



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Graduado/a en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Propuesta y desarrollo de contribuciones a la librería de
modelado fotovoltaico de código abierto pvlib python

Autor: Echedey Luis Álvarez

Co-Tutor:
Rubén Núñez Judez
DEPARTAMENTO DE INGENIERÍA
ELÉCTRICA, ELECTRÓNICA, AUTOMÁTICA
Y FÍSICA APLICADA (D180)

Tutor:
César Domínguez Domínguez
DEPARTAMENTO DE INGENIERÍA
ELÉCTRICA, ELECTRÓNICA, AUTOMÁTICA
Y FÍSICA APLICADA (D180)

Madrid, Septiembre, 2024

Este Trabajo Fin de Grado se ha depositado en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Título: Propuesta y desarrollo de contribuciones a la librería de modelado fotovoltaico de código abierto pvlib python

Septiembre, 2024

Resumen

La finalidad de este Trabajo Fin de Grado es la contribución de modelos científicos aplicados a la fotovoltaica dentro de la iniciativa de código abierto pvlib python. Adicionalmente, dentro del marco de contribuciones, se han añadido datasets, mejoras a la documentación ya existente, varios arreglos al flujo de integración y desarrollo continuo (CI/CD) y corregir múltiples bugs.

Palabras Clave: fotovoltaica, código libre, pvlib python, simulación, modelado

Abstract

The purpose of this final-year thesis is the contribution of scientific models used in photovoltaic simulation and research. Contributions have been proposed to the free and open source software pvlib python. Merged pull requests range from new dataset inclusion, improvements to the existing documentation, various fixes to the continuous integration and continuous development (CI/CD) workflow and multiple bugfixes.

Keywords: photovoltaic, open source, pvlib python, simulation, modelling

Agradecimientos

Me gustaría agradecer a mis tutores Rubén Núñez Judez y César Domínguez Domínguez por darme la posibilidad de invertir mi tiempo y capacidad en un proyecto que se alinea con mis objetivos de autorrealización, así como en su indispensable ayuda para entender y aplicar algunos de los modelos.

A la UPM por ofrecer máquinas virtuales de Linux al alumnado, tanto para el desarrollo de este proyecto como para hacer pruebas en terceros proyectos.

A Nuria Martín Chivelet por explicarme detalladamente el funcionamiento de su modelo científico y ofrecerme continuar en esa misma línea de trabajo.

A todos los mantenedores de la librería `pvl` python por sus revisiones en profundidad. En especial a Kevin Anderson y a Adam Jensen por ofrecerme y guiarme en obtener una beca bajo el programa *Google Summer of Code*¹.

A todas las personas que públicamente han contribuido directa o indirectamente en crear ecosistemas de desarrollo de software libre y de código abierto, en especial a aquellos involucrados en la comunidad de Python y Visual Studio Code entre innumerables otros.

Finalmente, a Aurelio Acevedo Rodríguez por inculcarme la importancia de la sección de agradecimientos. En paz descanse.

¹<https://summerofcode.withgoogle.com/programs/2024/projects/fxPFQqZc>

Índice general

1. Introducción	1
1.1. Motivación del proyecto	1
1.2. Contexto del proyecto	1
1.3. Objetivos	2
1.4. Estructura del Documento	3
2. Trabajo relacionado y Estado del Arte	4
2.1. La energía solar fotovoltaica	4
2.2. Simulación de sistemas fotovoltaicos	6
2.3. La librería pvlib python	7
2.3.1. Objetivos de la librería	7
2.3.2. Funcionalidades	9
2.3.3. Repositorio del proyecto	10
2.3.4. Herramientas de desarrollo del proyecto	11
3. Desarrollo	12
3.1. Entorno de desarrollo y herramientas utilizadas	12
3.1.1. La documentación	13
3.2. Contribuciones científicas	18
3.2.1. Modelado de ajuste espectral	18
3.2.1.1. Fundamento teórico	18
3.2.1.2. Resultado	19
3.2.2. Proyección del cenit solar sobre las coordenadas de un colector	19
3.2.2.1. Fundamento teórico	19
3.2.2.2. Resultado	20
3.2.3. Cálculo de fracción de sombra unidimensional	20
3.2.3.1. Fundamento teórico	20
3.2.3.2. Resultado	21
3.2.4. Pérdidas por sombreado en módulos con diodos de bypass	21
3.2.4.1. Fundamento teórico	21
3.2.4.2. Resultado	22
3.2.5. Fracción diaria de radiación difusa fotosintetizable en función de la fracción difusa global	22
3.2.5.1. Fundamento teórico	23
3.2.5.2. Resultado	23
3.2.6. Modelo de pérdidas por heterogeneidad de irradiancia por célula	23
3.2.6.1. Fundamento teórico	24
3.2.6.2. Resultado	24

3.2.7. Transformación de respuesta espectral a eficiencia cuántica externa	25
3.2.7.1. Fundamento teórico	25
3.2.7.2. Resultado	25
3.2.8. Adición de base de datos de respuesta espectral de algunas tecnologías	25
3.2.8.1. Fundamento teórico	25
3.2.8.2. Resultado	25
3.2.9. Adición de espectro estándar completo ASTM G173-03	25
3.2.9.1. Fundamento teórico	25
3.2.9.2. Resultado	25
3.2.10. Cálculo geométrico de sombras a partir de escenas en 3D	25
3.2.10.1. Fundamento teórico	25
3.2.10.2. Resultado	25
3.3. Contribuciones técnicas	25
3.3.1. Arreglo a los tests de integración continua en Windows al emplear Conda	25
3.3.1.1. Resultado	25
3.3.2. Arreglo a un parámetro ignorado en una función de transposición inversa	25
3.3.2.1. Resultado	26
3.3.3. Dar soporte a otra función para el cálculo del IAM en el flujo orientado a objetos	26
3.3.3.1. Resultado	26
3.3.4. Suprimir una advertencia al publicar la distribución en PyPI	26
3.3.4.1. Resultado	26
3.3.5. Exponer parámetros de tolerancia para resolver el modelo de un diodo	26
3.3.5.1. Resultado	26
3.3.6. Modificar tolerancias erróneas en varios tests unitarios	26
3.3.6.1. Resultado	26
3.3.7. Arreglo de un bug que ignoraba parámetros de una función	26
3.3.7.1. Resultado	26
3.3.8. Actualizar versiones de las dependencias de la documentación	26
3.3.8.1. Resultado	26
3.4. Contribuciones menores	26
3.4.1. Corrección de erratas en la documentación	26
3.4.2. Corrección de erratas tipográficas	27
3.4.3. Modificación de escritura de los parámetros opcionales	27
3.4.4. Limpieza de advertencias al construir la documentación	27
4. Impacto del trabajo	28
4.1. Impacto general	28
4.2. Objetivos de Desarrollo Sostenible	28
5. Resultados y conclusiones	29
5.1. Resultados	29
5.2. Conclusiones personales	29
5.3. Trabajo futuro	29

Bibliografía	30
A. Anexo	32

Índice de Figuras

3.1. Un ejemplo de renderizado de la documentación de una función en <i>pulib</i> <i>python</i>	15
3.2. Un ejemplo de página de ejemplo en la documentación de <i>pulib python</i> . .	17
3.3. Esquema de un módulo con 3 diodos de bypass.	22

Índice de Tablas

Índice de Listings

Capítulo 1

Introducción

El cambio climático es un tema de actualidad que plantea un reto social, económico y tecnológico. Dentro de este marco, las energías renovables se presentan como una solución tecnológica a la dependencia de los combustibles fósiles, que son los principales responsables de la emisión de gases de efecto invernadero. Una de las tantas fuentes de energía renovables más prometedoras es la solar fotovoltaica, ya que es renovable y no contamina en su explotación directa.

Este Trabajo de Fin de Grado pretende potenciar la adquisición, investigación e implementación de la energía solar, mejorando herramientas de simulación y diseño de instalaciones fotovoltaicas. Para ello, se han realizado múltiples contribuciones a un proyecto de código abierto llamado *pulib python*, que es una biblioteca de herramientas escrita en Python para el análisis de sistemas fotovoltaicos.

1.1. Motivación del proyecto

El alumno declara que la motivación para la realización de este Trabajo de Fin de Grado se fundamenta en su interés por las energías renovables, el código libre y su ya experiencia previa en desarrollo de software de Python, también de acceso abierto.

Asimismo una de las principales inquietudes del alumno es aplicar sus conocimientos a generar nueva ciencia de forma accesible y contrastable, y que pueda ser utilizada por la comunidad científica y técnica de manera completamente transparente.

1.2. Contexto del proyecto

Con el incremento del interés por las energías renovables y las facilidades del desarrollo software como caldo de cultivo, se ha propuesto al alumno la realización de este Trabajo de Fin de Grado, que se enmarca en el desarrollo de la biblioteca *pulib python*.

La propuesta parte de los tutores del alumno, que son miembros del grupo de investigación *Instruments and Systems Integration* del *Instituto de Energía Solar* de la propia Universidad Politécnica de Madrid. Asimismo, el proyecto *pulib python* es un

proyecto de código abierto que se encuentra en desarrollo por otros investigadores de centros de investigación y universidades públicas de múltiples países, y algunos miembros de empresas privadas del mismo campo.

El perfil de usuario de esta biblioteca presenta ciertos tipos principales:

- Investigadores que desean realizar simulaciones y estudios de sistemas fotovoltaicos.
- Ingenieros y técnicos que desean optimizar el diseño de instalaciones fotovoltaicas.
- Ingenieros y técnicos que quieren identificar faltas en este tipo de instalaciones, mediante la comparación de datos reales con simulaciones.

Por supuesto, tratándose de un proyecto de código abierto, cualquier persona puede utilizar la biblioteca, por lo que no se descarta la posibilidad de que otros perfiles de usuario puedan beneficiarse de las mejoras realizadas en este Trabajo de Fin de Grado. En este aspecto, sucede que esta iniciativa democratiza el acceso por parte de usuarios más noveles en el campo de la energía solar fotovoltaica, que no tienen acceso a herramientas comerciales.

1.3. Objetivos

La línea principal de este trabajo es la adición de nuevas funcionalidades a la biblioteca *pulib python*, que permitan mejorar la simulación, investigación y diseño de plantas fotovoltaicas. Para ello, se han establecido los siguientes objetivos:

- Contribuir modelos variados científicos en propósito y utilidad.
- Añadir otras funcionalidades, que no siendo modelos, sean útiles para el usuario.
- Validar el funcionamiento mediante tests unitarios.
- Facilitar su uso con una documentación didáctica y concisa.
- Seguir los rigurosos estándares de calidad de un proyecto científico de código libre.
- Ayudar a la comunidad de usuarios de la biblioteca a través de la resolución de dudas y problemas.
- Participar en la revisión de código de otros contribuyentes.
- Crear asuntos y cuestiones que promuevan la mejora de la biblioteca, en especial para animar a otros contribuyentes a participar.
- Ayudar a mantener la biblioteca actualizada para mejorar su ciclo de vida y arreglar fallos de forma preventiva.

Por otra parte, entre los objetivos secundarios destacan:

- Dar visibilidad a autores nacionales y de la Universidad Politécnica de Madrid.
- Potenciar proyectos de beneficio común desde la Universidad pública.

Si bien no es el propósito específico de este trabajo tratar estos últimos objetivos, se considera que son una consecuencia natural y deseable del presente proyecto.

1.4. Estructura del Documento

La estructura de este Trabajo de Fin de Grado pretende ser intuitiva y distribuida por bloques sobre temas similares. El lector podrá leer a continuación:

- En Capítulo 2 Trabajo relacionado y Estado del Arte, se da a conocer el estado actual de la energía solar fotovoltaica y algunas de las herramientas de simulación utilizadas.
- En Sección 2.3 La librería *pvl* python, se presentan las características de la biblioteca *pvl* python.
- En Capítulo 3 Desarrollo, se detalla el desarrollo de las contribuciones propuestas a la biblioteca, con factores tanto técnicos como humanos sobre el resultado.
- En Capítulo 5 Resultados y conclusiones, se resume el estado final o intermedio de las propuestas realizadas

Capítulo 2

Trabajo relacionado y Estado del Arte

En este capítulo se cubre el estado del arte de la energía solar fotovoltaica y, particularmente, la librería *pvlíb python*. El lector podrá encontrar en las siguientes secciones:

- En Sección 2.1 La energía solar fotovoltaica, se explica el estado actual de la energía solar fotovoltaica y su fundamento teórico base.
- En Sección 2.2 Simulación de sistemas fotovoltaicos, se detallan las herramientas de simulación utilizadas en el sector fotovoltaico y el marco general de comparación de la librería *pvlíb python*.
- En Sección 2.3 La librería *pvlíb python*, se presentan las características de la biblioteca *pvlíb python*.

2.1. La energía solar fotovoltaica

En esta sección se presentará el estado del arte de las diferentes tecnologías, estudios y sistemas relacionados con la energía solar fotovoltaica.

La energía solar fotovoltaica es una tecnología que convierte la radiación solar en electricidad utilizando células solares mediante el efecto fotoeléctrico [1, pp. 701-706]. Este fenómeno consiste en la generación de una corriente eléctrica cuando la luz incide sobre un material semiconductor y excita los electrones de la banda de valencia a la banda de conducción. Esta excitación genera un par electrón-hueco que se separa por la acción de un campo eléctrico externo (en cuyo caso no se produce energía neta positiva) o mediante la distribución de cargas en un semiconductor p-n, que permite la extracción de energía. Este último caso es el de aplicación en células solares fotovoltaicas, pues la intención es obtener energía eléctrica.

Existen varias tecnologías de células solares, como las de silicio, las de película delgada y las más experimentales de concentración y de otros materiales orgánicos y multiunión, que se agrupan en generaciones [2]. Cada generación responde a una serie de características e implantación en el mercado, donde destacan:

- Primera generación: células de silicio monocristalino y policristalino. Se encuentran bien implantadas en el mercado y son las más utilizadas en aplicaciones fotovoltaicas. Según el límite teórico que alcanzaban Shockley y Queisser en 1961, el silicio es el material más apropiado para la fabricación de células solares, ya que su banda prohibida de 1.1 eV es la que mejor se ajusta al máximo de la radiación solar [1, p. 1126]. Sin embargo, presentan un coste de producción moderado y un alto uso de material. El límite de eficiencia teórico obtenido por los anteriores autores es del 33.7% [3], pero asumen que no tratan con células solares de concentración ni con células solares de múltiples uniones o tandem.
- Segunda generación: células de película delgada, como las de telururo de cadmio (*CdTe*), las de di-seleniuro de cobre, indio y galio (*Copper indium gallium selenide*) (*CuInGaSe₂*), las de arseniuro de galio (*Gallium arsenide*) (*GaAs*) y las de silicio amorfo (*a-Si:H*). Destacan por ser de capa delgada (*thin-film*) y, consecuentemente, más baratas de producir por el bajo uso de material, si bien pueden llegar a ser composiciones más caras. Nótese que el silicio amorfo es ampliamente utilizado, en especial en aplicaciones de baja potencia, como calculadoras, pero su eficiencia es inferior a las células de silicio cristalino.
- Tercera generación: células de concentración, células de múltiples uniones y células orgánicas. Se encuentran en fase de investigación y desarrollo, y se caracterizan por su alta eficiencia y coste elevado. Por un lado destacan la tecnología de concentración, que emplea lentes para concentrar la luz solar en células solares de alta eficiencia, normalmente de múltiples uniones, que pueden alcanzar eficiencias superiores al 40% [4, Tabla 5]. Se emplean sistemas ópticos para disminuir el uso de material semiconductor, el principal factor de coste en estas células.

Cada una tiene sus propias características y aplicaciones específicas. Las células de primera y segunda generación son las más comunes en aplicaciones fotovoltaicas, mientras que las de tercera generación se encuentran en fase de investigación y desarrollo. Se remite el lector a [2] para una revisión más detallada de cada grupo y las peculiaridades de cada material.

En cuanto a los sistemas fotovoltaicos, se han desarrollado diferentes configuraciones, como sistemas conectados a la red, sistemas autónomos y sistemas de bombeo [5]. Cada configuración tiene sus propias ventajas y desafíos, y se han realizado investigaciones para optimizar su diseño y operación. En todos estos casos, es fundamental contar con herramientas de simulación y análisis para evaluar el rendimiento de los sistemas, optimizar su diseño y diagnosticar fallos a partir de datos meteorológicos.

Toda simulación para un sistema fotovoltaico tradicional -en referencia a los colectores planos y no aquellos de concentración, que presentan otras características- se basa en la radiación solar incidente en la superficie de los módulos. Los datos de partida de una simulación siempre son determinados valores meteorológicos como la radiación solar, la temperatura ambiente y la velocidad del viento. Adicionalmente, se emplean modelos empíricos para obtener valores como la radiación solar extraterrestre en una superficie normal al vector posición solar y deducir otros parámetros de entrada para el resto del flujo de cálculos, en función de los instantes de tiempo que se vayan a analizar.

2.2. Simulación de sistemas fotovoltaicos

A partir de unos valores de GHI , DNI y DHI -que son las componentes de la radiación solar global en una superficie horizontal, normal al vector solar y difusa en la superficie horizontal, respectivamente- se pueden obtener los valores de radiación solar en una superficie inclinada y orientada en función de los ángulos de inclinación y orientación de los módulos. Estas se relacionan mediante:

$$GHI = DNI \cdot \cos(\theta) + DHI \quad (2.1)$$

La existencia de los parámetros de entrada mencionados se explica mediante la necesidad de independizar las medidas de irradiancia solar de la superficie de interés, y lo que permiten cuantificar los instrumentos de medida en campo. Estos se tratan de piranómetros para medir GHI ; pirheliómetros, para DNI ; y piranómetros de sombra para DHI .

Esta radiación se descompone en tres componentes: la radiación directa, la radiación difusa y la radiación reflejada. La radiación directa es la que proviene del sol y llega directamente a la superficie de los módulos, la radiación difusa es la que proviene del cielo y se dispersa en todas direcciones, y la radiación reflejada es la que proviene de la reflexión de la radiación directa en superficies cercanas. Estas componentes se suman para obtener la radiación total incidente en la superficie de los módulos, que es la que se emplea para calcular la producción de energía.

En resumen, el estado del arte de la energía solar fotovoltaica abarca una amplia gama de tecnologías, estudios y sistemas. En este Trabajo de Fin de Grado, se tratarán algunas de las propuestas que se han realizado en el ámbito de la simulación y el análisis de sistemas fotovoltaicos, con un enfoque particular en la librería *pvl* *python*.

2.2. Simulación de sistemas fotovoltaicos

La importancia de simulaciones y análisis previo a la implantación de sistemas fotovoltaicos tanto para inversores, operaciones de financiación y diseñadores ha dado lugar a múltiples herramientas software y modelos como V Watts, PVGIS, PV-Online, PV*SOL, PVsyst, System Advisor Model (SAM) y muchas más [6, 7]. Normalmente estas herramientas propietarias tienen un foco muy específico en que un usuario pueda calcular el rédito energético y económico de una instalación fotovoltaica, pero no en la investigación y validación de modelos científicos. En 2.3 se comprobará que la librería *pvl* *python* se centra en la investigación y validación de modelos científicos para la simulación de sistemas fotovoltaicos, y que estos cálculos de rédito energético tienen más peso en el lado del usuario.

Dentro de este grupo de herramientas, surge la iniciativa de código abierto PVLIB-MatLab con origen en los laboratorios de *Sandia National Laboratories*, EEUU hacia el año 2009 [8]. Posteriormente, en 2013¹ inicia el desarrollo de la versión en Python [9, 10, 11, 12, 13], que es de la que trata este trabajo.

La librería *pvl* *python* aporta una serie de mejoras a su contrapartida en MATLAB, entre ellas la infraestructura de tests, la documentación y otros procedi-

¹<https://web.archive.org/web/20240411190506/https://pvl-lib-python.readthedocs.io/en/stable/#history-and-acknowledgement>

mientos de Integración Continua y Desarrollo Continuo (CI/CD) que facilitan el desarrollo, opuesto a la ausencia de tests, control de versiones en un documento de Microsoft Word y, en general, la falta de mantenimiento y actividad de la versión en MATLAB².

2.3. La librería *pvl*ib python

La librería *pvl*ib python es una biblioteca de código abierto que proporciona herramientas para la simulación, análisis e investigación de sistemas fotovoltaicos. Se encuentra desarrollada para el lenguaje de programación interpretado Python [14], que es ampliamente utilizado en la comunidad científica y de desarrollo de software por su sintaxis sencilla, facilidad de desarrollo y diseño orientado a objetos.

Como todo proyecto de código abierto que se encuentra en constante desarrollo, esta librería cuenta con un grupo de desarrolladores y colaboradores que contribuyen a su mantenimiento y mejora. Estas personas partícipes del proyecto son mayoritariamente científicos e investigadores de diferentes instituciones y universidades, normalmente públicas, e ingenieros u otros investigadores del ámbito privado. El código y la colaboración se realiza a través del repositorio de código abierto GitHub³.

El código se encuentra disponible al público bajo la licencia *BSD 3-Clause*⁴, que permite su uso, modificación, redistribución y uso en otros proyectos siempre que haga la atribución de autoría pertinente y no emplee la imagen de *pvl*ib python con fines publicitarios o promotores. La librería se distribuye a través del índice de paquetes de Python, PyPI⁵ y en el canal de *conda-forge*⁶ para la distribución de Python *Conda*, orientada a facilitar conjuntos de librerías para ciencia de datos sin que existan conflictos. *Conda* se distribuye en dos sabores -o conjuntos de librerías- conocidas como *Anaconda* y *Miniconda*, cada una con sus propias características y ventajas en tiempo de instalación, espacio en disco y paquetes preinstalados.

La documentación de la librería⁷ se encuentra disponible en la plataforma *ReadTheDocs*, donde se detallan las funcionalidades, los ejemplos de uso, la estructura del código y la API⁸, las herramientas de desarrollo, las directrices para contribuir al proyecto y la historia de cambios y versiones. La documentación se encuentra disponible exclusivamente en inglés.

2.3.1. Objetivos de la librería

Como se ha comentado anteriormente, la librería *pvl*ib python tiene como objetivo principal proporcionar herramientas para la simulación, análisis e investigación de sistemas fotovoltaicos. Para ello, cuenta con una serie de funciones y clases que

²http://web.archive.org/web/20211207215130/https://pvl-lib-python.readthedocs.io/en/v0.9.0/comparison_pvl-lib_matlab.html

³Véase <https://github.com/pvl-lib/pvl-lib-python>

⁴Texto de la licencia disponible en <https://opensource.org/licenses/BSD-3-clause>

⁵Véase <https://pypi.org/project/pvl-lib/>

⁶Véase <https://anaconda.org/conda-forge/pvl-lib>

⁷Accesible en <https://pvl-lib-python.readthedocs.io/en/stable/>

⁸API es la contracción de *Application Programming Interface* y se trata del conjunto de utilidades que se proveen para programadores que empleen la librería.

permiten realizar cálculos de radiación solar, generación de energía, sombreado, eficiencia de módulos, pérdidas de energía, entre otros. Estas herramientas se basan en modelos científicos y métodos de cálculo validados por la comunidad científica y se han implementado en Python para facilitar su uso y extensión. Es importante denotar que los autores pretenden establecer la implementación de la librería como una referencia fiable y certera de los modelos científicos que se emplean en la simulación de sistemas fotovoltaicos, con un enfoque en la rigurosidad de las publicaciones y artículos científicos que se emplean como base.

Enumerando los objetivos de la librería, se pueden destacar los siguientes:

1. Proporcionar herramientas para la simulación y análisis de sistemas fotovoltaicos.
2. Implementar modelos científicos y métodos de cálculo validados por la comunidad científica.
3. Establecer un referente fiable de las implementaciones de las características citadas anteriormente.
4. Auditar las contribuciones para garantizar la funcionalidad de las aportaciones.
5. Facilitar la integración con servicios de datos meteorológicos y bases de datos de radiación solar.
6. Fomentar la colaboración y la contribución de la comunidad científica y de desarrollo de software.

Cabe destacar que también existen los no-objetivos de la librería, que se refieren a las características que no se pretenden implementar o documentar. Entre estos destacan:

1. Proporcionar contenido didáctico sobre la energía solar fotovoltaica en términos generales, más allá del necesario para entender y aplicar las herramientas de la librería por usuarios con una base de los fundamentos.
2. Hacer inferencias, sugerencias o recomendaciones que no estén publicadas formalmente sobre el uso de los modelos implementados.
3. Facilitar un sistema *ready-to-go* para usuarios finales.

Puede parecer que estos no-objetivos son limitantes, pero en realidad se trata de una forma de garantizar la calidad y la fiabilidad de la librería, ya que se centra en la implementación rigurosa de publicaciones de la comunidad científica.

Una nota importante, en especial sobre el último punto, es que la definición de la misma librería es que se trata de una *toolbox*⁹. Esta nota es en contraposición a las herramientas de simulación y análisis de sistemas fotovoltaicos comerciales, que limitan y simplifican la experiencia del usuario, varias veces dejando de lado la documentación y la transparencia de los cálculos¹⁰.

⁹En español, “caja de herramientas”. <https://github.com/pvlib/pvlib-python/blob/99619e8fc5aea5c5dc4dacabb75b617786cc4a1a/README.md?plain=1#L61>

¹⁰Como hecho anecdótico, se puede observar alguna crítica sobre esto entre los participantes del proyecto, por ejemplo en <https://github.com/pvlib/pvlib-python/issues/2057#issuecomment-2197279047>

2.3.2. Funcionalidades

El proyecto de código abierto *pvlib python* cuenta múltiple características que abarcan un amplio rango de temas relacionados con la producción de energía solar fotovoltaica. Al momento de la redacción de este documento, la versión de la librería es la 0.11.0, publicada el 21 de junio de 2024. A continuación, se detallan algunas de las funcionalidades más destacadas:

- Cálculo de la posición del sol: para determinar la posición del sol en el cielo en función de la localización y la hora del día, en el submódulo `pvlib.solarposition`.
- Cálculo de valores estándares de la radiación solar: la librería proporciona herramientas para calcular la radiación solar en una superficie plana y horizontal en función de la localización, la posición del sol y parámetros atmosféricos. Véase `pvlib.solarposition` y `pvlib.clearsky`.
- Procedimientos de descomposición, transposición y transposición inversa de la radiación solar: se emplean para, a partir de la radiación solar en una superficie plana y horizontal, obtener la radiación solar en una superficie inclinada y orientada determinados ángulos. Y una vez obtenidas las componentes directa, difusa y de albedo de la irradiación, aplicar correcciones convenientes para obtener la radiación efectiva colectada en la superficie. También se puede realizar el proceso inverso, de las componentes en el plano inclinado a la radiación en una superficie plana y horizontal. Estas utilidades se encuentran en el submódulo `pvlib.irradiance`.
- Obtención de parámetros atmosféricos como la columna o masa de aire, un número que representa cuánta atmósfera hay, relativa a la columna completamente vertical ($A_M=1$), en el submódulo `pvlib.atmosphere`.
- Valores de albedo predefinidos, bien de materiales sólidos o de superficies naturales de agua, en `pvlib.albedo`.
- Corrección de la radiación incidente en función del ángulo de incidencia: debido a la reflexión que sufre la luz solar al incidir en una superficie de forma oblicua. Las utilidades se encuentran en el submódulo `pvlib.iam`.
- Producción fotovoltaica: mediante puntos característicos como el de máxima potencia (*MPP*) o calculando curvas I-V completas. Se emplean modelos de eficiencia de módulos y pérdidas de energía para estimar la producción de energía en un sistema fotovoltaico, mediante el modelo de un único diodo. Se encuentra en el submódulo `pvlib.singlediode`.
- Cálculo de sombreado: métodos analíticos para calcular la fracción sombreada de hileras de filas. Véase el submódulo `pvlib.shading`.
- Cálculo de ángulos de seguimiento: para seguidores de uno y dos ejes, que permitan coleccionar la máxima radiación solar directa, que normalmente se corresponderá con la máxima producción de energía. Véase el submódulo `pvlib.tracking`.
- Cálculo de temperatura de los módulos: normalmente empíricos, facilitan estimar la producción de energía ya que la eficiencia de las células es bastante susceptible a la temperatura. Estas utilidades se encuentran en `pvlib.temperature`.

- Estimación de ganancias o pérdidas por efectos del espectro solar: pues en función de la composición del espectro solar incidente en los módulos, la eficiencia de los mismos puede variar. En `pvl.lib.spectrum`.
- Modelado de eficiencia de inversores: para estimar la eficiencia de los inversores DC-AC en función de la potencia de entrada y sus demás características. En `pvl.lib.inverter`.
- Modelado de pérdidas en transformadores: para estimar las pérdidas en los transformadores de conexión a red. En `pvl.lib.transformer`.
- Modelado de pérdidas en cables: para estimar las pérdidas resistivas en los cables. En `pvl.lib.pvsystem`.
- Integración de APIs externas para la obtención de datos meteorológicos relacionados con la fotovoltaica, en `pvl.lib.iotools`.
- Abstracciones de los componentes que conforman una instalación fotovoltaica, mediante las clases `Location`, `Array`, `PVSystem` y la clase que agrupa el flujo computacional `Modelchain`, en los submódulos `pvl.lib.location`, `pvl.lib.pvsystem` y `pvl.lib.modelchain`.
- Cálculos adicionales para sistemas bifaciales, que son aquellos colectores planos que permiten la captación de radiación solar por ambas caras del módulo. En `pvl.lib.bifacial`.

2.3.3. Repositorio del proyecto

Como se ha mencionado anteriormente, el código de la librería *pvl* python se encuentra alojado en un repositorio de código abierto. Un repositorio es una carpeta que contiene una serie de archivos, de los cuales los más importantes se gestionan mediante un sistema de control de versiones, VCS por sus siglas en inglés. Este sistema permite llevar un registro de los cambios realizados en los archivos, así como la posibilidad de volver a versiones anteriores en caso de que se produzca un error o un fallo en el código. El uso de esta herramienta es indispensable en proyectos de software, ya que facilita la colaboración entre los desarrolladores y la gestión de las versiones del código.

En el caso de *pvl* python, el repositorio se encuentra alojado en la plataforma GitHub, que es una de las más populares para el alojamiento de proyectos de código abierto.

El repositorio cuenta con múltiples archivos y sub-carpetas que configuran el proyecto. Algunos de los más destacables son:

- `README.md`: es el primer archivo en verse en la plataforma online y contiene la descripción del proyecto, su propósito y cómo instalarlo, entre otros.
- `LICENSE`: es el archivo que contiene la licencia del proyecto, en este caso la licencia *BSD 3-Clause*.
- `pyproject.toml`, `setup.cfg`, `setup.py`, `MANIFEST.in`: son los archivos que definen la configuración del proyecto, como el nombre, la versión, la descripción, las dependencias, los archivos que componen una distribución precompilada o

de código fuente, etc. que los usuarios verán en la plataformas de distribución de paquetes.

- `docs/`: es la carpeta que contiene la documentación del proyecto, que se genera automáticamente a partir de los comentarios en el código.
- `pplib/`: es la carpeta que contiene el código fuente de la librería, organizado en sub-carpetas y archivos según su funcionalidad. Los tests unitarios y de integración se encuentran en esta carpeta.
- `AUTHORS.md`, `CODE_OF_CONDUCT.md`, `codecov.yml`, `readthedocs.yml`, `paper/`, `ci/`, `benchmarks/` y `.github/`: son archivos y carpetas que contienen información adicional sobre el proyecto y la configuración de todos los flujos de integración continua: destacan los tests, los benchmarks y la documentación.

2.3.4. Herramientas de desarrollo del proyecto

Como tal, el desarrollo del proyecto requiere el conocimiento de algunas herramientas que permiten la colaboración con el repositorio principal y cumplir con las directrices de desarrollo del proyecto.

En el lado de las herramientas esenciales, se encuentran:

- *Git*: como sistema de control de versiones que se emplea para gestionar los cambios en el código.
- *GitHub*: es la plataforma de alojamiento de proyectos de código abierto que se emplea para colaborar en el desarrollo del proyecto. Permite crear *issues* para reportar errores o sugerir mejoras, proponer cambios y revisarlos públicamente en *pull requests*. Además proporciona máquinas virtuales para ejecutar los procedimientos de integración y desarrollo continuos.
- *ReadTheDocs* junto con *sphinx*: es la plataforma que se emplea para ejecutar *sphinx*, el framework que produce la documentación en formato HTML, y alojar el producto. La documentación se escribe en formato *reStructuredText* y se genera automáticamente a partir de los comentarios en el código, scripts con los ejemplos y archivos fuente con el cuerpo más general.
- *pytest*: es la librería de Python que se emplea para escribir y ejecutar tests unitarios y de integración en el código. Facilita la creación de *mocks* u observadores del estado de partes del mismo proyecto, la reutilización de datos de tests y la ejecución opcional de los tests en función de las dependencias disponibles y visualizar los resultados en un informe. Posiblemente la mejor utilidad es poder ejecutar los tests uno a uno, en grupos o todos a la vez.
- *Codecov*: es la plataforma que se emplea para medir la cobertura de los tests en el código. Permite visualizar la cobertura de los tests en un informe y comprobar si se están realizando pruebas suficientes en el código.
- *flake8*: es una utilidad que detecta errores de estilo en el código, como la longitud de las líneas, la indentación, la presencia de espacios en blanco, entre otros. Facilita la escritura de código limpio y legible, por ende, mantenible.

Capítulo 3

Desarrollo

En este capítulo se describen el *modus operandi* del autor de este Trabajo Fin de Grado para contribuir a la librería de modelado fotovoltaico *pulib python*. Se detallan las herramientas y el entorno de desarrollo empleados, así como las contribuciones de todo índole realizadas, abiertas o planificadas.

3.1. Entorno de desarrollo y herramientas utilizadas

En el caso del autor de este Trabajo Fin de Grado, se emplea el siguiente software para el desarrollo del proyecto consistentemente:

- *Visual Studio Code*: es el editor de código que se emplea para escribir y editar el código fuente del proyecto. Es un editor de código semi-abierto, ligero y rápido que cuenta con una amplia gama de extensiones para facilitar el desarrollo de software. Es una alternativa multipropósito a otros editores de código como *PyCharm* o *Spyder* en el caso de Python. Es importante denotar que la gran inmensa utilidad que proporciona *VSCode* es gracias a las extensiones creadas por la comunidad de desarrolladores, que permiten desde la edición de archivos de texto plano hasta la depuración de código, pasando por la integración con servicios de control de versiones y la ejecución de tests de forma visual. He aquí extensiones que se emplean en el desarrollo del proyecto:
 - *Python*: es la extensión que se emplea para el desarrollo de código en Python. Proporciona funcionalidades como la autocompletación de código, la visualización de la documentación de las funciones, la ejecución de scripts y la depuración de código.
 - *Ruff*: permite formatear el código según las directrices de estilo de *flake8*.
 - *GitHub Copilot*: un asistente de generación de código en línea integrado en el editor, que se emplea para sugerir fragmentos de código y documentación en función del contexto. Agiliza el desarrollo.
 - *Code Spell Checker*: es un corrector ortográfico que se emplea para detectar errores de ortografía en el código y en la documentación.
 - *Jupyter Notebooks*: es la extensión que se emplea para editar y ejecutar *notebooks* de Jupyter en el editor, un formato que permite visualizar las

variables del contexto y facilita el debugging interactivo. Los ejemplos de *pulib python* se realizan con un formato similar a las celdas de texto o código de una *notebook*.

- Resaltado de sintaxis de varios formatos de archivos, como *reStructuredText*, *Markdown*, *YAML* y *TOML*: para facilitar la edición de la documentación y los archivos de configuración.
- *GitHub Desktop*: es la interfaz gráfica que se emplea para colaborar en el desarrollo del proyecto. Permite visualizar los cambios, crear *branches*, hacer *commits*, *pull requests* y *merges*, entre otros.
- *pip* y *venv*: como gestor de paquetes y entornos aislados de desarrollo de Python nativos. Se emplean para instalar las dependencias del proyecto y usar entornos con las versiones específicas requeridas aislado del resto del sistema.

De forma discreta, se ha hecho uso de otras herramientas como:

- *Git*: para clonar las ramas del autor de este TFG y correr partes de la integración continua, normalmente la documentación, en las máquinas virtuales de linux provistas por la UPM¹.
- *Miniconda*: una distribución de Python que se emplea para instalar y gestionar las dependencias del proyecto. Facilita la creación de entornos virtuales y la instalación de paquetes de Python. Se utilizó para diagnosticar un error de precisión por la compilación de algunas librerías y que hacía fallar un test.
- *LibreOffice Calc*: para generar datos de prueba y comprobar las implementaciones de las ecuaciones de los modelos.

El desarrollo de la librería se ha hecho habitualmente en Windows 10, en un portátil HP 15-dw2003ns así que se ha aprovechado a usar el subsistema de Windows para Linux (WSL) para ejecutar tests y construir la documentación en un entorno similar al de integración continua. No obstante, por los recursos limitados del portátil, y por poder cambiar de ramas mientras se construye la documentación, se ha hecho uso de las máquinas virtuales de la UPM para ejecutar los tests y construir la documentación.

3.1.1. La documentación

En esta sección pondremos en valor lo que realmente es lo más importante de un proyecto de programación, en especial de código abierto: la documentación. Es más, el valor de todas las aportaciones es documentar lo más rigurosamente posible los modelos y métodos que se implementan, tanto para dar a conocerlos como para que el uso sea correcto.

Como se comentaba anteriormente, la documentación de la librería *pulib python* se encuentra alojada en la plataforma *ReadTheDocs*, que expone los archivos HTML para que los usuarios puedan consultarla en línea. La documentación se genera automáticamente a partir de los comentarios en el código fuente, que se escriben en formato *reStructuredText* y se construye con el framework *sphinx*.

¹Accesible a través de <https://escritorio.upm.es/>

3.1. Entorno de desarrollo y herramientas utilizadas

Existen archivos específicos para indicar qué funciones o métodos son públicos, hacer páginas de inicio, de referencia, de ejemplos, de instalación, de contribución, etc. Además, se pueden incluir imágenes, tablas, gráficos, enlaces, referencias, entre otros elementos que facilitan la comprensión de los conceptos.

sphinx emplea el estilo de documentación de *pydata-sphinx-theme*, que organiza las secciones de la documentación y da un estilo homogéneo a la web. Además, para la creación de los ejemplos se emplea la extensión *sphinx-gallery*, que ejecuta unos scripts de Python similares en secciones a una *notebook* de Jupyter y captura la salida de texto estándar y los gráficos para mostrarlos en la documentación.

La *docstring* de cada función, que es el comentario que se escribe en la primera línea de la definición de la función y se emplea para autogenerar la documentación, utiliza el estilo de *numpydoc*. Este estilo permite incluir información sobre los parámetros de entrada, los valores de retorno, las excepciones que se pueden lanzar, entre otros. Además, se pueden incluir ejemplos de uso de la función e informar avisos de precaución sobre aspectos más específicos.

A continuación, una plantilla de ejemplo de documentación y de código de una función cualquiera:

```
1 def example_model(param1, param2):
2     """
3     Brief model description.
4
5     Long model description, also found at [1]_.
6
7     .. versionadded:: 0.1.0
8
9     .. warning::
10        This docstring is an example.
11
12    Parameters
13    -----
14    param1 : numeric
15        Description of the parameter.
16    param2 : numeric
17        Description of the parameter.
18
19    Returns
20    -----
21    float
22        Return type of the function.
23
24    Notes
25    -----
26    Additional notes about the function, detailed explanations if needed. Even an equation:
27
28    .. math::
29
30        f(x, y) = x + y
31
32    Examples
33    -----
34    >>> example_model(1, 5)
35    6.0
36
37    References
38    -----
39    .. [1] Author, A. (2024). Title of the paper. Journal, 1(1), 1-10. :doi:'10.0001/populate'
40    """
41    return float(param1 + param2)
```

Esta función, una vez listada en el archivo correspondiente del índice que nos interese -aquí usamos el submódulo `pvlb.solarposition` como ejemplo-, creará una página como la que se muestra en la figura 3.1:

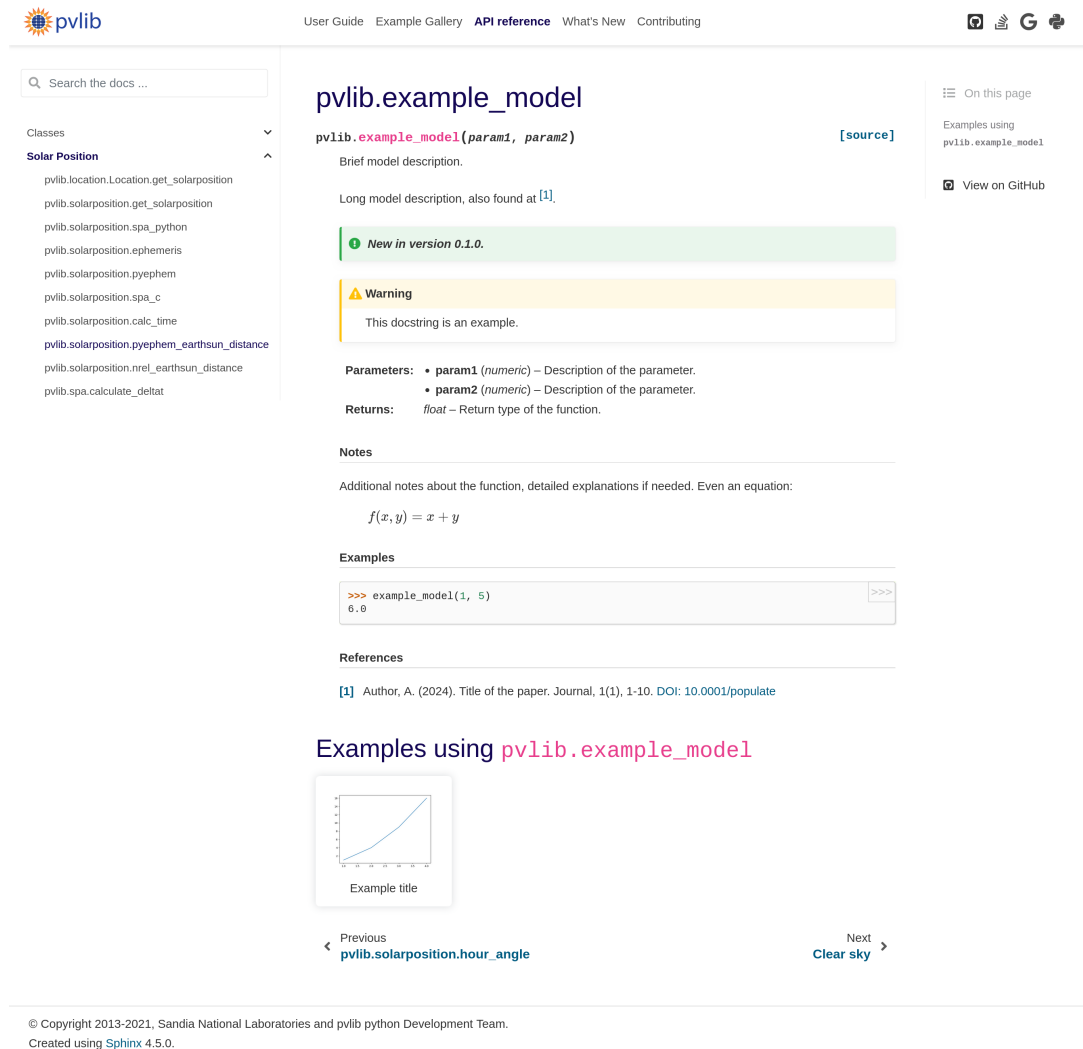


Figura 3.1: Un ejemplo de renderizado de la documentación de una función en *pvlb python*.


Por otro lado, un ejemplo cuenta con la siguiente estructura:

```
1 """
2 Example title
3 =====
4
5 Brief model description (shown in preview card).
6 """
7
8 # %%
9 # Text paragraph, in reStructuredText format. Can use sections, subsections, etc., and math as
10 #   in LaTeX.
11 # More text.
12
13 # This is a comment (there is a newline above)
14 import matplotlib.pyplot as plt
```


3.1. Entorno de desarrollo y herramientas utilizadas

```
14 from pvlib import example_model
15 print("Hello, world!")
16 plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
17 plt.show()
18
19 # %%
20 # Return to paragraph text.
21
22 sum_val = example_model(1, 5)
23 print(f"Sum of 1 and 5 is {sum_val}.")
```

Este archivo, ubicado en la carpeta correcta (aquí el archivo es `docs\examples\solar-position\example_example.py`), hará que se cree automáticamente una página como la que se muestra en la figura 3.2.



[User Guide](#)
[Example Gallery](#)
[API reference](#)
[What's New](#)
[Contributing](#)

 Search the docs ...

ADR Model for PV Module Efficiency

Agrivoltaic Systems Modelling

Bifacial Modeling

Irradiance Decomposition

Irradiance Transposition

I-V Modeling

Reflections

Shading

Soiling

Solar Position

Example title

Sun path diagram

Note

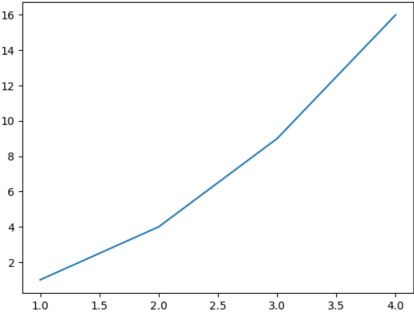
Go to the end to download the full example code.

Example title

Brief model description (shown in preview card).

Text paragraph, in reStructuredText format. Can use sections, subsections, etc., and math as in LaTeX. More text.

```
# This is a comment (there is a newline above)
import matplotlib.pyplot as plt
from pvlib import example_model
print("Hello, world!")
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```



Out: Hello, world!

Return to paragraph text.

```
sum_val = example_model(1, 5)
print(f"Sum of 1 and 5 is {sum_val}.")
```

Out: Sum of 1 and 5 is 6.0.

Total running time of the script: (0 minutes 0.060 seconds)

[Download Jupyter notebook: example_example.ipynb](#)

[Download Python source code: example_example.py](#)

Gallery generated by Sphinx-Gallery

Previous

Solar Position

Next

Sun path diagram

© Copyright 2013-2021. Sandia National Laboratories and pvlib python Development Team.

Created using Sphinx 4.5.0.

Figura 3.2: Un ejemplo de página de ejemplo en la documentación de *pvlib python*.

A continuación se muestran los comandos que deben ejecutarse para construir la documentación:

- En un sistema basado en *Linux* o *WSL*.
- Clonando el repositorio original de la librería con *Git*.
- Empleando las mismas versiones que a día de la redacción de este documento se emplea en la integración continua (*pvlib python*=0.11.0) - en especial hay que hacer la instalación de Python3.8.
- Con el entorno aislado de desarrollo (*venv*) activado.

17

- Y realizando una instalación local mediante *pip*.

```

1 # instalar Python 3.8 desde el repositorio de deadsnakes,
2 # en las librerías por defecto de Ubuntu no se encuentra disponible por antigüedad
3 sudo add-apt-repository ppa:deadsnakes/ppa -y
4 sudo apt-get update
5 sudo apt-get install python3.8 python3.8-venv -y
6
7 # clonar el repositorio de pvlib python
8 git clone https://github.com/pvlib/pvlib-python
9 cd pvlib-python
10
11 # crear el entorno virtual, instalar pvlib y las dependencias de la documentación
12 python3.8 -m venv .venv
13 source .venv/bin/activate
14 python3.8 -m pip install .[doc]
15
16 # construir la documentación
17 cd docs/sphinx
18 make html
19
20 # abrir la documentación en el navegador
21 xdg-open build/html/index.html

```

3.2. Contribuciones científicas

3.2.1. Modelado de ajuste espectral

- *Pull Request*: #1658

El desarrollo inició en Septiembre de 2022, en Febrero de 2023 se planteó la *pull request*, que finalmente se cerró en Mayo de 2023 sin incluirse los cambios en la librería.

3.2.1.1. Fundamento teórico

El modelo [15] plantea una relación entre la efectividad bajo un espectro estándar y la efectividad bajo un espectro arbitrario caracterizado por la masa de aire y el índice de claridad. Al principio de la implementación, tanto el autor de este TFG como el mentor del proyecto, César Domínguez, pensaron que se trataba de un modelo de ajuste similar a otros en la literatura que corrigen la irradiancia incidente, para dar lugar a la efectiva. Ejemplos de estos modelos ya se encontraban en la librería, como el modelo desarrollado por la empresa *First Solar* descrito en [16]. Además, para la versión 0.11.0 de la librería un compañero del programa *Google Summer of Code* implementó un par de modelos más que funcionan de la misma forma. Los modelos en cuestión crean un factor de ajuste M que se define como:

$$M = \frac{S_{efE(\lambda)}}{S_{ef\bar{G}(\lambda)}} \quad (3.1)$$

Resultó que, tras un elevado esfuerzo que llevó meses, el modelo en [17] no se ajustaba a esta descripción. Sin embargo, era una de las tres relaciones que se planteaban en la tesis doctoral de Nuria Martín Chivelet [17], que es:

$$PS = 1 - \frac{S_{efE(\lambda)} E_{\lambda < \lambda_0} \bar{G}}{S_{ef\bar{G}(\lambda)} \bar{G}_{\lambda < \lambda_0} E} \quad (3.2)$$

Se puede observar que la ecuación 3.1 solo contempla una parte de la definición que aplica del ajuste espectral, por ende, invalida la aplicación que se esperaba del modelo de dicho artículo.

3.2.1.2. Resultado

Se descarta aplicar los procedimientos descritos en su tesis [17] por:

- La ausencia de acceder al documento de forma online.
- La ausencia de una versión en inglés, que pueda servir como referencia.
- La particularidad de los datos sobre los que se hace el modelo.

Lamentablemente, incluso tras contactar presencialmente con la autora y habiendo disfrutado tanto de una explicación detallada de su modelo científico como de la posibilidad de continuar en esa misma línea de trabajo, y contando con una copia física de su tesis, se desestima continuar en esa línea de trabajo debido a las razones anteriores.

Se cierra la *pull request*.

3.2.2. Proyección del cenit solar sobre las coordenadas de un colector

- *Issue*: #1734
- *Pull Request*: #1904

Esta es la primera de una trilogía de contribuciones que se plantean para aplicar un modelo de pérdidas por sombreado en módulos con diodos bypass, cuyo autores pertenecen a la Escuela Técnica Superior de Ingeniería y Diseño Industrial, *ETSIDI*. El objetivo de esta primera contribución es calcular la proyección del cenit solar sobre las coordenadas de un colector, que se emplea para calcular la fracción de sombra unidimensional en geometrías de paneles que comparten eje de rotación en común.

Las otras dos contribuciones son 3.2.3 y 3.2.4.

Esta funcionalidad ya existía como parte de alguna función de la librería, así que el aporte consistió en rehacerla de nuevo empleando una referencia bibliográfica y contrastando las implementaciones. Por supuesto, la documentación se cobro la parte más importante del tiempo de desarrollo.

3.2.2.1. Fundamento teórico

Dos cálculos de bastante interés en geometría solar es obtener los ángulos óptimos de seguimiento para un colector y calcular la fracción de sombra unidimensional incidente. Ambos cálculos tienen en común un paso muy importante, que es saber con qué ángulo inciden los rayos directos del Sol sobre la superficie del colector, pero referenciado al plano de rotación del mismo. Nos encontramos estos dos casos de uso:

- Para el cálculo de los ángulos óptimos de seguimiento en seguidores de un solo eje, asumiendo que interesa seguir al Sol en su trayectoria diaria, se debe conocer el ángulo de incidencia de los rayos solares sobre la superficie del colector en el plano de rotación del mismo. Es decir, proyectar el cenit solar en el plano perpendicular al eje de rotación del colector, que es aquel que contiene todos los vectores normales al plano del colector.
- En el caso de la fracción unidimensional de sombra, interesa saber en donde impactan los límites del colector frontal sobre el trasero. Para ello, se proyecta el cenit solar en el plano perpendicular al eje de rotación del colector, que es aquel que contiene los dos vectores normales a los planos de los colectores.

3.2.2.2. Resultado

Después de cincuenta y tres comentarios en la propuesta, finalmente se incluyen los cambios como parte de la librería en la versión 0.10.4.

Accesible en `pvlib.shading.projected_solar_zenith_angle`.

3.2.3. Cálculo de fracción de sombra unidimensional

- *Issue*: #1689
- *Pull Request*: #1962

Esta propuesta es la segunda de la trilogía de propuestas para poder aplicar un modelo de pérdidas por sombreado. Continúa con la propuesta anterior, 3.2.2, y se encarga de calcular la fracción de sombra unidimensional en paneles con determinadas geometrías. La última propuesta de la trilogía es 3.2.4.

Aquí se plantea la aplicación de un modelo para conocer la fracción de sombra unidimensional en paneles que comparten eje de rotación en común. La implementación se basa en [18]. No obstante, la implementación inició con un póster de conferencia anterior, a partir de una propuesta de cambios de un trabajador de *First Solar* en la librería *pvlib python*, que se puede consultar en <https://github.com/pvlib/pvlib-python/pull/1725>.

3.2.3.1. Fundamento teórico

El modelo propuesto en [18] parte de un diseño de dos colectores que comparten la misma dirección del eje, y uno se encuentra más cercano al Sol que el otro. Lo interesante de este diseño es que tiene en cuenta múltiples variables de diseño, como la pendiente, la separación entre el eje y el plano colector, e inclinaciones distintas del colector sombreado y el que sombrea.

Se requiere conocer el ángulo proyectado del cenit. A partir de este ángulo, mediante intersección de rectas, se puede conocer la fracción de sombra unidimensional. Realmente se trata de una función muy compleja por el número de entradas que tiene, pues adicionalmente el cálculo de esta proyección se hace internamente en la función para simplificar la API -es decir, la interfaz programática.

Las ecuaciones no son de interés para este documento, pero se pueden consultar en [18].

3.2.3.2. Resultado

Después de 102 comentarios en la propuesta, finalmente se incluyen los cambios como parte de la librería en la versión 0.11.0, junto con un ejemplo ilustrativo que tiene en cuenta posibles dificultades que pueda encontrar un usuario².

Accesible en `pvlib.shading.shaded_fraction1d`.

3.2.4. Pérdidas por sombreado en módulos con diodos de bypass

- *Issue*: #2063
- *Pull Request*: #2070

Con esta propuesta finaliza la trilogía de contribuciones del modelo de pérdidas por sombreado en módulos con diodos de bypass. Las dos propuestas anteriores son 3.2.2 y 3.2.3.

La propuesta es de un modelo realizado por personal de la *ETSIDI* y se encarga de calcular las pérdidas por sombreado en módulos con diodos de bypass. La implementación se basa en el trabajo de Martínez-Moreno, F. and Muñoz, J. and Lorenzo, E., en [19].

3.2.4.1. Fundamento teórico

Los módulos de paneles solares de silicio se conforman de múltiples células fotovoltaicas. Una célula, cuando es irradiada por la luz solar, genera una corriente eléctrica. Si conectamos todas las células en serie, la corriente generada por todas las células es la misma, pero la tensión generada por cada célula se suma. Si una célula se sombrea, la corriente generada por ella disminuye, y por tanto la corriente generada por el conjunto disminuye. Esta célula sombreada se comporta como una carga para el resto, pues deben forzar la corriente que pasa por este elemento (que se suele modelar como un diodo). Supone un riesgo de seguridad y de degradación, pues una célula sombreada puede calentarse excesivamente y dañar las capas materiales de su módulo.

La solución que se emplea industrialmente consiste en añadir *diodos de bypass*. Estos permiten que, cuando una serie de células tiene sombra, la corriente mayoritaria generada por el resto de células bien iluminadas fluya a través de este diodo, y no de la célula sombreada. De esta forma, se protege frente al sobrecalentamiento y la degradación temprana.

Ha de hacerse énfasis en que un diodo protege varias células, ya que no es rentable económicamente poner un diodo por cada célula. El planteamiento que nos encontramos en [19] consiste en, a partir del número de grupos de células protegidos por un diodo, asumir que la tensión de este grupo es nula por tener un diodo en conducción y cancelar la potencia que generaría.

²Véase en https://pvlib-python.readthedocs.io/en/latest/gallery/shading/plot_shaded_fraction1d_ns_hsat_example.html.

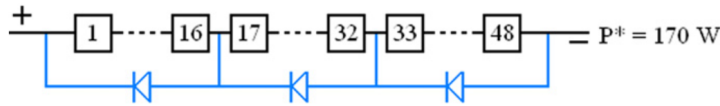


Figura 3.3: Esquema de un módulo con 3 diodos de bypass.

En la imagen 3.3, si suponemos que la célula número 1 está sombreada, la corriente mayoritaria generada por los grupos 17 a 32 y 33 a 48 fluye a través del diodo que está en paralelo con las células 1 a 16. No importa en este caso la corriente del grupo 1 a 16, pues la tensión de este grupo es nula por tener un diodo en conducción.

Nótese que estos grupos se definen en [19] como *bloques*, y un bloque está sombreado en cuanto una de sus células recibe una fracción infinitesimal de sombra.

Lo más complicado de esta contribución es explicar en detalle cómo identificar el número de bloques y su disposición en el módulo, pues existen varias posibilidades. Hay módulos lo suficientemente pequeños para que solo tengan un diodo, los hay con dos y con tres diodos, y los hay *half-cut* que también tienen 3 diodos, pero con una disposición que crea 6 bloques en vez de 3. Además, la progresión de los bloques que se va sombreando depende del sistema y la geometría de las sombras.

El resultado de este modelo establece la cantidad de potencia que se perdería respecto de las mismas condiciones sin sombra, $1 - \frac{P_{\text{sombreado}}}{P_{\text{no sombreado}}}$. Además, anular la potencia de un bloque sombreado se hace sobre la componente directa de la irradiancia, que es la que normalmente genera las sombras, pues la componente difusa sigue impactando en las células sombreadas y creando un mínima parte de aporte energético.

La expresión de pérdidas de potencia es la ecuación 3.2.4.1, Eqs. [6] y [8] en [19].

$$(1 - F_{ES}) = (1 - F_{GS}) \left(1 - \frac{N_{SB}}{N_{TB} + 1} \right) \quad (6)$$

$$\left(1 - \frac{P_S}{P_{NS}} \right) = \left(1 - \frac{[(B + D^{CIR})(1 - F_{ES}) + D^{ISO} + R]}{G} \right) \quad (8)$$

3.2.4.2. Resultado

Tras unos 85 comentarios, finalmente se incluyen los cambios como parte de la librería en la versión 0.11.0, junto con un ejemplo de caso de uso³.

Accesible en `pvlib.shading.direct_martinez`.

3.2.5. Fracción diaria de radiación difusa fotosintetizable en función de la fracción difusa global

■ Issue: #2047

³Véase en https://pvlib-python.readthedocs.io/en/latest/gallery/shading/plot_martinez_shade_loss.html.

- *Pull Request*: #2048

Esta contribución se trata de un pequeño modelo que abre un nuevo tema en la librería: la agrivoltaica. La agrivoltaica es una técnica en la que coexisten la producción de energía solar y la producción agrícola en un mismo terreno. La ventaja es que un exceso de irradiancia puede ser perjudicial para la plantación, y el retorno económico del terreno puede ser mayor que si solo se dedicase a la producción de energía solar. Además, contribuye positivamente en la polémica de utilizar suelo exclusivamente para la producción de energía.

El modelo en cuestión es la continuación de un trabajo realizado por Spitters C. J. T. et al. que cubre la separación de irradiación en directa y difusa en general [20], y que posteriormente él simplifica en una sola expresión en [21]. El objetivo de este modelado que hace es calcular la fracción de irradiación difusa que es útil para las plantas -es decir, fotosintetizable- a partir de la fracción de irradiación difusa global.

3.2.5.1. Fundamento teórico

La irradiación difusa fotosintetizable (PAR, por sus siglas en inglés), es la radiación difusa que se encuentra en el rango de longitudes de onda que las plantas pueden absorber y utilizar para la fotosíntesis. Es interesante de cara a simular el crecimiento y producción de las plantas, y se emplea en modelos de cultivo. Esto último queda fuera del alcance de la librería, pero se plantea como un paso en motivar y facilitar el diseño de sistemas agrivoltaicos.

Se suele utilizar irradiación diaria [$J/m^2/dia$], en vez de valores instantáneos como en la simulación de sistemas fotovoltaicos [W/m^2].

Un análisis de distintos modelos y la validación de los mismos se puede encontrar en [22]. Este artículo es del cual origina la propuesta de contribución.

3.2.5.2. Resultado

Con la mayor complicación de discernir las unidades de entrada, finalmente se incluyen los cambios como parte de la librería en la versión 0.11.0, junto con un ejemplo de uso⁴.

Accesible en `pvlib.irradiance.diffuse_par_spitters`.

3.2.6. Modelo de pérdidas por heterogeneidad de irradiancia por célula

- *Issue*: #1541
- *Pull Request*: #2046

Esta contribución implementa un modelo de pérdidas sobre la potencia de salida para módulos bifaciales, es decir, aquellos que pueden recibir irradiación solar tanto por una cara delantera como por la trasera. El modelo se aplica para tener en cuenta irradiancia que no es homogénea en la superficie del módulo. Se trata del trabajo descrito en [23]. Presenta interés en sistemas bifaciales, donde la cara trasera

⁴Véase en https://pvlib-python.readthedocs.io/en/latest/gallery/agrivoltaics/plot_diffuse_PAR_Spitters_relationship.html.

normalmente se expone a la luz reflejada por el suelo y otras obstrucciones, y por tanto la irradiancia de esa cara no es homogénea.

3.2.6.1. Fundamento teórico

Anteriormente se explicaba el mecanismo de interconexión de células solares fotovoltaicas, y cómo una célula sombreada puede actuar como una carga para el resto de células. De forma similar ocurre a pequeña escala, cuando una o varias células reciben valores de irradiancia ligeramente distintos al resto. La célula que recibe menos irradiancia actúa como una carga para el resto, y la potencia de salida del módulo se ve reducida.

Desde un punto de vista computacional, resolver un sistema de múltiples células, cada una con su propia irradiancia, es realmente costoso en tiempo y en recursos. Cada célula tendría su propia curva I-V, que representa cuanta corriente y tensión genera dependiendo del punto de tensión de trabajo. El planteamiento que se hace en [23] es realizar un trabajo previo de caracterización de la heterogeneidad, cuantificarla y establecer un modelo de menor orden de complejidad.

Para caracterizar distribuciones de irradiancia, en el artículo de Deline et al. se plantea utilizar la desviación estándar, muy común para distribuciones normales, o la *Diferencia Absoluta Media Relativa (RMAD*, por sus siglas en inglés), que es una medida de dispersión que se argumenta ser más adecuada para distribuciones no normales [24].

La pérdida de potencia de salida M (véase 3.2.4.1) se calcula con un polinomio evaluado en la *RMAD*,

$$M = 1 - \frac{P_{\text{array}}}{\sum P_{\text{cells}}} \quad (3.3)$$

donde P_{array} es la potencia de salida del módulo, y P_{cells} es la potencia máxima de salida de cada célula.

Se proponen dos modelos para el polinomio que define M , para dos perfiles de irradiancias globales distintos: uno para sistemas sujetos fijos y otro para seguidores de un eje. No obstante, solo se implementa el de sistemas fijos ya que la referencia indica que para valores anuales parece ser más preciso.

3.2.6.2. Resultado

Tras una ardua discusión de 86 comentarios en los que se plantean distintas formas de implementar el modelo, aclaraciones sobre las unidades de entrada y salida, y modificaciones al planteamiento original, finalmente se incluyen los cambios como parte de la librería en la versión 0.11.1 (sin publicar oficialmente a día de la redacción de este documento), junto con un ejemplo de uso⁵.

Accesible en `pvlib.bifacial.power_mismatch_deline`.

⁵Véase en https://pvlib-python.readthedocs.io/en/latest/gallery/bifacial/plot_irradiance_nonuniformity_loss.html.

3.2.7. Transformación de respuesta espectral a eficiencia cuántica externa

- *Issue:* #2040
- *Pull Request:* #2041

3.2.7.1. Fundamento teórico

3.2.7.2. Resultado

3.2.8. Adición de base de datos de respuesta espectral de algunas tecnologías

- *Issue:* #2037
- *Pull Request:* #2038

3.2.8.1. Fundamento teórico

3.2.8.2. Resultado

3.2.9. Adición de espectro estándar completo ASTM G173-03

- *Issue:* #2039
- *Pull Request:* #1963

3.2.9.1. Fundamento teórico

3.2.9.2. Resultado

3.2.10. Cálculo geométrico de sombras a partir de escenas en 3D

- *Issue:* #2069
- *Pull Request:* #2106

3.2.10.1. Fundamento teórico

3.2.10.2. Resultado

3.3. Contribuciones técnicas

3.3.1. Arreglo a los tests de integración continua en Windows al emplear Conda

- *Issue:* #2000
- *Pull Request:* #2007

3.3.1.1. Resultado

3.3.2. Arreglo a un parámetro ignorado en una función de transposición inversa

- *Issue:* #1970

- *Pull Request*: #1971

3.3.2.1. Resultado

3.3.3. Dar soporte a otra función para el cálculo del IAM en el flujo orientado a objetos

- *Issue*: #1742
- *Pull Request*: #1832

3.3.3.1. Resultado

3.3.4. Suprimir una advertencia al publicar la distribución en PyPI

- *Pull Request*: #1778

3.3.4.1. Resultado

3.3.5. Exponer parámetros de tolerancia para resolver el modelo de un diodo

- *Issue*: #1249
- *Pull Request*: #1764

3.3.5.1. Resultado

3.3.6. Modificar tolerancias erróneas en varios tests unitarios

- *Pull Request*: #2082

3.3.6.1. Resultado

3.3.7. Arreglo de un bug que ignoraba parámetros de una función

- *Issue*: #2018
- *Pull Request*: #2020

3.3.7.1. Resultado

3.3.8. Actualizar versiones de las dependencias de la documentación

- *Pull Request*: #2112

3.3.8.1. Resultado

3.4. Contribuciones menores

3.4.1. Corrección de erratas en la documentación

- *Pull Request*: #1599
- *Pull Request*: #1833

3.4.2. Corrección de erratas tipográficas

- *Pull Request:* #1776
- *Pull Request:* #1860
- *Pull Request:* #1996

3.4.3. Modificación de escritura de los parámetros opcionales

- *Issue:* #1574
- *Pull Request:* #1828
- *Pull Request:* #2084

3.4.4. Limpieza de advertencias al construir la documentación

- *Pull Request:* #2030

Capítulo 4

Impacto del trabajo

«Breve explicación, por secciones, de los contenidos de este capítulo»

4.1. Impacto general

«Análisis del impacto potencial de los resultados obtenidos durante la realización del TFG, en los diferentes contextos para los que se aplique. Además, se harán notar aquellas decisiones tomadas a lo largo del trabajo que tienen como base la consideración del impacto.»

4.2. Objetivos de Desarrollo Sostenible

«Se recomienda analizar también el potencial impacto respecto a los Objetivos de Desarrollo Sostenible (ODS), de la Agenda 2030, que sean relevantes para el trabajo realizado (ver enlace 1, ver enlace 2)»

Capítulo 5

Resultados y conclusiones

«Breve explicación, por secciones, de los contenidos de este capítulo»

5.1. Resultados

«Resumen de resultados obtenidos en el TFG»

5.2. Conclusiones personales

«Conclusiones personales del estudiante sobre el trabajo realizado»

5.3. Trabajo futuro

«Trabajo futuro que no se haya podido realizar o siguientes pasos que tomará el desarrollo realizado en este TFG»

Bibliografía

- [1] K.W. Böer y D. Bimberg. *Survey of Semiconductor Physics, Survey of Semiconductor Physics*. Wiley, 2002. ISBN: 9780471355724. URL: https://books.google.es/books?id=_pdvAQAAACAAJ.
- [2] Mahmood H. Shubbak. «Advances in solar photovoltaics: Technology review and patent trends». En: *Renewable and Sustainable Energy Reviews* 115 (nov. de 2019), pág. 109383. ISSN: 1364-0321. DOI: 10.1016/j.rser.2019.109383.
- [3] William Shockley y Hans J. Queisser. «Detailed Balance Limit of Efficiency of p-n Junction Solar Cells». En: *Journal of Applied Physics* 32.3 (mar. de 1961), págs. 510-519. ISSN: 0021-8979. DOI: 10.1063/1.1736034.
- [4] Martin A. Green et al. «Solar cell efficiency tables (Version 63)». en. En: *Progress in Photovoltaics: Research and Applications* 32.1 (2024), págs. 3-13. ISSN: 1099-159X. DOI: 10.1002/pip.3750.
- [5] O. Perpiñán. *Energía Solar Fotovoltaica*. 2020. URL: <http://oscarperpinan.github.io/esf/>.
- [6] Joshua S Stein y Geoffrey T Klise. «Models used to assess the performance of photovoltaic systems.» En: (dic. de 2009). DOI: 10.2172/974415. URL: <https://www.osti.gov/biblio/974415>.
- [7] Nallapaneni Manoj Kumar. «Simulation Tools for Technical Sizing and Analysis of Solar PV Systems». En: *Proceedings of the 6th World Conference on Applied Sciences, Engineering and Technology (WCSET-2017), 26-27 August 2017, UMP-PO, Indonesia, ISBN 13: 978-81-930222-3-8, pp 218-222, At Universitas Muhammadiyah Ponorogo, Indonesia*. (ene. de 2017). URL: https://www.academia.edu/35141273/Simulation_Tools_for_Technical_Sizing_and_Analysis_of_Solar_PV_Systems.
- [8] Joshua S. Stein et al. «PVLIB: Open source photovoltaic performance modeling functions for Matlab and Python». En: *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*. Jun. de 2016, págs. 3425-3430. DOI: 10.1109/PVSC.2016.7750303. URL: <https://ieeexplore.ieee.org/abstract/document/7750303>.
- [9] Kevin S. Anderson et al. «pvlb python: 2023 project update». En: *Journal of Open Source Software* 8.92 (2023), pág. 5994. DOI: 10.21105/joss.05994. URL: <https://doi.org/10.21105/joss.05994>.
- [10] Joshua S. Stein. «The photovoltaic Performance Modeling Collaborative (PVPMLC)». En: *2012 38th IEEE Photovoltaic Specialists Conference*. Jun. de 2012, págs. 003048-003052. DOI: 10.1109/PVSC.2012.6318225. URL: <https://ieeexplore.ieee.org/document/6318225>.
- [11] Robert W. Andrews et al. «Introduction to the open source PV LIB for python Photovoltaic system modelling package». En: *2014 IEEE 40th Photovoltaic Spe-*

- cialist Conference (PVSC)*. Jun. de 2014, págs. 0170-0174. DOI: 10.1109/PVSC.2014.6925501. URL: <https://ieeexplore.ieee.org/document/6925501>.
- [12] William F. Holmgren et al. «PVLIB Python 2015». En: *2015 IEEE 42nd Photovoltaic Specialist Conference (PVSC)*. Jun. de 2015, págs. 1-5. DOI: 10.1109/PVSC.2015.7356005. URL: <https://ieeexplore.ieee.org/document/7356005>.
- [13] William F. Holmgren y Derek G. Groenendyk. «An open source solar power forecasting tool using PVLIB-Python». En: *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*. Jun. de 2016, págs. 0972-0975. DOI: 10.1109/PVSC.2016.7749755. URL: <https://ieeexplore.ieee.org/document/7749755>.
- [14] G. van Rossum. *Python tutorial*. Inf. téc. CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI), mayo de 1995.
- [15] N. Martín y J. M. Ruiz. «A new method for the spectral characterisation of PV modules». en. En: *Progress in Photovoltaics: Research and Applications* 7.4 (1999), págs. 299-310. ISSN: 1099-159X. DOI: 10.1002/(SICI)1099-159X(199907/08)7:4<299::AID-PIP260>3.0.CO;2-0.
- [16] Mitchell Lee y Alex Panchula. «Spectral correction for photovoltaic module performance based on air mass and precipitable water». En: *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*. Jun. de 2016, págs. 1351-1356. DOI: 10.1109/PVSC.2016.7749836. URL: <https://ieeexplore.ieee.org/abstract/document/7749836>.
- [17] Nuria Martín Chivelet. «Estudio de la influencia de la reflexión, el ángulo de incidencia y la distribución espectral de la radiación solar en los generadores fotovoltaicos». PhD Thesis. 1999.
- [18] Kevin S. Anderson y Adam R. Jensen. «Shaded fraction and backtracking in single-axis trackers on rolling terrain». En: *Journal of Renewable and Sustainable Energy* 16.2 (abr. de 2024), pág. 023504. ISSN: 1941-7012. DOI: 10.1063/5.0202220.
- [19] F. Martínez-Moreno, J. Muñoz y E. Lorenzo. «Experimental model to estimate shading losses on PV arrays». En: *Solar Energy Materials and Solar Cells* 94.12 (dic. de 2010), págs. 2298-2303. ISSN: 0927-0248. DOI: 10.1016/j.solmat.2010.07.029.
- [20] C. J. T. Spitters, H. A. J. M. Toussaint y J. Goudriaan. «Separating the diffuse and direct component of global radiation and its implications for modeling canopy photosynthesis Part I. Components of incoming radiation». En: *Agricultural and Forest Meteorology* 38.1 (oct. de 1986), págs. 217-229. ISSN: 0168-1923. DOI: 10.1016/0168-1923(86)90060-2.
- [21] C. J. T. Spitters. «Separating the diffuse and direct component of global radiation and its implications for modeling canopy photosynthesis Part II. Calculation of canopy photosynthesis». En: *Agricultural and Forest Meteorology* 38.1 (oct. de 1986), págs. 231-242. ISSN: 0168-1923. DOI: 10.1016/0168-1923(86)90061-4.
- [22] S. Ma Lu et al. «Photosynthetically active radiation decomposition models for agrivoltaic systems applications». En: *Solar Energy* 244 (sep. de 2022), págs. 536-549. ISSN: 0038-092X. DOI: 10.1016/j.solener.2022.05.046.
- [23] Chris Deline et al. «Estimating and parameterizing mismatch power loss in bifacial photovoltaic systems». en. En: *Progress in Photovoltaics: Research and Applications* 28.7 (2020), págs. 691-703. ISSN: 1099-159X. DOI: 10.1002/pip.3259.
- [24] URL: <https://www.semanticscholar.org/paper/Gini%E2%80%99s-Mean-difference%3A-a-superior-measure-of-for-Yitzhaki/e4d00851cbbadf386ed051397c>

Apéndice A

Anexo

«Este capítulo (anexo) es opcional, y se escribirá de acuerdo con las indicaciones del Tutor.»