Estimating installation parameters of photovoltaic systems in northern latitudes by mathematical model fitting

Timo Salola

MSc Thesis

November 2023

Department of Physics and Mathematics

University of Eastern Finland

Preface

As someone whose interests and strenghts lie in geometry and programming, the real world problem of panel installation parameter estimation was appealing from the beginning. I would like to thank William Wandji and Juha Karhu for their insights on the effects of clouds, snows and temperature on solar PV installation data. And Luna for being the softest and purriest of cats.

This research was funded by the Academy of Finland, decision 350695.

Helsinki, the 15th of August 2023 *Timo Salola*

Contents

Chapter I

Introduction

This thesis examines a specific applied mathematics problem suggested by the Finnish Meteorological Institute(FMI). The goal is to solve the geographic location and panel installation angles of photovoltaic solar power installations using only the power output data. The chosen method breaks the problem of solving the geographic location and panel angles into separate algorithms. In practice, this means that the algorithms used can be less complex as they do not have to solve every unknown variable simultaneously and visualizations of individual algorithms should also be more straightforward. Using multiple algorithms also splits the parameter space into smaller spaces, thus improving the performance of fitting algorithms. And the final benefit is the ability to focus on solving only the unknown parameters. For example, if the geolocation of a system is known and the panel installation angles are not, instead of estimating the geolocation, the known geolocation can be used for panel angle estimation, resulting in higher prediction accuracy.

The applications of parameter estimation algorithms would be in improving the quality of metadata in solar PV datasets. This could have implications for solar PV research but the existance of such algrithms poses privacy and security related questions as well. Whether these research benefits and concerns are realized, depends on the accuracy, and to some extent the ease of use of the algorithms.

A similar study was done by N. Haghdadi et al. in 2017 [1]. The 2017 article contains results from five case studies where the standard deviation of longitude prediction errors is less than 1.5°, reaching as low as 0.08° with case study 1-2. The standard deviation of latidude predictions is higher at less than 3.5° with the best result in case study 2-2 with standard deviation of 1.65°. Azimuth and tilt

predictions are reported as two separate angle error values with azimuth prediction errors reaching values between 4° and 27° and tilt 1.3° to 11.5°. These forementioned results are not directly comparable to the results shown in this thesis due to different datasets, geographic location and local climate, but they provide some perspective.

Another article of relevance written by M.K. Williams et. al. in 2012 [2] proposes multiple methods for determining the locations and orientations of solar PV installations. Perhaps the most interesting contribution of the 2012 article is the network approach to determining geographic location. This method relies on a grid of installations with known and accurate geolocation and installation angles. According to the authors, this networked approach works up to a 10-mile accuracy when the grid of known installations is dense around the estimated installation. This could make the network approach preferable for electric companies or institutions with large amounts of data. But as of now, it does not seem usable for FMI.

Chapter II

Installations and datasets

Figure 2.1: FMI Kumpula solar power installation string.

The two datasets used in this thesis were provided by FMI. They contain power generation measurements from installations in Kuopio and Helsinki, with the physical parameters listed in table 2.3. Due to the high elevation of the installations, shading is unlikely to be a major factor in either of the datasets. The same data was previously used in Herman Böök's *Photovoltaic output modeling* [3] and thus installation parameters and datasets have previously been verified.

Data in the two datasets follows a similar structure as shown in table 2.2. This snapshot from the Helsinki dataset shows that the temporal resolution is one measurement per minute and that there are multiple power values for each minute. The

fields *String 1* and *String 2* represent power from two identical sets or strings of solar PV panels installed at the same location and their sum should be near equal to inverter input. One of these strings is shown in figure 2.1. Datasets from other sources may differ from the FMI datasets in several ways. Power measurements may be taken at different intervals, once per 1, 5, 15, or 60 minutes and they are unlikely to contain more than one power value. Due to these reasons, the algorithms in this thesis are designed to operate on datasets with one measurement per minute and they use only the inverter output value as that is the most likely power value included in solar PV datasets.

Timestamp[UTC]	Inverter out	Inverter in	String 1	String 2
$2015-08-26\ 03:34$	NaN	NaN	0.5	NaN
$2015-08-26\ 03:36$	11.1	7.5	2.6	4.9
$2015-08-26\ 03:37$	25.4	26.1	9.8	16.3
$2015-08-26\ 03:38$	30.7	NaN	NaN	0.4
$2015-08-26\ 03:39$	46.4	44.8	20	24.8
$2015-08-26\ 03:40$	3.3	NaN	NaN	0.4
$2015-08-26\ 03:41$	29.3	18	9.1	8.9
$2015-08-26\ 03:42$	33.1	27.4	10.6	16.9
⋮	⋮	⋮	⋮	⋮
$2015-08-26\ 12:42$	12374.8	14619.1	7152	7467.1
$2015-08-26\ 12:43$	15442.2	15482.1	7708.9	7773.2
$2015-08-26\ 12:44$	14085.8	12898.7	6387	6511.8
⋮	⋮	⋮	⋮	⋮

Table 2.2: A section from FMI's Kumpula solar site PV production data, only the timestamp and inverter output values are used by the algorithms in this thesis. All power measurements are in watts.

	Helsinki	Kuopio
Latitude	60.204°	62.892°
Longitude	24.961°	27.634°
Nominal capacity	21 kW	20.28 kW
Panel tilt	15°	15°
Panel azimuth	135°	217°
Elevation	17m	10m

Table 2.3: Parameters for the FMI's Kumpula(Helsinki) and Kuopio PV installations as listed in Böök 2020 [3].

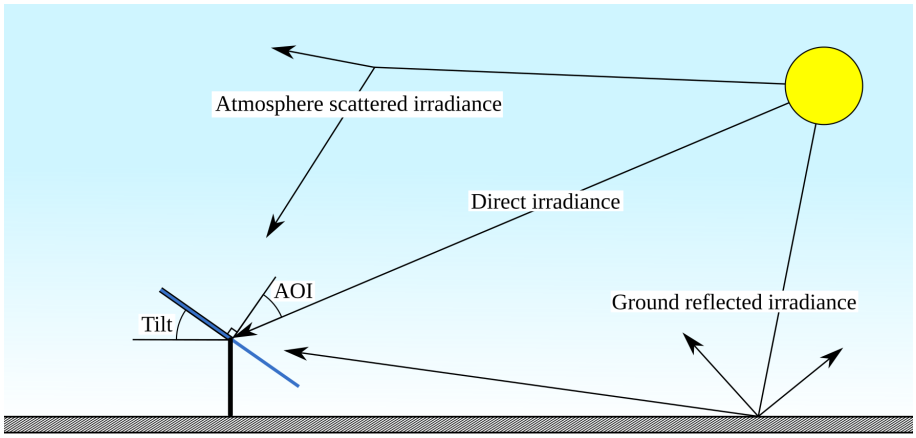


Figure 2.4: Simplified installation diagram for a PV system in ideal conditions.

Figure 2.4 shows a two dimensional simplification of a solar PV installation. The before unmentioned abbreviation AOI stands for angle of incidence and it is defined as the angle between direct irradiance and solar panel normal. AOI can be calculated with computer programs and software libraries when geolocation, panel tilt and azimuth and the time are known. The azimuth angle is not marked on the figure and azimuth represents the second panel normal component. Azimuth for geographic north is 0 degrees and azimuth is measured in clockwise degrees from north. The angles given for Helsinki(135°) and Kuopio(217°) installations tell us that the panels are facing southeast and southwest respectively.

2.1 Visualizing the data

The figure 2.5 contains a 3D point cloud generated by plotting one year of data from FMI Helsinki dataset and it shows that there are patterns in the data. The clearest pattern is formed by the first and last non-zero power minutes and this pattern can be used for geolocation estimation. Similarly if one day slices from the dataset are taken as shown in 2.6, the power generation plot shape can be used for panel angle estimation. However the difference between clear and cloudy or otherwise noisy days is significant and automating the process of cloud free day selection would be beneficial.

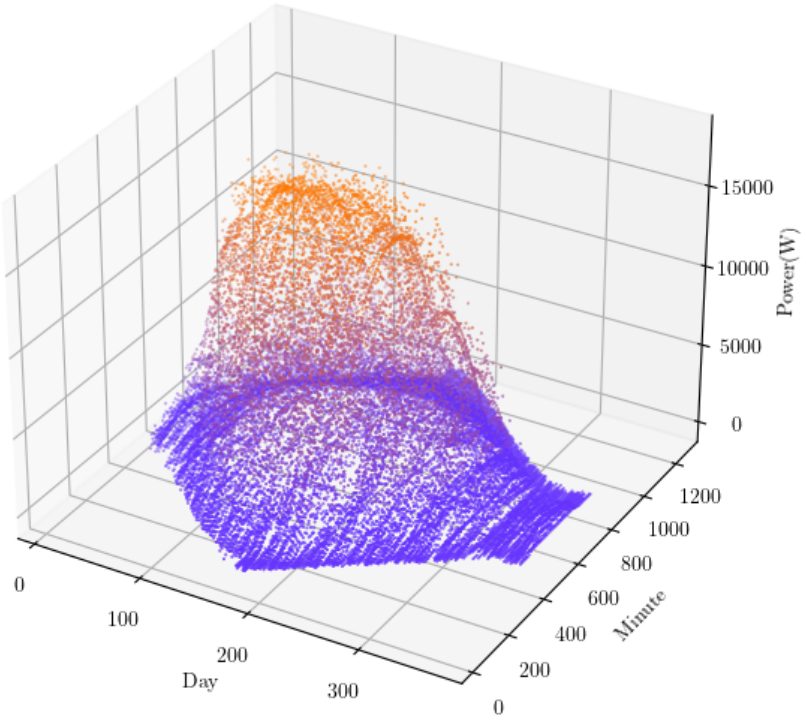


Figure 2.5: One year of data from FMI Kumpula installation as a 3D point cloud.

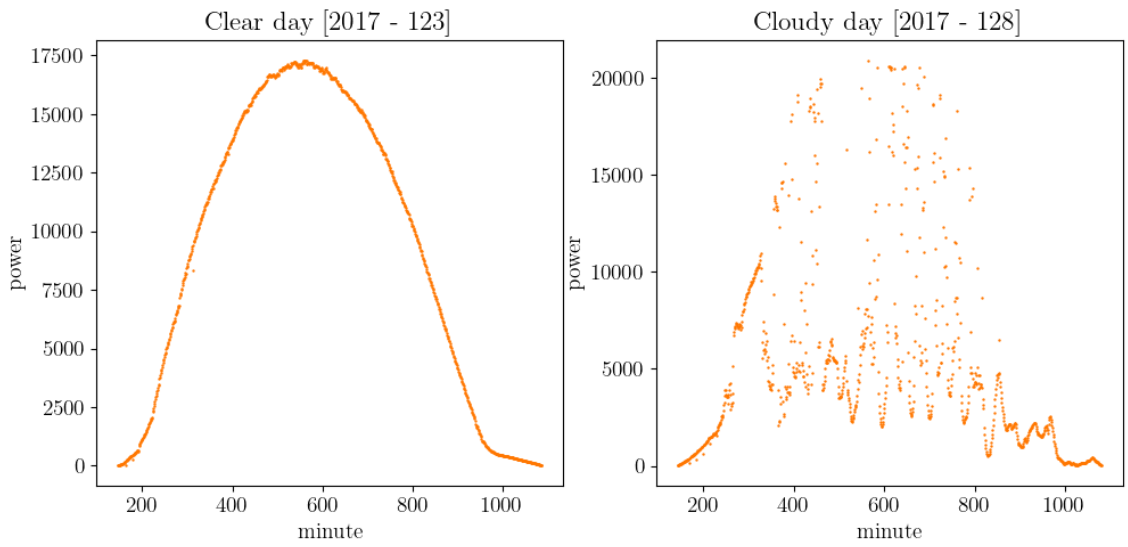


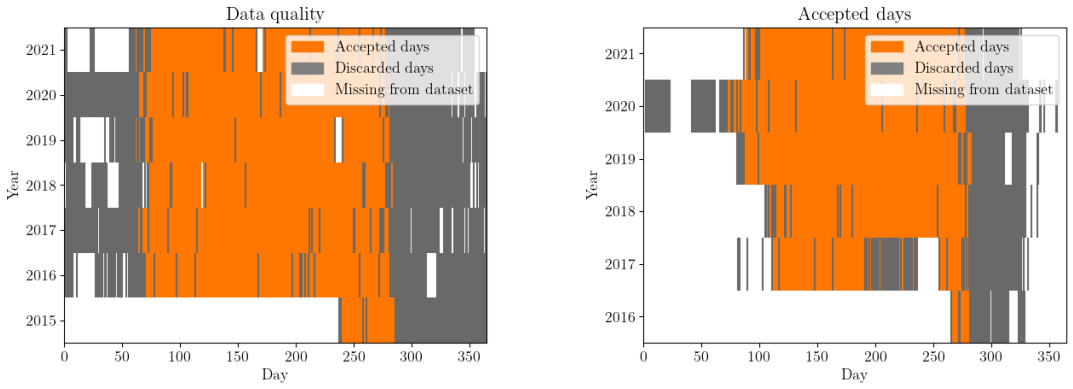
Figure 2.6: Two days from FMI Kumpula dataset with different charasteristics.

The presumably cloud free day 2017-123 has some notable charasteristics in addition to the geolocation derived first and last minutes. The shape resembles a skewed normal distribution and the knee section near 950 minutes may signify a moment in time during which the angle of incidence reaches 90 degrees and thus power generation from direct irradiance drops to zero. Note that this transition appears to be smooth and this may be a result of high reflective losses at high AOI. Similarly, intuition would suggest that the peak power minute occurs when the angle of incidence is at its minimum. Measurable traits such as these could have uses for parameter estimation.

2.2 Data pre-processing

The data pre-processing required by the algorithms in this thesis can be split into two gategories, classification and repairing. Classifying preprocessing is used to determine if a certain section of data is useful of analysis or not, the primary example here is the cloud free day detection algorithm which is discussed more throroughly in the next chapters. The second type of preprocessing, reparing preprocessing refers to the use of algorithms which fill in gaps or otherwise attempt to repair data which is unusable as is, but which could be used after repairing.

The data preprocessing algorithms used in this thesis load the data from csv files and examine whether individual days in the dataset meet set qualification requirements. These are the minimum and maximum measurement count, whether first and last measurements are taken too close to minute 0 or 1440 and the the percentage of measurements included between the first and last measurement. Figure 2.7 shows which days met the set quality requirements.



(a) Days in Helsinki dataset which met data quality thresholds.

(b) Days in Kuopio dataset which met data quality thresholds.

Figure 2.7: Visualizations of data quality. Measurement count between 400 and 1200, first minute is 5th or later, last minute is 1435 or earlier. More than 95% of measurements between first and last minute must be included.

After acceptable days are chosen with the classification algorithm, the next step is data repairing. Missing measurements and Nan values can be linearly interpolated, meaning that if a power measurement or a set of measurements is missing between two datapoints, the missing datapoints are estimated to describe a linear transition

breaching the gap between the known datapoints. When noise level is low, linearly approximating the missing values is unlikely to result in signficant errors. After this is done, the resulting data is ready for analysis.

2.3 Clear day detection algorithm

While the previous preprocessing steps have filtered and repaired days according to measurement counts and data gaps, these algorithms did not take the quality of measurements into account. If the interference in measurements caused by clouds or other sources is significant, the value of a day for model fitting is reduced. An example of strong interference can be seen in figure 2.6. Detecting the presence of such interference with an algorithm would help with automating the process of model fitting as that would eliminate the need to manually select good days from datasets. The following steps describe the process used for cloud free day detection in this thesis.

1. Dataset is split into days based on utc timestamps.
2. A copy of the power measurements for each individual day is taken and fed to a low pass filter algorithm.
3. A delta value is calculated based on the difference between original power measurements and low pass filtered measurements.
4. Days are discarded if their delta values fail to meet a set smoothness threshold.

The mathematically non-trivial parts here are the threshold selection, difference measurement and low pass filtering. Low pass filtering is a term borrowed from the field of signal processing and it refers to any algorithm which removes frequencies higher than a given limit from a signal, allowing lower frequencies to pass.

Here the filtering is done with the help of discrete Fourier transformations(DFT) and inverse discrete Fourier transformations(IDFT). When a list of numbers is given to DFT, the output is given as a list of ordered complex numbers, each of which represents a sine wave of a certain frequency, phase and amplitude. These wave equations form a continuous approximation of the input values and by sampling the continuous representation, the continuous trigonometric approximation can be

transformed back into discrete values. However if the complex numbers are adjusted before the IDFT operation, frequencies can be selectively modified. This means that DFT and IDFT can be used for frequency specific modification of given inputs, low pass filtering being one of the possibilities. Here low pass filtering was accomplished by zeroing out complex numbers which do not correspond to the 6 longest frequencies, the resulting smoothening can be seen in figure 2.8.

While this process is somewhat complicated, Fourier transformations are not the only tool for creating low pass filters. Similar results can also be achieved by locally averaging each power value to be the average of nearest k values. Discrete Fourier transformation based methods do however have an advantage in their universality. If the 6 or 7 or n longest frequencies can be determined to be a good low pass filter, then these same frequencies should result in similar outputs no matter the temporal resolution of the power measurement data. Where as a method based on local averages would require a different window size depending on measurement intervals.

The second component is not as complicated as the low pass filtering operation. Measuring the delta between a filtered and unfiltered set of measurements can be done by computing the discrete curve length or as was done here, measuring the absolute average deviation between filtered and unfiltered power measurement. This is shown with the following equations 2.1-2.5.

$$Power = [p_0, p_1, p_2, \ldots, p_n] \tag{2.1}$$

$$Power_{filtered} = [f_0, f_1, f_2, \ldots, f_n] \tag{2.2}$$

$$Power_{delta} = [|p_0 - f_0|, |p_1 - f_1|, |p_2 - f_2|, \ldots, |p_n - f_n|] \tag{2.3}$$

$$delta_{avg} = avg(Power_{delta}) \tag{2.4}$$

$$delta_{norm} = delta_{avg}/max(Power) \tag{2.5}$$

The last component is threshold selection. The value $delta_{avg}$ describes the average wattage difference between measured and low pass filtered measured power values. By definition, this delta value is dependent on noise and installation size, limiting its usability. A noise only -delta value can be calculated by normalizing the delta with the $max(Power)$. The resulting $delta_{norm}$ should now be comparable between installations of different sizes. Choosing to reject every day for which

$delta_{norm}$ value is higher than 0.05 would eliminate days with higher than 5% normalized noise.

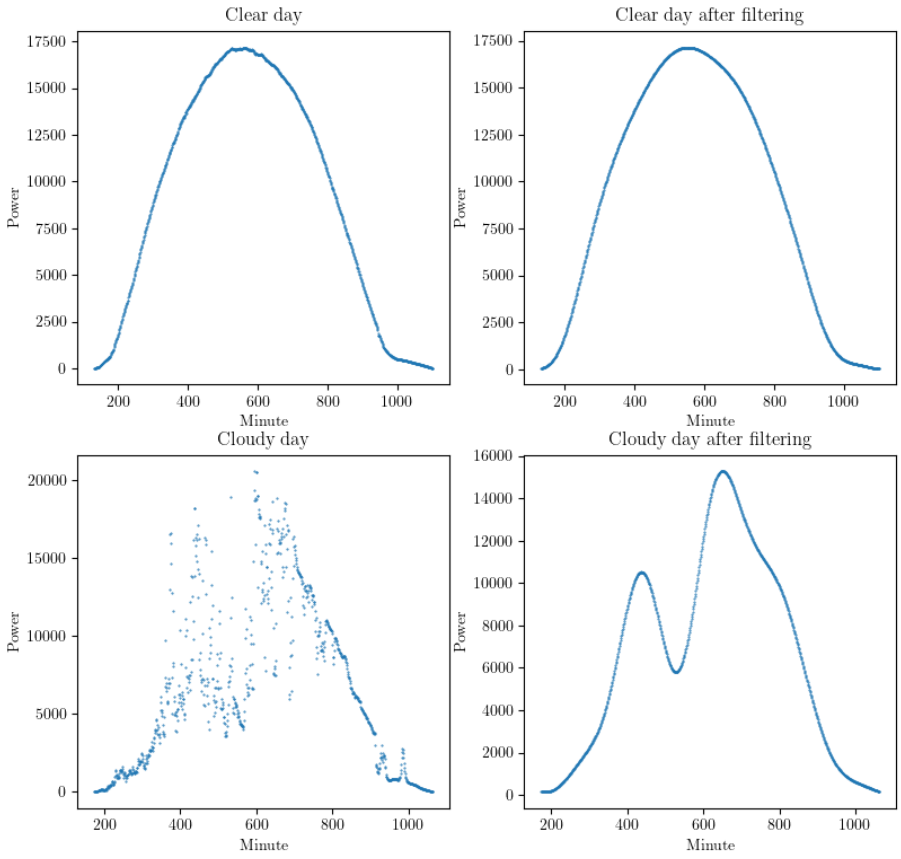


Figure 2.8: Cloud free day finder low pass filtering phase.

Chapter III

Solar irradiance simulation tools

The primary tool used in this thesis for simulating PV generation with different parameters is the python library PVlib. PVlib contains be built functions for estimating solar irradiance, angle of incidence and a multitude of other useful tools. As seen from table 3.1, the plane of array(POA) simulations which estimate the irradiance per $1m^2$ of solar panel surface resemble the earlier FMI PV datasets in their structure.

Timestamp[UTC]	Minute	POA(W)
$2018-05-30\ 00:00$	0	0.0
$2018-05-30\ 00:01$	1	0.0
$2018-05-30\ 00:02$	2	0.0
⋮	⋮	⋮
$2018-05-30\ 07:34$	454	800.691861
$2018-05-30\ 07:35$	455	802.110516
$2018-05-30\ 07:36$	456	803.517424
⋮	⋮	⋮
$2018-05-30\ 23:57$	1437	0.0
$2018-05-30\ 23:58$	1438	0.0
$2018-05-30\ 23:59$	1439	0.0

Table 3.1: One day of simulated plane of array irradiance values. Note that the minute column is added to the table for convinience and it is reduntant as minutes can be read from the timestamps.

3.0.1 Python function and input parameters

In the thesis code, the irradiance simulation function is declared with the following function header.

Listing 3.2: PVlib POA simulation function header.

```
def get_irradiance(year, day, lattitude, longitude, tilt, azimuth):
```

The following listing contains the parameters of the plane of array irradiance function and their domains.

- Year $\in \mathbb{N}$
- Day [1, 365/366] $\in \mathbb{N}$
- Latitude [-90, 90] $\in \mathbb{R}$
- Longitude [-180, 180] $\in \mathbb{R}$
- Tilt [0, 90] $\in \mathbb{R}$
- Azimuth [0, 360[$\in \mathbb{R}$

The day and year parameters can be assumed to be always known and this leaves four unknown system parameters. If each parameter combination results in an unique output, the likely system parameters can be solved by exhaustively searching the parameter space and comparing the resulting output data to known power generation data.

The combination of these four parameters and their ranges can be thought to form a subspace in four-dimensional Euclidean space. This so-called parameter space and its "volume" are both concepts that can be used for analyzing the difficulty of parameter estimation problems, behavior of parameter estimation functions and their efficiency. In general, the more parameters and thus dimensions there are, the larger the resulting parameter space is and the harder the problem becomes. And the more parameters an algorithm is attempting to solve at once, the slower the algorithm can be expected to be. If each parameter is discretized to 20 evenly spaced values, solving all the 4 parameters at once by testing out every possible

combination would require evaluating 20^4 or 160 000 unique combinations. However if the parameters could be solved one by one, isolted from the influence of other parameters, there would only be $20 * 4$ or 80 unique combinations, a reduction of 2000 to 1. This highlights how important it is to break larger problems into smaller problems whenever possible.

3.1 PVlib POA evaluation

Before implementing the parameter estimation functions, the POA simulations should be tested against known measurement data. Figure 3.3 indicates that in clear sky conditions the pvlib irradiance model is following real world measurements closely with a few exceptions. During cloudy days, the measurements can be seen to exceed the power generation estimated by the model. The cause for this increase is likely to be caused by the additional sunlight reflected from clouds towards solar panels in partly cloudy weather conditions and it shows that cloud induced noise can be positive as well as negative.

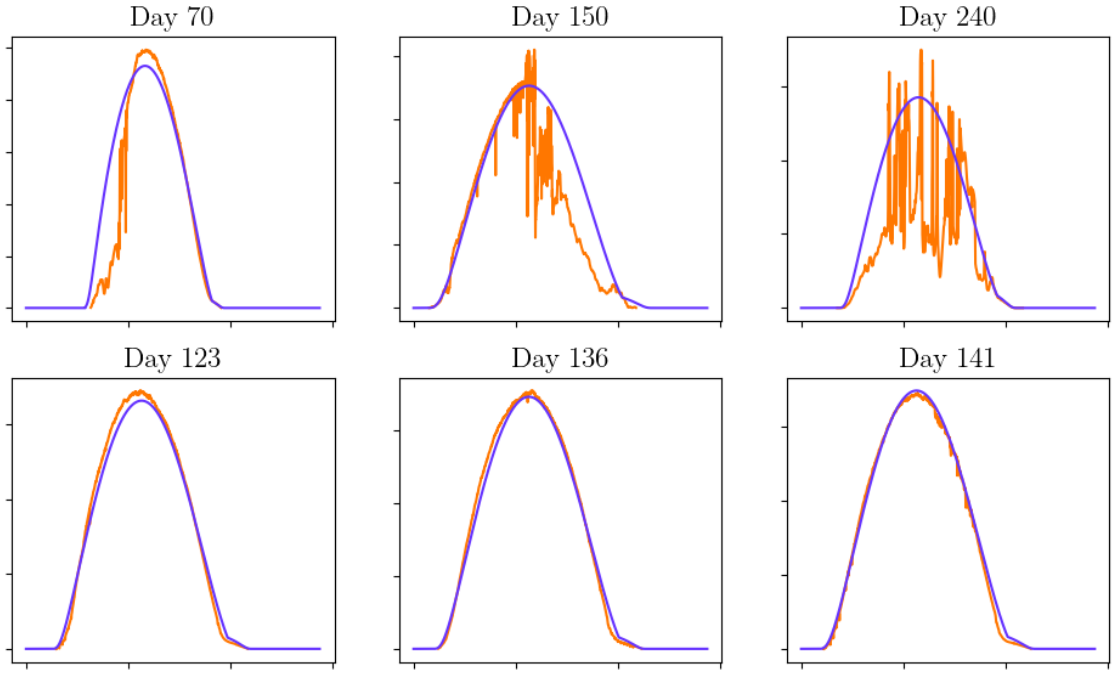


Figure 3.3: Power output of FMI Kumpula PV installation and the pvlib POA simulation computed with the parameters 2.3. Horizontal axis on the graphs corresponds to time and vertical axis marks the estimated power values. The purpose of the graphs is to display the different shapes and deviations from POA models and thus axis names and numbers were left out. Upper row contains randomly selected days while as the lower row has days chosen by a clear sky algorithm mentioned in chapter 2.3. Measurements are from 2017. POA irradiance values were multiplied by 20 in order to match the curves values on power axis.

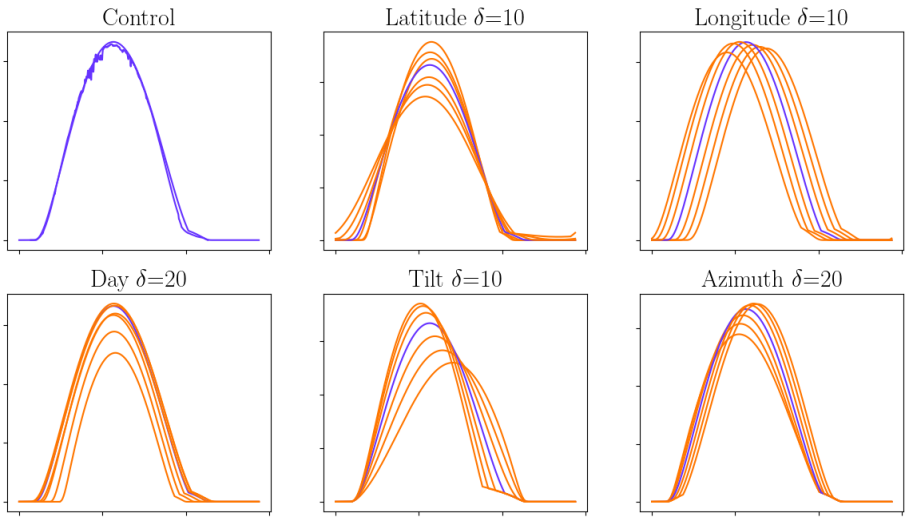


Figure 3.4: Influence of changes in PVlib simulation parameters on generated power output curves. Control shows FMI Helsinki measurements and simulation with the same parameters as the Helsinki installation. Simulated power values are multiplied by 19 in order to match values on y-axis.

By varying the different simulation parameters as shown in figure 3.4, we can examine the relationships between parameters and power generation curves. This can help us understand if there are usable patterns in the data. In the best case scenario each of the simulation function inputs would affect one measureable property in the irradiance plots and their relationship would be bijective. To give an example, if the peak power minute was isolated from all other parameters than the longitude and the relationship between longitude and peak power minute was linear, it would be possible to solve the peak power minute to longitude function with just a few plane of array irradiance simulations.

In the exact opposite case where every measureable property of irradiance plots is affected by every input parameter, solving the parameters would be much harder or even impossible. For example if all of the parameters influenced the same traits to different extents and the system was not bijective, multiple parameter combinations could result in the same simulated power graph. In a such system there would not be a single solution but rather a set of possible solutions.

The problem of solving installation parameters lies somewhere in between the two extremes. The longitude parameter would seem to shift the curve along the time

axis where as tilt and azimuth parameters do not affect the first or last non-zero minutes but they do affect the shape of the curve. Observations of parameter to trait interactions are listed on table 3.5.

Parameter	Traits affected
Latitude	Shape, first and last minute times
Longitude	First and last minute times
Tilt	Shape
Azimuth	Shape

Table 3.5: Function input to observed trait table.

3.1.1 Influence of different longitudes

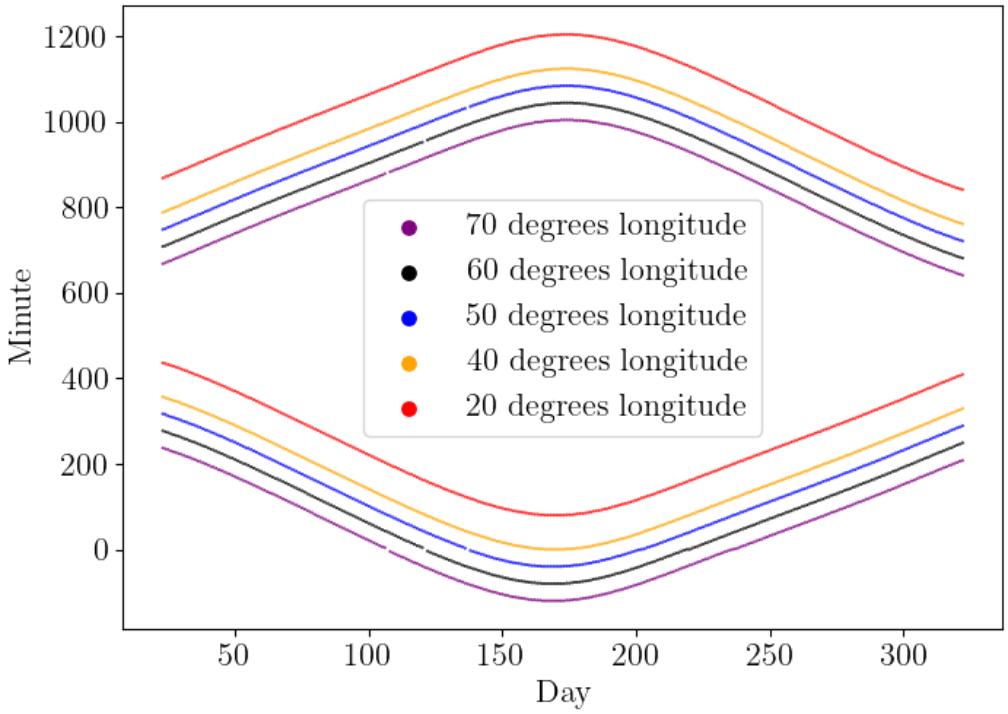


Figure 3.6: First and last non-zero minutes of each day from year long simulations at different longitudes.

Based on earlier observations listed in table 3.5, solving the longitude of installations would seem like a sensible starting point. The figure comparing the effects of different parameters seemed to suggest that the relationship between longitude and significant minute times is very close to linear and the same is seen here in figure 3.6. In Hagdadi 2017 [1] and in Williams 2012 [2] this relationship was used in order to determine the geographic longitude. The algorithms used by both of the articles relies on calculating an approximation for the time of the solar noon based on the average of the first and last minutes, this solar noon minute is then translated into a geographic longitude coordinate.

3.1.2 Influence of different latitudes

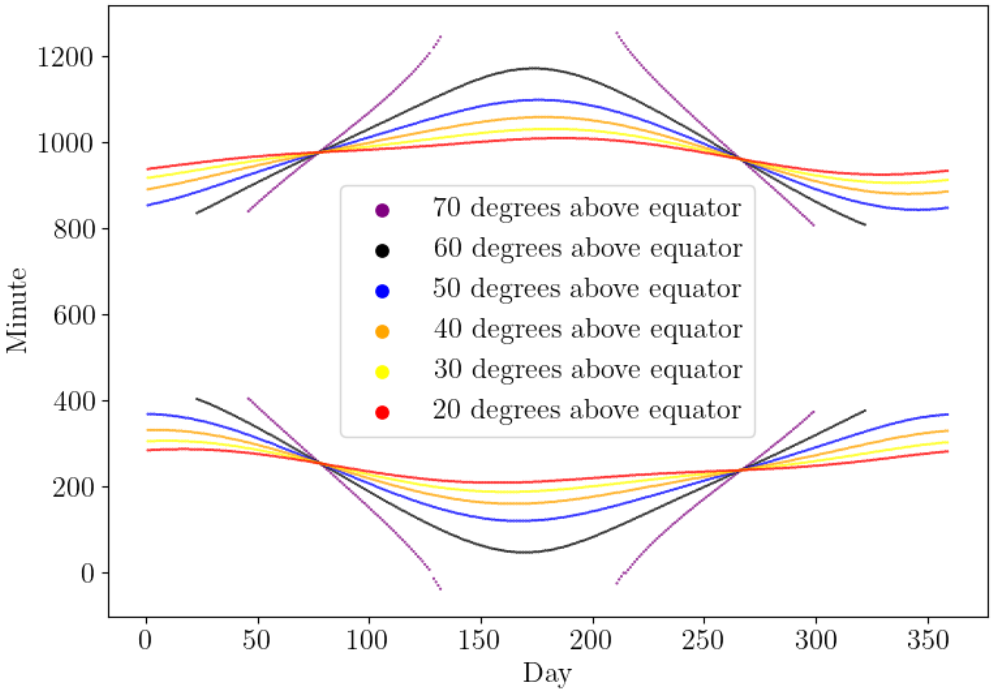


Figure 3.7: First and last non-zero power minutes of each day from year long simulations at different latitudes

The latitude simulations show that the day length stays fairly consistent for locations close to the equator, but with latitudes of 50° and higher, the day to day variation is significant. These POA simulations would imply that the region around equinoxes is ideal for day length based analysis as there day length is always well defined and the rate of change can be measured.

3.2 Increasing the accuracy of solar PV simulations

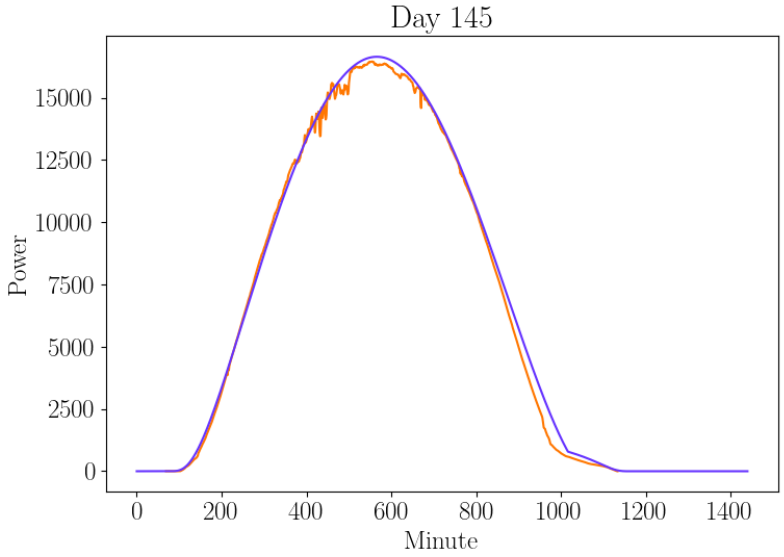
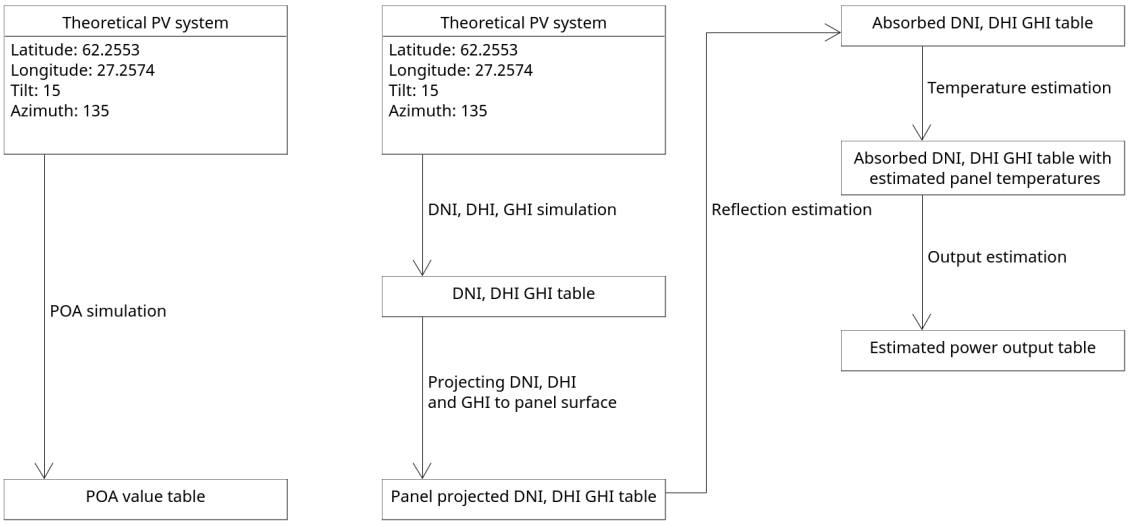


Figure 3.8: Figure comparing Pvlib simulated POA irradiance and actual measured PV output of FMI Helsinki installation.

Figure 3.8 compares the simulated and measured PV output during a sunny day in Helsinki. The shapes of the curves match quite closely, but there are some discrepancies. The estimated power values for the peak production hours are higher than measured power values and a similar phenomena is seen near minute 1000. These discrepancies can be explained by real world phenomena as PVlib POA values describe the amount of radiation reaching the surface of a solar panel. This is different from the power output of a PV system as thermal losses and panel reflections are not taken into account. This section examines an improved PV model which includes these losses.

3.2.1 Improved PV model



Original POA-based PV model and the improved model with reflection and temperature estimation.

The improved PV model introduces some new concepts, the most important of which are the following irradiance types.

- DNI is diffused normal irradiance which represents direct sunlight received by a $1m^2$ -sized plane with AOI of 0 degrees. Note that athmosphere scattered or ground reflected irradiance are not components of DNI.
- DHI is diffuse horizontal irradiance which represents the irradiance reaching a shaded $1m^2$ -sized plane with tilt of 0 degrees. DHI represents atmosphere scattered irradiance.
- GHI is global horizontal irradiance and it represents the amount of irradiance per $1m^2$ -sized plane with tilt of 0 degrees. GHI is made up of direct irradiance and atmosphere scattered irradiance. GHI combined with albedo can be used to estimate the ground reflected irradiance.

These irradiance types are present in both the simpler PVlib POA based model and the improved PV model. The difference between the two models is that in the POA-model, PVlib internally calculates three irradiance components DNI, DHI and GHI

and projects them to the panel surface, returning this panel projected irradiance as plane of array irradiance. Where as in the more complex PV model, PVlib estimates DNI, DHI and GHI, each of which are processed by multiple functions before they are combined into an estimated power output value, resulting in a physically more accurate PV output estimation.

3.2.2 Panel surface projections

The first step in the improved model is panel surface projection. Sandia National Laboratories, the original author of PVlib suggests the following two equations for DNI and GHI projection and five alternative models for DHI projection. Perez 1990 model [4] was chosen for DNI projections due to the use of the model by a co-worker at FMI and inclusion in Sandia suggested projection models [5]. For implementation of DNI, DHI and GHI equations, see thesis source code.

DNI projection [6]

$$DNI_{proj}(DNI, AOI) = DNI * cos(AOI) \tag{3.1}$$

GHI projection [7]

$$GHI_{proj}(GHI, albedo, tilt) = GHI * albedo * \frac{1 - cos(tilt)}{2} \tag{3.2}$$

3.2.3 Reflection estimation

3.2.4 Panel temperature estimation

3.2.5 Output estimation

3.2.6 Results

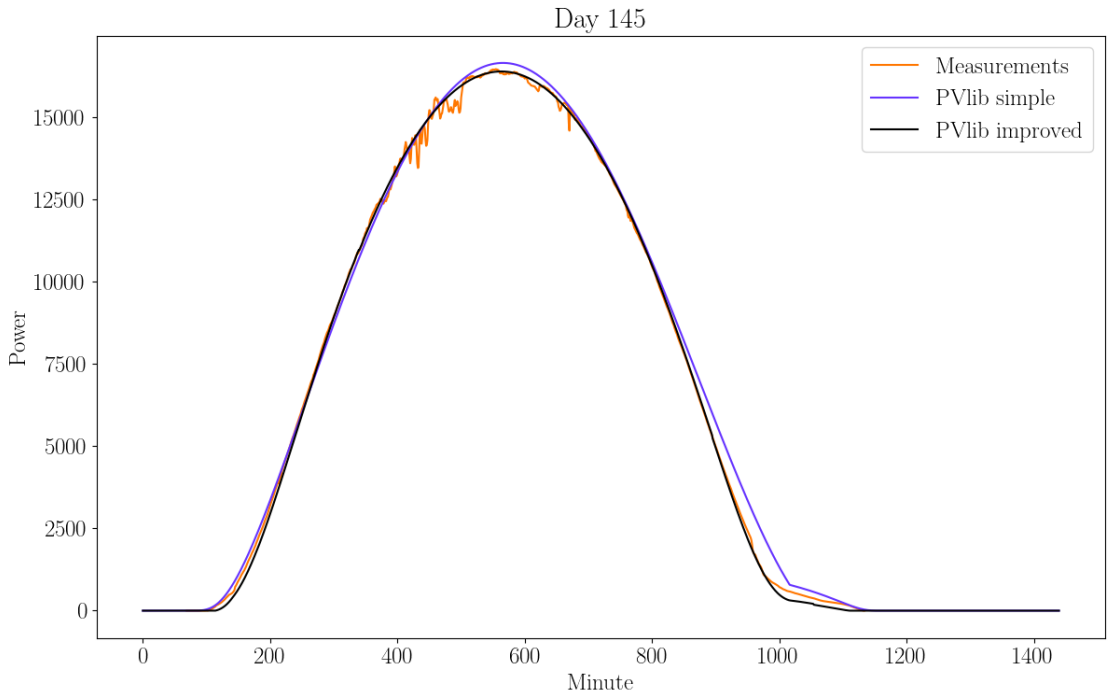


Figure 3.10: Comparison of PVlib POA and a more accurate PV power generation model.

Figure 3.10 shows that the improved PV model performs better than the simple POA-model at 1000 minutes where the AOI is high and thus reflective losses from direct irradiance are significant. And during peak production minutes where the panel temperatures are likely high enough to cause thermal losses.

The improved model would appear to be a more accurate approximation for PV generation than the POA model, but the POA model still has uses in applications where faster computation is beneficial or where only the timing of the first and last non-zero minutes matter. Which of these models is being used will be mentioned in the following chapters.

The model performance could be further tuned as inverter efficiency and other possible factors were not taken into account. However as there are only two datasets, this would increase the likelyhood of overfitting and thus the improved model is left as is.

Chapter IV

Estimating geographic location

In order to evaluate the performance of longitude and latitude estimation functions, it may prove useful to be able to translate the error values from degrees to kilometers. The following two equations 4.1 and 4.2 can be used to approximate the deltas of longitude and latitude estimation functions in kilometers. Note that these functions can be off by several percents as they rely on the assumption that the Earth is a perfect sphere and not an irregular ellipsoid.

Latitudinal distance to kilometers(Distance on North-South axis)

$$Distance_{latitudinal}(lat\_d) = (40000km/360°) * lat\_d \tag{4.1}$$

Where lat_d is the distance between two points in degrees latitude and 40000km is an approximation for Earths circumference.

Longitudinal distance to kilometers at given latitude(Distance on East-West axis)

$$Distance_{longitudinal}(lon\_d, lat) = (40000km/360°) * cos(lat) * lon\_d \tag{4.2}$$

Where lon_d is the distance in degrees longitude and lat is the latitude for which the distance is calculated.

As long as the deviations are small enough and highly accurate error values are not needed, the total error in absolute terms can be estimated by using the latitudinal and longitudinal distances as the x and y coordinates on a cartesian plane and computing the euclidean distance between the origin and resulting point.

4.1 Estimating geographic longitude

As mentioned in sections 3.1.2 and 3.1.1, the geographic location of a PV system has a strong correlation to the timing of the first and last non-zero measurements of each day whereas the influence of tilt and facing parameters seems to be nonexistent. The relationship would seem to be so clear that without further analysis it would be tempting to use fairly simplistic mathematical models for these estimations. The following longitude estimation function 4.3 can be derived with the use of two basic assumptions. These assumptions are that solar noon occurs at 12:00 or 720 minutes at longitude 0 each day and at 6:00 or 360 minutes at 90 degrees. Rest of the values can then be linearly interpolated. Note that here solar noon refers to the midpoint between the first and last non-zero minute which is different from astronomical solar noon which occurs nearly at the same time.

As only the first and last non-zero minute times are relevant for longitude and latitude estimation, PVlib POA model is used for both longitude and latitude estimation.

Naive solar noon to longitude equation

$$Longitude(sn) = 180^\circ - \frac{360^\circ}{1440} * sn \tag{4.3}$$

Where sn is the approximated solar noon minute calculated by taking the average of first and last non-zero power generation minute of a day.

The simplicity of 4.3 makes the equation appealing, but the assumption of solar noon occuring at 720 minutes should still be verified. In figure 4.1 solar noons can be seen to occur at around 720 minutes at longitude 0 but they can also be observed occuring 15 minutes earlier or later than that. This 15 minute delta would translate into an error range of ± 3.75 degrees or approximately $\pm 200km$ at the latitudes of Helsinki according to the equation 4.1.

Knowing that the PV installation is within a 400 kilometer wide slice should be in most cases be accurate enough for determining the country in which the PV installation is located in, but for most other purposes this level of accuracy is unlikely

to be valuable. Fortunately the naive model can be improved upon by taking the solar noon timing variation into account.

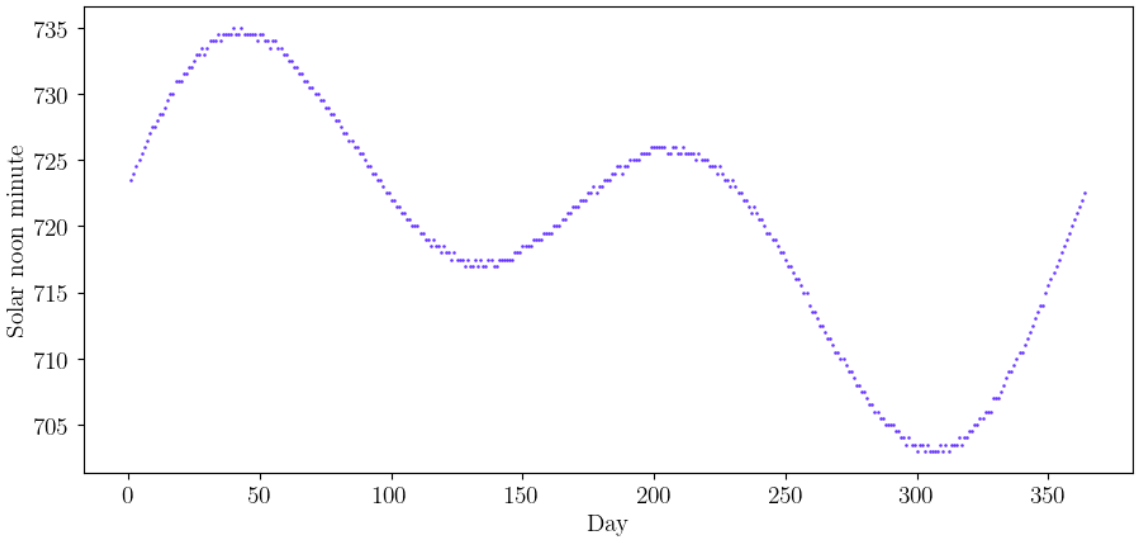


Figure 4.1: Approximations of solar noon minutes based on PVlib POA function at longitude 0° for year 2023. This pattern is caused by the Earths axial tilt and elliptical orbit around the Sun [8].

Improved longitude estimation function

$$Longitude(sn) = \frac{360}{1440}(sn_{poa} - sn) \tag{4.4}$$

Where sn is the solar noon estimate based on measurement data and sn_{poa} is the simulated solar noon at 0 degrees longitude. The new function parameter sn_{poa} compensates for the variation seen in 4.1.

The improved algorithm should no longer have a systematic error of up to 15 minutes after the analemma has been taken into account. In addition to correcting for the irregular solar noon timing, the algorithm can be improved even further by using the algorithm on larger sections of data and averaging the results, or alternatively the algorithm could be applied only on selected cloud free days where the expected errors are likely to be smaller.

If the unfiltered multi-day approach is used, choosing the right day range is crucial. If the range is too narrow, a single outlier value can distort the results significantly, however if the whole year is used, certain periods of the year may contain more noise than others and thus their use could decrease the accuracy of the results. The two scatterplots in figure 4.2 show that the data quality from the very first and last days of the year seem to be significantly worse than the data from the longest days of the year. Based on these visualizations, days outside the range of 100th to 280th would seem unsuitable for first and last minute based analysis for the Helsinki and Kuopio installations.

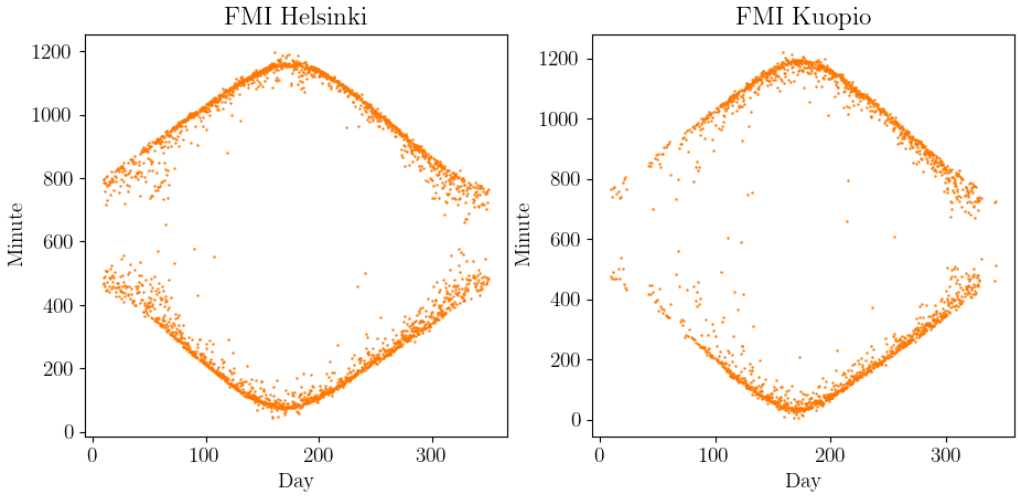


Figure 4.2: First and last non-zero power minutes of each day during years 2017 to 2021 from FMI Helsinki and Kuopio datsets.

4.1.1 Longitude estimation results

The improved algorihm was tested on the day range of 125th to 250th of each year from both FMI datasets and the results can be seen on the table 4.3. For the Helsinki installations these estimates are all off by less than 0.3° while as the Kuopio installation deltas are a bit higher with max of just over 1°. More impressively, the mean delta of multiple years for the Helsinki dataset is just 0.07° and 0.46° for Kuopio. In kilometers, the mean deltas can be approximated to 4 and 28 kilometers respectively. The lower accuracy of the Kuopio estimations could be due to multitude of factors ranging from differences in local climate or lower elevation of the installation among others.

Year	Longitude Helsinki	Δ°	Longitude Kuopio	Δ°
2021	25.115°	0.154°	26.625°	−1.009°
2020	25.029°	0.068°	27.691°	0.057°
2019	24.944°	−0.017°	27.411°	−0.223°
2018	25.243°	0.282°	26.862°	−0.772°
2017	24.836°	−0.125°	27.297°	−0.337°
mean	25.031°	0.07°	27.177°	−0.457°

Table 4.3: Means of multi-day longitude estimations from 125th to 250th day of each year.

4.1.2 Possible issues and further development ideas

While experimenting with the solar minute estimation functions, a curious trait was found. In figure 4.4, the average of the first and last minute is approximately the same for each day at different latitudes as long as the latitude is below 50 degrees. As the latitude is increased, the solar noon estimates begin to deviate significantly, becoming strongly skewed after 70 degrees.

At first this behavior seems strange as astronomical solar noon should occur happen at the same time when longitude and the day are the same regardless of latitude. However as the solar noon estimates are calculated based on the first and last non-zero irradiance minute of the day, it would make sense that the estimations could be off by significant amount during equinoxes due to rapid changes in day lengths. Measuring out how significantly this affects longitude estimations is challenging. In theory, if the same bias occurs in both the measurements and the model, no corrections would be needed. The effect should be also lessened by using longer day ranges for predicting longitudes or by making sure that the intervals include an equal amount of days from both halves of the year.

Improvements in the algorithm accuracy could also be achieved via by increasing the sampling interval of the irradiance simulations. PVlib POA simulations include a parameter for sampling frequency which is currently set to 1-per-minute in order to match the measuring frequency of FMI datasets. This could be increased to 1-per-second and the added resolution could help in determining more accurate estimates for solar noon times, resulting in possible gains in algorithm accuracy.

PVlib POA model was used instead of the more complex reflection and temperature aware model. This could be done as the geolocation is connected to first and last non-zero minute times which should be the same for both models. However even the POA model might be overly complicated as only two time values are needed.

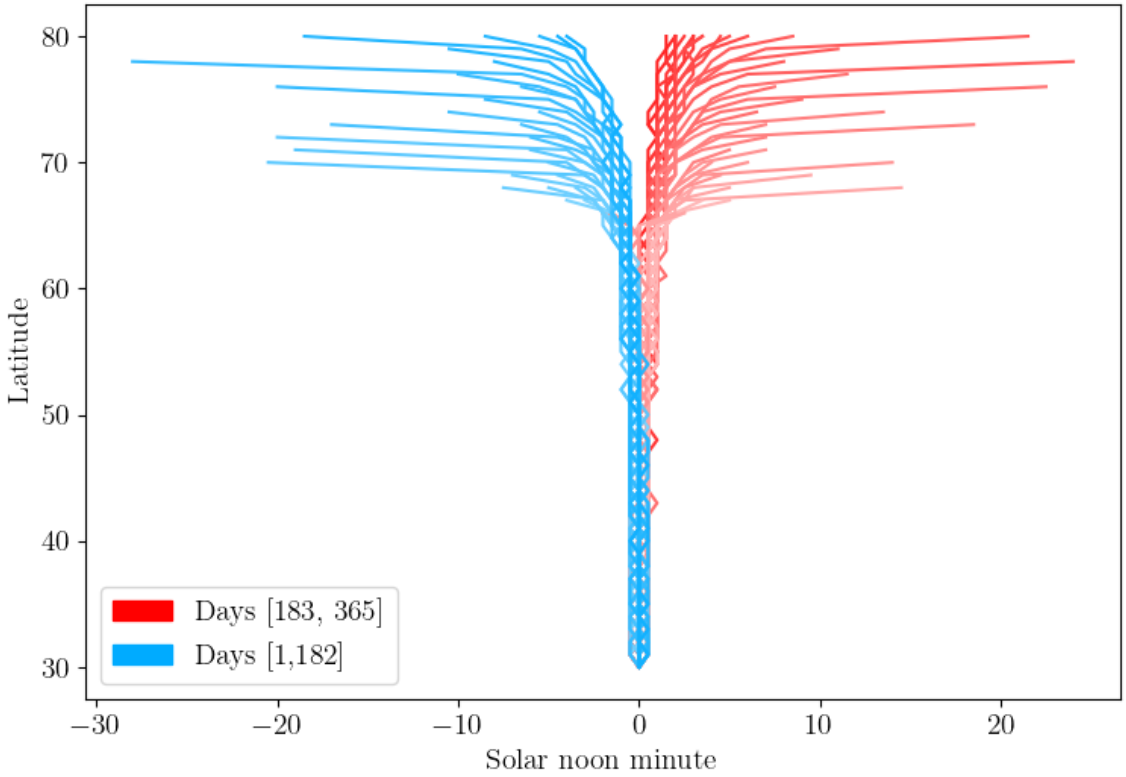


Figure 4.4: Relationship between latitude parameter and estimated solar noon time. Each line represents a different day of the year and x-axis values are normalized so that each line begins at 0 deviation. Lines with darker colors mark days which are further away from spring and fall equinoxes.

4.2 Estimating geographic latitude

Similarly to the longitude, the latitude of an installation is strongly connected to the timing of the first and last non-zero minutes of the day. This means that PVlib POA simulations can be used instead of the more complex PV model.

In figure 3.7, the simulated first and last minutes can be seen to change day by day at varying rates based on the latitude. In mathematical terms it could be said that the derivative of the day-to-first-minute function is determined by the latitude of the installation. And for the days around equinoxes, and at higher latitudes of 50° to 70°, this relationship would seem to be bijective as per earlier figure 3.7. The followinth algorithm is based on the former observations.

4.2.1 Latitude algorithm

1. Simulate first non-zero minutes over a specific day range at a given latitude.
2. Fit a linear equation to the simulated day to first minute pairs from step 1.
3. Repeat steps 1 and 2 for multiple latitudes and graph the relationship between days and first minutes.
4. Fit a line to the graph from step 3 and save the line slope.
5. Create a slope to latitude graph.
6. Fit an n-degree polynomial equation to the graph from step 5.
7. Calculate the slope of first minutes for real measurement data over the same range as was done in step 1 and use the slope as input for polynomial from step 6. The value of the polynomial is the estimated latitude.

Notes: Due to seasonal differences in data quality, polar winters and the midnight sun, the range of days chosen for the algorithm is important. If the range is short, individual outliers in measurements can result in large errors. Whereas if the range is too long, it will be harder to choose the range while avoiding low data quality sections. In the algorithm visualization figure 4.5, the range of 250th to 300th seems to result in acceptable slope to latitude curve smoothness.

4.2.2 Latitude algorithm visualization

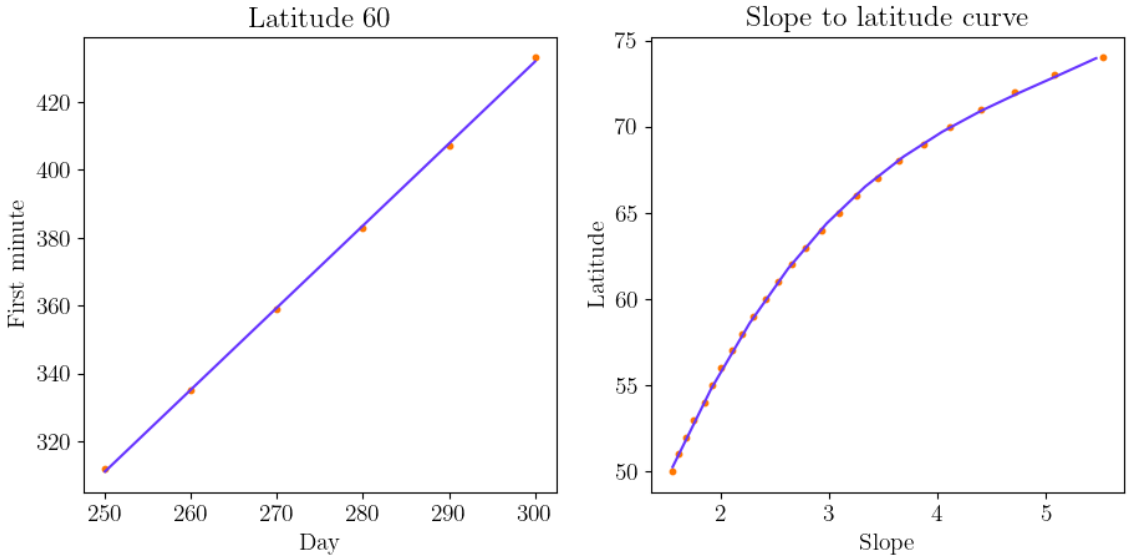


Figure 4.5: Left shows the almost linear relationship between day and simulated first minutes. Right shows the relationship between the slope angle and latitude.

FMI Kumpula		
Year	Predicted latitude	Error
2021	61.365°	1.161°
2020	64.493°	4.289°
2019	63.121°	2.917°
2018	61.190°	0.986°
2017	57.515°	−2.789°

Table 4.6: Results from estimating the latitude of FMI Kumpula PV installation with the preceeding algorithm. Day range of 250th to 300th was used.

4.2.3 Improving latitude prediction algorithm

The results of the algorithm shown in table 4.6 are somewhere in the correct range, but the delta of over 4° in the 2020 estimate is significant and much higher than the error of the longitude estimation algorithm. The first step in improving the algorithm

would be the use of last non-zero minutes as well as the first non-zero minutes. This doubles the amount of outputs from the algorithm and while doubling the amount of outputs does not directly increase the accuracy of the algorithm, it can provide additional insights into the performance of the algorithm. This is especially valuable as the available datasets are small.

FMI Kumpula				
Year	First min. p.	Error	Last min. p.	Error
2021	61.365°	1.161°	63.685°	3.481°
2020	64.493°	4.289°	64.288°	4.084°
2019	63.121°	2.917°	66.762°	6.558°
2018	61.190°	0.986°	60.230°	0.026°
2017	57.515°	−2.789°	62.256°	2.052°

Table 4.7: Latitude algorithm with added output for last minutes based prediction.

The second step in improving the algorithm is choosing the best possible day range for latitude estimation. One way of choosing the day ranges would be by testing multiple day ranges and choosing the range which results in the lowest average absolute error from the known latitude, but this is problematic as the correct latitude should not be assumed to be known. However if there were multiple datasets with complete metadata, this could be used in order to find universally well-behaving day ranges.

Standard deviation minimization is the second option for automated day range selection. As there are two estimated latitude values per year, datasets with n years of data would provide $n * 2$ estimated latitude values. Standard deviation of these values could expected to be small if the day interval does not contain days with bad data quality and this means that the interval selection can be automated. Following figure 4.8 shows the general shape of the standard deviation plane for FMI Kuopio instalation.

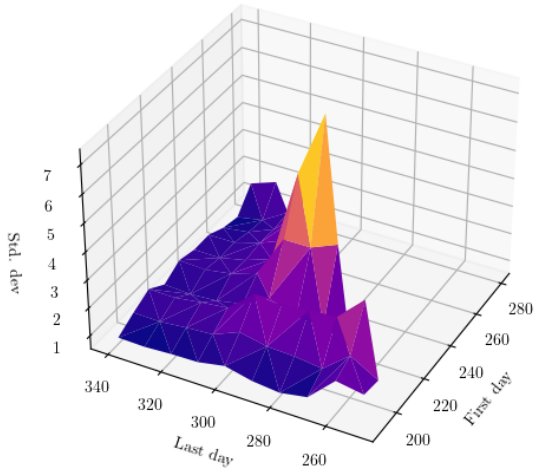


Figure 4.8: 3D -surface plot of day interval ranges and resulting standard deviations for FMI Kuopio dataset.

4.2.4 Latitude estimation results

The following two tables contain examples of the results of the latitude estimation algorithm. Results of the latitude estimation algorithm are not as good as the longitude estimations, but for now they will suffice. The predictions follow a similar patter as the previous longitude estimations in that predictions for the Helsinki installation are grouped tighter and their errors are lower than those of the Kuopio installation.

FMI Helsinki latitude estimation results				
Year	First min. p.	Error	Last min. p.	Error
2021	59.792°	−0.677°	60.186°	−0.334°
2020	59.792°	−0.412°	60.186°	−0.018°
2019	59.896°	−0.308°	59.558°	−0.646°
2018	59.945°	−0.259°	59.463°	−0.741°
2017	60.577°	0.373°	60.008°	−0.196°

Table 4.9: Estimated latitudes for FMI Helsinki Kumpula dataset with day range of 190th to 250th

FMI Kuopio latitude estimation results				
Year	First min. p.	Error	Last min. p.	Error
2021	62.626°	−0.266°	63.197°	0.305°
2020	62.259°	−0.633°	61.895°	−0.997°
2019	62.983°	0.091°	62.708°	−0.184°
2018	62.722°	−0.170°	62.874°	−0.018°
2017	61.669°	−1.223°	61.152°	−1.740°

Table 4.10: Estimated latitudes for FMI Kuopio Kumpula dataset with day range of 190th to 280th.

4.2.5 Possible issues and further development ideas

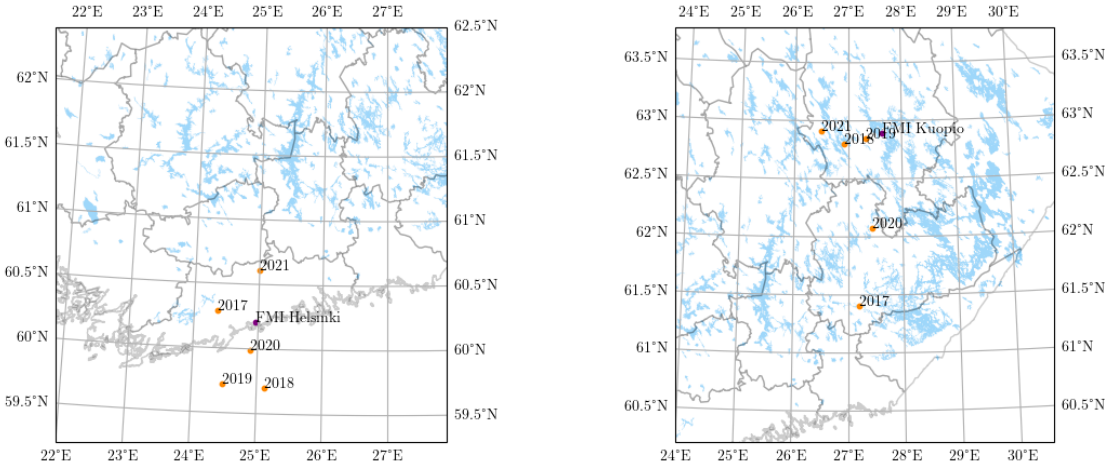
PVlib POA based first and last minute estimations are slow to compute and thus exhaustively searching the day interval space can take up to several hours of computing time. As only the first and last minute times are needed, the use of simpler computational methods could improve the computation speed significantly, allowing for the use of brute force day range selection algorithms.

Different methods could also be used. In Hagdadi 2017 [1] latitude estimations are done by fitting solar irradiance models with 3 unknown parameters to power generation measurement data. The latitude deltas of 1.65 to 3.42 degrees in the 2017 article are higher than those achieved in this thesis, however as the datasets, geographical regions and algorithms are different, direct comparison can not be made.

In earlier figure 4.5 the slope to latitude fitting can be seen to be slightly off. This is because the polynomial used is of 2nd degree and higher degree polynomials may result in a closer fit. Similarly a piecewise linear interpolation based fitting could result in a more accurate model and thus better estimation accuracy.

4.3 Combined latitude and longitude estimations

As it is unlikely that the longitude and latitude estimation algorithms are used in isolation from one another, their results should be examined together. This can be done by plotting the estimated locations on a map. Here the two installations in Helsinki and Kuopio and their predicted locations per year are plotted side by side with day range of 190 to 280.



(a) FMI Helsinki.

(b) FMI Kuopio.

Figure 4.11: Geolocation estimations for FMI datasets.

In the Helsinki predictions figure, the estimated geolocations are scattered around the known installation location, showing very little bias and some random noise. Similar behavior can be seen in Kuopio predictions where two outliers 2017 and 2020 deviate more significatly. One degree on the latidue axis is approximately 110 km regardless of latitude and longitude, one degree of longitude is 56km at 60° N and 50 km at 63° N. As the deviation is strogest on the latitude axis, it is likely that the latitude prediction algorithm is more sensitive to variations in the data and further development should be focused on more accurate latitude prediction and day range selection.

Chapter V

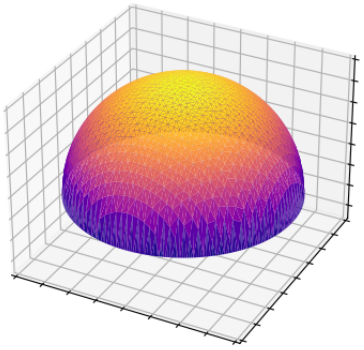
Estimating panel angles

Solar panel installation angles are a large factor in deciding the energy output of a PV system. If panel angles can be freely chosen during planning and installation phases, it can make sense to either optimize for total power generation or power generation during peak consumption hours. This means that even if installation angles could be freely chosen, installation angles are unlikely to be the same for every system in the same geographical region. Panel angles may also be restricted by installation sites and mounting types.

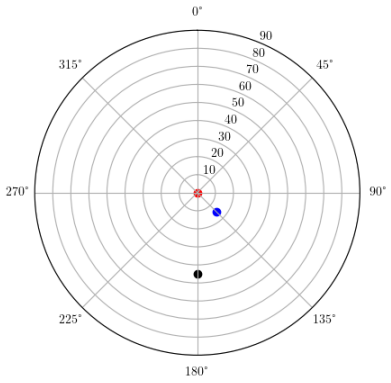
One reason for lacking of faulty metadata is that panel angles can be difficult to measure accurately. The tilt angle of the panels or the angle between the panel normal and zenith(the point directly above) can easily be measured with an angle ruler and a bubble level, but the azimuth angle of the panels is much harder to measure with the same degree of accuracy. If an accurate compass is used and the difference between the magnetic north and the geographic north is taken into account, metal structures and electrical systems nearby can still distort local magnetic fields enough to cause errors in measurements. The challenges in taking accurate measurements are not insurmountable, but they may contribute to the inaccuracies and the lack of available information in PV installation parameter metadata.

The space of possible panel installation angles can be thought as a half unit sphere in a spherical coordinate system where each point on the surface represents a direction to which the normal of the solar panels could be directed towards. A visualization of parameter space in 3D and 2D is shown in 5.1a and 5.1b. The 3 dots in the subfigure 5.1b mark the zenit for which azimuth is not well defined(red), the installation angles of FMI Helsinki installation azimuth 135° tilt 15°(blue) and

a close to power generation maximized installation with directly south facing panels with the tilt of 45°(black).



(a) The space of possible angles as a 3D half sphere surface. Each point represents a possible tilt and azimuth combination.



(b) 2D projection of the angle space, distance from center denotes the tilt angle of the panels and angle marks the azimuth.

Figure 5.1: Angle space visualizations.

Estimating panel installation angles requires the use of multiple functions, each of which can be defined in multiple ways. These functions are defined in the following sections.

- Prediction error function for quantifying how good a prediction was when the correct panel parameters are known.
- Model error function for measuring the error between simulated power values and measured power values.
- Multiplier matching function for matching the magnitude of simulated power values with the magnitude of measurements.
- Angle space discretization function for discretizing the angle space into n discrete points which can then be tested with model error function.

5.1 Prediction error function

In this thesis, the proposed error estimation method combines the tilt and azimuth delta values into one error angle value, the angular distance between two points on

a spherical surface. The goal is then to develop a panel angle estimation function which achieves the lowest angle error value with the available datasets.

Alternative approaches can also be chosen as the function or functions for measuring the distance between two points in angle space can be defined in multiple ways. The simplest way is to use the delta of known tilt and azimuth angles as two separate error values without normalizing in any way. This method was used in Hagdadi's 2017 article but such values are not direcly comparable between installations as the significance of azimuth delta depends on tilt angle.

Deriving angle space distance equation

Let $v = [v_1, v_2]$ and $k = [k_1, k_2]$ be two component angle-space vectors so that v_1, $k_1 \in [0, 90]$ and v_2, $k_2 \in [0, 360]$. These vectors represent points on the surface of a unit sphere and their components are the angles of spherical coordinate system. The cartesian coordinates of these points are:

$$x_v = sin(v_1)cos(v_2) \tag{5.1}$$

$$y_v = sin(v_1)sin(v_2) \tag{5.2}$$

$$z_v = cos(v_1) \tag{5.3}$$

And

$$x_k = sin(v_1)cos(v_2) \tag{5.4}$$

$$y_k = sin(v_1)sin(v_2) \tag{5.5}$$

$$z_k = cos(v_1) \tag{5.6}$$

And the cartesian distance between these two points can be calculated with the following equation:

$$d = \sqrt{(x_v - x_k)^2 + (y_v - y_k)^2 + (z_v - z_k)^2} \tag{5.7}$$

The two points and the origin form an isoceles triangle with the sides from the origin to the vector end points having the length of 1 while the distance between the vector end points is the same as d.

As the lengths of three sides are known, the angles of the triangle can be calculated with the cosine rule.

$$a^2 = b^2 + c^2 - 2bc\cos(A) \tag{5.8}$$

Where

a = Side opposing the angle A, same as earlier value d
b = Side opposing angle B, value is 1
c = Side opposing angle C, value is 1

Substituting known values into the cosine equation.

$$a^2 = b^2 + c^2 - 2bc\cos(A) \tag{5.9}$$
$$d^2 = 1^2 + 1^2 - 2\cos(A) \tag{5.10}$$
$$d^2 = 2 - 2\cos(A) \tag{5.11}$$

Solving for angle A

$$d^2 = 2 - 2\cos(A) \tag{5.12}$$
$$2\cos(A) = 2 - d^2 \tag{5.13}$$
$$\cos(A) = \frac{2 - d^2}{2} \tag{5.14}$$
$$A = \cos^{-1}(\frac{2 - d^2}{2}) \tag{5.15}$$

Renaming A as $Error$.

$$Error = \cos^{-1}(\frac{2 - d^2}{2}) \tag{5.16}$$

By first calculating the distance between the vectors using equations 5.1-5.7 and then substituting the distance into equation 5.16, the resulting angle can then be used as an error value between two panel angle measurements. Python code based on this proof is included in appedix .1.2.

5.2 Simulation error function

Simulation error function measures how much the predicted power generation values vary from the measured power generation values. The purpose of the simulation error function is to be able to generate a single numerical value which describes how well a certain parameter combination models the measurements. By then testing out multiple parameter combinations, the combination with the lowest simulation error function value should be the best fit and the parameters used for the simulation should be within a small error of the physical parameters of the solar PV installation. In 5.2 the error between a cloud free day and randomly chosen set of wrong simulation parameters is visualized.

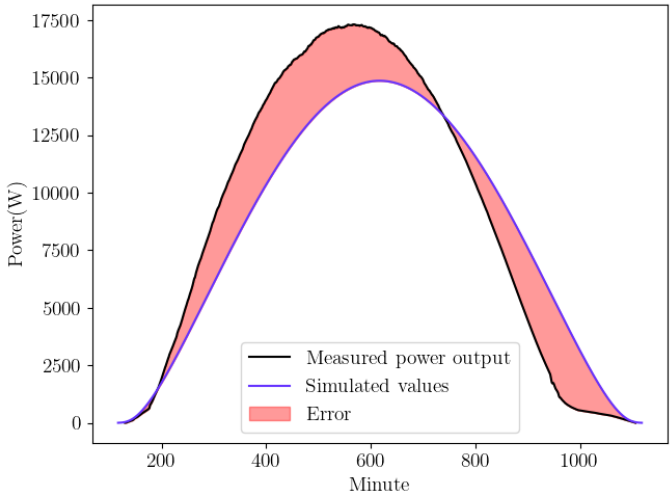


Figure 5.2: Error area between measured power values and simulated irradiance values with different panel installation angles.

5.2.1 Area based error function

The following function can be used in order to get a numerical value which signifies the area between simulated and measured power values.

$$Error = \Sigma_{t=0}^{1439} |m(t) - p(t)| \tag{5.17}$$

Where $m(t)$ is the measured power at minute t and $p(t)$ is simulated power at minute t.

The error values can also be normalized so that the error value reflects the average percentual deviation between the measured and modeled values for each minute.

$$Error\_per\_minute = \frac{\Sigma_{t=a}^{b} m(t)/p(t)}{b - a} \tag{5.18}$$

Where $m(t)$ is the measured power at minute t and $p(t)$ is simulated power at minute t. And a and b are the first and last non-zero power minutes.

5.2.2 Alternative simulation error functions

Other error estimation methods could be used as well. Skew values, peak power generation minutes and other similar charasteristics could be measured and matched. The benefit of such charasteristics matching based error algorithms is that charasteristics errors have a direction as well as magnitude and these could be used in order to estimate the proximity and direction of the best fit in angle space.

The downsides of using charasteristics are that measurement data contains noise which can distort the estimated values of charasteristics and having a direction and a magnitude is of limited value unless additional functions are created for turning these error values into directions in parameter space. These relationships between traits, their error directions and magnitudes could prove to be difficult to solve.

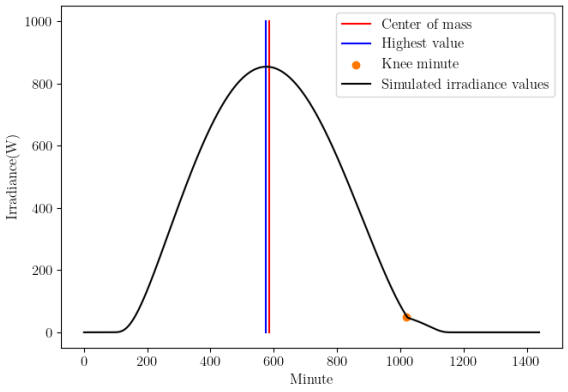


Figure 5.3: Center of mass, highest value and knee minute marked on a simulated POA irradiance figure. Note that the knee minute is not as distinct on the improved solar PV model.

5.3 Multiplier matching

As the plane of array irradiance model models the solar radiation towards a virtual 1 m^2 sized panel, using the values to simulate the power generation of a solar power installation requires the use of a multiplier. This multiplier is related to the efficiency and the surface area of the solar PV installation.

5.3.1 Area based multiplier matching

The multiplier value can be solved by making the assumption that a plane of array irradiance curve $p(t)$ with correct simulation parameters matches the measurements curve $m(t)$ in its shape but not magnitude. By calculating the sum of simulated and measured power values, their ratio can then be used as a multiplier. After scaling $p(t)$ with a multiplier to match $m(t)$ the two curves should be nearly inditinquishable.

$$M_{multiplier} = \frac{\Sigma_{t=0}^{1439} m(t)}{\Sigma_{t=0}^{1439} p(t)} \tag{5.19}$$

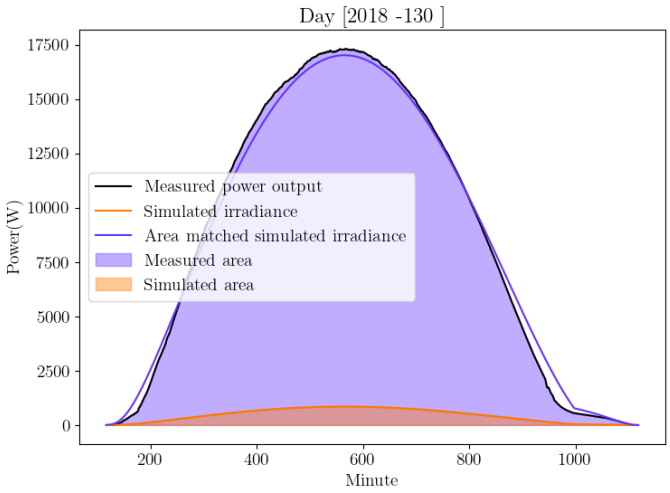


Figure 5.4: Visualization of automated multiplier matching. Measurement data from FMI Helsinki dataset, POA irradiance simulation was computed with FMI Kumpula coordinates and installation angle parameters.

5.3.2 Segmented multiplier matching

The earlier multiplier matching method falls apart if the measurements used for multiplier solving contain deviations caused by clouds, trees or other sources. If the energy production during a certain section of the day varies from the expected, then this abnormal segment will affect the resulting multiplier value. If these deviating segments could be avoided, then partially cloudy days could still be used for multiplier matching. The process of segmentation can be done with multiple different methods. Here the inverval between the first and the last non-zero power minute of the day was split into 10 segments of equivalent length.
Now that the data is split into segments, the earlier area based multiplier algorithm can be used to compute a multiplier value for each segment. A mathematical representation for the segments S_i multiplier M_i could be as follows.

$$M_i = \frac{\Sigma_{t=a_i}^{b_i} m(t)}{\Sigma_{t=a_i}^{b_i} p(t)} \tag{5.20}$$

Where a_i and b_i are the first and last minutes of the interval i.

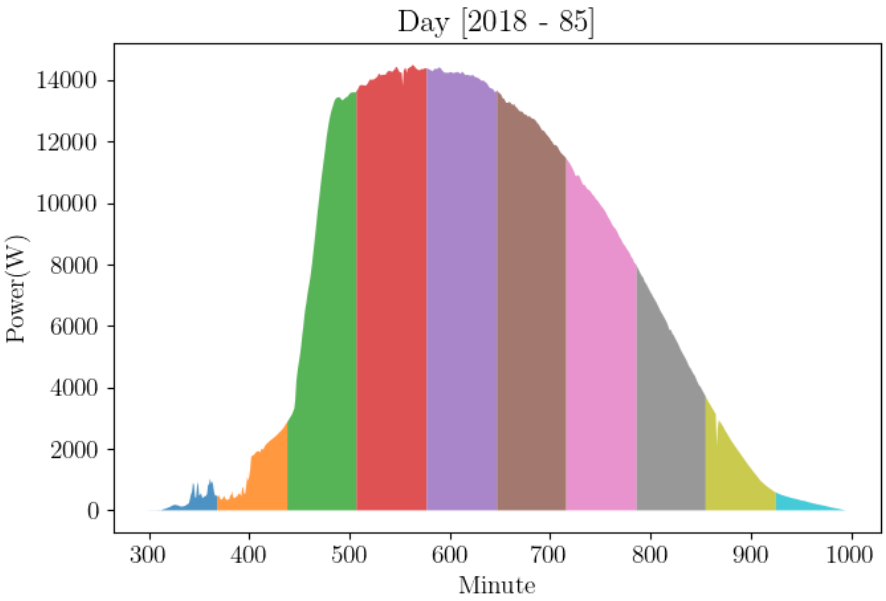


Figure 5.5: Partially cloudy day split into 10 segments.

Interval	Multiplier
[299, 367]	1.26
[368, 437]	3.22
[438, 506]	16.56
[507, 576]	21.68
[577, 646]	21.58
[647, 715]	21.52
[716, 785]	21.23
[786, 854]	19.80
[855, 924]	16.08
[925, 994]	16.80

Table 5.6: Segment based multiplier matching intervals and resulting multipliers. FMI Kumpula dataset day [2018 - 85] was used with irradiance simulation parameters listed in 2.3.

In 5.6 the segments 4 to 7 are all tightly grouped and their average is 21.50. These closely grouped segments, commonly referred to as clusters in the field of data science, can be identified algorithmically by locating a window that encom-

passes a relatively high number of values within it. For example a ±5% window $[M_i * 0.95, M_i * 1.05]$ would contain the cluster if any of the 4 cluster values was used as as M_i.

A possible issue with the clustering algorithm could rise from random chance. The last multiplier values 16.08 and 16.80 are close enough to the early 16.56 that due to random variation in the data, the algorithm could in some cases choose the wrong multiplier value cluster.

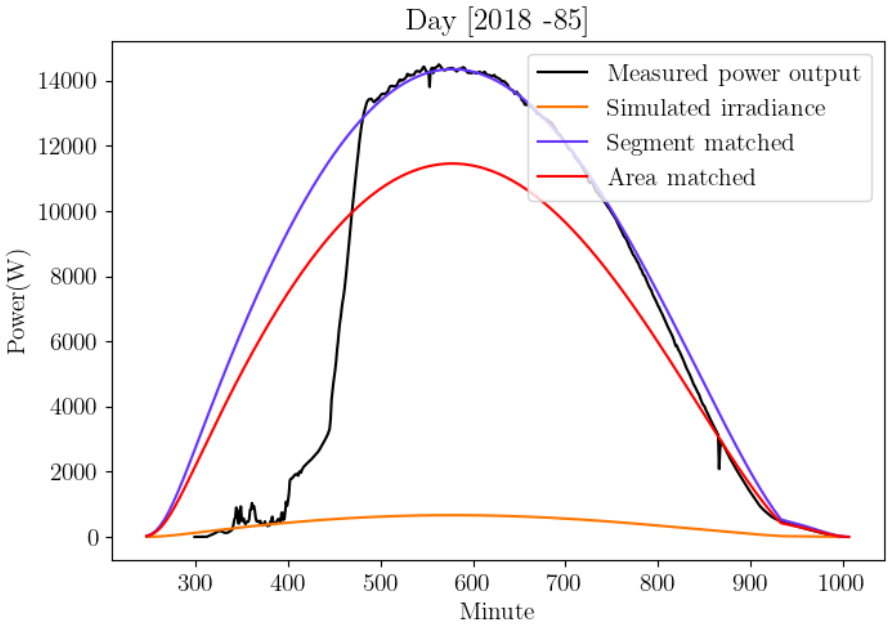


Figure 5.7: Comparison of area and segmented area based multiplier matching algorithms on a partially cloudy day.

5.3.3 Translating multiplier values to PV system rated power values

As the simulated irradiance values are an estimation of power radiated towards a sigle square meter sized imaginary panel, the scaling multiplier can be used for estimating the power rating of a solar PV installation. The power ratings of PV installations describe the expected power generation in watts that could be expected to be generated during peak power generation hours if the panels were oriented optimally.

With the help of the rated power value, the surface area of the panels can be estimated. The precise estimation is difficult as different panel types and panel age affect the efficiency of PV systems, but having some frame of reference for installation sizes could prove to be useful in other studies where lidar data or satellite images are available.

Rated power estimation equation

$$RatedPower = MP \tag{5.21}$$

Where M is the multiplier required to match the POA simulated values with measured values and P is an estimation for solar irradiance per square meter or approximately 1000w.

By using 5.21 and the clustered multiplier values from 5.6, the power rating of the Helsinki installation would be estimated as 21.5KW which is within close proximity the reported 21KW in 2.3.

Panel area estimation equation

$$PanelArea = RatePower * \frac{1m^2}{\eta} \tag{5.22}$$

Where η is the efficiency of solar panels, typically in the range of 0.15 to 0.20. The efficiency coefficient varies according to panel type, age, variance in manufacturing and other factors.

5.4 Angle space discretization

The next step is angle space discretization. The panel angles are denoted with a doublet of tilt and azimuth values, ranging from 0 to 90 and 0 to 360 respectively. If the tilt and azimuth axes are discretized individually in steps of 5 so that tilt is [0, 5, 10, 15... 90] and azimuth [0, 5, 10, 15... 355], the permuations of these tilt and azimuth values create an even grid in the euclidean projection of angle space where x = tilt, y = azimuth. However as the physical phenomena represented by the angle values is not a point on a plane but a point on a half-sphere surface, this results in an uneven discretization 5.8a. A better option is to use Fibonacci lattice [**?**] for a more even distribution of points on a spherical surface 5.8b.

Fibonacci lattice point n of k equation

$$s = n + 0.5 \tag{5.23}$$

$$\phi = acos(1 - 2s/k) \tag{5.24}$$

$$\theta = \pi s(1 + \sqrt{5}) \tag{5.25}$$

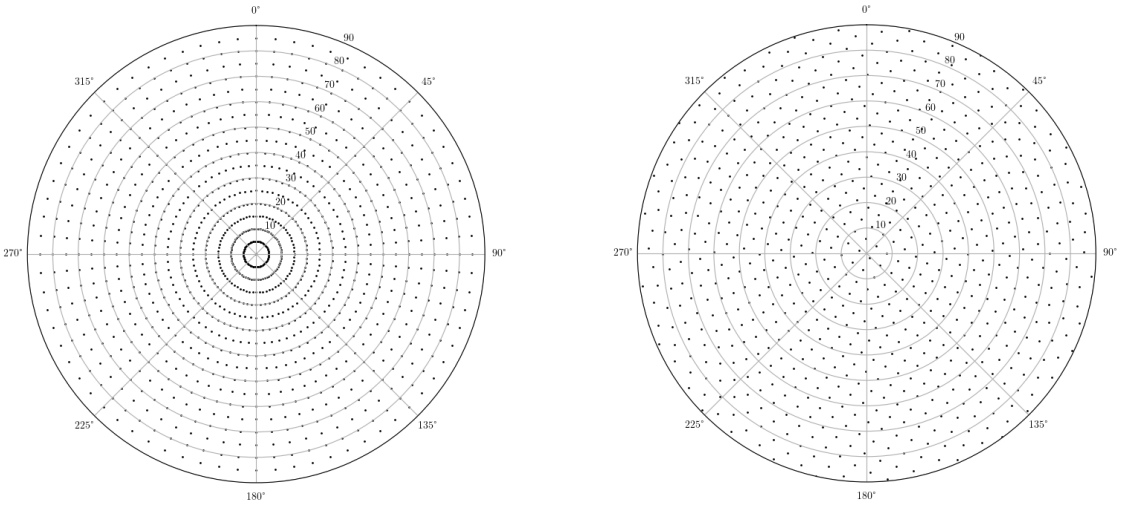
Where n is the point number, k is the amount of points, ϕ is the panel tilt angle and θ is the azimuth angle.

$$x = cos(\theta)sin(\phi) \tag{5.26}$$

$$y = sin(\theta)sin(\phi) \tag{5.27}$$

$$z = cos(\phi) \tag{5.28}$$

x, y and z are the corresponding cartesian coordinates.



(a) In steps of 5 discretization with 1296 points

(b) Fibonacci lattice-based discretization with 756 points.

Figure 5.8: Comparison of two different discretization patterns. Fibonacci lattice based discretization on right shows a more even distribution of points than the latitude-longitude lattice. The minimum density is approximately the same in both graphs despite the difference in point counts.

5.4.1 Importance of lattice density

The density of lattices is an important measurable charasteristic which proves useful during the optimization of angle estimation algorithm. The most useful metric would be the sphere center angle distance between neighboring points. This would be useful as it can be used for determining whether errors in predictions are lattice or function fitting related. For example, if the lattice neighbors are approximately 1 degree away from oneanother and the predicted angle is 5 degrees off from the known installation angle, then the error is caused by model fitting and not lattice density as there must have been multiple lattice points closer to the known angle point than the discovered best fit. However if grid density is near to or lower than angle estimation error, the lattice is likely to be a contriburing to angle estimation errors.

Calculating neighborign center angle distances for both *in-steps-of-n* and Fibonacci lattices is somewhat challenging. In *in-steps-of-n*, the value n can be used as an estimate for max center angle distance as n will always be the tilt distance to the nearest neighbor with different tilt angle. With Fibonacci lattices the easiest way of estimating center angle distances is taking the coordinates of the first two lattice points and calculating their center angle distance with earlier error equation 5.16. These first two points should be used as Fibonacci lattice points are distributed on a single arm spiral pattern, resulting in later sequential points being further from one another.

Another method for calculating Fibonacci lattice point distances is dividing the surface area of the angle space by the amount of calculated lattice points. This area-per-point value could then be used in order to estimate how far points are from oneanother on average. In later sections, the first two points derived distance will be used.

5.5 Solving panel angles

Now that the geographic location and multiplier value of installation are known to be solvable and error functions have been defined, the last step is to solve the panel installation angles. The chosen method relies on splitting angle space into n discrete points and evaluatin each of their fitness by calculating an error value. Here the angle space was split into 10 discrete points with fibonacci lattice 5.4 and the fitness of each point was evaluated with area error 5.17 and multiplier matching 5.19 functions.

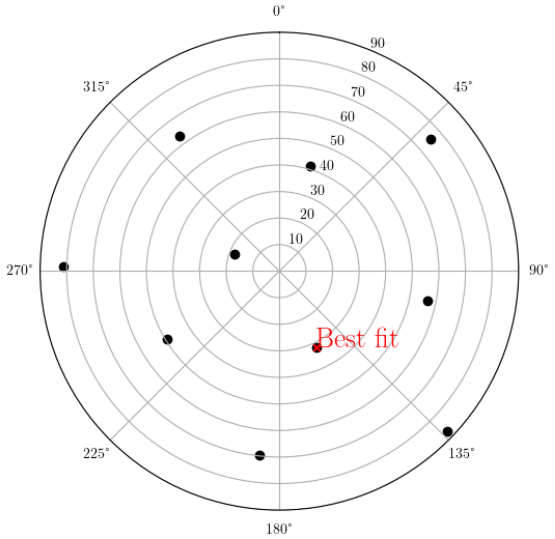


Figure 5.9: Polar plot of test points for a single day of data from FMI Kumpula dataset.

Tilt	Azimuth	Error
18.19	291.25	3785693.23
31.79	**153.74**	**265843.23**
41.41	16.23	4331163.39
49.46	238.72	4861757.21
56.63	101.22	3811371.79
63.26	323.71	7196389.58
69.51	186.2	1795822.66
75.52	48.69	5985516.23
81.37	271.18	7403298.37
87.13	133.68	2957569.38

Table 5.10: Tilt, azimuth and error table for single day.

Now that the method can be seen to work, it is time to improve the results. This can be done by generating larger lattices and thus by evaluating higher amount of datapoints, the algorithm has a higher chance of finding the global minimum error point. As per 5.4.1, Fibonacci lattice of 1000 points would have the angular resolution of approximately 4 degrees where as 10000 points would be near to 1.5 degrees. The performance can also be improved by evaluating best fits for multiple days at once. The resulting point cloud of best fits can then be used for averaging out noise in the predictions.

The plot 5.11 is a result of using the angle finding algorithm on 41 days from FMI Kumpula dataset with a Fibonacci lattice of 1000 points. The two darker

spots near the center of the graph are the two most common best fits, [17.0°, 138.4°] with 17 and [22.7°, 143.1°] with 16 out of 41 days. These groupings are as close to the known installation angles of [15°, 135°] as could be expected from a 1000 point lattice. The next step is tightening the cluster, this can be done by adjusting the smoothness requirement of the cloud free day algorithm or by restricting the day range. In 5.12 the tightening was accomplished with day range restrictions and 22 days were accepted by the algorithm.

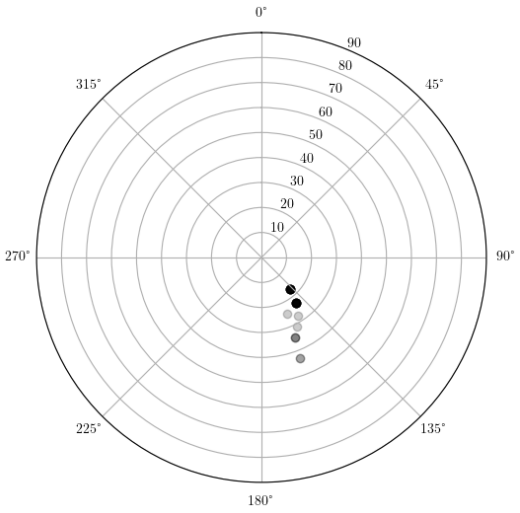


Figure 5.11: Best fits for multiple cloud free days from FMI Kumpula dataset.

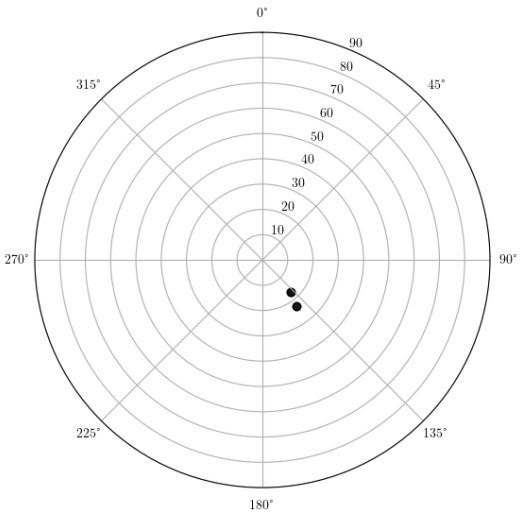


Figure 5.12: Best fits when day range is restricted to days 140-220.

As all of the estimates converge on two neighboring points, the final step is increasing the lattice resolution further. With 10 000 point lattice, the cluster tightened further 5.13. Out of the 22 days, 7/22 or 32% had best fit at [19.34, 136.87] with error of 4.38 degrees and 6/22 or 27% at [17.74, 135.06] with error of 2.74 degrees. Rest of the best fits were distributed in smaller clusters near these points. The lowest angle distance best fit was [17.09, 130.32] with angle distance of 2.46 degrees. As the angle distance errors are higher than the discretization resolution and as the predicted angles are systematically biased, it would seem that the error is caused by the solar irradiance model or model fitting and not angle space discretization.

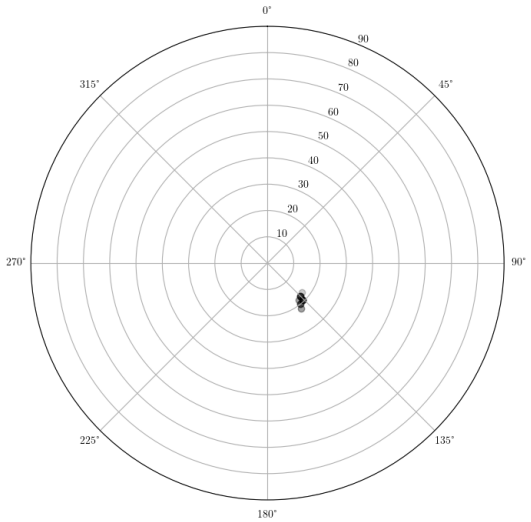


Figure 5.13: Best fits with 10 000 sample Fibonacci lattice and 22 days from FMI Kumpula dataset. Multiple years were used, day range restricted to 140-220. Actual installation angles were 15 degrees tilt, 135 azimuth.

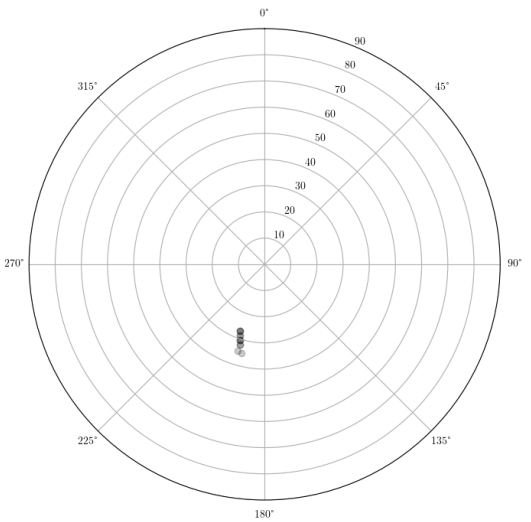


Figure 5.14: 10 000 sample Fibonacci lattice and 16 days from FMI Kuopio dataset. Multiple years were used, day range restricted to 140-220. Actual installation angles were 15 degrees tilt, 217 azimuth.

5.5.1 Evaluation of exhaustive search results

The estimated installation angles installations are fairly good. A delta of less than 4.5° as was achieved with FMI Kumpula is small enough to be a result of measurement or rounding error. The estimates for FMI Kuopio are off by more than 10 degrees which is less encouraging. As the reported angles for FMI Kuopio were 15° and 217°, it would seem like the angle measurements were rounded to nearest degree. This would eliminate the reporting accuracy as a plausible cause for the estimation errors and thus either there has been a reporting error or that the installation angle estimation algorithms are not performing as well for FMI Kuopio dataset.

Figures 5.17 and 5.18 shows that the models based on best found fits are better fits than simulations done with the known parameters. This is true for both Helsinki and Kuopio datasets. This would suggest that the model fitting works as intended and that either the model is inaccurate or there is an error in reported panel angles. The more likely cause of the two is the the solar irradiance model and for some undetermined reason the error of the model is more significant for the Kuopio

FMI Kumpula			
n	Tilt	Azimuth	Error
7	19.34°	136.87°	4.38°
6	17.74°	135.06°	2.74°
4	19.92°	141.60°	5.30°
2	21.37°	143.41°	6.87°
1	21.37°	143.41°	6.87°
1	17.09°	130.32°	2.46°

Table 5.15: Estimation results table for FMI Kumpula.

FMI Kuopio			
n	Tilt	Azimuth	Error
3	27.07°	200.24°	13.37°
3	30.50°	198.01°	16.96°
2	28.83°	199.13°	15.21°
2	32.09°	196.89°	18.65°
1	34.52°	197.58°	20.90°
1	34.52°	197.58°	20.90°
1	35.07°	194.65°	21.86°

Table 5.16: Estimation results table for FMI Kuopio.

installation. Possible causes could be related to lower sun angles resulting in higher reflective losses or shadowing during last production hours which would also explain the uneven structure visible in the last non-zero hours in 5.18.

5.6 Fast panel angle solving

Solving panel angles by exhaustively testing 10 000 possible grid points takes up to an hour of computation time on the test system. This is not a limiting factor in research code, but exhaustive searches are inelegant compared to more intelligent approaches. Geometric intuition would suggest that the surface created by panel angle pairs and fitness values would resemble a downwards pointing cone-like shape where the best fit is the peak of the cone. If this is true, searching the whole angle space is not necessary as a gradient descent algorithm can be used to approximate the direction in which the best fit resides.

Whether the cone assumption is true can be visually examined by plotting the fitness values and observing the resulting heatmap. In 5.19, the region where the global best fit resides is clearly the global minimum, but there appears to be a local minimum in the top part of the plot, near tilt 90°, azimuth 20°. Having more than one local minimum complicates efficient optimization but optimization is still feasible.

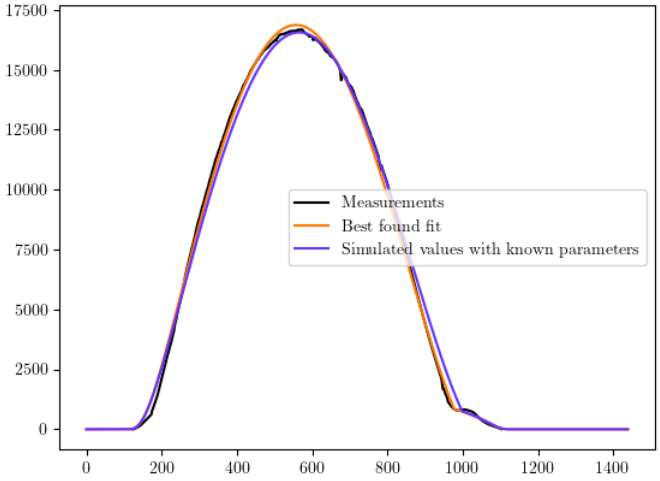


Figure 5.17: Comparison between a single day of measurements and two simulations. One with known installation angles and one with best found fit. Day is from FMI Kumpula dataset with reported installation angles of 15° and 135° degrees. Best fit at 19.34°, 136.87°

5.6.1 Gradient search

As the fitness surface is not defined by an easy to derivate multi parameter equation, directly solving for points where gradient reaches zero is challenging. However the gradient at a given point can still be numerically estimated and by taking steps in the direction of the gradient vector, local minimum can be found.

In x-y coordinate space, the numerical gradient for point

$$\frac{\partial f}{\partial x} = f(p_1 + x) - f(p_1) \tag{5.29}$$

$$\frac{\partial f}{\partial y} = f(p_1 + y) - f(p_1) \tag{5.30}$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{5.31}$$

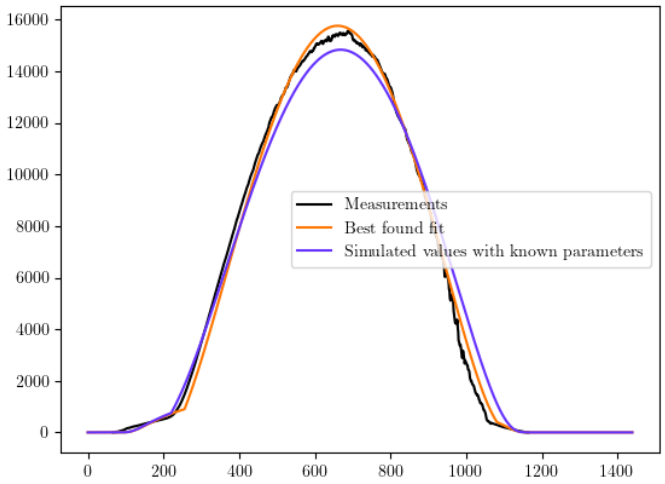


Figure 5.18: Comparison for FMI Kuopio installation. Panel angles are 15° and 217° degrees. Best found fit at 19.34°, 136.87°

5.6.2 Possible issues and further development ideas

The algorithm steps of cloud free day detection and lattice generation are computationally fast but evaluating a high density lattice for multiple days can take several minutes. For example, the time required for the generation of 5.12 consist of 9 seconds of data loading and preprocessing, 1 second of cloud free day detection and 11 minutes of angle pair evaluation. With 22 days and 1000 points per day, the resulting 22 000 evaluations were done at the average speed of 33 evaluations per second. This comparably long processing time is due to inefficient code but the algorithms speed can also be improved by more intelligent latticing. The general location of the best fit could be solved with a low density lattice and a local high density lattice could then be used to estimate a higher accuracy angle pair. Combining code optimizations and localized angle space lattices could reduce the computation time significantly.

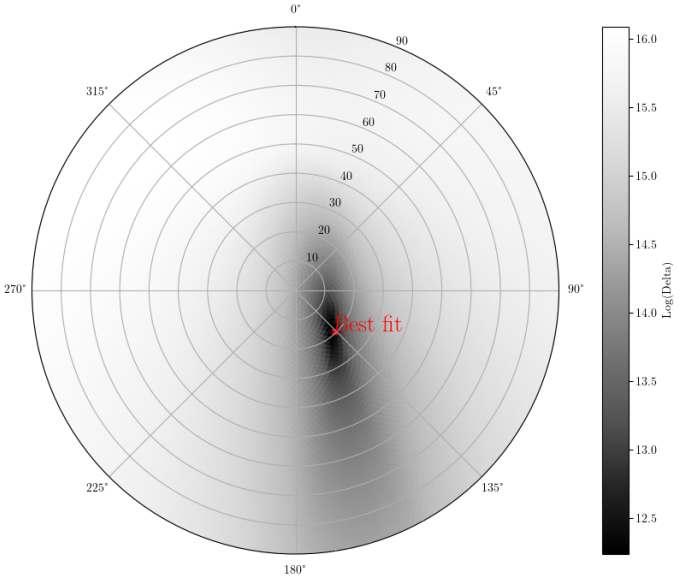


Figure 5.19: Fitness heatmap.

Chapter VI

Conclusion

The initial goal of this thesis was to find a simple way of estimating parameters of solar PV installations and this goal has been accomplished with moderate success. Some of the algorithms are thousands of lines of long, but the underlying mathematics was still kept simple. As a result, the code can be understood and modified by a wider audience. This is in particular contrast with AI and machine learning based approaches which often provide good results but which tend to be less insightful.

From the perspectives of mathematics and programming, model fitting problems are not particularly difficult. In this thesis, the challenges rose from optimization, understanding patterns in the data and discovering where the limits of the estimation algorithms come from. The insights gained while tackling these issues may be more valuable to other researchers than the final estimation algorithms. The most significant of which are the center angle error function, Fibonacci-lattice based angle space discretization and angle space resolution estimates. These were not mentioned in the cited literature and while the are most likely already used in other fields, they would most likely prove to be useful for similar studies conducted in the future.

While experimenting with the datasets, some interesting traits and phenomena were observed, some of which could warrant their own studies. For example, the figure 5.18 shows that the last hours of the specific day are noisy. This noise may play a significant role in the prediction erros and thus further studies in detection and classification of noise types in solar PV datsets would prove to be useful for all who perform analysis on solar PV installations. Perhaps the largest apparent obstacle in noise detection and classification studies is the temporal resolution as noise profiles of clouds, shadowing structures or temperature fluctuations may prove

to be impossible to detect accurately at low temporal resolutions.

Lastly I would like to encourage other researchers to publish their research and code openly. During preminary research and literature reviews, the code examples or datasets which were used for research papers did not seem to be available. While research code may not be useful as is,

open research should be encouraged. During preminary research and literature reviews, there did not seem to be

Lastly the state of open source research is somewhat concerning. Solar energy plays a significant role in green energy transition and

.1 Code samples

The algorithms presented in this thesis are not very useful or easy to understand when presented with mathematical notations. The following listings contain code samples from github repository [**?**] where the complete source code is published.

.1.1 Cloud free day selection algorithm

The three following code listings are used by the cloud free day selection algorithm. The first listing __find_smooth_days() returns the list of day numbers and data from days which can be considered smooth. The second is a helper function used for calculating a normalized smoothness value for a given day with the help of Fourier transform based low pass filtering and the last listing is the FFT based low pass filter.

Listing 1: Cloud free day finder main function

```
def __find_smooth_days(year_xa, day_start, day_end, threshold_percent):
    """
    :param year_xa: xarray of one year
    :param day_start: first day to consider
    :param day_end: last day to consider
    :param threshold_percent: smoothness percent, very best days for helsinki dataset have a
        smoothness value lower than 0.4. 1 seems to result in good values
    :return: list of xa days and a list of day numbers
    """

    # reading year from year_xa
    # if year_xa contains multiple years worth of data, the first will be chosen. Will most
        likely result in errors
    year = year_xa.year.values[0]

    smooth_days_xa = []
    smooth_days_numbers = []

    """
    The loop below goes through every day in given range from year of data
    If the range contains "bad days", this could cause issues. For example a day with zero
        power for every minute would be perfectly smooth, but at the same time it's the
        opposite of what we want
```

```
    """
    for day_number in range(day_start, day_end):
        day_xa = splitters.slice_xa(year_xa, year, year, day_number, day_number)

        smoothness_value = __day_smoothness_value(day_xa)


        if smoothness_value < threshold_percent:
            smooth_days_xa.append(day_xa)
            smooth_days_numbers.append(day_number)


    return smooth_days_xa, smooth_days_numbers
```

Listing 2: Cloud free day day smoothness function

```
def __day_smoothness_value(day_xa):
    """
    :param day_xa: one day of real measurement data in xarray format, has to have fields
        minute and power
    :return: percent value which tells how much longer the distance from point to point is
        compared to sine/cosine
    fitted curve. Values lower than 1 can be considered good. Returns infinity if too few
        values in day
    """

    # no values at all, returning infinity
    if len(day_xa["power"].values[0]) == 0:
        return math.inf

    day_xa = day_xa.dropna(dim="minute")

    # extracting x and y values
    minutes = day_xa["minute"].values
    powers = day_xa["power"].values[0][0]

    # too few values, returning inf
    if len(powers) < 10:
        return math.inf
```

```
    # transforming powers into fourier series and low pass filtering
    powers_from_fourier_clean = __fourier_filter(powers, 6)

    # this normalizes error in respect to value count, single value
    errors = abs(powers_from_fourier_clean - powers)
    errors_sum = sum(errors)
    errors_normalized = errors_sum / len(powers)

    # if max of powers is 0.0, then division by 0.0 raises errors. If we check max for 0.0
        and return infinity
    # our other algorithm should disregard this day completely
    if max(powers) == 0.0:
        return math.inf
    # normalizing in respect to max value and turning into percents
    errors_normalized = (errors_normalized / max(powers)) * 100
    # this line may cause errors due to division max power 0.0

    return errors_normalized
```

Listing 3: Cloud free day finder low pass filter

```
def __fourier_filter(values, values_from_ends):
    """
    :param values: array of values
    :param values_from_ends: how many of the longest frequencies to spare
    :return: values after shorter frequencies are removed
    """


    # FFT based low pass filter
    # Converting values to Fourier transform frequency representatives
    values_fft = numpy.fft.fft(values)
    # values in values_fft represent the frequencies which make up the values array.
        Structure is as follows:
    # [constant, low, low, ... med, med .... high, high .... med, med .... low,low]
    # this means that by zeroing out most of the values in the center, only the low frequency
        parts can be chosen.


    # zeroing out every value which is further than [values_from_ends] from the ends of the
```

```
        values_fft  array
    values_fft [1+values_from_ends:len( values_fft ) - values_from_ends] = [0] *
        (len( values_fft )-1 - 2 * values_from_ends)


    # reversing the fft operation, resulting in values with only low frequency components
    values_ifft  = numpy.fft. ifft ( values_fft )

    # ifft results can be partly imaginary, eq. 2.5 + 2i.
    values_ifft_real  = []

    # saving only real components
    for var in values_ifft :
        values_ifft_real .append(var.real)

    # returning the result of the low pass filter

    return values_ifft_real
```

.1.2 Angular distance equation

This sample contains code used for computing the angular distance between two points on unit sphere surface. Used in panel installation angle error measurements.

Listing 4: Angular distance function

```
def angular_distance_between_points(tilt1 , azimuth1, tilt2 , azimuth2):
    """
    Calculates the angular distance in degrees between two points on unit sphere surface.
    :param tilt1: point 1 tilt angle in degrees
    :param azimuth1: point 1 azimuth angle in degrees
    :param tilt2: point 2 tilt angle in degrees
    :param azimuth2: point 2 azimuth angle in degrees
    :return: sphere center angle between the two points
    """
    tilt1_rad  = numpy.radians(tilt1)
    azimuth1_rad = numpy.radians(azimuth1)
    tilt2_rad  = numpy.radians(tilt2)
    azimuth2_rad = numpy.radians(azimuth2)

    x1 = math.sin(tilt1_rad) * math.cos(azimuth1_rad)
```

```
    y1 = math.sin(tilt1_rad) * math.sin(azimuth1_rad)
    z1 = math.cos(tilt1_rad)

    x2 = math.sin(tilt2_rad) * math.cos(azimuth2_rad)
    y2 = math.sin(tilt2_rad) * math.sin(azimuth2_rad)
    z2 = math.cos(tilt2_rad)


    euclidean_distance = math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2)

    center_angle = numpy.degrees(math.acos((2 - euclidean_distance ** 2) / 2))

    return center_angle
```

[1] N. Haghdadi, J. Copper, A. Bruce, and I. MacGill, "A method to estimate the location and orientation of distributed photovoltaic systems from their generation output data," *Renew. Energy* **108**, 390–400 (2017).

[2] M. K. Williams, S. L. Kerrigan, and A. Thornton, "Automatic detection of PV system configuration," *Proceedings of World Renewable Energy Forum* (2012).

[3] H. Böök., Poikonen, A., Aarva, A., Mielonen, T., Pitkänen, M.R.A., Lindfors, and A.V., "Photovoltaic system modeling: A validation study at high latitudes with implementation of a novel DNI quality control method.," *Sol Energy* **204**, 316–329 (2020).

[4] R. Perez, P. Ineichen, R. Seals, J. Michalsky, and R. Stewart, "Modeling daylight availability and irradiance components from direct and global irradiance," *Solar Energy* **44**, 271–289 (1990).

[5] S. N. Laboratories, "POA Sky Diffuse," (**!!YEAR!!**).

[6] S. N. Laboratories, "POA Beam," (**!!YEAR!!**).

[7] S. N. Laboratories, "POA Ground Reflected," (**!!YEAR!!**).

[8] dunno lol, https://ysjournal.com/astrophysics/analemma-the-suns-journey-in-the-sky/ (visited on 2023-11-12).