



pvlib 2023 update: pvlib-python, pvanalytics, twoaxistracking

pvlib-python

Cliff Hansen (Sandia)
Kevin Anderson (Sandia)
Will Holmgren (DNV)
Mark Mikofski (DNV)
Adam R. Jensen (DTU)
Anton Driesse (PV Performance Labs)

pvanalytics

Cliff Hansen (Sandia)
Will Vining (Sandia)
Kevin Anderson (Sandia)

twoaxistracking

Adam R. Jensen (DTU)
Kevin Anderson (Sandia)

2023 European PVP MC Workshop
Mendrisio, Switzerland, November 9, 2023



- 1** What is pvlib?
- 2** What is pvlib python
- 3** Documentation and Tutorials
- 4** Enhancements since v0.6 (2018)
- 5** Community Growth
- 6** Introductions to pvanalytics and twoaxistracking



What is pvlib?

A python ecosystem of compatible packages for PV systems modeling and analysis that are **community-driven, free, open-source, and well-documented**

pvlib-python

Library of functions for weather-to-power modeling
Customizable end-to-end PV system modeling (ModelChain)
Batteries-included data import library

pvanalytics

Library of functions for analysis of data from PV systems
Filtering and quality checks
Feature labeling: e.g., inverter clipping

twoaxistracking

Simulate two-axis tracking solar collectors
Emphasis on self-shading

Find us at: <https://github.com/pvlib>



What is pvlib python?



A python library for PV performance modeling

Modeling Toolbox

Stand-alone models for:

Atmosphere	Snow
Solar position	Soiling
Transposition	Shading
Bifacial	I-V curves
Temperature	Inverters
Clear-sky	IAM

...and more!

Weather-to-power workflow

Customizable end-to-end PV system modeling
(ModelChain)

Scriptable and automatable by design

Data I/O

Batteries-included data import:

TMY	SURFRAD
EPW	SOLRAD
NSRDB	MIDC
PVGIS	BSRN
CAMS	UO SRML
ECMWF	NOAA USCNR



Each model function has a page with:

- Brief model description
- Inputs: description, data types, units
- Outputs: description, data types, units
- Published reference(s) for the model
- Links to other relevant functions
- Links to relevant gallery examples
- Other notes as needed

Several hundred model-level pages, all built automatically from in-code documentation

<https://pvlib-python.readthedocs.io>

pvlib.iam.ashrae

`pvlib.iam.ashrae(aoi, b=0.05)`

[\[source\]](#)

Determine the incidence angle modifier using the ASHRAE transmission model.

The ASHRAE (American Society of Heating, Refrigeration, and Air Conditioning Engineers) transmission model is developed in [1], and in [2]. The model has been used in software such as PVsyst [3].

Parameters:

- `aoi` (`numeric`) – The angle of incidence (AOI) between the module normal vector and the sun-beam vector in degrees. Angles of `nan` will result in `nan`.

- `b` (`float`, `default 0.05`) – A parameter to adjust the incidence angle modifier as a function of angle of incidence. Typical values are on the order of 0.05 [3].

Returns: `iam` (`numeric`) – The incident angle modifier (IAM). Returns zero for all `abs(aoi) >= 90` and for all `iam` values that would be less than 0.

Notes

The incidence angle modifier is calculated as

$$IAM = 1 - b(\sec(aoi) - 1)$$

As AOI approaches 90 degrees, the model yields negative values for IAM; negative IAM values are set to zero in this implementation.

References

[1] Souka A.F., Safwat H.H., "Determination of the optimum orientations for the double exposure flat-plate collector and its reflections". Solar Energy vol.10, pp 170-174. 1966.

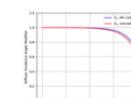
[2] ASHRAE standard 93-77

[3] PVsyst Contextual Help. https://files.pvsyst.com/help/index.html?iam_loss.htm retrieved on October 14, 2019

See also

`pvlib.iam.physical`, `pvlib.iam.martin_ruiz`, `pvlib.iam.interp`

Examples using `pvlib.iam.ashrae`

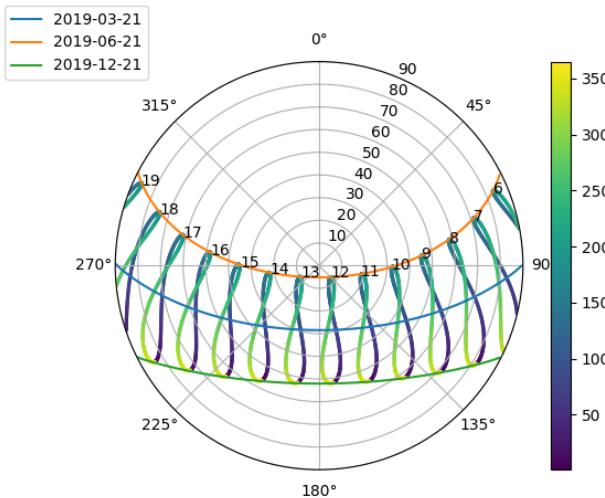


Diffuse IAM Calculation

pvlib python Documentation: Example Gallery

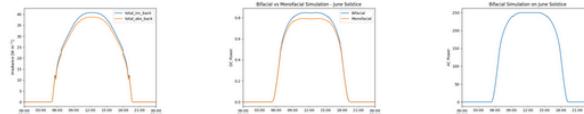


pvlib “cookbook” -- small self-contained scripts for various modeling tasks, intended as a starting point for your own code.



Want to make cool plots like this one?
Check out the example gallery!

Bifacial Modeling

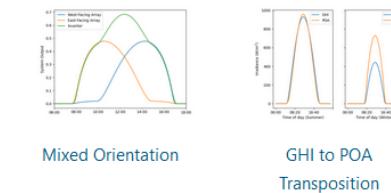


Fixed-Tilt Simulation
with pvfactors

Bifacial Modeling -
procedural

Bifacial Modeling -
modelchain

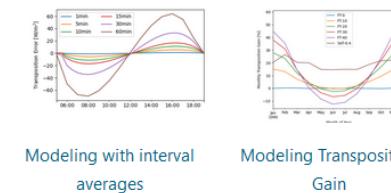
Irradiance Transposition



Mixed Orientation

GHI to POA
Transposition

Seasonal Tilt



Modeling with interval
averages

Modeling Transposition
Gain

HSU Soiling Model Example

Example of soiling using the HSU model.

This example shows basic usage of pvlib's HSU Soiling model [1] with `path_soiling.html`.

References

[1] 1,2,3 M. Coello and L. Boyle, "Simple Model For Predicting Time Series Soiling of Photovoltaic Panels," in IEEE Journal of Photovoltaics, doi:10.1109/JPHOTOV.2019.2919638

This example recreates Figure 3A in [1] for the Fixed Setting Velocity case. Rainfall data comes from Imperial County, CA TMY file PM2.5 and PM10 data come from the ERA. First, let's read in the weather data and run the HSU soiling model:

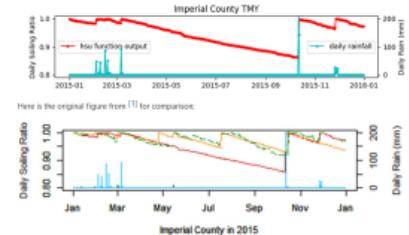
```
import pvlib
from pvlib import pylib as pl
from pytz import timezone
import numpy as np
import pandas as pd

# set full path to the data directory
DATA_DIR = pylib.Path('data') / 'data'

# read rainfall, PM2.5, and PM10 data from file
imperial_county = pd.read_csv(DATA_DIR / 'imperial_county_rainfall.csv', parse_dates=True)
rainfall = imperial_county['rain'].values[1:] # default values from [1] (m/s)
rain_mean_period = pd.Timedelta('1h') # default
cleaning_threshold = 8.0
TOL = 1.0
PM2_5 = imperial_county['PM2_5'].values
PM10 = imperial_county['PM10'].values
soiling_ratio = soiling_hsu_soiling_cleaning_threshold_tilt_pm2_5_pm10_rainfall_mean_period
rate_mean_periodequals_mean_periodequals
```

And now we'll plot the recorded daily soiling ratios and compare with Coello and Boyle Fig 3A:

```
daily_soiling_ratio = soiling_ratio.resample('1D').mean()
fig, ax = plt.subplots(figsize=(12, 3))
ax.plot(daily_soiling_ratio['rain'], 'r', label='Daily Soiling Ratio')
ax.set_xlabel('Daily Soiling Ratio')
ax.set_title('Imperial County 2015')
ax.legend(loc='center left')
# daily ratio = rainfall.resample('1D').mean()
# and so on for PM2.5, PM10, and soiling_ratio
# soiling_ratio = soiling_hsu_soiling_cleaning_threshold_tilt_pm2_5_pm10_rainfall_mean_period
# and so on for PM2.5, PM10, and soiling_ratio
# fig, ax = plt.subplots()
# fig.show()
```



Here is the original figure from [1] for comparison:

Note that this figure shows additional time series not calculated here: modeled soiling ratio using the 2015 PRISM rainfall dataset (orange) and measured soiling ratio (dashed green).

<https://pvlib-python.readthedocs.io/en/stable/gallery/index.html>

pvlib python: Tutorials

In-person and recorded tutorials for:

- Modeling concepts
- Implementation in pvlib

The next one is here, today!

Past tutorials

50th PVSC: <https://github.com/PVSC-Python-Tutorials/PVSC50>

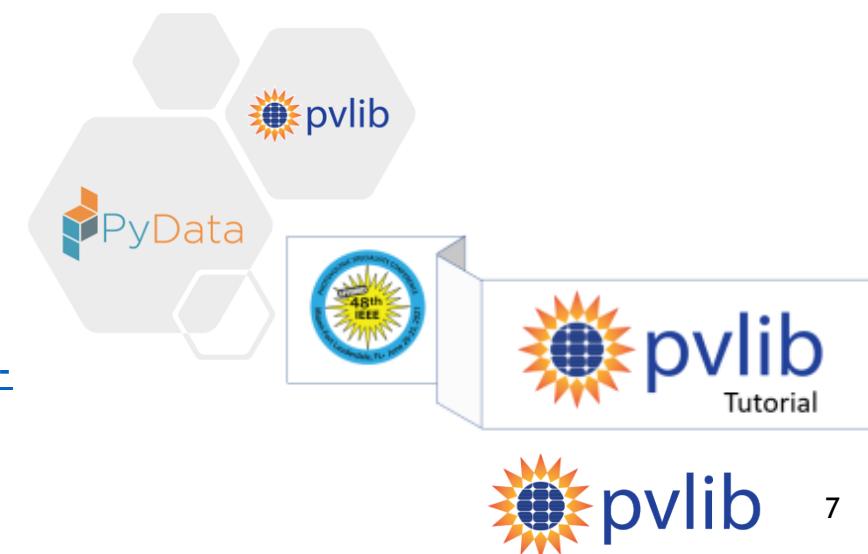
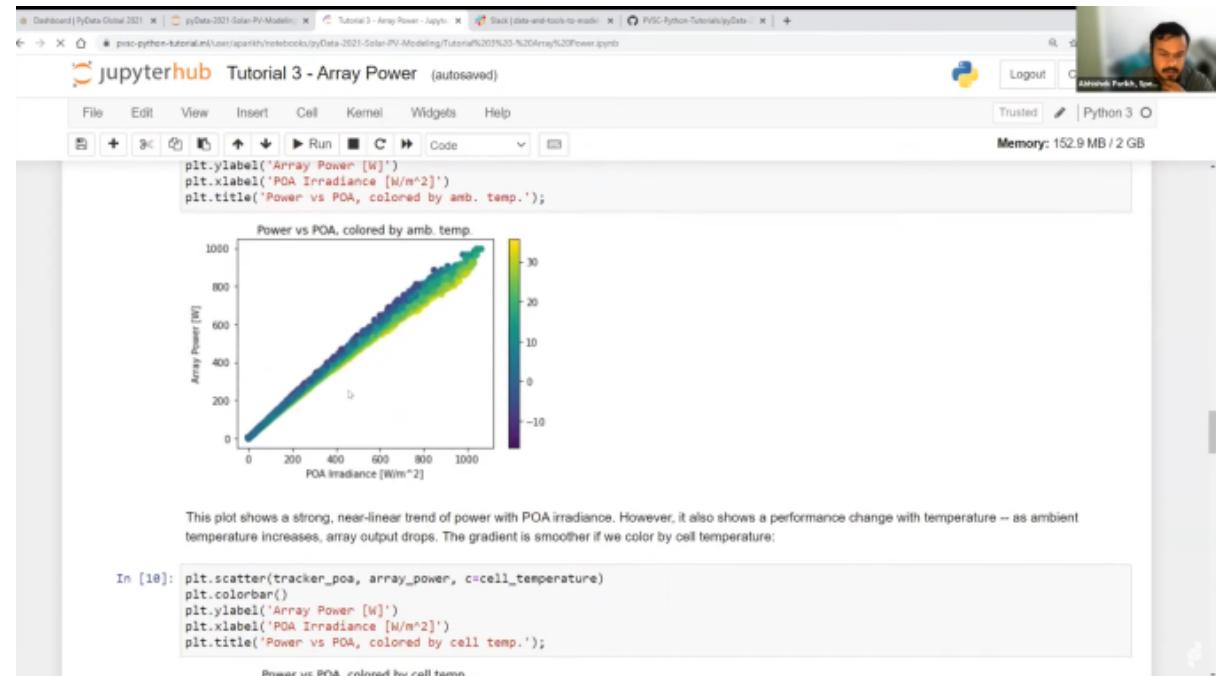
PVPMC 2022: https://github.com/PVSC-Python-Tutorials/PVPMC_2022

PVSC 2021: <https://github.com/PVSC-Python-Tutorials/PVSC48-Python-Tutorial>

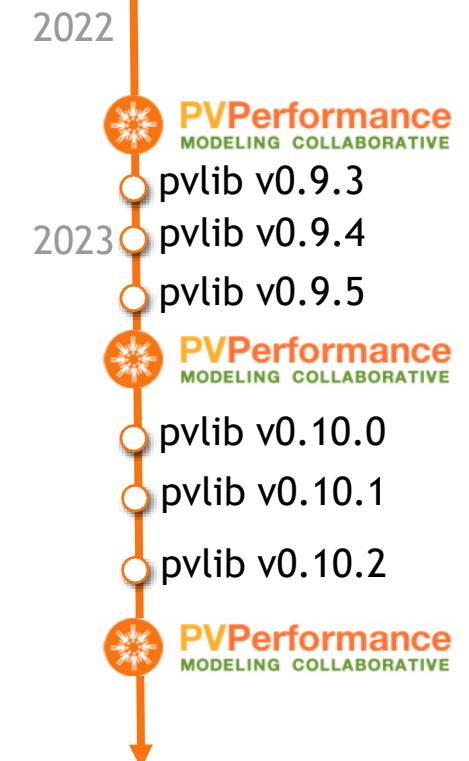
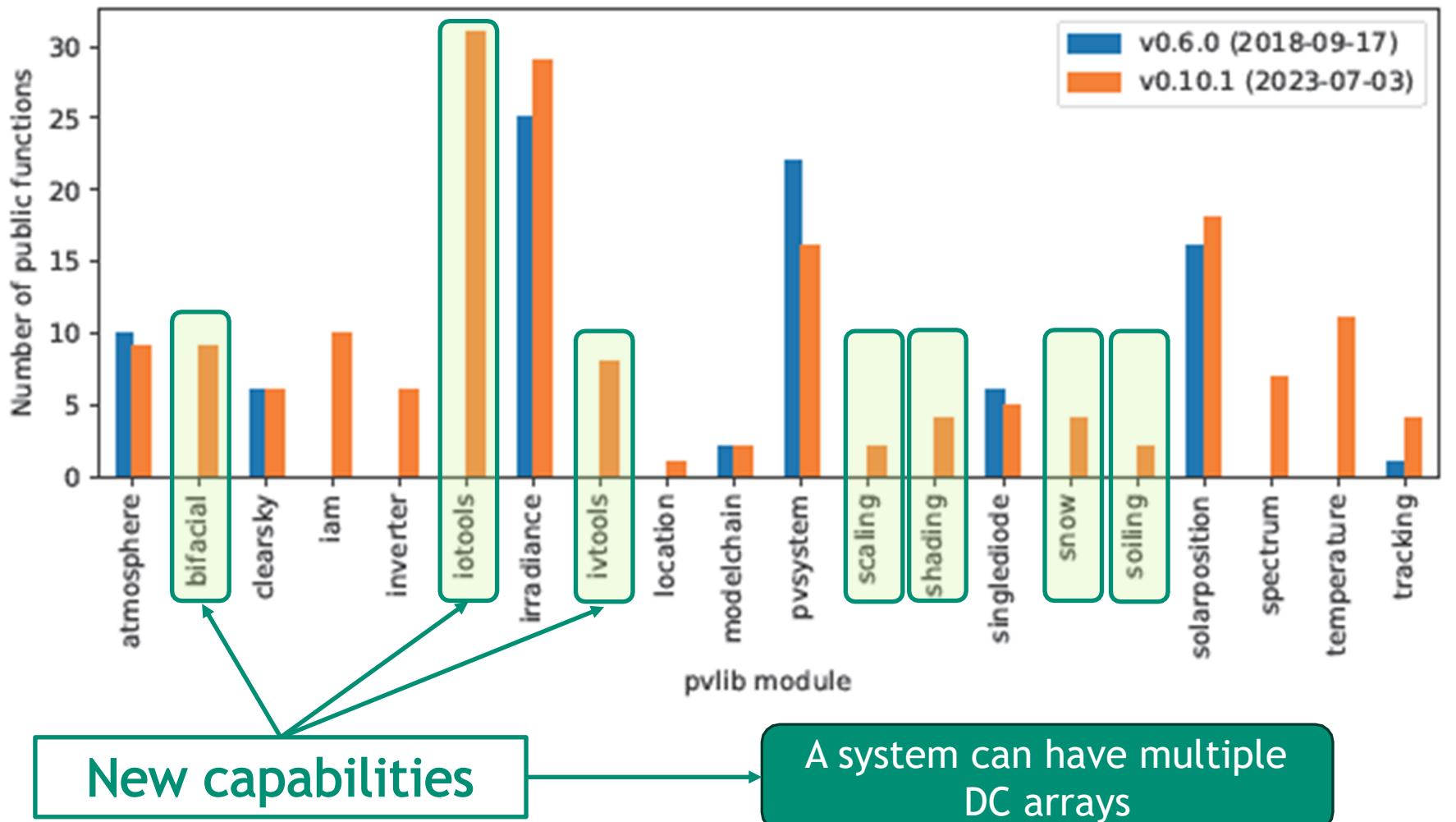
PyData Global 2021

Youtube recording: <https://www.youtube.com/watch?v=sweUakFg3I8>

Source material: <https://github.com/PVSC-Python-Tutorials/pyData-2021-Solar-PV-Modeling>

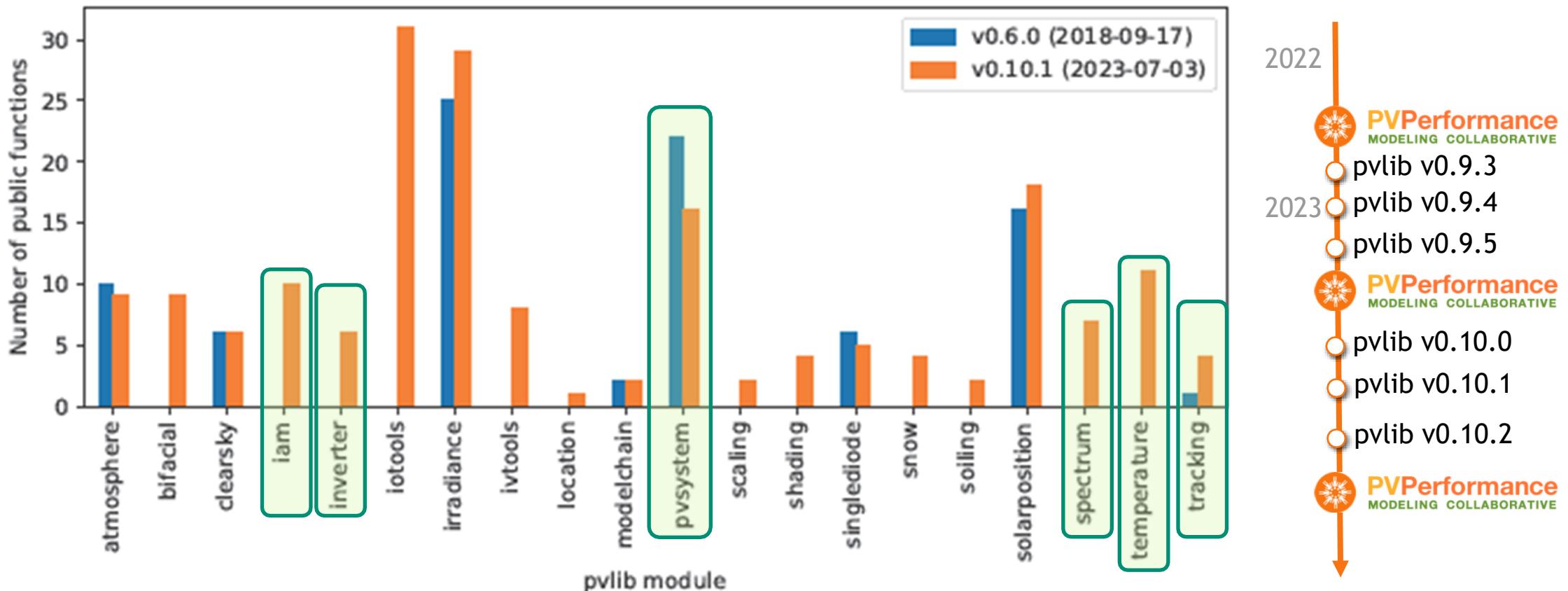


pvlib python since v0.6 (2018)



Full details: <https://pvlib-python.readthedocs.io/en/stable/whatsnew.html>

pvlib python since v0.6 (2018)



Better organization

Full details: <https://pvlib-python.readthedocs.io/en/stable/whatsnew.html>

pvlib python: Community Growth



Google Group (user discussion, announcements)

- 600+ 700+ members
- <https://groups.google.com/g/pvlib-python>

GitHub (code development)

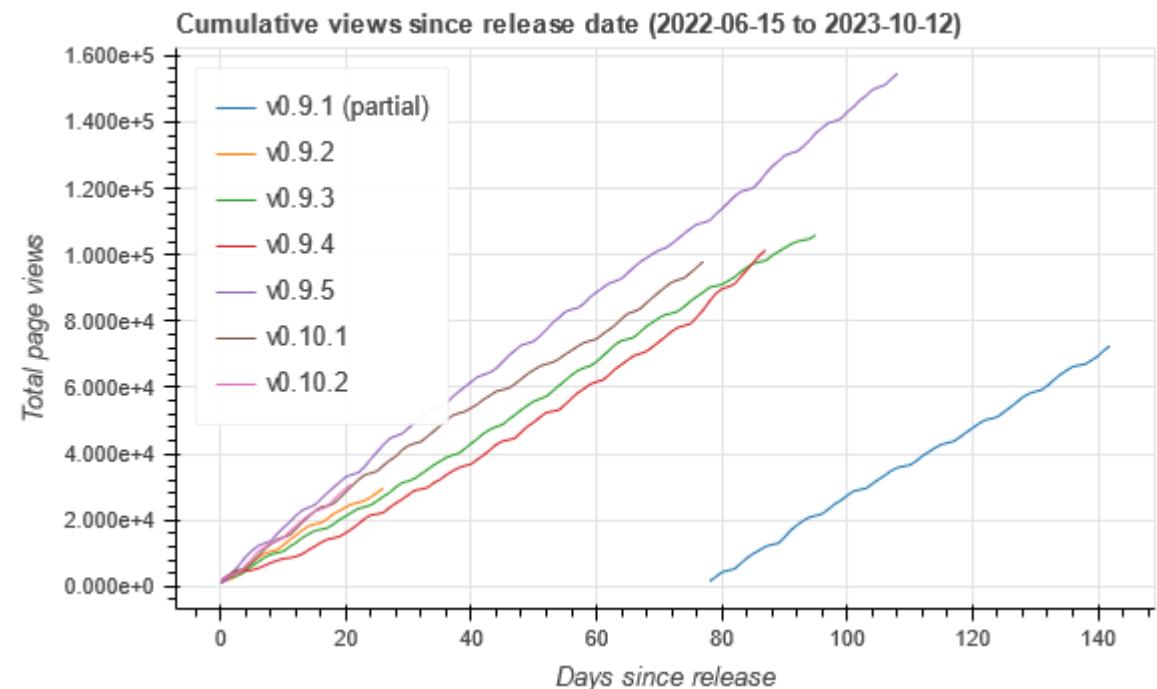
- Code contributions from 80+ 90+ 100+ people
- <https://github.com/pvlib/pvlib-python>

Citations

- 300+ since 2022
- Influence outside of PV modeling, e.g.,

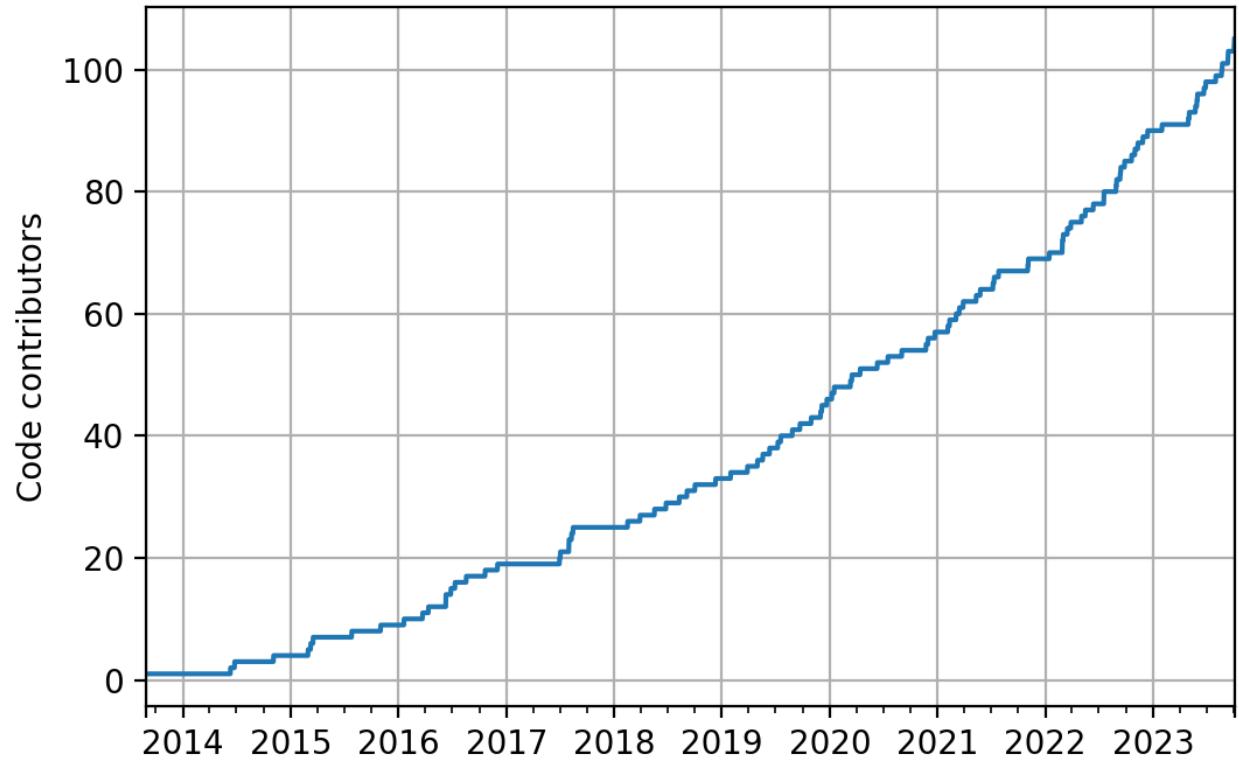
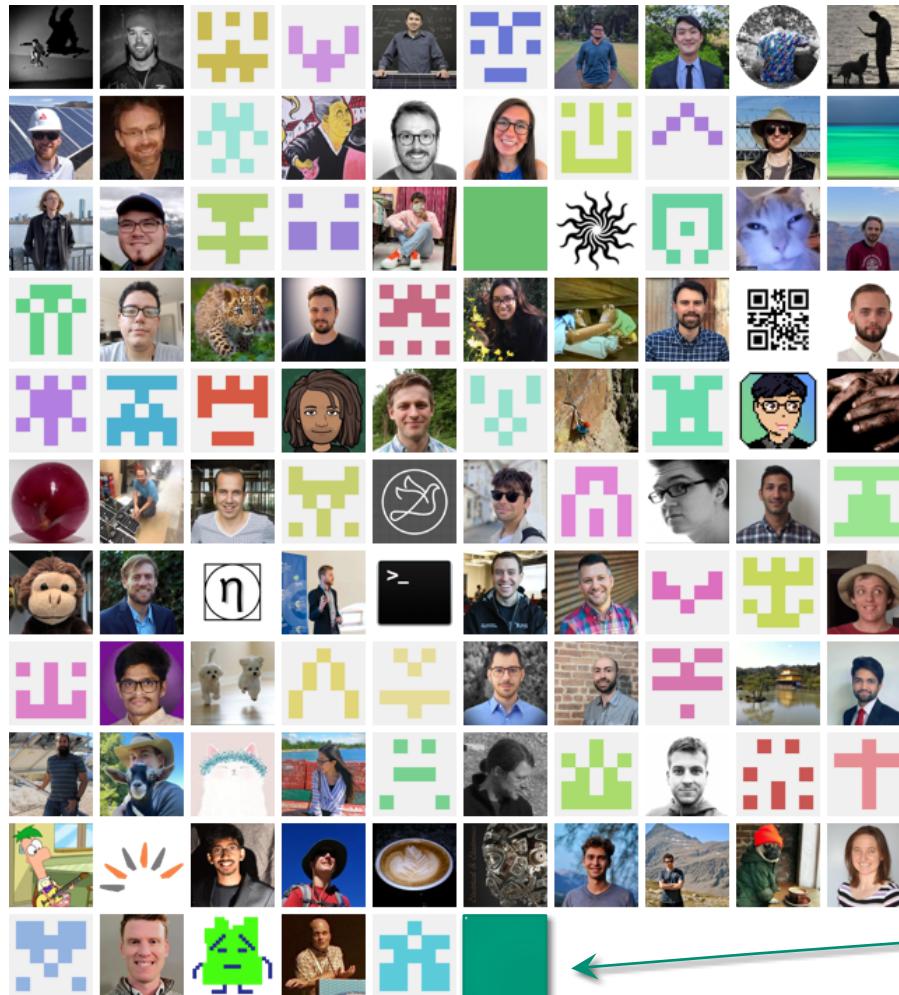
J. Rowland et al., *Scale-dependent influence of permafrost on riverbank erosion rates*. ESS Open Archive. February 09, 2023.

40k page views / month



pvlib python documentation page views

pvlb python: GitHub Contributors



*Not all contributions are code!

This software is made possible by contributions from people like you. [You can help!](#)

<https://pvlb-python.readthedocs.io/en/stable/contributing.html>

What is pvanalytics?



- Workflow-independent library of base functions
- Fully compatible with pvlib-python
- Launched Feb 2020, v0.1.3 Dec 2022
- 6 contributors, 23 forks, 69 stars

Quality control

- Plausibility of irradiance and weather measurements
- Identification of missing, interpolated, or stale data
- Outlier detection
- Identification of timestamp problems such as daylight savings shifts

Feature identification

- Inverter clipping
- Clear-sky periods
- Day/night detection from power or irradiance

Identification of system properties

- Tilt and azimuth from power data
- Differentiation between fixed and tracking PV systems

Metrics

- NREL weather corrected performance ratio

The screenshot shows a GitHub Actions workflow run titled "Check upper and lower limits on daily total irradiance". The run is triggered by a pull request and is currently in progress. It includes a "lint and test" step with results for various operating systems (ubuntu-latest 3.5, 3.6, 3.7, 3.8, macos-latest 3.5, 3.6, 3.7, 3.8, windows-latest 3.5, 3.6, 3.7, 3.8) and a "lint" step for Python 3.5, 3.6, 3.7, and 3.8. All tests and lints have passed, indicated by green checkmarks.

API Reference

Quality

Irradiance

The `check_*_limits_qcrad` functions use the QCRad algorithm [\[1\]](#) to identify irradiance measurements that are beyond physical limits.

<code>quality.irradiance.check_ghi_limits_qcrad(...)</code>	Test for physical limits on GHI using the QCRad criteria.
<code>quality.irradiance.check_dhi_limits_qcrad(...)</code>	Test for physical limits on DHI using the QCRad criteria.
<code>quality.irradiance.check_dni_limits_qcrad(...)</code>	Test for physical limits on DNI using the QCRad criteria.

All three checks can be combined into a single function call.

<code>quality.irradiance.check_irradiance_limits_qcrad(...)</code>	Test for physical limits on GHI, DHI or DNI using the QCRad criteria.
--	---

Irradiance measurements can also be checked for consistency.

<code>quality.irradiance.check_irradiance_consistency_qcrad(...)</code>	Check consistency of GHI, DHI and DNI using QCRad criteria.
---	---

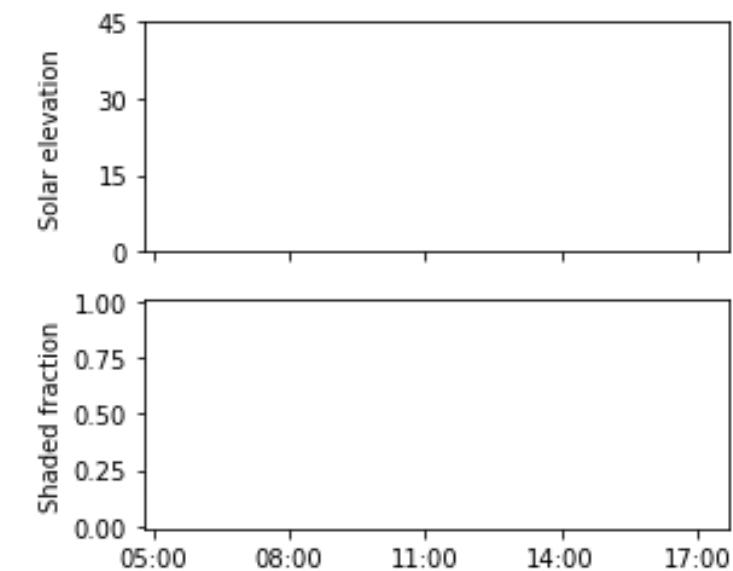
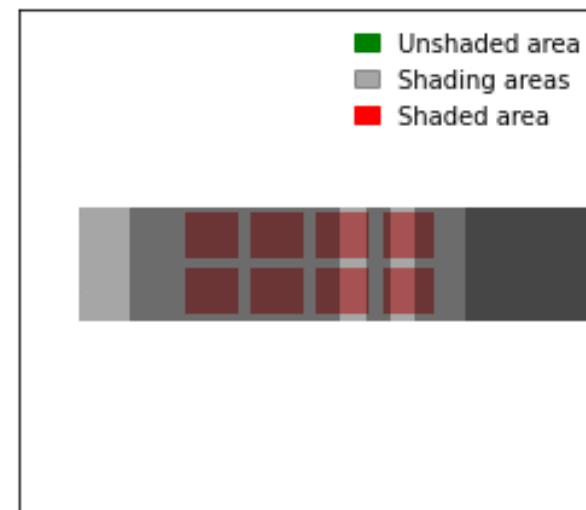
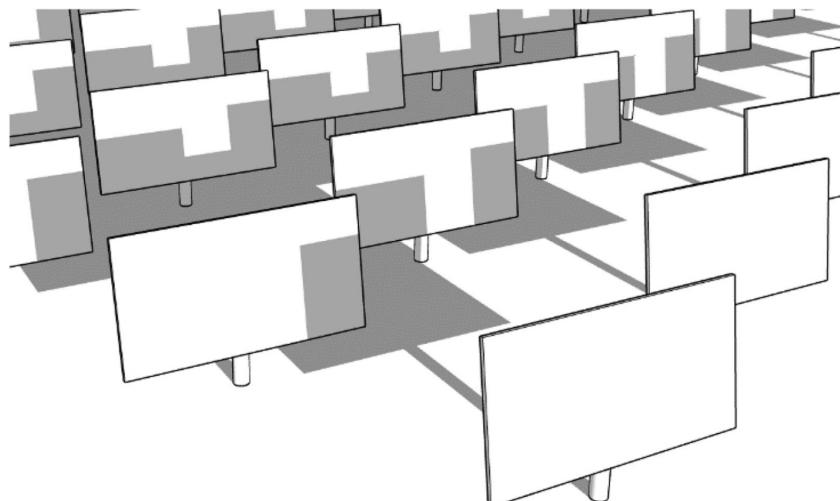
What is pvlib/twoaxistracking?



Shading of two-axis trackers

- Fully customizable field layouts
- Arbitrary panel shape
- Differentiation between active and frame area
- Extensive documentation, validated against literature

Validated against literature!



User's group: pvlib-python, pvanalytics

pvlib-python

Cliff Hansen (Sandia)
Kevin Anderson (Sandia)
Will Holmgren (DNV)
Mark Mikofski (DNV)
Adam R. Jensen (DTU)
Anton Driesse (PV Performance Labs)

pvanalytics

Cliff Hansen (Sandia)
Will Vining (Sandia)
Kevin Anderson (Sandia)
Kirsten Perry (NREL)



Sandia
National
Laboratories





1. Communicate what's planned for pvlib-python and pvanalytics
2. Identify and prioritize development ideas
3. Solicit involvement and contributions
4. Stickers!



Does 1.0.0 come right after 0.10.*?

- No. Expect 0.11.0 to come next. There is no ETA for pvlib 1.0.0 yet ☺

What does 1.0 mean?

- A declaration that pvlib is no longer “beta” (whatever that means)
- Mostly, no more changes that break people’s code (until 2.0, anyway)

What needs to happen before 1.0?

- Package-wide consistency in naming (mostly there already, but still room for improvement)
- Re-organization of code modules (much has already been done)
- Fill in some modeling gaps: transformer losses, direct shading, etc.
- Rewrite/reorg the docs to follow an intentional strategy instead of the current ad-hoc “pile of info”

New features in development: pvlib-python



Additions/improvements where **publications and/or data are available**

- Functions for horizon shading (e.g., input digital elevation data, output angle from ground to horizon) (**stalled**)
- Function(s) for LCOE (started, **help wanted**)
- Cell, module and string electrical mismatch calculations (started, **help wanted**)
- Functions to interchange data with PAN/OND files (started, **help needed**)

Additions where **new publications and/or data may be needed**

- Functions to translate parameters among models:
 - ✓ **temperature models**
 - ✓ **incidence angle modifier (IAM) models**
 - IV curve models
- Functions to fit models to data: temperature, IV curve and other power models

New features requested: pvlib-python



Additions/improvements where publications and/or data are **available**

- **Degradation of DC components**

Additions where new publications and/or data may be needed

- **Better models for inverters:**
 - Off-unity power factor, temperature derating, MPPT voltage limits, current limits, “smarter clipping”
- **Models for DC optimizers, AC transformer losses**

What else?

- Features with available references?
- Features needing research before implementation in pvlib?

Improving the user experience: pvlib-python



Overhaul the documentation

- The “middle” layer between the home page for pvlib-python and each function’s document page

Library of data for model benchmarking and comparison

- Non-expert users can be challenged when faced with e.g. seven clear-sky models
- Provide some guide to assist in selecting appropriate models
- Illustrate model accuracy and limitations

More examples (good progress here)

- We want to have examples showing how to use most major features

✓ More frequent releases (quarterly)

Quarterly community meetings (contact cwhanse@sandia.gov)

What are your ideas?



www.github.com/pvlib/pvanalytics.git

Welcoming contributors, developers, maintainers

Project will mature through contact with users

Guiding principles:

- Workflow-independent library of documented functions
 - Many functions have a common signature: input time series, output Boolean series
 - E.g., clear sky labeling, clipped inverters, GHI passes quality checks
- Workflows are higher level objects that combine sequences of functions for common use cases, e.g. performance ratio calculation
 - Prioritize flexibility
 - **Classes could help automate workflows**
 - **Example workflows should help to clarify definitions**



Thank You

[www.github.com/pvlib/pvlib-python](https://github.com/pvlib/pvlib-python)
<https://pvlib-python.readthedocs.io>

[www.github.com/pvlib/pvanalytics](https://github.com/pvlib/pvanalytics)
<https://pvanalytics.readthedocs.io>

SAND 2023-XXXXC

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.