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

Timo Salola

MSc Thesis

May 2024

Department of Physics and Mathematics

University of Eastern Finland

Preface

As someone whose interests and strengths 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, snow and temperature on solar PV installation data. And Luna for being a supportive cat.

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

Helsinki, the 20th of May 2024 *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 determine the geographic location and panel installation angles of photovoltaic solar power installations using only the power output data. The chosen method breaks down 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 existence of such algorithms 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 latitude predictions is higher at less than 3.5° with the best result in case study 2-2 with a 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.

The data in the two datasets follows the structure seen in table 2.2. This snapshot from the Helsinki dataset shows that the temporal resolution is one measurement per minute and that there are four power values for each minute. Power values in

columns String 1 and String 2 represent power output from two identical sets of solar PV panels, one of which is shown in 2.1. Electricity generated by these sets of solar panels is fed into the inverter and thus the inverter input should match the sum of String 1 and String 2. The inverter then increases the voltage and changes the direct current to alternating current. Losses in the process should result in an inverter output value which is lower than the inverter input, but this does not appear to be true for every dataframe column. A plausible cause is the input/output measuring method which may result in noise in either input or output sides of the inverter.

Datasets from other sources may have different temporal resolutions, use different units or measuring points. From the consumer point of view, the inverter output is the useful power value and so this will be the power value used by algorithms in this thesis. Temporal resolution of the FMI data will be used as is.

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].

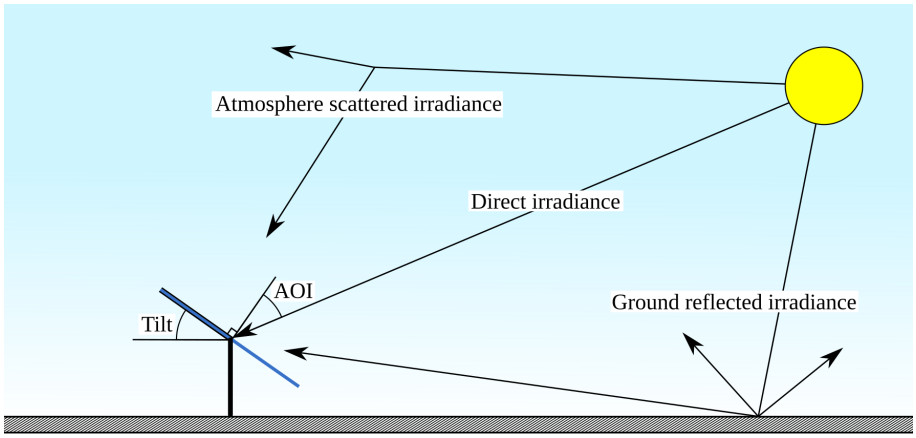


Figure 2.4: Simplified installation diagram for a PV system in ideal conditions.

Figure 2.4 shows a two dimensional simplification of a solar PV installation. The 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 visible structures in the data. The clearest structure in the 3D plot is the pattern formed by the first and last non-zero power minutes and this is later used for geolocation estimation. The second pattern is the dome-like shape of the point cloud. This shape can be examined by taking one day slices from the dataset and plotting them individually as shown in 2.6. These slices are used for panel installation angle estimation.

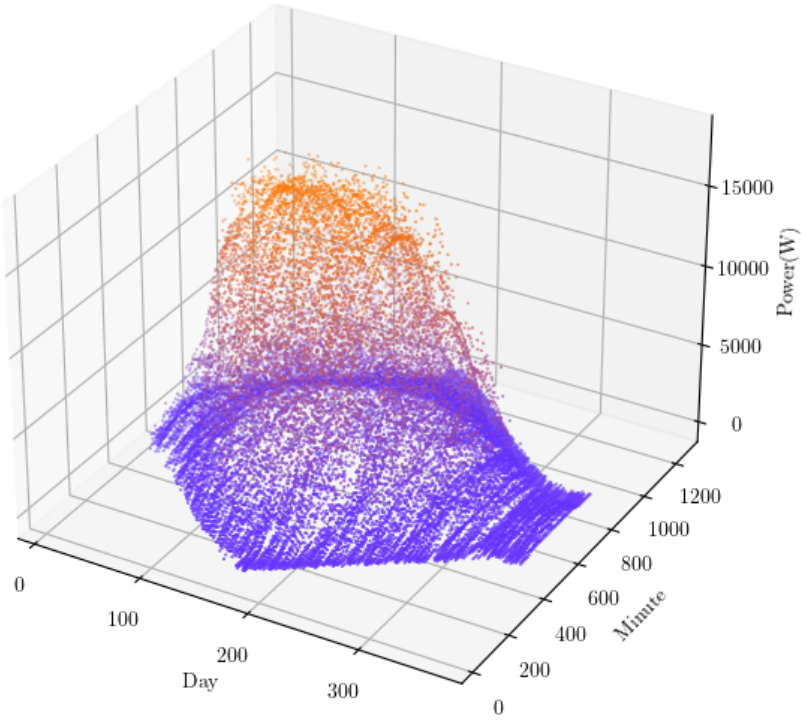


Figure 2.5: One year of data from FMI Kumpula installation as a 3D point cloud.

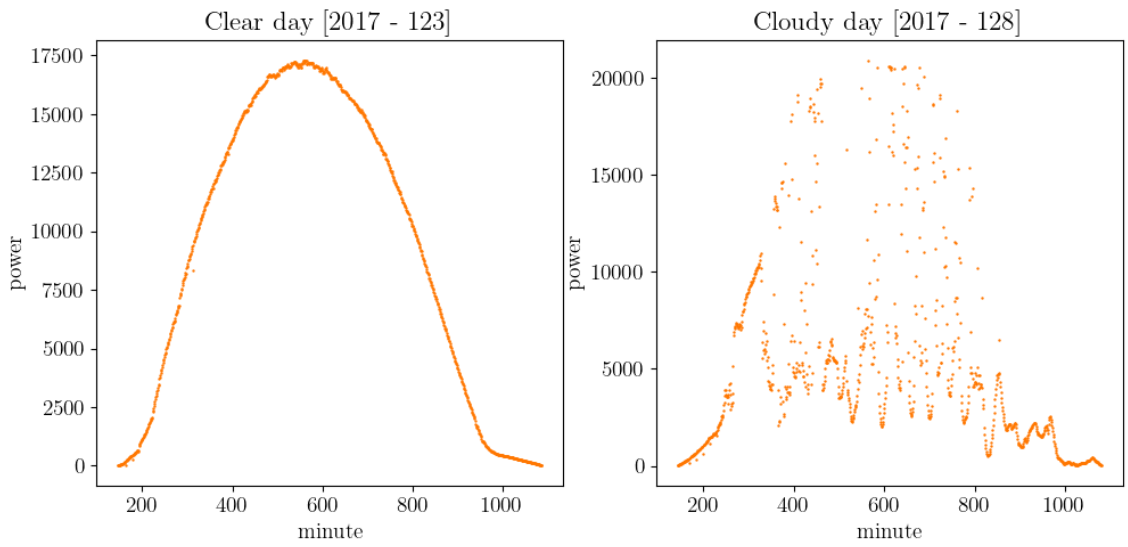


Figure 2.6: Two days from FMI Kumpula dataset with different charasteristics.

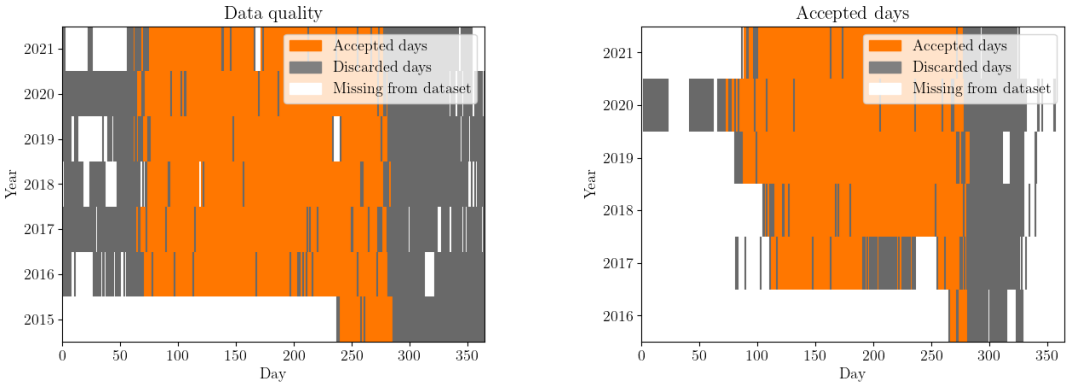
The presumably cloud free day 2017-123 in 2.6 has some notable charasteristics in addition to the geolocation derived first and last minutes. The shape resembles a right skewed normal distribution with a knee section near 950 minutes. As the absorbed irradiance of a PV system consists of direct and indirect irradiance, this knee section is likely to signify the point in time at which angle of incidence reaches 90 degrees.

Another measurable trait is the peak power generation minute. Geometric intuition would suggest that this should align with the moment in time with the lowest angle of incidence but this relationship could be fairly complex due to temperature induced efficiency variation and atmopshere induced losses which are higher when the Sun elevation is low.

2.2 Data pre-processing

The data pre-processing required by the algorithms in this thesis can be split into two gategories, classifying preprocessing and reparing preprocessing. 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 gaps measurement data 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 contains a comparison on which days in the datasets met the requirements.



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

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

Figure 2.7: Requirements were 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 classification has discarded days which did not meet the set requirements, the next step is data repairing. The accepted days still contain a small amount of missing measurements and Nan values which can be linearly interpolated, mean-

ing that if a power measurement or a set of measurements is missing between two known datapoints, the missing datapoints are estimated to describe a linear transition breaching the gap between the known datapoints. When noise 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 previous preprocessing steps have filtered and repaired days according to measurement counts and data gaps, these algorithms did not filter days based on the amplitude noise present in power measurements. This power noise is typically induced by clouds and the resulting power generation curves can not be reliably used for curve fitting. The following algorithm steps can be used for automating the process of selecting days with minimal high-frequency fluctuations.

1. Split dataset into individual days based on utc timestamps.
2. Create a copy of the power measurements for each day and process this copy with a low pass filter algorithm.
3. Calculate the difference between the original power values and the filtered power values. This delta value increases when high-frequency noise is present.
4. Discard days with a delta value higher than a set threshold.

The mathematically non-trivial parts here are 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 that removes frequencies higher than a given limit from a signal, allowing lower frequencies to pass.

Here the filtering is done with discrete Fourier transformations (DFT) and inverse discrete Fourier transformations (IDFT). When a list of numbers is used as the input of DFT, the output is a list of ordered complex numbers, each of which represents a sine wave of a certain frequency, phase and amplitude. The sum of these wave equations forms 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 numerical lists, low-pass filtering being one of the possibilities. In this case the 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 for PV power measurements, 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 as per 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 final component is threshold selection. The intermittent 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% nor-

malized noise.

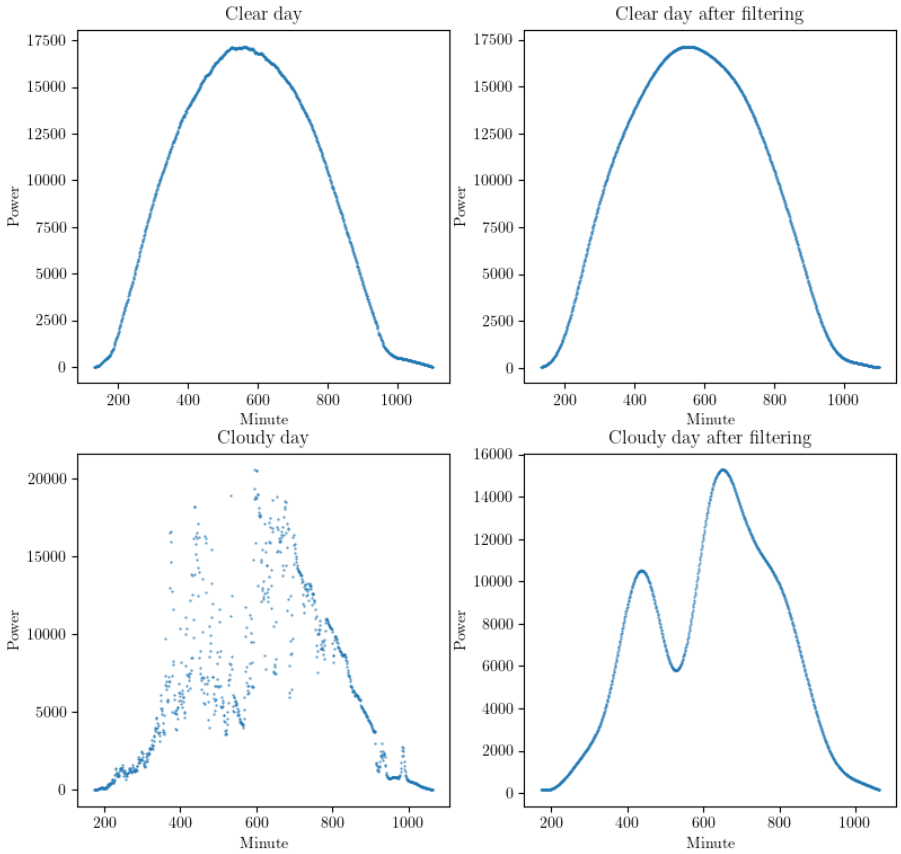


Figure 2.8: Cloud free day finder low pass filtering phase.

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 PVlib POA function inputs

The following listing contains the relevant 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 as datasets include timestamps and this leaves four unknown system parameters. These four parameters span a 4D parameter space of possible PV installations. The size of this parameter space is connected to the difficulty of the parameter estimation problem. As the parameters are in $\mathbb{R}$, the amount of sensible combinations is the product of the discretization of each of the parameter ranges. Tilt and azimuth parameters can be discretized as integers, resulting in $90 * 360$ or 32400 unique angle space points. The geographic latitude and longitude coordinates are somewhat harder to discretize. In an arbitrary 0.1 degree geographic discretization there would be $1800 * 3600$ or 6480000 coordinate combinations. The product of these two sums results in $2 * 10^{11}$ unique parameter combinations.

A parameter space this large is difficult to examine exhaustively. Even a system capable of evaluating 100 000 angle space points per second would need approximately 23 days in order to evaluate all of the possible combinations. However if the parameters can be solved in isolation from oneanother, the required computational time would decrease significantly. Instead of $90 * 360 * 1800 * 36000$, there would be $90 + 360 + 1800 + 3600$ combinations to evaluate. This highlights how important it is to break problems into smaller pieces whenever possible and much of this thesis focuses on how this can be done with solar PV parameter estimation.

3.1 PVlib POA evaluation

Before implementing the parameter estimation functions, the POA simulations should be tested against known measurement data. Figure 3.2 displays that in clear sky conditions the pvlib irradiance model is following real world measurements closely with a few exceptions. And during cloudy days the measurements are often lower than the clear sky model would indicate, but they can also peak higher than they would during cloud free days. These increases of power generation above clear sky estimated power are 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.

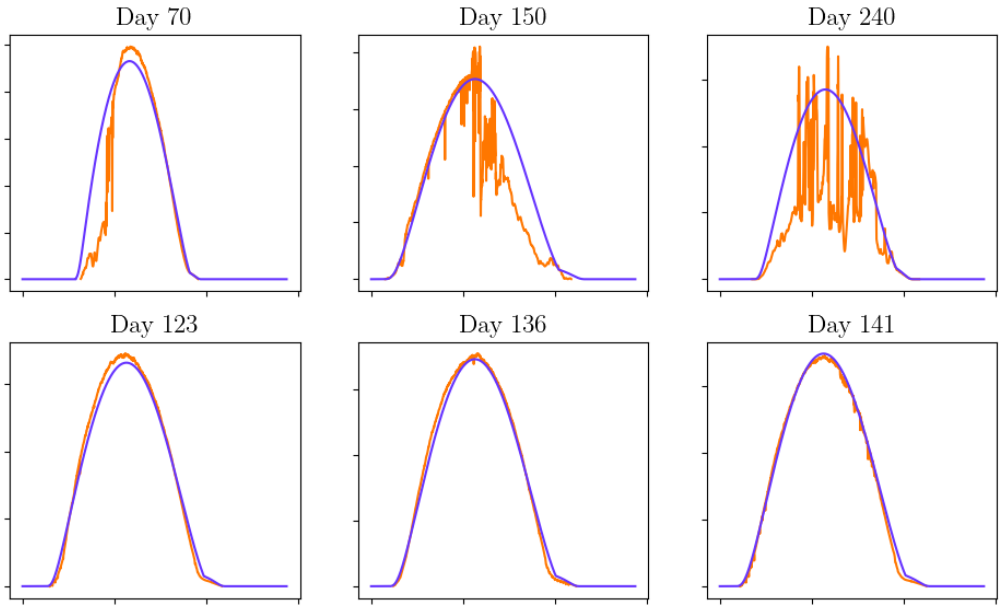


Figure 3.2: Power output of FMI Kumpula PV installation and the pvlib POA simulation computed with the parameters 2.3. Horizontal axis on the graphs corresponds to time and vertical axis marks the estimated power values. The purpose of the graphs is to display the different shapes and deviations from 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.

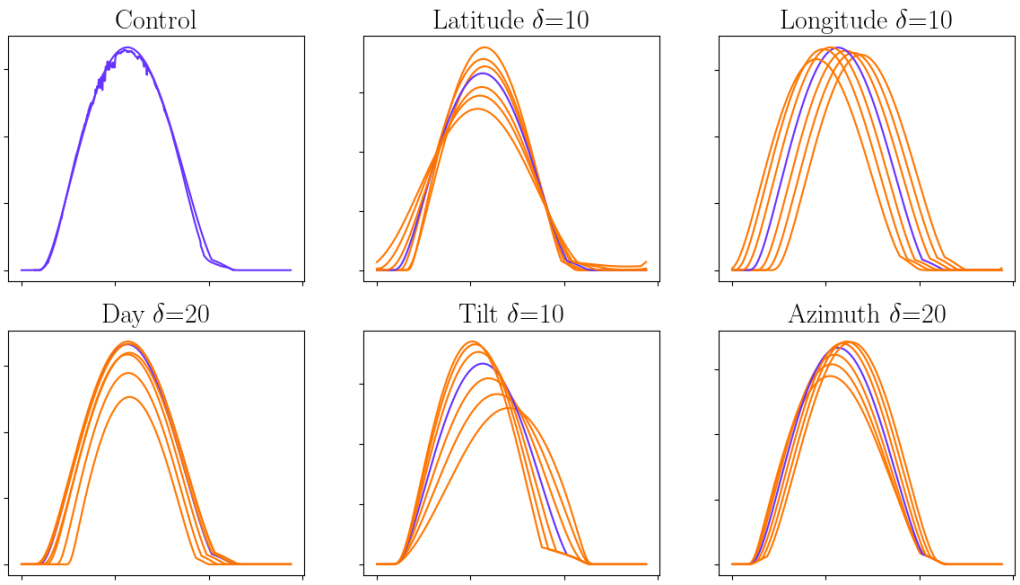


Figure 3.3: Influence of changes in PVlib simulation parameters on generated power output curves. Control shows FMI Helsinki measurements and simulation with the same parameters as the Helsinki installation. Simulated power values are multiplied by 19 in order to match values on y-axis.

By varying the different simulation parameters as shown in figure 3.3, 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 would appear to be 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 3.4.

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

Table 3.4: Function input to observed trait table.

3.1.1 Influence of different longitudes

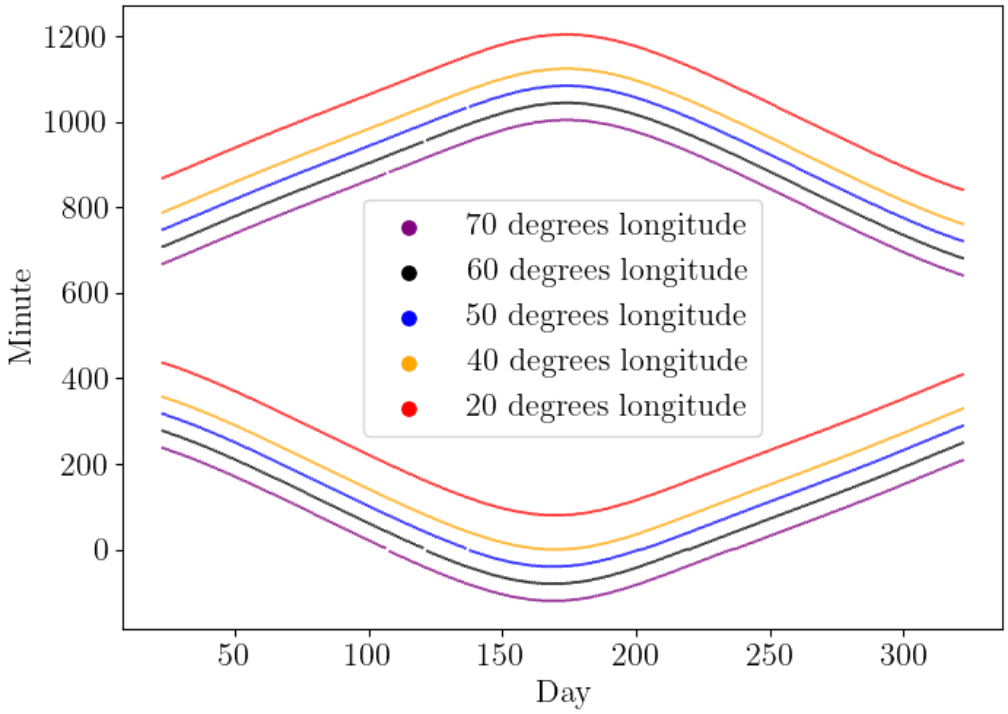


Figure 3.5: First and last non-zero minutes of each day from year long simulations at different longitudes.

Based on earlier observations listed in table 3.4, 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 3.5. 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

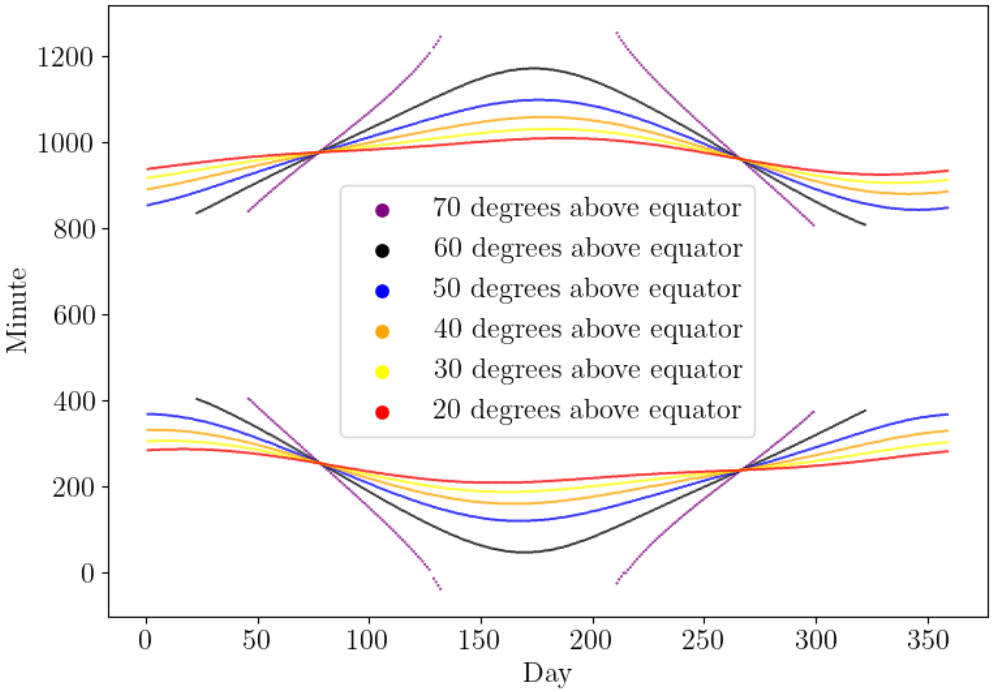


Figure 3.6: First and last non-zero power minutes of each day from year long simulations at different latitudes

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

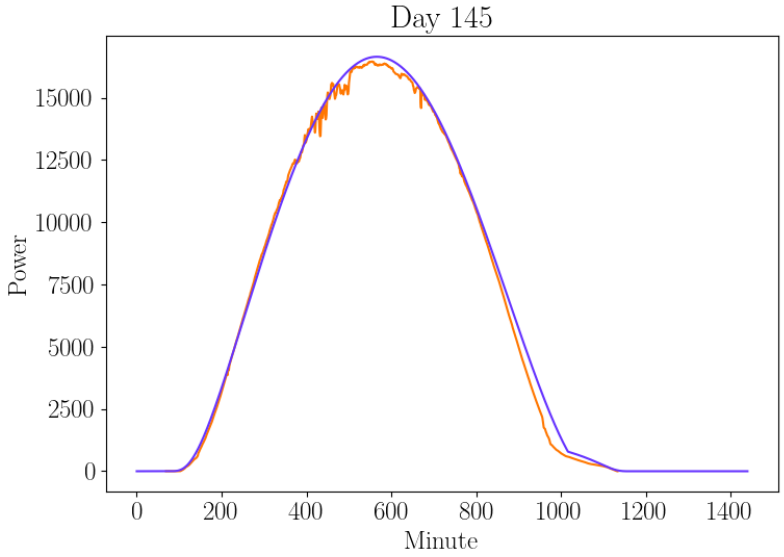


Figure 3.7: Figure comparing Pvlib simulated POA irradiance and actual measured PV output of FMI Helsinki installation.

Figure 3.7 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

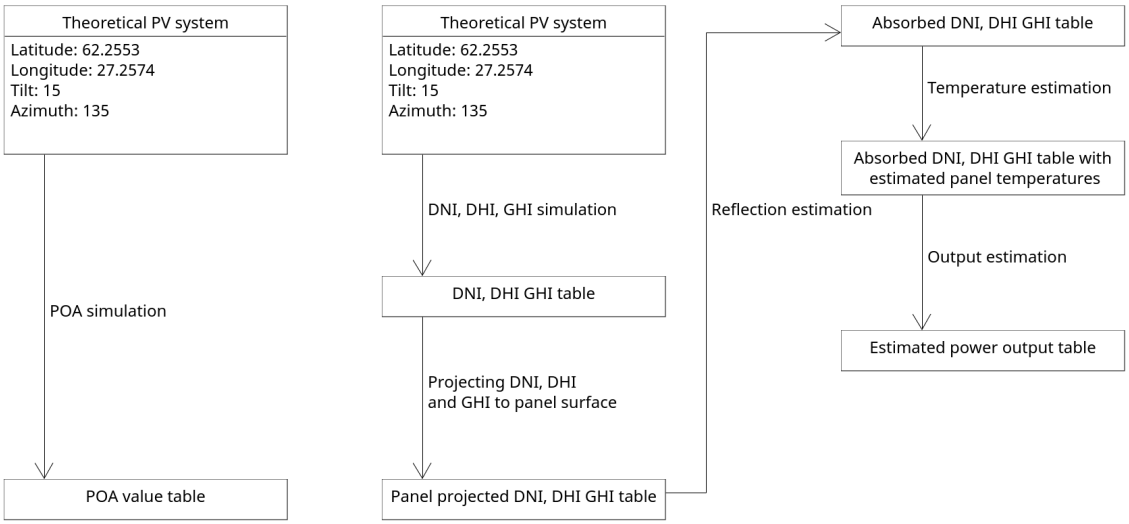


Figure 3.8: Original POA-based PV model and the improved model with reflection and temperature estimation.

The improved PV model introduces some new concepts, among 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 DHI 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]

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

Where AOI is the angle of incidence.

GHI projection [7]

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

Where $albedo = 0.151$. This represents ground reflectivity near the solar PV installation. The value varies significantly depending on the area, season and time of day. A fixed value is used as terrain, weather and snow cover are assumed to be unknown.

$tilt$ is the tilt angle of the PV installation, this is 15° for both FMI installations.

3.2.3 Reflection estimation and absorbed irradiance

The following Solar irradiance reflection equations originate from 2001 Martin and Ruiz paper [8]. This study used a physical contraption in which solar panels were rotated under a light beam from a solar simulator. Values from the reflective losses equations represent the fraction of irradiance reflected away from the panels e.q. a $DNI_{reflected}$ value of 0.24 would tell us that 24% of irradiance is lost due to reflections and 76% is absorbed.

DNI reflective lossess

$$DNI_{reflected} = exp(-\frac{1}{a_r}(c_1 p_1 + c_2 p_1^2)) \tag{3.3}$$

DHI reflective lossess

$$DHI_{reflected} = exp(-\frac{1}{a_r}(c_1 p_2 + c_2 p_2^2)) \tag{3.4}$$

GHI reflective lossess

$$GHI_{reflected} = \frac{exp(-cos(\alpha)/a_r) - exp(-1/a_r)}{1 - exp(-1/\alpha_r)} \tag{3.5}$$

Where

$a_r = 0.159$ Empirical reflectance constant for polycrystalline silicon solar PV panels.

$c_1 = \frac{4}{3\pi}$ Fitting parameter 1.

$c_2 = -0.074$ Fitting parameter 2.

$p_1 = sin(tilt) + \frac{tilt - sin(tilt)}{1 - cos(tilt)}$

$p_2 = sin(tilt) + \frac{\pi - tilt - sin(tilt)}{1 + cos(tilt)}$

Absorbed irradiance

$$POA_{absorbed} = DNI(1 - DNI_r) + DHI(1 - DHI_r) + GHI(1 - GHI_r) \tag{3.6}$$

Where DNI, DHI and GHI are solar irradiance values in watts and DNI_r, DHI_r, GHI_r refer to 3.3, 3.4 and 3.5.

3.2.4 Panel temperature estimation

Panel temperatures need to be estimated as the efficiency of solar panels depends on panel temperature. This can be accomplished with the following King et al 2004 model [9] if air temperature, wind speed and absorbed radiation are known. Note that as the model does not take the thermal capacity of the panels into account, the actual panel temperature is likely smoother and delayed compared to modeled temperature.

Panel temperature

$$T_{panel} = T_{air} + POA_{absorbed} e^{C_a + C_b * w} \tag{3.7}$$

Where

T_{air} is air temperature in $°C$.
$C_a = -3.47$ Model fitting constant.
$C_b = -0.0594$ Model fitting constant.
w is wind speed at panel elevation.

As air temperature and wind speed are unknown, dummy values have to be used for both variables. Air temperatures of $20°C$ and wind speed of 2m/s were used as dummy values.

3.2.5 Output estimation

Now that the absorbed radiation and panel temperature are known, the output can be estimated with the following Huld et al 2010 model [10].

PV output model

$$P_{output} = P_{rated} P_n \eta_{rel} \tag{3.8}$$

Where

P_{rated} is the rated power of the system in kW.

$P_n = \frac{POA_{absorbed}}{1000}$

$\eta_{rel} = 1 + k_1 ln(P_n) + k_2 ln(P_n)^2 + T_{diff} k_3 + k_4 ln(P_n) + k_5 ln(P_n)^2 + k_6 T_{diff}^2$

In η_{rel} the variables k_1 to k_6 are fitting parameters and T_{diff} is $T_{panel} - 25°C$.

k1	k2	k3	k4	k5	k6
-0.017162	-0.040289	-0.004681	0.000148	0.000169	0.000005

For unknown PV systems, the unknown P_{rated} value can be estimated by comparing the measured power output to simulated power output with a theoretical 1kW system. Visualization of this process with POA simulations and real measurements is shown in 3.9.

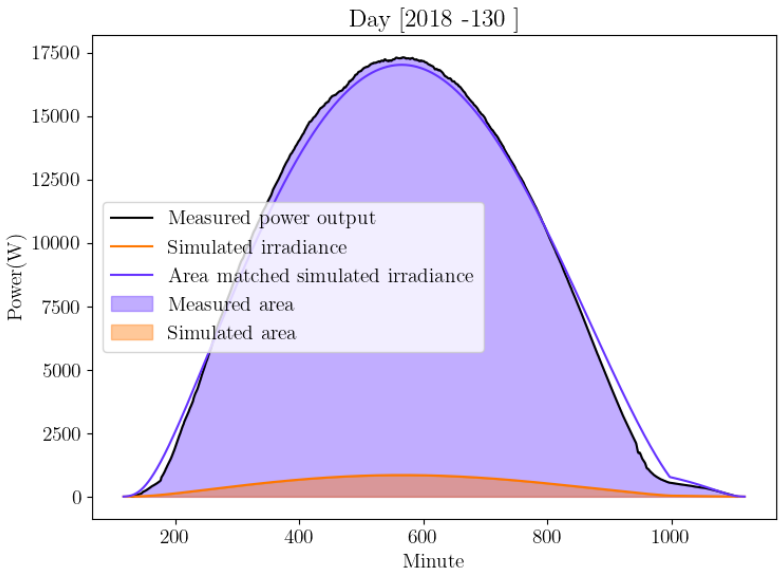


Figure 3.9: Measurement data from FMI Helsinki dataset, POA irradiance simulation was computed with FMI Kumpula coordinates and installation angle parameters.

3.2.6 Model improvement results

The models can be compared with the original measurement data with the following equation.

Model error

$$Delta_{model} = \sum_{t=1}^{1440} |P_{measured}(t) - P_{simulated}(t)|/1440 \tag{3.9}$$

The delta value represents a normalized per minute deviation between the model and the meausred data. Normalization by division with 1440 results in low delta values as this assumes that the system is constantly generating power regardless of the time of day and thus the resulting delta value is somewhat missleading but for comparison between models these values are still useful.

Delta values for detected cloud free days for FMI Helsinki installation are on average 244 for the POA model and 145 for the improved model based on a sample of 22 cloud free days. For FMI Kuopio the POA model achieved a delta of 487 and improved model 320 with 42 day dataset. This would indicate that the improved PV model is a better approximation for clear sky PV output than the PVlib POA model.

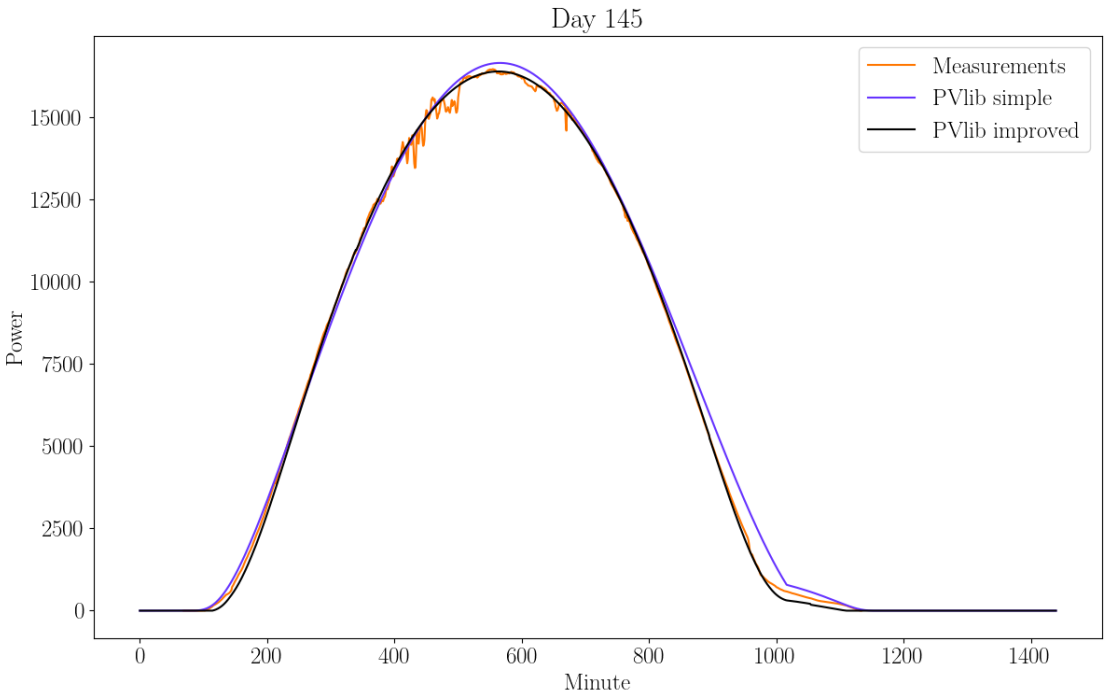


Figure 3.10: Comparison of PVlib POA and the improved PV power generation model on a day from Helsinki dataset.

Figure 3.10 shows what the improvement looks like in a best case scenario. The model can be seen to perform better during peak production hours and during the last productive hours. This 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 each use case in the following chapters.

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 are only approximations 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 two 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 $(\pm 15/1440) * 360^\circ = \pm 3.75^\circ$ 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.

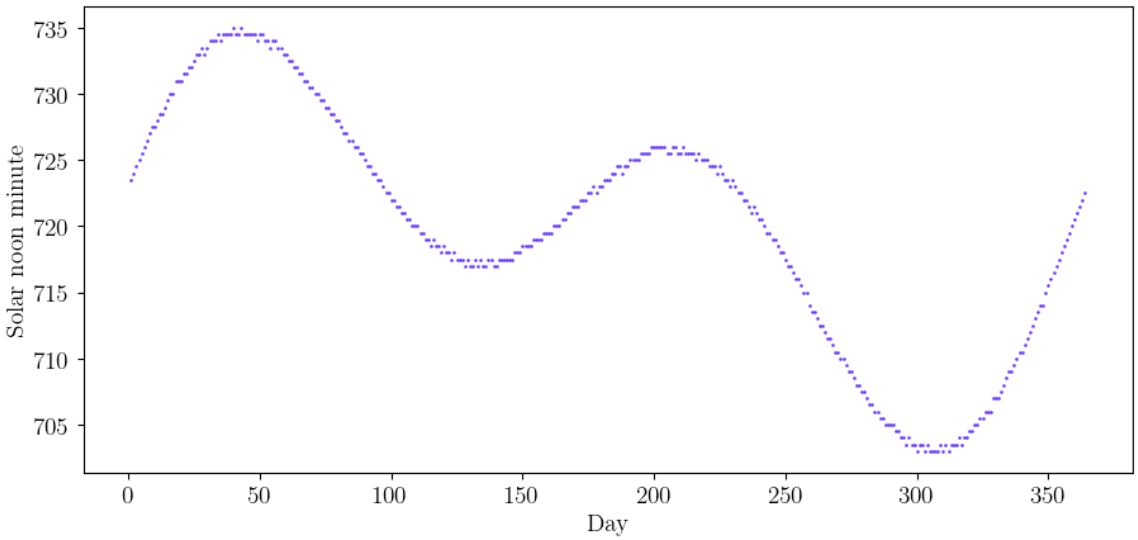


Figure 4.1: Approximations of solar noon minutes based on PVlib POA function at longitude 0° for year 2023. This pattern is caused by the difference in solar time and UTC time [11].

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° 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 solar time and utc time variation 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 inside the range 100th to 280th would seem best suited for first and last minute sensitive analysis algorithms for both Helsinki and Kuopio installations.

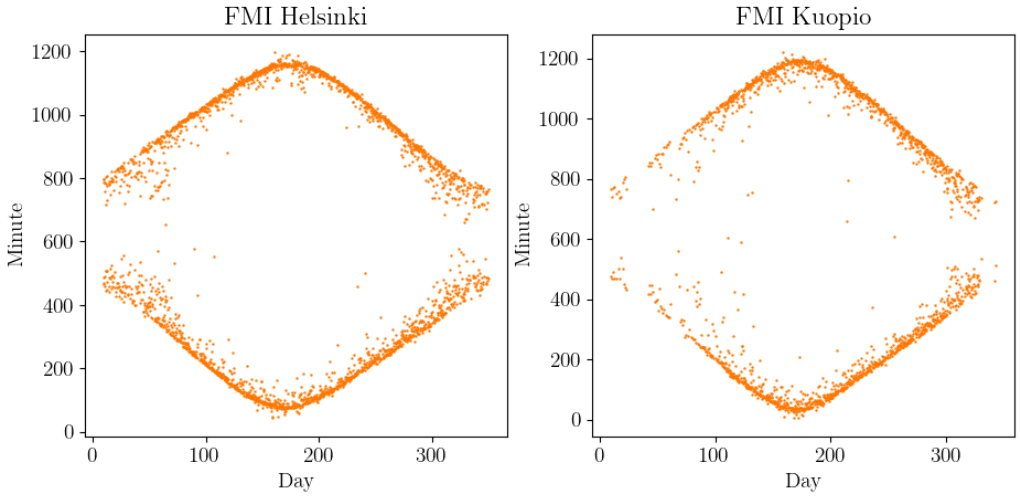


Figure 4.2: First and last non-zero power minutes of each day during years 2017 to 2021 from FMI Helsinki and Kuopio datsets.

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.

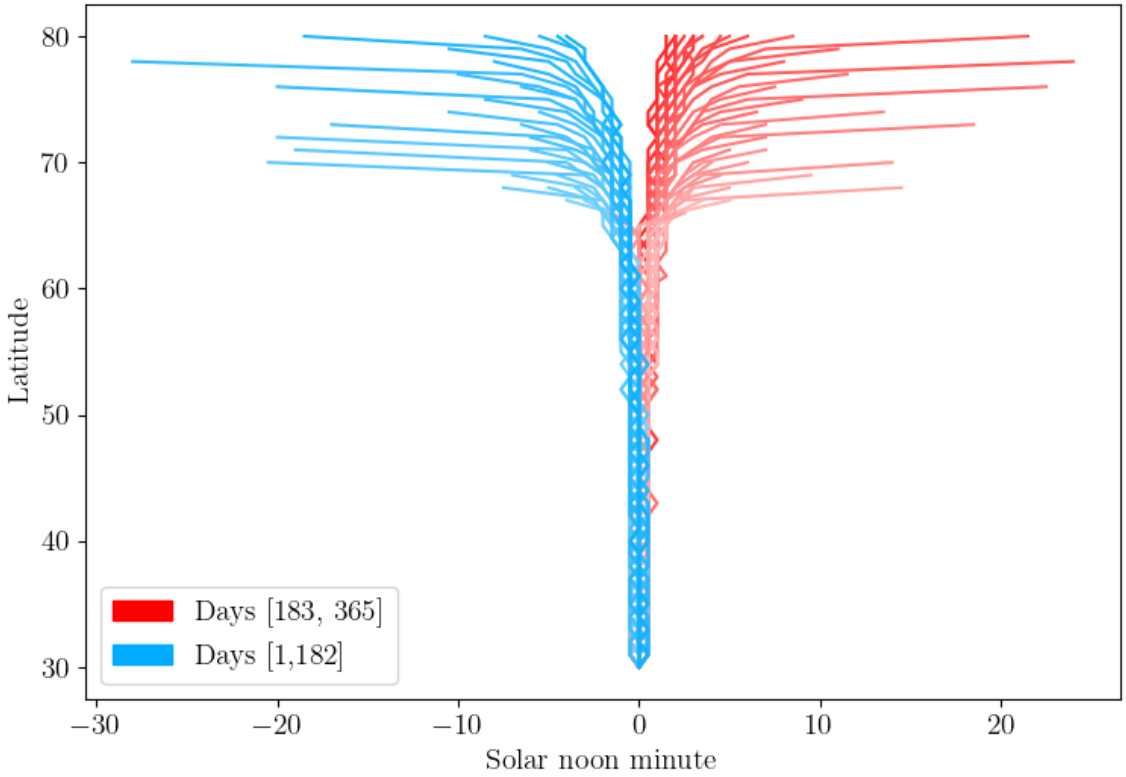


Figure 4.4: Relationship between latitude parameter and estimated solar noon time. Each line represents a different day of the year and x-axis values are normalized so that each line begins at 0 deviation. Lines with darker colors mark days which are further away from spring and fall equinoxes.

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 3.6, 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 slope 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 3.6. 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.

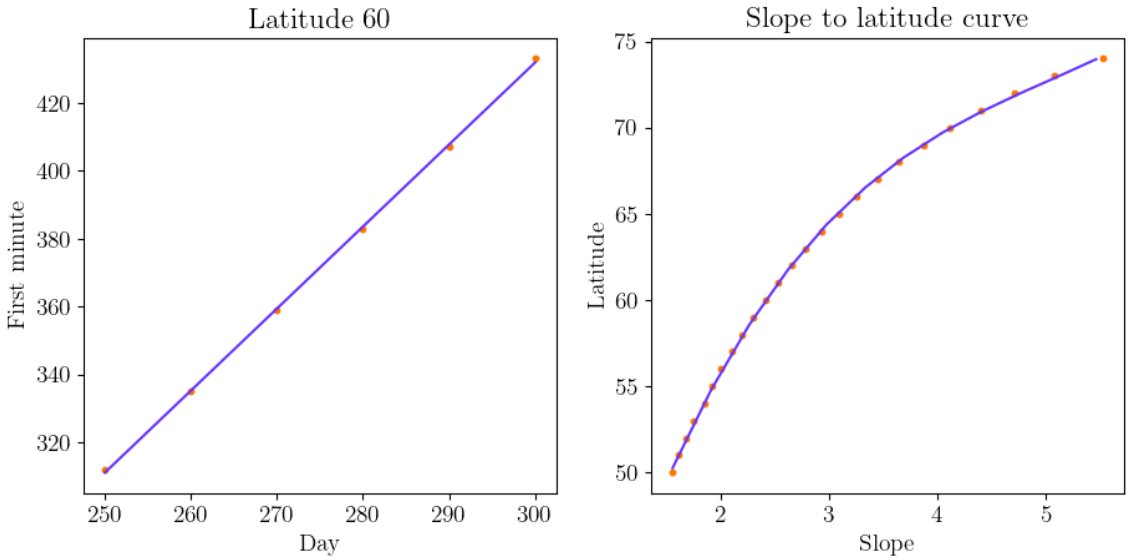


Figure 4.5: Left shows the almost linear relationship between day and simulated first minutes. Right shows the relationship between the slope angle and latitude.

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.

4.2.2 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.

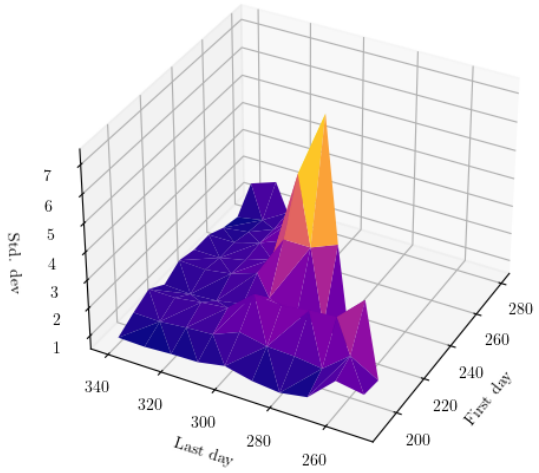


Figure 4.8: 3D -surface plot of day interval ranges and resulting standard deviations for FMI Kuopio dataset.

4.2.3 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.

4.2.4 Possible issues and further development ideas

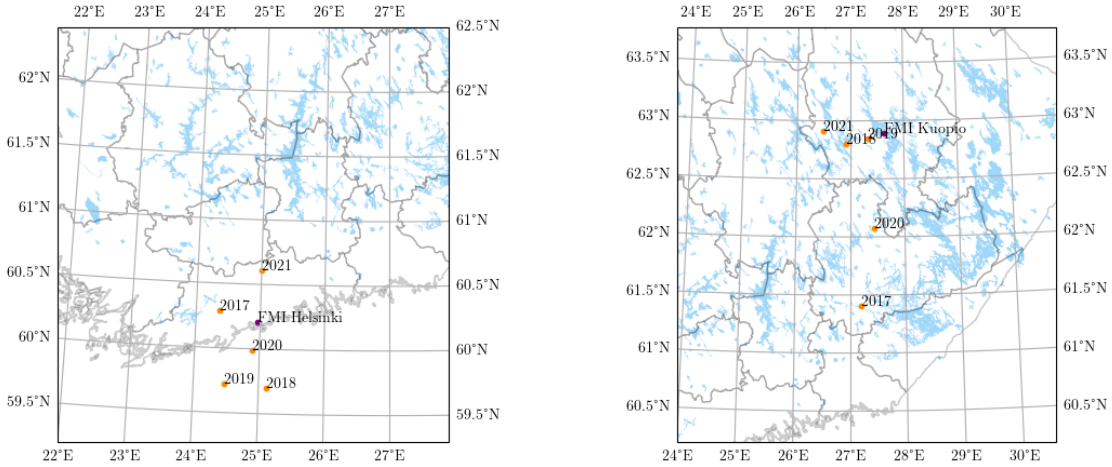
PVlib POA based first and last minute estimations are slower to compute than necessary as only two timestamps are needed. The use of a simpler sunrise and sunset equation would increase the speed significantly, allowing for the use of brute force day range selection algorithms.

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 at the risk of overfitting.

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 variance is strongest 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.

The results can be compared to the data resolution. One minute delta in measurements corresponds to a longtudinal shift of 14 kilometers in longitudinal axis at 60°. As the point cloud width is approximately 1° or 50km, the estimates can be thought to have a longitudinal range of 50km, 1° or 3.5 minutes with nearly the same

values for the Kuopio installation. As the temporal resolution of measurements is 1 minute, the algorithm should not be limited by temporal resolution.

Chapter V

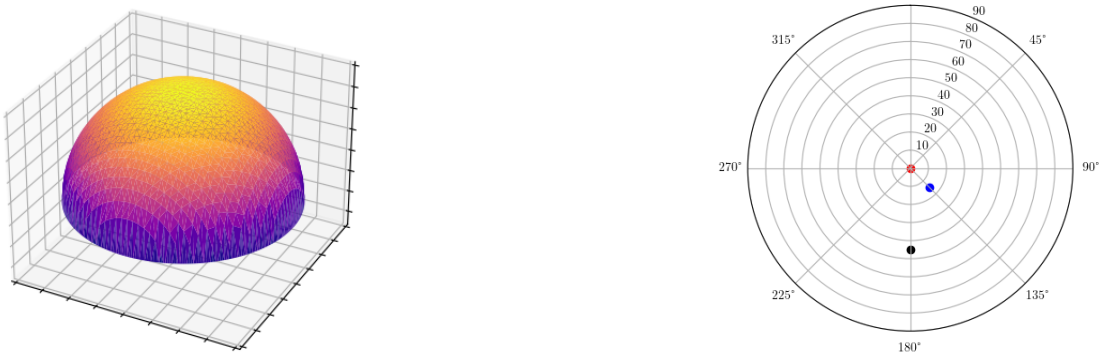
Estimating panel angles

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 or 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 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 fitness function for measuring the difference between simulated power values and measured power values.
- Angle space discretization function for discretizing the angle space into n discrete points which can then be tested with model fitness 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 [1] 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 fitness function

The earlier model error function defined in 3.9 can be re-used here as the simulation fitness function. By then computing multiple simulations with different panel angles and choosing the simulation with the best fitness, meaning lowest delta value, the panel angle values can be estimated.

Simulation fitness function

$$Delta_{model} = \sum_{t=1}^{1440} |P_{measured}(t) - P_{simulated}(t)|/1440 \tag{5.17}$$

As the function normalizes delta values by division with 1440, the delta for shorter days is lower than the delta for comparable long days. This should not matter for parameter estimation as the algorithm works by minimizing delta for each day independently.

Visualization of the function is shown in 5.2 where simulation with tilt of 90 degrees and azimuth of 135 is tested against known power measurements from a day in FMI Helsinki dataset, resulting in an average delta of 2959.5 W delta per minute.

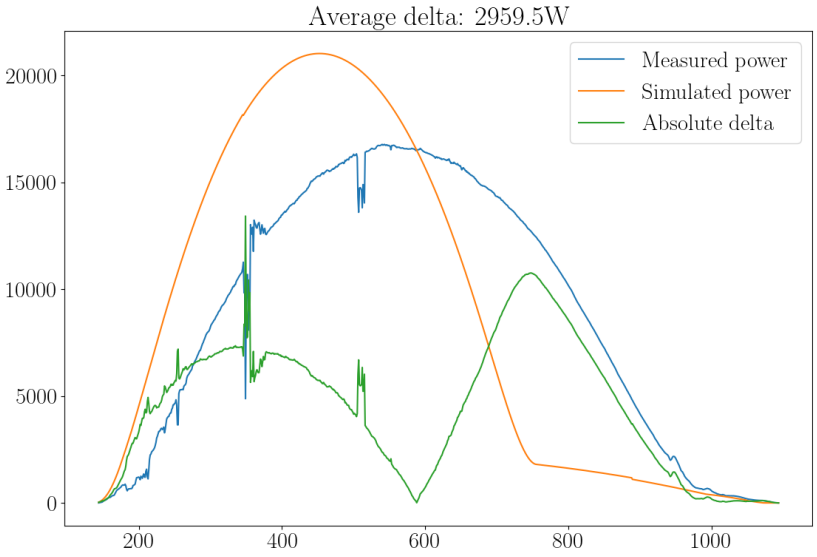


Figure 5.2: Visualization of fitness function.

5.3 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 flat plane but a point on a half-sphere surface, this results in an uneven discretization seen in figure 5.3a.

Sphere discretization problems are relevant for 3D graphics and real world problems involving geometry and so there are pre-existing methods available for discretization. One of the mathematically more elegant methods is the Fibonacci lattice which was used in a similar fashion in González 2009 [12]. The mathematical formulation of similar lattices is an older process and an earlier example is found in Vogel 1979 [13]. The following mathematical notation for the lattice is based on a code sample included in a blog post by Vagner Seibert [14].

Fibonacci lattice point n of k equation

$$s = n + 0.5 \tag{5.18}$$

$$\phi = acos(1 - 2s/k) \tag{5.19}$$

$$\theta = \pi s(1 + \sqrt{5}) \tag{5.20}$$

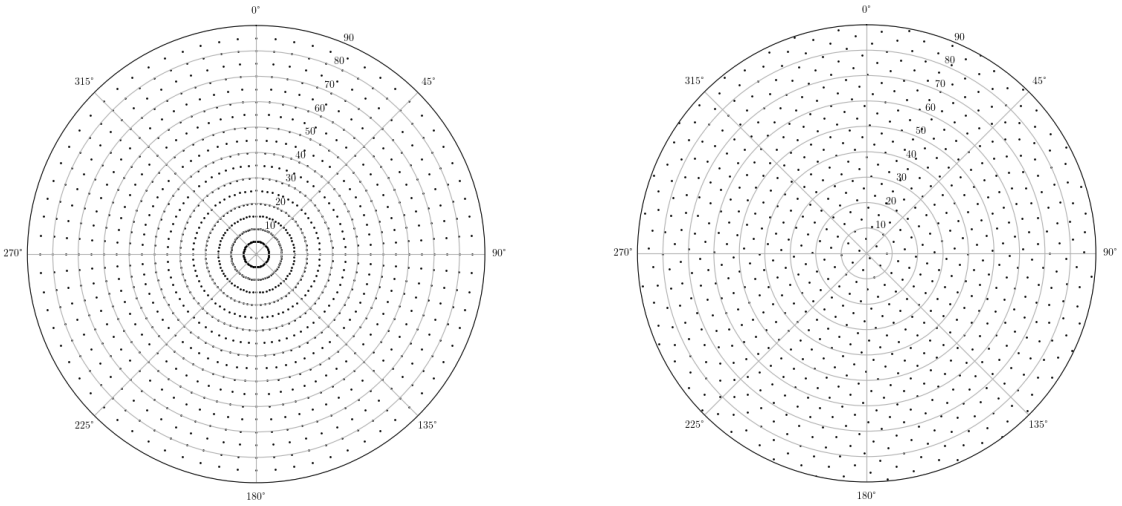
Where n is the point number, k is the amount of points, ϕ is the panel tilt angle and θ is the azimuth angle.

$$x = cos(\theta)sin(\phi) \tag{5.21}$$

$$y = sin(\theta)sin(\phi) \tag{5.22}$$

$$z = cos(\phi) \tag{5.23}$$

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 5.3: 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.

5.3.1 Importance of lattice density

Using the correct lattice density is important for using exhaustive search algorithms for panel angle estimation. Regardless of the lattice point count, a discrete lattice is unlikely to ever include the best fit in the whole $\mathbb{R}^2$ parameter subspace. This means that lattices of a sufficient density should be used in order to guarantee that a lattice point exists near to the optimal fit. However as incresing the lattice density increases the computational cost of angle estimation, choosing a good density becomes an optimization problem.

Lattice densities behave differently in different latticing patterns. With *in-steps-of-n* lattices, the main benefit is easy readibility. If the lattice is given one point for latitudinal and longitudinal degree, resulting in $360 * 90 = 32400$ points, then the lattice can be aligned so that each tilt and azimuth degree pair where angles are integers is tested. This discretization makes the results easily understanble as if the known installation angles are given as integers, a point representing the exact known installation angles already exists on the lattice.

With Fibonacci lattices and other lattice patterns in which lattice points may appear to be at seemingly random coordinates, evaluation of search algorithms is more difficult without visual aid. For example, in a 1000 point Fibonacci lattice the closest point to an arbitrary angle space coordinate is not obvious nor is the center angle distance between datapoints in the lattice as intuitive as with in-steps-of lattices.

5.4 Solving panel angles

Now that the geographic location and the multiplier values of installations are known to be solvable and fitness functions have been defined, the next step is solving the panel angles. The simple method is evaluating all points on a sufficiently dense lattice and choosing the point with the lowest delta value.

Figure 5.4 contains 10 Fibonacci lattice points and their normalized delta values. The best fit was at tilt 31.79°, azimuth 153.79° and it had a delta value of 369W. The lattice density leaves large gaps between lattice points and this found best fit is the closest point to the known installation angles of 15° and 135°. Center angle error as per 5.16 is 18.17°.

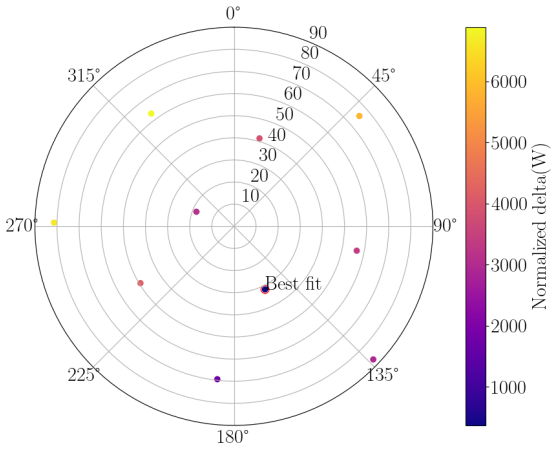


Figure 5.4: Polar plot of test points for a single day of data from FMI Kumpula dataset.

Tilt°	Azimuth°	Delta(W)
18.19	291.25	3128
31.79	**153.74**	**369**
41.41	16.23	3873
49.46	238.72	4183
56.63	101.22	3356
63.26	323.71	6888
69.51	186.2	1759
75.52	48.69	5814
81.37	271.18	6626
87.13	133.68	2938

Table 5.5: Tilt, azimuth and error table for single day.

The fit achieved in 5.4 is not very good and better results can be achieved by increasing the lattice density. The trivial method is to use a Fibonacci lattice with a higher point count, for example the fit achieved with a 10 000 point lattice 5.6 found the best fit at tilt 14.79°, azimuth 136.2°, delta value of 136.2W and center angle error of 0.3659 degrees°. This is a much better fit but increasing the lattice density comes with a higher computational cost. Depending on code optimizations, evaluating a single day from the dataset against a 10 000 point lattice can take up to several hours.

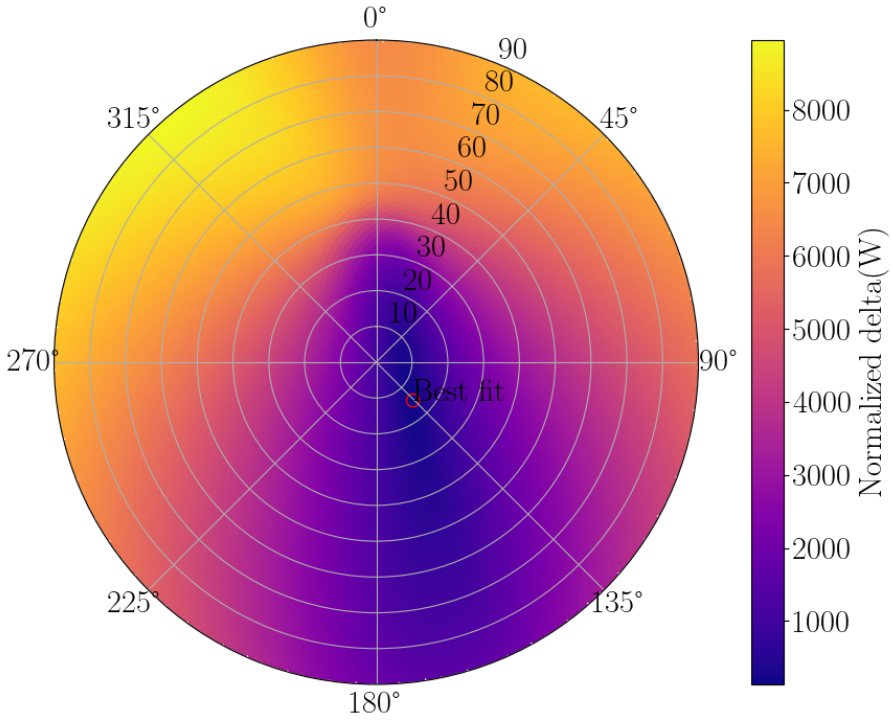


Figure 5.6: Results of a 10 000 datapoint lattice fitting against a single day from FMI Helsinki dataset. Center angle error between the found best fit and known installation angles is 0.3659°.

5.4.1 Results

The results of the exhaustive search algorithm are good for the Helsinki dataset. Angle delta values as low as achieved in 5.6 are likely to be irrelevant for analysis purposes and a delta of 0.37° is lower than the single digit precision of reported installation angles.

The next verification step is using the algorithm on multiple days from the same dataset and checking the spread pattern. This was done on sets of days with both Helsinki and Kuopio datasets. Figures 5.7 and 5.8 display the tight scatter patterns in the Helsinki predictions. For the tight grouping in the 2018 figure the average tilt of 13.89° and azimuth of 130.81° are close reported angles of 15° tilt and 135° azimuth. And with year 2019 where the scatter pattern is wider, the average of the results is 14.17° tilt, 132.62° azimuth which is even closer to the known installation angles. Fitness values or the per minute delta values are for the multi day 10 000

point evaluations were in the range of 76W to 117W for the Helsinki dataset.

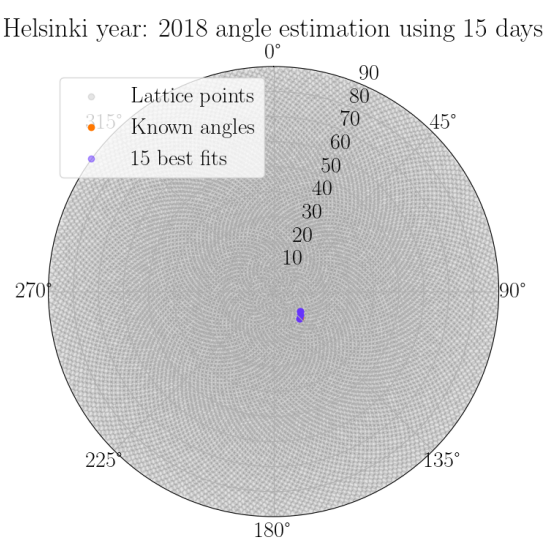


Figure 5.7: 10 000 point exhaustive search on 15 cloud free days from FMI Helsinki dataset using year 2018.

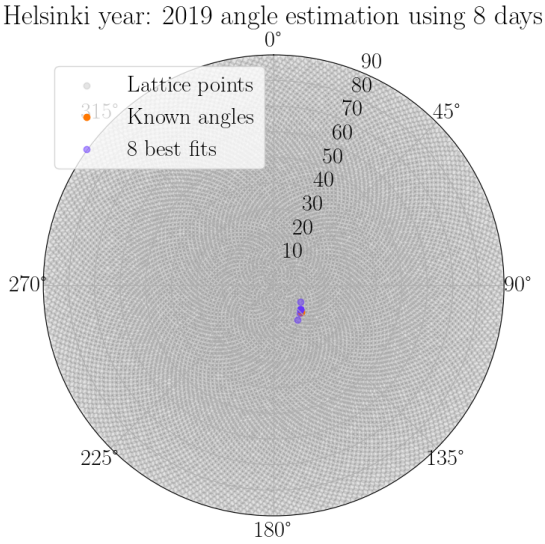


Figure 5.8: 10 000 point exhaustive search on 8 cloud free days from FMI Helsinki dataset using year 2019.

Results for the Kuopio dataset are not nearly as good as the Helsinki predictions. Figures 5.9 and 5.10 clearly indicate that estimations are converging approximately 13 degrees off from the known installation angles. The reason for this significant delta is not known and there could be multiple contributing factors. One possible explanation is panel shading which is more likely to occur when the Sun is near the horizon. This causes a proportionally higher energy output loss during first and last hours of the day, resulting in a more narrow power generation plot shape. This *sharpness* is also influced by the tilt angle of an installation as was previously seen in figure 3.3 and thus the error in the estimated tilt angle could be partially caused by either panel self shadowing or shadowing caused by other stuctures near the PV panels.

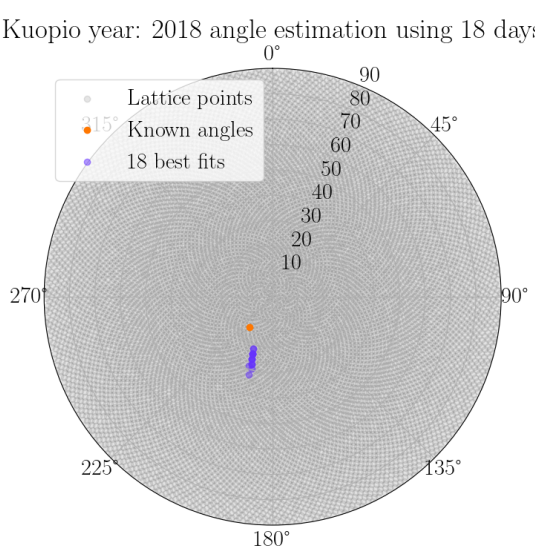


Figure 5.9: 10 000 point exhaustive search on 18 cloud free days from FMI Kuopio dataset using year 2018.

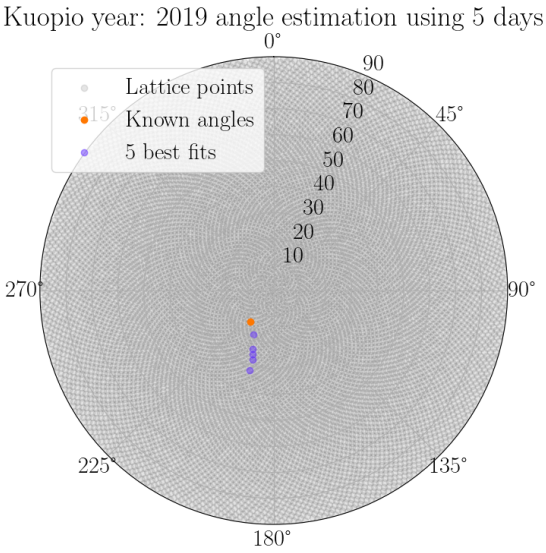


Figure 5.10: 10 000 point exhaustive search on 10 cloud free days from FMI Kuopio dataset using year 2020.

5.5 Solving panel angles iteratively

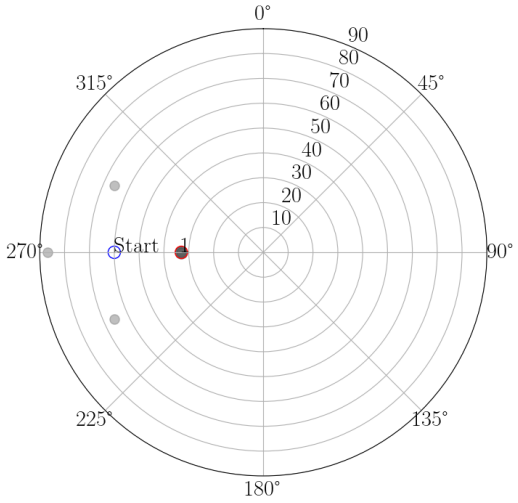
The exhaustive search used in earlier section suggest that the method is capable of estimating panel installation angles accurately. However evaluating 10 000 point lattices is somewhat inelegant and this can be avoided by using multiple less dense lattices iterarively if the fitness space satisfies some requirements.

The first requirement is that the fitness space should be smooth. This smoothness does not have to be perfect, fine patterns and details in the fitness space surface do not cause issues with iterative search algorithms if the noise pattern is not observable in the iterative lattices. Based on an earlier figure generated by exhaustive search 5.6, this requirement would appear to be met.

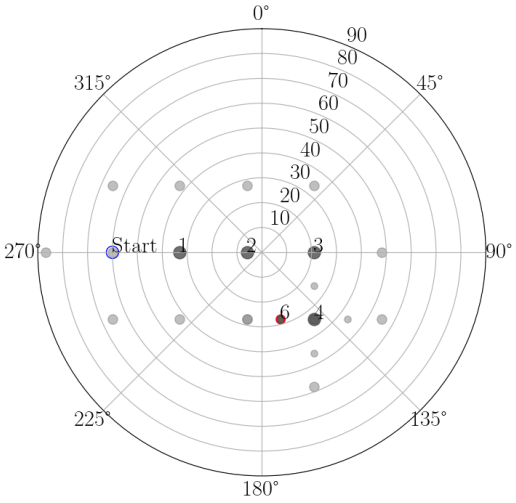
Second requirement is that the fitness space should contain as few convergence points and their respective basins as possible. These basins are regions in fitness space which are defined by a convergence point and their surrounding regions where the slope of the space leads to the respective convergence point. The region around best found fit in 5.6 forms a large basin but there would also appear to be a second converge point somewhere around azimuth 0° tilt 90°.

Iterative panel angle estimation algorithm

1. Choose a cloud free day from the dataset for evaluation.
2. Choose a starting or ”center” point from the angle space. This can be either the best result from a low density lattice or a fixed point such as tilt 45° azimuth 180°.
3. Evaluate the fitness at the center point and store that as the center point fitness.
4. Choose a few points near this center point within a given distance and evaluate their fitness. If any of the neighboring points results in a better fit than the center point, this point will then be chosen as the new center point.
5. Repeat steps 3 and 4 until step 4 does not find a better fit. When this happens, decrease the distance used for the local lattice in step 4.
6. Repeat steps 3,4 and 5 for a set number of times. Last center point is the best iteratively found fit.



(a) Iterative best fit search after 1 cross pattern search.



(b) Iterative best fit search after 6 cross pattern searches.

Figure 5.11: Visualization of a + pattern iterative best fit search algorithm. Numbers next to markers indicate that a point was the best found during n:th cross pattern search.

5.5.1 Local lattice generation

The step 4 in the iterative panel angle solving algorithm detailed above is the first non-trivial step in the algorithm as it requires the generation of local lattices. One option would be to use a subsection of a Fibonnaci lattice, but generating large Fibonnacci lattices and utilizing only a small subsection would require unnecessary computation.

The method shown in 5.11a is based on generating a + pattern by first transferring the starting point from angle space to a unit disk with 5.24. On this unit disk 4 points are generated, each of which is s units away from the start point either on x or y -axis using pattern 5.25. When these 4 points are transformed back into angle space with 5.26, they can be used as the local lattice surrounding the chosen center point.

Angle space to unit disk equations

$$\begin{aligned} d &= Tilt/90^\circ \\ x &= cos(Azimuth)d \\ y &= sin(Azimuth)d \end{aligned} \tag{5.24}$$

Unit disk points

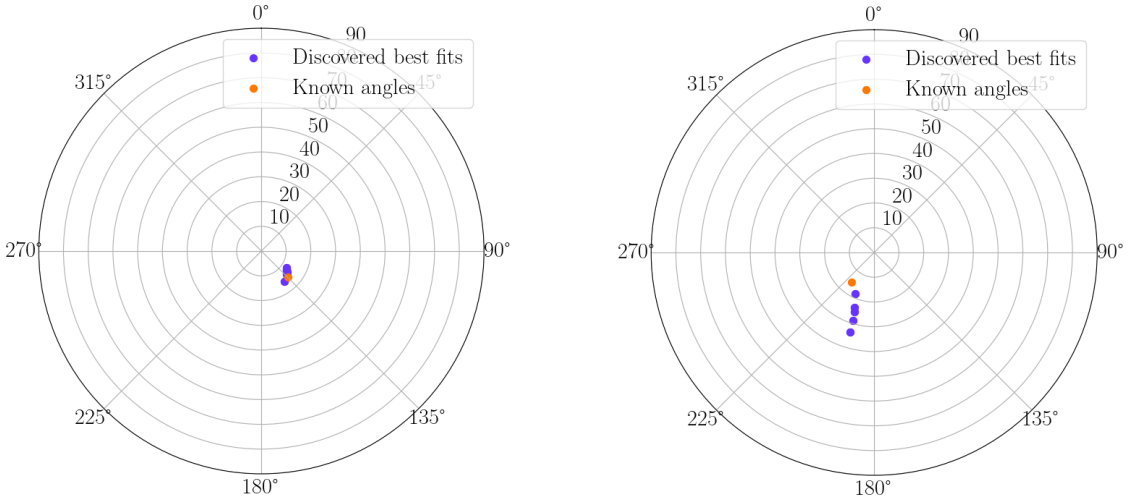
$$\begin{aligned} p_1 &= (x+s, y) \\ p_2 &= (x-s, y) \\ p_3 &= (x, y+s) \\ p_4 &= (x, y-s) \end{aligned} \tag{5.25}$$

Unit disk to angle space equations

$$\begin{aligned} d &= x^2 + y^2 \\ Tilt &= \sqrt{d} * 90^\circ \\ Azimuth &= tan^{-1}2(x/d, y/d) \end{aligned} \tag{5.26}$$

5.5.2 Results

By using 30 iterations with initial search distance of 0.3 on a set of cloud free days from the FMI Helsinki and Kuopio dataset, scatter patterns 5.12a and 5.12b are generated. The center angle distances between known angles and cluster means for the Helsinki and Kuopio scatter patterns are 1.97° and 12.47° respectively. The Helsinki scatter pattern is marginally worse than the pattern achieved by exhaustive search algorithm were as results for Kuopio data seem to have a similar large bias as before with the exhaustive algorithm. Exact values vary depending on day smoothness requirements and chosen year.



(a) Iterative search used on 8 days from FMI Helsinki dataset.

(b) Iterative search used on 5 days from FMI Kuopio dataset.

Figure 5.12: Scatter patterns for iterative panel estimation algorithm on multiple days.

One method for comparing the iterative method against the exhaustive is to compare the fitness values achieved by both methods. If for example the resulting fitness values are better for the exhaustive search, then there could be multiple convergence points in the basin of the fitness space and the iterative function could be converging on the wrong point. Where as if the iterative method achieves worse center angle distances and better fitness values than the exhaustive, the seemingly worse results of the iterative method would be a result of a better fit existing in the angle space which the Fibonnacci lattice points did not contain. By testing out individual days and sets of days, the difference between achieved fitness values with the different

estimation methods do not seem to vary significantly. For the Helsinki dataset the fitness values are somewhere in the 75W to 120W range for both algorithms depending on chosen parameters, with even performance with both algorithms.

5.5.3 Choosing between exhaustive and iterative angle estimation methods

Both methods are capable of achieving good to results depending on data quality and algorithm parameters. As there are multiple algorithm parameters and as the exhaustive is computationally expensive, it is difficult to prove that one method is better than the other. Each day in the datasets has an unique fitness space and parameters such as initial search distance, starting point and lattice point counts have a small and difficult to measure effects on the results of the algorithms.

That being stated, the computational speed of the iterative method is much higher than that of exhaustive search algorithms. If 30 steps are used with a 4 point local lattice, estimating the panel angles for a single day require 121 simulations and comparisons. This means that approximately 83 days can be iteratively evaluated in the time it takes to exhaustively evaluate one day with a 10 000 point lattice. As the 30 step iterative search algorithm reaches search distances as low as $9e-6$, the iterative method is using finer precision than would practical and the speed of the iterative algorithm can be increased further by setting a search distance limit.

Instead of choosing one method, the two can also be combined and used together. An iterative method with 5 to 10 steps is enough to find the general location of the best fit and a local subsection of a dense Fibonnaci lattice could be used in order to search a subsection of the angle space. This would however increase the algorithm complexity for small potential gains.

Figure 5.13 shows a comparison of fitting results for a single day. There the difference between pover curves is not significant regardless of the estimation method used and thus iterative method may be preferable. The difference between exhaustive and iterative search is likewise imperceivable in the example from FMI Kuopio dataset seen in figure 5.14 despite the higher center angle distance achieved with both exhaustive and iterative methods in all earlier examples.

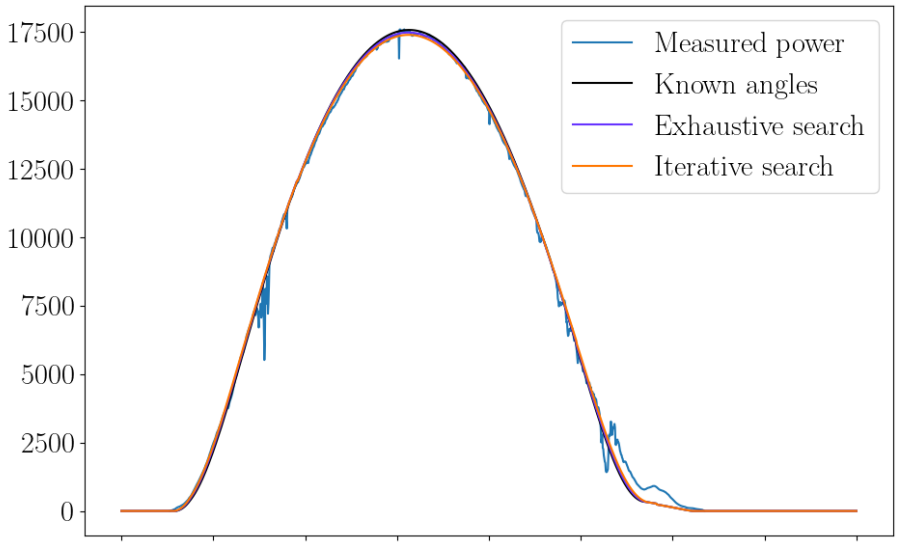


Figure 5.13: Comparison of power curves from measurements, simulation with known installation angles and estimated panel angles with exhaustive and iterative method. Using day 150 from year 2019 in Helsinki dataset.

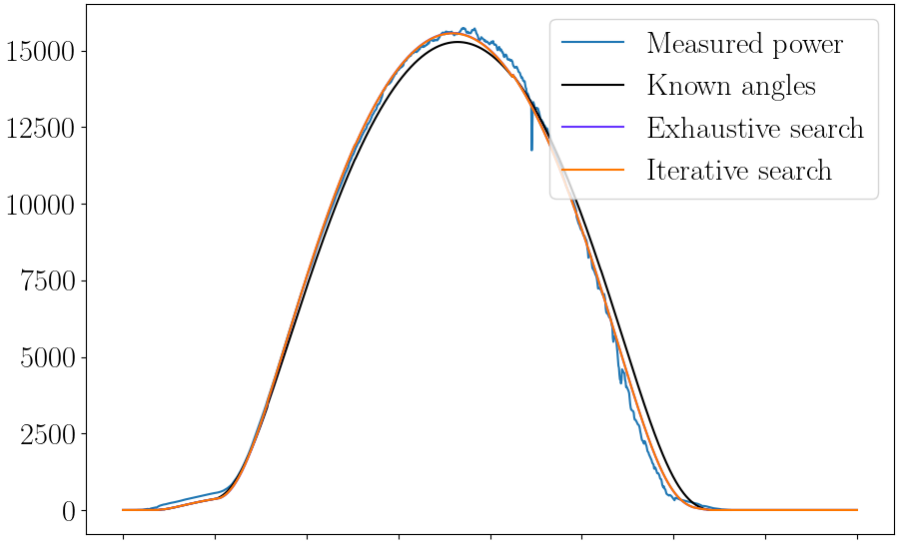


Figure 5.14: Comparison of power curves from measurements, simulation with known installation angles and estimated panel angles with exhaustive and iterative method. Using day 167 from year 2019 in Kuopio dataset.

5.5.4 Further development ideas

Both algorithms perform well enough to make some of the assumptions made on the properties of fitness spaces practically irrelevant. Thus concepts of basins were not explored thoroughly but they and the field of dynamic systems which studies attractors and basins could offer additional insights into similar optimization problems.

The problem of model fitting for angle estimation may also have connections to convolutions. Understanding the mathematical properties of signal impulses and their convolution shapes could help in proving how likely the formation of multiple basins are. This could be beneficial in starting point selection with iterative fitting algorithms.

The estimation algorithms should be evaluated with multiple different datasets in order to determine whether results from the Kuopio data are an anomaly or the algorithms are as noise sensitive as the Kuopio data suggests. Similarly data from different sources and with different temporal resolutions would help in verifying how well the algorithms perform with available data.

Chapter VI

Conclusion

PV system parameter estimation results for the FMI Helsinki dataset are very good to excellent. Angle estimation results with a center angle delta of less than 1° were achievable with both iterative and exhaustive algorithms. Geolocation estimation proved to be more difficult with the scatter pattern from multiple years having small amount of bias and a fair amount of noise. Scatter formation is spread around the FMI Helsinki installation and is approximately 50km by 200km in dimensions. Due to small sample size of 5 datapoints this algorithm is harder to evaluate.

Results for FMI Kuopio dataset were noticeably worse. Center angle delta with panel installation angles was approximately 13° regardless of estimation method used. Similarly the scatter pattern in geolocation estimation resulted in a 50km by 300km region with outliers.

The differences in the algorithm performances between the datasets can partially be explained by the noise present in the Kuopio data where something would appear to be casting shadowns onto the panels. Differences may also be partially caused by parameter estimation algorithm parameters such as used day ranges which affect the results of estimation algorithms.

.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
```

References

[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] N. Martin Chivelet and J. 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* **70**, 25–38 (2001).

[9] D. King, J. Kratochvil, and W. Boyson, *Photovoltaic Array Performance Model*, Vol. 8, , PhD thesis 2004).

[10] T. Huld, R. Gottschalg, H. G. Beyer, and M. Topič, "Mapping the performance of PV modules, effects of module type and data averaging," *Solar Energy* **84**, 324–338 (2010).

[11] D. W. Hughes, B. D. Yallop, and C. Y. Hohenkerk, "The Equation of Time," *Monthly Notices of the Royal Astronomical Society* **238**, 1529–1535 (1989).

[12] Á. González, "Measurement of Areas on a Sphere Using Fibonacci and Latitude–Longitude Lattices," *Mathematical Geosciences* **42**, 49–64 (2009).

[13] H. Vogel, "A better way to construct the sunflower head," *Mathematical Biosciences* **44**, 179–189 (1979).

[14] V. Seibert, "Distributing points on a sphere," (**!!YEAR!!**).