PV PERFORMANCE MODELLING WITH PVPMC/PVLIB

Steve Ransome[1], Josh Stein[2], Will Holmgren[3], Juergen Sutterlueti[4]

[1]SRCL steve@steveransome.com; [2]Sandia jsstein@sandia.gov;

[3] UA holmgren@email.arizona.edu ; [4]Gantner J.Sutterlueti@gantner-instruments.com

Introduction to PVPMC/PVLIB

The PVPMC (PV Performance Modelling Collaborative) [1] [2] [3] aims to improve the accuracy of PV performance models and their analysis for instantaneous PV performance, predicting energy yield and (with financial assumptions) calculating investment risk.

PVLIB (PV Library) is intended to become PVPMC's standard repository for high quality PV related analysis algorithms particularly from journals, conferences, or white papers. The code is open-source and is collaboratively developed and validated. PVLIB is available in both MATLAB and Python language versions. This paper will concentrate on the Python version.

Why do we need the PVPMC?

Most existing simulation programs such as PVsyst, PVsol, Helioscope, SAM, PVWatts, PVSim(Sunpower) and other in house proprietary programs are effectively “black boxes” in that users cannot always see what algorithms and assumptions have been made (although many are mentioned in the help files). It's difficult if not impossible to add features to enable simulating more complicated systems than the program authors allow (for example multiple array orientations, varied design, different PV panels or inverters). If a bug is found it can be time consuming to get the authors to fix it correctly and then send out an update to all users. There also may be limitations in input options that the user would like to exceed for example wiring losses, V_{MP} tracking limits etc. Creating new algorithms is time consuming and can lead to errors and are generally difficult to individually validate.

PVLIB - Open Source Toolbox

The source code for PVLIB is maintained on Git Hub

https://github.com/pvlib/pvlib-python

Git Hub is a web-based repository hosting service with revision control and source code management using a Web-based graphical interface allowing bug tracking, feature requests, task management, and wiki help files for projects. Advanced users can modify the code on their local systems. If a bug is fixed or a new function is developed, a request can be made through the GitHub system to integrate it into the main code. Requests are then reviewed by the maintainers, if accepted can be integrated and merged into the code. At specified milestones, official version updates will be released which will integrate the latest updates into a down loadable package.

PVLIB code is divided into the following modules

- atmosphere module
- clear sky module
- irradiance module
- location module
- pv system module
- solar position module
- tmy module
- tracking module
- tools module

Example scripts provided take the user through all of the modelling stages from tmy_weather data input to AC power output.

PVLIB has comprehensive documentation

http://pvlib-python.readthedocs.org/en/latest/package_overview.html
http://pvlib-python.readthedocs.org/en/latest/whatsnew.html#v0-3-0-2016

Introduction to Python

Python (https://www.python.org/) is a free and open source high-level programming environment, supported by a wide range of developers, with new features being regularly added and maintained. It also integrates with web, database and graphically intensive processes allowing great flexibility when developing models and algorithms. Python is designed to be easily written and interpreted, and because of the high-level nature of the code it is easily learned by those with a simple understanding of programming syntax perhaps from languages such as Basic, C or Pascal.

The language was named after the 1970s UK comedy program “Monty Python’s Flying Circus”. Help files have several references to their sketches.

Contributing to PVLIB

Many users will just be happy to run PVLIB algorithms as published or read the source to understand the algorithms. More advanced developers may wish to alter or develop new algorithms integrated into the PV LIB Python package which will require algorithmic and physical testing packages. The specific requirements for these are published on the PV LIB website.

Algorithmic testing packages ensure that the module will operate properly under all reasonable conditions imposed on it by users, including logical error handling.

Physical testing should include a validation dataset (included in the test file) which demonstrates the implied function of the algorithm. This provides users with an accessible method of ensuring the accuracy of results and ensuring the physical validity of their models.

Python Libraries

The following external libraries are used with Python for greater productivity.

Numpy http://www.numpy.org/ support for large, multi-dimensional arrays and matrices, high-level maths functions.
Scipy http://www.scipy.org/ scientific and technical optimisation, linear algebra, integration, interpolation, special functions.
Pandas http://pandas.pydata.org/ data manipulation and analysis, manipulating numerical tables and time series.
Matplotlib http://matplotlib.org/ plotting library for Python NumPy. I
Seaborn
https://stanford.edu/~mwaskom/software/seaborn/ visualization library based on matplotlib.for attractive statistical graphics.

PVPMC Website contents:

The PVPMC website [1] currently contains the following information

- **Modelling Steps**: includes descriptions of the mathematical formulations required to simulate a PV system.
- **Research**: descriptions of current PV modelling research projects submitted by members.
- **Applications and Tools**: includes specific modelling packages (PVLIB, GridPV-for modelling PV on distribution feeders, Wavelet variability model, and datasets).
- **Resources and Events**: includes workshop proceedings, documents, list of references and variable names, blog, and email sign-up.

PV performance Modelling steps

The steps modelled by the PVPMC are illustrated in Figure 1

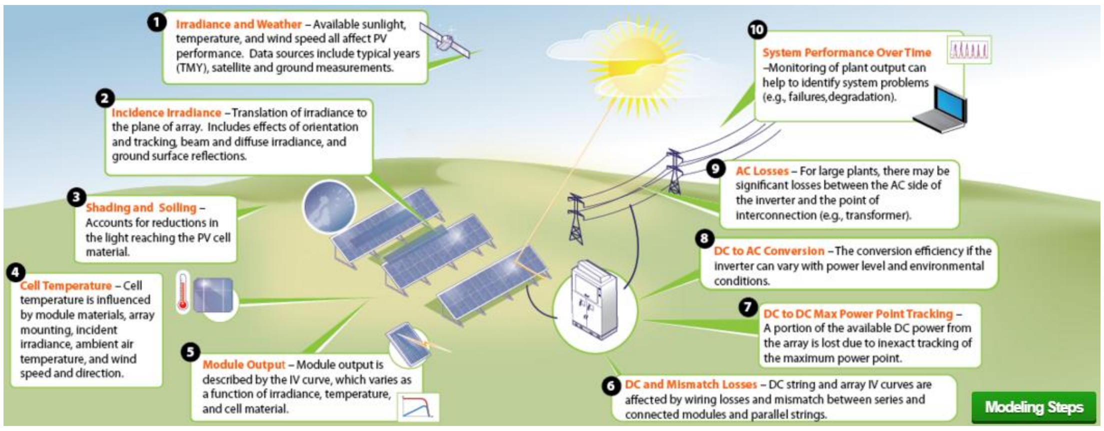

1 Irradiance and Weather – Available sunlight, temperature, and wind speed all affect PV performance. Data sources include typical years (TMY), satellite and ground measurements.
2 Incidence Irradiance –Translation of irradiance to the plane of array. Includes effects of orientation and tracking, beam and diffuse irradiance, and ground surface reflections.
3 Shading and Soiling – Accounts for reductions in the light reaching the PV cell material.
4 Cell Temperature – Cell temperature is influenced by module materials, array mounting, incident irradiance, ambient air temperature, and wind speed and direction.
5 Module Output – Module output is described by the IV curve, which varies as a function of irradiance, temperature, and cell material.
6 DC and Mismatch Losses – DC string and array IV curves are affected by wiring losses and mismatch between series and connected modules and parallel strings.
7 DC to DC Max Power Point Tracking – A portion of the available DC power from the array is lost due to inexact tracking of the maximum power point.
8 DC to AC Conversion – The conversion efficiency if the inverter can vary with power level and environmental conditions.
9 AC Losses – For large plants, there may be significant losses between the AC side of the inverter and the point of interconnection (e.g., transformer).
10 System Performance Over Time –Monitoring of plant output can help to identify system problems (e.g., failures,degradation).
Modeling Steps

Fig 1. Modelling stages (from the PVPMC website)

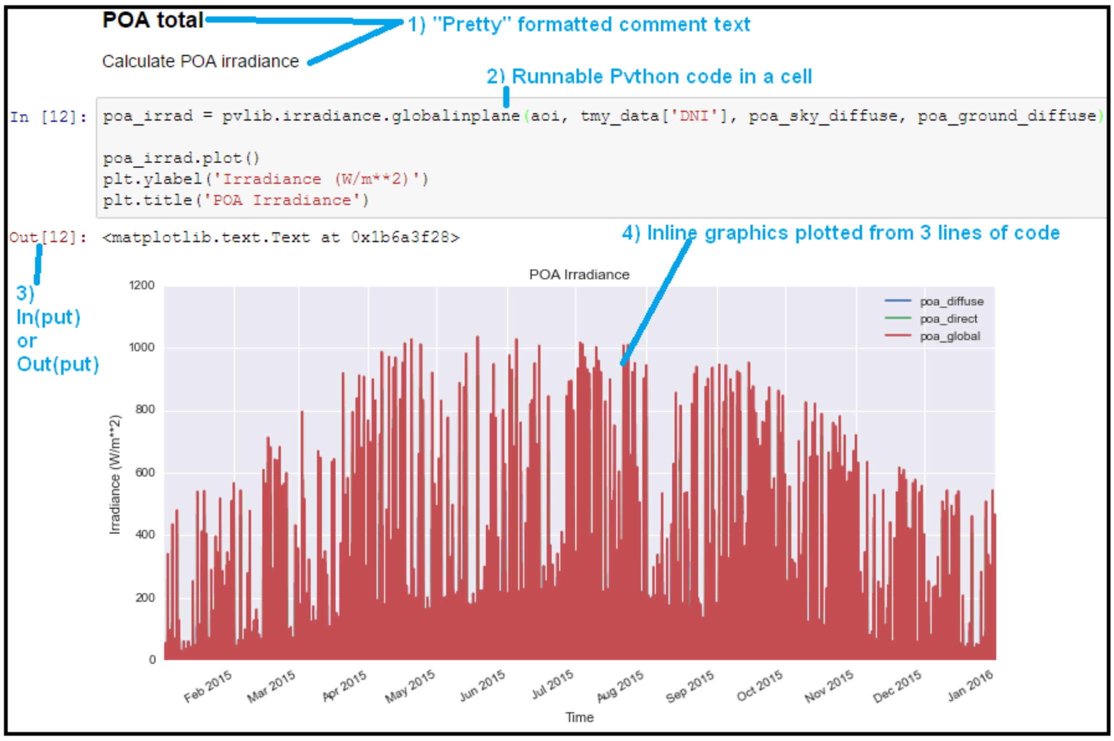
Running PVLIB Python scripts

The standard environment for developing and running PVLIB Python is “iPython Notebook” http://ipython.org/notebook.html.

The interface runs in a web browser such as Chrome, Firefox or Safari on a PC, Apple or Unixas shown in Figure 2.

Unlike many development environments it allows the following

1) “Pretty” formatted code (using different font sizes and bold for the text)
2) Interpreted language in cells.
3) Input or Output for interactive work.
4) Embedded graphics



```
In [12]: poa_irrad = pvlib.irradiance.globalinplane(aoi, tmy_data['DNI'], poa_sky_diffuse, poa_ground_diffuse)

         poa_irrad.plot()
         plt.ylabel('Irradiance (W/m**2)')
         plt.title('POA Irradiance')

Out[12]: <matplotlib.text.Text at 0x1b6a3f28>
```

Fig 2. Example Python session in iPython Notebook

Figure 3 illustrates some typical PVLIB Python code – this illustrates the SAPM thermal model to determine the module and cell temperature rises above ambient as functions of irradiance, wind speed and module mounting.

```
def sapm_celltemp(                                                          function name
    irrad, wind, temp, model='open_rack_cell_glassback'):                   and input series

    temp_models = {'open_rack_cell_glassback': [-3.47, -.0594, 3],          list models
                   'roof_mount_cell_glassback': [-2.98, -.0471, 1]          and values
                   etc. }

    a = model[0] b = model[1] deltaT = model[2]                             get model values
    E0 = 1000. # Reference irradiance                                       set reference

    temp_module = pd.Series(irrad*np.exp(a + b*wind) + temp)                calc Tmod (C)
    temp_cell = temp_module + (irrad / E0)*(deltaT)                         calc Tcell (C)

    return pd.DataFrame(                                                    output all
        {'temp_cell': temp_cell, 'temp_module': temp_module})               data series
```

Fig 3. Typical (simplified) Python code with explanations (right).

PVLIB functions need to be validated with measured data. Figure 4 shows a correlation between six anisotropic or isotropic sky models calculated vs. measured GI compared with Gantner Instruments measured data at their Tempe site [4].

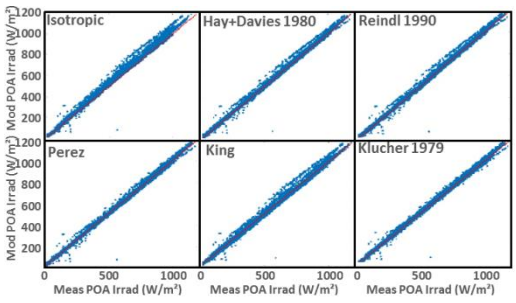


Fig. 4. Comparing calculated vs. measured tilted plane irradiance from six PV_LIB anisotropic or isotropic diffuse sky models [4].

Module and cell temperatures rise above ambient depending on plane of array irradiance, wind speed, the manufacturing technology (e.g. glass-glass, glass-polymer etc.) and also the mounting method (e.g. freely ventilated back, insulated back, bipv etc.)

Figure 5 validates the modelled vs. measured average temperature rise above ambient vs. wind speed (x axis) and irradiance (plots) for a free back CdTe module at GI's Tempe site using the default PV_LIB coefficients. A good overall agreement can be seen with discrepancies generally <±2C.

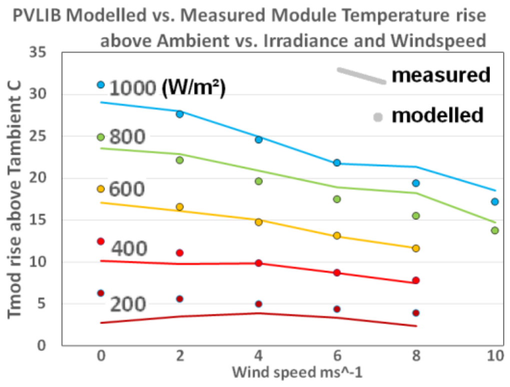


Fig. 5. PV_LIB modelled vs. average measured module temperature rise above TAMBIENT for a CdTe module at GI Tempe against wind speed (x axis) and irradiance (lines) [4].

Figure 6 shows how 3rd parties such as SRCL can use the iPython Notebook environment and PVLIB functions – the graph shows the Marion NREL dataset [5] analysed using the Gantner Instruments/SRCL LFM.

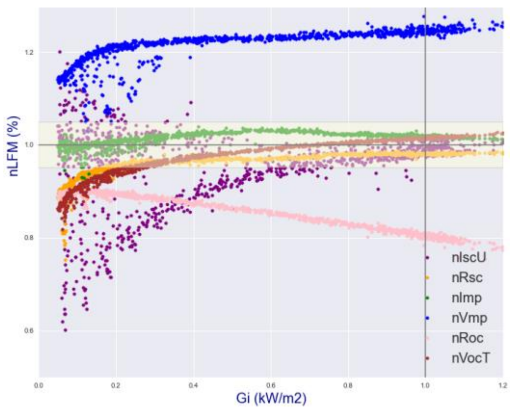


Fig 6. SRCL analysis of 3rd Party (NREL) data using the GI/SRCL LFM.

Conclusions

- PVMPC/PVLIB have been introduced with links to their website and details of their workshops.
- Anyone interested should download the toolbox and is encouraged to learn Python and contribute.
- SRCL can't get to the 5th Workshop but should be at the 6th if anyone has any requests for information.

Dates of next PVPMC Workshops

5th : 9th May, 2016 Santa Clara, USA

6th : 24-25th Oct, 2016 Freiburg, Germany.

Acknowledgements

PVPMC contributors, Rob Andrews (Heliolytics).

References

[1] PVPMC www.pvpmc.org
[2] PVSC 2014 Andrews, R. et al. "Introduction to the Open Source PV LIB for Python Photovoltaic System Modelling Package". 40th PVSC 2014 http://energy.sandia.gov/wp-content//gallery/uploads/PV_LIB_Python_final_SAND2014-18444C.pdf
[3] PVSC 2015 Holmgren, W. et al "PVLIB Python 2015", 42nd PVSC 2015 https://github.com/pvlib/pvsc2015/blob/master/pvlib_pvsc_42.pdf http://nbviewer.jupyter.org/github/pvlib/pvsc2015/blob/master/paper.ipynb
[4] Sutterlueti et al "Improved PV Performance Modelling by Combining the PV_LIB Toolbox with the Loss Factors Model (LFM)" 42nd PVSC 2015
[5] "User's Manual for Data for Validating Models for PV Module Performance" W. Marion et al, NREL CO, NREL/TP-5200-61610