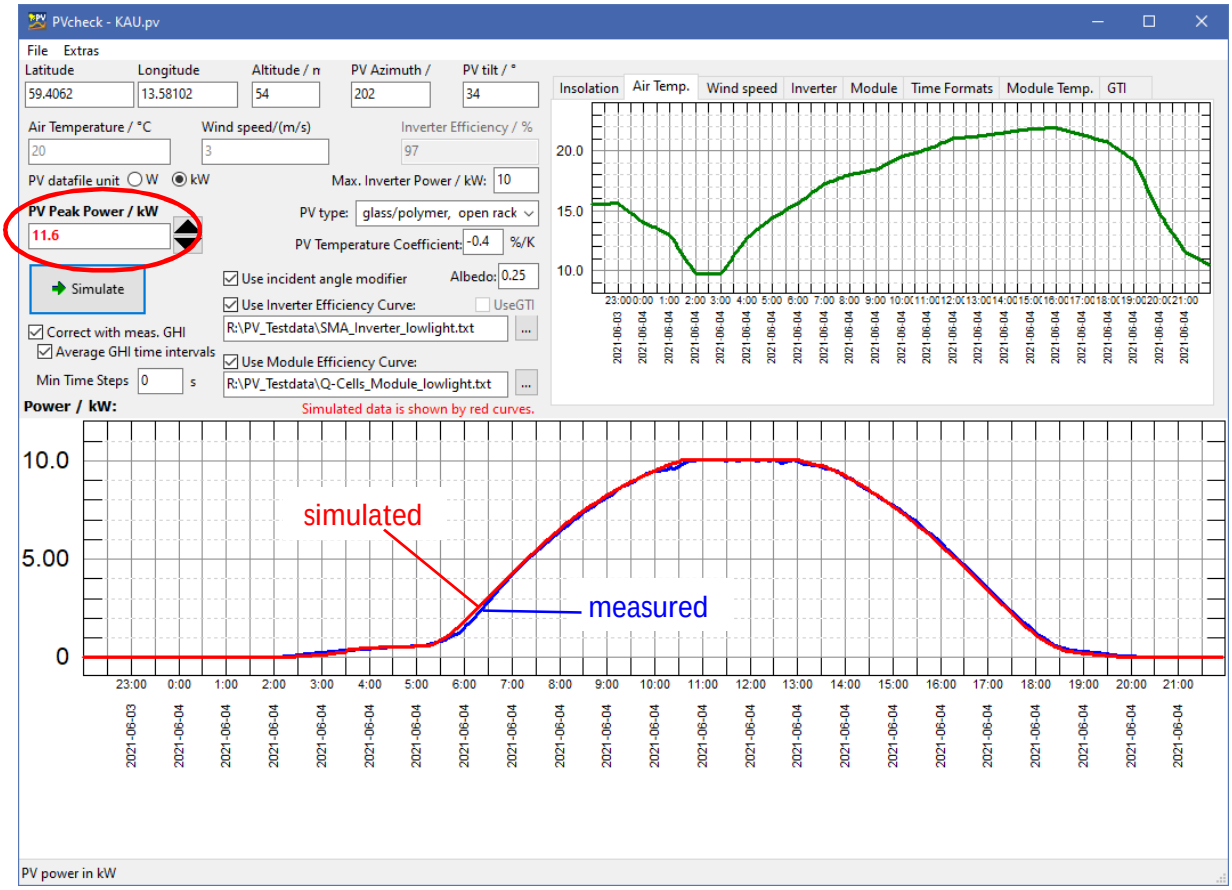
PVCheck

Version 0.35 for Windows

by Markus Rinio, Karlstad University



1 PVCheck manual

1.1 Introduction

The software PVCheck for Windows was developed at Karlstad University to give users of photovoltaic (PV) systems an easy tool to check their system performance.

Assume, you have a PV system and want to know if it performs correctly. Comparing the energy yield with the expected one under a certain period needs a long time and is weather dependent, which means that you can't be sure if a lower-than-expected yield is due to bad weather or to system failures. Also the effect of shadowing under certain hours of the day by trees and buildings is typically not part of the yield that your PV system seller predicted, which might add some extra uncertainty.

Based on the production data of a single sunny day, PVCheck calculates your system performance.[1]

The software simulates the system power as a function of time for a sunny day and compares it to the produced power curve by the PV system. The produced power curve can be downloaded from most inverters via software provided by their sellers. Hours where PV modules lie under shadow are easy to identify by eye when comparing with the simulation.

The simulation is based on the free library "PVlib python", ported from PVlib matlab code, originally developed at Sandia National Laboratories [1]. The program PVCheck consists of two parts: Part 1 is a short PVlib python script, which is packed together with the python library in one huge executable file. Part 2 is a graphical user interface (GUI), written with the free Pascal environment Lazarus.

The efficiency of solar cells is significantly temperature dependent. To account for this, air temperature and wind speed is needed to calculate the actual temperature of the solar cells inside the modules. These information can be given as two constant values, but we strongly recommend to download time-series of air temperature and wind speed from a local provider (e. g. SMHI in Sweden) for your place. The GUI can automatically interpret a large number of file formats downloaded.

The radiation of a clear blue sky can already be simulated quite well by time and place information. Even the typical turbidity of the air for each month and place is taken into account from a special database. To include further day-to-day deviations from varying aerosol and water concentrations of the air, irradiation data (global horizontal irradiation, GHI) from a nearby weather station can be used to fine-adjust the simulation.

Additional information about the intensity dependent efficiency of your inverter and the solar modules can be added to improve the simulation as described in chapters 3.3.3 and 3.3.4.

Within the program, the system peak power can be adjusted to get the best match of the simulation with the measured curve. The result will be your system peak power.

The so obtained system peak power can be compared to the expected peak power (nominal system peak power including aging effects). Presently, the accuracy of the simulation is roughly guessed to be about a few percent. Stronger deviations than 10 % might indicate problems of the PV system.

NOTE: The accuracy of this calculation is by no means guaranteed. Until now, PVCheck's accuracy is not sufficiently tested.

[1] You can improve the accuracy by averaging the obtained system power for several days.

2 General information

The software PVCheck simulates PV systems where all modules have the same orientation. Mixed systems are not supported.

The software runs under Microsoft Windows. It was tested on Windows 7 and Windows 10.

Part 1 of the program is unfortunately really huge (over 300 MBytes), although the python script has only 7 kBytes size. This is due to the fact that PVlib is written to be interpreted in Python and not to be compiled. It can take days to get a well running python installation, therefore we used a script named "pyinstaller" that packs together everything into a single executable file (PVlib.exe). It can be guessed that lots of unnecessary code is also packed into this file, but until now we have not found a way to avoid this. Any help from other users is welcome.

Part 2 (PVcheck.exe) is the main windows program containing the graphical user interface (GUI) that makes life easier. This is the only program that the user has to start. The Python program will be automatically loaded once this main program is started. It will be opened in a separate window, where also possible errors of the Python program will be displayed.

Make sure that some free space is available on your windows system partition for the python program to work correctly. It is recommended not to use too small time intervals (< 5 min) in your data files, because this leads to long calculation times and in some cases the software might freeze.

The program remembers choices and settings you made and stores them in a file "PV.ini" in the same folder where itself is located.

2.1 Known issues

2.1.1 Temp files

Due to an issue with pyinstaller (reported at https://github.com/pyinstaller/pyinstaller/issues/2379), each time PVCheck starts the python program, a large data folder named like **_MEIxxxx** is placed in the users directory by the python program PVlib.exe that was generated by pyinstaller, for example under

C:\Users*account*\AppData\Local\Temp (*account* is your user account name)

As a workaround, PVCheck will offer you to delete folders with these names in that directory when the program is closed. Don't delete them if you want to keep such folders, for example if another program or yourself created them.

Not deleting these folders might lead to low hard disk space at the end.

Until now this was only tested on one operation system. Please check, if the python program increasingly occupies other parts of your harddisk on your system and in that case report that back to the author to be fixed in a later version of PVcheck.

2.1.2 File select box

When opening a data file in PVCheck.exe, the file-open dialog box should display the last file opened. For some unknown reasons, its filename is only partly displayed, although it will be opened correctly. You can see the full file name by clicking on the name and press the home-button (Pos1) of your keyboard. This seems to be a Lazarus issue (reported at https://forum.lazarus.freepascal.org/index.php?topic=31796.0) that was not solved until Lazarus version 2.0.10.

3 Using “PVCheck”

3.1 Basic information

The simulation needs a few information about your PV system.

- *Longitude* and *latitude* of the PV place (GPS coordinates in decimal form like 13.5123, 58.4321)
- *Altitude* of the PV place (not very important, used for optical atmospheric corrections)
- *PV tilt* of the modules in degrees (0° = horizontal, 90° = vertical like in a facade, typical values are between 20° and 40° for modules on a tilted roof)
- *PV azimuth* (horizontal direction of the modules, e. g. 90° = east, 180° south, 270° = west)
- *Temperature* (constant air temperature, not used if time-dependent temperature data is loaded)
- *Wind speed* (constant value for wind chill temperature correction, not used if time-dependent wind speed data is loaded)
- ***PV peak power*** (this value can be adjusted until it best fits to the simulation). This value includes losses from cables and connectors as well as aging.
- *PV type* (used for temperature calculations, e. g. glass/polymer, open rack). This parameter has a strong influence on the assumed cell temperature and therefore on the calculated power. Silicon solar modules mounted with a few centimeters over roof-tiles might be best described by open rack, although the calculated temperatures can be some degrees too low and therefore lead to an underestimation of your system peak power. (For example can 6 °C underestimated temperature mean that you underestimate the power by 2.4 %.) Please check the exported file “Temp_cell.txt” after pressing the “simulate” button.
- PV *temperature coefficient* (from your module data sheet, typical around -0.4 %/K for silicon solar cells)
- If *Use Incident Angle Modifier* is checked, the simulation is modified for angle dependent light reflection at the module glass surface.
- *Albedo* (ground reflection coefficient, typically around 0.25, only important for large tilt and bright grounds like snow or bright sand)
- *Inverter efficiency* (from your inverter data sheet, typical 95% .. 98 %, not used if an inverter efficiency curve is used instead)
- *Max inverter power*: Nominal inverter power. Used to cut the simulated power if the combined module power is larger. Also used as upper value for the inverter efficiency curve.
- *Time Zones:* Time zone for each measured data curve. Should be 1 for middle Europe, and 0 for Greenwich time or Coordinated Universal Time (UTC).
- *Daylight Saving:* Check this box, if the corresponding measured data curve switches to summer time (+1h during summer half of the year). Not to be used with Coordinated Universal Time (UTC).
- *Min time steps*: If these time intervals in seconds is larger than the time intervals in your file with the measured PV power, some of your data will be skipped (see page 13)

3.2 Getting started

1. Download the production data of your PV system for a single sunny day. The file should be readable by a simple editor like windows Notepad or WordPad. A typical file format is csv. It should contain columns with date, time, and produced power. Note if the produced power is given in W or kW.

2. Put all files from “PVCheck” into a single folder on your hard drive.
3. Start the program PVCheck.exe. This will start the python program automatically, which opens in a separate window. (When everything is loaded, the second window shows the text “PVLib simulator started.” This might take some seconds.)
4. Type in the correct time zone for your PV system
5. Type in the other basic information as described above.
6. Check “W” or “kW” under “PV datafile unit” corresponding to your datafile of step 1.
7. Type in the wind speed of that day (typically 1 – 6 m/s). This value can also be obtained from weather websites.
8. Click File > “Open own PV data”. Chose a file downloaded in step 1. The program now tries to identify header lines (which will be ignored) as well as columns containing date, times and values. If more than one column with values is found, a window will be opened asking you for the column containing the produced power values. Just click on the button with the correct column number. If no date could be identified, you will be asked for that.
9. Press the “Simulate” button.

Depending on your computer, the simulation might take some seconds. As a result, you should see two curves. The red one shows the simulated values and the blue one the values measured by your inverter. You can now change the PV peak power value to get the best match between the two curves. Press the “simulate” button every time after you changed the peak power value.

Since the air temperature on a sunny day typically changes a lot over the day, we recommend to use real temperature and wind speed data from a local weather station as described in the following.

The incident angle modifier (IAM) was added to PVcheck from Version 2.0. If checked, the simulation includes the effect that the irradiation reaching the solar cells is a bit reduced during times where the sun shines under a shallow angle on the modules. The calculation is done via the model from Martin and Ruiz [2]–[5]. Presently, PVcheck uses the default value 0.16 for the empirical angular losses coefficient in this model. Activating the IAM leads to a little decrease in the simulated irradiation and therefore to a little increase in the obtained system peak power.

3.3 Advanced simulation

3.3.1 Using temperature and wind speed data from weather stations

IMPORTANT: PVCheck transformes all time values to *coordinated universal time* (UTC) for all calculations and also displays the results using UTC. Choose the correct time zone for all data files on the “time formats” tab *before* loading the data.

Download temperature and windspeed data that belong to your nearest weather station. In Sweden, such data is for instance available from SMHI.

SMHI:

https://www.smhi.se/data/meteorologi/ladda-ner-meteorologiska-observationer#param=airtemperatureInstant,stations=all

Search your nearest station and then “Lufttemperatur (h)”. Make sure that this station provides the data for the time period you are interesting in. All we need is that the day of your choice lies within that period and that at least hour-resolution is provided. Download the according data file.

Do the same with the windspeed. In the case of the above link, the parameter is “Vindriktning och vindhastiget (h)”.

Import these data files into our program via *File > Open temperature data* and File > *Open wind speed data*. You will eventually be asked to choose the correct data column. Typical wind speeds are in the range of 1 .. 6, which should help you to identify the right column. The wind speed has to be given in m/s.

When loading PV data from different sunny days, the program will automatically chose the correct day from these data and display it in the upper right corner.

3.3.2 Using irradiation data from weather stations

a) Basic correction

This software cannot simulate day-to-day variations of aerosols or water droplets in the atmosphere which exist even on sunny days. Deviations from these variations can be minimized by using real irradiation data measured by weather stations. Download the measured global horizontal irradiation (GHI) from a nearby weather station. In Sweden this data is provided for instance from SMHI.

IMPORTANT: A high time resolution (like on minute basis) is needed for this to work satisfactory. If the data has too large time-intervals, typical each data point reflects an averaged value over the last interval. This might lead to a misfit to our simulation and thus deteriorate the whole procedure. A possible solution is described in part b of this chapter.

If sufficiently small time intervals are available in the downloaded data, import them via *File > Open measured GHI data*. It is assumed that the data units are W/m². The relevant time period of the data will be shown under the *Insolation* Tab in the upper right of the main window. After the first simulation, the display will show both, the simulated (in red) and the measured (in orange) GHI curve. When checking the box named “Correct with meas. GHI”, the deviation of both curves will be used to correct the simulated curve in the main graph in the lower half of the main window.

It is essential that the measured GHI curve is almost free from negative peaks like from clouds. Additionally, it is important that the weather station is not too far away from the place of your PV plant.

b) Correction including averaging time intervals

Often, the PV production data can be obtained with finer time intervals like (5 minutes) than the GHI data from a weather station. If the latter shows GHI values only for every full hour, it typically reflects the average GHI from the previous hour. That smears out the measured GHI curve and delays the peak position:

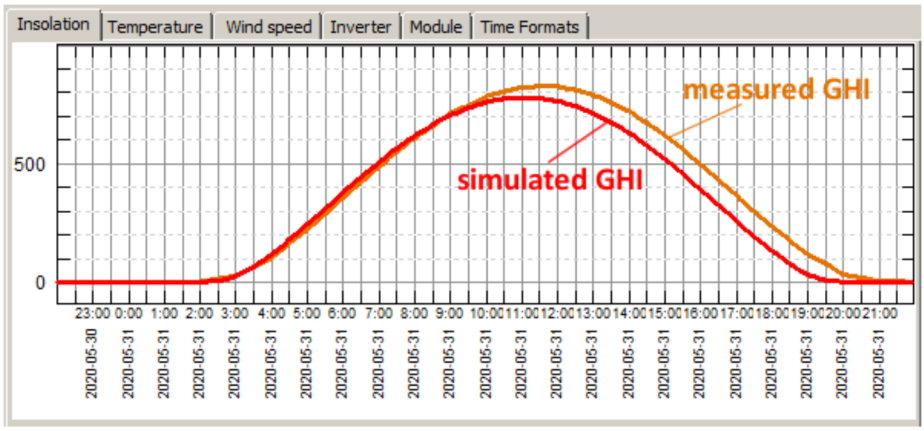


Fig. 1 Compared to the simulated GHI which has the same finer time interval than the PV production data, the measured GHI curve seems to be delayed against the simulated GHI curve.

A solution could be to smear out also the simulated GHI. This is done by activating “Average time intervals”. This function first calculates the simulated GHI values at the same times as the measured ones by averaging over the respective previous time intervals. It then linearly interpolates these new values within all time intervals.

After activating “Average time intervals” and re-doing the simulation, the delay is vanishing in our example:

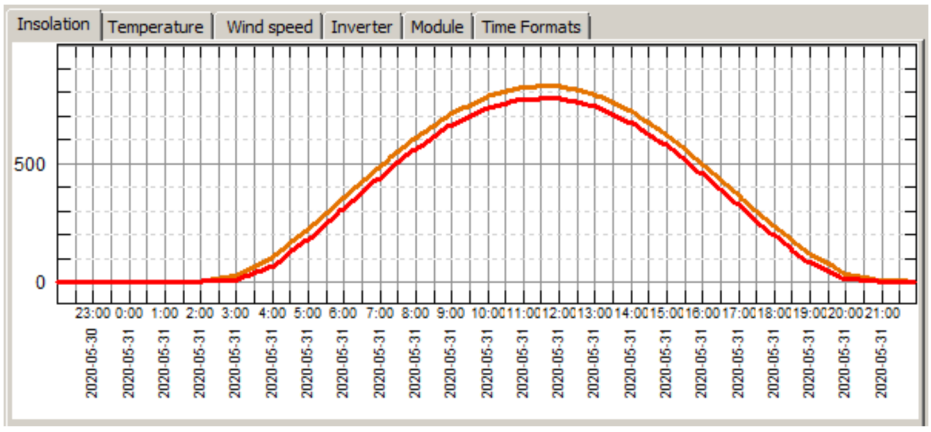


Fig. 2 Simulated GHI (red) and measured GHI (orange) after averaging time intervals

The averaging makes at least sure that an almost constant correction factor over a sunny day will be applied to the simulated PV power, although the maximum of these two curves are now shifted against the maximum of the measured PV power.

If faster GHI variations exist over the day, for example because of clouds, this procedure will probably not lead to satisfactory results.

Note: This averaging procedure is still experimental. It assumes “friendly” data sets (with monotonously increasing time values, etc.) so the results might be wrong under certain circumstances.

3.3.3 Using Inverter Efficiency Curves

The efficiency of your inverter typically depends on the power to be converted. The efficiency as a function of output power is found in the data sheet of your inverter.

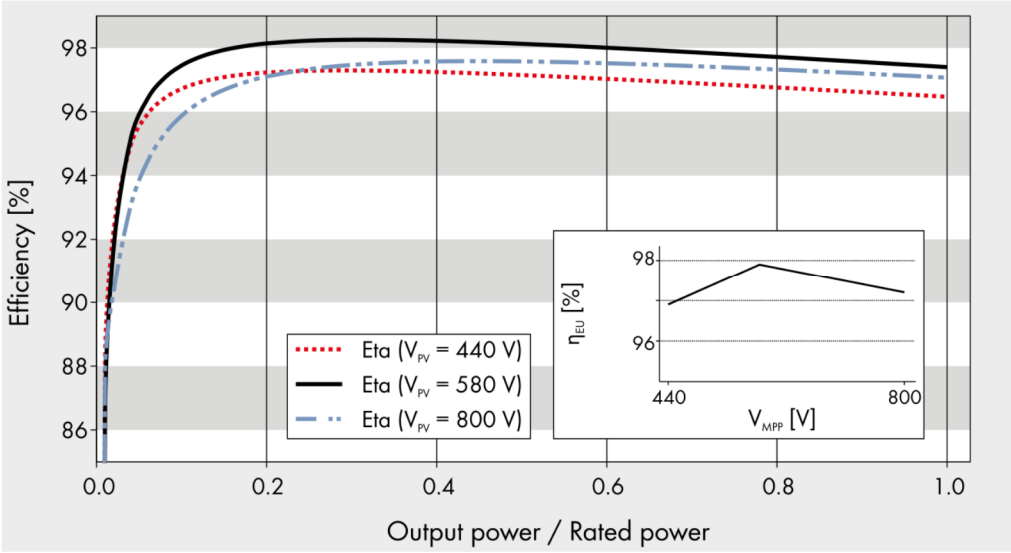


Fig. 3 Inverter efficiency curve for a SMA Sunny Tripower 12000TL *[6]*

As you can see from figure ***3***, stronger deviations from a constant efficiency occur only at very low power values. In the case of this figure, you have to choose a curve that lies nearest to your PV string voltage. To take this curve into account, a text file has to be made by the user reflecting some points on this curve. The text file has to look like this:

0.009	0.848
0.0172	0.923
0.0447	0.955
0.079	0.9655
0.154	0.973
0.307	0.9756
0.555	0.9732
0.830	0.970
1.005	0.967

The left column contains the power fraction (output power / rated output power) and the right column the efficiency (1 means 100%).

As a test, you can use the file “SMA_Inverter_lowlight.txt” provided with this program that reflects our example data here.

After loading the curve by pressing the button with the three dots and checking the according checkbox, you will see the curve on the upper right on the window under the “Inverter”-Tab. Check if it looks correct.

3.3.4 Using module efficiency curves

Similar to inverters, also PV modules have an irradiation dependent efficiency. This is handled in the same way as for inverters here. The module efficiency curve can be found on the data sheet of your PV module type.

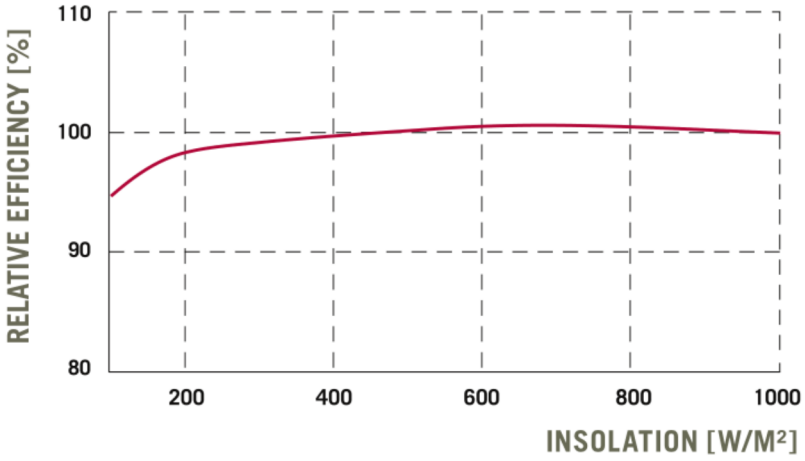


Fig. 4 Module efficiency curve from the datasheet of the Q-cells module Q.PLUS BFR-G4.1

To take this curve into account, a text file has to be made by the user reflecting some points on this curve. The text file has to look like this:

0	0.895
100.7	0.948
159.0	0.975
230.7	0.989
390.6	0.998
589.5	1.006
713.3	1.006
864.2	1.003
1020	1.000

The left column contains the insolation in W/m² and the right column the relative module efficiency (1 corresponds to the nominal efficiency and 0.5 means that only half of the nominal efficiency is reached at that insolation).

As a test, you can use the file “Q-Cells_Module_lowlight” provided with this program that reflects our example data here.

After loading the curve by pressing the button with the three dots and checking the according checkbox, you will see the curve on the upper right on the window under the “Module”-Tab. Check if it looks correct.

3.4 PV system profiles

If you wish to check different PV systems you might want to save all the fixed properties of these systems. This is provided by the functions *File > Load settings* and *Save settings*. Here you can save the properties of several special PV systems like longitude and latitude, power, etc., each in one file per PV system with extension .pv and load them afterwards.

3.5 Generated files

Each time you start a simulation by pressing the “simulate” button, a few text files will be generated in the same folder where PVcheck.exe is located. These text files will be overwritten every time.

The most important files are:

Filename	Meaning
Temp_cell.txt	The calculated cell temperatures based on the air temperature, wind speed, solar insolation, and the PV-type. (For example will “close-roof” lead to higher temperatures than “open-rack”.) If you have a chance to measure the temperatures near the cells in your modules, you can check if you have chosen the best version of PV-type.
Temp_cell_from_measured_GHI.txt	Experimental, not used yet
GHI.txt	Simulated GHI.
PV_Irradiation.txt	The irradiation in W/m² on your modules simulated.
PV_Irradiation_GHI_corrected.txt	The irradiation in W/m² on your modules simulated and afterwards corrected by the factor between the measured GHI of your weather station and the simulated GHI.
PV_Irradiation_from_measured_GHI.txt	Experimental, not used yet
Output.txt	The simulated power of your PV system before GHI correction.
Output_from_measured_GHI.txt	Experimental, not used yet
Output.log	Actual date and time written every time after the python program PVlib.exe has finished a calculation. Used as trigger for the GUI to start to read “Output.txt”.

4 Extras

In the main menu under *Extras*, some useful extra functions are available.

4.1 Open Module Temperature Data

With this you can bypass the simulation of the module temperature by loading a file with measured module temperatures.

4.2 Open Global Tilted Irradiation (GTI) Data

With this you can bypass the complete irradiance simulation by loading a file with measured irradiance in plane of the modules.

4.3 Calculate sun angle

This function uses a simpler algorithm than the Python simulation program. It gives you the approximate position of the sun at the sky in azimuth and height over the horizon for a given date and time with less precision.

4.4 Clean up data file

The program assumes that your data files start with some header followed by a larger number of lines all containing the same number of values (columns). It accepts all lines *after* the last line not containing these number of values. If the number of accepted lines seem to small compared with the total number of lines in your file, it is suspected that there are faulty lines in the data body and you will get a warning (see chapter 5.1).

In such cases you should run *Extras* > *Clean up data file*

This will try to remove faulty lines from your data body and save a new version of your data file under a new name.

4.5 Simplify one-day data file

If you wish to further analyze and present data files, this function helps you to bring it into a format which is more easy to handle by other programs. It removes the date and converts the time-string into a real value in decimal format containing the number of hours since the start of the day. If your data starts at 22h before the important day, the time will start with -2.0.

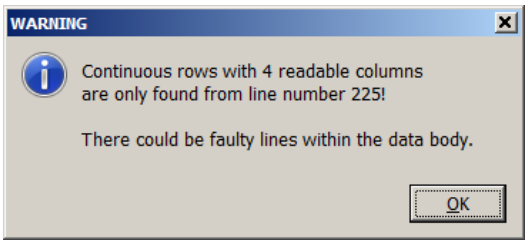
If your data file contains more value columns, all unnecessary columns except the one you choose are removed in the new file.

5 Error search

In case of unexpected behavior, please send an email to markus.rinio@kau.se and put “PVCheck” into the subject line.

5.1 Warning when opening a data file

When opening a data file, you might see an error message like this:



PVCheck analyses the data file line by line from the end. As soon as it finds a line with an unusual number of readable columns, it assumes this as the one before the begin of the data body. If such a line exists inside the data body, the lines before would normally be ignored.

Possible Solution: Run *Extras* > *Clean up data file* as described in chapter 4.4.

This deletes all lines with unusual number of readable columns and stores the result in a new file.

5.2 The command-line simulation window vanishes

This behavior was observed when not enough space was left on your system partition (typically C:). If this was not the reason, run CMD.exe on your computer, go to the folder of your “PVCheck” installation (using the cd command), and then type PVlib.exe to restart the PVLib simulation again. Any error message should now be visible even after it crashes.).

If not enough space is left on your system partition, clean up the temp-folder. (see chapter 2.1.1)

5.3 A time shift is visible between the measured and simulated curve

a) This could be due to wrong time zones or wrong clock settings of your inverter.

b) Another reason could be that the data files have much different time resolutions (e. g. minutes or just hour steps) and only one of the files contains averaged values from the previous time steps while the other contains current values of exactly that times.

Example: PVCheck calculates the simulated power curve for the exact times. If your PV system stores every hour a power value which is the average of the previous hour, this measured power curve will appear delayed (shifted to the right) against the red simulated curve.

Try to get data files with small time steps (high time resolution) in that case.

c) This error could occur if the longitude or azimuth value is wrong in the main window.

d) This error could occur if the tilt of the PV system is wrong in the main window.

5.4 Only a part of the curve deviates

a) This is a typical sign of shadowing. If for instance your PV system is shaded by trees between 8h and 10h in the morning, a lower power is measured during that time. This does not indicate a faulty PV system.

Another reason of course are clouds shading your system during the day. If this was too much, chose another day. Hint: Under special conditions the PV power of your system can be slightly larger than expected during time periods when thin clouds are near to the sun direction but not shading the sun.

When using the GHI correction with measured irradiation data from a weather station, check if this curve is also free from negative peaks.

b) It is possible that the incident angle modifier does not correctly describe the reflection of the special modules used, because a fixed empirical value for the angular loss coefficient is used here. Try to uncheck “Use incident angle modifier” in the main window.

c) A wrong value for the PV system tilt can also be a reason.

5.5 Memory problem

Make sure to clean up your TEMP folder. See chapter 2.1.1.

5.6 Calculations are too slow

PVcheck calculates a simulated power *for each time* of the measured PV power data. If such a file has for instance minute steps or even smaller, this is a huge amount of calculations.
Solution: Enter a larger number of seconds into “Min Time Steps”. A value of 300 should do it, but if this is still too slow, increase this number. If the number is too large, you will lose resolution in your curves.

6 License Terms

The Software PVCheck ©2020 by Markus Rinio, Karlstad University, is provided under the terms of this License.

Permission is hereby granted, free of charge, to any person obtaining a copy of this Software and associated documentation files (the "Software"), including without limitation the rights to use, copy, publish, distribute or sublicense copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

1. These license terms - containing the above copyright notice and this permission notice, the disclaimer and the forum clause, as well as the software manual - shall be included in all copies or substantial portions of the Software.

2. You may not redistribute the Software for profit, unless otherwise agreed in a separate written agreement.

3. You may not make changes in the Software or merge the Software with other Software unless otherwise agreed in a separate written agreement.

Because this Software is licensed free of charge, there is no warranty for the Software to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder and other parties provide the Software as is without warranty of any kind, either expressed or implied. The entire risk of the quality and the performance of the Software is with you.

In no event unless required by applicable law or agreed in writing will any copyright holder, or any other party who may redistribute the Software as permitted in this license, be liable to you for damages of any kind. These limits of warranty and liability are also valid if you have been granted the right to modify, merge or sell the Software in a separate agreement.

This license (and any non-contractual disputes/claims arising out of or in connection with it) are subject to the laws of Sweden and Swedish courts are the courts competent to try any disputes arising from this license.

The development of PVCheck was supported by the project "Solar Värmland" via Region Värmland, the Swedish Agency for Economic and Regional Growth, and Karlstad University under contract 20201237.

PVCheck is made possible thanks to the use of:
- PyInstaller (https://www.pyinstaller.org), for building the simulation part in binary format.
- PVlib (https://pvlib-python.readthedocs.io/en/stable/index.html)
- Free Pascal (https://www.freepascal.org/) and Lazarus (https://www.lazarus-ide.org/), for development of the graphical user interface.

As a part of this Software comes pvlib python Copyright (c) 2013-2020, Sandia National Laboratories and pvlib python Development Team which is licensed under the separate BSD 3-Clause License. The following applies for pvlib python:

7 Appendix

The PVLib python simulation uses high precision functions to calculate the sun position including light refraction in the atmosphere, special models to calculate the amount of irradiation on tilted PV modules including direct and diffuse light components as well as ground reflection and special diffuse light from around near the sun direction.

The atmospheric model includes a database for the turbidity of any place and time of the year on earth as well as altitude depending air pressure. This database is stored in the file “LinkeTurbidities.h5”.

The calculation of the modules AC power is done by the so-called “PV Watts” model to make it possible to simulate even unknown PV modules.

7.1 The Python packages installed

In Python, the list of installed packages and versions as obtained with the command “pip list” is

Package	Version
alabaster	0.7.12
altgraph	0.17
argh	0.26.2
astroid	2.4.2
atomicwrites	1.4.0
attrs	19.3.0
autopep8	1.5.4
Babel	2.8.0
backcall	0.2.0
bcrypt	3.1.7
bleach	3.1.5
brotlipy	0.7.0
certifi	2020.6.20
cffi	1.14.0
chardet	3.0.4
cloudpickle	1.5.0
colorama	0.4.3
cryptography	2.9.2
decorator	4.4.2
defusedxml	0.6.0
diff-match-patch	20200713
docutils	0.16
entrypoints	0.3
ephem	3.7.7.0
flake8	3.8.3
future	0.18.2
idna	2.10
imagesize	1.2.0
importlib-metadata	1.7.0
intervaltree	3.0.2
ipykernel	5.3.4
ipython	7.16.1
ipython-genutils	0.2.0
isort	4.3.21
jedi	0.17.1
Jinja2	2.11.2
jsonschema	3.2.0
jupyter-client	6.1.6
jupyter-core	4.6.3
keyring	21.2.1
lazy-object-proxy	1.4.3
llvmlite	0.33.0+1.g022ab0f
macholib	1.11
MarkupSafe	1.1.1
mccabe	0.6.1
mistune	0.8.4
mkl-fft	1.1.0
mkl-random	1.1.1
mkl-service	2.3.0
mock	4.0.2
nbconvert	5.6.1
nbformat	5.0.7
numba	0.50.1
numexpr	2.7.1
numpy	1.19.1
numpydoc	1.1.0
packaging	20.4
pandas	1.1.0
pandocfilters	1.4.2
paramiko	2.7.1
parso	0.7.0
pathtools	0.1.2
pefile	2019.4.18
pexpect	4.8.0
pickleshare	0.7.5
pip	20.2.1
pluggy	0.13.1
prompt-toolkit	3.0.5
psutil	5.7.0
pvlib	0.7.2
pycodestyle	2.6.0
pycparser	2.20
pycryptodome	3.8.2
pydocstyle	5.0.2
pyflakes	2.2.0
Pygments	2.6.1
PyInstaller	3.6
pylint	2.5.3
PyNaCl	1.4.0
pyOpenSSL	19.1.0
pyparsing	2.4.7
pyrsistent	0.16.0
PySocks	1.7.1
python-dateutil	2.8.1
python-jsonrpc-server	0.3.4
python-language-server	0.34.1
pytz	2020.1
pywin32	227
pywin32-ctypes	0.2.0
PyYAML	5.3.1
pyzmq	19.0.1
QDarkStyle	2.8.1
QtAwesome	0.7.2
qtconsole	4.7.5
QtPy	1.9.0
requests	2.24.0
rope	0.17.0
Rtree	0.9.4
scipy	1.5.0
setuptools	49.2.1.post20200807
sip	4.19.13
six	1.15.0
snowballstemmer	2.0.0
sortedcontainers	2.2.2
Sphinx	3.2.0
sphinxcontrib-applehelp	1.0.2
sphinxcontrib-devhelp	1.0.2
sphinxcontrib-htmlhelp	1.0.3
sphinxcontrib-jsmath	1.0.1
sphinxcontrib-qthelp	1.0.3
sphinxcontrib-serializinghtml	1.1.4
spyder	4.1.4
spyder-kernels	1.9.2
tables	3.6.1
testpath	0.4.4
toml	0.10.1
tornado	6.0.4
traitlets	4.3.3
ujson	1.35
urllib3	1.25.9
watchdog	0.10.3
wcwidth	0.2.5
webencodings	0.5.1
wheel	0.34.2
win-inet-pton	1.1.0
wincertstore	0.2
wrapt	1.11.2
yapf	0.30.0
zipp	3.1.0

7.2 The Python source code

```
# Definitions of abbreviations available on
# https://pvlib-python.readthedocs.io/en/stable/variables_style_rules.html?highlight=definition%20#variables-and-symbols

import pandas as pd
import pvlib.pvsystem
import pvlib.atmosphere
import pvlib.solarposition
import pvlib.irradiance
import pvlib.clearsky
import os
import sys

from time import gmtime, strftime


import datetime

try:
    path = os.getcwd()+'\\'

    # Indicate to graphical user interface program that python program is loaded and running
    log=open(path+'Output.txt','w')
    if log.mode == 'w':
        log.write('PVlib running in '+path)
        log.close()
    else:
        print('PVlib simulator can not write into file '+path+'Output.txt')
        sys.exit()

    # The file 'Input.log' must be generated or updated after this program started
    # to trigger a PV power calculation

    print('PVlib simulator started.')


    if os.path.exists(path+'Input.log'):
        oldmtime = os.path.getmtime(path+'Input.log')
    else:   # wait here until file exists
        while not os.path.exists(path+'Input.log'):
            oldmtime = 0

    while True:
        mtime = os.path.getmtime(path+"Input.log")

        if (mtime != oldmtime):

            oldmtime = mtime
```

```
f=open(path+"Input.txt","r")
print(strftime("%Y-%m-%d %H:%M:%S", gmtime())+"  Loading input file ..", file=sys.stdout)
if f.mode == 'r':   #check if the file opened for read successfully
    parameter=f.readlines()
    f.close()

    task = int(parameter[0])
    if task == 0:
        sys.exit('PVlib simulator stopped.')
    longitude = float(parameter[1])
    latitude = float(parameter[2])
    altitude = float(parameter[3])
    PVAzimuth = float(parameter[4])
    PVtilt = float(parameter[5])
    PVPeakPower = float(parameter[6])

    #Temperature-Model
    PVTypeA = float(parameter[7])
    PVTypeB = float(parameter[8])
    PVTypeDeltaT = float(parameter[9])

    PVTemperatureCoefficient = float(parameter[10])/100
    PValbedo = float(parameter[11])
    UseIAM = int(parameter[12])

    UseModuleTemperature = int(parameter[13])
    UseGTI = int(parameter[14])
    nTimes = int(parameter[15])
    nParams = int(parameter[16])
    Startline = int(parameter[17])

    times = pd.Series(dtype='datetime64[ns]')
    temp_air = pd.Series(dtype='float64')
    wind_speed = pd.Series(dtype='float64')
    GHI_measured = pd.Series(dtype='float64')
    ModuleTemperature = pd.Series(dtype='float64')
    GlobalTiltedIrradiance = pd.Series(dtype='float64')
    for i in range(0,nTimes):
        dt=(datetime.datetime.strptime(parameter[Startline+nParams*i],'%Y-%m-%d %H:%M:%S '))
        times.at[i]=dt
        temp_air.at[i]=float(parameter[Startline+nParams*i+1])
        wind_speed.at[i]=float(parameter[Startline+nParams*i+2])
        GHI_measured.at[i]=float(parameter[Startline+nParams*i+3])
        ModuleTemperature.at[i]=float(parameter[Startline+nParams*i+4])
        GlobalTiltedIrradiance.at[i]=float(parameter[Startline+nParams*i+5])

    print(strftime("%Y-%m-%d %H:%M:%S", gmtime())+"  Calculating PV power curve", file=sys.stdout)

    times = pd.DatetimeIndex(times)
```

```
temp_air = pd.DataFrame(temp_air)
temp_air.index = times
temp_air.columns = ['T']

wind_speed = pd.DataFrame(wind_speed)
wind_speed.index = times
wind_speed.columns = ['W']

GHI_measured = pd.DataFrame(GHI_measured)
GHI_measured.index = times
GHI_measured.columns = ['ghi']

ModuleTemperature = pd.DataFrame(ModuleTemperature)
ModuleTemperature.index = times
ModuleTemperature.columns = ['T']

GlobalTiltedIrradiance = pd.DataFrame(GlobalTiltedIrradiance)
GlobalTiltedIrradiance.index = times
GlobalTiltedIrradiance.columns = ['GTI']

system = pvlib.pvsystem.PVSystem(surface_tilt=PVtilt, surface_azimuth=PVAzimuth, albedo=PValbedo)

pressure = pvlib.atmosphere.alt2pres(altitude)

solpos = pvlib.solarposition.get_solarposition(times, latitude, longitude)
airmass = pvlib.atmosphere.get_relative_airmass(solpos['apparent_zenith'])
am_abs = pvlib.atmosphere.get_absolute_airmass(airmass, pressure)

dni_extra = pvlib.irradiance.get_extra_radiation(times)
#cwd = os.getcwd()
Turbidities = pvlib.clearsky.lookup_linke_turbidity(times, latitude, longitude,path+'LinkeTurbidities.h5', interp_turbidity=True)

AngleOfIncidence = pvlib.irradiance.aoi(PVtilt,PVAzimuth,solpos['apparent_zenith'],solpos['azimuth'])
IncidentAngleModifier = pvlib.iam.martin_ruiz(AngleOfIncidence,0.16)
IncidentAngleModifierDiffuse = pvlib.iam.martin_ruiz_diffuse(PVtilt,0.16,0.4244,None)
# Get ClearSky irradiation components for all the relevant times:

# DNI: direct normal irradiance
# GHI: Global horizontal irradiance
# DHI: diffuse horizontal irradiance
ClearSky = pvlib.clearsky.ineichen(solpos['apparent_zenith'], am_abs, Turbidities,
                           dni_extra=dni_extra, altitude=altitude)

# Calculate DNI and DHI from the measured GHI
Irrad_measured = pvlib.irradiance.erbs(GHI_measured['ghi'], solpos['zenith'], times)

# Calculate irradiation on tilted plane from simulated clearsky irradiation
pv_irrad = system.get_irradiance(solpos['apparent_zenith'],
                                 solpos['azimuth'],
                                 ClearSky['dni'], ClearSky['ghi'], ClearSky['dhi'],
```

```
                                        dni_extra,
                                        airmass,'perez'
                                        )
                if UseIAM:
                    pv_irrad2 = pv_irrad.copy()
                    pv_irrad2['poa_direct'] = pv_irrad['poa_direct']*IncidentAngleModifier
                    pv_irrad2['poa_sky_diffuse'] = pv_irrad['poa_sky_diffuse']*IncidentAngleModifierDiffuse[0]
                    pv_irrad2['poa_ground_diffuse'] = pv_irrad['poa_ground_diffuse']*IncidentAngleModifierDiffuse[1]
                    pv_irrad2['poa_diffuse'] = pv_irrad2['poa_sky_diffuse']+pv_irrad2['poa_ground_diffuse']
                    pv_irrad2['poa_global'] = pv_irrad2['poa_direct']+pv_irrad2['poa_diffuse']
                    pv_irrad = pv_irrad2.copy()


                # Calculate irradiation on tilted plane from measured GHI, experimental, presently not used
                pv_irrad_from_measured_GHI = system.get_irradiance(solpos['apparent_zenith'],
                                        solpos['azimuth'],
                                        Irrad_measured['dni'], GHI_measured['ghi'], Irrad_measured['dhi'],
                                        dni_extra,
                                        airmass,'perez'
                                        )
                if UseGTI:
                    pv_irrad['poa_global'] = GlobalTiltedIrradiance.squeeze()
                    pv_irrad_from_measured_GHI['poa_global'] = GlobalTiltedIrradiance.squeeze()

                # haydavies
                # for the following parameters, see https://pvlib-python.readthedocs.io/en/stable/generated/pvlib.temperature.sapm_cell.html
                if UseModuleTemperature:
                    temps = ModuleTemperature.squeeze()
                    temps_from_measured_GHI = ModuleTemperature.squeeze()
                if not UseModuleTemperature:
                    temps = pvlib.temperature.sapm_cell(pv_irrad['poa_global'], temp_air['T'], wind_speed['W'], PVTypeA, PVTypeB, PVTypeDeltaT)
                    temps_from_measured_GHI = pvlib.temperature.sapm_cell(pv_irrad_from_measured_GHI['poa_global'], temp_air['T'],
wind_speed['W'], PVTypeA, PVTypeB, PVTypeDeltaT)

                power = pvlib.pvsystem.pvwatts_dc(pv_irrad['poa_global'],temps,PVPeakPower,PVTemperatureCoefficient,25)
                power_from_measured_GHI =
pvlib.pvsystem.pvwatts_dc(pv_irrad_from_measured_GHI['poa_global'],temps_from_measured_GHI,PVPeakPower,PVTemperatureCoefficient,25)

                power.to_csv(path+'Output.txt',header=False)
                power_from_measured_GHI.to_csv(path+'Output_from_measured_GHI.txt',header=False)

                pv_irrad['poa_global'].to_csv(path+'PV_Irradiation.txt',header=False)
                pv_irrad_from_measured_GHI['poa_global'].to_csv(path+'PV_Irradiation_from_measured_GHI.txt',header=False)

                temps.to_csv(path+'Temp_cell.txt',header=False)
                temps.to_csv(path+'Temp_cell_from_measured_GHI',header=False)
                ClearSky['ghi'].to_csv(path+'GHI.txt',header=False)

                log=open(path+'Output.Log','a')
                log.write(strftime("%Y-%m-%d %H:%M:%S", gmtime()))
```

```
                log.close()


            else:
                print('Reading file "Input.txt" failed.', file=sys.stdout)

except Exception as e:
    #wait here before closing the commandline window
    print('\n\n\Error!\n\n'+str(e))
    key = input()
```

8 Changelog

Version 0.35

- An unnecessary error message was removed in function “Clean Up Data File”

- A preference “Min Time Steps” was added to speed up calculation time with files that have too small time steps

- Curves are now automatically re-loaded when Time-Zones or Daylight Saving settings are changed.

Version 0.32

- Temperature in the main form is now correctly named “Air temperature” to avoid mix-up with “module temperature”.

Version 0.31

- An error in the function “clean up data file” not removing lines containing the word NaN was removed.

Version 0.3

- Added function to use data from a temperature sensor attached to the PV modules (the temperature model to calculate module temperatures from air temperature and wind speed is not used in this case)

- Added function to use data from an irradiation sensor attached to the PV modules (the clearsky model as well GHI data are not used in this case)

- PVcheck now is able to load data files where the lines are not ascending in time and offers to sort the lines

- Fixed a little error where time zone changes directly before loading data are not recognized.

- Loading a settings file (*.pv) now automatically loads inverter efficiency curve and module efficiency curve, if exist.

Version 0.22

- If a file contains no date information, such date is tried to find *first* in the filename, and if not found there, it is *then* tried to find in the file path. Such date is then used as a proposal to the user when presenting the calendar.

Version 0.21

- Corrected program name in headline

- Uncheck box “Average GHI time intervals”, if GHI time intervals already are smaller than in PV data.

Version 0.2

- Added experimental calculation of irradiation from measured GHI, not used yet

- Removed a time issue in the function Extras > Simplify one-day data file

- Added incident angle modifier

- Fixed issue when clicking “Use inverter efficiency curve” and having no path defined below

9 References

[1] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, “pvlib python: A python package for modeling solar energy systems,” *Journal of Open Source Software*, vol. 3, no. 29, p. 884, 2018.

[2] N. Martin Chivelet and J. M. Ruiz, “Calculation of the PV modules angular losses under field conditions by means of an analytical model (vol 70, pg 25, 2001),” *Solar Energy Materials and Solar Cells - SOLAR ENERG MATER SOLAR CELLS*, vol. 70, pp. 25–38, Dec. 2001, doi: 10.1016/S0927-0248(00)00408-6.

[3] “Corrigendum to ‘Calculation of the PV modules angular losses under field conditions by means of an analytical model’ [Sol. Energy Mater. Sol. Cells 70 (1) (2001) 25–38] | Martin, N.; Ruiz, J.M. | download.” https://ur.booksc.eu/book/24106631/609401 (accessed Oct. 04, 2021).

[4] “pvlib.iam.martin_ruiz — pvlib python 0.9.0+0.g518cc35.dirty documentation.” https://pvlib-python.readthedocs.io/en/stable/generated/pvlib.iam.martin_ruiz.html (accessed Oct. 05, 2021).

[5] “pvlib.iam.martin_ruiz_diffuse — pvlib python 0.9.0+0.g518cc35.dirty documentation.” https://pvlib-python.readthedocs.io/en/stable/generated/pvlib.iam.martin_ruiz_diffuse.html (accessed Oct. 05, 2021).

[6] S. S. T. AG, “Downloads.” https://www.sma.de/en/service/downloads.html (accessed Oct. 05, 2021).