potencia de entrada según (2.58) y (2.59):

$$\eta_{inv} = \frac{p_{in} - (b_0 + b_1 p_{in} + b_2 p_{in}^2)}{p_{in}} \quad ; \quad p_{in} = \frac{P_{DC}}{P_{nom}} \tag{2.58}$$

$$P_{DC} = \sum_{i}^{N_{mpr}} N_{mppt,i} P_{mppt,out,i} \tag{2.59}$$

Siendo:

$\boldsymbol{\eta_{inv}}$: eficiencia de conversión CC/CA del inversor.

$\boldsymbol{P_{DC}}$: Potencia de entrada al inversor.

$\boldsymbol{p_{in}}$: potencia de entrada normalizada.

$\boldsymbol{P_{nom}}$: potencia nominal del inversor.

$\boldsymbol{N_{mpr}}$: número de *MPPT* representativos.

$\boldsymbol{P_{mppt,out,i}}$: potencia de salida del *MPPT i.*

$\boldsymbol{N_{mppt,i}}$: cantidad de *MPPT* semejantes al *MPPT i* con los que cuenta el inversor.

$\boldsymbol{b_0, b_1, b_2}$: constantes del modelo de eficiencia inversor.

Los valores implementados por defecto en *PV-AMP* para las constantes del modelo de eficiencia son $\{b_0 = 0.005,\ b_1 = 0.016,\ b_2 = 0.014\}$. En la Figura 26 se muestra la curva de eficiencia para a estos valores, en función de la potencia de entrada normalizada. Para valores de esta potencia por encima de 0.3 se tiene una eficiencia prácticamente constante, del 96.6%.

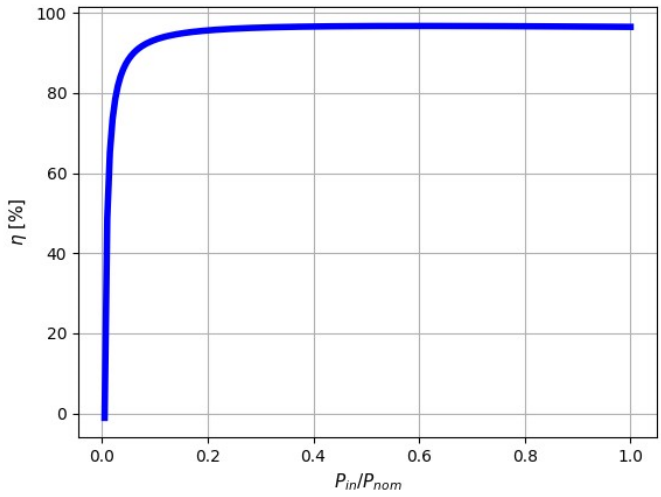

Figura 26. Curva de eficiencia del inversor implementado por defecto

Por lo general, la potencia de salida P_{inv} de un inversor suele quedar limitada a la potencia nominal de éstos. Por tanto, dicha potencia de salida vendrá dada por (2.60):

$$P_{inv} = \max\{P_{nom}, \eta_{inv} P_{in}\} \tag{2.60}$$

Finalmente, la potencia disponible tras considerar las pérdidas en el cableado del circuito de evacuación de alterna vendrá dada por (2.61) en caso de que la red sea monofásica, y por (2.62) en caso de que sea trifásica:

$$P_{AC,1ph} = P_{inv} - 2R_{AC}\left(\frac{P_{inv}}{V_L}\right)^2 \tag{2.61}$$

$$P_{AC,3ph} = P_{inv} - R_{AC}\left(\frac{P_{inv}}{V_L}\right)^2 \tag{2.62}$$

Siendo:

$\boldsymbol{P_{AC,1ph}}$: Potencia final disponible, en caso de red monofásica.

$\boldsymbol{P_{AC,3ph}}$: Potencia final disponible, en caso de red trifásica.

$\boldsymbol{V_L}$: tensión de línea de la red (230 V monofásico; 400 V trifásico).

2.4.11 Implementación del inversor

Para la implementación de un inversor, se hará del objeto *Inverter*. A través de este objeto, se simulará la conversión CC/CA de la energía procedente del campo fotovoltaico que gestiona. Los datos de entrada para inicializar e implementar este objeto son los siguientes:

- Etiqueta o alias con el que se identificará al inversor.
- *MPPT* representativos y sus respectivas cantidades. Serán instancias del objeto *mppTracker*.
- Potencia nominal del inversor.
- Tensión nominal del circuito de corriente alterna.
- Número de fases del circuito de corriente alterna.
- Resistencia eléctrica de los conductores del circuito de corriente de alterna.
- Constantes del modelo de eficiencia inversor (opcional).

Los *MPPT*, junto con sus respectivas cantidades, se añadirán de forma secuencial. El diagrama de flujo del proceso de inicialización e implementación se muestra en la Figura 27. Los comandos asociados a dicho proceso pueden consultarse en el Anexo B.8

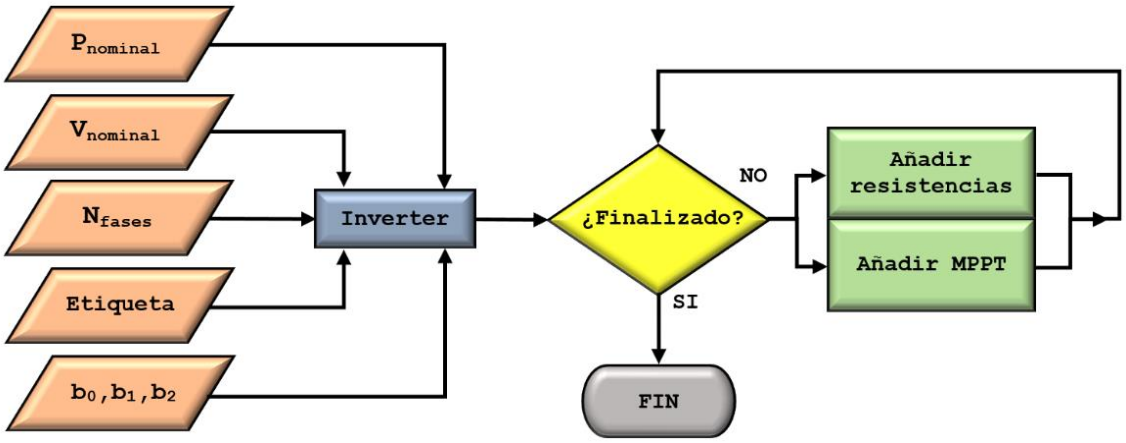

Figura 27. Inicialización e implementación del objeto *Inverter*

2.5 Modelado, implementación y simulación del sistema conjunto

En este TFM, se entenderá por subsistema a aquel conjunto de inversores tales que pueden ser representados por un único inversor, lo que implicará que la producción de energía, tanto en continua como en alterna, y las diferentes pérdidas, son las mismas para cada uno de ellos. La finalidad que se persigue con esta definición es poder escalar con facilidad los diseños, sin que ello incurra en mayores costes computacionales, siempre y cuando esta presunción sea admisible.

Por tanto, para finalizar la etapa del modelado, sólo resta definir los subsistemas que conforman la totalidad de la instalación fotovoltaica, mediante la especificación de los inversores representativos de la misma y las cantidades de éstos. Tras ello, se estará en disposición de simular su producción y las pérdidas en sus componentes. A su vez, se definirán un conjunto de métricas para caracterizar, en términos energéticos, el desempeño de la instalación, siendo presentadas en los siguientes apartados.

2.5.1 Pérdidas energéticas

2.5.1.1 PÉRDIDAS EN LOS MÓDULOS

Atendiendo al modelado realizado del módulo, las pérdidas energéticas del mismo se clasificarán en dos tipos: por un lado, las pérdidas inherentes al módulo, L_{mod}, que condensa las pérdidas por temperatura y las cuantificadas a través del coeficiente de minoración L_{pol} presentado en el aptdo. 2.4.1.2; por otro lado, las pérdidas por sombreado, L_{sh}.

Las pérdidas L_{mod} se calcularán como la diferencia entre la potencia teórica máxima que puede dar el módulo (funcionando a temperatura de célula *STC*, y sin sombreado), y la propia obtenida en una operación sin obstaculización de la radiación. El cálculo de ambas potencias parte de la determinación de la irradiancia en el plano de captación del módulo, mediante (2.63), (2.63a) y (2.63b), que resultan un caso particular de (2.12) en el que se considera que la superficie de captación sólo ve cielo y terreno.

$$G_s = \max\{0, \cos(\theta)\} G_{b,n} + \left(1 - F_{P\to g}\right) G_{d,h} + \rho_g F_{P\to g} G_{g,h} \tag{2.63}$$

$$\cos(\theta) = \begin{cases} \vec{Z} \cdot \vec{r_s} \to \text{cara frontal} \\ -\vec{Z} \cdot \vec{r_s} \to \text{cara trasera} \end{cases} ;\ \theta \in [0, \pi] \tag{2.63a}$$

$$F_{P\to g} = \begin{cases} \dfrac{1 - \cos\ \beta}{2} \to \text{cara frontal} \\ \dfrac{1 + \cos\ \beta}{2} \to \text{cara trasera} \end{cases} \tag{2.63b}$$

Siendo:

- $\boldsymbol{G_s}$: irradiancia sobre alguna de las caras de la superficie de captación.
- $\boldsymbol{G_{b,n}}$: irradiancia directa, proveniente del disco solar, por unidad de superficie normal a la dirección de los rayos solares.
- $\boldsymbol{\theta}$: ángulo formado entre la normal de la superficie de captación ($\vec{Z}$ si se trata de la cara frontal, y -$\vec{Z}$ si se trata de la trasera) y el vector unitario que apunta al disco solar ($\vec{r_s}$). Valores comprendidos en el intervalo [π/2, π] denotan que los rayos inciden en la cara opuesta.
- $\boldsymbol{G_{d,h}}$: irradiancia difusa del cielo sobre el plano horizontal.
- $\boldsymbol{F_{P\to g}}$: factor de visión de la superficie respecto al terreno, sin obstrucción.
- $\boldsymbol{\beta}$: ángulo de inclinación de la superficie con respecto al terreno.
- $\boldsymbol{G_{g,h}}$: irradiancia global horizontal.
- $\boldsymbol{\rho_g}$: reflectividad del terreno.

La irradiancia efectiva G_{eff} recibida por el módulo se calculará a través de (2.64), en el caso de que el módulo sea monofacial, y a través de (2.65) si es bifacial:

$$G_{eff} = G_{s,front} \tag{2.64}$$

$$G_{eff} = G_{s,front} + \varphi G_{s,rear} \tag{2.65}$$

Siendo:

- $\boldsymbol{G_{eff}}$: irradiancia efectiva sobre el módulo.
- $\boldsymbol{G_{s,front}}$: irradiancia de la cara frontal.
- $\boldsymbol{G_{s,rear}}$: irradiancia de la cara trasera.
- $\boldsymbol{\varphi}$: coeficiente de bifacialidad del módulo.

La potencia máxima teórica del módulo $P_{max,teo}$ se calculará según (2.66), donde $P_{mp,STC,mod}$ es la potencia máxima del módulo en condiciones *STC*, y G_{eff} tiene unidades de Wm^{-2}:

$$P_{max,teo} = P_{mp,STC,mod}\left(\frac{G_{eff}}{1000[Wm^{-2}]}\right) \tag{2.66}$$

La potencia máxima del módulo considerando las pérdidas por temperatura y el factor de minoración L_{pol}, se calculará según (2.67), (2.68) y (2.69):

$$G = \left(1 - L_{pol}\right)G_{eff} \tag{2.67}$$

$$T_{cell} = T_{amb} + G\frac{NOCT - 20[°C]}{800[Wm^{-2}]} \tag{2.68}$$

$$P_{max,wos} = P_{mp,STC,mod}\left(\frac{G}{1000[Wm^{-2}]}\right)\left(1 + \gamma(T_{cell} - 25[°C])\right) \tag{2.69}$$

Siendo:

- $\boldsymbol{P_{max,wos}}$: potencia máxima considerando pérdidas por temperatura y el factor de minoración de irradiancia, en las mismas unidades que $P_{mp,STC,mod}$.
- $\boldsymbol{L_{pol}}$: factor de minoración de la irradiancia.
- $\boldsymbol{T_{cell}}$: temperatura de operación de célula [ºC].
- $\boldsymbol{T_{amb}}$: temperatura ambiente [ºC].
- ***NOCT***: temperatura de célula en condiciones *NOCT* [ºC].
- $\boldsymbol{G}$: irradiancia efectiva minorada [Wm^{-2}].
- $\boldsymbol{\gamma}$: coeficiente de variación con la temperatura de la potencia máxima [$ºC^{-1}$].

Por tanto, las pérdidas en el módulo se calcularán según (2.70):

$$L_{mod} = P_{max,teo} - P_{max,wos} \tag{2.70}$$

Las pérdidas por sombreado L_{sh} se calcularán según (2.71), en la que $P_{max,ws}$ representa la potencia máxima posible del módulo, con eventual sombreado:

$$L_{sh} = P_{max,wos} - P_{max,ws} \tag{2.71}$$

Puesto que el sombreado parcial de un módulo se puede materializar en escalonamientos de sus curvas *I-V* y *P-V*, como consecuencia de la activación de sus diodos de *by-pass*, el punto de máxima potencia se determinará mediante una maximización de la función $P(I) = I \cdot V_{mod}(I)$. Al igual que en el caso del *MPPT*, la optimización se realizará a través del método de Brent. Esta potencia máxima se expresa mediante (2.72):

$$P_{max,ws} = \max\{V_{mod}(I) \cdot I\} \tag{2.72}$$

2.5.1.2 PÉRDIDAS POR CONEXIONADO Y PÉRDIDAS EN EL MPPT

Las pérdidas por conexionado en el lado de corriente continua se definirán en este TFM como aquellas que surgen por el mero hecho de interconectar, bajo una determinada topología eléctrica, los módulos fotovoltaicos. Por tanto, estas pérdidas engloban a las siguientes:

- Pérdidas en el cableado, por efecto Joule.
- Pérdidas por caída de tensión en los diodos de bloqueo.
- Pérdidas por *mismatch* o desajuste, producidas cuando se interconectan módulos que operan bajo condiciones distintas.

Puesto que los *MPPT* representan el final de los circuitos de corriente continua, sus entradas serán los puntos de la instalación en los que se evalúen estas pérdidas. Para cuantificarlas, primero se contabilizará la potencia máxima disponible, según (2.73), en el subgenerador, que se define en este TFM como el conjunto de módulos fotovoltaicos que confluyen en un mismo *MPPT*:

$$P_{ws,SGFV} = \sum_{i}^{N_{mr}} N_{mod,i} P_{max,ws,i} \tag{2.73}$$

Siendo:

- $\boldsymbol{P_{ws,SGFV}}$: potencia máxima disponible en el subgenerador.
- $\boldsymbol{N_{mr}}$: número de módulos representativos del subgenerador.
- $\boldsymbol{P_{max,ws,i}}$: potencia máxima disponible en el módulo *i*, calculada según (2.72).
- $\boldsymbol{N_{mod,i}}$: cantidad de módulos iguales al módulo *i* en el subgenerador.

Posteriormente, se determina la potencia disponible a la entrada del *MPPT*, $P_{mmt,in}$, según (2.74). Se destaca que la metodología es la misma que la empleada para calcular la potencia de salida del *MPPT* (2.57) (aptdo. 2.4.8), con la salvedad de que la optimización no se restringe al rango de tensión en la que opera el *MPPT*:

$$P_{mppt,in} = \max\{I_{mppt}(V_{mppt}) \cdot V_{mppt}\} \tag{2.74}$$

Finalmente, las pérdidas $L_{cnx,dc}$ por conexionado del subgenerador fotovoltaico conectado a un *MPPT* vendrá dada por (2.75):

$$L_{cnx,dc} = P_{ws,SGFV} - P_{mppt,in} \tag{2.75}$$

A su vez, las pérdidas en el *MPPT* se definirán como aquellas producidas por quedar la tensión del punto de máxima potencia del subgenerador fuera de la ventana de operación de éste: $[V_{min}, V_{max}]$. Estas pérdidas vendrán dadas por (2.76), donde $P_{mppt,out}$ se calcula según (2.57):

$$L_{mppt} = P_{mppt,in} - P_{mppt,out} \tag{2.76}$$

2.5.1.3 PÉRDIDAS EN EL INVERSOR Y EN EL CIRCUITO DE CORRIENTE ALTERNA

En virtud del modelado realizado para el inversor (aptdo. 2.4.10), las pérdidas que se producen en el inversor se deben a dos causas:

- Pérdidas por la eficiencia del inversor.
- Pérdidas por *clipping*, debidas al hecho de que la potencia máxima que puede dar el inversor se corresponde con su potencia nominal.

Para cuantificar la suma de ambas, L_{inv}, se aplicará (2.77), donde la potencia P_{DC} de entrada al inversor y la potencia P_{inv} de salida se calculan con (2.59) y (2.60), respectivamente:

$$L_{inv} = P_{DC} - P_{inv} \tag{2.77}$$

Finalmente, las pérdidas producidas en el circuito de alterna del inversor, $L_{cnx,ac}$, se calcularán según (2.78), donde la potencia de salida del circuito de alterna P_{AC} vendrá dada por (2.61), si el inversor es monofásico, o por (2.62), si es trifásico:

$$L_{cnx,ac} = P_{inv} - P_{AC} \tag{2.78}$$

2.5.2 Métricas de rendimiento

Para evaluar el desempeño de la instalación, los datos de producción serán relativizados mediante los siguientes parámetros:

- ***PR*** (***Performance Ratio***): Energía producida por el sistema fotovoltaico con respecto a la que se hubiese alcanzado si éste hubiese operado en todo momento a su rendimiento nominal, sin pérdidas, y con el mismo recurso solar. Es una cantidad adimensional.
- $\boldsymbol{Y_f}$ (***Final Yield***): Energía producida por el sistema en relación a su potencia pico, siendo esta última la potencia total *STC* instalada de módulos. Se expresa en unidades de [kWh/kWp].

La energía producida por el sistema en una determinada ventana temporal, en la que se han tomado N_t muestras, distanciadas en el tiempo una cantidad Δt, vendrá dada por (2.79):

$$E_{ac} = \Delta t \sum_{i}^{N_t} \sum_{j}^{N_{ir}} N_{inv,j} P_{ac,i,j} \tag{2.79}$$

Siendo:

- N_t: número de mediciones.
- Δt: intervalo de tiempo entre mediciones.
- N_{ir}: número de inversores representativos del sistema.
- $N_{inv,j}$: cantidad de inversores similares al inversor j.
- $P_{inv,i,j}$: potencia instantánea del inversor j, en la medición i.

A su vez, la energía máxima producida a rendimiento nominal constante, en el mismo intervalo de tiempo, vendrá dada por (2.80):

$$E_{max,teo} = \Delta t \sum_{i}^{N_t} \sum_{j}^{N_{mr}} N_{mod,j} P_{max,teo,i,j} \tag{2.80}$$

Siendo:

- N_{mr}: número de módulos representativos del sistema.
- $N_{mod,j}$: cantidad de módulos similares al módulo j.
- $P_{max,teo,i,j}$: potencia máxima teórica del módulo j, en la medición i, calculada según (2.66).

Finalmente, la potencia STC del generador fotovoltaico se calculará según (2.81):

$$P_{GFV,STC} = \sum_{i}^{N_{mr}} N_{mod,i} P_{mp,STC,mod,i} \tag{2.81}$$

En disposición de estos valores, el *PR* y el Y_f del sistema, referenciado a la ventana temporal considerada en las expresiones anteriores, se calculará según (2.82) y (2.83), respectivamente:

$$PR = \frac{E_{ac}}{E_{max,teo}} \tag{2.82}$$

$$Y_f = \frac{E_{ac}}{P_{GFV,STC}} \tag{2.83}$$

2.5.3 Implementación del sistema conjunto

En este punto de la implementación del sistema fotovoltaico, toda la información relativa a la topología eléctrica del lado de continua queda embebida en cada inversor definido, a través de sus *MPPT*. Por tanto, sólo resta especificar los diferentes subsistemas que conforman la instalación en su conjunto, siendo cada subsistema definido a través de su inversor representativo. Para ello, se hará uso del objeto *sysManager*, mediante el que se realizarán y gestionarán todas las simulaciones del sistema implementado. Todos los comandos asociados a este objeto pueden consultarse en el Anexo B.10

La inicialización parte de la especificación de si se quieren evaluar las diferentes pérdidas en el sistema durante las simulaciones, y de si se desea, además, contabilizar la irradiación acumulada sobre los receptores. Esta parametrización persigue aliviar el coste computacional de las simulaciones, en caso de que únicamente se desee tener acceso a los datos de producción del sistema y sus métricas de rendimiento. Posteriormente, los distintos subsistemas serán añadidos de forma secuencia, a través de la especificación de los inversores representativos de cada subsistema, sus respectivas cantidades y las etiquetas o alias con las que serán identificados a la hora de presentar los resultados.

Cuando finaliza el proceso de especificación de los subsistemas, se ejecutan los siguientes pasos:

- Generación de tablas para acumular los resultados. En particular, por cada subsistema, se generan dos tablas: una para acumular los resultados de las simulaciones horarias, y la otra para almacenar los resultados globales de la simulación, referidos a todo el intervalo temporal abarcado por ésta, que podrá tener carácter diario, semanal, mensual, etc. Además, se generarán las propias para todo el sistema conjunto.
- Generación de listados, sin repeticiones, para cada uno de los componentes representativos de la instalación (módulos, *strings*, cajas de conexiones, *MPPT* e inversores) y para las distintas escenas definidas. En los listados de los *strings* y cajas de conexiones, figurarán primero aquellos que estén más embebidos. Por ejemplo, si una caja de conexiones *A* forma parte de otra caja *B*, entonces *A* irá antes que *B* en el listado. Con ello se consigue simular cada componente en el orden correcto, desde los primeros del listado hasta los últimos.

El diagrama de flujo del proceso de inicialización e implementación descrito se muestra en la Figura 28:

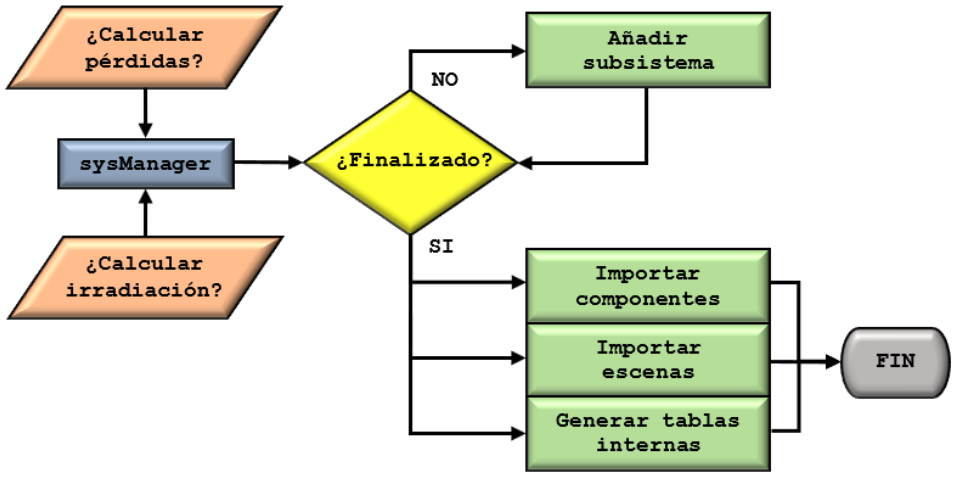

Figura 28. Inicialización e implementación del objeto *sysManager*

2.5.4 Simulación del sistema conjunto

En el caso más general, la simulación de la producción y de las pérdidas del sistema fotovoltaico, para una fecha y hora concretas, partirá de un acondicionamiento previo, dado por la siguiente secuencia de pasos:

1) Cálculo del balance radiante en las superficies de cada escenario.
2) Actualización de los parámetros del modelo de un diodo de cada célula del listado de módulos, seguido del cálculo de la curva interpolada V(I) de los módulos de célula partida.
3) Cálculo de las curvas I(V) de cada elemento del listado de *strings*.
4) Cálculo de las curvas I(V) de cada elemento del listado de cajas de conexiones, empezando por el primer elemento.

Posteriormente, la simulación se realizará a través de la ejecución de la siguiente secuencia:

1) Por cada módulo del listado, se calculará su potencia máxima teórica y sus pérdidas (por minoración de irradiancia, temperatura y sombreados). Los resultados se almacenarán en sendas variables internas del módulo.
2) Por cada *MPPT* del listado, se calculará su potencia de salida. Dicha potencia se almacenará en una variable interna del objeto *mppTracker*. A su vez, se consultarán las potencias máximas y pérdidas de los módulos asociados. Se calcularán y almacenarán las pérdidas por conexionado en el lado de continua y las propias por la ventana de operación del *MPPT*.

3) Por cada inversor del listado, se consultarán las potencias y pérdidas de sus *MPPT*. Posteriormente, se calcularán y almacenarán las potencias a la salida de éstos, junto con las pérdidas por conversión CC/CA, y las propias del cableado del circuito de alterna.

4) Calcular y almacenar en las tablas los parámetros y pérdidas de cada subsistema, y del sistema conjunto, mediante la consulta de los resultados de la simulación almacenados en cada objeto.

En las figuras 29 y 30 se muestran, respectivamente, el diagrama de flujo de la fase de acondicionamiento de las curvas I-V del sistema, y el diagrama de flujo del proceso de cálculo de la producción, pérdidas y rendimientos

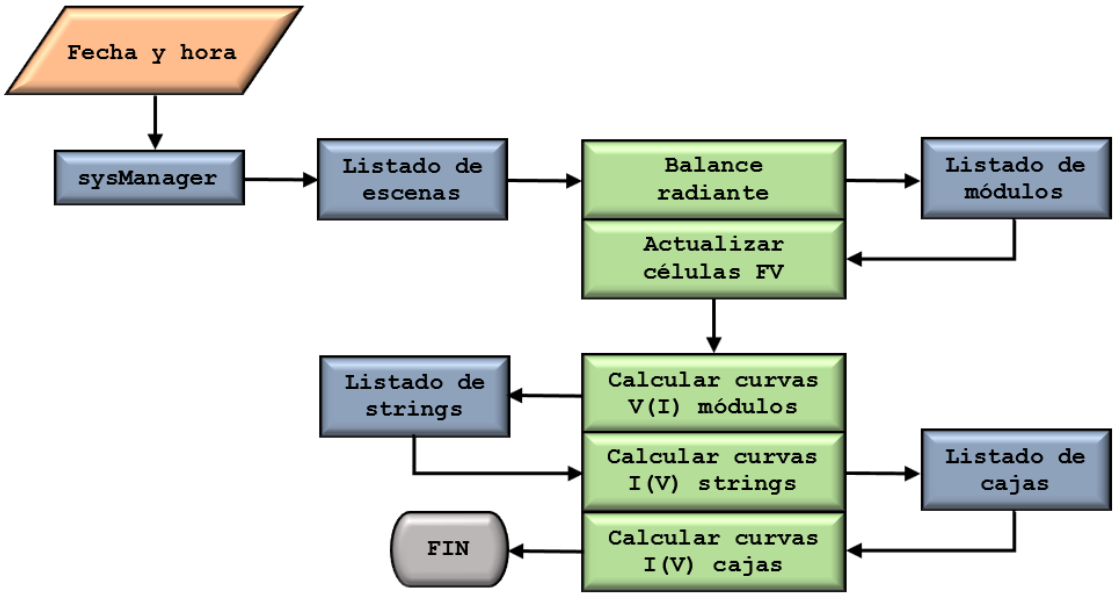

Figura 29. Fase previa al cálculo de la producción, pérdidas y rendimientos

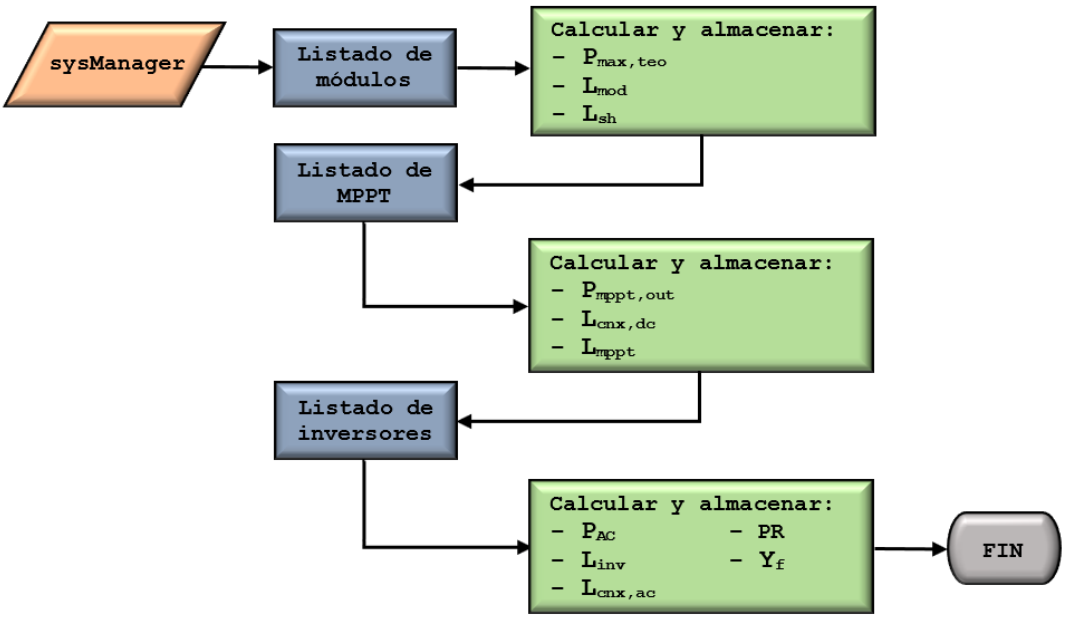

Figura 30. Proceso de simulación horaria

3 CASOS DE ESTUDIO

3.1 Simulación de la irradiación horaria

En este ejemplo se presenta la implementación un escenario compuesto por una parcela cuadrada y un obstáculo basado en el *triángulo de Sierpinski*, al objeto de simular la irradiación horaria sobre la parcela.

Como primer paso, se importan los objetos necesarios al *script* de trabajo:

```
# Importación de objetos
from pv_amp import *
```

En las siguientes líneas se implementan las superficies del escenario:

```
# Vértices del triángulo de Sierpinski
s_vertices = [
    [(0, 1-i/4), (0, 1-(i+1)/4), (1/4, 1-(i+1)/4)]
    for i in range(4)]

s_vertices += [
    [(1/4, 1-(2*i+1)/4), (1/4, 1-(i+1)/2), (1/2, 1-(i+1)/2)]
    for i in range(0, 2)]

s_vertices += [
    [(1/2,1/2-i/4), (1/2,1/4-i/4), (3/4, 1/4-i/4)]
    for i in range(0, 2)]

s_vertices += [[(3/4, 1/4), (3/4, 0), (1, 0)]]

# Generación del obstáculo
sierpinski_obstacle = shadingMosaic(
    tiles_data=[
        {
            'vertices_2d': vert,
            'type': 'generic'
        }
    for vert in s_vertices])

# Movimientos para ubicar la hipotenusa del triángulo sobre el eje X
sierpinski_obstacle.rotation_3d(90, v=[-1, 1, 0])
sierpinski_obstacle.rotation_3d(45, axis='z')
sierpinski_obstacle.move_3d([0, 0, np.sqrt(2)/2])
sierpinski_obstacle.end_def()
```

```
# Generación del receptor rectangular horizontal
floor = receivingRectangle(
    x_interval=[-1, 1],
    y_interval=[-1/4, 3/2],
    nxt=80,
    nyt=80,
    nxb=4,
    nyb=4
)
floor.end_def()
```

A través de la siguiente función, se visualizan las superficies implementadas:

```
# Visualización de las superficies
floor.plot_mosaic_3d(
    data_env=[
        {
            'mosaic': sierpinski_obstacle,
            'color': 'tab:blue',
            'alpha': 1
        }
    ],
    color='g',
    alpha=0.3
)
```

La ejecución del código implementado hasta ahora abrirá una ventana, correspondiente con la Figura 31, en la que el obstáculo se representa en color azul.

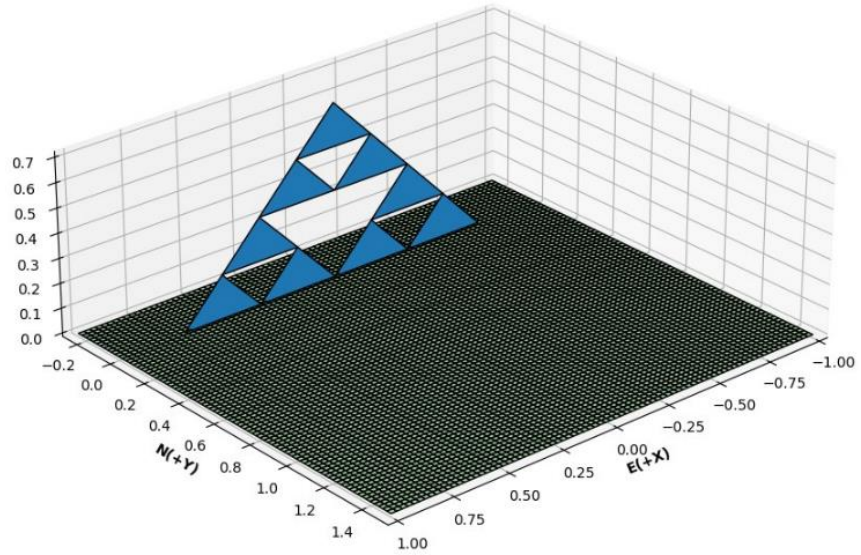

Figura 31. Visualización de las superficies implementadas

Posteriormente, se incorpora un objeto *solarResource* para la importación de los datos meteorológicos y de las coordenadas solares:

```
# Recurso solar
sr = solarResource(
    lat=36.7726,
    lon=-4.1005,
    remove_nighttime=True
)
```

Una vez definidos el recurso solar y las superficies, se implementa un objeto *Scene*:

```
# Implementación del escenario
scene = Scene(
    sr,
    receivers=[floor],
    obstacles=[sierpinski_obstacle],
    rho_ground=0.2
)
```

Finalmente, se realiza un balance de irradiación horaria centrado en el medio día solar del 1 de diciembre, y se muestran los resultados (Figura 32), en el que se puede apreciar la simetría respecto al eje norte-sur, debido a un valor nulo de azimut solar.

```
date = '12-01 12:00:00' # Fecha y hora de la simulación
# Balance de irradiación horaria, centrado en el medio día solar
scene.irradiation_balance(date, date)

# Visualización de los resultados
scene.scene_irradiation_plot(
    receivers_data=[
        {
            'mosaic': floor,
            'face':'front'
        }
    ],
    obstacles_data=[
        {'mosaic': sierpinski_obstacle,
         'color': 'tab:blue',
         'alpha':1
        }
    ],
    cmap='gnuplot2'
)
```

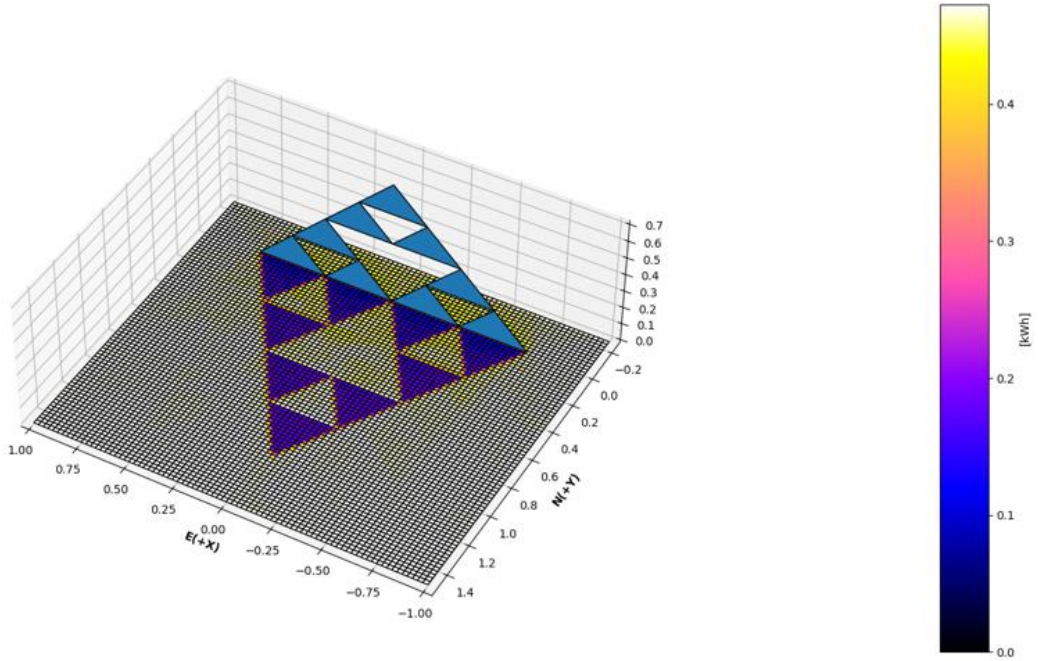

Figura 32. Distribución de la irradiación horaria, para el 1 de diciembre, a medio día solar

En la Figura 33 se muestran los resultados correspondientes a la simulación realizada para las 10:00, hora solar, del mismo día. En ella, puede apreciarse que la sombra del obstáculo se proyecta hacia el oeste y resulta más alargada, como consecuencia de la menor altura solar.

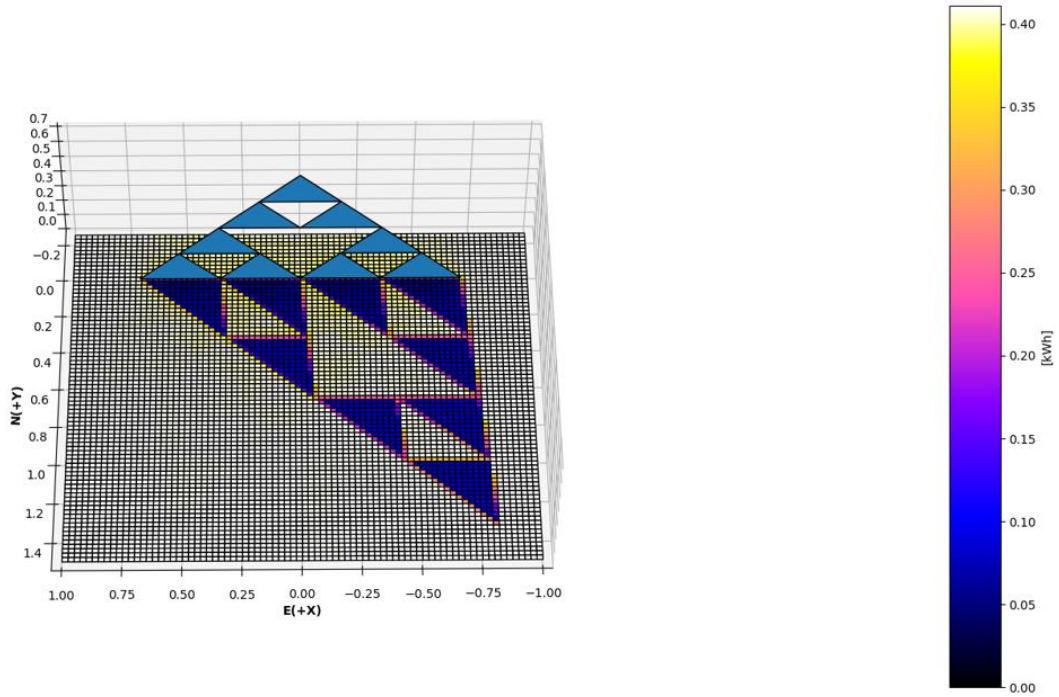

Figura 33. Distribución de la irradiación horaria, para el 1 de diciembre, 10:00 hora solar

En ambos resultados presentados se destaca la presencia de una zona de menor irradiación entorno al obstáculo, como consecuencia de la obstrucción a la radiación difusa que éste provoca.

3.2 Simulación de la irradiación mensual y anual

En este apartado se muestra un ejemplo de simulación de la irradiación mensual y anual de una parcela rectangular, en cuyo centro se hallan dos rectángulos perpendiculares a modo de obstáculos.

Como primer paso, se importan los objetos necesarios al *script* de trabajo:

```
# Importación de objetos
from pv_amp import *
```

En las siguientes líneas se implementan las superficies del escenario:

```
# Dimensiones de las superficies
w_rec = 10   # lado de la parcela
h_obs = 1.5  # altura de los obstáculos
w_obs = 7.5  # largo de los obstáculos

# Generación de los obstáculos
wall_NS = shadingRectangle(
    x_interval= [0, h_obs],
    y_interval= [0, w_obs]
)
wall_EW = shadingRectangle(
    x_interval= [0, w_obs],
    y_interval= [0, h_obs]
)
# Movimiento de los obstáculos para ubicarlos en el centro
# de la parcela
for obstacle, axis in zip([wall_NS, wall_EW], ['y', 'x']):
    obstacle.centroid_to_0()
    obstacle.rotation_3d(90, axis=axis)
    obstacle.move_3d([0, 0, h_obs/2])
    obstacle.end_def()

# Generación del receptor
floor = receivingRectangle(
x_interval=[-w_rec/2, w_rec/2],
y_interval=[-w_rec/2, w_rec/2],
    nxt=80,
    nyt=80,
    nxb=1,
    nyb=1
)
floor.end_def()
```

A través de la siguiente función, se visualizan las superficies implementadas:

```
# Visualización de las superficies
floor.plot_mosaic_3d(
    color='g',
    alpha=1,
    label='ground',
    data_env=[
        {
            'mosaic': wall_EW,
            'color': 'tab:gray',
            'alpha': 0.3,
            'label': 'wall E-W'
        },
        {
            'mosaic': wall_NS,
            'color': 'tab:blue',
            'alpha': 0.3,
            'label': 'wall N-S'
        }
    ]
)
```

La ejecución del código implementado hasta ahora abrirá una ventana, en la que se puede visualizar la superficie implementada:

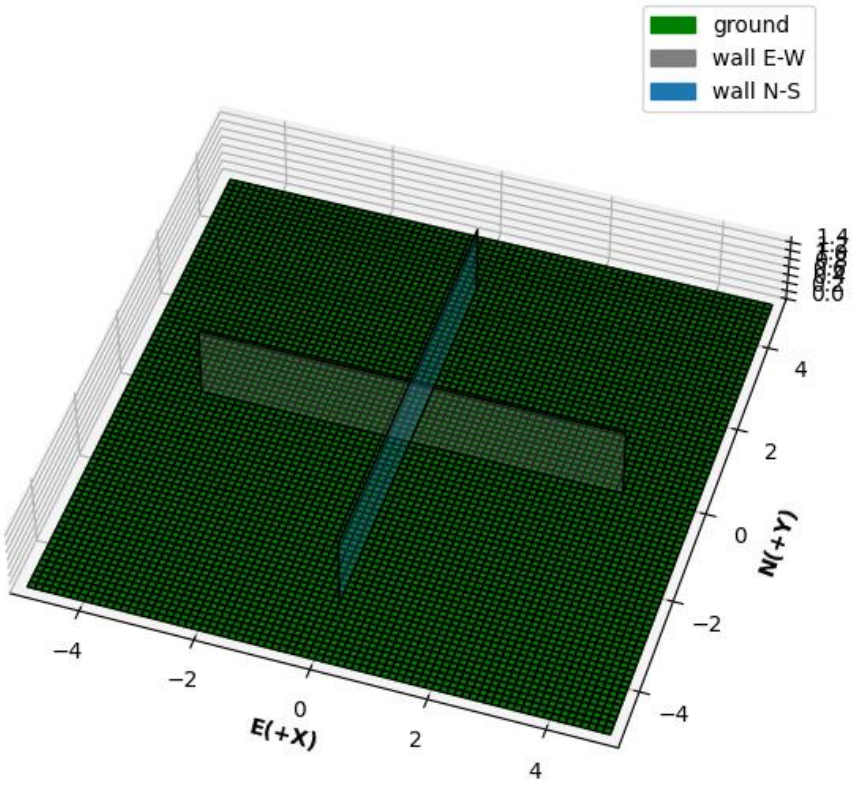

Figura 34. Visualización de las superficies implementadas

Posteriormente, se incorpora un objeto *solarResource* para la importación de los datos meteorológicos y de las coordenadas solares, y se implementa el escenario:

```
# Importación del recurso solar
sr = solarResource(
    lat=36.7726,
    lon=-4.1005,
    remove_nighttime=True
)
```

Una vez definidos el recurso solar y las superficies, se implementa un objeto *Scene*:

```
# Implementación del escenario
scene = Scene(
    sr,
    receivers=[floor],
    obstacles=[wall_NS, wall_EW],
    rho_ground=0.3
)
```

Finalmente, se realiza un balance de irradiación para el mes de junio, otro para diciembre y, por último, para un año.

```
# Intervalos temporales a simular
intervals = [
    ('06-01 00:00:00', '06-30 23:59:59'),  # junio
    ('12-01 00:00:00', '12-31 23:59:59'),  # diciembre
    ('01-01 00:00:00', '12-31 23:59:59')  # año entero
]
# Títulos de las gráficas
titles = [
    'Ground irradiation in June',
    'Ground irradiation in December',
    'Annual ground irradiation'
]
for interval, title in zip(intervals, titles):
    # Cálculo de la irradiación
    scene.irradiation_balance(interval[0], interval[1])
    # Visualización de los resultados
    scene.receiver_irradiation_plot(
        receiver=floor,
        nx_points=1, ny_points=1,
        faces='front',
        centroid_interpolation=True,
        front_title=title,
        cmap='gnuplot2')
```

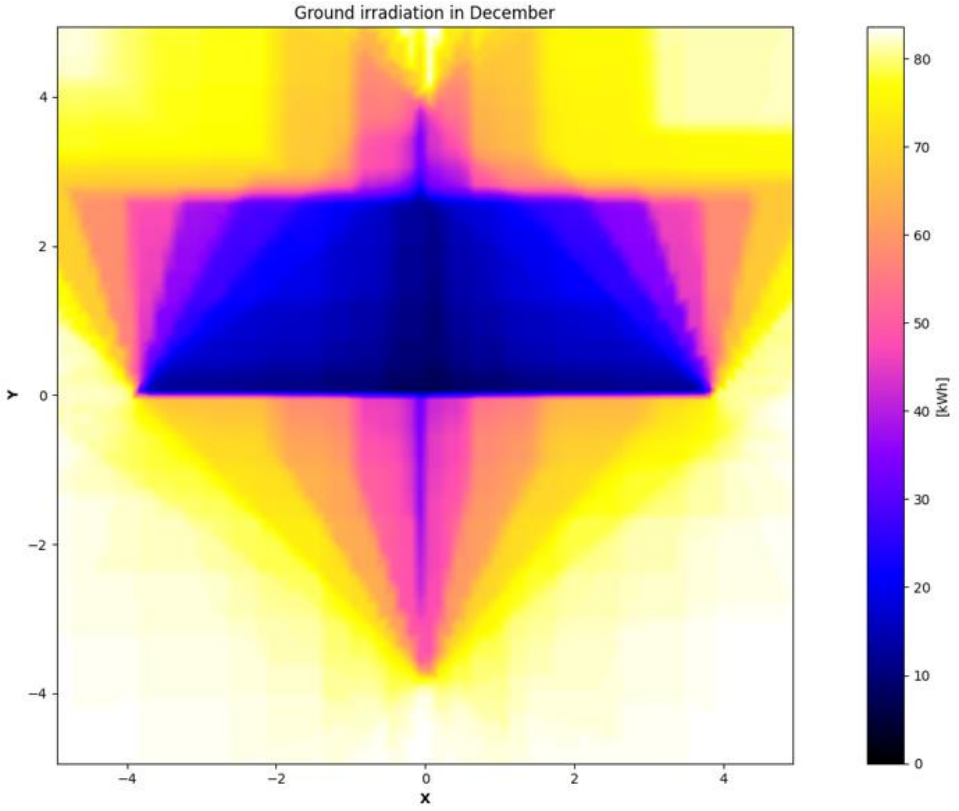

Figura 35. Distribución de la irradiación sobre el receptor en el mes de diciembre

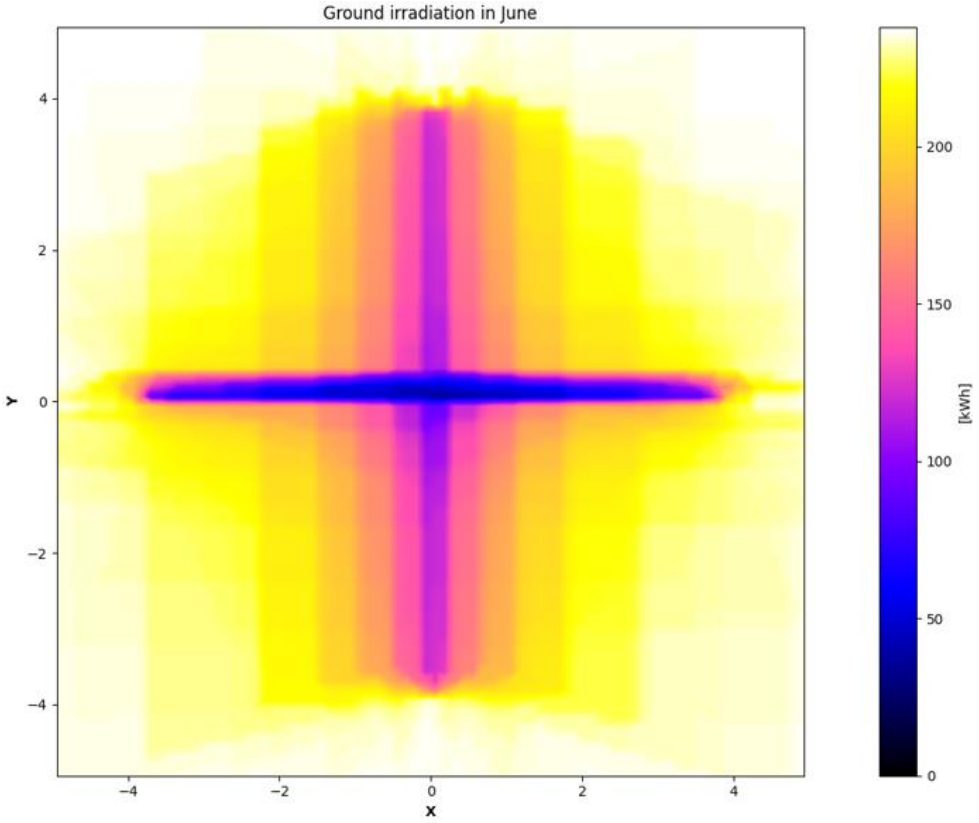

Figura 36. Distribución de la irradiación sobre el receptor en el mes de junio

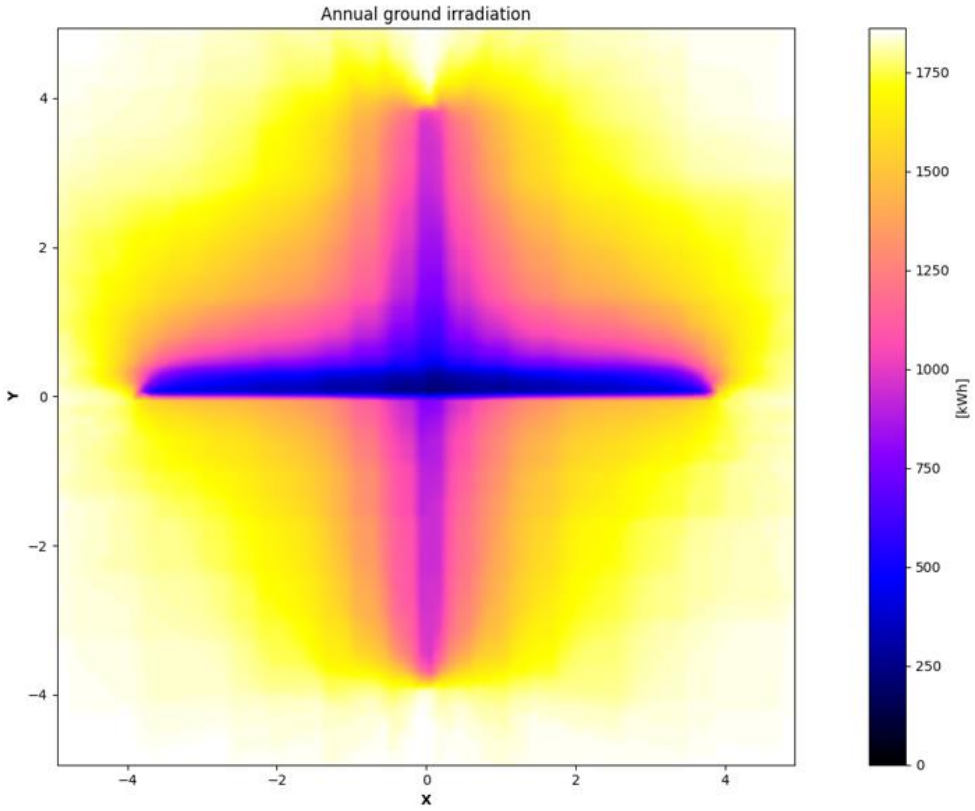

Figura 37. Distribución de la irradiación anual sobre el receptor

3.3 Simulación de una instalación de 3,6 kWp

En este ejemplo se presenta la implementación y simulación de una instalación fotovoltaica con las siguientes características:

- Instalación conformada por 18 módulos de silicio policristalino de 200 Wp, distribuidos en seis filas de tres módulos por fila, en disposición horizontal. Cada dos filas consecutivas, se forma un *string.*
- Módulos con una inclinación de 30º respecto al plano horizontal, y con orientación sur pura.
- Ancho del pasillo entre filas de 30 cm.
- La instalación cuenta con un único inversor de 3 kW de potencia nominal, monofásico, con un único *MPPT* de tres entradas. La búsqueda del punto de máxima potencia se limita al intervalo [200 V, 300 V].

En la Figura 38 se muestra la distribución de los módulos en el espacio. En esta misma figura se han representado en el mismo color aquellos módulos que se prevé que operen bajo las mismas condiciones de sombreado parcial, debido a la simetría de la distribución. De esta distribución se

infiere, además, que la instalación podrá modelarse a través de la definición de las siguientes cadenas de módulos:

- ***Cadena A*** (**x1**): conformada por módulos de tipos *A*(x3), *B*(x1), *C*(x1) y *D*(x1).
- ***Cadena B*** (**x2**): conformada por módulos de tipos *B*(x2), *C*(x2) y *D*(x2).

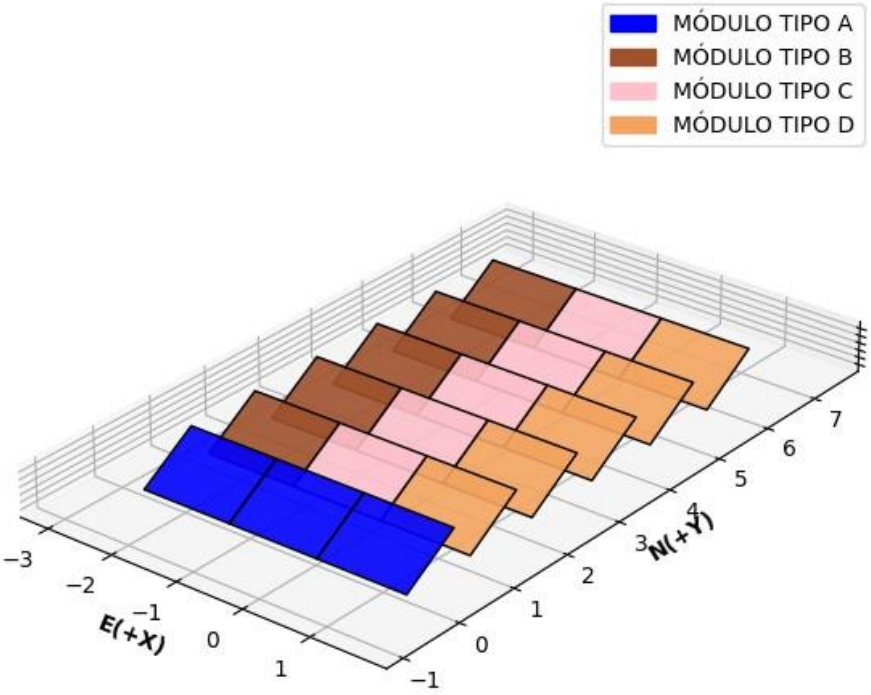

Figura 38. Distribución de los módulos fotovoltaicos e identificación de sus variedades

En las siguientes líneas de código se importan la librería y el recurso solar, y se definen los parámetros constructivos y eléctricos de los módulos:

```
# Importación de la librería y de numpy para cálculos trigonométricos
from pv_amp import *
import numpy as np

# Importación del recurso solar
sr = solarResource(
    lat=36.7726,
    lon=-4.1005,
    remove_nighttime=True
)

# Parámetros geométricos
L_v = 1.332  # lado largo del módulo
L_h = 0.992  # lado corto del módulo
beta = 30  # inclinación de los módulos [º]
sep = 0.3  # ancho del pasillo entre filas
```

```
# Datos del módulo 200W 24V ERA (Si-p)
data_ERA200W = {
    'I_sc_stc': 5.81,  #  corriente de cortocircuito [A]
    'V_oc_stc': 44.5,  # tensión de circuito abierto [V]
    'V_mpp_stc': 36.5, # tensión en el punto de máx potencia [V]
    'P_mpp_stc': 200,  # potencia máxima [W]
    'I_mpp_stc': 5.48,  # corriente en el punto de máx potencia [A]
    'beta_oc': -0.29506,  # coeficiente de variación de la Voc [%/ºC]
    'alpha_sc':  0.08558,  # coeficiente de variación de la Isc [%/ºC]
    'gamma_mpp': -0.38001,  # coeficiente de variación del Pmp [%/ºC]
    'NOCT': 45,  # temperatura del módulo en NOCT [ºC]
    'n_diodes': 3,  # número de diodos del módulo
    'cols_per_diode': 2,  # columnas por diodo
    'tiles_per_col': 12,  # baldosas por columna
    'cells_per_tile': 1,  # células fv por columna
    'nxb': 3,  # número de nodos directos en la dirección X
    'nyb': 3,  # número de nodos directos en la dirección Y
    'method': 'saloux',  # método de extracción de parámetros
    'is_bifacial': False,  # módulo monofacial
    'loss_coef': 0.05  # coeficiente de pérdidas
}
```

En las siguientes líneas se definen y ubican los módulos fotovoltaicos:

```
# Módulo tipo A (no sombreado)
mod_A = standardModule(
    **data_ERA200W,
    label='module A',
    x_interval=[0, L_h],
    y_interval=[0, L_v]
)

# Se dispone en horizontal y luego se inclina:
mod_A.rotation_3d(axis='z', angle_deg=90)
mod_A.rotation_3d(axis='x', angle_deg=beta)
mod_A.end_def()

# Se generan y almacenan los módulos de tipo B, C y D (sombreados)
# en la siguiente lista:
mod_BCD = []
delta_x = -L_v  # Posición en el eje X del primer módulo
for i, label in enumerate(['module B', 'module C', 'module D']):
    mod_i = standardModule(
        **data_ERA200W,
        label=label,
        x_interval=[0, L_h],
        y_interval=[0, L_v]
    )
```

```
    # Se disponen en horizontal y luego se inclinan:
    mod_i.rotation_3d(axis='z', angle_deg=90)
    mod_i.rotation_3d(axis='x', angle_deg=beta)

    # Se separan de la primera fila y se ubican contiguos:
    mod_i.move_3d([delta_x, sep + L_h*np.cos(np.deg2rad(beta)), 0])
    mod_i.end_def()
    delta_x += L_v # Se actualiza para el siguiente módulo
    mod_BCD.append(mod_i)  # Se almacena en el listado
```

En lo que al modelado del intercambio radiante respecta, bastaría con definir un único escenario compuesto por las dos primeras filas. No obstante, si se prevé que el intercambio de energía reflejada entre las superficies es despreciable, deberá optarse por definir un único escenario por módulo, ya que se reduce notablemente el tiempo de ejecución y los recursos de memoria empleados por la simulación. En tal caso, la primera fila de módulos (de tipo *A*) será modelada a través de un obstáculo equivalente en los escenarios de los módulos *B*, *C* y *D*, tal y como se ilustra en las siguientes líneas de código:

```
# El siguiente obstáculo simula la fila de módulos tipo A
row_A = shadingRectangle(
    x_interval=[-2*L_v, L_v],
    y_interval=[0, L_h]
)
row_A.rotation_3d(axis='x', angle_deg=beta)
row_A.end_def()

# Escena del módulo tipo A
scene_A = Scene(
    solar_resource=sr,
    receivers=[mod_A],
    obstacles=[], # sin obstáculos
    rho_ground=0.2
)

# Se generan y almacenan las escenas de los módulos
# de tipo B, C y D en la siguiente lista:
scenes_BCD = []
for mod in mod_BCD:
    scene = Scene(
        solar_resource=sr,
        receivers=[mod],
        obstacles=[row_A],  # fila de módulos tipo A
        rho_ground=0.2
    )
    scenes_BCD.append(scene)  # se almacena la escena
```

Una vez implementados los módulos fotovoltaicos y sus respectivos escenarios, se pasa a generar los dos *strings* representativos del sistema:

```
# Generación del string A: filas 1-2
# (x3)mod_A + (x1)mod_B + (x1)mod_C + (x1)mod_D
string_A = String('string A')
string_A.add_module(mod_A, 3)
for mod in mod_BCD:
    string_A.add_module(mod, 1)
string_A.end_def()

# Generación del string B: filas 3-4 y 5-6
# (x2)mod_B + (x2)mod_C + (x2)mod_D
string_B = String('string B')
for mod in mod_BCD:
    string_B.add_module(mod, 2)
string_B.end_def()
```

Se prosigue con la implementación del *MPPT* y del inversor:

```
# Implementación del único MPPT del sistema:
# (x1)string_A + (x2)string_B
mppt = mppTracker(
    label='MPPT',
    V_min=200,
    V_max=300
)
mppt.add_pole_pair(string_A, 1)
mppt.add_pole_pair(string_B, 2)
mppt.end_def()

# Implementación del inversor: (x1)mppt
inverter = Inverter(
    label='inverter',
    P_nom=3000,
    V_nom=230,
    phases=1
)
inverter.add_mppt(mppt, 1)
inverter.end_def()
```

Finalmente, se implementa un objeto *sysManager* para gestionar la simulación:

```
# Gestor del sistema: compuesto por un único inversor
sys = sysManager()
sys.add_subsystem(inverter, 1, label='system')
sys.end_def()
```

La simulación del sistema y la representación de los resultados, para distintas escalas de tiempo, se realizará a través del siguiente fragmento de código:

```
# Se simula un año completo
init_date = '01-01 00:00:00'
final_date = '12-31 23:00:00'

sys.simulate_system(
    init_date=init_date,
    final_date=final_date
)

# Gráficos con los resultados de la simulación
# Resumen anual:
sys.plot_summary_balance(
    label='global',
    title='Annual summary'
)

# Distribución mensual:
sys.plot_monthly_balance(
    label='global',
    title='Monthly distribution'
)

# Distribución horaria de la primera semana de diciembre:
sys.plot_hourly_balance(
    init_date='12-01 00:00:00',
    final_date='12-07 23:59:59',
    label='global',
    title='Hourly distribution'
)

# Irradiación anual sobre el módulo C
scenes_BCD[1].receiver_irradiation_plot(
    mod_BCD[1],
    faces='front',
    front_title='module C',
    centroid_interpolation=False
)
```

En la Figura 39 se muestra el desglose de las pérdidas de energía del sistema para un año completo de operación, y el porcentaje que suponen respecto a la energía producida en un escenario sin pérdidas. En dicha figura puede observarse que los principales focos de pérdidas se deben al sombreado de los módulos (L_{sh}) y a la incapacidad del *MPPT* de fijar la tensión que maximiza la energía de salida del generador (L_{mppt}).

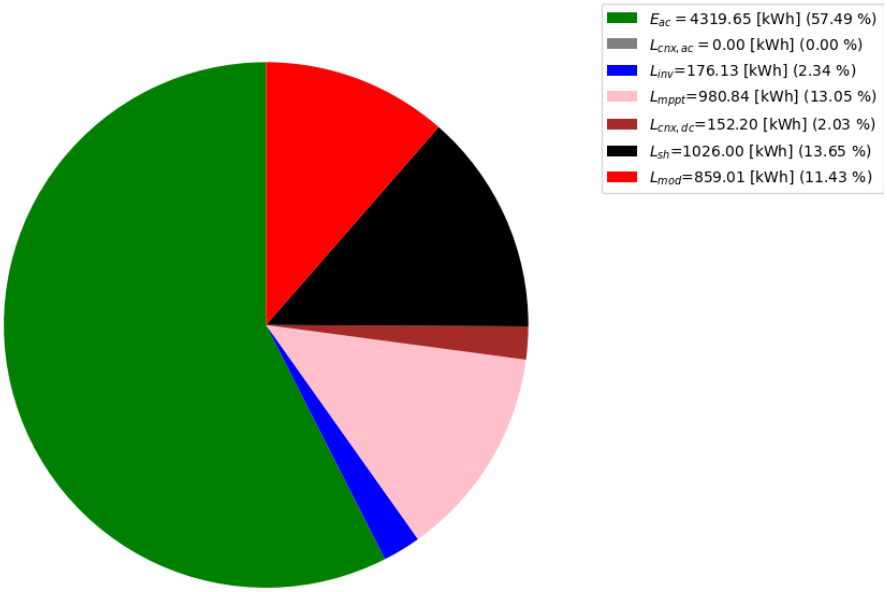

Figura 39. Resumen del balance de energía anual del sistema

En las figuras 40 y 41 se muestran la distribución mensual y horaria de las pérdidas, la producción del sistema y su *PR*. En ellas puede observarse que las pérdidas por sombreado y por la ventana de operación del *MPPT* son muy pronunciadas en los meses de invierno (hemisferio norte), pues la menor altura solar genera sombras más alargadas. Debido, además, a la instalación horizontal de los módulos, los diodos de *by-pass* correspondientes al tercio inferior de los módulos serán los que se polaricen en directa con mayor frecuencia (ver Figura 42). El disparo de dichos diodos conlleva, por lo general, que el punto de máxima potencia se alcance a tensiones menores, pudiendo llegar a encontrarse fuera de la ventana de operación del *MPPT*.

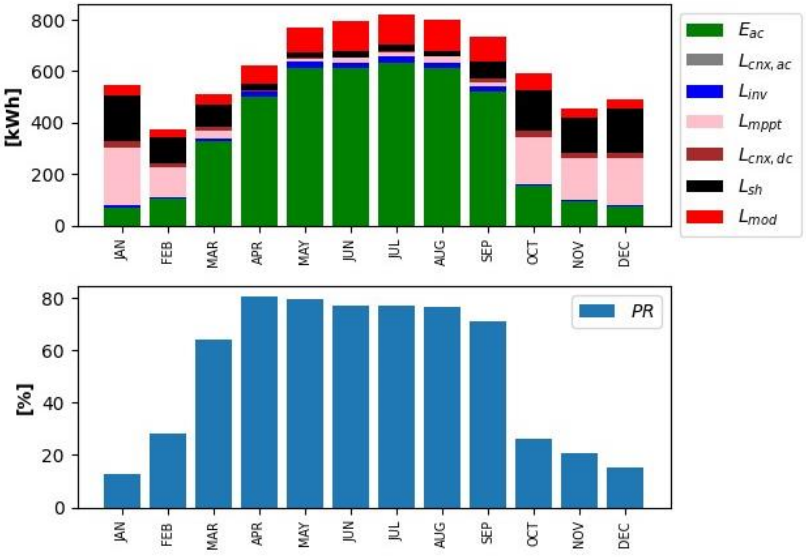

Figura 40. Distribución mensual de la producción, pérdidas y PR del sistema

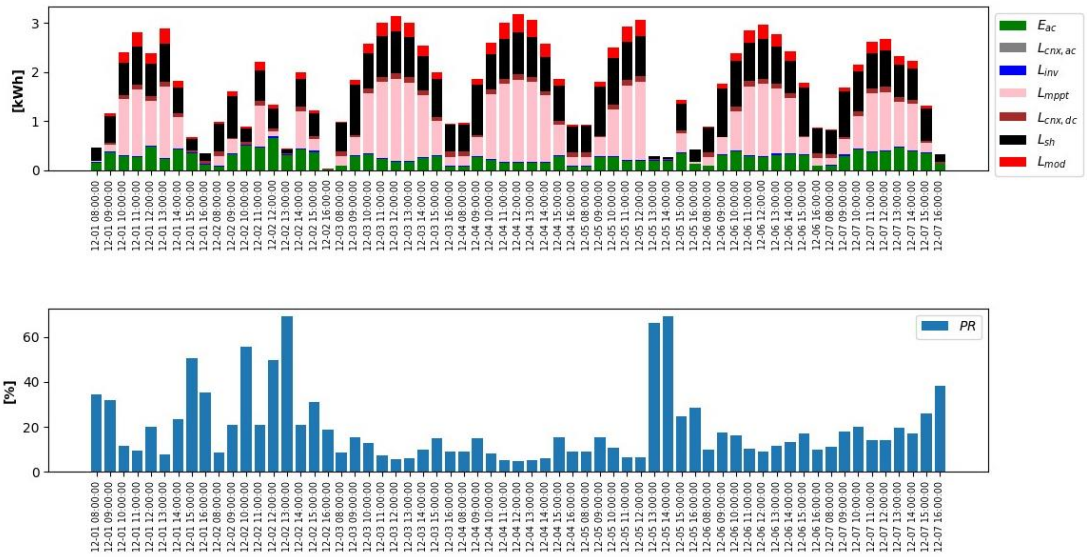

Figura 41. Distribución horaria para la primera semana de diciembre

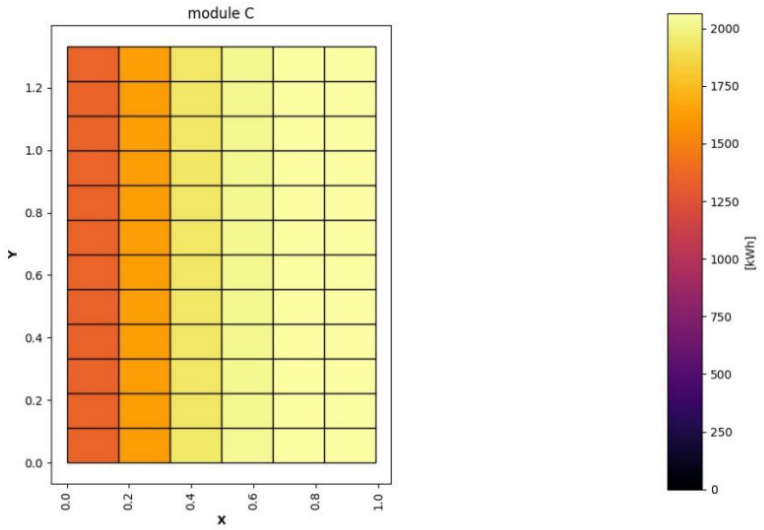

Figura 42. Distribución de la irradiación anual sobre el módulo tipo C

Para constatar lo expuesto en el párrafo anterior, se finaliza con una simulación horaria:

```
# Simulación horaria para las 10:00:00 del 1 de enero
date='01-01 10:00:00'
sys.simulate_system(init_date=date, final_date=date)
# Sombreado del módulo C
scenes_BCD[1].scene_irradiation_plot(
    receivers_data=[{'mosaic': mod_BCD[1], 'face': 'front'}],
    obstacles_data=[{'mosaic': row_A, 'color': 'b', 'alpha':0.4}],
    suptitle='Module C')
# Curvas IV/PV del string A y del MPPT
string_A.plot_curves(min_coef=0.0, max_coef=1.1)
mppt.plot_curves(min_coef=0, max_coef=1.0)
```

En las figuras 43 y 44 se muestran las curvas de la cadena tipo *A* y las propias de los módulos que la conforman. En ellas, puede observarse los escalonamientos típicos producidos por el sombreado parcial de los módulos (ver Figura 15), que se materializa en consecutivas polarizaciones en directa de los diodos de *by-pass* conforme disminuye la tensión, comenzando por los más sombreados.

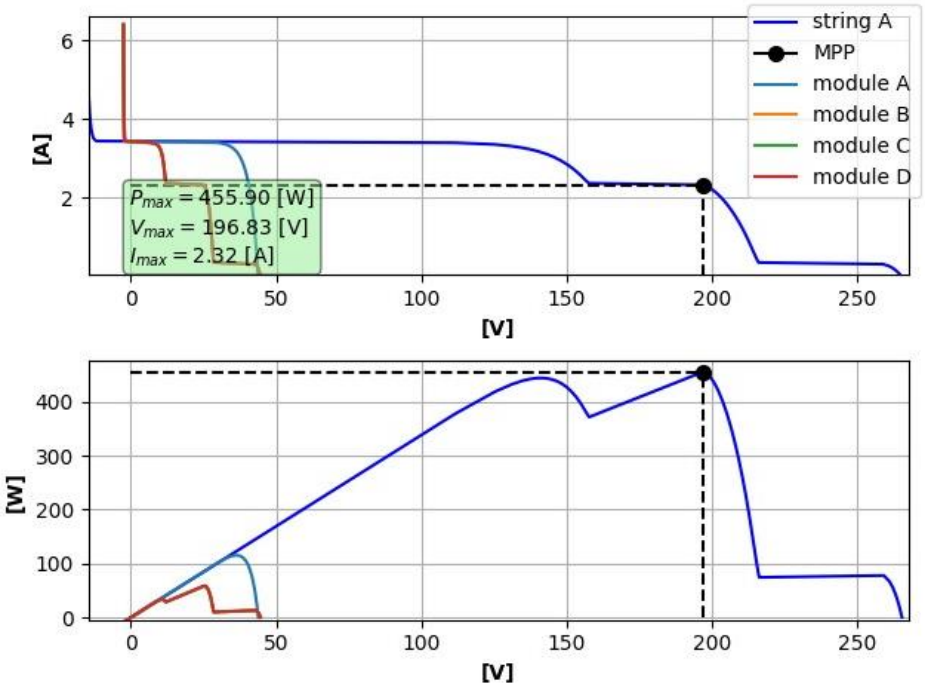

Figura 43. Curvas del string tipo A y de sus módulos integrantes

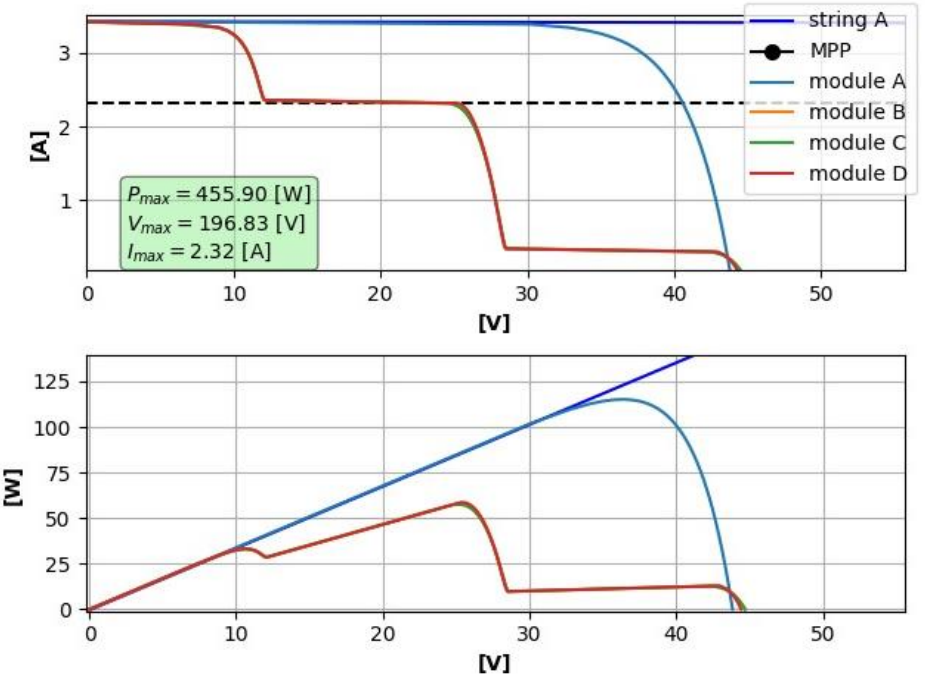

Figura 44. Detalle de las curvas de los módulos que integran al string tipo A

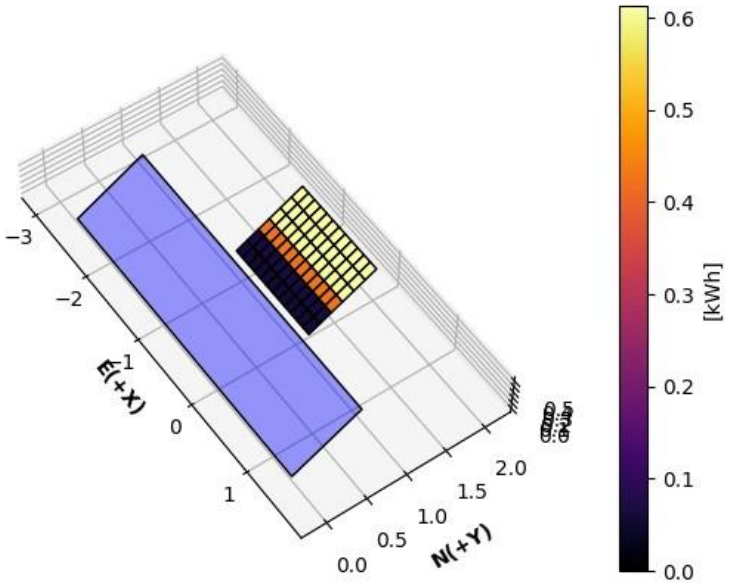

Figura 45. Visualización del sombreado parcial del módulo tipo C

En la Figura 46 se muestran las curvas del conjunto que se conecta a la entrada del *MPPT* en la fecha y hora a las que se simulan, así como su ventana de operación. En dicha figura, puede observarse cómo la tensión del punto de máxima potencia del generador se encuentra fuera de la ventana de búsqueda del *MPPT*. La tensión que es capaz de fijar el *MPPT* para extraer la máxima potencia se encuentra, en este caso, en el límite inferior de dicha ventana.

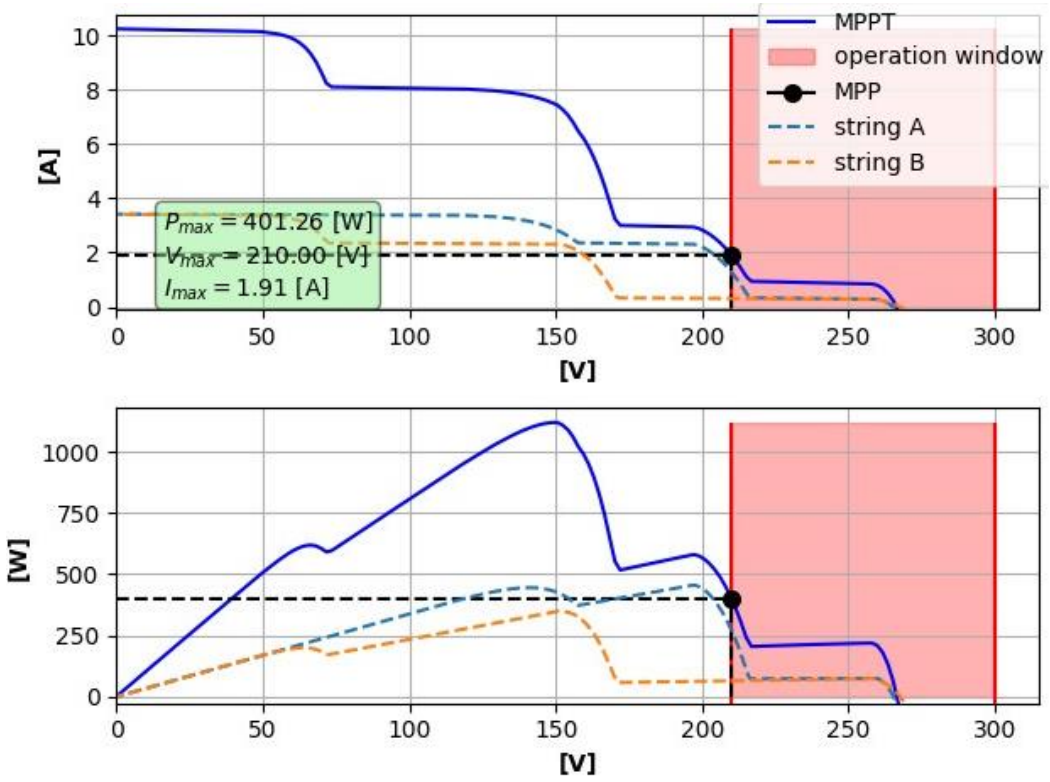

Figura 46. Curvas del generador fotovoltaico y ventana de operación del MPPT

4 CONCLUSIONES Y TRABAJOS FUTUROS

4.1 Conclusiones

El desarrollo de la librería sobre las bases metodológicas expuestas en el Capítulo 2 consigue el objetivo principal de este TFM: disponer de un software que permita modelar, simular y analizar instalaciones fotovoltaicas conectadas a red, a través de un lenguaje de programación abierto, como *Python*, y que permita introducir como parte del modelado los principales elementos agentes que pueden afectar al desempeño energético de estas instalaciones.

No obstante, para constatar la fidelidad de los resultados derivados del uso de este software, deberán aplicarse las metodologías de validación oportunas, tales como un estudio exhaustivo de comparación con respecto a otros programas de referencia, y con respecto a mediciones realizadas en instalaciones afectadas por patrones de sombreado predecibles. Estas pruebas de validación deberán ser capaces de demostrar que las desviaciones con respecto a los valores que se tomen como referencia no son fruto factores tales como:

- Imprecisión de los datos de referencia.
- Que las condiciones bajo las que se realizan las comparaciones no son equiparables.
- Falta de resolución en el mallado de las superficies, o en la escala temporal.

A su vez, estas pruebas deberán ser capaces de dar información sobre la naturaleza de estas discrepancias: en qué aspectos se producen y bajo qué circunstancias, pues ello permitirá localizar el origen de estas desviaciones, tales como la incapacidad del software para modelar factores que también influyen sobre el desempeño energético, errores en la algoritmia, o el uso de modelos que no son lo suficientemente precisos.

Por tanto, aunque se haya logrado desarrollar una herramienta capaz de modelar y simular la distribución heterogénea de la radiación sobre el generador fotovoltaico, y la consecuente respuesta eléctrica del sistema, y que los resultados presentados en el Capítulo 3 parecen ser consistentes, no puede asegurarse que sea *efectiva* hasta que se haya validado de manera adecuada.

4.2 Trabajos futuros

En virtud de lo expuesto en el apartado anterior, la validación de la librería desarrollada en este TFM amerita el emprendimiento de una línea de trabajo propia, en la que se defina e implemente de forma rigurosa un marco comparativo que permita extraer conclusiones útiles.

Por otro lado, en el Capítulo 1 se argumentó la importancia de contar con modelos lo suficientemente completos como para poder ser eficaces en el ejercicio de anticipar las prestaciones energéticas y/o económicas de un determinado diseño. Sin embargo, esta cuestión técnica es tan sólo una parte del proceso del diseño, ya que éste debe afrontar también el hecho de que unos objetivos concretos, que se pretenden alcanzar con la implementación de una instalación fotovoltaica, no conllevan un diseño unívoco de la misma, sino que existe un conjunto de ellos, potencialmente correctos, capaces de satisfacer los requerimientos exigidos. Si las variables de diseño, objetivos y restricciones se prestan a ser cuantificados y relacionados bajo un modelo matemático, puede resultar de interés aplicar técnicas de optimización para automatizar, o al menos facilitar, la restricción de los diseños a los que se consideren más adecuados, en los que *PV-AMP* podría utilizarse para evaluar el diseño de resultante de cada iteración.

En lo que respecta a la herramienta en sí, existen múltiples líneas de trabajo orientadas a incrementar y mejorar sus funcionalidades, estando la mayoría relacionadas con la mejora de los modelos implementados:

- Incorporación de modelos anisótropos para evaluar la irradiancia difusa sobre los módulos.
- Extensión de los métodos con los que se extraen los parámetros del modelo de la célula fotovoltaica.
- Incorporación de modelos adicionales para estimar la temperatura de la célula.
- Incorporación de modelos que reflejen la dependencia del rendimiento de los módulos fotovoltaicos con los niveles de irradiancia, especialmente cuando éstos son bajos.
- Ampliar el modelado del inversor para considerar los límites de intensidad máximos con los que puede operar.
- Implementar estrategias de seguimiento en los módulos fotovoltaicos.
- Adecuar el cálculo de los rendimientos y de las pérdidas a la Norma IEC-61724-1.
- Desarrollo de una interfaz gráfica para la implementación de los diseños.

ANEXO A. ALGORITMOS Y MÉTODOS NUMÉRICOS

A.1 Pertenencia de un punto al interior de un polígono

Entradas:

- Punto $P = (P_x, P_y)$
- Polígono *S* de *n* lados: $S = \{(v_{x1}, v_{y1}), (v_{x2}, v_{y2}), \dots, (v_{xn}, v_{yn})\}$

Salidas:

- Determina si el punto *P* pertenece al interior del polígono *S*.

$\delta \leftarrow \delta > 0$

$E_x \leftarrow \max(v_{x1}, v_{x2}, \dots, v_{xn}) + \delta$

$E_y \leftarrow \max(v_{y1}, v_{y2}, \dots, v_{yn}) + \delta$

$E \leftarrow (E_x, E_y)$

$cont \leftarrow 0$

FOR *i=1* **TO** *i=n* **DO:**

IF $i < n$ **THEN:**

$(A_x, A_y) \leftarrow S[i]$
$(B_x, B_y) \leftarrow S[i+1]$

ELSE:

$(A_x, A_y) \leftarrow S[i]$
$(B_x, B_y) \leftarrow S[0]$

END IF

$$(\lambda_1, \lambda_2) \leftarrow \begin{bmatrix} A_x - P_x \\ A_y - P_y \end{bmatrix} = \begin{bmatrix} E_x - P_x & A_x - B_x \\ E_y - P_y & A_y - B_y \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

IF $0 \leq \lambda_1 \leq 1$ **AND** $0 \leq \lambda_2 \leq 1$ **THEN:**

$cont \leftarrow cont + 1$

END IF

END FOR

IF $cont$ es impar **AND** $cont \geq 0$ **THEN:**

RETURN *True*

ELSE:

RETURN *False*

END IF

Algoritmo A.1. Comprobación de pertenencia de un punto al interior de un polígono

A.2 Generación de puntos interiores a un polígono

Entradas:

- Parámetros *nx* y *ny*
- Polígono *S* de *n* lados: $S = \{(v_{x1}, v_{y1}), (v_{x2}, v_{y2}), \dots, (v_{xn}, v_{yn})\}$

Salidas:

- Colección de puntos *L* interiores a *S*.

$X_{min} \leftarrow \min(v_{x1}, v_{x2}, \dots, v_{xn})$; $Y_{min} \leftarrow \min(v_{y1}, v_{y2}, \dots, v_{yn})$

$X_{max} \leftarrow \max(v_{x1}, v_{x2}, \dots, v_{xn})$; $Y_{max} \leftarrow \max(v_{y1}, v_{y2}, \dots, v_{yn})$

$L \leftarrow \emptyset$

$C_x \leftarrow X_{min} + \frac{X_{max} - X_{min}}{2 \cdot nx}$

FOR *i*=1 **TO** *i*=*nx* **DO:**

    $C_y \leftarrow Y_{min} + \frac{Y_{max} - Y_{min}}{2 \cdot ny}$

    FOR *j*=1 **TO** *j*=*ny* **DO:**

        $C \leftarrow (C_x, C_y)$

        IF *C* es interior a *S* (Algoritmo A.1) **THEN:**

            $L \leftarrow L \cup \{(C_x, C_y)\}$

        END IF

        $C_y \leftarrow C_y + \frac{Y_{max} - Y_{min}}{ny}$

    END FOR

    $C_x \leftarrow C_x + \frac{X_{max} - X_{min}}{nx}$

END FOR

RETURN *L*

Algoritmo A.2. Generación de puntos interiores a un polígono

A.3 Cálculo del factor de obstrucción de la cara de una baldosa

Entradas:

- Base y origen del sistema de referencia de la baldosa *i*: $M_i = \{\overrightarrow{X_i}, \overrightarrow{Y_i}, \overrightarrow{Z_i}\}$, $\overrightarrow{O_i}$
- Colección de nodos directos de la baldosa *i*, expresados en su referencia local: $P = \{\overrightarrow{P_{l,i}}\}$
- Colección de baldosas: $S = \{S_j\}$
- Base y origen del sistema de referencia de cada baldosa S_j: $M_j = \{\overrightarrow{X_J}, \overrightarrow{Y_J}, \overrightarrow{Z_J}\}$, $\overrightarrow{O_J}$
- Vector unitario $\overrightarrow{r_s}$ que apunta al disco solar

Salidas:

- Determina el factor Γ_i producido por el conjunto de baldosas de *S*

$\overrightarrow{n_i} \leftarrow \begin{cases} \overrightarrow{Z_i} \text{ (cara frontal)} \\ -\overrightarrow{Z_i} \text{ (cara trasera)} \end{cases}$

$\Gamma_i \leftarrow 0$

$N_p \leftarrow$ Número de elementos de *P*

$N_s \leftarrow$ Número de elementos de S

IF $\overrightarrow{r_s} \cdot \overrightarrow{n_i} \leq 0$ **THEN:**

$\Gamma_i \leftarrow 1$ (La cara no recibe irradiancia directa)

RETURN Γ_i

END IF

FOR *i*=1 **TO** *i*=N_p **DO:**

$\overrightarrow{P_g} \leftarrow \overrightarrow{P_{l,i}} \cdot (M_i)^T + \overrightarrow{O_i}$ (Expresión del nodo en la referencia global)

FOR *j*=1 **TO** *j*=N_s **DO:**

$\lambda \leftarrow \frac{\overrightarrow{Z_J} \cdot (\overrightarrow{O_J} - \overrightarrow{P_g})}{\overrightarrow{r_s} \cdot \overrightarrow{Z_J}}$ (Distancia entre $\overrightarrow{P_g}$ y el punto de intersección con el plano de S_j)

IF $\lambda \geq 0$ **THEN:**

$\overrightarrow{I_g} \leftarrow \overrightarrow{P_g} + \lambda \cdot \overrightarrow{r_s}$

$\overrightarrow{I_{l,J}} \leftarrow (\overrightarrow{I_g} - \overrightarrow{O_J}) \cdot \left((\mathrm{M_j})^T \right)^{-1}$ (Expresión de $\overrightarrow{I_g}$ en la referencia de S_j)

IF $\overrightarrow{I_{l,J}}$ es interior a S_j (Algoritmo A.1) **THEN:**

$\Gamma_i \leftarrow \Gamma_i + 1$

BREAK (El nodo es sombreado. Se continúa con otro)

END IF

END IF

END FOR

END FOR

$\Gamma_i \leftarrow \frac{\Gamma_i}{N_p}$

RETURN Γ_i

Algoritmo A.3. Cálculo del factor de obstrucción de la cara de una baldosa

A.4 Cálculo del factor de visión de un nodo respecto a su entorno

Entradas:

- Nodo *P* perteneciente a la baldosa *i*, expresado en su sistema de referencia: $\overrightarrow{P_{l,i}}$
- Base y origen del sistema de referencia de la baldosa *i*: $M_i = \{\overrightarrow{X_i}, \overrightarrow{Y_i}, \overrightarrow{Z_i}\}$, $\overrightarrow{O_i}$
- Vectores directores de las semirrectas emitidas desde el nodo *P*: $V = \{\overrightarrow{v_k}\}$
- Colección de baldosas: $S = \{S_j\}$
- Base y origen del sistema de referencia de cada baldosa S_j: $M_j = \{\overrightarrow{X_j}, \overrightarrow{Y_j}, \overrightarrow{Z_j}\}$, $\overrightarrow{O_j}$

Salidas:

- Factores de visión de *P* con respecto a las caras frontales y traseras de cada baldosa S_j, cielo y terreno: $F_{P,f}$, $F_{P,r}$, $F_{P,s}$ y $F_{P,g}$, respectivamente.

$\overrightarrow{n_i} \leftarrow \begin{cases} \overrightarrow{Z_i} \text{ (cara frontal)} \\ -\overrightarrow{Z_i} \text{ (cara trasera)} \end{cases}$

$\overrightarrow{P_g} \leftarrow \overrightarrow{P_{l,i}} \cdot (M_i)^T + \overrightarrow{O_i}$ (Expresión del nodo en la referencia global)

$N_v \leftarrow$ Número de elementos de *V*

$N_s \leftarrow$ Número de elementos de *S*

$Y \leftarrow 0$ (Peso total de los rayos emitidos desde *P*)

$F_{P,f} \leftarrow \bigcup_{i=1}^{N_s}\{0\}$; $F_{P,r} \leftarrow \bigcup_{i=1}^{N_s}\{0\}$ (Se inicializan en 0)

$F_{P,s} \leftarrow 0$; $F_{P,g} \leftarrow 0$ (Se inicializan en 0)

FOR *k*=1 **TO** *k*=N_v **DO:**

$\lambda_{min} \leftarrow \infty$ (Distancia recorrida por el rayo emitido desde *P* antes de ser interceptado)

$peso \leftarrow \overrightarrow{v_k} \cdot \overrightarrow{n_i}$ (Peso del rayo emitido)

$Y \leftarrow Y + peso$

$idx \leftarrow \emptyset$ (Índice de la baldosa que intercepta al rayo)

FOR *j*=1 **TO** *j*=N_s **DO:**

$\lambda \leftarrow \frac{\overrightarrow{Z_j} \cdot (\overrightarrow{O_j} - \overrightarrow{P_g})}{\overrightarrow{v_k} \cdot \overrightarrow{Z_j}}$ (Distancia entre $\overrightarrow{P_g}$ y el punto de intersección con el plano de S_j)

IF $\lambda \geq 0$ **THEN:**

$\overrightarrow{I_g} \leftarrow \overrightarrow{P_g} + \lambda \cdot \overrightarrow{v_k}$

$\overrightarrow{I_{l,j}} \leftarrow (\overrightarrow{I_g} - \overrightarrow{O_j}) \cdot \left((\mathrm{M_j})^T\right)^{-1}$ (Expresión de $\overrightarrow{I_g}$ en la referencia de S_j)

IF $\overrightarrow{I_{l,j}}$ es interior a S_j (Algoritmo A.1) **AND** $\lambda < \lambda_{min}$ **THEN:**

$\lambda \leftarrow \lambda_{min}$

$idx \leftarrow j$

END IF

END IF

END FOR

IF $idx \neq \emptyset$ **THEN:**

IF $\overrightarrow{v_k} \cdot \overrightarrow{Z_{idx}} < 0$ **THEN:** (Impacto con la cara frontal)

$F_{P,f}[idx] \leftarrow F_{P,f}[idx] + peso$

ELSE: (Impacto con la cara trasera)

$F_{P,r}[idx] \leftarrow F_{P,r}[idx] + peso$

END IF

ELSE: (El rayo no ha impactado contra ninguna superficie)

IF $v_{k,z} \geq 0$ **THEN:** (Rayo ascendente: impacta con el domo celeste)

$F_{P,s} \leftarrow F_{P,s} + peso$

ELSE: (Rayo descendente: impacta con el terreno)

$F_{P,g} \leftarrow F_{P,g} + peso$

END IF

END IF

END FOR

FOR j=1 **TO** j=N_s **DO:**

$F_{P,f}[j] \leftarrow \frac{F_{P,f}[j]}{Y}$

$F_{P,r}[j] \leftarrow \frac{F_{P,r}[j]}{Y}$

END FOR

$F_{P,s} \leftarrow \frac{F_{P,s}}{Y}$

$F_{P,g} \leftarrow \frac{F_{P,g}}{Y}$

RETURN $F_{P,f}$, $F_{P,r}$, $F_{P,s}$ y $F_{P,g}$

Algoritmo A.4. Cálculo del factor de visión de un nodo respecto a su entorno

A.5 Cálculo del factor de visión de una de las caras de una baldosa

Entradas:

- Colección de nodos difusos de la baldosa *i*, expresados en su referencia local: $P = \{\overrightarrow{P_{l,\iota}}\}$
- Base y origen del sistema de referencia de la baldosa *i*: $M_i = \{\overrightarrow{X_\iota}, \overrightarrow{Y_\iota}, \overrightarrow{Z_\iota}\}, \overrightarrow{O_\iota}$
- Colección de baldosas: $S = \{S_j\}$
- Base y origen del sistema de referencia de cada baldosa S_j: $M_j = \{\overrightarrow{X_J}, \overrightarrow{Y_J}, \overrightarrow{Z_J}\}, \overrightarrow{O_J}$
- Cantidad de rayos emitidos desde cada nodo difuso: N_v

Salidas:

- Factores de visión de una de las caras de la baldosa *i* con respecto a las caras frontales y traseras de cada baldosa S_j, cielo y terreno: $F_{i,f}$, $F_{i,r}$, $F_{i,s}$ y $F_{i,g}$, respectivamente.

$N_s \leftarrow$ Número de elementos de *S*

$N_p \leftarrow$ Número de elementos de *P*

$F_{i,f} \leftarrow \bigcup_{i=1}^{N_s}\{0\}$; $F_{i,r} \leftarrow \bigcup_{i=1}^{N_s}\{0\}$ (Se inicializan en 0)

$F_{i,s} \leftarrow 0$; $F_{i,g} \leftarrow 0$ (Se inicializan en 0)

$N_p \leftarrow$ Número de elementos de *P*

$V \leftarrow \emptyset$ (Vectores directores de las semirrectas N_v emitidas desde cada nodo difuso)

FOR *k*=1 **TO** *k*=N_v **DO:** (Distribución áurea de los vectores)

$z_k \leftarrow 1 - \frac{k}{N_v}$; $\varphi_k \leftarrow \pi \cdot (1+\sqrt{5}) \cdot k$; $r_k \leftarrow \sqrt{1-z_k^2}$

$\overrightarrow{v_k} \leftarrow (r_k \cdot \cos\varphi_k\,, r_k \cdot \sin\varphi_k\,, z_k) \cdot (M_i)^T$ (Expresión del vector en la referencia global)

$\overrightarrow{v_k} \leftarrow \begin{cases} \overrightarrow{v_k} \text{ (cara frontal)} \\ -\overrightarrow{v_k} \text{ (cara trasera)} \end{cases}$

$V \leftarrow V \cup \{\overrightarrow{v_k}\}$

END FOR

FOR *k*=1 **TO** *k*=N_P **DO:**

$F_{P,f}$, $F_{P,r}$, $F_{P,s}$ y $F_{P,g} \leftarrow$ Factores de visión del nodo difuso *k* (Algoritmo A.4)

$F_{i,s} \leftarrow F_{i,s} + F_{P,s}$; $F_{i,g} \leftarrow F_{i,g} + F_{P,g}$

FOR *j*=1 **TO** *j*=N_s **DO:**

$F_{i,f}[j] \leftarrow F_{i,f}[j] + F_{P,f}[j]$; $F_{i,r}[j] \leftarrow F_{i,r}[j] + F_{P,r}[j]$

END FOR

END FOR

FOR *j*=1 **TO** *j*=N_s **DO:**

$F_{i,f}[j] \leftarrow \frac{F_{i,f}[j]}{N_p}$; $F_{i,r}[j] \leftarrow \frac{F_{i,r}[j]}{N_p}$

END FOR

$F_{i,s} \leftarrow \frac{F_{i,s}}{N_p}$; $F_{i,g} \leftarrow \frac{F_{i,g}}{N_p}$

RETURN $F_{i,f}$, $F_{i,r}$, $F_{i,s}$ y $F_{i,g}$

Algoritmo A.5. Cálculo del factor de visión de una de las caras de una baldosa

A.6 Función W de Lambert

La función W de Lambert W(z) se define como la función inversa de $f(w) = we^w$, de manera que se tiene la equivalencia dada por (A1):

$$z = we^w \leftrightarrow W(z) = w \ ; \ z \in \mathbb{C} \tag{A1}$$

En el dominio de los números reales, esta función está definida en el intervalo $\left[-\frac{1}{e}, \infty\right)$ y es inyectiva en $\left(-\frac{1}{e},\ 0\right)$, ya que un mismo valor *x* está asociado a dos valores de $W(x)$. Por ello, esta función se divide en una rama inferior $W_{-1}(x) \leq -1$, en el intervalo $\left[-\frac{1}{e}, 0\right)$, y otra rama principal $W_0(x) \geq -1$, en el intervalo $\left[-\frac{1}{e}, \infty\right)$.

La rama principal $W_0(x)$ es aproximada a través de (A2) y (A3) [19]:

Para $0 \leq x \leq 9$:

$$\begin{aligned} W_0(x) = u &+ \frac{u}{1+u}p + \frac{1}{2}\frac{u}{(1+u)^3}p^2 - \frac{1}{6}\frac{u(2u-1)}{(1+u)^5}p^3 \\ &+ \frac{1}{24}\frac{u(6u^2-8u+1)}{(1+u)^7}p^4 \\ &- \frac{1}{120}\frac{u(24u^3-58u^2+22u-1)}{(1+u)^9}p^5 \end{aligned} \tag{A2}$$

$$u = \frac{x}{e} \ \ ; \ \ p = 1 - u \tag{A2.1}$$

Para $x > 9$:

$$\begin{aligned} W_0(x) = L_1 - L_2 &+ \frac{L_2}{L_1} + \frac{L_2(-2+L_2)}{2L_1^2} + \frac{L_2(6-9L_2+2L_2^2)}{6L_1^3} \\ &+ \frac{L_2(-12+36L_2-22L_2^2+3L_2^3)}{12L_1^4} \\ &+ \frac{L_2(60-300L_2+350L_2^2-125L_2^3+12L_2^4)}{60L_1^5} \end{aligned} \tag{A3}$$

$$L_1 = \ln x \ \ ; \ \ L_2 = \ln L_1 \tag{A3.1}$$

La rama inferior $W_{-1}(x)$ es aproximada a través de (A4) [30]:

$$W_{-1}(x) = -1 - \delta - \frac{2}{M_1}\left(1 - \frac{1}{1 + \frac{M_1\sqrt{\sigma/2}}{1 + M_2\sigma\exp\left(M_3\sqrt{\sigma}\right)}}\right) \tag{A4}$$

$$M_1 = 0.3361 \ ; \ M_2 = -0.0042 \ ; \ M_3 = -0.0201 \tag{A4.1}$$

A.7 Métodos para la extracción de los parámetros del modelo de un diodo

A.7.1 Método de Saloux

El método de extracción de parámetros propuesto por Saloux et al. [21] se caracteriza por considerar nulas las resistencias en serie y paralelo del modelo de un diodo. Los datos de partida vienen dados por (A5):

$$\{V_{oc}, I_{sc}, V_{mp}, I_{mp}\} \tag{A5}$$

Los parámetros se obtendrán a través de (A6) hasta (A9):

$$Rs = 0 \;\; ; \;\; R_{sh} = 0 \tag{A6}$$

$$a = \frac{V_{mp} - V_{oc}}{\ln\left(1 - \frac{I_{mp}}{I_{sc}}\right)} \tag{A7}$$

$$I_{ph} = I_{sc} \tag{A8}$$

$$I_s = \frac{I_{sc}}{e^{\frac{V_{oc}}{a}} - 1} \tag{A9}$$

A.7.2 Método de Batzelis

El método propuesto por Batzelis et al. [22] parte de los datos (A10):

$$\{V_{oc,STC}, V_{oc}, I_{sc}, V_{mp}, I_{mp}, T_{cell}, \alpha, \beta\} \tag{A10}$$

Los parámetros se obtendrán a través de (A11) hasta (A18), en las que T^* denota que la temperatura debe estar expresada en Kelvin:

$$\delta_0 = \frac{1 - \beta T^*_{STC}}{50.1 - \alpha \cdot T^*_{STC}} \tag{A11}$$

$$\delta = \delta_0 \frac{V_{oc,STC}}{V_{oc}} \frac{(T^*_{cell})}{T^*_{STC}} \tag{A12}$$

$$w = W_0\left(e^{\frac{1}{\delta}+1}\right) \tag{A13}$$

$$a = \delta V_{oc} \tag{A14}$$

$$R_s = \frac{a(w-1) - V_{mp}}{I_{mp}} \tag{A15}$$

$$R_{sh} = \frac{a(w-1)}{I_{sc}\left(1-\frac{1}{w}\right) - I_{mp}} \tag{A16}$$

$$I_{ph} = \left(1 + \frac{R_s}{R_{sh}}\right) I_{sc} \tag{A17}$$

$$I_s = I_{ph} e^{-\frac{1}{\delta}} \tag{A18}$$

A.7.3 Método de Cubas

Los datos de partida del método propuesto por Cubas et al. [18] vienen dados por (A19), siendo *n* el factor de idealidad del diodo. En *PV-AMP*, se asume un valor de *n* = 1.1 [18] [17].

$$\{V_{oc}, I_{sc}, V_{mp}, I_{mp}, T_{cell}, n\} \tag{A19}$$

Los parámetros se obtendrán a través de (A20) hasta (A28), donde T^* denota que la temperatura debe estar expresada en Kelvin, *k* la constante de Boltzmann ($1.38 \cdot 10^{-23}$ [J/K^2]) y q la carga del electrón ($1.60 \cdot 10^{-19}$ [C]):

$$a = \frac{nkT_{cell}^*}{q} \tag{A20}$$

$$A = \frac{a}{I_{mp}} \tag{A21}$$

$$B = -\frac{V_{mp}(2I_{mp} - I_{sc})}{\left(V_{mp}I_{sc} + V_{oc}(I_{mp} - I_{sc})\right)} \tag{A22}$$

$$C = -\frac{2V_{mp} - V_{oc}}{a} + \frac{V_{mp}I_{sc} - V_{oc}I_{mp}}{\left(V_{mp}I_{sc} + V_{oc}(I_{mp} - I_{sc})\right)} \tag{A24}$$

$$D = \frac{V_{mp} - V_{oc}}{a} \tag{A25}$$

$$R_s = A \cdot \left(W_{-1}(Be^C) - (D + C)\right) \tag{A25}$$

$$R_{sh} = \frac{(V_{mp} - I_{mp}R_s)(V_{mp} - R_s(I_{sc} - I_{mp}) - a)}{(V_{mp} - I_{mp}R_s)(I_{sc} - I_{mp}) - aI_{mp}} \tag{A26}$$

$$I_s = \left(I_{sc}\left(1 + \frac{R_s}{R_{sh}}\right) - \frac{V_{oc}}{R_{sh}}\right) e^{-\frac{V_{oc}}{a}} \tag{A27}$$

$$I_{ph} = \left(1 + \frac{R_s}{R_{sh}}\right) I_{sc} \tag{A28}$$

ANEXO B. COMANDOS DE LA LIBERÍA

B.1 Instalación e importación de la librería

Los ficheros *.py* que conforman la librería se encuentran en la carpeta denominada *pv_amp-base*, adjunta a la entrega de este documento. Esta librería puede instalarse a través de la introducción del siguiente comando en la terminal del sistema operativo que se esté utilizando, particularizándolo a la ruta relativa en la que se encuentra la carpeta:

```
python -m pip install -e ruta_relativa_pv_amp-base
```

Finalmente, para importar todos los objetos implementados en la librería en un *script*, se incluirá en él la siguiente línea de código:

```
from pv_amp import *
```

B.2 Recurso solar

B.2.1 Inicialización a través de un fichero externo

```
solarResource(file_name)
```

- ***file_name*** (***str***): nombre del fichero, incluyendo la extensión, que contiene los datos. Este fichero debe ubicarse en la misma carpeta que en la que se está trabajando, y debe contener las siguientes columnas de datos horarios:
  - ***time(UTC)***: columna con las fechas y horas, que se incrementan a razón de una hora. El formato debe ser *"mm-dd HH[ ]:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.
  - ***T2m***: temperatura ambiente, en ºC.
  - ***G(h)***: irradiancia global en el plano horizontal, en W/m^2.
  - ***Gb(n)***: irradiancia directa en plano normal a los rayos solares, en W/m^2.
  - ***Gd(h)***: irradiancia difusa del cielo en plano horizontal, en W/m^2.
  - ***WS10m***: velocidad del viento, en m/s.
  - ***solar_ray_x***: componente *X* del vector unitario de los rayos solares, con origen en el disco solar, y sentido hacia la tierra. El semieje positivo *X* se corresponde con el punto cardinal este.

- ***solar_ray_y***: componente *Y* del vector unitario de los rayos solares, con origen en el disco solar, y sentido hacia la tierra. El semieje positivo *Y* se corresponde con el punto cardinal norte.
- ***solar_ray_z***: componente *Z* del vector unitario de los rayos solares, con origen en el disco solar, y sentido hacia la tierra. El semieje positivo *Z* se corresponde con la normal del terreno. En horas diurnas, el valor de esta componente será negativo.

B.2.2 Inicialización a través de la especificación de las coordenadas geográficas

```
solarResource(lat, long, usehorizon, remove_nighttime)
```

- ***lat*** (***float***): latitud, en grados decimales. Norte: positivo; sur: negativo.
- ***long*** (***float***): longitud, en grados decimales. Este: positivo; oeste: negativo.
- ***usehorizon*** (***bool***): Si su valor es *True*, los datos de radiación horarios tendrán en cuenta el efecto del perfil del horizonte lejano: se anulan las componentes directas de la radiación en aquellas horas en las que el Sol está tras dicho perfil.
- ***remove_nighttime*** (***bool***): Si valor es *True*, no se importarán los datos horarios nocturnos.

B.2.3 Exportación a un fichero externo

El siguiente método genera un fichero *.csv* al que se exportarán los datos horarios meteorológicos y las coordenadas solares. Este fichero se genera en la misma carpeta que en la que se está trabajando, y tendrá la estructura descrita en el apartado anterior, por lo que podrá utilizarse en simulaciones posteriores como un fichero de importación.

```
solarResource.export_data(file_name)
```

- ***file_name*** (***str***): nombre del fichero, incluyendo la extensión.

B.3 Superficies

B.3.1 Inicializaciones de mosaicos genéricos

`Obstáculo:`	`shadigMosaic(tiles_data)`
`Receptor:`	`receivingMosaic(tiles_data, rho_front, rho_rear)`

- ***tiles_data*** (***array***): secuencia con los datos de cada baldosa que integra al mosaico, en forma de *diccionarios*. Los campos de cada *diccionario* dependerán de si la baldosa es rectangular genérica:
    - En el caso de que la baldosa sea rectangular, el *diccionario* tendrá los siguientes campos:
        - ***type*** (***str***): se le asignará el valor constante *'rectangular'*.
        - ***x_interval*** (***array***): intervalo en el que se definen los lados paralelos al eje X del rectángulo → [*x_min*, *x_max*].
        - ***y_interval*** (***array***): intervalo en el que se definen los lados paralelos al eje Y del rectángulo → [*y_min*, *y_max*].
    - En el caso de que la baldosa sea genérica, el *diccionario* tendrá los siguientes campos:
        - ***type*** (***str***): se le asignará el valor constante *'generic'*.
        - ***vertices_2d*** (***array***): secuencia en la que cada elemento son las coordenadas de un vértice de la baldosa asociada, en la forma (X, Y) → [(x_1, y_1), (x_2, y_2), ... , (x_n, y_n)].
    - Si la baldosa pertenece a un mosaico de tipo *receivingMosaic*, deberán añadirse al diccionario, además, los siguientes campos:
        - ***nxb*** (***int***): parámetro que controla la densidad de nodos en la dirección X empleados para el cálculo de la obstrucción a la radiación directa.
        - ***nyb*** (***int***): parámetro con la misma finalidad que *nxb*, pero en la dirección Y.
        - ***nxd*** (***int***): parámetro que controla la densidad de nodos en la dirección X empleados para el cálculo de los factores de visión.
        - ***nyd*** (***int***): parámetro con la misma finalidad que *nxd*, pero en la dirección Y.
- ***rho_front*** (***float***): reflectividad de la cara frontal del receptor (entre 0 y 1).
- ***rho_rear*** (***float***): reflectividad de la cara trasera del receptor (entre 0 y 1).

B.3.2 Inicialización de mosaicos rectangulares

`Obstáculo:`	`shadigRectangle(x_interval, y_interval)`
`Receptor:`	`receivingRectangle(x_interval, y_interval, …)`

- ***x_interval*** (***array***): intervalo en el que se definen los lados paralelos al eje X del mosaico rectangular → [*x_min*, *x_max*].
- ***y_interval*** (***array***): intervalo en el que se definen los lados paralelos al eje Y del mosaico rectangular → [y_*min*, y_*max*].
- ***nxt*** (***int***): número de columnas de baldosas en las que se divide el receptor rectangular.
- ***nyt*** (***int***): número de filas de baldosas en las que se divide el receptor rectangular.

- ***nxb*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para el cálculo de la obstrucción a la radiación directa.
- ***nyb*** (***int***): parámetro con la misma finalidad que *nxb*, pero en la dirección *Y*.
- ***nxd*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para el cálculo de los factores de visión.
- ***nyd*** (***int***): parámetro con la misma finalidad que *nxd*, pero en la dirección *Y*.
- ***rho_front*** (***float***): reflectividad de la cara frontal del receptor (entre 0 y 1).
- ***rho_rear*** (***float***): reflectividad de la cara trasera del receptor (entre 0 y 1).

B.3.3 Inicialización de receptores convexos

`Receptor:`	`receivingConvex(…)`

- ***vertices_2d*** (***array***): secuencia en la que cada elemento son las coordenadas de un vértice del receptor, en la forma $(X, Y) \rightarrow [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$.
- ***nxt*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para la generación de las baldosas.
- ***nyt*** (***int***): parámetro con la misma finalidad que *nxt*, pero en la dirección *Y*.
- ***nxb*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para el cálculo de la obstrucción a la radiación directa.
- ***nyb*** (***int***): parámetro con la misma finalidad que *nxb*, pero en la dirección *Y*.
- ***nxd*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para el cálculo de los factores de visión.
- ***nyd*** (***int***): parámetro con la misma finalidad que *nxd*, pero en la dirección *Y*.
- ***rho_front*** (***float***): reflectividad de la cara frontal del receptor (entre 0 y 1).
- ***rho_rear*** (***float***): reflectividad de la cara trasera del receptor (entre 0 y 1).

B.3.4 Movimientos y finalización de la implementación

Los métodos que implementan los mosaicos para cada tipo de desplazamiento son los siguientes:

- ***Traslaciones en el plano.***

  Desplaza una cantidad $(\Delta x, \Delta y)$ a todos los elementos de un mosaico, en el sistema de referencia local.

`mosaic.move_2d(delta_P2d)`

  - ***delta_P2d*** (***array***): vector de desplazamiento $(\Delta x, \Delta y)$.

- ***Traslaciones en el espacio.***

  Desplaza una cantidad $(\Delta x, \Delta y, \Delta z)$ a todos los elementos de un mosaico, en el sistema de referencia global.

```
mosaic.move_3d(delta_P3d)
```

  - ***delta_P3d*** (***array***): vector de desplazamiento $(\Delta x, \Delta y, \Delta z)$.

- ***Traslación del centroide al origen.***

  Calcula y desplaza el centroide de un mosaico al origen del sistema de referencia global.

```
mosaic.centroid_to_0()
```

- ***Rotaciones en el plano.***

  Rota a un mosaico en torno al eje *Z* de su sistema de referencia local, bajo la regla de la mano derecha.

```
mosaic.rotation_2d(angle_deg)
```

  - ***angle_deg*** (***float***): ángulo de giro, expresado en grados decimales.

- ***Rotaciones en el espacio.***

  Rota a un mosaico en torno a un determinado eje, bajo la regla de la mano derecha.

```
mosaic.rotation_3d(angle_deg, axis, v)
```

  - ***angle_deg*** (***float***): ángulo de giro, expresado en grados decimales.
  - ***axis*** (***str***): Eje del sistema de referencia global en torno al cual se realiza la rotación. Los valores válidos son *'x'*, *'y'*, y *'z'*. En caso de realizarse en torno a otro eje, este campo debe quedar sin especificar (*axis=None*), y definirse el campo *v*.
  - ***v*** (***array***): vector unitario, con origen en el origen del sistema de referencia global, en torno al cual se realiza la rotación, bajo la regla de la mano derecha.

- ***Finalización del proceso.***

  Cuando los movimientos de un mosaico han sido finalizados, se debe indicar a través del siguiente método:

```
mosaic.end_def()
```

  Una vez invocado, no se podrán realizar más modificaciones al mosaico.

B.3.5 Visualización de las superficies en el espacio

El siguiente método abre una ventana con la representación 3D de las baldosas de un mosaico, a la que se le pueden añadir más superficies.

`mosaic.plot_mosaic_3d(color, alpha, data_env, label, title)`

- ***color*** (***str***): color del mosaico (opcional). El listado puede consultarse en este enlace.
- ***alpha*** (***float***): opacidad del mosaico, a efectos de visualización: 0→transparente, 1→opaco (opcional).
- ***data_env*** (***array***): Secuencia opcional para graficar otros mosaicos en el espacio. Cada elemento de la secuencia es un diccionario con los siguientes campos:
  - ***mosaic*** (***receivingMosaic*** | ***shadingMosaic***): instancia del mosaico adicional.
  - ***color*** (***str***): color (opcional).
  - ***alpha*** (***float***): opacidad, a efectos de visualización (opcional).
  - ***label*** (***str***): etiqueta con la que figura en la leyenda del gráfico (opcional).
- ***label*** (***str***): etiqueta con la que figura en la leyenda del gráfico (opcional).
- ***title*** (***str***): título del gráfico (opcional).

B.4 Escenario del balance radiante

B.4.1 Inicialización

`Scene(…)`

- ***solar_resource*** (***solarResource***): objeto *solarResource* que contiene los datos meteorológicos de la escena.
- ***receivers*** (***array***): secuencia con los receptores de la escena. De forma previa, deben haberse declarado como finalizados a través de sus respectivos métodos *end_def()*.
- ***obstacles*** (***array***): secuencia con los obstáculos de la escena. De forma previa, deben haberse declarado como finalizados a través de sus respectivos métodos *end_def()*.
- ***rho_ground*** (***float***): albedo del terreno, comprendido entre 0 y 1.

B.4.2 Simulación de la irradiancia

Para calcular la irradiación horaria o la irradiancia[8] sobre las superficies de una escena, se hará uso del siguiente método:

`Scene.irradiance_balance(date)`

- ***date*** (***str***): fecha y hora para la que se desea realizar el cálculo de la irradiancia. El formato debe ser *"mm-dd[ ]HH:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.

[8] Al considerarse la irradiancia constante a intervalos de una hora, y al utilizarse unidades de Wh, el valor de la irradiancia coincide con el de la irradiación horaria.

Se deben tener en cuenta los siguientes puntos:

- Si los datos meteorológicos han sido cargados desde un fichero *.csv*, esta fecha debe figurar en la columna "*time(UTC)"* del mismo.
- En caso de haber sido cargados a través de la *API* de *PVGIS,* los datos están referenciados al comienzo de cada hora, por lo que el formato será *"mm-dd[ ]HH:00:00"*. Además, si dichos datos se han filtrado para eliminar las horas nocturas, se debe asegurar que la hora sea diurna.

B.4.3 Simulación de la irradiación

Para calcular la irradiación sobre las superficies de una escena, durante un determinado intervalo temporal, se hará uso del siguiente método:

```
Scene.irradiation_balance(init_date, final_date)
```

- ***init_date*** (***str***): fecha y hora del comienzo del intervalo temporal. Si dicha fecha no figura en los datos meteorológicos cargados, se pasará a la siguiente más cercana.
- ***final_date*** (***str***): fecha y hora de finalización del intervalo temporal. Si dicha fecha no figura en los datos meteorológicos cargados, se pasará a la inmediatamente anterior.

Se deben tener en cuenta los siguientes puntos:

- El formato de las fechas debe ser *"mm-dd[ ]HH:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.
- Los extremos de los intervalos están incluidos.
- El cálculo asume que los datos cargados son horarios ($\Delta t \rightarrow 1$ hora), y que los propios de irradiancia están expresados en W/m^2.

B.4.4 Visualización de la distribución de la irradiación

- ***Distribución de la irradiación sobre un receptor.***

  Muestra la distribución de la irradiación sobre un receptor de la escena, en base a los resultados del último balance de irradiación realizado. Esta distribución se representará a través de un *mapa de calor*, donde las coordenadas *X* e *Y* están referidas al sistema de referencia local del receptor.

```
Scene.receiver_irradiation_plot(…)
```

  - ***receiver*** (***receivingMosaic***): instancia del receptor.
  - ***faces*** (***str***): especificación de la cara del receptor a la que estará referida la distribución de la irradiación. Para visualizar la cara frontal, se especificará el valor *'front'*; para la

trasera: *'rear'*. Si se desea mostrar dos gráficos para sendas caras, se especificará el valor *'both'*.

- ***centroid_interpolation*** (***bool***): en caso de valor *False*, se representará la irradiación discretizada sobre cada baldosa. Si vale *True*, y el receptor posee cuatro o más baldosas, se asociará la irradiación de cada una de ellas a sus respectivos centroides. La irradiación de los puntos intermedios entre los centroides será calculada a través de una interpolación lineal, dando así a la gráfica un aspecto *continuo*. La densidad de estos puntos entre centroides se controla a través de los siguientes campos:
  - ***nx_points*** (***int***): número de puntos entre centroides, en la dirección *X* del sistema de referencia local.
  - ***ny_points*** (***int***): número de puntos entre centroides, en la dirección *Y* del sistema de referencia local.
- ***suptitle*** (***str***): título principal del gráfico (opcional).
- ***front_title*** (***str***): título del gráfico asociado a la cara frontal (opcional).
- ***rear_title*** (***str***): título del gráfico asociado a la cara trasera (opcional).
- ***cmap*** (***str***): estilo cromático del mapa de irradiación. Los valores válidos para este campo pueden consultarse en este enlace.

- ***Distribución de la irradiación sobre varios receptores***

Muestra la distribución de la irradiación sobre varios receptores, en base a los resultados del último balance de irradiación realizado. Esta distribución se representará sobre las superficies a través de un *mapa de calor*, visualizadas en el espacio tridimensional.

`Scene.scene_irradiation_plot(…)`

- ***receivers_data*** (***array***): secuencia en la que se especifican las caras de los receptores a representar. Cada elemento de la secuencia es un diccionario con los siguientes campos:
  - ***mosaic*** (***receivingMosaic***): instancia del receptor.
  - ***face*** (***str***): cara del receptor cuya distribución de irradiación se desea representar. Para visualizar la frontal: *'front'*; para la trasera: *'rear'*.
- ***obstacles_data*** (***array***): secuencia en la que se especifican los datos para la representación de los obstáculos. Cada elemento de la secuencia es un diccionario con los siguientes campos:
  - ***mosaic*** (***shadingMosaic*** | ***receivingMosaic***): instancia del obstáculo.
  - ***color*** (***str***): color del mosaico (opcional). El listado puede consultarse en este enlace.
  - ***alpha*** (***float***): opacidad del obstáculo, a efectos de visualización: 0→transparente, 1→opaco (opcional).
- ***suptitle*** (***str***): título principal del gráfico (opcional).

- ***cmap*** (***str***): estilo cromático del mapa de irradiación. Los valores válidos para este campo pueden consultarse en este enlace.

- ***Animación de la distribución de la irradiación sobre un receptor***

  Muestra una animación en la que se aprecia la evolución de la distribución de la irradiación sobre un receptor de la escena, donde cada fotograma se corresponde con un intervalo de tiempo específico. La distribución se representará a través de un *mapa de calor*, donde las coordenadas *X* e *Y* están referidas al sistema de referencia local del receptor.

```
Scene.receiver_irradiation_animation()
```

  - ***receiver*** (***receivingMosaic***): instancia del receptor.
  - ***faces*** (***str***): especificación de la cara del receptor a la que estará referida la distribución de la irradiación. Para visualizer la de la cara frontal, se especificará el valor *'front'*; para la trasera: *'rear'*. Si se desea mostrar dos gráficos para sendas caras, se especificará el valor *'both'*.
  - ***date_ranges*** (***array***): secuencia en la que se especifican los intervalos de tiempo para los que se calcula la irradiación de cada fotograma de la animación. Cada elemento es una lista de dos elementos, en la que el primero es la fecha de inicio de un intervalo, y el segundo la fecha de finalización. Ambas fechas se incluyen en el cálculo de la irradiación, y deben tener el siguiente formato: *"mm-dd[ ]HH:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.
  - ***suptitle*** (***str***): título principal del gráfico (opcional).
  - ***front_title*** (***str***): título del gráfico asociado a la cara frontal (opcional).
  - ***rear_title*** (***str***): título del gráfico asociado a la cara trasera (opcional).
  - ***show_date_ranges*** (***str***): si vale *True*, se mostrará en la animación el intervalo temporal asociado a cada fotograma de la animación.
  - ***cmap*** (***str***): estilo cromático del mapa de irradiación (opcional). Los valores válidos para este campo pueden consultarse en este enlace.

- ***Animación de la distribución de la irradiación sobre varios receptores***

  Muestra una animación en la que se aprecia la evolución de la distribución de la irradiación sobre los receptores de la escena, donde cada fotograma se corresponde con un intervalo de tiempo específico. Esta distribución se representará sobre las superficies a través de un *mapa de calor*, visualizadas en el espacio tridimensional.

```
Scene.scene_irradiation_animation(…)
```

- ***receivers_data*** (***array***): secuencia en la que se especifican las caras de los receptores a representar. Cada elemento de la secuencia es un diccionario con los siguientes campos:
  - ***mosaic*** (***receivingMosaic***): instancia del receptor.
  - ***face*** (***str***): cara del receptor cuya distribución de irradiación se desea representar. Para visualizar la frontal: *'front'*; para la trasera: *'rear'*.
- ***obstacles_data*** (***array***): secuencia en la que se especifican los datos para la representación de los obstáculos. Cada elemento de la secuencia es un diccionario con los siguientes campos:
  - ***mosaic*** (***shadingMosaic*** | ***receivingMosaic***): instancia del obstáculo.
  - ***color*** (***str***): color del mosaico (opcional). El listado puede consultarse en este enlace.
  - ***alpha*** (***float***): opacidad del obstáculo, a efectos de visualización: 0→transparente, 1→opaco (opcional).
- ***date_ranges*** (***array***): secuencia en la que se especifican los intervalos de tiempo para los que se calcula la irradiación de cada fotograma de la animación. Cada elemento es una lista de dos elementos, en la que el primero es la fecha de inicio de un intervalo, y el segundo la fecha de finalización. Ambas fechas se incluyen en el cálculo de la irradiación, y deben tener el siguiente formato: *"mm-dd[ ]HH:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.
- ***suptitle*** (***str***): título principal del gráfico (opcional).
- ***cmap*** (***str***): estilo cromático del mapa de irradiación (opcional). Los valores válidos para este campo pueden consultarse en este enlace.

B.5 Módulos fotovoltaicos

B.5.1 Inicialización

`Módulo estándar:`	`standardModule(…)`
`Módulo de célula partida:`	`halfCellModule(…)`

- ***x_interval*** (***array***): intervalo en el que se definen los lados paralelos al eje *X* del módulo fotovoltaico → [*x_min*, *x_max*].
- ***y_interval*** (***array***): intervalo en el que se definen los lados paralelos al eje *Y* módulo fotovoltaico → [y_*min*, y_*max*].
- ***nxb*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para el cálculo de la obstrucción a la radiación directa.
- ***nyb*** (***int***): parámetro con la misma finalidad que *nxb*, pero en la dirección *Y*.

- ***nxd*** (***int***): parámetro que controla la densidad de nodos en la dirección *X* empleados para el cálculo de los factores de visión.
- ***nyd*** (***int***): parámetro con la misma finalidad que *nxd*, pero en la dirección *Y*.
- ***rho_front*** (***float***): reflectividad de la cara frontal del receptor (entre 0 y 1).
- ***rho_rear*** (***float***): reflectividad de la cara trasera del receptor (entre 0 y 1).
- ***label*** (***str***): alias de identificación del módulo.
- ***I_sc_stc*** (***float***): corriente de cortocircutio, en *STC* [A].
- ***V_oc_stc*** (***float***): tensión de circuiro abierto, en *STC* [V].
- ***P_mpp_stc*** (***float***): potencia máxima, en *STC* [W].
- ***I_mpp_stc*** (***float***): corriente en el punto de máxima potencia en *STC* [A].
- ***V_mpp_stc*** (***float***): tensión en el punto de máxima potencia, en *STC* [V].
- ***alpha_sc*** (***float***): tasa de variación de I_{sc} con la temperatura [%/ºC].
- ***beta_oc*** (***float***): tasa de variación de V_{oc} con la temperatura [%/ºC].
- ***gamma_mpp*** (***float***): tasa de variación de P_{mpp} con la temperatura [%/ºC].
- ***NOCT*** (***float***): temperatura de célula en condiciones *NOCT* [ºC]
- ***n_diodes*** (***int***): número de diodos del módulo.
- ***cols_per_diode*** (***int***): número de columnas por diodo en las que se divide el módulo. En el caso de un módulo estándar, todas las células de las columnas asociadas a un diodo conformarán la cadena asociada a dicho diodo. En el caso de un módulo de célula parida, las mitades superiores de estas columnas albergarán todas las células de una de las cadenas del diodo asociado, mientras que las inferiores, la otra cadena.
- ***tiles_per_col*** (***int***): sólo en caso de que el módulo sea estándar. Representa el número de baldosas en las que se divide cada columna. Todas las células que comparten una baldosa son indistinguibles, pues reciben la misma radiación y pertenecen a la misma cadena.
- ***tiles_per_half_col*** (***int***): sólo en caso de que el módulo sea de célula partida. Representa el número de baldosas en las que se divide cada semicolumna.
- ***cells_per_tile*** (***int***): Número de células existentes en cada baldosa. Todas las células que comparten una baldosa son indistinguibles, pues reciben la misma radiación y pertenecen a la misma cadena.
- ***method*** (***str***): Método empleado para la extracción de los parámetros del modelo circuital de la célula fotovoltaica. Las opciones son: *'saloux'*, *'batzelis'* y *'cubas'*.
- ***is_bifacial*** (***bool***): Si el módulo es bifacial, se especificará el valor *True*.
- ***bifacial_coef*** (***float***): Coeficiente de bifacialidad del módulo. Sólo aplica en caso de que el módulo sea bifacial. Valor de 0 a 1.
- ***loss_coef*** (***float***): Coeficiente para modelar pérdidas, siendo aplicado a la irradiancia neta incidente sobre cada cara. 0 es sin pérdidas, y 1 equivale a que no incide radiación.

B.5.2 Movimientos y finalización de la implementación

Los métodos para mover y declarar la finalización de los módulos son los mismos que los expuestos en el Anexo B.3.4 ***Movimientos y finalización de la implementación***.

B.6 Strings

B.6.1 Inicialización

```
String(label)
```

- ***label*** (***str***): alias de identificación del *string*.

B.6.2 Incorporación de elementos y finalización de la implementación

- ***Añadir módulos.***

  Los módulos se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
String.add_module(module, multiplier)
```

  - ***module*** (***standardModule*** | ***halfCellModule***): instancia del módulo representativo a añadir.
  - ***multiplier*** (***int***): número de módulos semejantes al especificado que se añaden.

- ***Añadir resistencias.***

- Las resistencias de los polos de salida se inicializan como nulas, y se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
String.add_rs(rs)
```

  - ***rs*** (***float***): valor de la resistencia, expresada en ohmios, que se añade en serie al *string*.

- ***Añadir diodo de by-pass.***

  El diodo de *by-pass* se añade una única vez.

```
String.add_bd()
```

- ***Finalización del proceso.***

```
String.end_def()
```

Una vez invocado, no se podrán realizar más modificaciones al *string*.

B.7 Cajas de conexiones

B.7.1 Inicialización

```
junctionBox(label)
```

- ***label*** (***str***): alias de identificación de la caja de conexiones.

B.7.2 Incorporación de elementos y finalización de la implementación

- ***Añadir entradas.***

  Los pares de polos se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
junctionBox.add_pole_pair(obj, multiplier)
```

  - ***obj*** (***String*** | ***junctionBox***): instancia del objeto (*string* o caja de conexiones) representativo a añadir.
  - ***multiplier*** (***int***): número de objetos semejantes al especificado que se añaden, quedando conectados en paralelo.

- ***Añadir resistencias.***

  Las resistencias de los polos de salida se inicializan como nulas, y se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
junctionBox.add_rs(rs)
```

  - ***rs*** (***float***): valor de la resistencia, expresada en ohmios, que se añade en serie a uno de los polos de salida de la caja de conexiones.

- ***Finalización del proceso.***

```
junctionBox.end_def()
```

  Una vez invocado, no se podrán realizar más modificaciones a la caja de conexiones.

B.8 MPPT

B.8.1 Inicialización

```
mppTracker(label, V_min, V_max)
```

- ***label*** (***str***): alias de identificación del *MPPT*.
- ***V_min*** (***float***): tensión mínima de operación del *MPPT* [V].
- ***V_max*** (***float***): tensión máxima de operación del *MPPT* [V].

B.8.2 Incorporación de elementos y finalización de la implementación

- ***Añadir entradas.***

  Los pares de polos se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
mppTracker.add_pole_pair(obj, multiplier)
```

  - ***obj*** (***String*** | ***junctionBox***): instancia del objeto (*string* o caja de conexiones) representativo a añadir.
  - ***multiplier*** (***int***): número de objetos semejantes al especificado que se añaden, quedando conectados en paralelo.

- ***Finalización del proceso.***

```
mppTracker.end_def()
```

  Una vez invocado, no se podrán realizar más modificaciones al *MPPT*.

B.9 Inversores

B.9.1 Inicialización

```
Inverter(…)
```

- ***label*** (***str***): alias de identificación del inversor.
- ***P_nom*** (***float***): potencia nominal del inversor [W].
- ***V_nom*** (***float***): tensión nominal del circuito de corriente alterna [V]. Si el sistema es trifásico, será la de línea; si es monofásico, la propia entre fase y neutro.
- ***phases*** (***int***): número de fases del circuito de corriente alterna.
- ***b_0, b_1, b_2*** (***float***): coeficientes del modelo polinomial de eficiencia del inversor. Si no se especifican, se les asigna por defecto los siguientes valores: $\{b_0 = 0.005, b_1 = 0.016, b_2 = 0.014\}$

B.9.2 Incorporación de MPPT y finalización de la implementación

- ***Añadir MPPT.***

  Los *MPPT* se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
Inverter.add_mppt(mppt, multiplier)
```

  - ***mppt*** (***mppTracker***): instancia del *MPPT* representativo a añadir.
  - ***multiplier*** (***int***): cantidad de *MPPT* semejantes al especificado que se añaden.

- ***Añadir resistencias.***

  La resistencia de cada conductor del circuito de alterna se inicializa con un valor nulo. De forma secuencial, se pueden incrementar a través del siguiente método:

```
Inverter.add_rs(rs)
```

  - ***rs*** (***float***): valor de la resistencia, expresada en ohmios, que se añade en serie a cada conductor activo del circuito de alterna.

- ***Finalización del proceso.***

```
Inverter.end_def()
```

  Una vez invocado, no se podrán realizar más modificaciones al inversor.

B.10 Gestor del sistema

B.10.1 Inicialización

```
sysManager(calc_losses, update_scenes)
```

- ***calc_losses*** (***bool***): Si value *True*, se calcularán las pérdidas de forma desglosada, para ser posteriormente presentadas en los resultados.
- ***update_scenes:*** Si vale *True*, se calculará la irradiación acumulada sobre los receptores durante el intervalo de tiempo abarcado en cada simulación.

B.10.2 Incorporación de subsistemas y finalización de la implementación

- ***Añadir subsistemas.***

  Los subsistemas se añaden de forma secuencial, empleando el siguiente método las veces necesarias:

```
sysManager.add_subsystem(inverter, multiplier, label)
```

  - ***inverter*** (***Inverter***): instancia del inversor representativo del subsistema.
  - ***multiplier*** (***int***): número de inversores que conforman al subsistema, semejantes al especificado.
  - ***label*** (***str***): alias de identificación del subsistema.

- ***Finalización del proceso.***

```
sysManager.end_def()
```

  Una vez invocado, no se podrán realizar más modificaciones al gestor.

B.10.3 Simulación del sistema fotovoltaico

- ***Simulación horaria de las curvas características.***

  Para simular las curvas de corriente y potencia, en función de la tensión, de los componentes del sistema fotovoltaico en una fecha y hora concreta (consideradas constantes durante esa hora), se deberán simular la irradiancia y los parámetros eléctricos de éstos a través del siguiente método:

```
sysManager.load_hourly_data(date)
```

  - ***inverter*** (***Inverter***): fecha y hora de la simulación. El formato debe ser *"mm-dd[ ]HH:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.

  Se deberá tener en cuenta los siguientes puntos:

  - Si los datos meteorológicos han sido cargados desde un fichero *.csv*, esta fecha debe figurar en la columna "*time(UTC)"* del mismo.
  - En caso de haber sido cargados a través de la *API* de *PVGIS,* los datos están referenciados al comienzo de cada hora, por lo que el formato será *"mm-dd[ ]HH:00:00"*. Además, si dichos datos se han filtrado para eliminar las horas nocturnas, se debe asegurar que la hora sea diurna.

  Una vez ejecutado el método anterior, se podrán graficar las curvas *I-V* y *P-V* de los módulos, *strings*, cajas de conexiones y *MPPT* a través del siguiente método común:

```
(String | junctionBox | mppTracker | … ).plot_curves(…)
```

  - ***input_curves*** (***bool***): si vale *True*, se mostrarán en la misma ventana gráfica las curvas características de los elementos integrantes.
  - ***min_coef*** (***float***): constante por la que se multiplica el parámetro de referencia del componente ($I_{sc,STC}$ si es un *string*, y $V_{oc,STC}$ si es una caja de conexiones o un *MPPT*) para obtener el valor mínimo de la variable independiente empleada para generar las curvas (corriente si es un *string*, y tensión si es una caja de conexiones o un *MPPT*).
  - ***max_coef*** (***float***): constante por la que se multiplica el parámetro de referencia del componente ($I_{sc,STC}$ si es un *string*, y $V_{oc,STC}$ si es una caja de conexiones o un *MPPT*) para obtener el valor máximo de la variable independiente empleada para generar las curvas (corriente si es un *string*, y tensión si es una caja de conexiones o un *MPPT*).
  - ***npoints*** (***int***): número de puntos evaluados para generar las curvas.
  - ***title*** (***str***): título del gráfico.

- ***Finalización del proceso.***

Para simular la producción y las pérdidas energéticas en los distintos componentes de la instalación fotovoltaica, se empleará el siguiente método:

```
sysManager.simulate_system(init_date, final_date)
```

- ***init_date*** (***str***): fecha y hora del comienzo de la simulación. Si dicha fecha no figura en los datos meteorológicos cargados, se pasará a la siguiente más cercana.
- ***final_date*** (***str***): fecha y hora de la finalización de la simulación. Si dicha fecha no figura en los datos meteorológicos cargados, se pasará a la inmediatamente anterior.

Se deben tener en cuenta los siguientes puntos:

- El formato de las fechas debe ser *"mm-dd[ ]HH:MM:SS"*, donde *"[ ]"* representa un espacio en blanco.
- Los extremos de los intervalos están incluidos.
- Los datos de irradiación acumulada, producción y pérdidas serán reiniciados a 0.
- Si se desea simular una fecha y hora particular, los campos *init_date* y *final_date* deben ser igualados a dicha fecha y hora, debiéndose tener en cuenta los puntos señalados en el apartado anterior. Este método también habilita la representación de las curvas características para la fecha y hora señaladas.

B.10.4 Visualización y exportación de los resultados

- ***Exportación de los resultados a un fichero .ods***

El siguiente método genera un fichero *.ods* con los resultados horarios y globales de la última simulación realizada, para cada subsistema y para el sistema conjunto:

```
sysManager.export_data(file_name)
```

- ***file_name*** (***str***): nombre (sin extensión) del fichero a generar.

- ***Exportación de los resultados horarios a una variable***

El siguiente método devuelve una variable tipo *DataFrame*[9] con los resultados horarios de la última simulación realizada:

```
sysManager.return_hourly_data(label, init_date, final_date)
```

- ***label*** (***str***): alias del subsistema al que estarán referidos los datos. Si se especifica el valor *'global'*, los datos estarán referidos al sistema conjunto.
- ***init_date*** (***str***): fecha y hora del primer dato a exportar. Si no figura en los datos calculados, se pasará a la siguiente más cercana.

[9] Un *DataFrame* es un tipo de dato estructurado en filas y columnas etiquetadas, a modo de tabla.

- ***final_date*** (***str***): fecha y hora del último dato a exportar. Si no figura en los datos calculados, se pasará a la inmediatamente anterior.

- ***Exportación de los resultados mensuales a una variable***

  El siguiente método devuelve una variable tipo *DataFrame* con los resultados mensuales de la última simulación realizada:

```
sysManager.return_monthly_data(label)
```

  - ***label*** (***str***): alias del subsistema al que estarán referidos los datos. Si se especifica el valor *'global'*, los datos estarán referidos al sistema conjunto.

- ***Exportación del resumen de los resultados a una variable***

  El siguiente método devuelve una variable tipo *DataFrame* con el resumen de los resultados, referidos a todo el intervalo de tiempo abarcado en la última simulación realizada:

```
sysManager.return_summary_data(label)
```

  - ***label*** (***str***): alias del subsistema al que estarán referidos los datos. Si se especifica el valor *'global'*, los datos estarán referidos al sistema conjunto. Si se especifica el valor *'all'*, se devolverán los datos del conjunto y de cada subsistema.

- ***Visualización de los resultados horarios***

  El siguiente método muestra la distribución horaria del balance de energía y del *PR* de la última simulación realizada:

```
sysManager.plot_hourly_balance(…)
```

  - ***label*** (***str***): alias del subsistema al que estarán referidos los datos. Si se especifica el valor *'global'*, los datos estarán referidos al sistema conjunto.
  - ***init_date*** (***str***): fecha y hora del primer dato a graficar. Si no figura en los datos calculados, se pasará a la siguiente más cercana.
  - ***final_date*** (***str***): fecha y hora del último dato a graficar. Si no figura en los datos calculados, se pasará a la inmediatamente anterior.
  - ***title*** (***str***): título del gráfico.

- ***Visualización de los resultados mensuales***

  El siguiente método muestra la distribución mensual del balance de energía y del *PR* de la última simulación realizada:

```
sysManager.plot_monthly_balance(label, title)
```

  - ***label*** (***str***): alias del subsistema al que estarán referidos los datos. Si se especifica el valor *'global'*, los datos estarán referidos al sistema conjunto.

- ***title*** (***str***): título del gráfico.

- ***Visualización del resumen del balance de energía***

  El siguiente método muestra un gráfico con el resumen de los resultados del balance de energía, referidos a todo el intervalo de tiempo abarcado en la última simulación realizada:

```
sysManager.plot_summary_balance(label, title)
```

  - ***label*** (***str***): alias del subsistema al que estarán referidos los datos. Si se especifica el valor *'global'*, los datos estarán referidos al sistema conjunto.
  - ***title*** (***str***): título del gráfico.

BIBLIOGRAFÍA

[1] William F. Holmgren, Clifford W. Hansen, & Mark A. Mikofski, «pvlib python: a python package for modeling solar energy systems,» *Journal of Open Source Software*, vol. 29, pp. 884-884, 2018.

[2] EU Science Hub, «PVGIS user manual,» European Commision., [En línea]. Available: https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/getting-started-pvgis/pvgis-user-manual_en. [Último acceso: Diciembre 2023].

[3] I. Reda & A. Andreas, «Solar position algorithm for solar radiation applications,» *Solar Energy*, vol. 76, nº 5, pp. 577-589, 2004.

[4] OuYang, D., Feng, H.Y., «Reconstruction of 2D polygonal curves and 3D triangular surfaces via clustering of Delaunay circles/spheres,» *Computer-Aided Design*, vol. 43, pp. 839-847, 2011.

[5] Virtanen P., Gommers R., Travis E. Oliphant et al., «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,» *Nature Methods,* vol. 17, pp. 261-272, 2020.

[6] Liu, B.Y.H. & Jordan, R.C., «Daily Insolation on Surfaces Tilted towards Equator,» *ASHRAE Transactions*, vol. 3, pp. 526-541, 1961.

[7] Hay, J. E. & McKay, D. C., «Estimating Solar Irradiance on Inclined,» *International Journal of Solar Energy*, nº 3, pp. 203-240, 1985.

[8] Yang, D., «Solar radiation on inclined surfaces: Corrections and benchmarks,» *Solar Energy*, vol. 136, pp. 288-302, 2016.

[9] Hay, J. E. & Davies, J. A., «Calculation of the Solar Radiation Incident,» *First Canadian Solar Radiation Data Workshop*, pp. 59-72, 1980.

[10] Reindl, D. T., Beckman, W. A. & Duffie, J. A., «Evaluation of hourly tilted surface radiation models,» *Solar Energy*, vol. 45, pp. 9-17, 1990.

[11] Perez, R., Ineichen, P., Seals, R., Michalsky, J. & Stewart, R., «Modeling daylight availability and irradiance components from direct and global irradiance,» *Solar Energy*, vol. 44, pp. 271-289, 1990.

[12] Duffie, J. A. & Beckman, W. A., Solar Engineering of Thermal Proceses: Fourth Edition, 2013.

[13] Iturbide, P., Stern, V., Righini, R. y Aristegui, R., «Evaluación del albedo en la estación solarimétrica de Luján, suelo característico de la pampa húmeda,» *Avances en Energías Renovables y Medio Ambiente*, vol. 24, pp. 222-231, 2020.

[14] Sönmez, FF, Ziar, H, Isabella, O & Zeman, M, «Fast and accurate ray-casting-based view factor estimation method for complex geometries,» *Solar Energy Materials & Solar Cells*, vol. 200, pp. 1-11, 2019.

[15] Casado Vera, Roberto, Métodos numéricos en álgebra lineal, Barcelona: FUOC, 2019.

[16] W. De Soto, S. A. Klein, and W. A. Beckman, «Improvement and validation of a model for photovoltaic array performance,» *Sol. Energy*, vol. 80, pp. 78-88, 2006.

[17] Batzelis, E., «Non-Iterative Methods for the Extraction of the Single-Diode Model Parameters of Photovoltaic Modules: A Review and Comparative Assessment,» *Energies*, vol. 12, 2019.

[18] Cubas, J., Pindado, S., de Manuel, C, «Explicit expressions for solar panel equivalent circuit parameters based on analytical formulation and the Lambert W-function,» *Energies*, vol. 7, pp. 4098-4115, 2014.

[19] Batzelis, E.I., Routsolias, I.A., Papathanassiou, S.A., « An explicit PV string model based on the Lambert W function and simplified MPP expressions for operation under partial shading,» *IEEE Trans. Sustain. Energy*, vol. 5, pp. 301-312, 2014.

[20] Lineykin, S.; Averbukh, M.; Kuperman, A. , «An improved approach to extract the single-diode equivalent circuit parameters of a photovoltaic cell/panel.,» *Renew. Sustain. Energy Rev*, vol. 30, pp. 282-289, 2014.

[21] Saloux, E. Teyssedou, A. Sorin, M, «Explicit model of photovoltaic panels to determine voltages and currents at the maximum power point,» *Sol. Energy,* vol. 85, pp. 713-722, 1011.

[22] Batzelis, E.I., Papathanassiou, S.A, « A method for the analytical extraction of the single-diode PV model,» *IEEE Trans. Sustain. Energy,* vol. 7, pp. 504-512, 2016.

[23] Toledo, F.J., Blanes, J.M, ; Galiano, V, « Two-Step Linear Least-Squares Method For Photovoltaic Single-Diode Model Parameters Extraction,» *IEEE Trans. Ind. Electron.,* vol. 62, pp. 4181-4193, 2018.

[24] Osterwald C.R., «Translation of device performance measurements to reference conditions,» *Solar Cells,* vol. 18, pp. 269-279, 1986.

[25] Cuce, P.M. & Cuce, E., «A novel model of photovoltaic modules for parameter estimation and thermodynamic assessment,» *Int. J. Low-Carbon Technol.,* vol. 7, pp. 159-165, 2011.

[26] Xiao, W., Dunford, W.G. & Capel, A., «A novel modeling method for photovoltaic cells.,» de *In Proceedings of the 2004 IEEE 35th Annual Power Electronics Specialists Conference*, Aachen, Germany,, 20-25 June 2004.

[27] Harris, C.R., Millman, K.J., van der Walt, S.J. et al., «Array programming with NumPy,» *Nature,* vol. 585, pp. 357-362, 2020.

[28] Brent, R.P., «An algorithm with guaranteed convergence for finding a zero of function,» *The Computer Journal,* vol. 14, nº 4, pp. 422-425, 1971.

[29] M. Jantsch, H. Schmidt & J. Schmid, «Results on the concerted action on power conditioning and control,» de *11th European photovoltaic Solar Energy Conference*, Montreaux, 1992.

[30] Barry, D., Parlange, J.Y., Sander, G., Sivaplan, M., « A class of exact solutions for Richards' equation,» *J. Hydrol,* vol. 142, pp. 29-46, 1993.