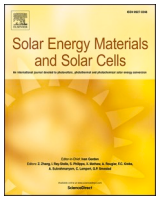
Solar Energy Materials & Solar Cells 279 (2025) 113245

Contents lists available at ScienceDirect

Solar Energy Materials and Solar Cells

journal homepage: www.elsevier.com/locate/solmat



cpvlib: A comprehensive open-source tool for modeling CPV systems

Rubén Núñez [a,b,*], Marcos Moreno [b], Rebeca Herrero [a,b], Steve Askins [a], Ignacio Antón [a], César Domínguez [a,b]

[a] Instituto de Energía Solar, Universidad Politécnica de Madrid, Avda. Complutense, 30, 28040, Madrid, Spain
[b] Escuela Técnica Superior de Ingeniería y Diseño Industrial, Universidad Politécnica de Madrid, Spain

A B S T R A C T

The design and simulation of concentrator photovoltaic (CPV) systems necessitate precise modeling tools, for which some commercial and open-source options exist. However, when new technologies or applications need to be modeled, they can present some limitations: lack of documentation transparency and inability to extend existing models, or little flexibility to do it. For instance, the novel hybrid CPV/flat-plate module, conceived by Insolight and developed within the HIPERION project, required the ability to model integrated tracking and dual use of incident irradiance, which was not possible with existing tools. Addressing these issues, cpvlib is introduced as a comprehensive, open-source tool offering modular and adaptable functionalities for CPV-based systems, built as an extension of the popular pvlib python library.

cpvlib's design enables the simulation of various CPV-based configurations, incorporating advanced architectures such as integrated tracking and hybrid CPV-flat plate modules. The library uses PVSyst's utilization factors to model deviations from the single-diode model, accounting for spectral and thermal effects. Its class structure leverages object-oriented programming principles, ensuring ease of use and extension.

The validation of cpvlib is carried out through the modeling and long-term monitoring of Insolight's hybrid Si/III-V translucent planar micro-tracking modules, achieving a root mean square error of 3.5 % in case of Si cells and 2.7 % for III-V CPV cells. The tool accounts for complex behaviors like air mass impact on CPV performance, angle of incidence limits, and light spillage. The annual energy yield for a hybrid module is computed using typical meteorological year data, showcasing cpvlib's practical application.

1. Introduction

Concentrator photovoltaics (CPV) is a PV technology that uses optical elements to concentrate sunlight onto small high-efficiency solar cells, so that higher system efficiencies can be obtained [1].

In order to estimate the output of CPV systems taking into account their peculiarities, CPV systems designers require specific modeling tools. The vast majority of models are usually ad-hoc, created mostly by the designers themselves [2]. These models are highly flexible and are implemented directly in code, making them fully accessible, but due to their customized nature they are not easily interoperable with other models. Among standard PV modeling software, System Advisor Model (SAM) [3] presents basic functionalities to simulate CPV systems; HOMER Pro [4] can use CPV modules as PV arrays elements; PVSyst [5,6] added the possibility to simulate two-axis tracking CPV modules [7,8,9]. Since all these software packages are standardized programs, they allow models to be easily modified and shared among different users. However, they all have limited expandability because PVSyst and HOMER Pro are proprietary software and SAM, although it is open source, has a graphical user interface and is organized around predefined modeling workflows very hard to extend. In contrast, the pvlib python library is a collaborative open-source modeling toolbox based on modular building blocks that provide full access, documentation and flexibility to the users so they can create their own custom modeling workflows. However, pvlib python lacks appropriate building blocks and workflows able to model CPV systems.

In this work, we present a Python toolbox called cpvlib that relies on pvlib to build a comprehensive set of modeling functions and workflows for a diverse range of state-of-the-art CPV-based technologies, leading to the development of an adaptable, open-source solution. Unlike the other PV simulation tools that incorporate CPV models seen previously, cpvlib stands out by providing flexibility and adaptability. Traditional models often lack the capability to integrate state-of-the-art architectures and modeling methods, hindering progress in the field. Examples of such innovations that are currently implemented include integrated tracking [10], planar micro-tracking [11], and hybrid CPV-flat plate modules for efficient diffuse light harvesting [12]. The cpvlib library bridges this gap by offering a flexible model designed for these novel architectures, with the capacity to easily accommodate further advancements.

* Corresponding author. Instituto de Energía Solar, Universidad Politécnica de Madrid, Avd. Complutense, 30 28040, Madrid, Spain.
E-mail address: ruben.nunez@upm.es (R. Núñez).

https://doi.org/10.1016/j.solmat.2024.113245
Received 29 June 2024; Received in revised form 16 October 2024; Accepted 20 October 2024
Available online 30 October 2024

2. Modeling of CPV features. Utilization factors

Compared to PV systems, CPV systems present a series of features that require specific modeling:

1. Depending on the level of irradiance concentration, the solar resource collected by a CPV system is only direct solar radiation.
2. They have higher operating temperatures than conventional systems.
3. In most cases, III-V multi-junction cells are used, which makes them strongly sensitive to changes in the solar spectrum.
4. They may be sensitive to the temperature of the optical components.

The modeling of features 1 (taking only Direct Normal Irradiance, DNI as input resource) and 2 (fitting the thermal model to the corresponding parameters) is obtained by directly adapting the classical models of PV systems. However, features 3 and 4 involve nonlinearities mainly in the current with respect to irradiance and ambient temperature. cpvlib is able to simulate these peculiarities of CPV systems through utilization factors (UF). A UF describes the power deviation of a CPV system from its nominal value according to a given parameter [7, 13]. It was originally intended to extend PVSyst capabilities to model CPV systems [7,8], and can also be found in other software with CPV system modeling capabilities such as SAM [3].

The idea is to be able to characterize behaviors that deviate from the model of single diode in order to be able to continue using this model, characterizing spectral or thermal effects through proxy parameters such as air mass or ambient temperature. It therefore acts as a dimensionless power reduction factor applied directly to power value, as shown in Equation (1). Its derivation is eminently experimental.

$$Pot_{w.UF} = Pot_{w/o\ UF} \cdot UF \tag{1}$$

UF is generally modeled as piecewise functions as in Equation (2), with different slopes (m) around a threshold value (x_{thld}) where UF is equal to one. An example of a graphical representation of a UF model can be seen in Fig. 5 left, where a piecewise function is derived from the module data in operation.

$$UF = \begin{cases} 1 + (x - x_{thld}) \cdot m_{x \le x_{thld}} & if\ x \le x_{thld} \\ 1 + (x - x_{thld}) \cdot m_{x > x_{thld}} & if\ x > x_{thld} \end{cases} \tag{2}$$

If several proxy parameters are identified, independent UF functions should be defined and then combined with experimentally obtained weighting factors that should add up to one [7].

3. Description of the tool

The foundation of cpvlib lies in the well-established pvlib python [14,15] (hereinafter referred to as pvlib), a collection of functions and classes designed for simulating photovoltaic energy systems developed in Python sustained by the open-source community. pvlib incorporates a collection of various models for the geometry or solar irradiance, production prediction and electrical or thermal behavior of photovoltaic systems among others taken from the scientific literature. In addition, these models can be interconnected in high-level objects that allow to obtain the weather-to-power simulation chain of a PV system.

pvlib's code is hosted on a public repository, fostering a collaborative environment with features like online documentation, issue tracking, code style guidelines, continuous integration development, and comprehensive test coverage.

Derived from the philosophy of pvlib, cpvlib [13] emerges as an open-source, well-documented, and modular tool specifically tailored for modeling the performance of CPV systems. Its primary goal is to accurately predict the energy production of several CPV configurations. cpvlib is mainly a layer over pvlib that provides a set of functions and classes for simulating the performance of concentrator photovoltaic energy systems.

In contributing to the growing ecosystem of open-source tools for solar energy [16] (including tools to access remote solar irradiance data [17], two-axis tracking modeling [18] or power forecasting tools [19]), cpvlib actively supports community-driven software. This approach encourages reproducibility, replicability, transparency, and collaboration in scientific studies. Within the PV community, utilizing open-source software like cpvlib may also reduce project financing soft costs, as the transparency of algorithms and implementations, vetted by users, inherently mitigates risks.

3.1. Code organization and structure of classes

To implement CPV systems in cpvlib, the inheritance property of classes in an object-oriented programming language, such as Python, has been used. Classes are templates for the creation of data objects. In the case of pvlib they allow for example to model PV systems, so that an object of this type contains all the information describing the system as attributes (size, inclination …) and the methods that can be applied to the system (calculate the effective irradiance on the plate, obtain the cell temperature …).

The cpvlib classes inherit from the pvlib.pvsystem.PVSystem class directly or through intermediate classes modifying the attributes and methods that differ from a conventional PV system. Hence those fields that are not modified work as in the parent class. For example, in CPVSystem the get_irradiance method is rewritten to take into account that the irradiance seen by the system is the one seen by a two-axis tracker, but it keeps the same syntax as in pvlib. In addition, CPV specific methods or attributes (i.e. to implement UFs) have also been incorporated when necessary.

Following pvlib structure, a main Python module serves as recipient of the following classes: CPVSystem (on 2-axis tracker CPV (sub)system), StaticCPVSystem (internal tracking CPV (sub)system), StaticFlatPlateSystem (flat plate (sub)system) and StaticHybridSystem (hybrid static CPV + flat plate system). StaticHybridSystem is a composition of StaticCPVSystem and StaticFlatPlateSystem, while StaticCPVSystem inherits from CPVSystem and StaticFlatPlateSystem inherits from pvlib.pvsystem.PVSystem. These relationships between the classes are shown in Fig. 1 using a Unified Modeling Language (UML) diagram.

3.2. PV system modeling

In order to obtain expected power from a PV system, pvlib contains several models that simulate it. The Sandia Array Performance Model (pvlib.pvsystem.sapm ()) is one of them. However, to implement this model necessitates fitting a large number of empirical coefficients, which are not directly linked to physical quantities, making implementation challenging. Another model to simulate power also included

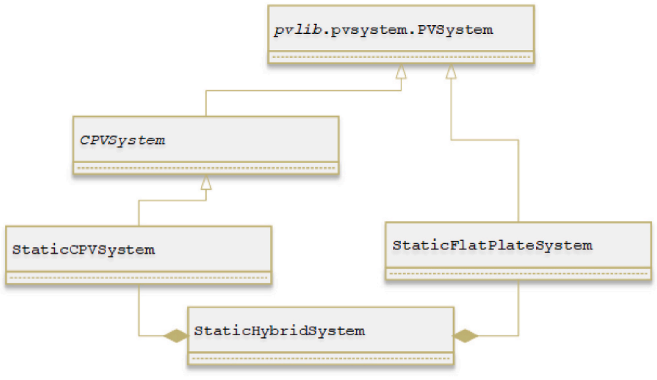


Fig. 1. UML diagram showing the interdependency of cpvlib classes. The triangles represent inheritance and the rhombuses, composition.

in pvlib is “PVSyst v6”, namely pvlib.pvsystem.calcparams_PVSyst () for obtaining the parameters of single diode model. There is also a thermal modeling parameter ‘PVSyst’ for obtaining the cell temperature by calling get_cell_temperature () method. Since these models can be traced to physical quantities, “PVSyst" is the chosen approach for the cpvlib implementation.

4. Hybrid CPV/flat plate module. Validation

cpvlib has been mainly used in modeling hybrid Si/III–V and translucent planar micro-tracking modules with planar micro-tracking [20] developed by Insolight company as part of the HIPERION European project [12,21].

4.1. Hybrid CPV/flat plate module description

The structure of a Insolight hybrid Si/III-V module with planar micro-tracking is shown in Fig. 2. The planar micro-tracking system allows Insolight modules to be installed at a fixed tilt, concentrating sunlight through a polymer lens array onto III–V triple-junction cells with a 180x concentration factor. As the sun moves, a planar micro-tracking mechanism shifts the backplane to track the sun for angles of incidence (AOI) up to 55° [20].

A biconvex aspheric lens ensures optimal concentrating power and transmission over a wide range of AOIs, excluding Fresnel and plano-convex lenses due to their poor off-axis performance [20]. Direct sunlight focuses on high-efficiency III-V cells, while diffuse light is captured by lower-cost Si cells. Direct sunlight is concentrated in high-efficiency III-V cells, while diffuse light is captured by lower-cost Si cells. This hybrid configuration allows for the possibility that the system can be viable in locations with low DNI. Hybrid Insolight modules place III–V cells between spaced Si cells, so this approach captures both diffuse and direct light effectively, even beyond the tracking range. In some conditions, some DNI spillage may occur from III-V to Si cells [22].

4.2. Hybrid CPV-flat plate module modeling using cpvlib

Using cpvlib, Insolight system can be modeled by an object of the class StaticHybridSystem, so both static CPV and flat plate systems are covered with the same entity. A diagram of this class, inner class relations and input/output data objects are shown in Fig. 3.

As indicated in Section “PV system modeling”, the PVSyst single diode model implemented in pvlib is used for both subsystems, together with specific CPV methods or the Insolight architecture. An StaticCPVSystem object models the CPV submodule. Within this object both the irradiance seen by the submodule (DII, with an AOI limit) and the CPV-specific UFs are included. In the case of the flat plate submodule, it is modeled by an object of the class StaticFlatPlateSystem; this is a conventional PV module whose effective irradiance is affected by the tracking limit of the CPV submodule and the possible light spillage.

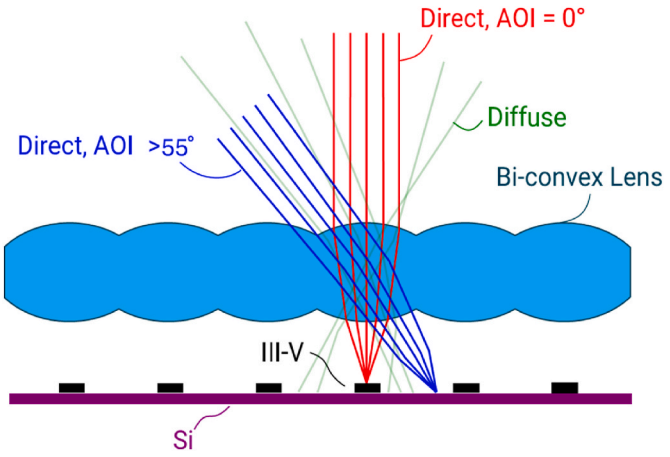


Fig. 2. Schematic diagram illustrating the operation of the micro planar tracking and diffuse collection system. The direct beam irradiance is captured by III-V cells, while the diffuse light is collected by the Si cell. When the angle of incidence (AOI) is not 0°, the biconvex lens shifts the focal spot, and a mechanism adjusts the back plane to track this focal spot. When the 55° limit is reached, the focus is captured by the Si cell.

4.3. cpvlib validation

An Insolight module was mounted on the roof of the Instituto Energía Solar building and monitored during several weeks during the summer of 2021. From the current and voltage measurements of the module together with the values of irradiance, wind speed and ambient temperature collected by the weather station, a dataset has been created to adjust the parameters required in both StaticCPVSystem and StaticFlatPlateSystem classes. The specific values of these parameters for the monitored module are listed in Appendix A.

Once the model parameters have been adjusted, the model can be used to generate the expected power. Of particular relevance is the modeling of the maximum power current (I_{MP}), since most of the differences with respect to traditional PV systems in the modeling of a hybrid module affect the current.

Fig. 4 shows the comparison of CPV and flat plate I_{MP}, both modeled and experimentally measured, and the irradiance seen by both subsystems for two days. A root mean square error (RMSE) between model and experiment for CPV of 6.4 % is obtained, which is reduced to 2.7 % when UFs are considered. For the flat plate, the RMSE is halved to 3.5 % when light spillage modeling is included.

The following section details the specific parts of major relevance when modeling a hybrid module of this nature and how they are implemented in cpvlib.

4.4. Specific features of CPV and hybrid modeling with cpvlib

Four of the distinguishing features with respect to the modeling of classical PV systems required for the modeling of an Insolight hybrid system are shown below, namely 1) the utilization factors, 2) the AOI limit for solar tracking, 3) the definition of an incidence angle modifier function and 4) the presence of light spillage between sub-modules. A description of each of the features and how they are implemented in cpvlib is given.

This section is followed by an example of using the full model to estimate the annual production of such a hybrid module at a given site.

4.4.1. Utilization factors (UF)

Plotting the ratio between CPV I_{MP} and direct inclined irradiance (DII) as a function of air mass reveals a nonlinear dependency due to the current mismatch between the multi-junction III-V subcells (Fig. 5 left). For low air mass values, the spectrum is rich in blue light, causing the middle subcell to limit the current. As air mass decreases, the current mismatch increases, leading to greater losses. Conversely, for high air mass values, the top cell becomes the limiting factor for current. These trends are what define the airmass UF function that reflects the sensitivity of the module to the solar spectrum. Thus, the CPV I_{MP} will be that obtained by a classical PV system model multiplied by the UF function. Fig. 5 right shows the histogram of the residuals when using UF to model experimental I_{MP}, proving that bias due to model is significantly reduced.

In addition to the airmass UF, other UFs are also implemented in cpvlib, for example to take into account the effect of lens temperature using the ambient temperature as proxy, but Insolight module’s sensitivity is negligible.

UF are modeled in the CPVSystem class as new methods, using linear piecewise functions, modeled as presented in the equations of Section

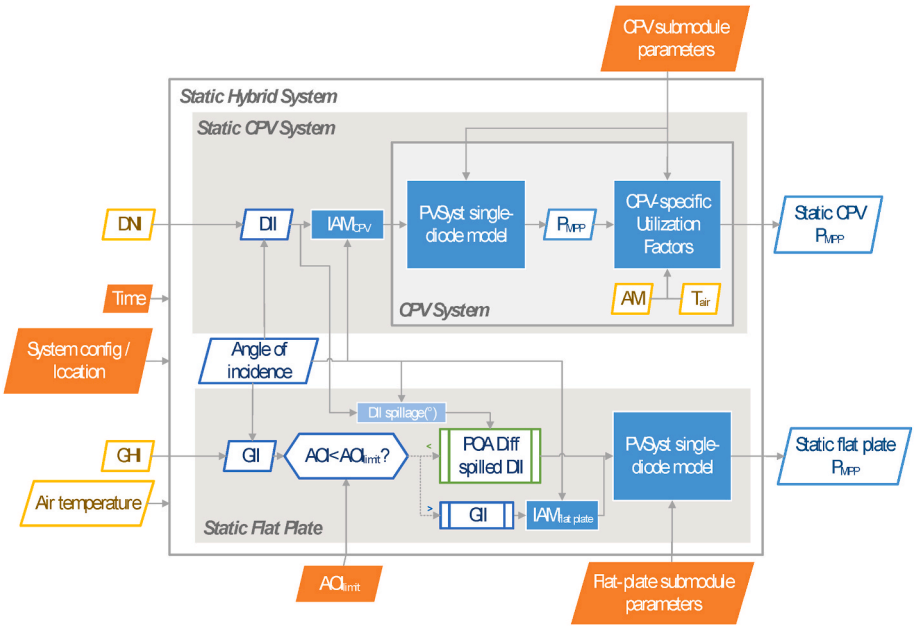


Fig. 3. Detailed schematic of the StaticHybridSystem class designed for simulating integrated-tracking hybrid systems, featuring a CPV submodule capturing direct irradiance and a flat plate PV submodule gathering diffuse irradiance. The rhombuses represent variables: yellow-line ones for ambient inputs, orange ones describe the rest of the inputs and blue-lined are derived results, including main outputs. The modeling blocks are represented by blue boxes and the hexagon represents a conditional bifurcation.

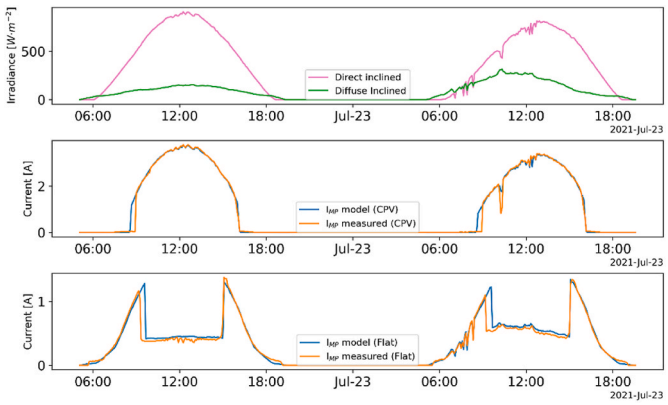


Fig. 4. Hourly evaluation of the performance of the III-V and Flat current submodules within the hybrid module, showcasing a side-by-side comparison between measured and modeled data, plus the available irradiance in each case, direct and diffuse inclined irradiances. By 09h and 15h AOI is 55°, so the AOI limit is reached. Flat plate I_{MP} includes the spillage adjustment described in "Light spillage correction" section.

"Modeling of CPV features. Utilization factors". These functions are obtained as a linear fit, which for the example are shown as fitting lines in FIGURE 5 left.

4.4.2. AOI limit

As previously presented, Insolight modules have a tracking limitation up to a given AOI, which is reflected in both the CPV submodule and the flat plate one. Fig. 4 shows by 09h and 15h the effect of the tracking limit on both the I_{MP} of the CPV subsystem (as it stops being generated) and on the I_{MP} of the flat plate subsystem, as it increases at those times.

In cpvlib the parameter 'aoi_limit' is entered as a threshold value to consider whether the diffuse or the global irradiance is the irradiance seeing by the flat plate submodule. This is implemented in the method get_effective_irradiance () in the StaticFlatPlateSystem class.

4.4.3. Incidence angle modifier (IAM)

The actual behavior of the submodules regarding the AOI is more complex than just a limit angle and can be defined by the Incidence Angle Modifier (IAM), a property that is considered already in pvlib. In Fig. 6 both CPV and flat plate IAM functions for the Insolight are shown, obtained from normalized current to direct and diffuse inclined irradiances respectively.

In both submodules, the get_iam () is directly used in the reimplementation of get_effective_irradiance () method in the StaticCPVSystem class to take into account the specific IAM that includes the AOI tracking limit and in the StaticFlatPlateSystem class. Besides, get_irradiance () is also reimplemented to account for the DNI on the plane of the CPV (static) module.

4.4.4. Light spillage correction

In a hybrid module in which the III-V cells of the CPV submodule are above the Si cells of the flat plate submodule, there is a possibility that as the lenses are moved to keep the light on the III-V cells, the light spot could grow and light spill over the Si cells could occur (see Fig. 7).

Fig. 4 shows the already adjusted flat plate I_{MP}, including the spillage correction as a percentage of the direct inclined irradiance; in this case, without spillage correction the modeled I_{MP} level is approximately 15 % lower than measured. When light spillage is considered, the RMSE between the modeled and measured P_{MP} is reduced from 10.5 % to 3.5 %.

A new parameter 'spillage' is directly used in the reimplementation of get_effective_irradiance () method in the StaticFlatPlateSystem class to take into account the increase of irradiance on the flat plate submodule. It is defined as the fraction of direct irradiance scattered into the flat plate for AOI within the tracking range.

4.5. Expected annual yield of use case module

Once the modeling of a hybrid module has been validated, the parameters describing that module can be used to estimate the annual production at a given location.

Through the geographical coordinates a Typical Meteorological Year (TMY) is obtained directly with the pvlib.iotools.get_pvgis_tmy ()

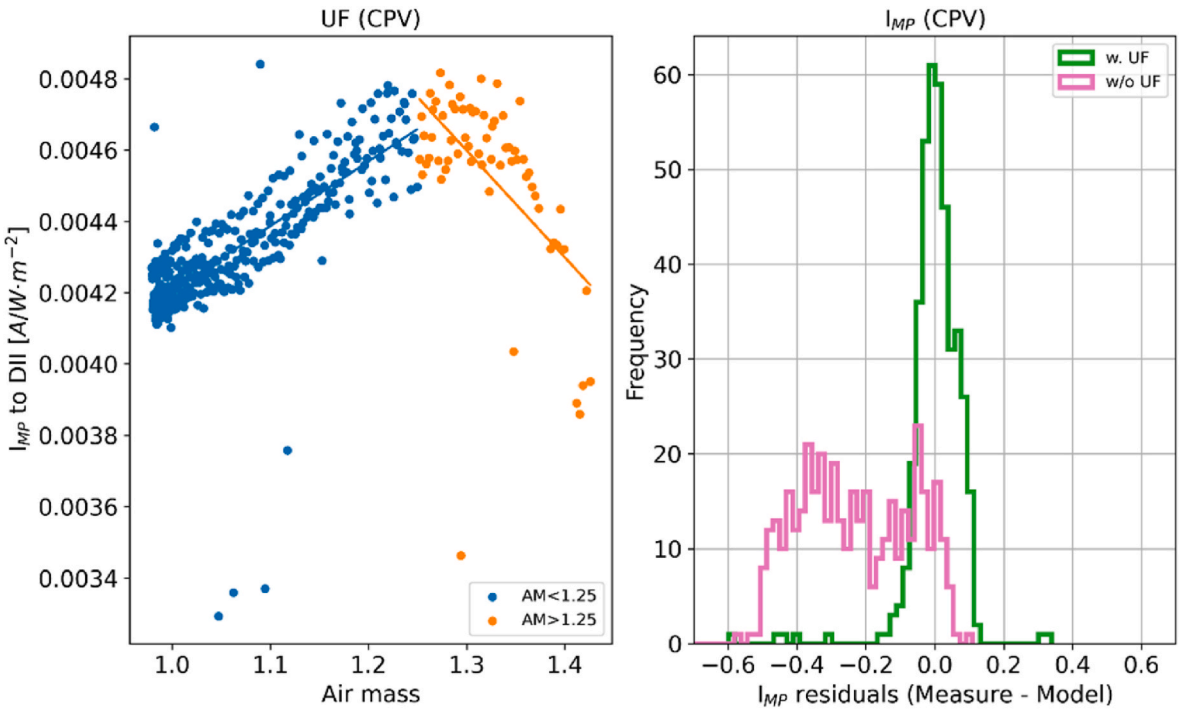


Fig. 5. (Left) Ratio of I_{MP} to Direct Inclined Irradiance (DII), showing the nonlinearity between the two magnitudes. It can also be identified that the mismatch between the currents of the subcells of the CPV system is strongly dependent on the value of the air mass (AM). Linear fits representing the piecewise functions used as UF are also plotted. (right) Histogram of the residuals between measurement minus model: in red without taking into account the UF and in green taking it into account.

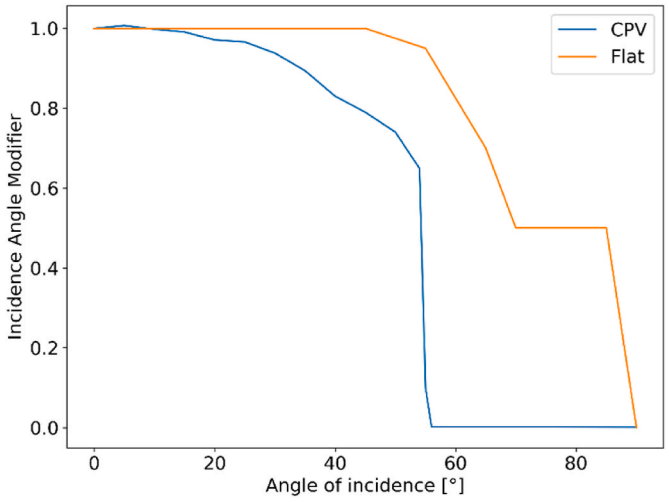


Fig. 6. Incidence Angle Modifier (IAM) for both CPV and flat plate submodules, showing the AOI limit and high AOI effects.

method of pvlib (in this case from the EU PVGIS data source) which is used to obtain the AOI, solar irradiance, ambient temperature and wind speed. From here, the instantaneous power of the Insolight hybrid submodules is obtained, as shown in Fig. 8 for Madrid (Spain). The annual yield of the hybrid module in Madrid is therefore 150.2 kWh for the CPV submodule and 65.1 kWh for the flat plate submodule.

As an example, the code that generates the annual yield and Fig. 8 is presented in Appendix B.

5. User materials, dependencies and license

cpvlib can be easily installed by means of the *pip* tool (Python standard package installer) in https://pypi.org/project/cpvlib/. The source is hosted in a public repository in https://github.com/isi-ies-group/cpvlib/, and a specific website contains its documentation in https://cpvlib.readthedocs.io/.

Its only dependency is the pvlib library itself (including its own dependencies) and Python, at least version 3.6.

cpvlib code is open-source and released under a BSD 3-Clause license.

6. Conclusions

The development of cpvlib as a modeling tool for CPV systems represents a significant advancement in precise and adaptable simulation of these systems. It has been demonstrated that cpvlib overcomes the limitations of existing tools by offering an open, modular, and flexible solution based on the Python pvlib library. This approach not only enhances the accuracy of energy production predictions but also facilitates collaboration and transparency within the solar energy research community.

By validating cpvlib through the modeling of hybrid Si/III-V modules with planar micro-tracking, developed by Insolight, its capability to address advanced configurations and complex effects such as air mass UF, incidence angle limits, and light spillage has been highlighted. Specifically, an RMSE of 6.4 % is obtained for the CPV submodule, which is reduced to 2.7 % when UFs are included. For the flat plate submodule, the RMSE is more than halved to 3.5 % if light spillage is included.

Finally, cpvlib, available under a BSD 3-Clause license, promotes community-driven development and reduces costs associated with solar project financing by offering transparency in algorithms and implementations, inherently lowering risks. In conclusion, cpvlib positions itself as an essential tool for CPV system modeling and simulation, fostering replicability and reproducibility.

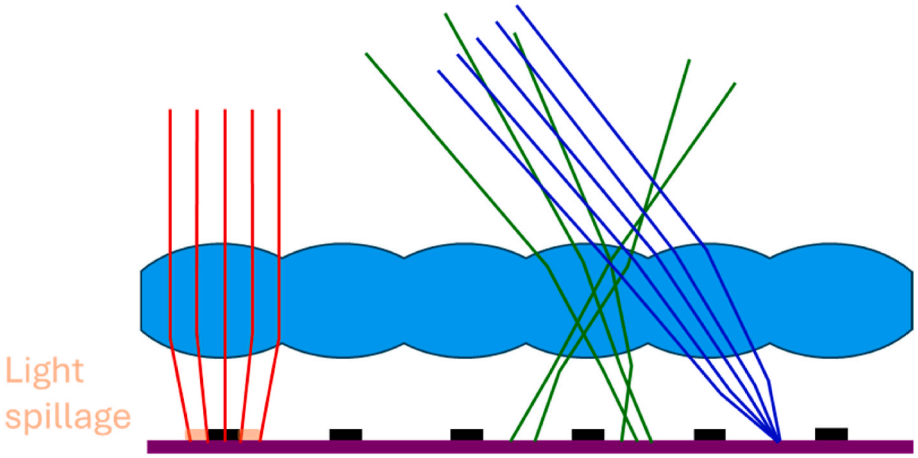


Fig. 7. Schematic diagram illustrating the light spillage that occurs when the light spot on the III-V cell is larger than the cell itself, illuminating also the Si cell.

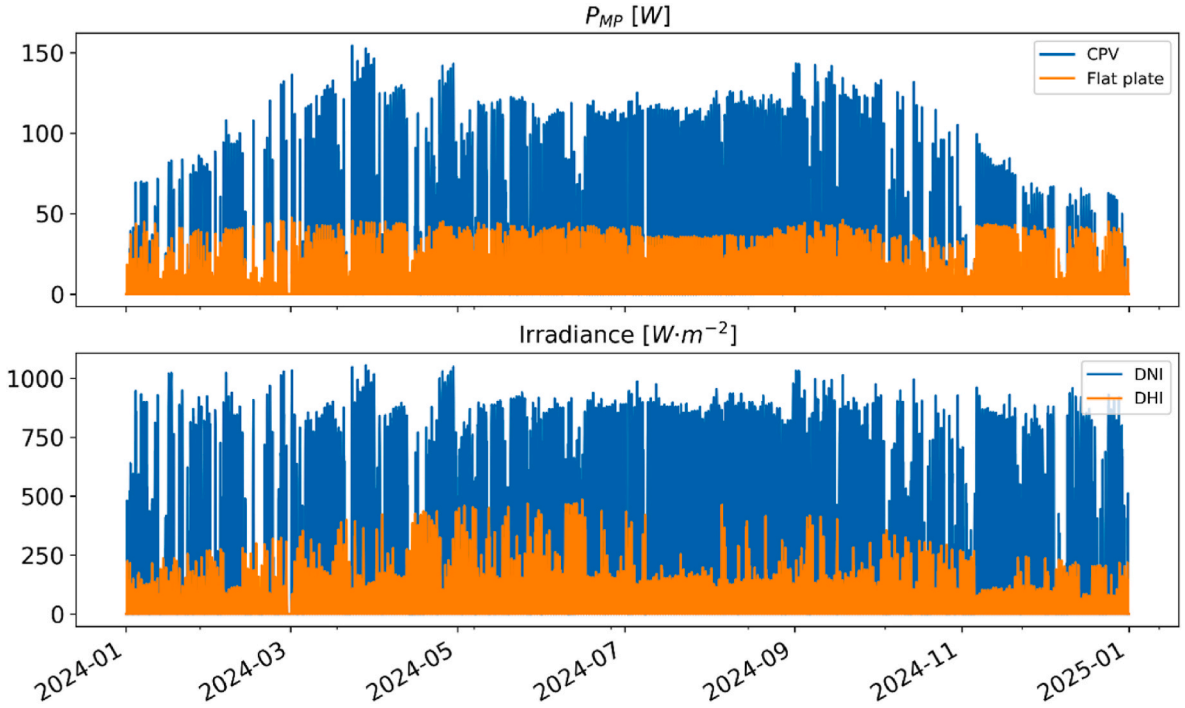


Fig. 8. (Top) P_{MP} for both CPV and flat plate submodules modeled from TMY data from PVGIS for Madrid (Spain). (bottom) DNI and DHI irradiances from that TMY.

CRediT authorship contribution statement

Rubén Núñez: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Marcos Moreno:** Software, Investigation, Formal analysis, Data curation. **Rebeca Herrero:** Writing – review & editing, Methodology, Investigation. **Steve Askins:** Validation, Formal analysis, Data curation. **Ignacio Antón:** Supervision, Resources, Funding acquisition, Conceptualization. **César Domínguez:** Writing – review & editing, Validation, Supervision, Methodology, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the European Union's Horizon 2020 program under Grant Agreement 857775 (project HIPERION) and Grant Agreement 787289 (project GRECO). The work also received support by the grant SMARTWIN TED2021-130920B-C21, funded by MCIN/AEI 10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR", and by the project MICROBEAM ref. PID2021-127810OB-I00, funded by MCIN/AEI 10.13039/501100011033 "ERDF A way of making Europe".

Special thanks to Mathieu Ackermann, Gaël Nardin and the entire Insolight SA team for the development and manufacture of the hybrid modules used in this article.

Appendix A

CPV submodule model parameters

Parameter	Value	Comments
gamma_ref	5.25	PVSyst parameter
mu_gamma	0.0004	PVSyst parameter
I_L_ref	0.97*5	PVSyst parameter
I_o_ref	1.7e-10	PVSyst parameter
R_sh_ref	5226	PVSyst parameter
R_sh_0	21000	PVSyst parameter
R_sh_exp	5.50	PVSyst parameter
R_s	0.01	PVSyst parameter
alpha_sc	0.00	PVSyst parameter
EgRef	3.91	PVSyst parameter
irrad_ref	1000	PVSyst parameter
temp_ref	25	PVSyst parameter
cells_in_series	12	PVSyst parameter
eta_m	0.32	PVSyst parameter
alpha_absorption	0.9	PVSyst parameter
iam_model	'interp'	PVSyst parameter
theta_ref	[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 54, 55, 56,90]	Angles for iam_ref
iam_ref	[1.000, 1.007, 0.998, 0.991, 0.971, 0.966, 0.938, 0.894, 0.830, 0.790, 0.740, 0.649, 0.1, 0.002,0.001]	Incidence angle modifier for direct light on CPV
u_c	9.5	PVSyst parameter
u_v	0	PVSyst parameter
UF parameters		
IscDNI_top	1	I_{SC}/DNI top value
am_thld	1.25	AM threshold
am_uf_m_low	0.4	AM low slope
am_uf_m_high	−0.5	AM high slope
ta_thld	15	Temp. threshold
ta_uf_m_low	0	Temp.low slope
ta_uf_m_high	0	Temp.high slope
weight_am	1	AM weight distribution
weight_temp	0	Temp. weight distribution

Flat plate submodule model parameters

Parameter	Value	Comments
gamma_ref	1.1	PVSyst parameter
mu_gamma	−0.0003	PVSyst parameter
I_L_ref	2.05	PVSyst parameter
I_o_ref	1.5e-10	PVSyst parameter
R_sh_ref	200	PVSyst parameter
R_sh_0	600	PVSyst parameter
R_sh_exp	5.5	PVSyst parameter
R_s	0.6	PVSyst parameter
alpha_sc	0.0	PVSyst parameter
EgRef	1.121	PVSyst parameter
irrad_ref	1000	PVSyst parameter
temp_ref	25	PVSyst parameter
cells_in_series	64	PVSyst parameter
eta_m	0.2	Module efficiency
alpha_absorption	0.6	More light is absorbed in the front cover than in conventional flat plate
iam_model	'interp'	
theta_ref	[0, 5, 15, 25, 35, 45, 55, 65, 70, 80, 85, 90]	Angles for iam_ref
iam_ref	[1, 1, 1, 1, 1, 1, 0.95, 0.7, 0.5, 0.5, 0.5, 0]	Incidence angle modifier for direct light on flat plate beyond tracking range
theta_ref_spillage	[0, 10, 20, 30, 40, 50, 55, 90]	Angles for iam_ref_spillage
iam_ref_spillage	[1, 1, 1.02, 1.16, 1.37, 1.37, 1.37, 1.37]	Incidence angle modifier for the spillage of DNI to the flat plate within the tracking range: spillage (°) = spillage * IAM_ref_spillage (°)
spillage	0.15	Fraction of direct irradiance scattered into the flat plate for AOI within the tracking range
u_c	24	PVSyst parameter
u_v	0.05	PVSyst parameter

Appendix B

Sample workflow code to model the expected annual yield of an Insolight module from a TMY with cpvlib. It reproduces Fig. 8.

```
import matplotlib.pyplot as plt
import pvlib
from cpvlib import cpvsystem

mod_params_cpv = {
    "gamma_ref": 5.25,
    "mu_gamma": 0.0004,
    "I_L_ref": 1*5,
    "I_o_ref": 9e-10,
    "R_sh_ref": 5226,
    "R_sh_0": 21000,
    "R_sh_exp": 5.50,
    "R_s": 0.01,
    "alpha_sc": 0.00,
    "EgRef": 3.94,
    "irrad_ref": 1000,
    "temp_ref": 25,
    "cells_in_series": 14,
    "eta_m": 0.32,
    "alpha_absorption": 0.9,
    "iam_model": 'interp',
    "theta_ref": [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 54, 55, 56,90],
    "iam_ref": [1.000, 1.007, 0.998, 0.991, 0.971, 0.966, 0.938, 0.894, 0.830, 0.790, 0.740, 0.649,
0.1, 0.002,0.001],
}

UF_parameters_cpv = {
    "IscDNI_top": 1,
    "am_thld": 1.25,
    "am_uf_m_low": 0.4,
    "am_uf_m_high": -0.5,
    "ta_thld": 15,
    "ta_uf_m_low": 0.0,
    "ta_uf_m_high": -0.00,
    "weight_am": 1,
    "weight_temp": 0,
}

mod_params_cpv.update(UF_parameters_cpv)

mod_params_flatplate = {
    "gamma_ref": 1.1, #ideality factor
    "mu_gamma": -0.0003,
    "I_L_ref": 2.05,
    "I_o_ref": 1.5e-10,
    "R_sh_ref": 200,
    "R_sh_0": 600, # 4 times Rsh_ref typically acc. PVSYST
    "R_sh_exp": 5.5,
    "R_s": 0.6, #standard value
    "alpha_sc": 0,
    "EgRef": 1.121,
    "irrad_ref": 1000,
    "temp_ref": 25,
    "cells_in_series": 64, # 8*8,
    "eta_m": 0.2,  # module efficiency for pvlib.temperature.PVSyst_cell
    "alpha_absorption": 0.6,
    "aoi_limit": 55,
    "iam_model": 'interp',
    "theta_ref": [0, 5, 15, 25, 35, 45, 55, 65, 70, 80, 85, 90],
    "iam_ref":   [1, 1, 1, 1, 1, 1, 0.95, 0.7, 0.5, 0.5, 0.5, 0],

    "theta_ref_spillage": [0, 10, 20, 30, 40, 50, 55, 90],
    "iam_ref_spillage": [1, 1, 1.02, 1.16, 1.37, 1.37, 1.37, 1.37]
}


heat_dissipation_params_CPV = {'model': 'PVSyst', 'u_c': 9.5, 'u_v': 0}
heat_dissipation_params_flatplate = {'model': 'PVSyst', 'u_c': 24, 'u_v': 0.05}

lat, lon = 40.4, -3.7

data_pvgis = pvlib.iotools.get_pvgis_tmy(lat, lon)

data = data_pvgis[0].rename(columns={
    'Gb(n)': 'dni',
    'G(h)': 'ghi',
    'Gd(h)': 'dhi',
    'T2m': 'temp_air',
    'WS10m': 'wind_speed',
})

data = data.set_index(data.index.map(lambda t: t.replace(year=2024)))
```

```
location = pvlib.location.Location(
    latitude=lat, longitude=lon, altitude=695, tz='utc')

solar_zenith = location.get_solarposition(data.index).zenith
solar_azimuth = location.get_solarposition(data.index).azimuth

# %%
# StaticHybridSystem
static_hybrid_sys = cpvsystem.StaticHybridSystem(
    surface_tilt=30,
    surface_azimuth=180,
    module_cpv=None,
    module_flatplate=None,
    module_parameters_cpv=mod_params_cpv,
    module_parameters_flatplate=mod_params_flatplate,
    modules_per_string=1,
    strings_per_inverter=1,
    inverter=None,
    inverter_parameters=None,
    temperature_model_parameters_cpv=heat_dissipation_params_CPV,
    temperature_model_parameters_flatplate=heat_dissipation_params_flatplate,
    losses_parameters=None,
    name=None,
)

# get_effective_irradiance
data['dii_effective'], data['poa_flatplate_static_effective'] =
static_hybrid_sys.get_effective_irradiance(
    solar_zenith,
    solar_azimuth,
    dii=None,
    ghi=data['ghi'],
    dhi=data['dhi'],
    dni=data['dni'],
    spillage=0.15
)

# PVSyst_celltemp
data['temp_cell_35'], data['temp_cell_flatplate'] = static_hybrid_sys.PVSyst_celltemp(
    dii=data['dii_effective'],
    poa_flatplate_static=data['poa_flatplate_static_effective'] + data['dii_effective'],
    temp_air=data['temp_air'],
    wind_speed=data['wind_speed']
)

# calcparams_PVSyst
diode_parameters_cpv, diode_parameters_flatplate = static_hybrid_sys.calcparams_PVSyst(
    dii=data['dii_effective'],
    poa_flatplate_static=data['poa_flatplate_static_effective'],
    temp_cell_cpv=data['temp_cell_35'],
    temp_cell_flatplate=data['temp_cell_flatplate'],
)

# singlediode
dc_cpv, dc_flatplate = static_hybrid_sys.singlediode(
    diode_parameters_cpv, diode_parameters_flatplate)

# uf_global (uf_am, uf_temp_air)
data['am'] = location.get_airmass(data.index).airmass_absolute
uf_cpv = static_hybrid_sys.get_global_utilization_factor_cpv(
    data['am'], data['temp_air'])

# TMY Power
fig, axs = plt.subplots(figsize=(10, 6), nrows=2, ncols=1, sharex=True, constrained_layout=True)

(dc_cpv.p_mp * uf_cpv).plot(ax=axs[0], legend=True, label="CPV")
dc_flatplate.p_mp.plot(ax=axs[0], secondary_y=False, legend=True, label="Flat plate")
axs[0].set_title(r'$P_{MP}$ [$W$]', fontsize=14)
axs[0].tick_params(axis='both', labelsize=14)

data[['dni', 'dhi']].plot(ax=axs[1], label=['DNI', 'DHI'])
axs[1].set_title(r'Irradiance [$W·m^{-2}$]', fontsize=14)
axs[1].set_xlabel('')
axs[1].tick_params(axis='both', labelsize=14)

# Energy
energy_cpv = (dc_cpv['p_mp'] * uf_cpv).sum()
energy_flatplate = dc_flatplate['p_mp'].sum()
```

Data availability

Code is public, a link to code repository is provided in the paper

References

[1] Concentrator photovoltaics, in: A. Luque, V.M. Andreev (Eds.), Springer Series in Optical Sciences, vol. 130, Springer, Berlin, 2007.

[2] B. Fisher, K. Ghosal, D. Riley, C. Hansen, B. King, S. Burroughs, Field performance modeling of Semprius CPV systems, in: Photovoltaic Specialist Conference (PVSC), 2014 IEEE, 2014, pp. 759–765, https://doi.org/10.1109/PVSC.2014.6925030.

[3] Nate Blair, et al., *System Advisor Model, SAM 2014.1. 14: General description*. No. NREL/TP-6A20-61019, National Renewable Energy Lab. (NREL) (2014). Technical Report.

[4] T. Lambert, P. Gilman, P. Lilienthal, Micropower system modeling with homer, in: F.A. Farret, M.G. Simões (Eds.), Integration of Alternative Sources of Energy, first ed., Wiley, 2005, pp. 379–418, https://doi.org/10.1002/0471755621.ch15.

[5] A. Mermoud, Use and validation of PVSYST, a user-friendly software for PV-system design, in: 13rd European Photovoltaic Solar Energy Conference, H.S. Stephens, Nice, France, 1995 [Online], https://archive-ouverte.unige.ch/unige:119365. (Accessed 25 February 2023).

. (*continued*).

40th, Jun.

[6] A. Mermoud, T. Lejeune, Performance Assessment of a Simulation Model for Pv Modules of Any Available Technology, 25th EU PVSEC, Valencia, Spain, 2010.
[7] T. Gerstmaier, S. van Riesen, A. Gombert, A. Mermoud, T. Lejeune, E. Duminil, Software modeling of FLATCON CPV systems, in: AIP Conference Proceedings, 2010, p. 183 [Online], http://link.aip.org/link/?APCPCS/1277/183/1. (Accessed 8 October 2013).
[8] T. Gerstmaier, et al., Validation of the PVSyst performance model for the concentrix CPV technology, in: AIP Conference Proceedings, Las Vegas, 2011, pp. 366–369, https://doi.org/10.1063/1.3658363.
[9] M. Steiner, et al., CPVIndia – energy yield forecasting with PVsyst, AIP Conf. Proc. (Aug. 2019) 060005, https://doi.org/10.1063/1.5124198.
[10] N. Jost, T. Gu, J. Hu, C. Domínguez, and I. Antón, "Integrated micro-scale concentrating photovoltaics: a scalable path toward high-efficiency, low-cost solar power," Sol. RRL, vol. n/a, no. n/a, p. 2300363, doi: 10.1002/solr.202300363.
[11] C. Domínguez, N. Jost, S. Askins, M. Victoria, I. Antón, A review of the promises and challenges of micro-concentrator photovoltaics, AIP Conf. Proc. 1881 (1) (Sep. 2017) 080003, https://doi.org/10.1063/1.5001441.
[12] G. Nardin, et al., Towards industrialization of planar microtracking photovoltaic panels, AIP Conf. Proc. (Aug. 2019) 040001, https://doi.org/10.1063/1.5124185.
[13] M. Moreno, et al., Expanding PVLIB Python for modeling static CPV systems, in: Proccedings 13th PVPMC Workshop KunShan | 13th PVPMC Workshop KunSha | 08/12/2019 - 13/12/2019 | KunShan, China, Ies, KunShan, China, 2019, pp. 1–2 [Online], https://oa.upm.es/67194/. (Accessed 30 June 2022).
[14] W.F. Holmgren, C.W. Hansen, M.A. Mikofski, Pvlib python: a python package for modeling solar energy systems, J. Open Source Softw. 3 (29) (Sep. 2018) 884, https://doi.org/10.21105/joss.00884.
[15] K.S. Anderson, C.W. Hansen, W.F. Holmgren, A.R. Jensen, M.A. Mikofski, A. Driesse, Pvlib python: 2023 project update, J. Open Source Softw. 8 (92) (Dec. 2023) 5994, https://doi.org/10.21105/joss.05994.
[16] W.F. Holmgren, C.W. Hansen, J.S. Stein, M.A. Mikofski, Review of open source tools for PV modeling, in: *2018 IEEE 7th World Conference On Photovoltaic Energy Conversion (WCPEC) (A Joint Conference Of 45th IEEE PVSC, 28th PVSEC & 34th EU PVSEC)*, Waikoloa Village, HI, IEEE, Jun. 2018, pp. 2557–2560, https://doi.org/10.1109/PVSC.2018.8548231.
[17] A.R. Jensen, et al., Pvlib iotools—open-source Python functions for seamless access to solar irradiance data, Sol. Energy 266 (Dec. 2023) 112092, https://doi.org/10.1016/j.solener.2023.112092.
[18] A.R. Jensen, I. Sifnaios, K. Anderson, *twoaxistracking* – a python package for simulating self-shading of two-axis tracking solar collectors, MethodsX 9 (Jan. 2022) 101876, https://doi.org/10.1016/j.mex.2022.101876.
[19] W.F. Holmgren, D.G. Groenendyk, An open source solar power forecasting tool using PVLIB-Python, in: Photovoltaic Specialists Conference (PVSC), 2016 IEEE 43rd, IEEE, 2016, pp. 972–975 [Online], http://ieeexplore.ieee.org/abstract/document/7749755/. (Accessed 27 January 2017).
[20] S. Askins, et al., Performance of hybrid micro-concentrator module with integrated planar tracking and diffuse light collection, in: 2019 IEEE 46th Photovoltaic Specialists Conference (PVSC), Jun. 2019, pp. 2507–2512, https://doi.org/10.1109/PVSC40753.2019.8980519.
[21] G. Nardin, et al., Industrialization of hybrid Si/III–V and translucent planar micro-tracking modules, Prog. Photovoltaics Res. Appl. 29 (7) (2021) 819–834, https://doi.org/10.1002/pip.3387.
[22] S. Askins, G. Nardin, M. Ackermann, F. Gerlich, C. Dominguez, Outdoor Monitoring of a Hybrid Micro-CPV Solar Panel with Integrated Micro-tracking and Diffuse Capture, Zenodo, 2019, https://doi.org/10.5281/zenodo.3346823. Jul. 23.