Accurate Irradiance Measurement
on Large Scale Photovoltaic System Application

Anindio Prabu Harsarapama

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

ACCURATE IRRADIANCE MEASUREMENT

ON LARGE SCALE PHOTOVOLTAIC SYSTEM APPLICATION

by

Anindio Prabu Harsarapama

in partial fulfillment of the requirements for the degree of

Master of Science
in Sustainable Energy Technology

at the Delft University of Technology,
to be defended publicly on Wednesday March 29, 2017 at 10:00 AM.

Supervisor:	Dr. Olindo Isabella	
Thesis committee:	Prof. Dr. Miro Zeman,	TU Delft
	Dr. Milos Cvetkovic,	TU Delft
	Dr. Marc Korevaar,	Kipp & Zonen B.V.

This thesis is confidential and cannot be made public until March 29, 2017.

Picture on the cover page and its copyright belong to Kipp & Zonen B.V.

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Abstract

As the number of large scale photovoltaic power plants (which also known as solar parks) is growing in recent years, standard activities related to the system such as the operation and maintenance of the plant and grid management have become more crucial. In order to ensure a standard quality of these activities, accurate irradiance measurement is required. Due to the relatively wide area of a large scale solar park, the irradiance measurement would most likely be suffering from additional error induced by spatial variation. Therefore proper planning regarding the optimum amount of irradiance measurement devices and how these devices are arranged in a space is one solution to minimize the issue.

A statistical model based on data interpolation, clustering, and searching concepts has been built in order to approximate the optimum irradiance measurement devices configuration on an actual solar park. The statistical model utilizes satellite-derived irradiance data as its main input. In return, the model estimates the measurement uncertainty for every additional irradiance measurement devices and also roughly approximates the most probable positions for the devices.

The study case result shows that an imaginary solar park in Leipzig with a total area of 1.8 km^2 can reduce its measurement uncertainty for about 4% by increasing the amount of irradiance measurement devices from 2 to 30. While in similar case, identical imaginary solar park built in Arizona is only able to reduce its measurement uncertainty by no more than 2%. It has also been shown that utilizing a dataset with 4 times less temporal resolution would lead to reduction of measurement uncertainty for a maximum of 1%.

Instead of being uniformly distributed throughout the solar park, the model's suggested positions for the measurement devices are concentrated on certain areas within the solar park. It is found that these concentrated areas are formed due to yearly spatial variations that occur within the particular region. In the end, it is hoped that the information provided by the model can accommodate the need of some actors within the solar park industry with regards to how relatively large area of a solar park influences the accuracy of irradiance measurement.

PREFACE

The thesis 'Accurate Irradiance Measurement on Large Scale Photovoltaic System Application' that you are currently reading has been written for fulfilling graduation requirements of master program in Sustainable Energy Technology at Delft University of Technology. Overall researching and writing thesis activities had been done within April 2016 - March 2017 period.

The topic was firstly offered by Kipp & Zonen B.V. to PVMD research group. The topic was then conveyed to me through Olindo when I was still seeking for thesis topic. The topic was very abstract and hard to interpret, no wonder at early stage the progress was very slow. Fortunately after several discussions with many people, the progress was steadily improving as I started to grasp the real issue that had to be solved in order to satisfy the main research objective.

First I have to thank Olindo for introducing me to this interesting topic and of course guiding me throughout the whole thesis process. I am very grateful for all of your supports during this period. Second I would like to thank Marc for willingly taken the role of being my daily supervisor even though it was so sudden. I really appreciate all suggestions that you gave me during our weekly discussion, they really helped me a lot. Next I would like to thank Bagas for patiently answering all my questions regarding some statistical terms that I never heard of, let us meet up again sometimes in Jakarta. I would also like to thank everyone on Kipp & Zonen Delft's office for treating me very nicely even though I am just a student who rarely showed up in the office.

To all of my friends who have been hanging out with me for the last 2 years, thank you very much for taking care of me. Your indirect support really helped me getting through the hard times, I will not forget the wonderful time that we have spent together. Lastly, this thesis would not be completed without the support from my family at Indonesia especially my amazing parent who always gave me irreplaceable support at every occasion.

I wish you enjoy the reading and gain something from it.

Anindio Prabu Harsarapama
Delft, March 2017

CONTENTS

List of Figures

List of Tables

1

INTRODUCTION

1.1. BACKGROUND

In recent years photovoltaic technology has emerged as one of leading renewable energy technology currently available. According to International Energy Agency (IEA) report, the global photovoltaic market grew by at least 48.1 GW in 2015, which was 25% more than the photovoltaic market growth in 2014. This growth made the total installed capacity of worldwide photovoltaic power plants into at least 227.1 GW by the end of 2015, or comparable to 1.3% of the global electricity demand [9].

IEA also estimates that 10-15% of existing worldwide photovoltaic installations are large scale power plants (over 20 MW in capacity) [10], or what are commonly called 'solar parks'. Following the trend of the global PV market, the total number of solar park installations is also climbing up in recent years especially in countries that currently have high photovoltaic market growth, such as China and USA. Besides their total number, solar parks are also growing in terms of individual plant capacity. IEA records show that during early development phase of solar parks around the mid-2000s, the largest capacity of a single solar park was only around 20 MW. Nowadays it is not uncommon to see that many operating solar farms already exceed 100 MW in term of capacity, with the largest one rated for more than 500 MW.

In order to create such a massive power plant, detail and careful planning of the system is necessary; starting from a pre-feasibility study of the potential site to regular monitoring of the plant. Most of these steps require accurate site-specific weather datasets (especially incident sun irradiance-related datasets) to produce a satisfactory output by the end of each process. Even though there are several approaches to indirectly derive this kind of dataset (e.g. the physical model, the satellite model), direct on-site ground measurement is still the most appropriate way to acquire accurate weather datasets. The following points briefly describe the importance of on-site ground measurement in solar park applications.

1.1.1. ON-SITE MEASUREMENT FOR SITE ADAPTATION PURPOSE

During the feasibility study of a potential solar park site, the long-term variability of solar resource on the point of interest has to be assessed thoroughly to minimize the financial risk of typical long term investments in solar parks. Since the new potential site most likely does not have any long term datasets (on-site measurements station has yet to be installed or just has been installed very recently), a reliable long term resource dataset has to be derived from combining both short term datasets acquired from on-site measurement and other secondary sources. Several secondary sources that are currently available in present practice [11–15] are either dataset derived from physical models based on satellite images or numerical models based on the physical state of atmosphere (NWP model). The datasets acquired from these sources are usually only utilized as a preliminary study for screening potential sites in a particular region, mostly due to their low spatial and temporal resolution for site-specific application.

The adjustment process of long term datasets obtained from these secondary sources with available short-term on-site measurement datasets is often known as 'site adaptation'. Several studies regarding site adaptation using satellite-derived datasets are presented on [16–19] while the study that utilizing dataset from NWP model is available on [20]. In summary, the site adaptation process can be classified into two main approaches, namely physical-based and statistical-based method more commonly known as Model Output Statistics (MOS) [21].

The main idea of adaptation in the physical based method is to adjust the atmospheric properties input of the main model which is usually derived from another model. Nowadays, aerosol turbidity [e.g. Aerosol Optical Depth (AOD)] is the common target to be adapted from on-site measurement since it has a significant influence on lowering the bias of satellite-derived irradiance or, in other words higher accuracy on predicting site-specific long term solar resources.

Unlike the physical-based method which indirectly improves the result by refining the input, statistical based method directly does the adaptation on its result. The basic idea behind the statistical site adaptation method is to find and remove bias error (if it is evident) of the secondary source dataset. By doing this, bias error contribution to long term datasets derived from secondary sources will be minimized and only random error will still remain [21]. According to [19], at least one year high-quality on-site measurement dataset is required to see the effectiveness of MOS analysis.

Depending on which method has been chosen, bias error of the model will be either removed or reduced hence the overall uncertainty of site-specific solar resource long term prediction will also be reduced. This reduction will undoubtedly decrease the financial risk of long term investment in that particular site. In short, to have reliable long term solar resource prediction direct on-site measurement is simply a must.

1.1.2. ON-SITE MEASUREMENT TO MINIMIZE GRID INTERMITTENCY ISSUE

In order to ensure the balance between power generation and load within the grid (intermittency issue), the utility operator needs to have some kind of prediction of how the output power of the solar park would fluctuate in realm of space and time. By doing so, a necessary countermeasure strategy and reserves can be arranged beforehand.

Figure 1.1: Cloud cover on 23.3 MW solar farm in Spain

A lot of studies [22–28] regarding this issue have been conducted in the past couple of years due to the increasing number of solar park's high penetration into the grid. All of them agree that the biggest problem in this issue lies on short term (less than 1 hour) variability of solar resource instead of long term variability. Longer term solar resource variability is more predictable than the shorter one since the effect of moving clouds which is considered as the primary reason of irradiance short term variability has been evened up by the averaging process required for lower time resolution (long term variability).

The current practice of short term forecasting for PV application relies strongly on the availability of on-site measurement data. For instance, two models that have been accepted so far for irradiance short term variability prediction, namely the dispersion factor method and wavelet variability model (WVM) require high quality (high time resolution) on-site measurement datasets to be able to predict short term irradiance variability with acceptable accuracy [23, 24]. In short without on-site data measurement dataset, these two approaches cannot be conducted at all.

1.1.3. ON-SITE MEASUREMENT FOR PERFORMANCE MONITORING

Upon completing the construction phase, the monitoring process of the plant will be started. The main activity in PV monitoring practice is to compare actual and estimated energy yield of the plant. This comparison can be analyzed further on for more detailed questions such as; how efficient is the performance of the plant?

Is there any design fault inside the system? What can be done to improve the performance? How often should the maintenance and cleaning of the plant be scheduled? This kind of analysis is required to ensure that the plant is working according to what it has been promised to the investor or to optimize it even further.

A guideline for monitoring activity of all operating PV system (small to large scale) was firstly introduced in [29]. This particular report was later on adapted as the international standard for PV system monitoring activity which is currently presented in IEC 61724 [30]. The National Renewable Energy Laboratory (NREL) has suggested several revisions regarding this outdated standard, especially related to the Performance Ratio (PR) issue [31]. According to the official website of the International Electrotechnical Commission (IEC), the revised version of this standard is still being discussed [32], and will hopefully be available in near future. In the mean time, 'corrected PR' term suggested by NREL has been already taken into real practice for quite sometimes due to its crucial effect on PV plant monitoring activity.

According to the standard [30], PR is the ratio between actual energy that is being produced and energy that is expected to be produced by the power plant. Actual energy yield can be derived directly from power sensors placed either in the DC or AC side of the power plant and the measurement reading is then integrated within a selected time range to obtain energy yield in a certain time resolution. On the other hand, the expected energy yield is indirectly derived from in-plane measured irradiance which is then processed further to estimate the energy yield of the plant. Previous statements implicitly state that both power and in-plane irradiance measurements play an important role in determining PR value.

Even though some proposed PV system performance monitoring models prefer to not include PR in their analysis [33, 34], PR is still considered as the most suitable non-dimensional indicator due to its independence from other site specific variables such as particular weather conditions and modules orientation [35, 36]. Moreover, most of the recent proposed PV system performance monitoring models decided to utilize PR as the main indicator to detect abnormalities within PV system [36–38]. Since PR has been proven to be a crucial indicator for performance monitoring activity, a slight inaccuracy in both power and irradiance measurements may lead to misleading performance interpretation by the system analyst.

Based on the above explanation, it is clear that the irradiance measurement is one of the most important on-site measurements that have to be conducted on solar park practice. As mentioned on IEC 61724, irradiance should be measured by either a calibrated reference cell or pyranometer. Since different technology in the reference cell causes a considerable difference on measured value [39], there is no clear standard regarding which technology should be used when the owner decides to use a reference cell instead of a pyranometer. Moreover, recent study also revealed that a reference cell is less suitable to be irradiance measurement sensor since it is less stable in term of performance compared to a pyranometer [40].Nevertheless, in order to simplify the matter, further analysis will be limited only on irradiance measurement instead of other on-site measurements related to solar parks.

1.2. RESEARCH OBJECTIVES

Upon confirming the importance of on-site irradiance measurements in solar park practice from several interests point of view, the next question that immediately comes to mind would be how dense or how the configuration of the on-site irradiance sensors should be to provide accurate measurement datasets for further analysis (e.g. performance monitoring or short term prediction analysis).

As an analogy, a 550 MW Topaz solar farm in USA is spread over 19 km^2 of land [41]. Within this vast area there might be a wide spatial variability in actual in-plane irradiance all over PV arrays while at the same time there are only a limited number of irradiance sensors available. Therefore it is inevitable that irradiance values within a certain spatial resolution are only represented by reading from a single irradiance sensor; a higher density of irradiance sensors leads to less reading area coverage by each sensor which also means less data smoothing in the spatial realm. This process will undoubtedly influence the uncertainty of estimated energy yield (the main parameter in most solar farm related analysis) which is indirectly derived from these measurement datasets.

Even though it is an important issue, only a limited amount of information is known so far. For instance, the IEC 61724 document only specifies that the location of irradiance sensors should be able to represent the actual irradiation condition of array [30] without any further explanation regarding the recommended amount. There is also a solar performance monitoring handbook published by SunSpec Alliance which recommends that there should be at least one weather station for every 10 MW capacity of the power plant [42]. However it is not very clear how they ended up with that particular number since there is no valid derivation or any other scientific reference presented within the document.

Apart from these two documents, there are several studies which have tried to figure out the optimum number and position of irradiance sensors required to cover a certain spatial area [43–45]. Unfortunately, all of these studies conducted the research only over relatively vast area compared to the total area of the largest solar park that exists today. Therefore similar questions for a case involving much less area like a single solar park still remain unanswered and answering these questions will be the main topic of the report.

1.3. METHODOLOGY

Statistical modelling is used as the main approach to analyze irradiance input data. Due to the limitation of available irradiance data, a particular on-line satellite-derived irradiance database is utilized as main data resource. On the other hand, the uncertainty of average measured values becomes the main indicator of the model to estimate the optimum irradiance sensors configuration (amount and position). Further on, the recommended configuration of irradiance sensors suggested by the model is evaluated by simple cost-benefit analysis to obtain a broad picture about the effect of additional irradiance sensors from an economic point of view.

2

LITERATURE REVIEW

As mentioned previously, only a limited amount of prior related studies have been done so far, and most of these studies mainly focus on statistical data clustering analysis which has been proven to be a very handful tool to extract information out of the available satellite-derived irradiance database [43, 45, 46]. However, to be able to implement this tool into the analysis, a large amount of data points are required; otherwise the data clustering process will not be able to serve its purpose. This requirement was not an issue for all of those prior studies since they limit their input dataset to a relatively large spatial area (country scale) which contains more or less sufficient data points for data clustering process. It would be a different matter when we are talking of similar case but with much smaller spatial area like a solar park. The total area of the largest solar park that exists today is still relatively small compared to the finest spatial resolution of present-day satellite-derived irradiance data (± 12.3 km^2). Due to this issue, the input irradiance dataset would contain insufficient number of data points for the data clustering process. In order to tackle this issue, spatial interpolation process is introduced as a data points generator within the area of interest. The second chapter will discuss all the important concepts regarding these two statistical tools. Apart from that, this chapter also explains a little about the combination searching algorithm which is also incorporated into the model. Last but not least, this chapter provides brief information with regard to how satellite-derived irradiance as the main input dataset for the model is derived.

2.1. SATELLITE DERIVED IRRADIANCE DATABASE

Satellite technology has been utilized by human civilization for decades, and during those times, it has been able to make many contributions to several scientific researches. One of these researches is related to how satellite imagery can be utilized to predict sun irradiance on the earth's surface at a particular time. There are several models to derive satellite-based irradiance datasets and generally, apart from satellite images, most of the models need several other atmospheric parameters such as Aerosol Optical Depth (AOD) and Precipitable Water Vapor (PWV) as inputs. By doing so, the atmosphere transmittance can be estimated and this value will be then utilized to correct the estimated clear sky irradiance which is separately obtained from the other model. However, before we dive a little bit deeper into these models, it is advisable to understand more about the instrument that has been embedded into the satellites for capturing the images.

2.1.1. GEOSTATIONARY SATELLITES

Currently, there are five geostationary satellites on the equatorial area around the world which capture images in global coverage; due to the satellites positions, the images only cover the area within 66° south and north latitude range. The first two satellites which are operating around the American continent and Pacific Ocean belong to the US; namely GOES-West (or GOES-15) and GOES-East (or GOES-13). These two satellites are equipped with an imager instrument that measures five different wavelength bands; one within the visible channel (VIS) with 1 km resolution(630 nm) and four within the infrared channel (IR) with 4 km resolution (3900 nm, 6480 nm, 10700 nm, and 13300 nm) [47]. For the northern hemisphere, the satellites capture the images every 30 minutes while for southern hemisphere the images are taken every three hours. The combined images from these two satellites have been used as input for constructing several solar irradiance databases such as NSRDB (National Solar Resource DataBase),SolarGIS and Clean Power Research.

Figure 2.1: Position of current geostationary satellites

The next two satellites which are operating around the Atlantic Ocean, Europe, Africa, and a part of Indian Ocean belong to European Organization; they are called METEOSAT-7 (eastern region) and METEOSAT-9 (western region). METEOSAT-7 which has been launched earlier than METEOSAT-9 is equipped with the MVIRI (Meteosat Visible and Infrared Imager) sensor that has three wavelength channels with 30 minutes temporal resolution; VIS (450-1000 nm) on 2.5 km resolution, IR (10500-12500 nm) and water vapor (5700-7100 nm) on 5 km resolution [48]. On the other hand, METEOSAT-9 is equipped with a newer generation of imager instrument, called SEVIRI (Spinning Enhanced Visible and Infrared Imager) which has 11 wavelength channels on 3 km resolution and one high resolution visible channel on 1 km resolution; table 2.1 summarizes the information regarding each channel. Unlike its previous generation (MVIRI), SEVIRI is able to provide the image every 15 minutes. Images from these satellites have been utilized for building solar irradiance databases such as SOLEMI, HelioClim, CAMS and SolarGIS.

Table 2.1: Summary of SEVIRI wavelength channels [7]

No.	Band Channel	Band width (nm)
1	High resolution VIS	500-900
2	VIS 0.6	560-710
3	VIS 0.8	740-880
4	IR 1.6	1500-1780
5	IR 3.9	3480-4360
6	IR 8.7	8300-9100
7	IR 9.7	9380-9940
8	IR 10.8	9800-11800
9	IR 12	11000-13000
10	IR 13.4	12400-14400
11	Water Vapor (WV) 6.2	5350-7150
12	WV 7.3	6850-7850

The last geostationary satellite that covers the Asia and Oceania region is called Himawari-8 and belongs to Japan. In its earlier versions (Himawari-7 and beyond), this satellite was better known by the with GMS and MTSAT name, instead of Himawari. Compared to the other four previous satellites, Himawari-8 is equipped with the most sophisticated imager instrument called AHI (Advanced Himawari Imagers). This instrument has 16 wavelength bands which is considered to be a huge leap compared to the one embedded on MTSAT-2 (Himawari-7) which only had five wavelength bands; a full summary of each band within AHI is presented on table 2.2. Similar to the other previous four satellites, images generated by Himawari have also been implemented in global solar database such as SolarGIS.

Table 2.2: Summary of AHI wavelength channels [8]

No.	Band Channel	Spatial resolution (km)	Temporal resolution (min)
1	VIS 0.46	0.5-1	10
2	VIS 0.51		
3	VIS 0.64		
4	Near Infrared (NIR) 0.86	1-2	
5	NIR 1.6		
6	NIR 2.3		
7	IR 3.9		
8	IR 6.2		
9	IR 7.0		
10	IR 7.3		
11	IR 8.6		
12	IR 9.6		
13	IR 10.4		
14	IR 11.2		
15	IR 12.3		
16	IR 13.3		

2.1.2. GENERAL DERIVATION ALGORITHM

Upon knowing general information about images (*e.g.* temporal and spatial resolution, bandwidths) generated by different satellite imager technologies, the discussion about the topic continues to how are surface irradiance datasets (GHI, DNI) derived from those images. According to [49], satellite-based methods for estimating surface irradiance can generally be divided into three categories; empirical, semi-empirical and physical models. The following points explain briefly how these methods differ from each other.

- **Empirical approach**
  The empirical method is based on the correlation between irradiance value sensed by the satellite image and actual surface irradiance value measured on ground stations. The linear regression model obtained from this correlation is defined as the relation between atmospheric transmittance and the ratio of incoming and outgoing radiation at TOA (Top of Atmosphere). Further on the atmospheric transmittance will be utilized to estimate GHI on the earth's surface. Of all solar resources database that are still operational nowadays, none of them uses pure empirical approach as their base algorithm.

- **Semi-empirical approach**
  The semi-empirical method is a hybrid method that combines both the physical and empirical approach. The main idea of this method is to figure out cloud cover indices from satellite visible channel images which is actually composed of correlation of solar irradiance at TOA and surface radiation of the earth. Furthermore, clear-sky GHI which is approximated through separate physical model (atmospheric) will be modified accordingly to cloud cover indices in order to obtain irradiance on earth surface. The previous statement also implicitly means that semi-empirical approach is able to capture surface irradiance variation due to cloud movement, which might be hard to do using the pure empirical method (regression analysis). HELIOSAT-2 is an example of a semi-empirical based model and is currently being used to construct the HelioClim database.

- **Physical approach**
  As the name implies, the physical method does not use satellite images directly to estimate irradiance values on the earth's surface as the other methods do. On the other hand it only uses satellite imagery to retrieve information related to cloud products (*e.g.* cloud optical depth,cloud height,etc.) as main input for the radiative transfer model which is able to estimate surface irradiance directly. Even though it seems like a simple process, the inner algorithm inside the model has to take into account numerous physical laws to simulate all possible physical occurrences that might influence the surface irradiance value such as multiple light reflections and scattering within the atmosphere, the effect of elevation, etc. Due to that reason, this method is considered to be more demanding in term of computation requirement when compared to the other two methods. Physical Solar Model (PSM) and HELIOSAT-4

are simple examples of the physical based model; both of them are currently being used to build NSRDB and CAMS databases, respectively.

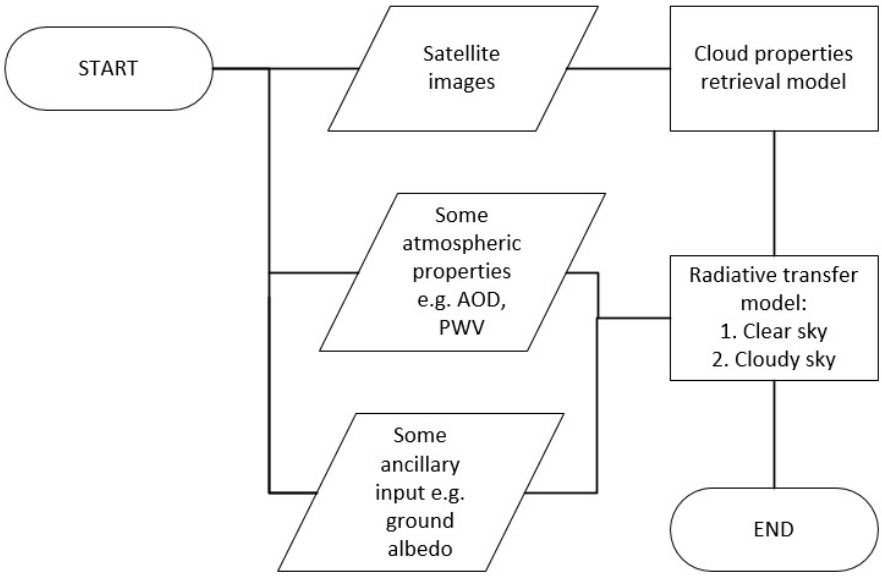


Figure 2.2: General flowchart of current semi-empirical and physics model to derive irradiance data from satellite images

From the above explanation it is clear that apart from the complexity of radiative transfer model in each approach, the semi-empirical and physical approaches only differ in term of how the satellite images are processed and used. Since both approaches are currently being implemented in solar databases nowadays, a simple flowchart that the summarizes general derivation process of irradiance data from satellite images is provided in 2.2.

2.1.3. EXISTING SATELLITE DERIVED IRRADIANCE DATABASE

The following section gives important information regarding several well-known solar databases that utilize satellite images as its main input. All of the mentioned databases are mostly free to use except several options that are only available for premium users. Discontinued solar databases (*e.g.* DLR-ISIS, SOLEMI, NASA SSE) and full premium databases (*e.g.* SolarAnywhere, 3TIER, SOLARGIS) will not be covered in this report.

HELIOCLIM

The HelioClim database spatially covers the earth's surface within -66°to 66°both in latitude and longitude. This database simultaneously collects and merges images taken by Meteosat-9 and Meteosat-7 and then passes them through into HELIOSAT-2 model to obtain ground irradiation (Wh/m^2) data that can be accessed through SoDa service[1]. The best temporal resolution that can be provided by this database is fixed at 15 minutes[1]. Spatial resolution varies depending on point location as shown in fig 2.3. This variation exists due to the effect of the earth's curvature.

HELIOSAT-2 model was developed by MINES ParisTech between 1999-2002 with the European Commission support. As briefly mentioned before, HELIOSAT-2 uses the semi-empirical approach that combines the clear-sky model and empirical 'cloud indices' obtained from satellite images interpretation to obtain local irradiation[49]. Current version of HelioClim database(HC3v5) uses the McClear clear-sky model replacing its predecessor (HC3v4) which used the ESRA (European Solar Radiation Atlas) clear-sky model[1]. The ESRA clear-sky model is considered to be more demanding in term of data input compared to the McClear clear-sky model since it needs Linke turbidity factor which has never been updated in local scale; local effect such pollution or volcanoes eruption could lead to drastic change in atmosphere turbidity [1]

The validation of HelioClim database has been done on 14 BSRN (Baseline Surface Radiation Network) stations and the overall result shows satisfactory performance from this database. The validation of sum of GHI for each 15 minutes gives out relative bias values around -4.3%-4.7% and relative RMSE(Root Mean

[1] See http://www.soda-pro.com/web-services/radiation/helioclim-3-for-free.

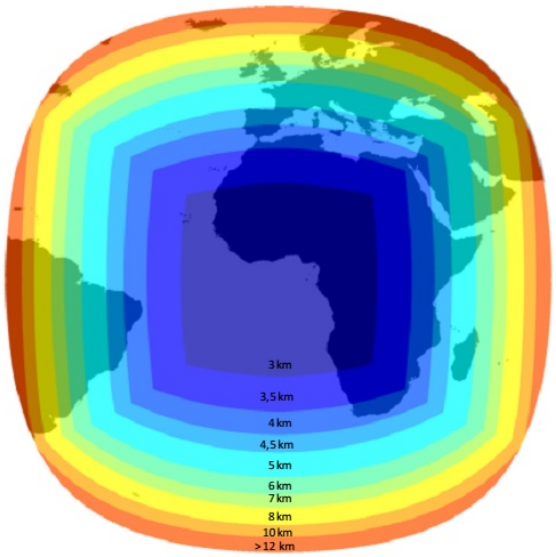


Figure 2.3: Spatial resolution of image taken by Meteosat-9 and Meteosat-7[1]

Square Error) values around 14%-37% depending on the location[50]. Average of RMSE values over these 14 measurement points is around 27% and further on within the model will be used as a representative value to simulate built-in error of HelioClim database.

COPERNICUS ATMOSPHERE MONITORING SERVICE(CAMS) RADIATION SERVICE

CAMS radiation database is built from HELIOSAT-4 model and free to be accessed through SoDa service[2]. This service is part of the Copernicus Programme coordinated by European Commission. Since the CAMS solar database also relies on images taken by Meteosat-9 and Meteosat-7, common properties of the database such as spatial and temporal resolution are similar to the HelioClim database.

HELIOSAT-4 model has been being developed by collaboration between MINES ParisTech and German Aerospace Agency DLR under MACC (Monitoring Atmospheric Composition and Climate) projects funded by the European Commission. Unlike HELIOSAT-2, HELIOSAT-4 uses McCloud abacus as main feed for fast RTM (Radiative Transfer Model) to compute ground irradiance. McCloud abacus is composed from clear-sky indices which are separately obtained by interpreting satellite images through McClear clear-sky and McCloud models.

This database has been validated through 13 BSRN ground stations and the result shows more or less similar performance level to HelioClim database. Validation of GHI summed at every 15 minutes has relative bias around 0%-12% while the relative RMSE is around 15%-43%. The average of RMSE values over these 13 measurement points is around 27% and further on will be used as a representative value to simulate built-in error of CAMS service database[51].

NATIONAL SOLAR RADIATION DATABASE (NSRDB)

The NSRDB database which is provided for free by NREL(National Renewable Energy Laboratory)[3] spatially covers almost all parts of the American continent and parts of the Atlantic Ocean. GOES-15 and GOES-13 are currently supplying images with fixed 4-km spatial resolution and 30-minutes temporal resolution for the input model. The new version of NSRDB database is built from PSM replacing previous SASRAB (Shortwave Radiation Budget) model which suffered from relatively low bias at clear sky condition[52].

PSM was developed by NREL in collaboration with University of Wisconsin and NOAA (National Oceanic and Atmospheric Administration). The basic algorithm of PSM is almost similar to what HELIOSAT-4; interpreting satellite images into cloud properties and then feed this output as RTM's inputs. The only difference lies in chosen clear-sky model, while HELIOSAT-4 relies on McClear clear-sky model, PSM uses modified MAC (MMAC) clear-sky model whose core analysis is mainly based on AOD (Aerosol Optical Depth) and PWV (Precipitable Water Vapor).

[2]See http://www.soda-pro.com/web-services/radiation/cams-radiation-service
[3]See https://nsrdb.nrel.gov/nsrdb-viewer

A comprehensive validation (country scale) of new NSRDB database has been done, and similar to two previous databases, NSRDB database relatively shows satisfactory performance[53]. One downside though, the result of this validation is only shown as absolute error instead of relative error which is required to simulate the built-in error of NSRDB database within the model. Alternatively, there is another study which has done another verification process for NSRDB database but only on local scale. The verification result from this studies shows that the relative bias varies from -4%-11% and relative RMSE is around 32%-50% [54]. Further on, the average relative RMSE of 39% becomes the error representative for simulating built-in error of NSRDB database.

2.1.4. CLEAR SKY MODEL AND INDEX

As the name implies, 'clear-sky' stands for sky condition where there are no clouds on the sky, regardless the actual sky condition. This term is very important since all models algorithm of the current satellite-derived irradiance databases use this term to firstly estimate the surface irradiance value before applying any satellite images interpretation product. That is the reason why the dataset extracted from these databases is always accompanied by clear-sky based output.

There are several well-known clear-sky models that have been universally accepted; namely the Bird clear-sky model[55], ESRA model[56], Solis model[57], McClear model[58], and MMAC model[59]. All of them are trying to estimate clear-sky surface irradiance through the physical approach by feeding computed terrestrial irradiance along with a variety of atmospheric properties into specific radiative transfer computations. Even though more detailed explanations of these models will not be provided within this report, the reader is still encouraged to understand a little bit more about these models to get the feeling of how clear-sky irradiance is generally derived.

$$CSI = \frac{I_{estimated}}{I_{clear-sky}} \tag{2.1}$$

CSI : clear-sky index
$I_{estimated}$: either estimated irradiance or irradiation (W/m^2 or Wh/m^2)
$I_{clear-sky}$: either clear-sky irradiance or irradiation (W/m^2 or Wh/m^2)

The other clear-sky term that is important to be understood is clear-sky index (CSI). Clear-sky index theoretically scales sky condition from 0 to 1 with value close to 0 shows a very cloudy sky condition and 1 means clear-sky condition; eq 2.1 shows the formula to calculate CSI. Even though generally CSI value stays within 0-1 range, there are 2 main occasions where CSI of GHI would be outside this range. First, it occurs for datasets within nocturnal period; eq 2.1 returns Not a Number (NaN) value. Second, it happens when estimated GHI is dominated by diffuse component instead of direct component. This condition might trigger the case where $I_{estimated} > I_{clear-sky}$ which obviously returns a CSI value higher than 1. In order to remove variation due to cyclic relative movement of the sun from earth surface, most of prior related studies utilize CSI as main parameter for their analysis[43, 45, 46], and so does the model in this report.

2.2. SPATIAL INTERPOLATION

Spatial data interpolation process is expected to be able to simulate on ground irradiance spatial variation over the solar park, since it makes less continuous (limited) known spatial data points into more continuous (denser) spatial data points. The most common idea of interpolation process is to estimate unknown data point(s) within the range of known data points by using weighted expected value(s) from known neighborhood points. The magnitude of the weight literally explains the level of influence of known data points to the point where the data is being estimated. Apart from the previously mentioned approach, forecasting global or local trend of the known data points using polynomial regression could also be a viable approach to do spatial interpolation. However, limited amount of available data makes this kind of approach no longer applicable; therefore this approach will not be considered inside the report.

2.2.1. COMPARISON OF EXISTING SPATIAL INTERPOLATION METHODS

There are several methods to do spatial data interpolation with weighting approach, each with its own advantages and drawbacks depending on the characteristics of available datasets. The following sections describe briefly the comparison between several known methods.

- **Inverse Distance Weighting (IDW)**
  This spatial interpolation method was first proposed by Donald Shepard on 1968; that is the reason why this method is sometimes called Shepard's method. As the name implies, this method sets weight depending on the Euclidean distance between known and estimated points; eq. 2.2 and 2.3 show general relation between these two variables inside IDW[60].

$$Z^*(u) = \sum_{i=1}^{n} w_i \cdot Z(u_i) \tag{2.2}$$

$$w_i = \frac{\left(1/d_i^{\beta}\right)}{\sum_{i=1}^{n}\left(1/d_i^{\beta}\right)} \tag{2.3}$$

  Z : data on particular position vector
  u : position vector of points
  i : number of known points ($i = 1, ..., n$)
  w_i : weight for each known points
  d_i : Euclidean distance between u and u_i
  β : power to distance exponent value; default $\beta = 2$

  The weight for each known point is usually (not necessarily) limited with total sum equals to one. This kind of restriction is applied to IDW practice to prevent overwhelming influence of some nearby known data points that could lead to illogical estimation of unknown data points. Furthermore, eq 2.3 also explicitly states that the weight in IDW must be positive, which means there will be no negative contribution from neighborhood known data points. A negative contribution is useful to nullify the effect of multiple contributions from other selected data points which have similar influence to the estimated point. That is why the neighborhood of query points has to be carefully selected before it becomes the input of IDW method. A slight mistake in defining the right neighbor(s) might lead to over estimation of the estimated point(s).

- **Modified Shepard's method**
  This method is the modified version of original IDW method which lacks the localized effect of the known data point(s). Instead of using a fixed value of estimator throughout the method, the estimator is defined as local function(s) of Euclidean distance around an estimated point(s). This local function which could be based on polynomial or trigonometric functions is then estimated by the least squares method. Addition of effective radius of known point(s) (R_w) and radius for determining local neighbor(s) (R_p) terms into the method enables the local influence characterization of each known data point(s)[61]. However the fact that Rp is derived from a predetermined constant implies that local function estimation process ignores the correlation between a certain known data point and its neighbors. In the end, modified version of IDW is only able to partially solve the main issue of original IDW since the method still has the probability to select an uncorrelated local neighbor data point for its least squares process input.

- **Triangulation with Linear Interpolation**
  The basic algorithm of this method utilizes a concept of Delaunay triangulation to determine the weight of each of the three vertices which together build the triangle that contains the query point[62]. Delaunay triangulation is a group of triangles constructed from data points that satisfies following requirements:

  1. All three vertices of each triangle have to lie exactly on circumference line of the associated circle.
  2. No data point is allowed to be inside the circle.

  The sum of the three weights has to be one since there will not be any other surrounding data points to be included in the process. Even though Delaunay triangulation construction honors all known data points, it is only able to evaluate one type of distance at a time; it could be either spatial metric distance

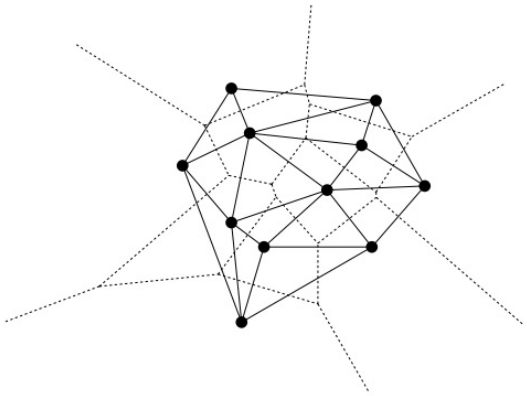


Figure 2.4: Example of Delaunay triangulation along with its dual Voronoi tessellation (dotted line)[2]

or correlation between each data point. Unfortunately both parameters are important for spatial interpolation basic algorithm therefore it is hard to determine which parameter is more suited to construct the triangulation. Nevertheless, defining fixed neighborhoods beforehand is indeed quite useful since it is able to tackle the issue regarding neighborhoods selection, and at the same time reduces overall computation time of the interpolation process.

- **Natural Neighbor method**
  Unlike previous method which defines a fixed number of neighborhoods to estimate unknown point(s), Natural Neighbor method employs Voronoi tessellation to determine which known data points are going to be in the neighborhood of the query point[62]. Voronoi tessellation is the dual of same Delaunay triangulation used on previous method.

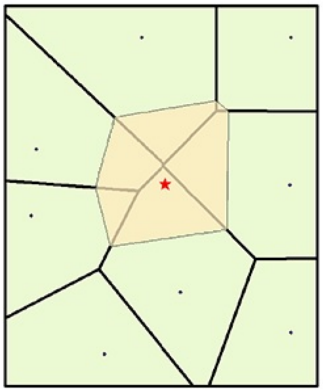


Figure 2.5: Example of 'borrowed area' within Voronoi tessellation(star symbol is the query point)[3]

Imagine a Delaunay triangulation constructed from a certain number of known data points, suddenly a query point is added into this triangulation. Due to this change, the existing triangulation has to be reformed to be able to satisfy the Delaunay triangulation requirement once again. Upon reformation, a new polygon will be formed inside the previous Voronoi tessellation. The area inside this new polygon is temporarily borrowed from its neighborhoods, that is the reason why this area is often called 'borrowed area'. The weight of each neighbor data point is then determined proportionally according to fraction of 'borrowed area' they used to hold. Since Voronoi tessellation is constructed from Delaunay triangulation, this method also suffers from similar issue of previous method.

- **Kriging method**
  Kriging method is quite popular in many fields, especially the field that is related to geospatial analysis[63]. Basically, kriging method tries to find out the most suitable trend that can be drawn from available data points by using covariance functions. The previous statement implicitly says that linear correlations

between known data points will be automatically checked during the kriging process therefore the issue regarding neighborhood selection is no longer exist. Within the kriging process, generally only two covariance terms are involved for determining the weights of each data point relative to the query point; covariance between each data point and empirical covariance function between data and query points. The latter covariance term has to be derived separately from other function that incorporates metric distance and semivariance between data points together; this function is known as variogram. The fact that kriging method is able to solve major issues of all previous mentioned interpolation methods altogether proves that the capability of kriging is generally above other spatial interpolation methods.

According to above discussion, it is clear that the common issue of many interpolation methods is how resilient each method is to the inclusion of spatially uncorrelated neighbors. This issue can be minimized by adding cross-correlation method to select the most appropriate neighborhood data points beforehand. Cross-correlation is a common procedure on some studies that focus on figuring out the effect of irradiance spatial variation to grid stability [23, 64] but not in spatial interpolation. The main reason is because the cross-correlation process requires a wide variety of data points to get a general picture of the data correlation in the spatial realm; which is quite hard to fulfill when satellite derived data is the only available source. Therefore the only option left for the spatial interpolation method is kriging which at least has a built-in countermeasure mechanism to avert the effect of uncorrelated neighborhood data points in the absence of cross-correlation process.

Moreover, Gutierrez et al have tried to compare several spatial interpolation concepts for estimating surface irradiance at some points in Spain by only relying on irradiance dataset obtained from a limited number of ground measurement stations within Spain. The validation result concludes that several variants of kriging methods show better performance among other spatial interpolation methods, considering only limited amount of sample points were utilized[65]. These two arguments become the main motivation to choose kriging as the most viable spatial interpolation method for the model.

2.2.2. KRIGING

Kriging is basically a spatial interpolation algorithm based on the regression of neighborhood data points which are weighted according to covariance values. As remainder, covariance in statistical terms describes how two random variables tend to deviate from their respective expected value; in case of discrete variables, expected value is referring to means value. The basic form of kriging equation is shown in eq. 2.4 whereas the main goal is to determine weight (λ_α) that minimizes variances of unknown real value($Z(u)$) and estimator($Z^*(u)$) difference at a certain position vector as shown in eq 2.5. The data value Z is treated as random field that is composed from addition of residual (R) and trend(m) components. Therefore it is possible to reconstruct eq 2.4 into eq 2.6 which literally shows that kriging tries to estimate residual component of $Z(u)$ by summing up all weighted residual components of neighborhood data points.

$$Z^*(u) - m(u) = \sum_{\alpha=1}^{n(u)} \lambda_a [Z(u_\alpha) - m(u_\alpha)] \tag{2.4}$$

$$\sigma_E^2 = Var\left\{Z^*(u) - Z(u)\right\} \tag{2.5}$$

$$Z^*(u) - m(u) = \sum_{\alpha=1}^{n(u)} \lambda_a \cdot R(u_\alpha) \tag{2.6}$$

u : position vector of query point
α : position vector index of neighboring data point
n : number of neighboring data points

Before we dive deeper into the kriging topic, it is advisable to understand a little bit more about kriging variant. Generally, kriging is divided into three types namely simple kriging (SK), ordinary kriging (OK), and kriging with trend which is also known as universal kriging (UK). Each type of kriging method has a different assumption about how they treat the trend component (m) within the process. The SK assumes m as a constant mean value throughout the space while OK assumes m as constant mean, which is only valid on each respective local neighborhood (local mean). Almost similar to OK, UK also assumes local m value but instead

of just using constant value, a higher order of trend model (*e.g.*linear,) is applied. Since the model within this report is still in its earlier version, only the OK method is included and explained. Other kriging methods might be added into the model on future development.

Local constant mean of trend component on OK process transforms eq 2.4 into eq 2.7 which is the most appropriate form for starting the derivation of OK process.

$$Z^*(u) = \sum_{\alpha=1}^{n(u)} \lambda_\alpha(u) Z(u_\alpha) + \left[1 - \sum_{\alpha=1}^{n(u)} \lambda_\alpha(u)\right] m(u) \tag{2.7}$$

When total weights of neighboring data points equals to 1 is defined as the constraint for eq 2.7, it simplifies the original equation into eq 2.8.

$$\sum_{\alpha=1}^{n(u)} \lambda_\alpha(u) = 1 \Rightarrow Z^*(u) = \sum_{\alpha=1}^{n(u)} \lambda_\alpha(u) \cdot Z(u_\alpha) \tag{2.8}$$

Since eq 2.8 is a multivariate function with an embedded constraint function, Lagrange multipliers concept can be used as a tool for finding the minima or maxima of this function [66]. The real function whose minima has to be found is the variance (σ_E^2) not minima of σ_E therefore constraint function shown in eq 2.8 (before arrow) will be accompanied by two Lagrange multipliers which hold similar value. The new function that already has Lagrange multipliers included in it is shown on eq 2.9

$$L(\lambda_\alpha, \mu_{OK}) = \sigma_E^2 + 2\mu_{OK}(u) \left[1 - \sum_{\alpha=1}^{n(u)} \lambda_\alpha(u)\right] \tag{2.9}$$

Partial derivative products of eq 2.9 in respect to weight(λ_α) and Lagrange multiplier(μ_{OK}) have to be determined in order to find its extreme conditions. The partial derivative of 2.9 in respect to Lagrange multiplier can be directly done since the first term of eq 2.9(right-hand side) does not contain any Lagrange multiplier; eq 2.10 shows the result of this partial derivative when it is set equal to 0.

$$\frac{\partial L(\lambda_\alpha, \mu_{OK})}{\partial \mu_{OK}} = 2 \sum_{\alpha=1}^{n(u)} \lambda_\alpha - 2 = 0 \Rightarrow \sum_{\alpha=1}^{n(u)} \lambda_\alpha = 1 \tag{2.10}$$

On the other hand, partial derivative of eq 2.9 with respect to weight(λ_α) can be quite tricky to determine since eq 2.9 may contain other unseen λ_α element. Therefore eq 2.9 has to be expanded first by using the rule of variance of a linear combination of random variables [63, 66]. Two product equations of this rule are presented on eq 2.11 and eq 2.12.

$$Var\left(Z^*(u) - Z(u)\right) = Var\left[\sum_{\alpha=1}^{n(u)} \lambda_\alpha Z(u_\alpha)\right] + Var\left[Z(u)\right] - 2Cov\left[\sum_{\alpha=1}^{n(u)} \lambda_\alpha Z(u_\alpha), Z(u)\right] \tag{2.11}$$

$$Var\left[\sum_{\alpha=1}^{n(u)} \lambda_\alpha Z(u_\alpha)\right] = \sum_{\alpha=1}^{n(u)} \sum_{\beta=1}^{n(u)} \lambda_\alpha \lambda_\beta Cov\left[Z(u_\alpha), Z(u_\beta)\right] \tag{2.12}$$

The expanded version of eq 2.10 is then obtained by substituting eq 2.11 and eq 2.12 into the original equation; it is presented on eq 2.13.

$$\begin{aligned} L(\lambda_\alpha, \mu_{OK}) = \sum_{\alpha=1}^{n(u)} \sum_{\beta=1}^{n(u)} \lambda_\alpha \lambda_\beta Cov\left[Z(u_\alpha), Z(u_\beta)\right] + Var\left[Z(u)\right] \\ - 2Cov\left[\sum_{\alpha=1}^{n(u)} \lambda_\alpha Z(u_\alpha), Z(u)\right] + 2\mu_{OK}(u)\left[1 - \sum_{\alpha=1}^{n(u)} \lambda_\alpha(u)\right] \end{aligned} \tag{2.13}$$

Upon obtaining eq 2.13, it is possible to determine partial derivative product of eq 2.9 in respect to λ_α and then find the second extreme condition by setting this product equals to 0; eq 2.14 and eq 2.15 show respectively the partial derivative product in respect to λ_α and second extreme condition of σ_E^2.

$$\frac{\partial L(\lambda_\alpha, \mu_{OK})}{\partial \lambda_\alpha} = 2 \sum_{\beta=1}^{n(u)} \lambda_\beta Cov\left[Z(u_\alpha), Z(u_\beta)\right] - 2Cov\left[Z(u_\alpha), Z(u)\right] + 2\mu_{OK}(u) = 0 \tag{2.14}$$

$$\sum_{\beta=1}^{n(u)} \lambda_\beta Cov\left[Z(u_\alpha), Z(u_\beta)\right] + \mu_{OK}(u) = Cov\left[Z(u_\alpha), Z(u)\right] \tag{2.15}$$

Both extreme conditions of σ_E^2 shown on eq 2.10 and eq 2.15 are then translated into matrix notation as presented on eq. 2.16.

$$C \times \lambda = D \tag{2.16}$$

$$\begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n(u)} & 1 \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n(u)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C_{n(u),1} & C_{n(u),2} & \cdots & C_{n(u),n(u)} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \times \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n(u)} \\ \mu \end{bmatrix} = \begin{bmatrix} C_{1,u} \\ C_{2,u} \\ \vdots \\ C_{n(u),u} \\ 1 \end{bmatrix}$$

Matrix C is the covariance function of local neighborhood data points while matrix λ holds weights and Lagrange multiplier. On the right-hand side of equation, matrix D represents the covariance vector of the query point. Unlike C which can be directly computed from available data, D has to be indirectly derived from the other geostatistical model known as variogram. Under the assumption that Z is a random field variable with stationary mean of 0 and stationary covariance (second-order stationarity), matrix D can be simplified into eq 2.17 [63].

$$Cov\left[Z(u_\alpha), Z(u)\right] = Cov\left[Z(u+h), Z(u)\right] = C(h) \tag{2.17}$$

This simplification transforms D which was originally a covariance vector of u into a covariance function of lag(h). Unlike its previous form which considers both direction and distance, the simplified function only regards the Euclidean distance between query point and neighboring data points; eq 2.18 shows the equation for computing this function.

$$C(h) = C(0) - \gamma(h) \tag{2.18}$$

The last term of eq 2.18 is an external function which is commonly used in geospatial field called semivariogram function($\gamma(h)$); more explanation about this topic is available in the next section. Since we have known how to estimate both matrices C and D, eq 2.16 can be easily solved by using simple matrix operation as shown on eq 2.19.

$$\lambda = C^{-1} \times D \tag{2.19}$$

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n(u)} \\ \mu \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n(u)} & 1 \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n(u)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C_{n(u),1} & C_{n(u),2} & \cdots & C_{n(u),n(u)} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \times \begin{bmatrix} C(h_1) \\ C(h_2) \\ \vdots \\ C(h_{n(u)}) \\ 1 \end{bmatrix}$$

In order to get the estimated value of the query point, someone just have to feed the obtained weights and Lagrange multiplier values back into eq 2.8. Please note, this process is only valid on local scale since the total of local weights equals to 1 constraint is only valid for local neighborhood data points.

SEMIVARIOGRAM

As previously mentioned, the semivariogram model holds a very crucial role in the kriging process, since it is an alternative way to estimate covariance vector of query point. Generally, the semivariogram model is divided into two variants namely, isotropic and anisotropic semivariogram. Isotropic semivariogram treats neighborhood data points based only on their lag(h) and ignores their relative direction from the query point of view, while on the other hand anisotropic variogram takes into account this detail [67]. Isotropic semivariogram model is less demanding in term of computational resources since the nature of its algorithm is less complicated compared to anisotropic semivariogram. Due to this reason, only the isotropic semivariogram model is included into the model; the anisotropic semivariogram might be added on the future development of the model.

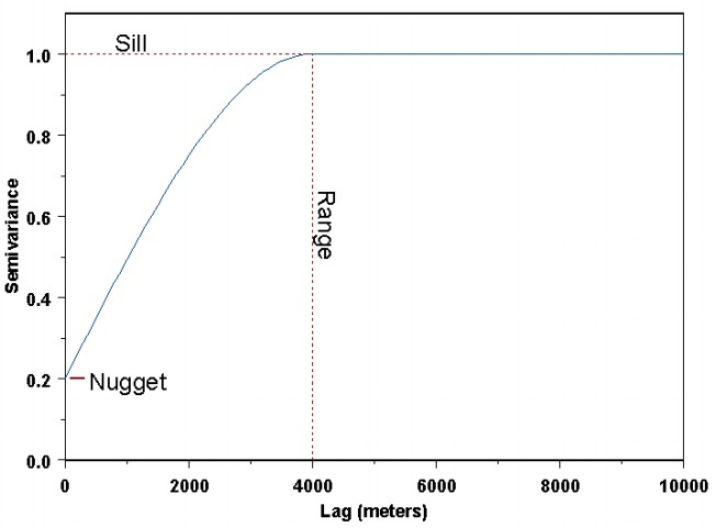


Figure 2.6: Example of semivariogram model[4]

$$\gamma(h) = \frac{1}{2N(h)} \sum_{\alpha=1}^{N(h)} [Z(u_\alpha + h) - Z(u_\alpha)]^2 \tag{2.20}$$

$$C(h) = \frac{1}{N(h)} \sum_{\alpha=1}^{N(h)} Z(u_\alpha) \cdot Z(u_\alpha + h) - m_0 \cdot m_{+h} \tag{2.21}$$

$$m_0 = \frac{1}{N(h)} \sum_{\alpha=1}^{N(h)} Z(u_\alpha) \quad , \quad m_h = \frac{1}{N(h)} \sum_{\alpha=1}^{N(h)} Z(u_\alpha + h) \tag{2.22}$$

In geostatistical terms, semivariogram explains the relation between semivariance values in regard of Euclidean metric distance commonly known as lag (h). While covariance(C) measures the similarity, semivariance(γ) itself measures spatial dissimilarity of data points; basic semivariance and covariance formulas are presented on eq 2.20 & 2.21, respectively. Under second-order stationarity condition, the semivariogram is utilized as base function to estimate covariance function.

The construction process of an empirical semivariogram is started by separating all possible data points pair combinations according to their particular lag value and then computing the semivariance for each constant lag value. If the data points are not uniformly distributed across the space, semivariance is computed by using the average value of ranged lag. Obtained semivariance values are then mapped out along with their respective lag value on Cartesian coordinates; an example of semivariogram curve is presented on fig 2.6.

Generally, the empirical semivariogram will not be as well-ordered as shown on fig 2.6, however it will still have at least some similar basic characteristics that any other semivariogram model has. These characteristics are briefly discussed on following points:

- **Sill**
  Sill term stands for the maximum semivariance value that can be achieved by semivariogram. Upon reaching this value, most of the semivariogram model starts to level off and obtains its saturation condition.

- **Range**
  Range term tells us at which lag distance the sill value is first achieved within the semivariogram. There is no clear benchmark for how large the range value can be, since it depends on the type of physical parameter that is being modelled.

- **Nugget**
  Ideally, when lag distance equals or is close to zero (measured point) the semivariance value is zero. In most actual conditions which are not ideal(*e.g.* measurement error) this rule will not hold, and 'nugget' represents this non-ideal condition within semivariogram model. Unfortunately this characteristic is

not applicable inside the model, since the built-in error of satellite-derived data covers quite a large area which in the end escalates range of semivariogram into spatial resolution value. Simply put, this makes zero to spatial resolution lag distance returns constant semivariance.

For the sake of a stochastic simulation such as kriging process, empirical semivariogram has to be translated into the commonly used semivariogram model. The first reason why kriging process requires the semivariogram model instead of empirical semivariogram is because during the kriging process the algorithm may be asked to estimate value at some positions whose lag distance are not represented by any available empirical lag distance. The second reason is because the kriging process has some numerical rules that need to be obeyed, otherwise kriging equations are not longer solvable [4]. Eq. 2.23, 2.24, and 2.25 show the most frequently used semivariogram model namely exponential, gaussian, and spherical models; only these 3 semivariogram models are included into the model. In order to select the most suitable semivariogram model for empirical semivariogram, a curve fitting process based on non-linear least squares method is utilized inside the model.

$$g(h) = c.\left(1 - exp\left(\frac{-3h}{a}\right)\right) \tag{2.23}$$

$$g(h) = c.\left(1 - exp\left(\frac{-3h^2}{a^2}\right)\right) \tag{2.24}$$

$$g(h) = \begin{cases} c.\left(1.5\left(\frac{h}{a}\right) - 0.5\left(\frac{h}{a}\right)^3\right) & \text{if} \quad h \leq a \\ c & \text{else} \end{cases} \tag{2.25}$$

$g(h)$: semivariogram function
a : range of semivariogram model
c : sill of semivariogram model

As you might expect, the selection of semivariogram model relies a lot on how accurate the curve fitting of the model is. Therefore instead of using the hold-out (single split dataset) method to validate the result of curve fitting, the cross validation method (double split dataset) is more preferable since there will be no averaging step on estimating the risk value[68]. By doing this, the selection step expects to propose more accurate semivariogram model that can be inferred from existing data. Further explanation regarding this approach is available on the next section.

CROSS VALIDATION

Cross validation (CV) is a popular algorithm to assess the result of the curve fitting process and in this case it is useful as an assessment tool to select the most suitable semivariogram model for kriging. The basic concept of cross validation is to split the existing data into 2 sets; the first set is meant for training purpose while the other one is a dataset for evaluating purpose [68]. The training process proposes the most suitable model according to the data inside the training set. Next, the proposed model is evaluated further with the data inside the evaluating set. The previous steps (training and evaluation) are repeated until all possible combinations from existing data have been covered, the final model is then selected based on the performance of each proposed model in the evaluation step (the least error).

The method by which the data is separated into training and evaluation groups in the cross validation algorithm is often known as the 'splitting scheme'. Splitting schemes of cross validation method generally can be divided into two types, exhaustive data splitting and partial data splitting. The former is considered the classical approach since it sets a fixed size of the training set throughout the entire process which is quite 'exhaustive' in term of computational resources. On the other hand, the latter implements flexible size of the training set (depending on the algorithm) which could be the alternative solution if the size of the existing dataset is quite large. Since in this case the size of the dataset is not an issue, exhaustive data splitting scheme is still feasible to be applied, and therefore there is no need to discuss partial data splitting any further on this report.

Exhaustive data splitting scheme is divided into two types of procedures, Leave-one-out (LOO) and Leave-p-out(LPO). As the name implies, LOO leaves one data point out from the existing dataset to be used for

validation, while the rest of data points will be utilized for training purposes. Similar to what LOO does, the LPO procedure also leaves a fixed amount of data point(s) out from existing dataset; however the amount should be more than one, otherwise it becomes LOO. Since the LPO procedure decreases the number of possible combinations within the CV process, it becomes alternative choice for user who intends to stick with an exhaustive data splitting scheme but does not have much computation resource. Nevertheless, the LOO procedure has been chosen to be the splitting procedure in this report due to the relatively small size of available dataset and adequate computational resources.

GSTAT TOOLBOX

The gstat was originally an un-compiled source code that could carry out geostatistical computations. This source code was then developed further into a standalone package for R platform. Package gstat is popular among geostaticians since it can handle a lot of spatial modelling such as kriging, co-kriging, Gaussian simulation and many more [69]. The mgstat is a Matlab toolbox that is adapted from the gstat R package [70]. Instead of just having the original gstat interface this adaptation is also able to handle geostatistical modelling such as kriging by using the native environment of Matlab. Due to compatibility issues, only the main kriging function is implemented inside the model, although the toolbox itself provides other functions related to kriging such as variogram modelling; the rest of algorithm was built from scratch on the Matlab environment.

2.3. K-MEANS CLUSTERING ALGORITHM

In order to approximate the optimum placement position of irradiance sensors, the dataset generated by spatial interpolation has to undergo a couple of other statistical processes and one of this processes is data clustering. Several prior related studies have tried to incorporate different data clustering algorithms such as k-means [43, 44, 46] and quadtree [45] into their main analysis. Even though both algorithms utilize two completely different approaches, both of them give more or less similar result types, which proves that data clustering process is a proper tool to estimate the effect of spatial variations on irradiance measurement.

The main objective of data clustering is to infer meaningful information from the dataset by splitting it up into a certain number of clusters. Each cluster holds at least one data point whose value can be used as a representation for entire data points within that particular cluster; this data point is called the cluster's centroid. In the model, this centroid acts as a real irradiance sensor which records irradiance value at every temporal resolution time step. Therefore the clustering process inside the model is only meant to predict the locations of these centroid points. There are 3 simple algorithms that are commonly used for clustering purposes; k-means, agglomerative hierarchical clustering and DBSCAN. The following points briefly describe how each algorithm works to split up the dataset:

- **K-means**
  The k-means algorithm uses the data centroid concept to split up the dataset into a number of clusters whose value is set by the user. The centroids attract nearby data points (based on the distance) to create their own clusters. After that, the algorithm recalculates the new centroids and once again assigns all the data points respectively onto the nearest centroids. This process is repeated until stable centroid values have been achieved.

- **Agglomerative Hierarchical Clustering**
  Agglomerative Hierarchical clustering starts the process by assuming that all data points are singleton clusters. The singleton clusters are then paired up accordingly based on the chosen linkage type and their respective distance. The linkage term refers to the criteria of distance for merging one cluster with another; *e.g.* 'shortest distance' and 'average distance'. This pairing process continues until a complete cluster tree diagram, commonly known as dendrogram has been built. Next someone just has to decide where this dendrogram should be cut to create clusters that satisfy the needs.

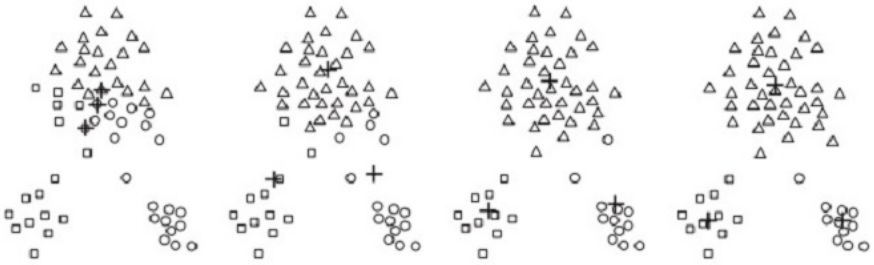
- **DBSCAN**
  Unlike the other two algorithms which based their analysis on similarity between data points (the distance), DBSCAN focuses its algorithm on the density of data points over the dataset. Before the main algorithm starts, someone has to define two main parameters namely distance (Eps) and threshold parameters($MinPts$). The algorithm then labels all data points as either core, border or noise points. The point has a right to become a core point when the number of surrounding data points within the distance parameter radius (Eps) exceeds the threshold parameter($MinPts$). The data point which is

located within the core points neighborhood is known as the border point; it might be located in between several core points. Data points which do not fall to either core or border categories are considered noise points or outliers and will be left out during the main process. The main DBSCAN algorithm starts the process by searching for pairs of core points whose distance is still within the Eps radius. These pairs along with its members are then put up into the same cluster. Similarly, border points are also assigned onto the closest core point's cluster. By doing this, the number of clusters is automatically defined by the algorithm without any user interference.

In the end, k-means has been chosen to be the most suitable clustering algorithm for the model mostly due to the fact that it focuses its analysis on the cluster's centroid which is also the main objective of the clustering process inside the model. The second reason is because of its efficiency, k-means demands less computational resource compared to the other 2 algorithms, especially compared to hierarchical clustering algorithm which computes everything without a clear objective [5]. The last reason is because k-means is a flexible algorithm whose process can be easily adjusted by the user depending on their general preferences. For instance, one of sub-objective of the model is to see the influence of additional irradiance sensors to the final uncertainty of measurement. This objective requires the clustering algorithm to be run on several different numbers of clusters which is not an issue at all for k-means. However, it is slightly harder to do on hierarchical clustering algorithms, and impossible to do on the DBSCAN algorithm.

The main drawback of k-means is, the algorithm is very susceptible to outliers, and when it happens the clusters centroids will stick on this trap made by outliers and obviously cease the overall clustering process [5]. Therefore several trials of k-means are needed to obtain a good clustering product. The next drawback of k-means algorithm is related to how the cluster's centroid is determined. The centroid is computed every time a new cluster has been formed and this computation is basically only a simple process like averaging or finding the median from data points within that cluster.

We know that both the mean and median of a certain dataset only compute statistical parameters that represent a dataset; they are not real data points within the dataset. It is indeed an issue, since by applying the data clustering process the model expects to find a recommendation of actual data points that are suitable to become measurement points (centroids) of the solar park area. Inside the model, this issue is prevented by assigning actual data points which resemble computed means as the centroid.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Figure 2.7: Illustration of k-means algorithm [5]

A complete illustration of the k-means algorithm when the chosen number of cluster equals to 3 is presented on fig 2.7. From the figure, it is clear that the algorithm starts the process by randomly putting the centroids within the dataset. Each data point is then assigned to a particular cluster which the closest centroid belongs to. The closest centroid is defined by the help of 'distance' parameters between each data point and centroid. The 'distance' term in the data clustering algorithm refers to a certain type of parameter that represents the similarity of each data point to its respective cluster's centroid; further discussion regarding 'distance' is available in the next section. Upon successfully assigning all data points onto available clusters, the k-means algorithm will recompute either mean or median (depending on the distance type) values from each cluster. Data points whose value resemble either the new means or medians will be then appointed as the new centroid for each cluster. Each data points is then once again assigned onto a cluster which is currently holding the closest centroid. These steps are repeated until stable centroids condition similar to what has shown on fourth iteration of fig 2.7 has been achieved.

2.3.1. DISTANCE

As mentioned previously, the distance term determines the similarity of each data point to its respective cluster's centroid. There are several ways to interpret this similarity and the following points discuss a little about several variants of the 'distance' parameter:

- **Squared Euclidean**
  As you might expect, squared Euclidean distance determines the similarity by computing the mean squared error(MSE) of each data point separately; the formula is presented on eq 2.26. During the process, the centroid of each cluster will be represented by the local mean value of data points within this cluster. The quality of clustering process is determined by how small the sum of the squared Euclidean distance over the cluster is. A smaller sum of distance means a better clustering product, since sum of distance implicitly describes the dissimilarity of data points to their centroid. A smaller value means a better centroid to represent all data points within that particular cluster.

$$d_{st} = (x - c)^2 \tag{2.26}$$

- **Manhattan**
  The second variant of the 'distance' parameter is called Manhattan distance, also known as cityblock distance. Similar to squared Euclidean distance, Manhattan distance also minimizes the error of data points with regard to their respective centroid. However instead of using MSE concept, the distance uses mean absolute error (MAE) concept; the formula is presented on 2.27. During the process, the cluster's median data point will be acting as a centroid for that particular cluster.

$$d_{st} = |x - c| \tag{2.27}$$

- **Cosine**
  Cosine distance determines the similarity between each data point and its centroid by using the cosine similarity concept. Basically, cosine similarity only measures how similar the two non-zero vectors are by computing the cosine of the angle between these two vectors which in this case will be the data point and its centroid. As remainder, the cosine of two vectors is obtained by dividing the dot product of these vectors with the length of both vectors; complete formula of this distance is presented on 2.28. Contrary to the two previous distances, a larger sum of cosine distance shows better clustering product since cosine distance directly measures similarity, not error, which implicitly means dissimilarity.

$$d_{st} = 1 - \frac{xc'}{\sqrt{(xx')(cc')}} \tag{2.28}$$

Out of these three common distances, the squared Euclidean distance has been chosen to be the main distance type for k-medoids algorithm within the model due to the fact that it can minimize the effect of outliers [71]. Furthermore, the input data for the model is not a high-dimensional data which requires vector based computation like cosine distance that has been proven to be quite demanding in term of computational resources [71].

2.4. GREEDY SEARCH ALGORITHM

The next important statistical analysis that is incorporated into the model is the searching algorithm. This algorithm is required to find the right combinations of data points returned by the data clustering process. Several combinations of centroids are needed to see how different measurement points configurations affect the final uncertainty of irradiance measurements. Due to its simplicity, greedy search algorithm has been selected to be the main algorithm for this process.

The greedy algorithm is a straight forward searching algorithm which bases its decision on immediate advantage regardless what the future consequences might hold. The algorithm builds up the right combination piece by piece and always selects the next piece which offers the most immediate benefit for the combination. Even though this algorithm seems quite reckless, for some cases the optimum combination can really be achieved [72]. The trade-off between using much less computational resources and the risk of obtaining a sub-optimum combination is something that the user must aware of before implementing the greedy algorithm.

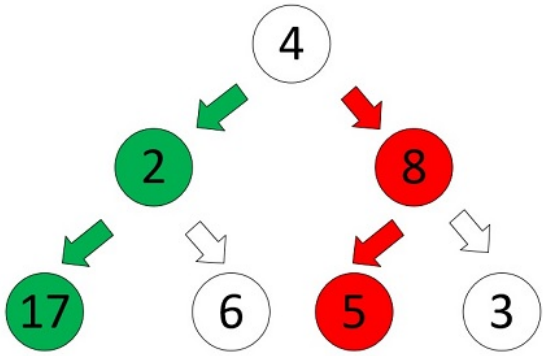


Figure 2.8: Illustration of greedy algorithm

The illustration of how the greedy algorithm works in a case in which the objective function of algorithm is to find the most optimum combinations with the highest sum is presented on fig 2.8. As you can see, the red route is the choice which the greedy algorithm would take to solve this problem. Unfortunately this solution is not optimal, since the green route gives a combination with a higher sum; therefore in this case, the solution proposed by the greedy algorithm only satisfies the objective function in local scale. This example also intends to give the reader an idea that even though the greedy algorithm might not propose a global solution for a given objective function, in the worst case it is still able to proposes solution that valid in local scale. Further explanation regarding how exactly this algorithm is implemented inside the model environment is available in the next chapter.

2.5. UNCERTAINTY DETERMINATION

Uncertainty plays a very important role in the model since both the data input and the output have either direct or indirect relation to uncertainty. Originally, uncertainty value is only used to quantize all the doubts that are presents during the actual measurement of a certain physical parameter. Nowadays, when many types (*e.g.* physical and statistical) of modelling are enabled due to fast development of computers, the uncertainty concept is also utilized to assess the quality of a model's result. In general terms, uncertainty is derived from the standard error (σ) of the result (either model or actual measurement), but unlike standard error which only holds magnitude of a certain parameter, uncertainty holds additional parameter known as the confidence interval which actually describes how certain someone can expect to find the real value within the range of result $\pm k \cdot \sigma$ [73].

This confidence interval is actually derived from stochastic analysis which assumes a certain parameter as random value that has its own probability density function(PDF). Normal distribution is utilized as the PDF to derive the confidence interval and coverage factor (k) of nowadays uncertainty value. The value for coverage factor (k) follows the value of confidence interval as shown of fig 2.9. This basic concept will be used in further uncertainty analysis within the model.

There are two involvements of uncertainty analysis within the model. The first is the built-in uncertainty of satellite-derived irradiance data which is acting as the main input for the model, and the second is the final output of the model itself which estimates the uncertainty of measured points within the solar park. Each of these uncertainty analyses is discussed in detail in the following points:

- **Built-in Error of Input Data**
  The built-in error of satellite-derived data can be directly incorporated into the model by applying all necessary rules of error propagation into each mathematical processes inside the model [74]. However, this approach is very hard to conduct if it is not properly planned beforehand. There are 2 main reasons for this statement; the first is related to the fact that error propagation analysis requires a built-in error value of all external data that are fed into the model. Apart from satellite-derived irradiance which has been validated, other minor inputs such as metric spatial resolution and GPS coordinates of the solar park still do not have any standard error that could be applied to the model. The second reason is how the model is composed from a lot of uncommon mathematical operations which makes error propagation analysis for the whole model very complicated. Due to these two reasons, the inclusion of

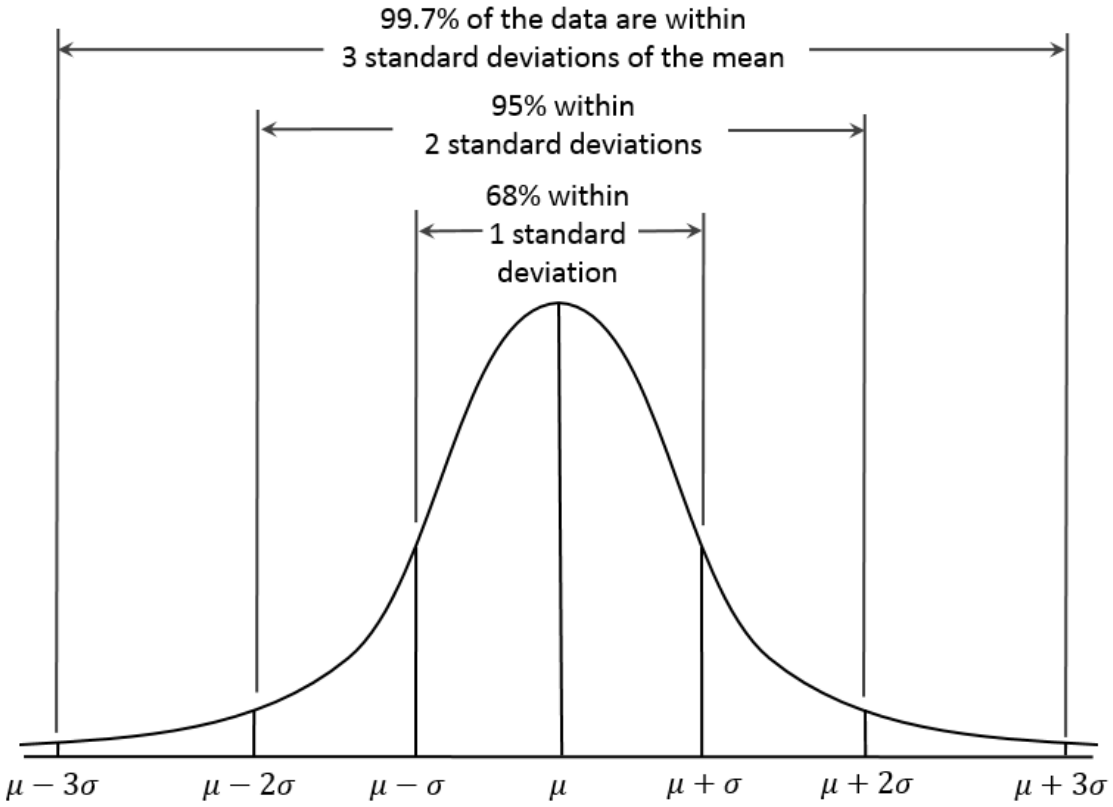


Figure 2.9: Illustration of confidence interval within normal distribution [6]

built-in error of input data is done through stochastic simulation of the original data.

The basic idea of this stochastic process is to simulate the true value of satellite-derived irradiance data by using root mean square error (RMSE) obtained from the validation result. If satellite-derived and true irradiance values are assumed to be random values that follow normal distribution function then both values most likely lie around the centroid of each respective PDF as shown on fig 2.10. As you might notice, the true value does not coincide with the satellite-derived value due to the existence of built-in error, however it still lies within the range of 95% confidence interval of the satellite-derived value. Boundaries for this interval are usually computed from standard error(σ) but in this case it is replaced by RMSE. The simulation tries to guess the true value by utilizing eq 2.29; where rnd is a floating random number within $-1 \leq rnd \leq 1$ range.

$$irr_{true} = irr_{sat} + B \cdot rnd \leftarrow B = 1.96 \cdot RMSE \tag{2.29}$$

In order to keep the original RMSE value unchanged after the input data modification, an array composed from j number of rnd values is first created. The j represents number of non-clear sky conditions throughout the original dataset; in this stochastic simulation, the clear-sky condition is assumed to be free from error. Furthermore, this array is built by applying the concept of uniform probability with zero bias which makes sure that rnd values within the array are uniformly distributed from -1 to 1. By this process, the model is able to modify the original dataset into a simulated dataset that carries a built-in error on it. The main drawback of this simulation is the fact that additional random component (rnd) might bring satellite-derived value further from the true value instead of the other way around. As a countermeasure, the model creates several sets of modified data instead of just one.

- **Estimated Uncertainty of Measured Points**
  The second uncertainty analysis is related to how the final uncertainty of the model should be presented. Nowadays, it is common practice to use the average of several irradiance measurement points as a representation of true irradiance value within the solar park. Therefore the model tries to find how

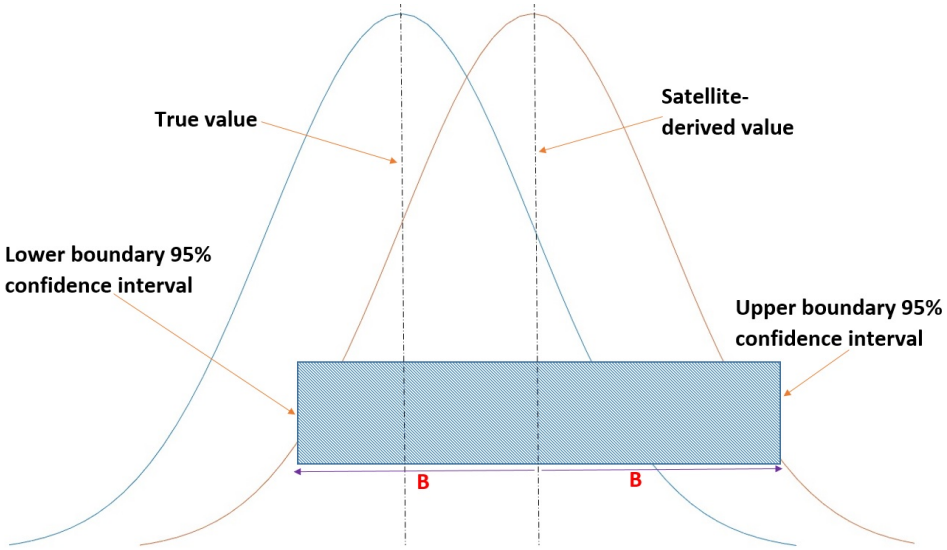


Figure 2.10: Illustration of stochastic simulation of satellite-derived irradiance data

certain this average value is when it is compared to the true average value throughout the solar park.

$$dev_{rel} = 1 - \frac{\frac{1}{N}\sum_{\beta=1}^{\beta=N} x_\beta}{\frac{1}{n}\sum_{k=1}^{k=n} x_k} \tag{2.30}$$

x : irradiance data point
k, β : position vector of data point
n : number of measurement points
N : number of data points within the data array of solar park

The process of deriving the uncertainty value is started by computing the relative deviation of average irradiance value from several measurement points with regards to true average irradiance value as shown on eq 2.30. As can be inferred from eq 2.30, the true irradiance average is estimated by integrating all irradiance values within the solar park whose total area equals to A and at the end, the integration result is normalized by A to return it to its original unit; eq 2.30 shows the discrete version of the formula.

Since each relative deviation is only meant for a certain time, the process is repeated until a year's relative deviation values have been acquired. These values are then gathered together to build a histogram whose mean and median values are assumed to be zero. Since it is uncertain what kind of PDF the histogram resembles, the final uncertainty will not be estimated through the common normal distribution approach. A certain study presents a theoretical approach to estimate uncertainty of the result (measurement or model) when the PDF does not resemble normal distribution [75]. This approach is quite practical to be implemented within the model however since the mean of the histogram is assumed to be zero (the actual mean is slightly less or more than zero), there is another method that even more practical for the model. This model involves built-in ECDF (Empirical Cumulative Density Function) inside Matlab which has been selected to be the base environment for the model.

Basically, ECDF translates the histogram into an array that contains x-axis positions and cumulative probability values on each of these positions. Next the model only has to employ a simple script that utilizes imaginary expandable lower and upper boundaries around an array created by ECDF. During the process both boundaries are simultaneously expanded to their respective directions, starting from the mean position (x=0). Every expansion step, the script computes probability density within lower and upper boundaries. The expansion stops when probability density value reaches the preference confidence interval value (the default is 95%). The x-axis position of both boundaries show the uncertainty value of this particular one year CSI dataset.

3

Model Architecture

This chapter explains the core algorithm of the statistical model one step over time, and hopefully it is able to give the reader a basic idea of how this model works. The purpose of some common built-in functions within the algorithm will be explicitly mentioned in the text. However, a more detailed explanation regarding how these functions work will not be provided. On the other hand, a basic algorithm of customized functions will be elaborated in some detail to help the reader gain a deeper understanding of the model.

3.1. Main Parameters and Assumptions

Before starting the main topic of this chapter, it is important to discuss some issues related to how the model works. The first issue of course is related to the main parameters of the model. As implicitly mentioned in a previous chapter, the input data array consists of Global Horizontal Irradiance (GHI) and Clear-sky GHI which are derived from satellite images and radiative transfer model. The data array contains yearly data with each respective temporal resolution (depending on the database). Plane of Array (PoA) irradiance is indeed more preferable to be the main input for the model instead of GHI since PoA really represents the actual incident irradiance on PV panels. Unfortunately, satellite derived PoA irradiance data suffers from relatively high error (both bias or random) when compared to satellite derived GHI. Since the chosen data will become the main input of the model, it is reasonable to pick up the data with the least built-in error (GHI) in the first place.

At the beginning of the model, GHI and clear-sky GHI are combined into a single parameter known as the clear sky index of GHI which from now on in this report will only be referred to as CSI. This CSI is the main parameter inside the model that is passed around from one process into another, until the model finally concludes the process by returning the estimated uncertainty of irradiance measurements. Additional parameters that may come along during a certain process will be mentioned specifically in each respective section.

Other important information regarding the model that has to be known is related to general limitation and assumption of the model. The following points describe briefly all these assumptions and limitations and how they may influence the final result of the model:

- **No shading from surrounding**
  The model assumes that there is no major obstacle around the area that is being investigated which may caused recurring shade from time to time due to sun movement. The inclusion of this assumption indeed simplifies the problem since the output returned by the model completely reflects the condition of the sky.

- **No soiling effect on the device that represents measurement point inside the model**
  The soiling effect refers to a condition where some small materials (mostly dust or sand) stick to the glass cover of measurement device. As you may expect, this condition increases the reading error of the measurement device since some surface area that used to contribute for the reading is being covered by an obstacle. In the report, the measurement device is assumed to be free from this sticky stuff all year long, and therefore the induced error from this issue can be neglected.

- **No measurement error under uniform sky condition**
  Uniform sky condition refers to both clear-sky and uniform cloudy-sky conditions. Inside the model,

both sky conditions are assumed to be free from measurement error due to spatial irradiance variation. Due to this assumption, a particular solar park regardless of its size only requires one measurement point to read the true value under both conditions. As you may see later in the report, clear-sky condition is also exempted from built-in error stochastic simulation, which also means that there is no difference between actual measurement reading and satellite estimation of irradiance within the model. This assumption is not completely true, since all clear-sky models that are being used by current solar databases also have their own error. The exemption of clear-sky condition from built-in error simulation is necessary to minimize the possibility of some data points to hold CSI data larger than one.

- **Each pixel is represented by its geometry centroid** Inside the model, a pixel which indicates a portion of area on the solar park is only represented by its 2D geometrical centroid. The 2D geometry of each pixel itself is assumed to be a perfect square whose centroid location is easy to define.

- **Limited to medium to large scale PV application**
  The main limitation of the model is related to the fact that it is only applicable to PV system which involves a relatively large area. The best spatial resolution for current satellite imager is around 4 km x 4 km while the largest solar park that is available nowadays only covers more or less an eighth of the area resolution of the imager. It can be imagined that a small scale solar park which covers 0.01-0.02 km^2 is only a tiny component inside a single pixel of satellite image that holds only one data. In this case, further statistical analysis (like the one in the report) can still be conducted but will be missing some details that cannot be explained by large spatial resolution data alone. In the end it is decided that only mid-large size of PV system is allowed to be analyzed by the model.

3.2. OVERVIEW OF THE MODEL

The backbone algorithm of the model presented on fig 3.1 shows that the model is composed of six main processes, which most of the processes are statistical analysis except data preparation. The process is started by preparing the raw data into usable input data, and then the built-error which was originally embedded into the data are imaginary removed from the data through stochastic simulation process. Upon removing this error, the data is then used as base for generating new data points through a spatial interpolation process in order to enlarge the spatial data resolution within the solar park. After getting a sufficient amount of spatial data points, the clustering process is enabled to filter the data points which have a high probability of being the right measurement point positions for one year's data period. In the end clustering process will give out some recommendations regarding the positions of measurement points which allows the next process to step in for tailoring several possibilities of measurement point spatial configurations. Finally these configurations are set as fixed configurations of measurement points for one year period and the measurement uncertainty due to spatial variations can be approximated. Detailed explanation of each process is available in the next sections.

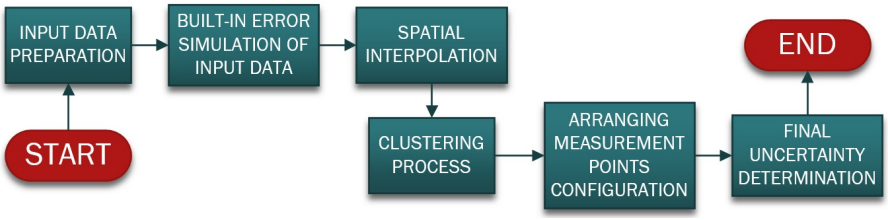


Figure 3.1: An overview flowchart of the model

In addition, since not all the reader understand the general rule of how flowchart is read within this report, following section is made to give a brief introduction about several common flowchart symbols used within this report.

Legend of fig 3.2 :
1 : it represents either the beginning or the end of a certain procedure
2 : it represents either the input or output data of a certain process block
3 : it is a decision block that decides which direction the procedure should take depending on the condition
4 : it is a process block that acts according to written order on the block

Figure 3.2: Flowchart common symbols

5 : in the report this symbol particularly represents the data storage during some looping procedure
6 : this symbol is acting as an intersection block
7 : this symbol is acting as off-page connector between one flowchart and the other
8 : this symbol is acting as on-page connector between one procedure and the other within a single flowchart

3.3. DATA INPUT PREPARATION

Data input preparation is started by asking the user to manually insert several parameters that have to be defined before other process can be initiated; the following points show these particular inputs:

1. Corner GPS coordinates (WGS84 format) of solar park geometry
2. Confidence interval target of several processes which utilize probability density function concept
3. Which solar database is going to be used for this particular case
4. How large is the temporal resolution of the dataset
5. How many configuration trials the model has to perform
6. What is the maximum amount of measurement points to be included in the analysis
7. How many error-simulated datasets the model has to built
8. How large of a threshold value (in CSI), that determines whether a dataset belongs to a uniform or non-uniform sky condition. It is a very important parameter since without this, spatial interpolation on some datasets cannot be done due to small spatial variations in the data

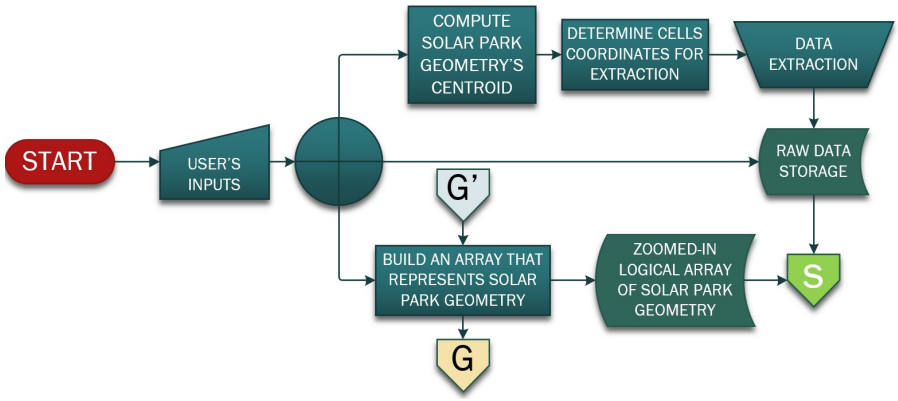


Figure 3.3: Flowchart showing how input data for the model is prepared

There are two parallel processes that have to be done in this step, the first is to extract the data manually from the on-line solar database, and the second is to build a logical array that represents 2D geometry of the solar park. In order for the user to extract the data, a set of pixel coordinates has to be provided first by the model, and computing the 2D centroid of solar park geometry is the first step to do it. The 2D centroid is basically computed by using a built-in function called 'polygeom' which runs on the Matlab environment; a more detailed explanation regarding the algorithm is presented in its manual [76]. Upon obtaining the

centroid coordinate, the model moves on to other process that decides which pixels within the on-line solar database have to be extracted for the input of the model. The selection is started by seeking a pixel that contains a centroid point in it, and then assigns it as centre pixel whose data has to be extracted. The next step is simply to assign the nearest 8 surrounding pixels into the list of dataset that have to be extracted; the illustration of the pixels configuration is shown on fig 3.4.

PIXEL 1 (X_{C-1},Y_{C+1})	**PIXEL 4** (X_C,Y_{C+1})	**PIXEL 7** (X_{C+1},Y_{C+1})
PIXEL 2 (X_{C-1},Y_C)	**PIXEL 5** (X_C,Y_C)	**PIXEL 8** (X_{C+1},Y_C)
PIXEL 3 (X_{C-1},Y_{C-1})	**PIXEL 6** (X_C,Y_{C-1})	**PIXEL 9** (X_{C+1},Y_{C-1})

Figure 3.4: Cells numbering within the model of manually extracted dataset

Upon finishing the pixels configuration, the model will compute the centroid coordinates for each pixel and return it as reference coordinates for the user to extract the data from the on-line solar database. The user is asked to extract both GHI and clear-sky GHI for each of the nine pixels and then arrange them together into a standard format for raw data array as shown in the following matrix.

$$\text{raw data} = \begin{bmatrix} I_{1,1} & I_{2,1} & \cdots & I_{9,1} \\ I_{1,2} & I_{2,2} & \cdots & I_{9,2} \\ \vdots & \vdots & \ddots & \vdots \\ I_{1,t} & I_{2,t} & \cdots & I_{9,t} \end{bmatrix}$$

In this array, each pixel irradiance dataset is represented by a single column while a single row represents the irradiance dataset of solar park area and its surrounding in a certain time. The 'I' term in the above matrix stands for either GHI or clear-sky GHI values on each data point. While the 't' term represents the length of a year dataset in the temporal realm; the t value varies depending on which temporal resolution had been used to extract the data. Lastly, the numbering rule of pixels in that array strictly follows pixels numbers shown on fig 3.4; note that the centre pixel is located in the fifth column of the array not the first column.

3.4. INTERPRETING SOLAR PARK ACTUAL 2D GEOMETRY INTO AN ARRAY

The second parallel process in the data preparation block is to create a logical array that represents 2D geometry of the solar park. The sub-procedure that is called by the main data preparation procedure for completing this task is presented on fig 3.5. As can be seen from the flowchart, this procedure is started by decomposing the nine pixels area with relatively low spatial resolution into an array of pixels with high spatial resolution. For the previously explained reason, the high spatial resolution within the model is set to be constant at 100 m.

Next, the model will define the centroid position (in metric Cartesian coordinate) of each newly created pixel. Each of these pixels are then divided further into four smaller temporary pixels with equal size as presented on fig 3.6. The coordinate of these temporary pixel centroids are then put together into a coordinates array along with the original pixel's centroid; on fig 3.6 the original pixel's centroid is indicated in blue, while the green points represent temporary pixel centroids. This process is conducted on every original pixel with no exception.

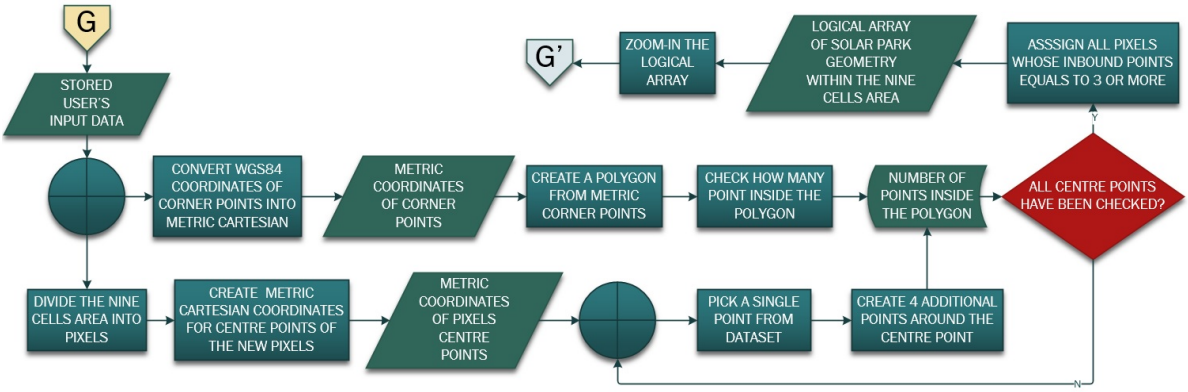


Figure 3.5: Flowchart regarding how geometry logical is created within the model

$$\text{raw logical array} = \begin{bmatrix} b_{1,1} & b_{2,1} & \cdots & b_{n,1} \\ b_{1,2} & b_{2,2} & \cdots & b_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1,m} & b_{2,m} & \cdots & b_{n,m} \end{bmatrix}$$

$$n = \frac{Resx_{lo}}{Resx_{hi}} \quad , \quad m = \frac{Resy_{lo}}{Resy_{hi}} \tag{3.1}$$

In a separate process, the model also converts the WGS84 coordinates of the solar park 2D geometry corner points into metric Cartesian coordinates and then builds a polygon of the solar park out of it. The previously created centroids Cartesian coordinates of original and temporary pixels are then checked by Matlab's built-in function known as 'inpolygon' to see how many out of these 5 points are located inside the polygon of the solar park. When the amount of points is equal to 3 or more, that particular pixel is deemed to be an active pixel which holds a value of one on logical array; otherwise it is considered a passive pixel which holds a value of zero on the logical array. Similarly, this process is also conducted for every pixel with high spatial resolution. The created logical array resembles the one shown on eq 3.1 where *Res* term stands for spatial resolution of each Cartesian axis. As the name implies this array is only allowed to hold either 0 or 1 (binary) for each of b elements.

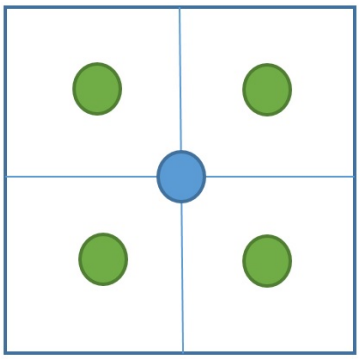


Figure 3.6: Illustration of four additional points around centroid of each pixel

Upon obtaining this array, the model continues to the next process which intends to zoom-in to the logical data array in order to reduce memory usage during the process. It is a simple process which involves several matrix operations of the original logical array to make it into a more compact array that minimizes the number of idle elements in it. The array below shows what the zoomed-in logical array looks like. Since it is created from the original logical array, the original indices of each element are still attached. Due to this, i&k indices cannot be less than one and similarly, j&l also cannot be higher than m&n, respectively.

$$\text{zoomed-in logical array} = \begin{bmatrix} b_{i,k} & b_{(i+1),k} & \cdots & b_{(j-1),k} & b_{j,k} \\ b_{i,(k+1)} & b_{(i+1),(k+1)} & \cdots & b_{(j-1),(k+1)} & b_{j,(k+1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{i,(l-1)} & b_{(i+1),(l-1)} & \cdots & b_{(j-1),(l-1)} & b_{j,(l-1)} \\ b_{i,l} & b_{(i+1),l} & \cdots & b_{(j-1),l} & b_{j,l} \end{bmatrix}$$

Along with raw data extracted by the user previously, the zoomed-in logical array becomes the main output arrays of the data preparation process.

3.5. SIMULATION OF BUILT-IN ERROR INTO DATA INPUT

As mentioned previously, this process is meant to simulate original satellite-derived data into a dataset that carries no error (the true value). Simply put, it tries to remove the built-in error of the original dataset. Since it is a simulation process, the result may or may not resemble the true dataset value, even though the basic statistical characteristic of the original data has been followed. In order to increase the probability of obtaining datasets that resemble the true dataset, several simulated datasets are created. Nevertheless, fig 3.7 shows how this procedure is generally conducted inside the model. The procedure is started by computing Clear Sky Index (CSI) for every satellite pixel according to eq 2.1 and then filters the CSI dataset from nocturnal data points. Within the CSI array, nocturnal data points are indicated with Not a Number(NaN) elements as a result from zero to zero division. Upon obtaining all data points for day conditions, the model splits the dataset into two groups, namely clear-sky and non-clear sky conditions. Inside the CSI array, the clear-sky condition is indicated by a row of data points whose total sum equals 9 or in other word, CSI on every satellite pixel is equal to 1. The clear-sky condition dataset will remain as it is while the non clear-sky condition dataset will be inserted into simulation sub-procedure for further processing.

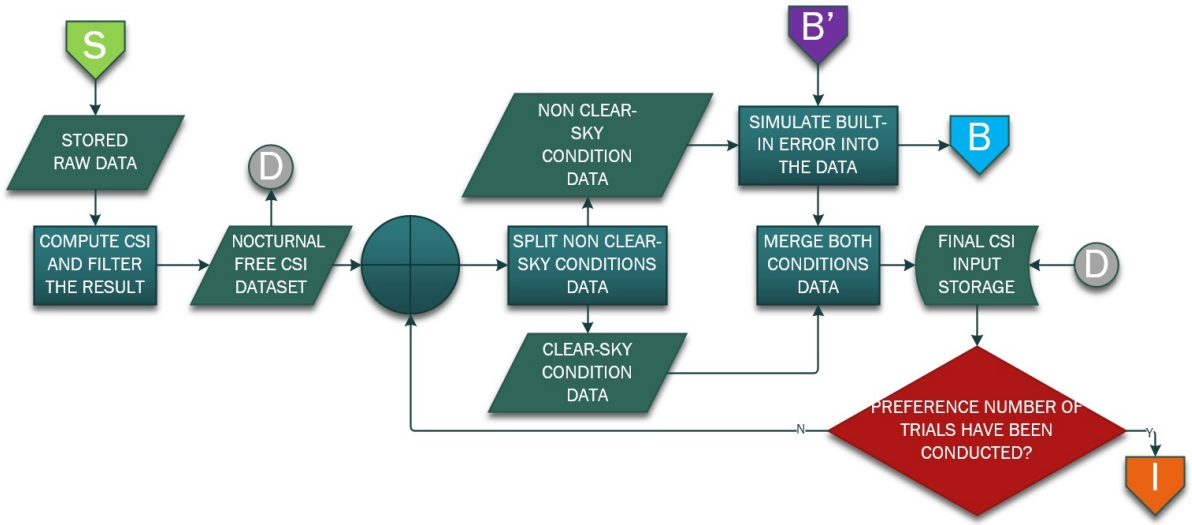


Figure 3.7: Overview flowchart of how built-in error is embedded into input data

A detailed process within error simulation sub-procedure is presented on fig 3.8. The main idea of this procedure is to generate random components whose statistical parameters follow the original dataset and then incorporate it into the original dataset. The model firstly determines how many random components have to be built. This can be done by finding the number of non-clear sky conditions within the dataset. Next, the model generates uniformly distributed random floating numbers between -1 to 1. Each of these random numbers is then incorporated into each respective temporal non clear-sky condition data point of a certain satellite pixel according to eq 2.29. By doing this, the modified CSI dataset will forcefully follow the temporal RMSE value of satellite dataset. This process is repeated for the other 8 satellite pixels.

The output of the simulation sub-procedure is then returned back into the main procedure to be recoupled once again with clear-sky condition dataset for creating a complete modified CSI dataset. Lastly, several sets of modified CSI array will be merged together along with the unmodified CSI array into a 3D array as the output for the error-simulation procedure.

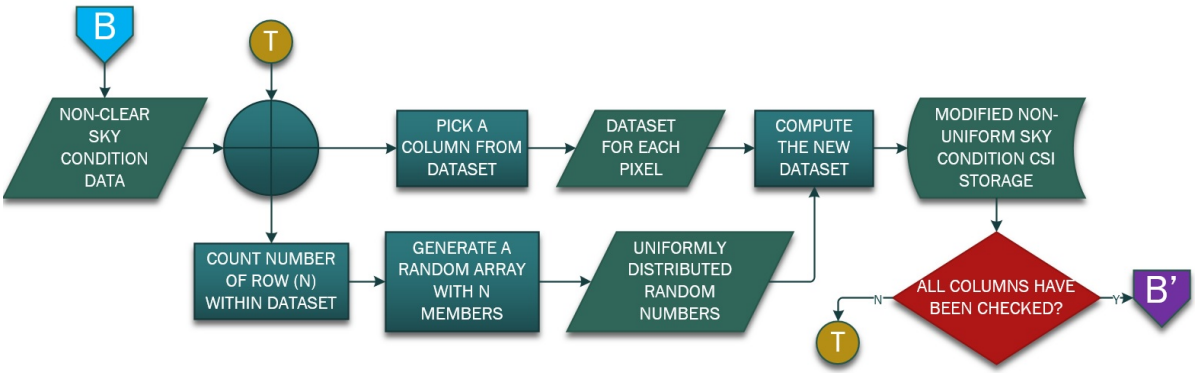


Figure 3.8: Flowchart regarding how built-in error is simulated under non-clear sky condition

3.6. SPATIAL INTERPOLATION OF DATA INPUT

As mentioned previously, spatial data interpolation is utilized by the model to increase the spatial resolution of satellite-derived data. As can be seen from fig 3.9 the model starts the procedure by taking a row of data points from the CSI dataset (either modified or unmodified) which is basically a dataset that represents CSI values for a whole 9 pixels area in a certain time. After that, the model will decide whether this dataset belongs to a uniform or non-uniform clear-sky condition. The main indicator to make the decision is simply the standard deviation of the dataset. When standard deviation of the selected dataset is less than the error threshold, the dataset is deemed to be in uniform sky condition; otherwise it is considered to be non-uniform sky condition. The default error threshold value inside the model is set to be 0.005 in CSI terms. This threshold is necessary to filter out the near-uniform sky condition dataset from entering the main kriging process. The reason is because near-uniform sky condition only shows a slight spatial variation within the dataset, and this kind of condition is not acceptable for the kriging process which bases its estimation mainly on semivariance between data points. Instead of completely excluding near uniform sky condition datasets from the analysis, it is decided that near-uniform sky condition within the limit of error threshold will be treated as uniform sky condition.

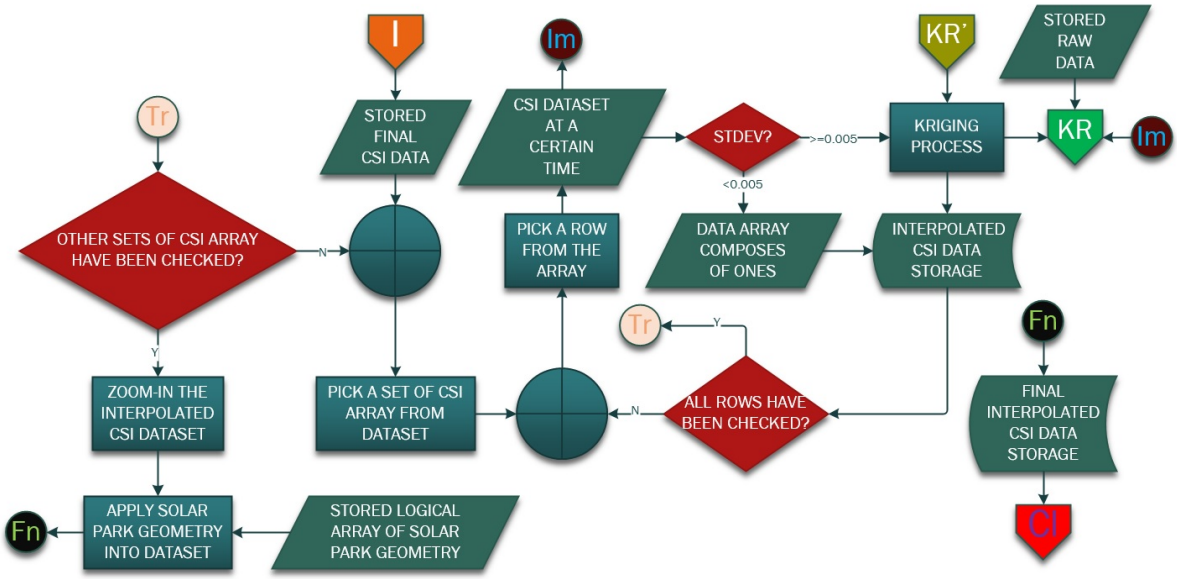


Figure 3.9: Overview flowchart of how spatial interpolation is done within the model

The model will directly return an array consisting of ones when the uniform sky condition dataset is detected, otherwise it will first pass the dataset into kriging sub-procedure for further analysis. Please note that an array of ones does not necessarily represent interpolated CSI values under uniform sky condition; CSI values under uniform cloudy condition are similar (nearly) but not ones.

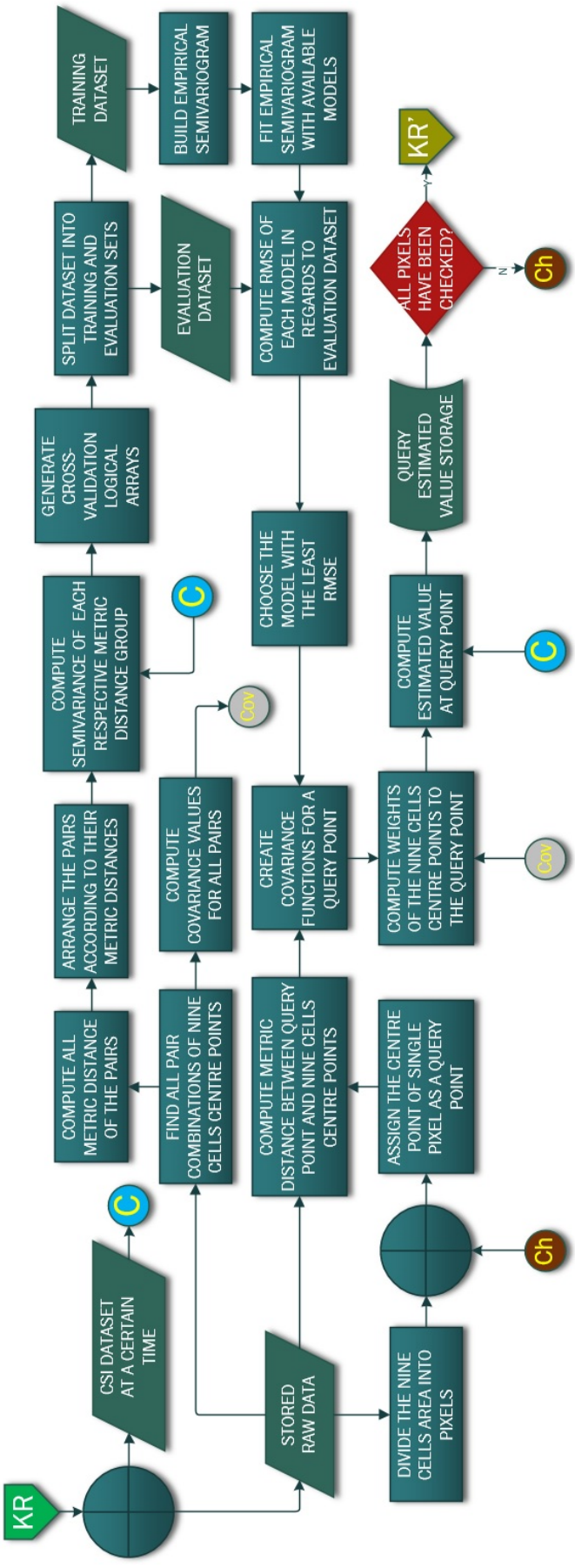


Figure 3.10: Kriging algorithm flowchart

Detail of the kriging sub-procedure algorithm is presented on fig 3.10 and as can be seen it is quite an intensive procedure that involves a lot of processes. Before the whole procedure can be started, the model has to call back an array composed of centroid positions of decomposed pixels. Even though the main objective of kriging process is to estimate all CSI values of these decomposed pixels (query points), kriging is only able to process one query point at one time. Therefore as a solution, the model will repeat the kriging process several times until all the CSI values of decomposed pixels have been estimated.

The kriging procedure is begun by finding all possible data points pairs from the input CSI dataset whose amount is basically just ${}_9C_2$ which equals to 36 pairs. Metric distance between points of each possible pairs is then computed. Next, the pairs are grouped accordingly to their respective metric distance and then empirical semivariance and covariance of each group can be computed according to eq 2.20 & 2.21, respectively. The obtained semivariance values are then plotted together with their respective group's metric distance on the 2D Cartesian plane to obtain empirical semivarigram model of CSI dataset. On the other hand, obtained covariance values are used to construct the covariance matrix between known data points as shown on first term of eq 2.16.

The next step is the fitting process of the empirical semivariogram into common semivariogram models through the cross-validation method. As explained in a previous chapter, cross-validation Leave One Out (LOO) is utilized to select which semivariogram model is the best fit for an empirical semivariogram. The process is begun by generating logical arrays for all possible combinations of semivariance under the LOO scheme. Next, the semivariance dataset is split into training and evaluation groups depending on each logical array. The semivariogram created from the training dataset is then fitted with all available semivariogram models through the built-in nonlinear least square algorithm inside Matlab. The obtained semivariogram functions are then validated with the evaluation dataset to obtain the RMSE value of each semivariogram model. This process is repeated until all logical arrays have been checked. The semivariogram model with the least RMSE value is then selected as the most suitable model for this particular dataset.

On a separate process, the model also computes the metric distance between a query point (centroid point of a decomposed pixel) and centroid points of the satellite pixels. The distance array is then inserted into selected a semivariogram model to create specific covariance functions of a query point to its surrounding data points as shown on eq 2.18. The next step is an element-wise multiplication between the inverse of the previously obtained covariance matrix between data points and specific covariance matrix of query points to obtain the weights matrix as presented on eq 2.19. Finally, CSI value at the query point can be estimated by simply doing a dot product operation between the weight matrix and original data points matrix as shown on eq 2.8. The kriging process is then repeated for the other centroid point of decomposed pixels.

$$\text{zoomed-in interpolated array} = \begin{bmatrix} CI_{1,1} & CI_{2,1} & \cdots & CI_{t,1} \\ CI_{1,2} & CI_{2,2} & \cdots & CI_{t,2} \\ \vdots & \vdots & \ddots & \vdots \\ CI_{1,p} & CI_{2,p} & \cdots & CI_{t,p} \end{bmatrix}$$

$$p = (j - i + 1) \cdot (l - k + 1) \tag{3.2}$$

Upon checking all query points, the kriging sub-procedure will return an array consisting of estimated CSI values on all decomposed pixels centroid points into the main spatial interpolation procedure. Spatial interpolation procedure continues to run until all datasets at every time condition in a year period have been analyzed. All interpolated CSI dataset, regardless of their respective sky condition characteristic, will be incorporated together into a single array and then zoomed-in through the same method as the geometry logical array. The next step is to embed solar park 2D geometry into the zoomed-in array. This process is done through element-wise multiplication between geometry logical and zoomed-in arrays; the resulting array structure is presented on eq 3.2. This array is then stacked together into a 3D array with other similar arrays under different CSI datasets (unmodified or modified) to create the final output array of the spatial interpolation procedure.

3.7. APPROXIMATING THE OPTIMUM SPATIAL CONFIGURATION OF MEASUREMENT POINTS

After getting higher spatial resolution CSI dataset, the model moves on to another procedure which approximates the optimum spatial configurations of measurement points in the solar park. The procedure is com-

posed from two main statistical analyses, namely K-means clustering and Greedy search process. The K-means process is utilized to filter some active pixels within interpolated CSI arrays to be chosen as measurement points candidates. On the other hand, Greedy search process is used to tailor the optimum configurations of measurement points selected from the candidates array recommended by the K-means process. Unlike the spatial interpolation procedure which is conducted on every CSI dataset, this procedure only focuses its analysis on unmodified interpolated CSI dataset. More detailed discussion regarding these two approaches are available in the next sub-sections.

3.7.1. PROBABILITY OF EACH PIXELS BEING THE RIGHT CENTROID

As the title implies, the candidates array is created based on the probability of each pixel being a centroid when the solar park area is divided into several clusters. As it can be seen from fig 3.11, the procedure is started by picking a dataset for a certain time condition from the interpolated CSI data array. The model then determines whether the dataset belongs to uniform or non-uniform sky condition. Since all the datasets under uniform sky condition had been specifically tagged with arrays of ones in a previous procedure, the model can easily separate non-uniform sky condition datasets from the interpolated CSI array.

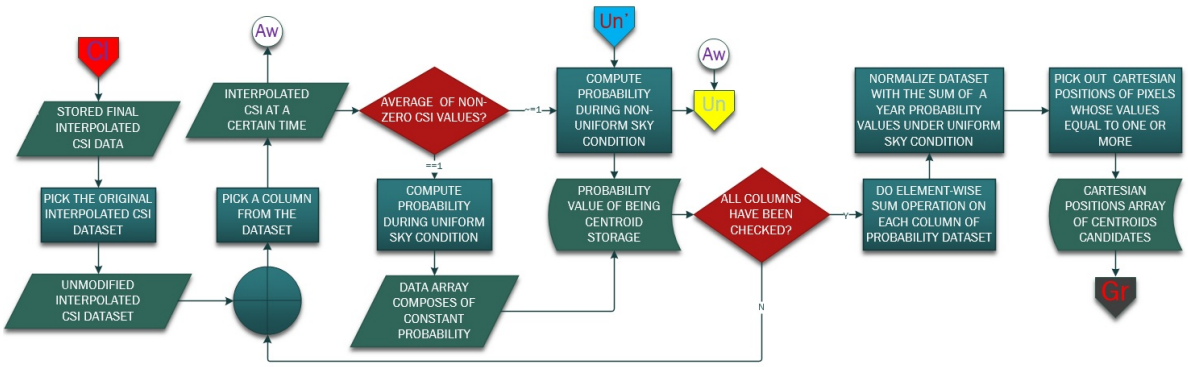


Figure 3.11: Flowchart showing how the model estimates probability value of each cell being a centroid

In order to assign the probability value for each pixel within a certain time period, the following points have been assumed:

- Under uniform sky condition, all active pixels inside the array have an equal chance to become a centroid of clusters. According to the basic assumption that has been mentioned in an earlier part of this chapter, regardless what kind of spatial measurement points configuration is being applied on the solar park, the measurement error will always be zero. It implicitly means that all active pixels on the solar park have an equal probability to be the right measurement point position. The sum of probability array (SmP_u) of each pixel under the multiple clusters case can be then computed through eq 3.3.

$$[SmP_u] = \frac{N_c}{N_{ap}} \tag{3.3}$$

N_c : Number of clusters , N_{ap} : Number of active pixels within geometry logical array

- While under non-uniform sky condition things are a little more complicated, since there should be some spatial variations of CSI in the solar park. In order to obtain a probability value of each pixel under this sky condition, the dataset has to be analyzed by K-means clustering sub-procedure as shown on fig 3.12.

At the earlier stage of this sub-procedure, the dataset is assessed by standard K-means algorithm in order to find the centroid value of each cluster. Upon obtaining all centroid values, the model will search for any active pixels whose value resembles centroid values the most. To avoid the possibility of one pixel being a centroid for more than one cluster, the previous process is conducted locally within each respective cluster.

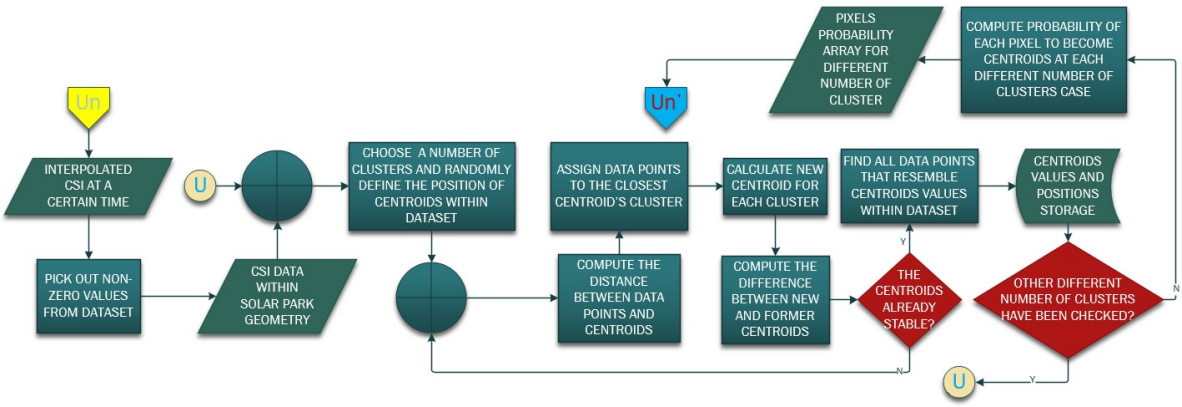


Figure 3.12: Flowchart showing how the model estimates probability value under non-uniform sky condition

The searching process has two possible outcomes; the first is when the model only finds a single pixel that resembles the centroid value and the second is when the model finds more than one pixel. In case of the first outcome, the single pixel will have a probability value equals to one; otherwise the probability value has to be distributed equally between similar active pixels. At the end, both outcomes will assign the probability values onto selected active pixel(s) and let the remaining active pixels having zero probability. Probability array for each cluster under the non-uniform sky condition formula is summarized on 3.4.

$$[P] = \begin{bmatrix} P_{1,1} \\ P_{1,2} \\ \vdots \\ P_{1,p} \end{bmatrix}$$

$$[P] = \begin{cases} P(i \cong Cnt) = \frac{1}{N_r} & \text{resembled pixels} \\ P(i \ncong Cnt) = 0 & \text{other pixels} \end{cases} \tag{3.4}$$

Cnt : Centroid's value of each cluster , N_r : Number of selected pixel(s)

The above process is repeated until all probability values for each cluster have been computed. A simple matrix sum operation of probability array of from each cluster as shown is conducted to obtain the sum of probability arrays under a particular non-uniform sky condition(SmP_{no}). This array is then returned to the main procedure for further processing.

$$[SmP_{no}] = \begin{bmatrix} P(1)_{1,1} \\ P(1)_{1,2} \\ \vdots \\ P(1)_{1,p} \end{bmatrix} + \begin{bmatrix} \cdots \\ \cdots \\ \vdots \\ \cdots \end{bmatrix} + \begin{bmatrix} P(N_c)_{1,1} \\ P(N_c)_{1,2} \\ \vdots \\ P(N_c)_{1,p} \end{bmatrix}$$

Upon obtaining all probability arrays for each temporal dataset, the model continues the main procedure by doing an element-wise summation of each temporal condition probability array as illustrated below, regardless of the type of sky condition. This operation will return a year sum of probability values of each cell under a certain number of clusters scheme(SmP_{yr}). In order to make the interpretation of this array easier, the array is firstly normalized by a constant that represents a year sum of the probability of a single active pixel under clear-sky condition (SmP_{yc}) which can be computed through eq 3.5.

$$[SmP_{yr}] = \begin{bmatrix} SmP(1)_{1,1} \\ SmP(1)_{1,2} \\ \vdots \\ SmP(1)_{1,p} \end{bmatrix} + \begin{bmatrix} \cdots \\ \cdots \\ \vdots \\ \cdots \end{bmatrix} + \begin{bmatrix} SmP(t)_{1,1} \\ SmP(t)_{1,2} \\ \vdots \\ SmP(t)_{1,p} \end{bmatrix}$$

$$SmP_{yc} = \frac{t \cdot N_c}{N_{ap}} \tag{3.5}$$

The normalized sum of the probability array describes how high the probability is of each pixel being the right measurement position with regard to ideal probability (under clear-sky condition) within a year period. A lesser value of this parameter simply means that the pixel has a smaller probability of being a centroid. Nevertheless, by obtaining this array the model will finally be able to filter out any dataset elements whose value is less than one, and gather the remaining pixels to form a centroid candidates array. The whole process is repeated for several different clusters number and different candidate pixels might be returned by the process depending on the clusters number.

3.7.2. THE MOST APPROPRIATE COMBINATION OF SCREENED DATASET

As you may expect from the title, this procedure selects and arranges the candidate pixels into several configurations of measurement points. Detailed work flow of this procedure is presented on fig 3.13 and, as can be seen the procedure mainly relies on the greedy search algorithm to determine the optimum candidate pixels combination. Unfortunately, before greedy algorithm can be utilized the model has to define the starting pair of candidate pixels. The main reason for this is because there is a chance that the greedy algorithm will pick out a wrong starting pixel for creating the combination and congest the whole procedure, since the local objective function cannot be fulfilled.

The pair is selected by a limited random picking approach. Basically the approach randomly picks two pixels from the logical geometry array under one condition, the Euclidean metric distance between these points must be equal or more than the threshold distance(D_{th}). The threshold metric distance is derived by assuming that the solar park area is composed of two identical(in term of total area) clusters whose area resembles perfect circle geometry. Moreover, both circles are perfectly coincident on their border line therefore the distance between both circles centroids is just their own diameter. The diameter value, which can be calculated through eq 3.6 is then assigned as the threshold metric distance. Since it is a randomized process, several pairs are created to see the consistency of the recommended measurement points configurations.

$$D_{th} = 2 \cdot \left(\sqrt{\frac{A_c}{\pi}} \right) \tag{3.6}$$

A_c : Total area of each cluster (m^2)

Upon obtaining several starting pixels pairs, the model will pick a pair and check the number of clusters that is being analyzed. If the requested number of clusters is equal to two, the model has to pick up the two closest pixels to the pair in term of Euclidean metric distance from the respective candidates array. Both pixels will simultaneously be added into final measurement points configuration and removed from the respective candidate array. If the requested number of clusters is more than two, in addition to the above procedure the model also has to conduct the greedy algorithm to approximate which candidates would be suited to become the other remaining measurement point(s).

As explained in previous chapter, the greedy algorithm selects the solution based on the immediate effect to the global objective of the algorithm. In this case, the global objective of the model is to have stable downward trend of uncertainty value as the number of measurement point increases. In order to satisfy this global objective, the greedy algorithm has to find candidates with the smallest positive uncertainty difference. The uncertainty difference is the difference between prior uncertainty, where the candidates did not contribute to computation, and the new uncertainty value where the candidates did contribute.

The greedy algorithm starts the process by finding out the number of pixels that has to be selected from the candidate array in order to satisfy the requested number of clusters. In the next step, the model simulates the uncertainty value of the existing measurement point configuration. Upon obtaining this value, the model pairs up the existing configuration with a single pixel from the respective candidate array. The model then once again simulates the uncertainty value of the new measurement points configuration. The process is repeated until the uncertainty values of all possible configurations have been obtained. Next, the model computes the difference between uncertainty of the existing configuration and uncertainty of all configurations. The pixel that creates a configuration with the smallest positive uncertainty difference is then added

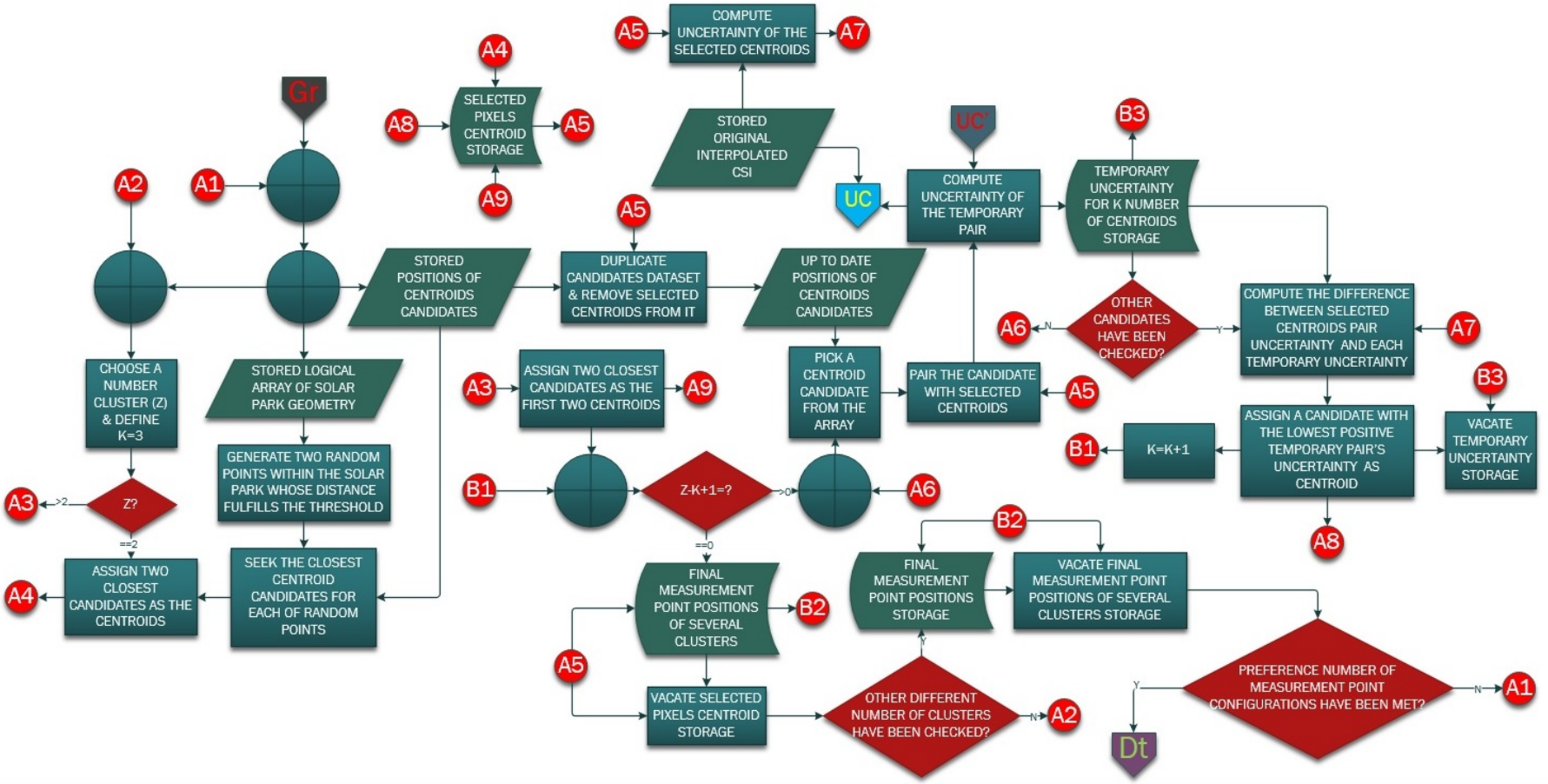


Figure 3.13: Flowchart regarding how the model selects the most appropriate measurement points from centroid candidate's array (greedy algorithm)

into the final measurement points configuration and removed from the candidate array. The greedy algorithm is repeated until the required number of pixels have been reached. The following matrix illustrates how the output array that presents final measurement points positions(Po_{fin}) would look like after running the procedure within the following number of clusters interval $2 \leq N_c \leq N_{cmax}$. Since the model has generated several starting pixels pairs earlier, the user might expect to get several different measurement points configurations under the same number of measurement points scheme.

$$[Po_{fin}] = \begin{bmatrix} Po_{1,1} & Po_{2,1} & \cdots & Po_{(N_{cmax}-1),1} \\ Po_{1,2} & Po_{2,2} & \cdots & Po_{(N_{cmax}-1),2} \\ 0 & Po_{2,3} & \ddots & Po_{(N_{cmax}-1),3} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & Po_{(N_{cmax}-1),N_c} \end{bmatrix}$$

The only thing missing from the previous discussion is a clear explanation of how the uncertainty of a certain measurement points configuration can be estimated. The detailed work flow of this process is presented on fig 3.15 and it can be seen that the process is quite simple due to utilization of built-in ECDF function from Matlab. The basic concept of this procedure is to test out how a particular measurement points configuration performs under the scheme where the configuration is fixed for a one year period. Similar to many other procedures within the model, this sub-procedure also analyzes the data at every certain temporal condition dataset and in the end all the processed datasets are put together inside a single array for further analysis. One important thing to remember here is the fact that under uniform sky condition the relative deviation value computed from eq 2.30 would be zero. The main process of expanding the ECDF created from the histogram of relative deviations over a year's data is simply a looping border extension process, which stops after the preference confidence interval has been achieved. In addition, this sub-procedure was created as a function that can be used by multiple main procedures therefore the output array will have a flexible structure depending on the input array.

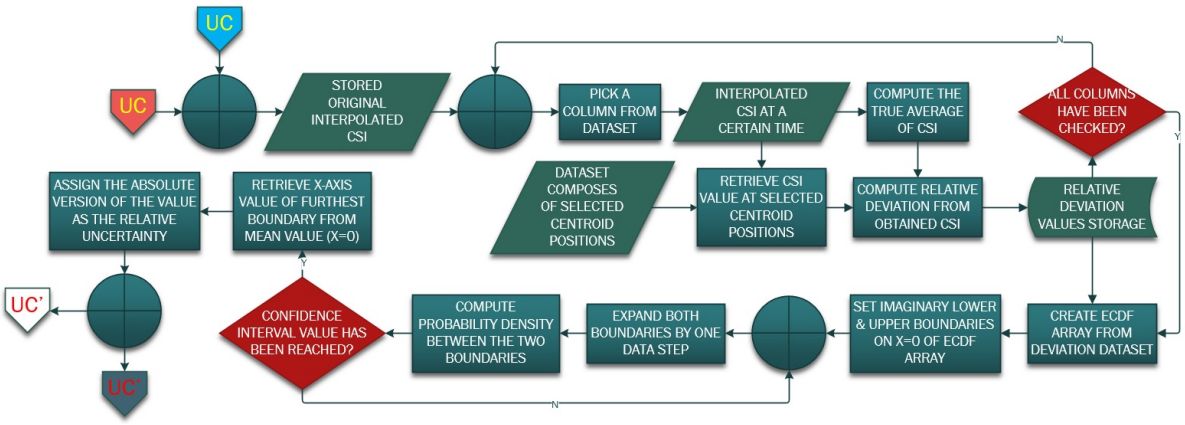


Figure 3.14: Flowchart regarding how the relative uncertainty of a certain CSI dataset is computed within the model

3.8. DETERMINING THE FINAL UNCERTAINTY VALUE

As can seen from the overview flowchart on fig 3.1, this is the final main procedure before the model can return the final uncertainty output to the user. The fundamental idea of this procedure is to implement the measurement points configurations recommended by previous procedures into the modified, interpolated CSI dataset. The recommended measurement points configurations are built from the interpolated original satellite CSI dataset which did not go through built-in error simulation procedure therefore these configurations are only meant for ideal conditions. Seeing how these ideal configurations perform under non-ideal conditions as on modified CSI datasets is the main purpose of this procedure. The work flow of this procedure is presented on fig 3.15 and it can be seen that the procedure is simply a looping of the uncertainty computation process which is done by the previously explained sub-procedure.

Nevertheless, the procedure is started by selecting one measurement points configuration from the dataset. The uncertainty of this particular configuration (Un) is then estimated on a particular modified CSI dataset.

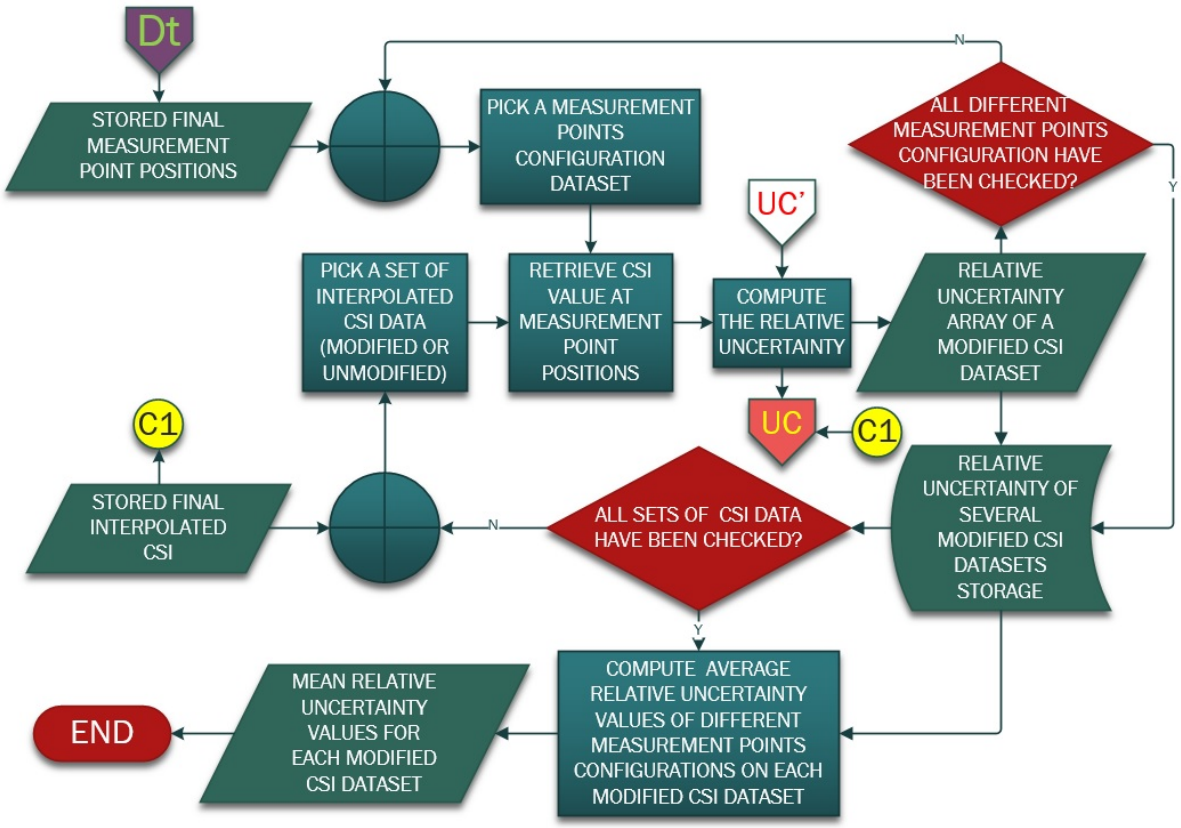


Figure 3.15: Flowchart showing how the final uncertainty value of the dataset is estimated within the model

This process is repeated until all the different configurations of different number of measurement points have been checked. The output array after running it within the following number of clusters interval $2 \leq N_c \leq N_{cmax}$ would have the structure that resembles array on eq 3.7. This kind of array is only meant for one modified CSI dataset therefore in order to accommodate output from other modified datasets, a 3D array is required.

$$[Un] = \tag{3.7}$$

$$\begin{bmatrix} Un(2)_{1,1} & Un(2)_{2,1} & \cdots & Un(2)_{(N_{sp}-1),1} & Un(2)_{N_{sp},1} \\ Un(3)_{1,2} & Un(3)_{2,2} & \cdots & Un(3)_{(N_{sp}-1),2} & Un(3)_{N_{sp},2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Un(N_{cmax}-1)_{1,(N_c-1)} & Un(N_{cmax}-1)_{2,(N_c-1)} & \cdots & Un(N_{cmax}-1)_{(N_{sp}-1),(N_c-1)} & Un(N_{cmax}-1)_{N_{sp},(N_c-1)} \\ Un(N_{cmax})_{1,(N_c-1)} & Un(N_{cmax})_{2,(N_c-1)}) & \cdots & Un(N_{cmax})_{(N_{sp}-1),(N_c-1)} & Un(N_{cmax})_{N_{sp},(N_c-1)} \end{bmatrix}$$

N_{sp} : Number of starting candidates pairs that form different measurement points configurations

Upon obtaining the uncertainty array for all modified CSI datasets, the procedure continues with the final process within the model. The process is simply an element-wise averaging computation between each single column within the uncertainty array of a particular modified dataset as presented on eq 3.8. The main objective of this process is to extract an overall uncertainty values array (Unm) from multiple number of measurement points configurations whose uncertainty values might vary considerably.

$$[sUn] = \begin{bmatrix} Un(2)_{1,1} \\ Un(3)_{1,2} \\ \vdots \\ Un(N_{cmax}-1)_{1,(N_c-1)} \\ Un(N_{cmax})_{1,(N_c-1)} \end{bmatrix} + \begin{bmatrix} \cdots \\ \cdots \\ \vdots \\ \cdots \\ \cdots \end{bmatrix} + \begin{bmatrix} Un(2)_{N_{sp},1} \\ Un(3)_{N_{sp},2} \\ \vdots \\ Un(N_{cmax}-1)_{N_{sp},(N_c-1)} \\ Un(N_{cmax})_{N_{sp},(N_c-1)} \end{bmatrix}$$

$$[Unm] = \frac{[sUn]}{N_{sp}} \tag{3.8}$$

The mathematical operation on eq 3.8 returns an array that represents average uncertainty values of several measurement points configurations under a certain interval measurement points number. The averaging computation is repeated for the uncertainty array of other modified CSI datasets. Upon computing average uncertainty array of all modified CSI datasets, the model closes the whole procedure by merging unmodified CSI dataset's uncertainty array along with obtained uncertainty arrays of several modified CSI datasets as the final output of the model; the final array is illustrated on eq 3.9. Please note that the first column of this array represents uncertainty values of unmodified CSI datasets, while the rest belongs to modified CSI datasets.

$$[Un_{fin}] = \tag{3.9}$$

$$\begin{bmatrix} Unm(2)_{1,1} & Unm(2)_{2,1} & \cdots & Unm(2)_{(N_t-1),1} & Unm(2)_{N_t,1} \\ Unm(3)_{1,2} & Unm(3)_{2,2} & \cdots & Unm(3)_{(N_t-1),2} & Unm(3)_{N_t,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Unm(N_{cmax}-1)_{1,(N_c-1)} & Unm(N_{cmax}-1)_{2,(N_c-1)} & \cdots & Unm(N_{cmax}-1)_{(N_t-1),(N_c-1)} & Unm(N_{cmax}-1)_{N_t,(N_c-1)} \\ Unm(N_{cmax})_{1,(N_c-1)} & Unm(N_{cmax})_{2,(N_c-1)}) & \cdots & Unm(N_{cmax})_{(N_t-1),(N_c-1)} & Unm(N_{cmax})_{N_t,(N_c-1)} \end{bmatrix}$$

N_t : Number of modified CSI dataset(s)

4

CASE STUDY

In order to check how the model generally performs, some case studies have to be conducted. The case studies chosen in this report are based on imaginary solar parks that were generated beforehand and mainly focus on following three parameters:

- Difference in local climate characteristics
- Difference in temporal and spatial resolution of the data
- Difference in total area of the imaginary solar park

By comparing the output result from variations of the imaginary solar park under different parameter settings, the model is evaluated on whether it is able to give out a relevant result that follows fundamental logic; *e.g.*the measurement uncertainty due to spatial irradiance variations would most likely be larger for a solar park in wet climate area compared to one in dry climate (under the assumption that all other parameters are identical). In addition, fixed numbers of sample measurement points configurations of 10 and modified CSI datasets of 5 have been used within all current case studies. Lastly, satellite built-in error is set to identically follow each respective database average RMSE values while built-in error due to the measurement device is assumed to be equal to 2.62% (pyranometer Kipp&Zonen CMP3).

4.1. GENERAL INFORMATION OF IMAGINARY SOLAR PARK SITES

4.1.1. WET CLIMATE

For representing a solar park in a wet climate region, a random area in Leipzig, Germany has been chosen to be the imaginary solar park. An irregular polygon with 4 sides has been selected to be the base 2D geometry for all imaginary solar parks within the report; a satellite view of the solar park is presented on 4.1. As can be seen from the satellite view, two areas with more or less similar geometry but different in size have been chosen to be the imaginary solar parks in Leipzig. Complete information regarding both solar parks is summarized on table 4.1.

Table 4.1: Information summary of imaginary solar parks on Leipzig

No.	WGS 84 Coordinates	Total Area	Solar Database	Temporal Resolution
1	c1 (12.3857 °E, 51.3356 °N) c2 (12.3956 °E, 51.3401 °N) c3 (12.401 °E, 51.3329 °N) c4 (12.3929 °E, 51.3211 °N)	±1.88 km^2	CAMS (2007)	15 minutes 60 minutes
2	c1 (12.3757 °E, 51.3356 °N) c2 (12.3956 °E, 51.3501 °N) c3 (12.411 °E, 51.3329 °N) c4 (12.3929 °E, 51.3111 °N)	±8.57 km^2		15 minutes

As you can see from the table, apart from varying the total area of the solar park, the Leipzig site is also intended to be an example of how different temporal resolution datasets of identical solar parks influence the

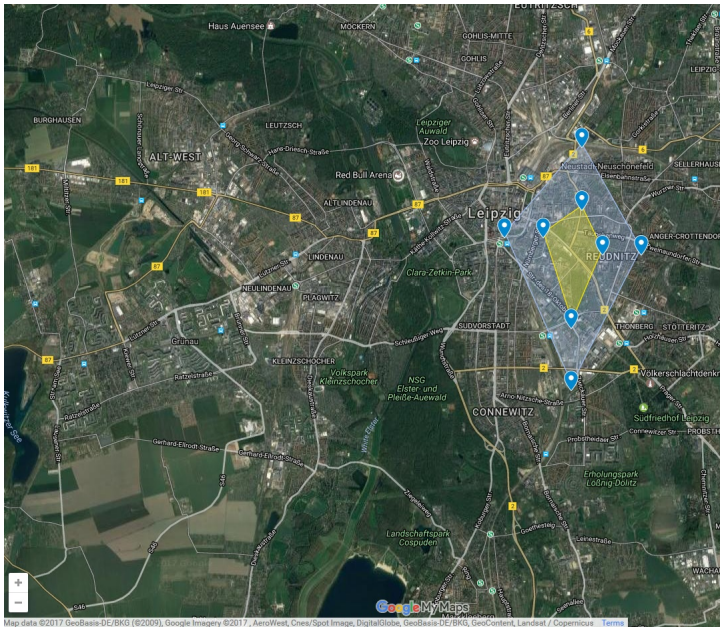


Figure 4.1: Satellite view of imaginary solar park on Leipzig

final output of the model. Lastly, the CAMS solar database has been chosen to be the data source within this study case, due to the fact that it does not have any limitations in term of service (which is quite the opposite of the HelioClim database).

4.1.2. DRY CLIMATE

On the other hand for a dry climate region, a random area in Arizona USA has been selected to become the imaginary solar park for the model. In order to compare dry and wet climate regions equally, the solar park has been designed to resemble the 2D geometry of the one in Leipzig. The satellite view of the imaginary solar park is presented on fig 4.2 while all related information regarding the solar park is summarized on table 4.2.

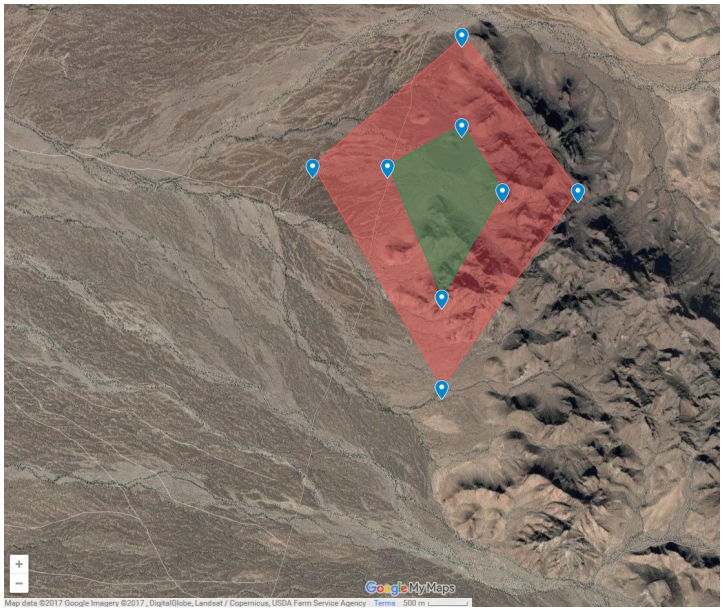


Figure 4.2: Satellite view of imaginary solar park on Arizona

As a reminder, the whole model is constructed from several stochastic simulations which means that the result does not necessarily predict the actual value. Instead, it predicts statistical pattern of the actual value

Table 4.2: Information summary of imaginary solar parks on Arizona

No.	WGS 84 Coordinates	Total Area	Solar Database	Temporal Resolution
1	c1 (114.1143 °W, 33.7356 °N) c2 (114.1044 °W, 33.7401 °N) c3 (114.099 °W, 33.7329 °N) c4 (114.1071 °W, 33.7211 °N)	±1.87 km^2	NSRDB (2007)	30 minutes
2	c1 (114.1243 °W, 33.7356 °N) c2 (114.1044 °W, 33.7501 °N) c3 (114.089 °W, 33.7329 °N) c4 (114.1071 °W, 33.7111 °N)	±8.58 km^2		

based on detected behaviours. Unlike absolute uncertainty (W/m^2) which tries to predict the actual value in irradiance unit, the relative uncertainty due to spatial variation is a dimensionless parameter which is a suitable indicator to predict statistical patterns in the actual data. Therefore the analysis in this chapter will mainly be based on relative uncertainty. All absolute uncertainty results coming out from the model are separately presented in the appendix. In order to better understand what exactly is occurring inside the model, indicators called sky uniformity (Su) and clarity (Cr) are introduced within the report. The Su and Cr are computed through eq 4.1 and ref 4.2 after filtering out all nocturnal data within the original dataset.

$$Su = \frac{N_{uni}}{N_{tot}} \tag{4.1}$$

$$Cr = \frac{N_{cr}}{N_{tot}} \tag{4.2}$$

N_{uni} : Number of CSI dataset under uniform sky condition throughout a nocturnal free year CSI dataset
N_{cr} : Number of CSI dataset under clear-sky condition throughout a nocturnal free year CSI dataset
N_{tot} : Total number of dataset within a nocturnal free year CSI dataset

Due to time constraints, the analysis within this chapter is only based on a single year dataset. The year 2007 dataset has been chosen due to the fact that it resembles the average characteristics of five years of Leipzig datasets. The first two characteristics are the newly-defined sky uniformity and clarity of each annual dataset. Both characteristics are utilized to measure CSI spatial variation of the dataset throughout a year. Less inter-annual variation of both characteristics means more similarity in overall sky conditions from year to year. The last characteristic is the average of CSI values under non-clear sky conditions throughout a year. The main objective of this characteristic is to measure non-clear sky CSI magnitude. There is no need to include clear-sky CSI into the average, since it has been represented by the sky clarity characteristic.

Table 4.3: Leipzig datasets characteristics from 2007–2011

Year	Sky Uniformity	Sky Clarity	Average of CSI
2007	24%	15%	0.609
2008	22%	12%	0.614
2009	23%	14%	0.603
2010	22%	13%	0.591
2011	26%	17%	0.631

4.2. RESULT AND ANALYSIS

4.2.1. INFLUENCE OF SATELLITE DERIVED DATA BUILT-IN UNCERTAINTY INTO THE MODEL

This section aims to discuss the influence of built-in error simulation to suggested relative uncertainty values. The discussion is started by comparing the relative uncertainty result of both modified and unmodified input datasets. The first uncertainty comparison is for the imaginary solar park in Leipzig, whose result is presented

on fig 4.3. It is fairly clear that the uncertainty value estimated from dataset without any embedded error is slightly lower than other datasets with embedded error. This result is expected since the built-in error of satellite-derived data and measurement devices should have some contribution to CSI spatial variation on the solar park and most likely the contribution would be to escalate the uncertainty value. The escalation is most likely caused by the addition of random components to non-clear sky elements of the original dataset. This effect can be clearly seen from the significant reduction of sky uniformity of modified datasets (±15%) compared to unmodified ones (±24%). The reduction in the number of uniform sky conditions explicitly means the increasing number of non-uniform sky conditions, which basically induces more uncertainty due to spatial variation. Another interesting issue to discuss is related to how many datasets within the uniform sky conditions belong to clear-sky conditions. The result shows that the sky clarity of Leipzig dataset (±15%) is smaller than its unmodified sky uniformity value. This fact implies that uniform sky condition in Leipzig region is not dominated by clear-sky conditions but uniform cloudy conditions.

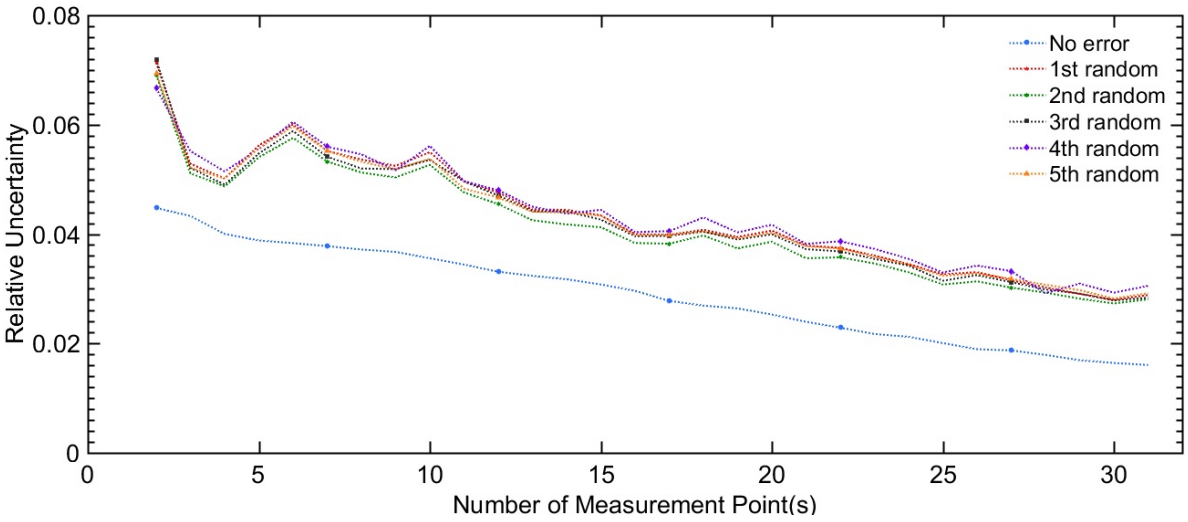


Figure 4.3: Relative uncertainty of Leipzig solar park under the inclusion and exclusion of built-in error schemes

The next case comes from the imaginary solar park on Arizona whose result is presented on fig 4.4. Unlike the Leipzig solar park where the uncertainty of its modified dataset is larger than the unmodified one, uncertainty of the modified dataset in the Arizona region is smaller compared to the original one. This occurrence is quite strange since similar to the previous case it has been expected that the built-in error simulation would escalate the final uncertainty value. The most probable reason for this occurrence lies in the combination of the effects of built-in bias of randomized numbers and difference of local climate characteristics of each respective solar park which are discussed in more detail in the next section.

As shown by sky uniformity values of unmodified datasets from both solar parks, it is clear that the Arizona region has more uniform sky conditions (±72%) compared to Leipzig(±24%). Furthermore, unlike the Leipzig case in which the clear-sky condition does not fully dominate uniform sky condition in the region, the clear-sky condition seems to fully dominate the uniform sky condition in the Arizona region. This is shown by the sky clarity value (±72%) which is more or less similar to the sky uniformity value within this region. As the basic algorithm of the model operates, the built-in error simulation of imaginary solar park dataset in Arizona is literally limited to only ±28% of the total available temporal datasets, which is fairly small when compared to that of Leipzig (±85%).

It seems that the random numbers generated by Matlab are not perfectly spread into uniform distribution. This has been proven by running several trials, and all of them suffer from notable negative bias values. The effect of negative bias within the model is indicated by a lower overall average of non-uniform sky condition data within modified datasets with regard to its original version. Randomized error with negative bias also induces other effects depending on the characteristic of the dataset it has been embedded into. As mentioned previously, the built-in error simulation is only done with non-clear sky condition data, or in short the randomized error is only embedded into a portion of the data inside the original dataset. The relatively fixed bias of randomized error has to be equally divided into all data inside this portion. Therefore the number of data within this portion determines the effect of negative bias.

In a case where the portion only consists of a small number of data, like the one in Arizona (around

2400 occurrences of non-clear sky condition), the bias influence on the final uncertainty value seems very prominent. The bias prominence is shown by a direct reduction of relative uncertainty values in response to the reduction of the overall average due to negative bias. Meanwhile, this prominence does not exist in the case of Leipzig, where the portion consists of a relatively large number of data (around 15000 occurrences of non-clear sky condition). Different effects in the two datasets are most likely caused by the fact that each non-clear sky condition data within Arizona dataset figuratively holds a larger bias value fraction compared to the one in the Leipzig dataset. Since non-clear sky conditions data plays an important role in determining the final uncertainty values, the significance of embedded bias value on each non-clear sky condition data will also indirectly determine the final uncertainty.

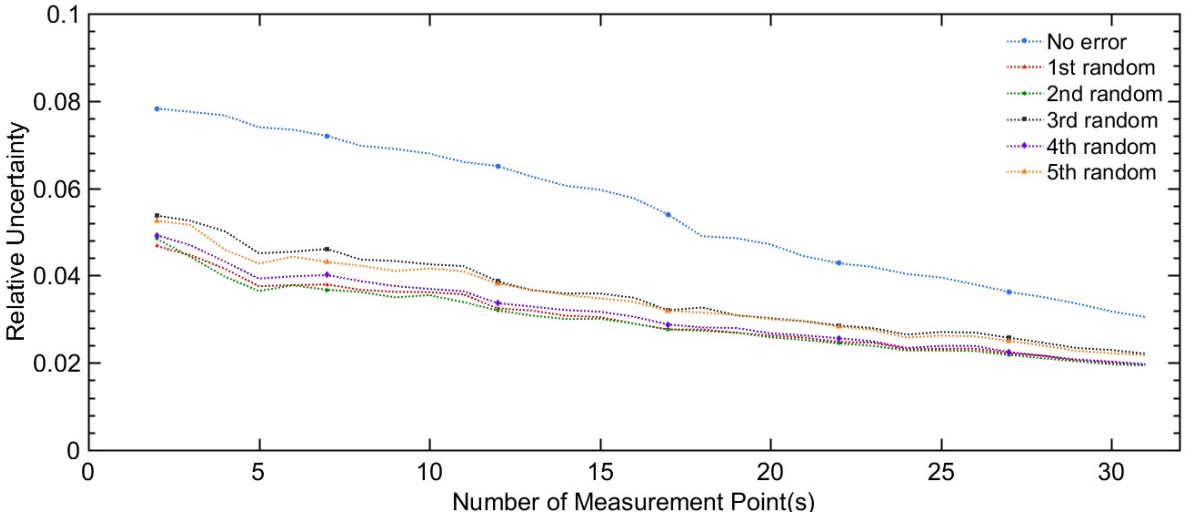


Figure 4.4: Relative uncertainty of Arizona solar park under the inclusion and exclusion of built-in error schemes

As the reader might notice from the above discussion, the built-in error simulation basically converts most of the uniform sky condition datasets which do not belong to clear-sky condition into non-uniform sky condition. This is proven by sky uniformity values of all modified datasets which more or less resembles their own sky clarity values regardless of local climate characteristics of each region. This proves that the error simulation of the input dataset is able to create additional variations of the original dataset by imitating the built-in error of satellite-derived data. Moreover, all tested cases within this chapter also suffer from built-in negative bias of the Matlab random number generator with more or less similar magnitude (4–5% of average value), therefore it makes sense to say that the current outputs of the model are in equal terms. Due to both reasons, it is decided that the modified dataset is sufficient to become the base input for estimating uncertainty value due to spatial variation and will be used in further analysis within this chapter.

4.2.2. INFLUENCE OF TOTAL AREA OF THE SOLAR PARK ON RELATIVE UNCERTAINTY

Next, the discussion intends to find out how the total area of the imaginary solar park affects the relative uncertainty. The first case is the comparison between two different-sized solar parks in the Leipzig region whose uncertainty values are presented on fig 4.5. As is seen, in every number of measurement points, relative uncertainty of larger solar park is always higher than the uncertainty value of the smaller one. This result is in line with the basic logic of extending the coverage area means inducing more spatial variation within the solar park. Since both solar parks use similar datasets to derive their own relative uncertainty values, they also hold similar sky uniformity and clarity values. The uncertainty difference between two solar parks varies depending on the number of measurement points. At small number of measurement points, the uncertainty difference is relatively high however as the number of measurement points increases, the difference decreases. This occurrence is present due to the fact that both curves which have downward trend ideally have to reach zero relative uncertainty values together when the number of measurement points is equal to the number of active pixels within the data array; referring to eq 2.30. Additionally, this requirement forces both curves to possess the region where uncertainty reductions due to increased number of measurement points is almost negligible (*i.e.*saturated condition).

Holding on the previous ideal requirement, the curve fitting process can be then conducted with the

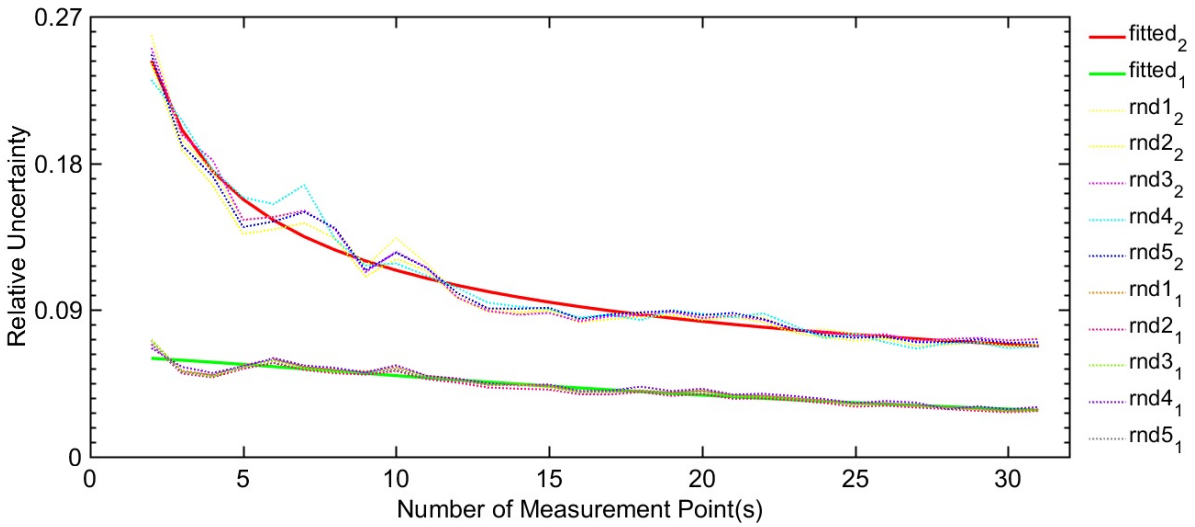


Figure 4.5: Relative uncertainty comparison between two different sizes solar park on Leipzig

average dataset of five modified datasets for each solar park. The requirement component $(N_{ap},0)$ has to be added into the original result array before the curve fitting can be conducted. The function presented on eq 4.3 shows the best fit that can be obtained from most of study case results within this chapter; therefore it will be used in further similar curve fitting processes as the base function.

$$f(x) = \frac{1}{(a \cdot x^b + c)} \tag{4.3}$$

The next step is to see whether an area extension of the solar park in dry climate will induce a similar effect to that of wet climate. An uncertainty comparison is conducted for two different sizes of imaginary solar parks in the Arizona region, and the result is presented on fig 4.6. Generally, the result shows similar behaviour to the one in Leipzig, where the larger solar park has higher relative uncertainty compared to the smaller one. However in terms of the magnitude of the difference between two uncertainty values, they are a little different. In the Leipzig case, the uncertainty difference drops significantly for quite awhile before it settles at around N_c=15. While in the Arizona case, the uncertainty difference only drops significantly for a short period before it starts to level off at around N_c=8. There are two possible explanations regarding this issue. The first is related to their different climate characteristics, and the second is related to the fact that both of them are derived from solar databases whose spatial resolutions are different. Another possibility is a little harder to discuss, since whatever the original solar datasets spatial resolutions were, all of them have been decomposed through kriging process into new datasets under uniform spatial resolution. Therefore, before something can be said about the influence of the original dataset's spatial resolution to relative uncertainty, the kriging result has to first be validated. The only possible explanation left is, the issue occurs due to the different climate characteristic which is the most likely reason as will be discussed on the next section.

4.2.3. RELATIVE UNCERTAINTY COMPARISON UNDER DIFFERENT CLIMATE CHARACTERISTICS

In order to discuss this topic more thoroughly, the main indicator of analysis in this section will be slightly changed from relative uncertainty(U_r) into relative uncertainty reduction rate(U'_r) with regard to the increasing number of measurement points(N_c). The basic formula to compute the reduction rate is presented on eq 4.4. The reduction rate basically describes how quickly the relative uncertainty value drops with every addition of one measurement point. Since the continuous function that represents average uncertainty of five modified datasets has been approximated in the previous curve fitting process, the reduction rate can be easily computed by doing the first differentiation of this function.

$$U'_r(N_c) = \frac{dU_r(N_c)}{dN_c} \tag{4.4}$$

The discussion is started by comparing relative uncertainty reduction rates of two similar size solar parks

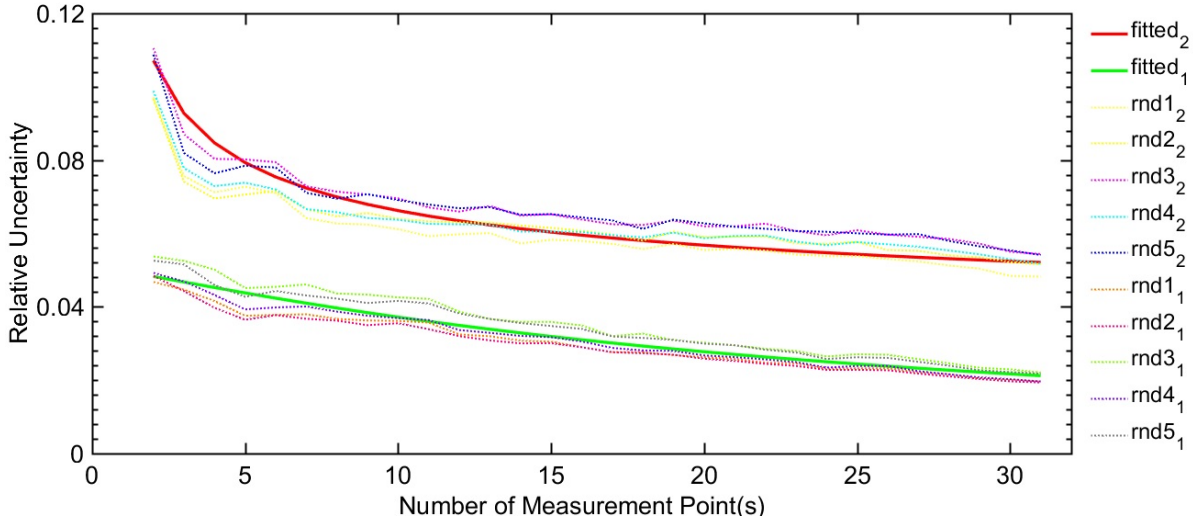


Figure 4.6: Relative uncertainty comparison between two different sizes solar park on Arizona

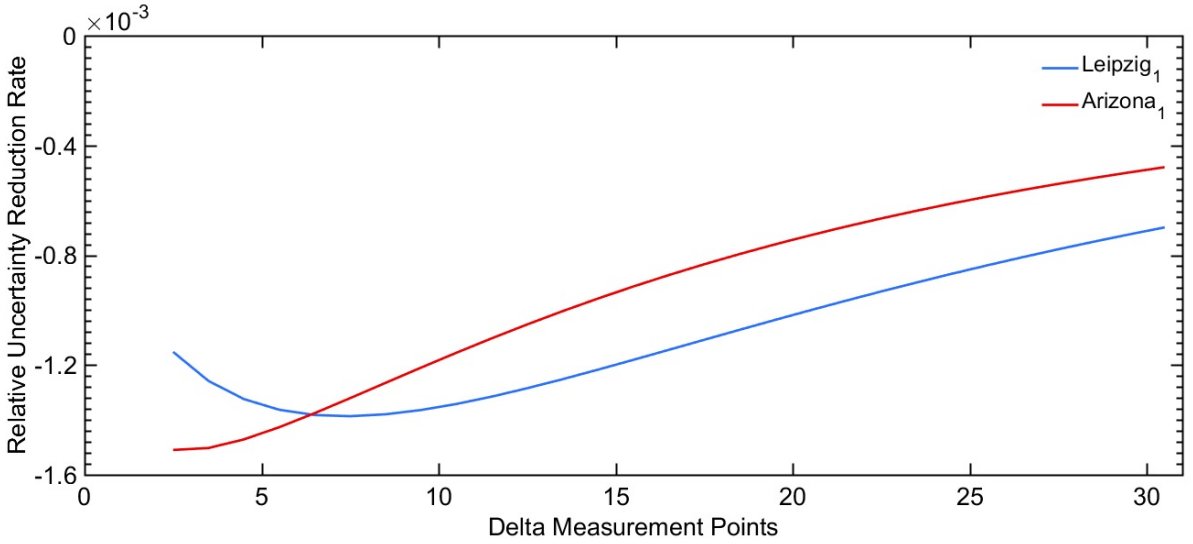


Figure 4.7: Uncertainty reduction rate comparison between normal size Leipzig and Arizona solar parks

with different climate characteristics. The comparison of the result of the normal sized solar park is presented on fig 4.7 while the comparison result of the large solar park is available on fig 4.8. According to the result, it is clear the solar park built in Arizona reaches the saturation condition faster than the one in Leipzig regardless of the size of the solar park. The saturation term stands for the condition where uncertainty reduction rate value starts to level off at every additional measurement point, or in other words the condition where addition of measurement point has lost its effectiveness to bring down the uncertainty value. This result is in line with the fact shown by sky uniformity and clarity of each respective region. As discussed previously, Arizona's yearly sky condition is dominated by clear-sky while in Leipzig, non-uniform cloudy sky condition is more dominant. During clear-sky condition, the model assumes that one measurement point is sufficient to cover measurement activity throughout the solar park without any error. The previous statement implies that the remaining measurement points are idling during that period, and this kind of occasion would definitely degrade the effectiveness of the current measurement points configuration. Effectiveness degradation also may occur in non-uniform sky condition; however it will not be as bad as the one in clear-sky condition since during this condition at least more than one measurement points are required to conduct measurement activity with the least possible error. In the end it makes sense that the uncertainty reduction rate of the region with a lot of clear-sky condition throughout the year reaches its saturation condition quicker compared to the region whose sky is dominated by non-uniform cloudy condition such as Leipzig. This is also the reason

why generally uncertainty reduction rate of the solar park in Arizona is smaller than the one in Leipzig.

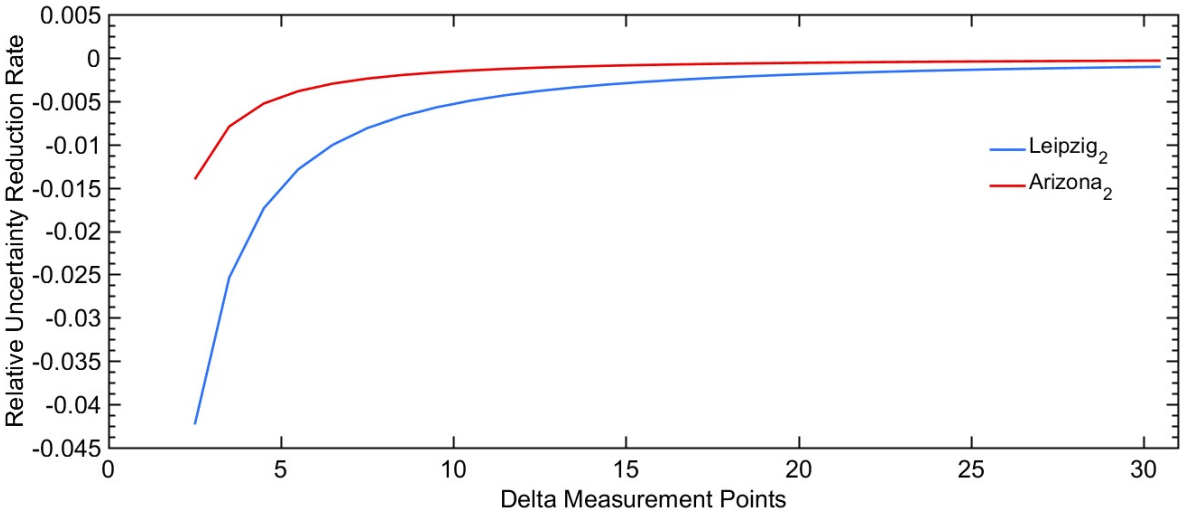


Figure 4.8: Uncertainty reduction rate comparison between large size Leipzig and Arizona solar parks

The next interesting issue to discuss is related to the difference between uncertainty reduction rates in each solar park. Since at a certain point before the number of measurement points equals the number of active pixels the difference between two reduction rates will become negligible, the difference between two reduction rates has to lessen at every addition of a measurement point. Even though there are a few irregularities on the earlier stage, the lessening process in normal sized solar park proceeds quite slowly (almost unnoticeable). While in the case of a large sized solar park, the lessening process is quite pronounced. This occurrence is most likely caused by different climate characteristics of each solar park which, and the effect has furthermore been amplified by the area extension of both solar parks.

The last discussion in this section is aimed to discuss the unanswered question from the previous section regarding the magnitude of relative uncertainty difference between two solar parks with different sizes. This question can be answered by referring to fig B.1 and B.2 which actually compare uncertainty reduction rates between the two solar parks whose size is different. The dashed vertical lines on both figures approximately show the point at which both curves intersect. After the intersection point, the uncertainty reduction rate will be more or less similar in both cases. Similar uncertainty reduction rates means that their relative uncertainty curves (referring to fig 4.5 and 4.6) are currently parallel to each other. Simply put, beyond this intersection point, the addition of measurement points would only lead to a small reduction of relative uncertainty regardless the size of the solar park. In the case of Arizona, this condition occurs when the number of measurement points exceeds 13, while in the case of Leipzig, it occurs after number of measurement points reaches 30. This result enhances the previous insight which states that in a region where the sky is dominated by clear-sky condition, the addition of a measurement point will lose its effectiveness faster compared to a region where clear-sky does not dominate the yearly sky condition.

As a reminder, both regions rely on two different databases whose best temporal resolutions differ by two times. Therefore one region has twice more elements in its dataset compared to the other region. This indeed creates an issue, as will be discussed on the next section; however in above analysis this effect is assumed to be negligible. Additionally, both regions have a different yearly total of sun hours therefore the number of elements inside the dataset of a region with more sun hours will be slightly higher compared to a region with less sun hours. Similar to the previous reminder, this indeed creates an issue but within the analysis once again this influence is assumed to be very small and can safely be ignored.

4.2.4. INFLUENCE OF TEMPORAL RESOLUTION TO FINAL UNCERTAINTY

The main objective of this section is to investigate the effect of dataset temporal resolution on relative uncertainty. In order to do that, an imaginary solar park in Leipzig has been analyzed with two input datasets whose temporal resolutions are different, and the result of this process is then presented on fig 4.9. From a single glance, one notices that the relative uncertainty of the low temporal resolution (60 minutes) dataset is generally rougher than the one produced by the high temporal resolution dataset (15 minutes). This issue is

most likely caused by the total elements within each dataset, in this case the high temporal resolution dataset has three times more elements compared to the low temporal resolution dataset. Therefore, it makes sense that when identical random components (controlled error) are added to both datasets which similarly represent sky conditions for a year period, the dataset with the smaller number of elements will have more noise in its output compared to the one with the larger number of elements in it, since each element of the smaller dataset figuratively has to shoulder more error than each element of the larger dataset.

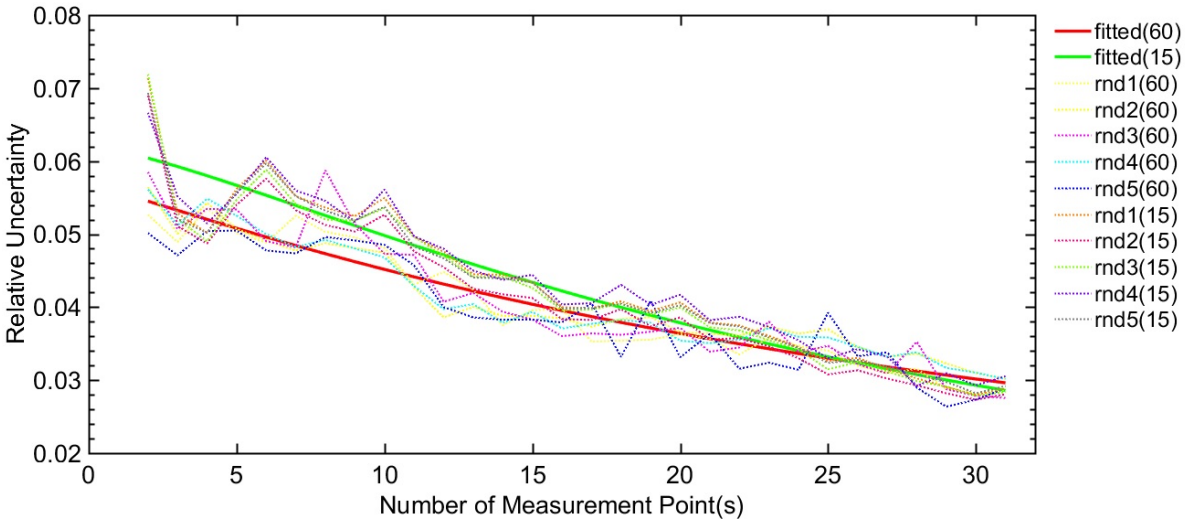


Figure 4.9: Relative uncertainty comparison between two identical solar parks which use different temporal resolution

The discussion is continued to the main point of this section which has been clearly answered by result comparison shown on fig 4.9. As has been expected, lower temporal resolution leads to a smaller relative uncertainty value. This happens due to the averaging process of several sky conditions into a single sky condition. In this particular case, the process averages each four temporal datasets and makes the whole input dataset much smaller compared to its original size. Imagine that each temporal dataset illustrates a particular sky condition at a certain time interval, and the four of these particular sky conditions are then merged together to create a single sky condition that represents its four prior conditions. It would definitely reduce the amount of variation details that originally existed within the prior dataset. Therefore it does make sense to have smaller relative uncertainty as the model's output in a case where a lower temporal resolution dataset is being used. The main question presently is how large the magnitude of the difference between both relative uncertainties would be. According to fig 4.9, a maximum relative uncertainty difference of around 1% exists when the number of measurement points is equal to two. This difference decreases rapidly as the number of measurement points increases. The difference finally hits zero value when both curves intersect approximately as the number of measurement points equals to 25. The result implies that even with such a large scale dataset simplification, relative uncertainty only drops by approximately 1%, and moreover this difference is quickly narrowed as the number of measurement points increases. Therefore the last remark regarding the exclusion of temporal resolution difference in the previous analysis section is still acceptable.

4.2.5. MEASUREMENT POINTS CONFIGURATIONS ON THE SOLAR PARK

The ten different measurement points configurations suggested by the model are analyzed to see whether they have some kind of distinctive pattern that can be used as a recommendation to some solar park's actors to place their irradiance measurement devices. For a normal sized Leipzig solar park, the top view of ten measurement points configurations are illustrated on fig C.1 – fig C.5. From the figures, it is clear that the position of measurement devices are mainly concentrated in three different regions namely, north-eastern, south-eastern and western regions. The concentration of possible measurement device positions is increased as the total number of measurement points increases but even so, the measurement device positions are still concentrated in the three mentioned regions. One possible explanation for this occurrence is related to how the k-means clustering algorithm works. For creating the right data clusters, the algorithm repeatedly computes the sum of Euclidean distances (not metric) between centroids and other data points within each respective cluster. In other words, each of the distance sums shows spatial irradiance gradient within each cluster with

regard to the centroid's value. Therefore each candidate within the ten configurations has its own typical irradiance spatial gradient. When a lot of candidates gather in a single region, it only means that the region has a prominent irradiance spatial gradient throughout a year that has to be covered by several measurement points. On the other hand, the rest of area that does not belong to any of the regions is considered the area that can be easily covered by measurement points placed on concentration regions. A similar occurrence also appears in normal sized solar park on Arizona; however the measurement points positions are concentrated in different area namely the south-eastern, north-western, and eastern regions.

Unlike in the case of normal sized solar park whose measurement points positions are already concentrated into certain regions at small number of measurement points (N_c=4), the concentration of measurement points positions in large solar park starts to become recognisable at a relatively large number of measurement points (N_c=7); referring to fig C.10 andC.20. As an early hypothesis, it seems that each solar park, depending on its total area has a threshold number of measurement points; the threshold gets larger as the total area extends. Before the number of measurement points reaches this threshold, the measurement points positions would still be scattered around the solar park without any clear pattern. Beyond this threshold, the addition of measurement points most likely would only reinforce the already formed patterns as shown in the case of previous normal sized solar park. The hypothesis can be tested by running the model in several solar parks with different total areas; however due to time constraints it will not be covered within the report.

Even though the model does not suggest the exact positions of measurement devices, the model is able to forecast where the devices should be generally placed in the solar park. Please note that the recommendation might be entirely different if all obstacles around the solar park are not ignored.

4.2.6. FINANCIAL ANALYSIS

The last section on this chapter is dedicated to give the reader an overall view of how all the outputs from the model influence the solar park with regard to financial terms. In order to do that, other assumptions have to be added into the imaginary solar park to make it more like an actual solar park; the following points describe all of these additional assumptions:

- Yearly total insolation (I_y) values of Leipzig and Arizona are assumed to be 1100 kWh/(m^2) and 2100 kWh/(m^2) as presented on SolarGIS service website [1].
- Ground Cover Ratio (GCR) is assumed to be constant at 0.3 for both imaginary solar parks.
- Overall power conversion efficiency (η_{sp}) of each solar park is assumed to be 15% for a whole year period.
- Levelized Cost of Electricity (LCOE) is assumed to be similar to sales price on Power Purchase Agreement (PPA) which is set to be fixed at US$ 74.4/MWh for Leipzig [77] and US$ 58.3/MWh for Arizona [78].

$$t_s = \frac{I_y}{365} \tag{4.5}$$

The analysis is started by computing the yearly average sun hours under AM 1.5 standard condition irradiance, which is commonly known as equivalent sun hours (t_s); the computation is done through eq 4.5. The result approximates that Leipzig and Arizona would have 3 and 5.8 equivalent sun hours throughout a year.

$$P = A_t \cdot GCR \cdot \eta_{sp} \cdot Irr_{std} \tag{4.6}$$

A_t : Total area of the solar park (m^2)

The next step is to estimate the capacity of solar park (P) that is feasible to be built on the selected area. This estimation is done through the simple computation that follows eq 4.6. Since both solar parks have more or less similar total area, the estimated capacities for both solar parks are also the same which is around 84.6 MW.

[1] http://solargis.com/products/maps-and-gis-data/free/download/

$$E = P \cdot t_s \cdot N_{at} \tag{4.7}$$

N_{at} : Number of operation days of the solar park in a year (days)

Upon estimating the capacity of the solar park, expected yearly energy yield (E) can be then computed through eq 4.7. Furthermore, it is also assumed that both solar parks operate for a full year without any days off. The result shows that Arizona's solar park would deliver around 200 GWh per year while Leipzig's solar park would only deliver 127 GWh per year. Since both solar parks have different total yearly sun hours, it is to be expected that the energy yield of both solar parks would be different.

$$U_s = E \cdot PPA \cdot U_r \tag{4.8}$$

The last step is to calculate the uncertainty of energy sales (U_s) through eq 4.8 on every different number of installed measurement points. The uncertainty is calculated under the assumption that irradiance measurement devices are the only devices meant to estimate energy yield of the solar park, which somehow makes the uncertainty of energy yield measurement able to be represented by uncertainty of irradiance measurement. The computation result of Leipzig and Arizona solar parks are presented on fig 4.10 and 4.11, respectively.

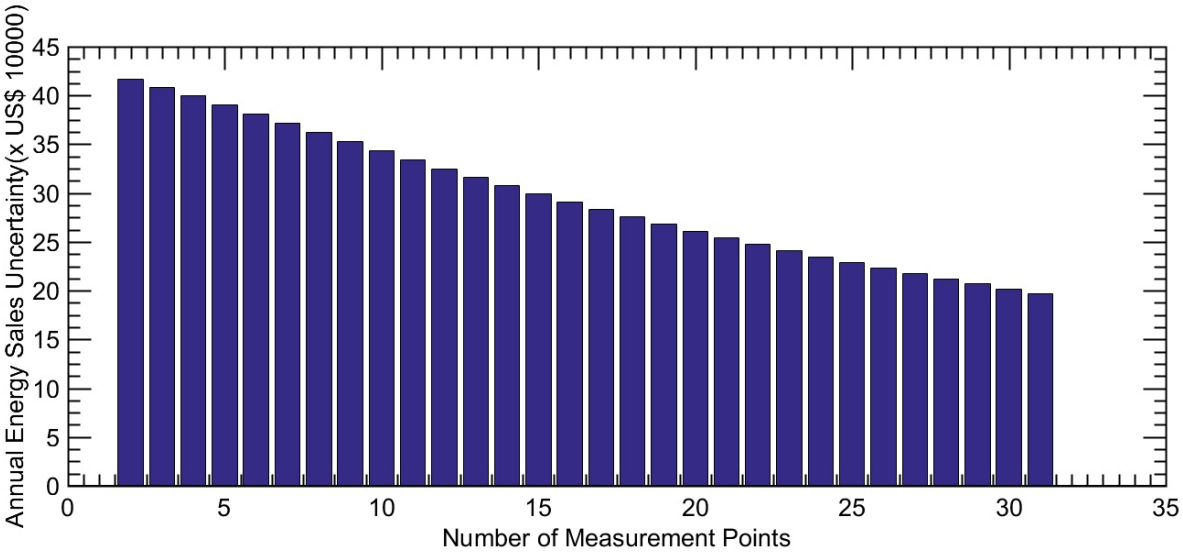


Figure 4.10: Annual energy sales uncertainty of normal size solar park on Leipzig

The downward pattern of annual sales uncertainty curves basically follows the downward pattern of their relative uncertainty curves shown in the previous section. From financial point of view, even though the relative uncertainty of Arizona reaches its saturation condition quicker than Leipzig, Arizona has nearly twice the equivalent sun hours of Leipzig. Due to this, Arizona's solar park energy yield (E) and energy sales uncertainty (U_s) values are generally expected to surpass the ones in Leipzig's solar park. The expectation is proven to be true by the result shown on fig 4.10 and 4.11 although the fact that standard PPA price in Arizona is lower than in Leipzig reduces the overall energy sales uncertainty of Arizona a little. Even though the energy sales uncertainties difference between both solar parks is getting smaller as the number of measurement points increases, Arizona energy sales will never be lower than Leipzig regardless of the number of measurement points. The main reason is because both functions will not intersect unless they are forcefully terminated to return zero relative uncertainty at ideal conditions where the number of measurement points is equal to number of active pixels.

The implicit insight that can be quoted from this discussion is how the optimum amount of measurement points should be chosen. Earlier in this chapter it has been observed that in overall, the imaginary solar park

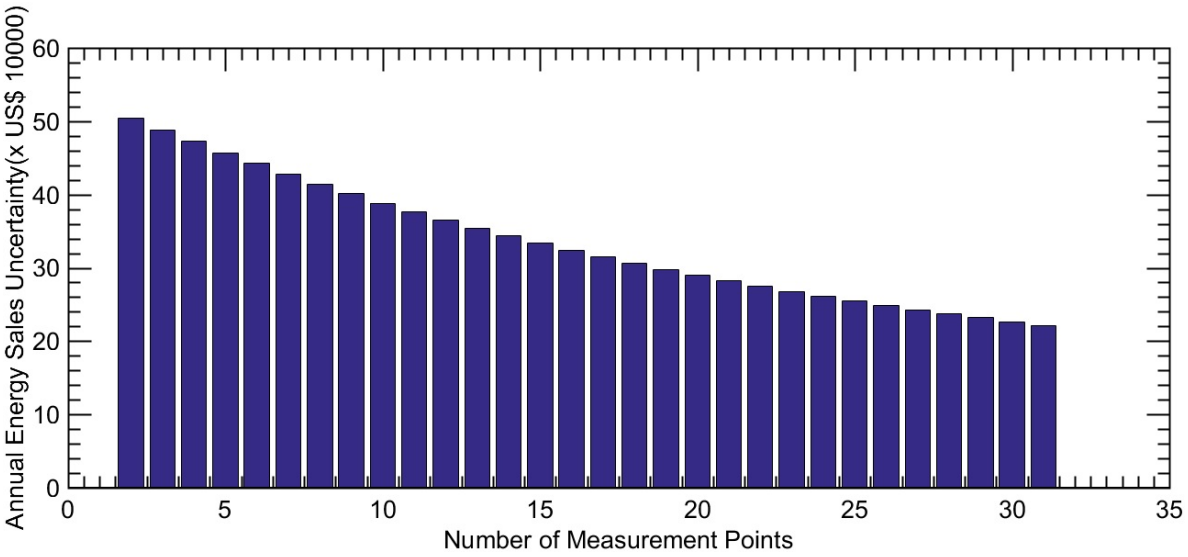


Figure 4.11: Annual energy sales uncertainty of normal size solar park on Arizona

on Arizona reaches its saturation condition quicker than the one in Leipzig. In short, under a similar amount of measurement points, the solar park in Arizona is expected to have less measurement uncertainty compared to Leipzig's solar park. However, this statement does not always hold out when we are seeing it from another angle. From financial point of view for instance, the energy sales uncertainty of Leipzig's solar park in overall is lower than the one on Arizona. These two arguments contradict each other but none of them is on the wrong either. In the end, the technical analysis within this chapter is only a part of the comprehensive information that is required to accurately determine the optimum number of measurement points.

5

CONCLUSION

5.1. CONCLUSION

Finally, a statistical model has been created to study the effect of spatial variation on the accuracy of irradiance measurement within a large scale photovoltaic system. Even though the model only utilizes satellite-derived data as its main input, the model still manages to estimate the change in irradiance measurement uncertainty with regard to the addition of measurement points. Obtaining this information is the first step to determine the optimum number of irradiance measurement devices that should be installed in a certain solar park.

According to several study cases using the model, the total area of solar parks has a prominent contribution to determine the magnitude of measurement uncertainty. Under a constant number of measurement points, the larger the area of the solar park, the higher the measurement uncertainty will be. Another notable contribution is given by local climate characteristics of the region where the solar park has been built. The model shows that under identical settings, a clear-sky condition dominated region induces less measurement uncertainty compared to the region which is not. In addition, temporal resolution of input dataset also gives minor contributions to determine the measurement uncertainty. Finally, the influence of spatial resolution of input dataset could not be checked due to the need of spatial interpolation process validation result which has yet to be done.

In order to satisfy the main objective of the research, the model is also designed to be capable of approximating where the irradiance measurement devices should be placed on the solar park. The suggestion does not specifically state the exact position but it gives the general insight into which area of the solar park the measurement devices would give the optimum contribution to bring down the measurement uncertainty value. Even though it is not yet a perfect model, this statistical model in is generally able to return satisfying answers for all research questions stated in the first chapter.

5.2. FUTURE WORK

As you may notice, the current model makes some crucial assumptions which might have serious effect on the result of the model. The detachment of these assumptions from the model and inclusion of a more sophisticated algorithm could be some ways to improve the model; the following points presents some work that can be done in the future to improve the model:

- Validate the result of the model with real measurement data, especially the results coming from spatial interpolation procedure.
- Inclusion of soiling effect into the model.
- Incorporate anisotropic based analysis into the kriging sub-procedure and check how the results from the new approach influence the overall result of the model.
- Incorporate other combination searching algorithms into the model to see whether there is better measurement points configuration than the one recommended by the greedy algorithm.

- Reconstruct some algorithms to improve the overall processing speed of the model which is currently quite slow even with the help of parallel processing.
- Run the model on several different one year datasets from a similar solar park and see what the inter-annual variation of the result looks like.
- Refine the random number generator in such a way that minimizes the bias of the generated number.
- Add a new spatial interpolation algorithm to analyze datasets with slight spatial variation characteristics (near-uniform sky condition).
- Run the model under different class of irradiance measurement device; *e.g.* Kipp & Zonen CMP 11, CMP 22,etc.

A

APPENDIX-A

A.1. MATLAB SCRIPT INP_INT.M

```
% Input script for the model
% Please adjust the value of variables those are indicated by !!!INPUT!!!
clc

database_name=2; % !!!INPUT!!! Insert '1' if NSRDB database is being used otherwise ...
    insert '2' (HelioClim or CAMS)
max_clust=31; % !!!INPUT!!! Number of maximum cluster to be included into model ...
    evaluation
stde_threshold=0.005; % !!!INPUT!!! Standard deviation threshold for spatial ...
    interpolation; it is an important component to avoid endless loop on kriging process
no_random=5; %!!!INPUT!!! Number of trial for adding built-in error into input data
repetation=10; %!!!INPUT!!! Number of repetation for determining pyranometer ...
    configurations
rmse_pyra=0.0262; % !!!INPUT!!! built-in uncertainty of pyranometer. Kipp&Zonen CMP3
co_int=1; % !!!INPUT!!! Normal distribution confidence interval; insert '1' for ...
    95%,'2' for 98% and '3' for 99%
temporal_res=0.25; % !!!INPUT!!! Temporal resolution of data in hr ->i.e. 15min==0.25hr

if co_int==1
    shape_factor=1.96;
    confd=0.95;
elseif co_int==2
    shape_factor=2.33;
    confd=0.98;
else
    shape_factor=2.56;
    confd=0.99;
end

if database_name==1
    rmse_avg=0.2; % !!!INPUT!!! average of relative RMSE of satellite data for NSRDB
else
    rmse_avg=0.2723; % !!!INPUT!!! average of relative RMSE of satellite data for ...
        HelioClim and CAMS
end


% GPS coordinates list; more input point is possible
p1=[12.3757 51.3356]; % !!!INPUT!!! position in [longitude latitude] format for 1st ...
    corner point
p2=[12.3956 51.3501]; % !!!INPUT!!! position in [longitude latitude] format for 2nd ...
    corner point
p3=[12.411 51.3329]; % !!!INPUT!!! position in [longitude latitude] format for 3rd ...
    corner point
p4=[12.3929 51.3111]; % !!!INPUT!!! position in [longitude latitude] format for 4th ...
    corner point

```

```

coords1=[p1;p2;p3;p4];

run('points_for_leeching.m')

ad=k_c*2*1000;
tot_x=ad*3;
tot_y=ad*3;
x_c=round(tot_x/100);
y_c=round(tot_y/100);
cells=x_c*y_c;
constraint=rmse_avg*shape_factor;
constraint_pyra=rmse_pyra*shape_factor;


clearvars p1 p2 p3 p4 k_c pos_cent1 pos_cent2
```

A.2. MATLAB SCRIPT POINTS_FOR_LEECHING.M

```
% Script to generate coordinates for extracting irradiance data from online
% database

cent_coordinate=zeros(1,2);
geom=polygeom(coords1(:,1),coords1(:,2)); % Call a function to estimate centroid of ...
    2D area
x_cent=geom(2);
y_cent=geom(3);

if database_name==1
   k_c=2.22; % constant spatial resolution of MTSAT
   run('cntr_deg_nsrdb') % center position of NSRDB database
   center=[pos_cent1;pos_cent2];
else
   k_c=(spat_sev(coords1(:,1),coords1(:,2)))/2; % estimating spatial resolution of SEVIRI
   [center1, center2,center3,center4]=cntr_deg_helio(k_c); % center position of ...
       HelioClim and Copernicus database
   center=[center1;center2;center3;center4];
end

x_temp=center(:,1);
y_temp=center(:,2);

centx_deter=x_temp-x_cent;
centy_deter=y_temp-y_cent;

centx_2=centx_deter.^2;
centy_2=centy_deter.^2;

cent_2=centx_2+centy_2;
cent_fin=cent_2.^(0.5);

[M,I]=min(cent_fin);

cent_coordinate(1)=x_temp(I);
cent_coordinate(2)=y_temp(I);

% Generate neighborhood cells coordinates
if database_name==1
centx_min=cent_coordinate(1)-0.04;
centx_max=cent_coordinate(1)+0.04;
centy_min=cent_coordinate(2)-0.04;
centy_max=cent_coordinate(2)+0.04;
else
centx_min=cent_coordinate(1)-(0.00898*2*k_c);
centx_max=cent_coordinate(1)+(0.00898*2*k_c);
centy_min=cent_coordinate(2)-(0.00904*2*k_c);
centy_max=cent_coordinate(2)+(0.00904*2*k_c);
end
```

```
48
49 long_fin=[centx_min;centx_min;centx_min;cent_coordinate(1);cent_coordinate(1)...
50 ;cent_coordinate(1);centx_max;centx_max;centx_max];
51 lat_fin=[centy_max;cent_coordinate(2);centy_min;centy_max;cent_coordinate(2)...
52 ;centy_min;centy_max;cent_coordinate(2);centy_min];
53 cells_coordinate=[long_fin lat_fin];
54
55 %calculates edge points of rectangular 3x3 cells configuration
56 if database_name==1
57 points_range=[(centx_min-0.02) (centx_max+0.02);(centy_min-0.02) (centy_max+0.02)];
58 else
59 points_range=[(centx_min-(0.00898*k_c)) ...
       (centx_max+(0.00898*k_c));(centy_min-(0.00904*k_c)) (centy_max+(0.00904*k_c))];
60 end
61
62 clearvars long_fin lat_fin centx_min centx_max centy_min centy_max cent_coordinate ...
       geom center1 center2 center3 center4 center x_temp y_temp centx_deter centy_deter ...
       centx_2 centy_2 M I cent_2 cent_fin
```

A.3. MATLAB SCRIPT CNTR_DEG_NSRDB.M

```
1  % Grid centre points for NSRDB
2  % The range is in between -66 - 66 degree of latitude and -130 - 130 degree
3  % of longitude
4
5  %complete explanation how this conversion works is presented on the report
6  long=(0.02:0.04:130)';
7  lat=(66.97:-0.04:-66.99)';
8
9  at1=numel(long);
10 at2=numel(lat);
11 at3=at1*at2;
12
13 long_fin=zeros(at3,1);
14 lat_fin=zeros(at3,1);
15
16
17 for u=1:at2
18
19
20 for w=1:at1
21
22     long_fin(at1*(u-1)+w)=long(w);
23     lat_fin(at1*(u-1)+w)=lat(u);
24
25 end
26
27
28 end
29
30 pos_cent1=[long_fin(:) lat_fin(:)];
31 pos_cent2=[((-1)*long_fin(:)) lat_fin(:)];
32
33
34 clearvars x_t y_t u w x_fin y_fin tot_x1 tot_y1 x_c1 y_c1 long lat at1 at2 at3 ...
       long_fin lat_fin
```

A.4. MATLAB SCRIPT SPAT_SEV.M

```
1 %Special script to estimate cells centers for SEVIRI imaging
2 %It is required due to difference on spatial resolution over some coordinates
3
4
5 function k_c=spat_sev(x_any,y_any)
6 %Spatial resolution lookup table(Based on spatial resolution map on HelioClim
```

```
%overview document)

bounds_long=[23.683;31.858;37.628;41.956;47.727;52.055;54.94;59.508;62.393];
bounds_lat=[23.358;31.742;37.492;41.804;48.033;51.866;54.74;59.771;62.407];
spa_res=[3;3.5;4;4.5;5;6;7;8;10];
glad=numel(x_any);

%Check where do the edge points belong to within the lookup table
%Spatial resolution is decided according to the position of this centroid

fog=0;
fug=1;
ko=1;

while fug>0
    x_temp=bounds_long(ko)-(spa_res(ko)*0.00898);
    y_temp=bounds_lat(ko)-(spa_res(ko)*0.00904);
    x_pol=[x_temp;x_temp;((-1)*x_temp);((-1)*x_temp)];
    y_pol=[y_temp;((-1)*y_temp);((-1)*y_temp);y_temp];

    [in,on] = inpolygon(x_any,y_any,x_pol,y_pol);
    fog=fog+sum(in);

    if fog==glad
       k_c=spa_res(ko);
       fug=0;
    end

   ko=ko+1;
end

clearvars x_any y_any fog fug ko in on x_pol y_pol spa_res bounds_lat bounds_long
```

A.5. MATLAB SCRIPT CNTR_DEG_HELIO.M

```
% Grid centre points for HelioClim and CAMS
% The range is in between -66 - 66 degree of longitude and latitude

%complete explanation how this conversion works is presented on the report
function [pos_cent1,pos_cent2,pos_cent3,pos_cent4]=cntr_deg_helio(k_c)


long=((0.00898*k_c):(0.00898*2*k_c):66)';
lat=((0.00904*k_c):(0.00904*2*k_c):66)';

at1=numel(long);
at2=numel(lat);
at3=at1*at2;

long_fin=zeros(at3,1);
lat_fin=zeros(at3,1);


for u=1:at2


for w=1:at1

    long_fin(at1*(u-1)+w)=long(w);
    lat_fin(at1*(u-1)+w)=lat(u);

end


end

pos_cent1=[long_fin(:) lat_fin(:)];
pos_cent2=[((-1)*long_fin(:)) lat_fin(:)];
```

```
pos_cent3=[((-1)*long_fin(:)) ((-1)*lat_fin(:))];
pos_cent4=[long_fin(:) ((-1)*lat_fin(:))];


clearvars x_t y_t u w x_fin y_fin tot_x1 tot_y1 x_c1 y_c1
```

A.6. MATLAB SCRIPT INTERPOLATION_SPATIAL.M

```
%Integration script for interpolating satellite derived irradiance data
%into higher spatial resolution

% Both inputs (raw_clear_sky_irr and raw_GHI) are 9x1 array.
% It is formed from 1 data of center cell and 8 data of surrounding
% cells.
%
% |1|4|7|
% |2|5|8|  --> |1|2|3|4|5|6|7|8|9|
% |3|6|9|
%
% Follow above numbering order to construct input array raw_clear_sky_irr and raw_GHI.
% Each row representing a specific irradiance condition on a certain
% temporal condition
% Number of rows is relative to temporal resolution of dataset -->i.e 30min
% resolution & a year long dataset has 17520 rows which makes the final
% size of raw_GHI or raw_clear_sky_irr array equals to 17520x9


clc
run('inp_int.m') % calling all input parameters
run('clear_sky_irr.m') % seperating clear sky irradiance of non-clear sky condition ...
    from database
amo_dat=size(alikl6_fin,1);

csi_array=zeros(amo_dat,9,(no_random+1));
abs_err_ovr=zeros(9,2,no_random);
abs_err_pyr=zeros(9,2,no_random);
abs_err_sat=zeros(9,2,no_random);
rel_err_ovr=zeros(9,2,no_random);
rel_err_pyr=zeros(9,2,no_random);
rel_err_sat=zeros(9,2,no_random);

% adding simulated rmse of satellite and pyranometer into original dataset
for trial=1:no_random

    if trial==2
        csi_array(:,:,1)=unmodified_csi;
    end

    % calling a randomizer to simulate additional built-in RMSE of
    % satellite data
    run('randomized_v_1_1.m')
    % calling a randomizer to simulate additional built-in RMSE of
    % pyranometer
    run('randomized_v_1_2.m')
    csi_array(:,:,(trial+1))=alikl6_1;
    abs_err_ovr(:,:,trial)=abs_res_ovr;
    abs_err_pyr(:,:,trial)=abs_res_pyr;
    abs_err_sat(:,:,trial)=abs_res_sat;
    rel_err_ovr(:,:,trial)=rel_res_ovr;
    rel_err_pyr(:,:,trial)=rel_res_pyr;
    rel_err_sat(:,:,trial)=rel_res_sat;

end

pos_est=cntr_coord(tot_x,tot_y,x_c,y_c);

%calling a function to convert coordinates degree into meter
coords2=deg2met(coords1,points_range,ad);
```

```
60
61 % creating an untouched 'dies' array to represent the original geometry shape of solar
62 % park
63 val_cell=multipoints_array(pos_est,coords2);
64
65 % creating a final 'dies' array to represent the original geometry shape of solar
66 % park
67 dies1=fin_dies(coords2,val_cell,x_c,y_c);
68 luke=numel(val_cell);
69 luke1=numel(dies1);
70
71 % converting final result array into 'single' to reduce the memory
72 zoom_fin_arr1=zeros(luke1,amo_dat,(no_random+1));
73 fin_arr1=zeros(luke,amo_dat,(no_random+1));
74 zoom_fin_arr=single(zoom_fin_arr1);
75 fin_arr=single(fin_arr1);
76 clearvars zoom_fin_arr1 fin_arr1
77
78 % looping spatial interpolation process for
79 % original dataset + simulated dataset
80 for hug=1:(no_random+1)
81 fin_data=zeros(cells,amo_dat);
82 csi_fin_1=csi_array(:,:,hug);
83
84 % looping spatial interpolation process throughout a year
85 % (the amount depends on temporal resolution of the dataset)
86 parfor t=1:amo_dat
87
88     %Processing the dataset for each temporal data points
89     val_known=(csi_fin_1(t,:))';
90
91     % calculating spatial rmse of data points
92     m2=moment(val_known,2);
93     stde=sqrt(m2);
94
95     tot=sum(val_known);
96
97     %checking whether there is still remaining diurnal condition within the
98     %dataset
99     if tot==0
100         fin_data(:,t)=0;
101     end
102
103
104     if tot≠0 && stde≤stde_threshold
105         %model returns ONE array (array filled with 1) when the detected
106         %sky condition is either uniform or nearly uniform
107         %(could be clear-sky or cloudy sky)
108         fin_data(:,t)=ones(cells,1);
109
110     else if tot≠0 && stde>stde_threshold
111         %model proceeds to kriging process when non-uniform sky condition
112         %is detected (default:spatial_rmse>0.005)
113
114         %calling function to calculate semivariance of each pair of dataset
115         %member
116         smvar=semvar(val_known);
117
118         %calling function to calculate distances between cells of dataset
119         %member
120         [dis_cells,pos_known]=dist_cent_cells(ad);
121
122         %calling function to calculate new coordinates of dataset
123         %the coordinates represent the new spatial resolution of dataset
124         pos_est=cntr_coord(tot_x,tot_y,x_c,y_c);
125
126         %calling function to make averages of grouped semivariance values
127         [real_smvar1,dist1]=avg_smvrg(ad,dis_cells,smvar);
128
129         %check semivariance values from outlier of preference model
130         xac=find(real_smvar1==0);
```

```
131          xab=7-numel(xac);
132          real_smvar=zeros(xab,1);
133          dist=zeros(xab,1);
134          xai=1;
135
136          for m1=2:6
137          xah=real_smvar1(m1);
138
139          if xah≠0
140             xai=xai+1;
141             real_smvar(xai)=real_smvar1(m1);
142             dist(xai)=dist1(m1);
143          end
144
145          end
146          %checking ends here
147
148
149          if xab<4
150              %when less than 4 datapoints are available at this point, the
151              %model will abort interpolation process and kick out this
152              %particular temporal dataset.
153
154              kick=1;
155              V=[];
156          else
157              %otherwise the model proceeds to determining the most suitable
158              %semivariogram model for this particular temporal dataset
159              [V,kick]=deter_smvgr(real_smvar,dist,ad);
160          end
161
162          if kick≥1
163              %kick-out dataset will be labelled as zero array inside the
164              %final result data array
165              fin_data(:,t)=0;
166          else
167              %when everthing is okay, the model finally proceeds to kriging
168              %function
169              [d_est,d_var]=krig(pos_known,val_known,pos_est,V);
170              fin_data(:,t)=d_est;
171          end
172          end
173      end
174
175
176  end
177
178  %calling a function to zoom in the data array into necessary range and
179  %together apply the original geometry of solar park into data array
180  run('zoom_post.m')
181
182  fin_arr(:,:,hug)=fin_data;
183  zoom_fin_arr(:,:,hug)=zoom_fin_data;
184
185  end
186
187
188  clearvars constraint constraint_pyra zoom_fin_data d_est prac tot stde csi_fin1 x1 x2 ...
         aa bb cc dd ee kick m2 nrmse_bench ad tot_x tot_y x_c y_c cells amo_dat t ...
         val_known sum_temp d_var dis_cells pos_est pos_known smvar nrmse_benchmark h
```

A.7. MATLAB SCRIPT CLEAR_SKY_IRR.M

```
1  clc
2  % Script to build clear sky irradiance array (alikl6)
3  % It is important for deriving absolute uncertainty in W/M2 unit
4
5  polol=raw_GHI./raw_clear_sky_irr;
```

```
alikl=sum(polol,2);
alikl1=isnan(alikl);
alikl2=find(alikl1==0);
alikl3=alikl(alikl2);
alikl4=find(alikl3≠9);

alikl5=numel(alikl2);
alikl6=[];

for alikl7=1:alikl5
    alikl8=alikl2(alikl7);
    alikl9=raw_clear_sky_irr(alikl8,:);
    alikl6=[alikl6;alikl9];
end

if database_name==1
alikl6_fin=alikl6; % already in instanteneous irradiance (W/m2)
else
alikl6_fin=alikl6/temporal_res; % conversion from irradiation (Wh/m2) @ each 15min to ...
    instanteneous irradiance (W/m2) within that 15 min
end
clearvars polol alikl alikl1 alikl2 alikl2 alikl3 alikl4 alikl5 alikl6 alikl7 alikl8 ...
    alikl9
```

A.8. MATLAB SCRIPT RANDOMIZED_V_1_1.M

```
clc
% Simulation of built-in rmse of satellite data

dist_type=1;

% excluding diurnal condition (CSI==0) from the original dataset
polol=raw_GHI./raw_clear_sky_irr;
alikl=sum(polol,2);
alikl1=isnan(alikl);
alikl2=find(alikl1==0);


alikl5=numel(alikl2);
alikl6=[];
alikl61=[];
for alikl7=1:alikl5
    alikl8=alikl2(alikl7);
    alikl9=polol(alikl8,:);
    alikl91=raw_clear_sky_irr(alikl8,:);
    alikl6=[alikl6;alikl9];
    alikl61=[alikl61;alikl91];
end

unmodified_csi=alikl6;

% temporarily excluding clear sky condition from dataset
juke0=size(alikl6);
juke=(alikl6)';
juke1=std(juke);
[juk2,juke3]=find(juke1≠0);

lupin=numel(juke3);
lupin1=[];
lupin101=[];

for lupin2=1:lupin
    juke4=juke3(lupin2);
    juke5=alikl6(juke4,:);
    juke51=alikl61(juke4,:);
    lupin1=[lupin1;juke5];
    lupin101=[lupin101;juke51];
end
```

```
43
44 if database_name==1
45 lupin102=lupin101;
46 else
47 lupin102=lupin101*4;
48 end
49
50 fgsake=size(lupin1);
51 length=fgsake(2);
52 random_length=fgsake(1);
53
54 rand_factor=[];
55
56 % Adding controlled random value into original satellite dataset
57 for roi=1:length
58
59    if dist_type==1
60        rng shuffle
61        abk1=-1+(1+1)*rand(random_length,1);
62    else
63        rng shuffle
64        abk=randn(random_length,1);
65        abk2=abs(abk);
66        abk3=max(abk2);
67        abk1=abk/abk3;
68    end
69
70    rand_fact1=abk1*constraint;
71    rand_factor=[rand_factor rand_fact1];
72 end
73
74 premodified_csi=[];
75 for rope=1:length
76     ahu=rand_factor(:,rope);
77     ahu1=lupin1(:,rope);
78     ahu2=ahu1.*ahu;
79     ahu3=ahu1+ahu2;
80     premodified_csi=[premodified_csi ahu3];
81 end
82
83 % Limiting datapoints whose CSI value is more than 1 or less than 0
84 % (impossible to happen in actual practice)
85 premodified_csi(premodified_csi>1)=1;
86 premodified_csi(premodified_csi<0)=0;
87
88
89 %Incorporate non-clear sky condition data with clear-sky condtion data
90 for hope=1:lupin
91  gokil=premodified_csi(hope,:);
92  gokil1=juke3(hope);
93  alikl6(gokil1,:)=gokil;
94 end
95
96 %checking the rmse of newly generated dataset
97 %please check it before proceed to next step
98 [abs_res_sat,rel_res_sat]=rmse_check1(premodified_csi,lupin102,lupin1);
99
100
101 clearvars lupin101 juke51 roi shape_factor alikl61 alikl91 juke0 hope gokil gokil1 ...
       abk abk1 abk2 abk3 rand_fact1 rope ahu ahu1 ahu2 ahu3 fgsake juke juke1 juk2 ...
       lupin2 juke4 juke5 polol alikl alikl1 alikl2 alikl2 alikl3 alikl4 alikl5 alikl7 ...
       alikl8 alikl9
```

A.9. MATLAB SCRIPT RMSE_CHECK1.M

```
1 %function to check bias and rmse of new-created data
2 %this function calculates bias and rms error of every cell in temporal
3 %realm
```

```
function [abs_res,rel_res]=rmse_check1(premodified_csi,lupin101,lupin1)

vis=size(premodified_csi);
vis1=vis(2);
rel_res=[];
abs_res=[];

%complete formulas for bias and rms error are available on the report
for vis2=1:vis1
    temp_csi_fin=premodified_csi(:,vis2);
    temp_csi_fin1=(premodified_csi(:,vis2)).*(lupin101(:,vis2));
    temp_un_csi=lupin1(:,vis2);
    temp_un_csi1=(lupin1(:,vis2)).*(lupin101(:,vis2));
    avg_model=mean(temp_un_csi);
    amo=numel(temp_un_csi);
    devi=temp_un_csi-temp_csi_fin;
    devi1=temp_un_csi1-temp_csi_fin1;

    %absolute bias error in W/m2
    bias=((sum(devi))/amo)/avg_model;
    %relative bias error
    bias1=((sum(devi1))/amo);
    %absolute rms error in W/m2
    rmse=(sqrt((sum(devi.^2))/amo))/avg_model;
    %relative rms error
    rmse1=(sqrt((sum(devi1.^2))/amo));
    temp_res=[bias rmse];
    temp_res1=[bias1 rmse1];

    %final array of absolute error
    abs_res=[abs_res;temp_res1];
    %final array of relative error
    rel_res=[rel_res;temp_res];
end

clearvars vis vis1 vis2 temp_csi_fin temp_un_csi avg_model amo devi bias rmse temp_res
```

A.10. MATLAB SCRIPT RANDOMIZED_V_1_2.M

```
clc
% Simulation of built-in rmse of pyranometer

alikl6_1=unmodified_csi;
rand_factor1=[];

% Adding controlled random value into modified satellite dataset (original
% + simulated rmse)
for roi=1:length

   if dist_type==1
       rng('shuffle')
       abk1=-1+(1+1)*rand(random_length,1);
   else
       rng('shuffle')
       abk=randn(random_length,1);
       abk2=abs(abk);
       abk3=max(abk2);
       abk1=abk/abk3;
   end

   rand_fact1=abk1*constraint_pyra;
   rand_factor1=[rand_factor1 rand_fact1];
end

premodified_csi1=[];
for rope=1:length
    ahu=rand_factor1(:,rope);
```

```
    ahu1=premodified_csi(:,rope);
    ahu2=ahu1.*ahu;
    ahu3=ahu1+ahu2;
    premodified_csi1=[premodified_csi1 ahu3];
end

% Limiting datapoints whose CSI value is more than 1 or less than 0
% (impossible to happen in actual practice)
premodified_csi1(premodified_csi1>1)=1;
premodified_csi1(premodified_csi1<0)=0;


for hope=1:lupin
 gokil=premodified_csi1(hope,:);
 gokil1=juke3(hope);
 alikl6_1(gokil1,:)=gokil;
end

[abs_res_pyr,rel_res_pyr]=rmse_check1(premodified_csi1,lupin102,premodified_csi);
[abs_res_ovr,rel_res_ovr]=rmse_check1(premodified_csi1,lupin102,lupin1);
clearvars length lupin lupin102 dist_type rmse_avg rmse_pyra juk3 roi abk abk1 abk2 ...
    abk3 rand_fact1 rope ahu ahu1 ahu2 ahu3 hope gokil gokil1
```

A.11. MATLAB SCRIPT CNTR_COORD.M

```
%building coordinates array for 100x100 m spatial resolution of data array
%each coordinate is represented by its respective 100x100m cell's centroid
%position

function pos_est1=cntr_coord(tot_x1,tot_y1,x_c1,y_c1)

x_t=[50:100:tot_x1];
y_t=[(tot_y1-50):-100:0];

x_fin=zeros(1,x_c1);
y_fin=zeros(1,y_c1);


for u=1:x_c1


for w=1:y_c1

    x_fin(x_c1*(u-1)+w)=x_t(u);
    y_fin(x_c1*(u-1)+w)=y_t(w);

end


end

pos_est1=[x_fin(:) y_fin(:)];

clearvars x_t y_t u w x_fin y_fin tot_x1 tot_y1 x_c1 y_c1
```

A.12. MATLAB SCRIPT DEG2MET.M

```
% Function to convert latitude and longitude coordinate into
% metric cartesian coordinate

function coord3=deg2met(coord4,points_range1,ad9)

aha=points_range1(1,2)-points_range1(1,1);
aja=points_range1(2,2)-points_range1(2,1);

```

```
tot_int_m=3*ad9;

xm_fin=((coord4(:,1)-points_range1(1,1))/aha)*tot_int_m;
ym_fin=((coord4(:,2)-points_range1(2,1))/aja)*tot_int_m;

coord3=[xm_fin ym_fin];

clearvars coord4 points_range1 ad9 aha aja tot_int_m xm_fin ym_fin
```

A.13. MATLAB SCRIPT MULTIPOINTS_ARRAY.M

```
% Script to estimate actual geometry of the solar park by small rectangular cell
% The basic algorithm is to represent each cell by 5 center points.
% Whenever 2 points or more are lying inside the polygon of the PV site
% that will be appointed as the valid one.

function val_cell1=multipoints_array(pos_est1,coords2)

aka=size(pos_est1);
aka1=aka(1);

tested_points_x=zeros(5,aka1);
tested_points_y=zeros(5,aka1);
val_cell1=zeros(aka1,1);


j1=1;

while j1<(aka1+1)
    x_catch=pos_est1(j1,1);
    y_catch=pos_est1(j1,2);

    tested_points_x(1,j1)=x_catch;
    tested_points_x(2,j1)=x_catch-25;
    tested_points_x(3,j1)=x_catch-25;
    tested_points_x(4,j1)=x_catch+25;
    tested_points_x(5,j1)=x_catch+25;

    tested_points_y(1,j1)=y_catch;
    tested_points_y(2,j1)=y_catch+25;
    tested_points_y(3,j1)=y_catch-25;
    tested_points_y(4,j1)=y_catch+25;
    tested_points_y(5,j1)=y_catch-25;

    j1=j1+1;
end


x_any=reshape(tested_points_x,[(5*aka1),1]);
y_any=reshape(tested_points_y,[(5*aka1),1]);

x_pol=coords2(:,1);
y_pol=coords2(:,2);

%calling built-in function for determining which points are located inside
%polygon's boundaries
[in,on] = inpolygon(x_any,y_any,x_pol,y_pol);

candidate=in+on;

can_fin=reshape(candidate,[5,aka1]);

j2=1;

%Determining which cells are still within solar park geometry
while j2<(aka1+1)
    tsy=sum(can_fin(:,j2));

```

```
    % val_cell array represents solar park geometries within rectangular frame.
    % Passive data (won't be included into analysis) is presented by zero
    % while active data is presented by one.
    if tsy≥2
        val_cell1(j2)=1;
    else
        val_cell1(j2)=0;
    end

    j2=j2+1;

end

clearvars pos_est1 coords2 candidate in on j1 j2 tsy can_fin coords1 x_pol y_pol ...
    x_any y_any x_catch y_catch tested_points_x tested_points_y aka aka1
```

A.14. MATLAB SCRIPT FIN_DIES.M

```
function [dies1,dies]=fin_dies(coords2,val_cell,x_c,y_c)

a1=(max(coords2(:,1)))/100;
b1=(min(coords2(:,1)))/100;
c1=(max(coords2(:,2)))/100;
d1=(min(coords2(:,2)))/100;

a=ceil(a1);
b=ceil(b1);
c=ceil(c1);
d=ceil(d1);

e=(a-b+1)*(c-d+1);

left_posx=b;
right_posx=a;
bottom_posy=y_c-d+1;
up_posy=y_c-c+1;


val_cell1=reshape(val_cell,[x_c,y_c]);
dies=val_cell1(up_posy:bottom_posy,left_posx:right_posx);
dies1=reshape(dies,[e,1]);
```

A.15. MATLAB SCRIPT SEMVAR.M

```
%Calculate semivariance of each temporal datapoints


function smvar1=semvar(val_known1)

smvar1=zeros(81,1);


for dts=1:9

% complete semivariance formula is presented on the report
for g=1:9

    i=val_known1(dts)-val_known1(g);
    if i==0
        i_2=0;
    else
        i_2=i^2;
    end

    smvar1((dts-1)*9+g)=i_2;
```

```
22
23  end
24
25  end
26
27  clearvars dts g i i_2 val_known1
```

A.16. MATLAB SCRIPT DIST_CENT_CELLS.M

```
1   %function to calculate the cartesian distance between one cell's center to
2   %another. The unit of distance is meter
3
4   function [dis_cells1,pos_known1]=dist_cent_cells(ad1)
5
6   %Distance between center cells (3x3 cells arrangement)
7
8   dis_cells1=zeros(81,1);
9   pos_x=zeros(9,1);
10  pos_y=zeros(9,1);
11
12  % Cells centers position in x-axis
13  pos_x(1:3)=0.5*ad1;
14  pos_x(4:6)=1.5*ad1;
15  pos_x(7:9)=2.5*ad1;
16
17  % Cells centers position in y-axis
18  pos_y(1)=2.5*ad1;
19  pos_y(2)=1.5*ad1;
20  pos_y(3)=0.5*ad1;
21  pos_y(4)=2.5*ad1;
22  pos_y(5)=1.5*ad1;
23  pos_y(6)=0.5*ad1;
24  pos_y(7)=2.5*ad1;
25  pos_y(8)=1.5*ad1;
26  pos_y(9)=0.5*ad1;
27
28
29
30  for b=1:9
31
32  for a=1:9
33
34      c=pos_x(b)-pos_x(a);
35      if c==0
36          c_2=0;
37      else
38          c_2=c^2;
39      end
40
41      d=pos_y(b)-pos_y(a);
42       if d==0
43          d_2=0;
44      else
45          d_2=d^2;
46       end
47
48      e=c_2+d_2;
49      if e==0
50          e_rt2=0;
51      else
52          e_rt2=sqrt(e);
53      end
54
55      dis_cells1((b-1)*9+a)=e_rt2;
56
57  end
58
59  end
```

```
60
61 pos_known1=[pos_x(:) pos_y(:)];
62
63 clearvars pos_x pos_y a b c c_2 d d_2 e e_rt2 ad1
```

A.17. MATLAB SCRIPT AVG_SMVRG.M

```
1  %The script for averaging semivariance for deriving semivariagram model
2  %Since there are several semivariance values grouped within the same
3  %distance
4
5
6  function [real_smvar1,dist1]=avg_smvrg(ad2,dis_cells2,smvar2)
7
8  real_smvar1=zeros(6,1);
9  dist1=zeros(6,1);
10 p1=0;
11 p2=0;
12 p3=0;
13 p4=0;
14 p5=0;
15
16
17 g1=ad2*2;
18 g2=ad2*sqrt(2);
19 g3=ad2*sqrt(5);
20 g4=ad2*2*sqrt(2);
21
22 for z=1:81
23     x=dis_cells2(z);
24     if x==ad2
25        p1=p1+smvar2(z);
26     else if x==g1
27             p2=p2+smvar2(z);
28            else if x==g2;
29                    p3=p3+smvar2(z);
30                 else if x==g3;
31                         p4=p4+smvar2(z);
32                     else if x==(2*ad2*sqrt(2));
33                             p5=p5+smvar2(z);
34                         else
35                             p0=0;
36                         end
37                     end
38                end
39         end
40     end
41
42
43 end
44
45 % All possible distance combinations from 3x3 cells configuration
46 dist1(2)=ad2;%horizontally and vertically 1 cell ahead
47 dist1(3)=g1;%horizontally and vertically 2 cells ahead
48 dist1(4)=g2;%diagonally with this combination: 1 cell horizontal & 1 cell vertical
49 dist1(5)=g3;%diagonally with these combinations: 1 cell horizontal & 2 cells vartical ...
       and the other way around
50 dist1(6)=g4;%diagonally with this combination: 2 cells horizontal & 2 cells vertical
51
52 real_smvar1(2)=p1/48; %average semivariance value for dist1(2)
53 real_smvar1(3)=p2/24; %average semivariance value for dist1(3)
54 real_smvar1(4)=p3/32; %average semivariance value for dist1(4)
55 real_smvar1(5)=p4/32; %average semivariance value for dist1(5)
56 real_smvar1(6)=p5/8; %average semivariance value for dist1(6)
57
58 clearvars x z p0 p1 p2 p3 p4 p5 g1 g2 g3 g4 ad2 dis_cells2 smvar2
```

A.18. MATLAB SCRIPT DETER_SMVGR.M

```
% Semivariogram model fitting, only following isotropic semivariogram
% models are included in this model; spherical, exponential, and Gaussian.

function [V1,kick1]=deter_smvgr(real_smvar3,dist3,ad4)

range=max(real_smvar3)-min(real_smvar3);


%Loop1 - gaussian -->train the empirical semivariance values with Gaussian
a=0.2; %first estimation of fitting rmse
kick1=0; %control variable to determine whether the dataset will be kicked out or not
n=1;
k=0;
while n>0

k=k+100;

if a>0.25 %upper limit of fitting rmse
    a=1;
    %when the fitting fails (fitting rmse>0.25), this model will be
    %excluded from analysis
    kick1=1;
else
    kick1=0;
end


%complete formula of gaussian,spherical,and exponential semivariogram are
%presented on the report
%calling gaussian function to be fitted with empirical semivariance
ft1 = fittype( 'gaussian(x,a,c)' );
%calling built-in curve fitting function from matlab
[curve1,gof1] = fit(dist3,real_smvar3,ft1,'StartPoint',[k,0]);

%calculating normalized rmse of curve fitting result
nrmse=(gof1.rmse)/range;
sill_temp=coeffvalues(curve1);
prac=sill_temp(1);

%assigning semivariogram from curve fitting result
if prac==100 && a==1
    prac_range_fin1=round(prac,0);
    sill1=sill_temp(2);
    n=0;
else if prac==100 && nrmse≤a %condition where spatial rmse is smaller than
k=ad4-100;
n=1;

else if prac>50000
    a=a+0.01;
    k=0;
    n=1;
else if prac>100 && nrmse≤a
    n=0;
    prac_range_fin1=round(prac,0);
    sill1=sill_temp(2);
else
    n=1;
    end
    end
    end
end

end

if sill1>1
    prac_range_fin1=0;
```

```
end



%Loop2 - spherical -->train the empirical semivariance values with
%Spherical
n=1;
k=0;

while n>0

k=k+100;
%calling spherical function to be fitted with empirical semivariance
ft2 = fittype( 'spherical(x,a,c)' );
%calling built-in curve fitting function from matlab
[curve2,gof2] = fit(dist3,real_smvar3,ft2,'StartPoint',[k,0]);

%calculating normalized rmse of curve fitting result
nrmse=(gof2.rmse)/range;
sill_temp=coeffvalues(curve2);
prac=sill_temp(1);

%assigning semivariogram from curve fitting result
if prac>50000
    a=1;
    prac=110;
end


if prac==100 && nrmse≤a
k=ad4-100;
n=1;

else if prac>100 && nrmse≤a
    n=0;
    prac_range_fin2=round(prac,0);
    sill2=sill_temp(2);
else
    n=1;
    end
end

end

if sill2>1
    prac_range_fin2=0;
end

%Loop3 - exponential -->train the empirical semivariance values with
%exponential
n=1;
k=0;
while n>0

k=k+100;

%calling exponential function to be fitted with empirical semivariance
ft3 = fittype( 'exponential(x,a,c)' );
%calling built-in curve fitting function from matlab
[curve3,gof3] = fit(dist3,real_smvar3,ft3,'StartPoint',[k,0]);

%calculating normalized rmse of curve fitting result
nrmse=(gof3.rmse)/range;
sill_temp=coeffvalues(curve3);
prac=sill_temp(1);

%assigning semivariogram from curve fitting result
if prac>50000
    a=1;
    prac=110;
end
```

```
139
140 if prac==100 && nrmse≤a
141 k=3900;
142 n=1;
143
144 else if prac>100 && nrmse≤a
145     n=0;
146     prac_range_fin3=round(prac,0);
147     sill3=sill_temp(2);
148 else
149     n=1;
150     end
151 end
152
153
154 end
155
156 if sill3>1
157     prac_range_fin3=0;
158 end
159
160
161 compare=[prac_range_fin1;prac_range_fin2;prac_range_fin3];
162
163
164 [M,I] = max(compare);
165
166 % kick-out the temporal dataset when the model cannot find satisfying model
167 % within empirical semivariance
168 if M==0
169     kick1=kick1+1;
170 end
171
172 %pick out semivariogram model with largest practical range value
173 if I==1
174    form_v = '%d Gau(%d)';
175    V1 = sprintf(form_v,sill1,prac_range_fin1);
176    else if I==2
177              form_v = '%d Sph(%d)';
178              V1 = sprintf(form_v,sill2,prac_range_fin2);
179              else
180                  form_v = '%d Exp(%d)';
181                  V1 = sprintf(form_v,sill3,prac_range_fin3);
182        end
183 end
184
185
186 clearvars real_smvar3 dist3 a n k ft1 ft2 ft3 curve1 curve2 curve3 gof1 gof2 gof3 M I ...
      form_v compare prac_range_fin1 prac_range_fin2 prac_range_fin3 sill1 sill2 sill3 ...
      nrmse range sill_temp
```

A.19. MATLAB SCRIPT GAUSSIAN.M

```
1  function y = gaussian(x,a,c)
2  % gaussian semivariogram model for curve fitting purpose
3
4  y = zeros(size(x));
5
6
7  for i = 1:length(x)
8
9      y(i) = c*(1-exp(-3*(x(i)^2)/(a^2)));
10
11 end
```

A.20. MATLAB SCRIPT EXPONENTIAL.M

```
function y = exponential(x,a,c)
% exponential semivariogram model for curve fitting purpose

y = zeros(size(x));


for i = 1:length(x)

    y(i) = c*(1-exp(-3*x(i)/a));

end
```

A.21. MATLAB SCRIPT SPHERICAL.M

```
function y = spherical(x,a,c)
% spherical semivariogram model for curve fitting purpose

y = zeros(size(x));


for i = 1:length(x)
    if x(i) ≤ a,
        y(i) = (c*1.5/a)*x(i)-(0.5*c/(a^3))*(x(i)^3);
    else
        y(i) = c;
    end
end
end
```

A.22. MATLAB SCRIPT ZOOM_POST.M

```
%Script for zooming in the data set
%The data array will be zoomed in into smaller rectangular frame to reduce
%the use of memory for storing large size variable
%smaller frame is made from edge cells of previously created 'dies' array
%named val_cell


a1=(max(coords2(:,1)))/100;
b1=(min(coords2(:,1)))/100;
c1=(max(coords2(:,2)))/100;
d1=(min(coords2(:,2)))/100;

a=ceil(a1);
b=ceil(b1);
c=ceil(c1);
d=ceil(d1);

e=(a-b+1)*(c-d+1);

left_posx=b;
right_posx=a;
bottom_posy=y_c-d+1;
up_posy=y_c-c+1;

zoom_fin_data=zeros(e,amo_dat);


k1=1;
while k1<(amo_dat+1)
temp1=(fin_data(:,k1)).*val_cell;
```

```
fin_temp1=reshape(temp1,[x_c,y_c]);
range_furt=fin_temp1(up_posy:bottom_posy,left_posx:right_posx);
abc=reshape(range_furt,[e,1]);

%zoom_fin_data array is the 'ready' to use data array for next step
zoom_fin_data(:,k1)=abc;

k1=k1+1;
end

clearvars abc fin_temp1 temp1 k1 a b c d e coords p1 p2 p3 p4 a1 b1 c1 d1 left_posx ...
    right_posx up_posy bottom_posy
```

A.23. MATLAB SCRIPT POST_OPT_PYRA.M

```
%Script to estimate pyranometers locations configuration on solar park
%Two main steps are involved; clustering process using Kmeans and
%combination searching process using Greedy algorithm
clc

zoom_fin_data=zoom_fin_arr(:,:,1); % calling unmodified interpolated data array

atb=size(zoom_fin_data);
atb1=atb(2);
atb6=atb(1);
atb1000=max_clust-1;
fin_cent_prob=zeros(atb6,atb1000);


%Loop of process to estimate the probability of each cells being centroids
%throughout a year, clustering process is used as primary tool to define the
%centroid cells
parfor kao=1:atb1

fin_cent_prob_temp=zeros(atb6,atb1000);
atb2=zoom_fin_data(:,kao);

for nopyr=2:max_clust %clustering looping for different number of pyranometer
    atb99=nopyr-1;
    atb3=nonzeros(atb2);
    atb4=mean(atb3);
    atb5=numel(atb3);
    atb7=atb2;
    atb7(atb7≠0)=1;

    %this condition belongs for kicked-out data set
    %it is presented by zero array and won't be included into analysis
    if atb5==0
        temp_prob1=zeros(atb6,1);
        fin_cent_prob_temp(:,atb99)=temp_prob1;
    end

    %this condition belongs to uniform or near uniform sky condition
    %all active cells hold same probability of being centroids therfore the
    %probability of each cell will equal to number of pyranometer/number of active
    %cells
    if atb4==1
        temp_prob=zeros(atb6,1);
        temp_prob(:,1)=(nopyr/atb5);
        temp_prob1=temp_prob.*atb7;
        fin_cent_prob_temp(:,atb99)=temp_prob1;
    end

    %this condition belongs to non-uniform sky condition
    %the data will be passed on into Kmeans algorithm
    if atb4≠1 && atb5≠0

       %calling function to do
```

```
54          temp_prob1=prob_hist2d_1(atb2,nopyr,dies1);
55          fin_cent_prob_temp(:,atb99)=temp_prob1;
56      end
57
58  end
59
60  %the probability value for each temporal data set is then cumulatively
61  %added up until final temporal data set of data array has been processed
62  fin_cent_prob=fin_cent_prob+fin_cent_prob_temp;
63  end
64
65
66  atb_x=sum(fin_cent_prob);
67  atb_z=nonzeros(dies1);
68  atb_w=numel(atb_z);
69  atb_xw=atb_x/atb_w;
70
71  %calling a function to randomize two starting points of greedy searching
72  %algorithm
73  run('geo_points_starter.m');
74  distance_fin1=zeros(atb_w,atb_w,atb1000);
75  distance_fin2=distance_fin1;
76  distance_fin3=distance_fin1;
77  atb501=size(range_furt);
78  atb502=atb501(1);
79  atb503=atb501(2);
80  fin_pos_rep=zeros(max_clust,atb1000,repetation);
81  final_pos_rep=zeros(max_clust,atb1000,repetation,2);
82  final_lump_post1=zeros(repetation,atb1000,2);
83
84  hintul = waitbar(0,'Please wait, data is being processed...');
85  stepus=repetation*(max_clust-1)*2;
86  for ct=1:2 %looping for absolute (ct==1) and relative (ct==2) uncertainty based analysis
87  for nopyra=2:max_clust
88      temp1=[];
89      atb100=nopyra-1;
90      atb101=nopyra-2;
91      atb200=fin_cent_prob(:,atb100);
92      atb_y=atb_xw(atb100);
93
94      %filtering the cells that whose normalized sum of probability are less
95      %than normalized sum of probability of uniform sky condition throughout
96      %a year
97      [B1,I1]=sort(atb200,'descend');
98      norm_b2=B1/atb_y;
99      fus=[I1 norm_b2];
100
101     atb_hi=find(norm_b2>1);
102     atb_hi_fin=max(atb_hi);
103     prob_stand=(1:atb_hi_fin)';
104
105     % Upon omitting the datasets that equals or less than average of sum of probability
106     filter_1=fus(1:atb_hi_fin,:);
107
108     coord_f1=filter_1(:,1);
109
110     if nopyra==2
111
112     % Searching for the closest cartesian coordinates of generated random
113     % points that is within coord_f1 data array
114     temp_ps1=zeros(2,2,repetation);
115
116     for xoc=1:repetation
117         temp_ps3=[];
118     for xok=1:2
119         I_ps=post_fin(xok,1,xoc);
120         yuno1=ceil(I_ps/atb502);
121         yuno2=rem(I_ps,atb502);
122
123         if yuno2==0
124             yuno3=atb502;
```

```
        else
            yuno3=yuno2;
        end

        temp_ps2=[yuno1 yuno3];
        temp_ps3=[temp_ps3;temp_ps2];
    end
    temp_ps1(:,:,xoc)=temp_ps3;
    end


    for kao5=1:atb_hi_fin
        I2=filter_1(kao5,1);
        yuno1=ceil(I2/atb502);
        yuno2=rem(I2,atb502);

        if yuno2==0
            yuno3=atb502;
        else
            yuno3=yuno2;
        end

        temp2=[yuno1 yuno3];
        temp1=[temp1;temp2];
    end

    fuh=(numel(temp1))/2;
    lump_post=[];
    lump_post1=[];


    for cov=1:repetation
        distance_fin11=[];
        unc_fin1=[];
        jok=[];
    for cok=1:2
    temp_ps1x=temp_ps1(cok,:,cov);
    temp_ps4=[temp_ps1x;temp1];

    %calculating cartesian metric distance of random points and all points
    %within coords_f1 array
    for kao13=1:(atb_hi_fin+1)
    x_min=(temp_ps4(1,1)-temp_ps4(kao13,1))*100;
    y_min=(temp_ps4(1,2)-temp_ps4(kao13,2))*100;
    x_2=x_min^2;
    y_2=y_min^2;
    res=(x_2+y_2)^(0.5);
    distance_fin11=[distance_fin11;res];
    end
    adk=distance_fin11(2:fuh);
    [duf,duf1]=sort(adk,'ascend');

    %check if there is any similar value
    knkt=0;
    knkt1=1;
    while knkt==0
        ah=duf1(knkt1);
        ah1=coord_f1(ah);
        kntb=ismember(ah1,jok);

        if kntb==1
        knkt1=knkt1+1;
        else
        duf4=ah1;
        knkt=1;
        end
    end

    jok=[jok;duf4];
    fin_pos1=jok;
    end
```

```
196
197     %below code is meant for calculating the uncertainty of 2 pyranometers
198     %configurations
199     for kaop=1:atb1
200         apb1=zoom_fin_data(:,kaop);
201         apb2=nonzeros(apb1);
202         apb3=numel(apb2);
203         apb4=mean(apb2);
204         apb5=alikl6_fin(kaop,:);
205         apb6=mean(apb5);
206
207         % uniform and near-unfirom sky condition
208         if apb3≠0 && apb4==1
209         temporary=0;
210         unc_fin1=[unc_fin1 temporary];
211         end
212
213         % non-uniform sky condition
214         if apb3≠0 && apb4≠1
215         temporary=nocskyun2(fin_pos1,apb1,nopyra,apb4,ct,apb6);
216         unc_fin1=[unc_fin1 temporary];
217         end
218      end
219
220      unc_v1=temporal_unc(unc_fin1,confd);
221
222     %final positions of two pyranometer on 10 trial of configurations
223     lump_post=[lump_post jok];
224     %final uncertainty value of two pyranometer on 10 trial of configurations
225     lump_post1=[lump_post1;unc_v1];
226     end
227     end
228
229     lump_post_temp=[];
230
231     %Below is greedy searching algorithm to estimate the right
232     %configuration of pyranometer when 3 or more pyranometers are used.
233     %Previously obtained configurations of 2 pyranometers are used as
234     %starter positions.
235
236
237     % looping for a number of different pyranometer configurations; the
238     % amount of pyranometer is still constant
239     for ghy=1:repetation
240
241         if nopyra==2
242         % when number of pyranometer equals to 2, assign similar
243         % pyranometers configurations as recommended on previous step
244         temp_john=lump_post(:,ghy);
245         jkt15=[];
246         else
247         % otherwise seek new pyranometers configurations with the help of
248         % greedy algorithm and obtained starting configurations for 2
249         % pyranometers
250         temp_john2=fin_pos_rep((1:atb100),atb101,ghy);
251         temp_john3=numel(temp_john2);
252         temp_john4=numel(coord_f1);
253         temp_john5=temp_john4-temp_john3;
254         temp_john6=coord_f1;
255
256         for kop=1:temp_john3
257             duk=temp_john2(kop);
258             temp_john6(temp_john6==duk)=0;
259         end
260         ik2=find(temp_john6≠0);
261         pongky=[];
262
263         for kush=1:temp_john5
264         tempy0=ik2(kush);
265         tempy=temp_john6(tempy0);
266         fin_pos=[temp_john2;tempy];
```

```
        unc_fin=[];

        % following code estimates the change in uncertainty value when one
        % more pyranometer cell is added into the configuration

        % every data point within coords_f1 array will be checked one by
        % one to be able to determine the right cell for the most optimum
        % pyranometer configuration
        parfor kaop=1:atb1
        apb1=zoom_fin_data(:,kaop);
        apb2=nonzeros(apb1);
        apb3=numel(apb2);
        apb4=mean(apb2);
        apb5=alikl6_fin(kaop,:);
        apb6=mean(apb5);


        % uniform and near-uniform sky condition
        if apb3≠0 && apb4==1
        temporary=0;

        %uniform and near-uniform sky condition generate zero value which
        %means no deviation between measured and actual irradiance
        unc_fin=[unc_fin temporary];
        end

        % non-uniform sky condition
        if apb3≠0 && apb4≠1
        temporary=nocskyun2(fin_pos,apb1,nopyra,apb4,ct,apb6);

        %following variable is an array of deviations that occur on every
        %temporal dataset, further this array will be used to determine the
        %uncertainty
        unc_fin=[unc_fin temporary];
        end
        end

        %calling a function to estimate the measurement uncertainty of proposed
        %pyranometers configurations in one year data
        unc_v=temporal_unc(unc_fin,confd);

        pongky=[pongky;unc_v];
        end

        %following code estimates which cell is the most appropriate cell
        %for being the next measurement point; complete explanation about
        %this selection procedure is presented on the report
        jkt6=lump_post1(ghy,atb101);
        pongky1=pongky-jkt6;
        jkt10=numel(pongky1);
        [jkt11,jkt21]=sort(pongky1,'ascend');
        jkt7=find(jkt11≥0);
        jkt9=numel(jkt7);
        jkt12=jkt10-jkt9;

        if jkt12==0
        jkt2=jkt21(1);
        else
        jkt2=jkt21(jkt12);
        end

        jkt15=pongky(jkt2);
        jkt3=ik2(jkt2);
        jkt4=temp_john6(jkt3);
        temp_john=[temp_john2;jkt4];

        end

    lump_post_temp=[lump_post_temp;jkt15];

    %following array contains recommended positions of pyranometers on
```

```
338     %several possible configurations
339     fin_pos_rep((1:nopyra),atb100,ghy)=temp_john;
340 
341     stepu=ghy+(nopyra-2)*repetation+(ct-1)*(max_clust-1)*repetation;
342     waitbar(stepu/stepus)
343     end
344 lump_post1=[lump_post1 lump_post_temp];
345 end
346 
347 %both of following array are only a group of data array for different type
348 %of uncertainty analysis; one uses absolute uncertainty in W/m2 and the
349 %other uses relative uncertainty
350 final_pos_rep(:,:,:,ct)=fin_pos_rep;
351 
352 final_lump_post1(:,:,ct)=lump_post1;
353 
354 if ct==1
355     fin_pos_rep=zeros(max_clust,atb1000,repetation); %reset the array
356 end
357 
358 end
359 close(hintul)
360 
361 
362 clearvars hintul stepu stepus duk tempy tempy0 pongky jkt1 jkt2 jkt3 jkt4 ik1 ik1 ...
        temp_john temp_john1 temp_john2 temp_john3 temp_john4 temp_john5 temp_john6
363 clearvars kao2 M_2 I_2 kao3 M_5 I_5 dist_stand2 koy_5 koy_21 koy_14 koy_15 koy_17 ...
        koy_16 koy_22 coord_f1 koy_3 koy_9 koy_8 koy_11 koy_12 koy_13 temp_koy M_8 I_8 ...
        dist_stand1 koy koy_1 koy_4 fus filter_1 kao1 I2 yuno1 yuno2 temp2 atb_hi ...
        atb_hi_fin  prob_stand atb502 atb503 atb_x atb_z atb_w atb_xw distance_fin1 ...
        atb501 temp1 atb100 atb200 atb_y B1 I1 norm_b2
364 clearvars apb5 ct apb6 apb7 kaop apb1 apb2 apb3 apb4 temporary I3 x_min y_min x_2 y_2 ...
        atb1000 atb2000 nopyra temp1 atb100 atb200 B2 I1 I2 kao yuno1 yuno2 temp2 kao3 ...
        kao2 res atb501 atb 502 atb503 atb1 atb2 atb3 atb4 atb5 atb6 atb7 atb temp_prob ...
        temp_prob1 nopyr kao
```

A.24. MATLAB SCRIPT PROB_HIST2D_1.M

```
1  % Function to estimate sum of probability of each cell for being a centroid
2  % throughout a year
3  
4  %the probability of each cell to become centroid is estimated in every
5  %temporal data set and then it is summed up cumulatively
6  function cell_hist=prob_hist2d_1(zoom_temp2,NO_PYRANOMETER1,dies2)
7  wop=NO_PYRANOMETER1;
8  zoom_temp3=nonzeros(zoom_temp2);
9  
10 % Kmeans process is repeated 30 times to obtain the best result of
11 % clustering within the data array
12 [idx,C] = kmeans(zoom_temp3,NO_PYRANOMETER1,'Replicates',30);
13 cao=numel(zoom_temp2);
14 cao1=numel(zoom_temp3);
15 cell_hist=zeros(cao,1);
16 acuk=find(dies2≠0);
17 
18 
19 for at6=1:wop
20     cell_hist2=zeros(cao,1);
21     cell_hist1=zeros(cao1,1);
22 
23     %following code makes sure that the cell that has been appointed as
24     %centroid belongs to the same cluster not from another cluster
25     temp_idx=idx;
26     temp_idx(temp_idx≠at6)=0;
27     temp_idx(temp_idx==at6)=1;
28     temp_mem=temp_idx.*zoom_temp3;
29     temp_mem1=nonzeros(temp_mem);
30 
```

```
    at7=abs(temp_mem1-(C(at6)));
    [at21,at8]=min(at7);
    at20=temp_mem1(at8);
    at9=find(temp_mem==at20);
    at10=numel(at9);
    cell_hist1(at9)=1/at10;

    %resize the probability value data array to match 'dies' data array
    for komt=1:cao1
        acuk1=acuk(komt);
        cell_hist2(acuk1)=cell_hist1(komt);
    end

    %final sum of probability of active cells
    cell_hist=cell_hist+cell_hist2;
end

clearvars at20 at21 at13 at14 zoom_temp2 NO_PYRANOMETER1 idx sumd cao C_temp po ...
    num_cent at0 at1 at2 at3 at4 at5 at6 at7 at8 at9 at10
```

A.25. MATLAB SCRIPT GEO_POINTS_STARTER.M

```
% Script to generate random configuration of 2 pyranometers positions.
% The only constraint is euclidean metric distance between these 2
% pyranometer.

clc


max_clust_star=2;
hul=max_clust_star-1;
dia_fin=[];
post_candid=zeros(max_clust_star,repetation,hul);
coeff_dist=zeros(repetation,hul);

% following code calculates minimum distance between 2 pyranometers
% detail explanation about how the distance is calculated is presented on
% the report
for clust=2:max_clust_star %no loop since max_clust_star equals to 2
apolo=nonzeros(dies1);
apolo1=numel(apolo);
area_tot=apolo1*10000;
circ_area=area_tot/clust;
dia_avg=2*(sqrt(circ_area/pi));
dia_fin=[dia_fin;dia_avg];
end

zoom_dat=find(dies1==1);

%following loop is only to randomly  generate different configuration of two
%pyranometers positions with minimum distance as stated on dia_fin array
for hull=1:hul
    dia_temp=dia_fin(hull);
    clust1=hull+1;
for rep=1:repetation
    [pos_temp,coeff]=rand_select2(zoom_dat,dia_temp,clust1,range_furt);
    post_candid(1:clust1,rep,hull)=pos_temp;
    coeff_dist(rep,hull)=coeff;
end
end

post_fin=zeros(max_clust_star,hul,repetation);

for pub=1:repetation
    atk=post_candid(:,pub,:);
    atk1=reshape(atk,[max_clust_star hul]);

    %below is the generated data array that contains several different
```

```
    %configurations of two pyranometers
    post_fin(:,:,pub)=atk1;
end



clearvars max_clust_star post_candid pub atk atk1 clust apolo apolo1 area_tot ...
    circ_area dia_avg zoom_dat rep dia_temp clust1 pos_temp
```

A.26. MATLAB SCRIPT RAND_SELECT2.M

```
% Function to pick out two random cells from solar park data array
% The random selection is only limited by minimal distance threshold

function [pos_temp_arr,k4]=rand_select2(zoom_fin_dat,dia_fin,clust,range_furt)
balik=numel(zoom_fin_dat);

% Computing metric distance between cells
distance_fin1=zeros(balik);
atb501=size(range_furt);
atb502=atb501(1);
temp1=[];
for kao1=1:balik
        I2=zoom_fin_dat(kao1);
        yuno1=ceil(I2/atb502);
        yuno2=rem(I2,atb502);

        if yuno2==0
            yuno3=atb502;
        else
            yuno3=yuno2;
        end

        temp2=[yuno1 yuno3];
        temp1=[temp1;temp2];
end

for kao3=1:balik
    for kao2=1:balik
    x_min=(temp1(kao3,1)-temp1(kao2,1))*100;
    y_min=(temp1(kao3,2)-temp1(kao2,2))*100;
    x_2=x_min^2;
    y_2=y_min^2;
    res=(x_2+y_2)^(0.5);
    distance_fin1(kao2,kao3)=res;
    end
end


%Starting point of random points picking
k1=0;
k2=0;
k3=1;
while k1==0

if k2>10 % Trials for every k3
   k3=k3-0.005; % Decrement of k3 everytime threshold metric distance value
                % could not be met
end

dia_fin1=dia_fin*k3;

% Filtering random pairs that does not fulfill distance threshold
% requirement
zoom_fin_dat1=zoom_fin_dat;
tempo=[];
[samp_dat,post]=datasample(zoom_fin_dat1,1);
black_list=distance_fin1(:,post);
```

```
black_list(black_list<dia_fin1)=0;
black_list_sum=black_list;
black_list(black_list≠0)=1;
tempo=[tempo;samp_dat];
k0=1;

while k0<clust
black_list1=black_list_sum;
black_list1(black_list1==0)=inf;
[G1,H1]=min(black_list1);

if G1==inf
    break
end

black_list_temp=distance_fin1(:,H1);
black_list_temp(black_list_temp<dia_fin1)=0;
black_list_temp1=black_list_temp;
black_list_temp1(black_list_temp1≠0)=1;
black_list=black_list.*black_list_temp1;
black_list_sum=(black_list_sum+black_list_temp).*black_list;
temp_coord=zoom_fin_dat(H1);

% The final array that holds random pairs positions
tempo=[tempo;temp_coord];

k0=k0+1;
end

boyu=numel(tempo);

if boyu==clust
    pos_temp_arr=tempo;
    k1=1;
    k4=k3;
else
    k2=k2+1;
end
end
```

A.27. MATLAB SCRIPT NOCSKYUN2.M

```
%this function calculates deviation between the average of irradiance
%values measured on certain active cells and the true average value of all
%active cells

%the only difference between this function and nocskyun function is, this
%model is only able to process the input with a single number of pyranometer
%instead of a multiple number of pyranometer like nocskyun does

function temporary=nocskyun2(fin_pos,zoom_fin_dat,clust,true_mean,ct,mean_cs)

    tempo=zeros(clust,1);
    true_mean_irr=true_mean*mean_cs;
    zoom_fin_dat_irr=zoom_fin_dat*mean_cs;

    %following loop retrieves irradiance values on the cells that have been
    %assigned as measurement points
    for ty=1:clust
       aku=fin_pos(ty,1);
       aku1=zoom_fin_dat_irr(aku);
       tempo(ty)=aku1;
    end

    %deviation calculation based on absolute irradiance value, it could be
    %CSI(dimensionless) or irradiance (W/m2)
    if ct==1
    mean_centr=mean(tempo);
```

```
    uncen=mean_centr-true_mean_irr;
    temporary=uncen;
    else
    %deviation calculation based on relative irradiance value
    mean_centr=mean(tempo);
    uncen=mean_centr-true_mean_irr;
    temporary=(uncen/mean_centr);
    end

end
```

A.28. MATLAB SCRIPT TEMPORAL_UNC.M

```
%This function estimates measurement uncertainty from a certain pyranometer
%positions configurations in one year period

%since the histogram from a year data deviation does not resemble normal
%distribution, a common normal distribution shape factor cannot be used

%the created empirical cumulative density function is tracked every each
%interval until a certain confidence value is achieved

function unc_v=temporal_unc(unc_fin,confd)

at3=size(unc_fin);
at4=at3(1);

unc_v=zeros(at4,1);

for x1=1:at4

loko=unc_fin(x1,:);

%calling built-in function in matlab to construct empirical cumulative
%density function of histogram created from a year deviation data array
%obtained from previous process
[f,x]=ecdf(loko);

I=find(x==0);


f_up=1;
f_lo=1;

k3=0;
n1=1;
n2=1;

%following loop locks tracking process until a certain confidence interval
%has been reached; default setting is confd=95%
while k3<confd

    c=I+n1*f_up;
    d=I-n2*f_lo;


    a=f(c);
    b=f(d);


    % Limiters for Empirical Cumulative Density Function (ECDF)
    if a==1
        cons_up=1;
        cons_do=b;
        n1=0;
        n2=n2+1;
    else if b==0
         cons_up=a;
```

```
            cons_do=0;
            n1=n1+1;
            n2=0;
          else
            cons_up=a;
            cons_do=b;
            n1=n1+1;
            n2=n2+1;
          end
      end


      k3=cons_up-cons_do;
  end

  temp=zeros(2,1);
  temp(1)=abs(x(c));
  temp(2)=abs(x(d));

  afg=max(temp);

  % following array contains final uncertainty values of the model; not final
  % result
  unc_v(x1)=afg;


  end
```

A.29. MATLAB SCRIPT FINAL_UNCERTAIN.M

```
%script to build final uncertainty array of all interpolated data array
%(unmodified and modified data arrays)
clc

atb1000=max_clust-1;
final_arr_uncert=zeros(atb1000,repetation,(no_random+1),2);
final_avg_uncert=zeros(atb1000,(no_random+1),2);

%following code estimates measurement uncertainty of different interpolated
%data arrays (unmodified and modified) based on fixed position of pyranometers
%over different randomized configurations which was obtained from
%clustering and combination searching processes that had been done
%previously

hintul1=waitbar(0,'Please wait, data is being processed...');
stepus1=repetation*(no_random+1)*2;
for ct=1:2
fin_pos_rep=final_pos_rep(:,:,:,ct);
efe=size(fin_pos_rep);
efe1=efe(3);
fin_temp_unc=zeros(atb1000,efe1,(no_random+1));
temp_avg_unc=[];

for hug1=1:(no_random+1)
zoom_fin_data=zoom_fin_arr(:,:,hug1);

atb=size(zoom_fin_data);
atb1=atb(2);
fin_unc_irr=zeros(atb1000,efe1);

for rep=1:efe1
fin_pos=fin_pos_rep(:,:,rep);
unc_fin_irr=[];
parfor kaop=1:atb1
apb1=zoom_fin_data(:,kaop);
apb2=nonzeros(apb1);
apb3=numel(apb2);
apb4=mean(apb2);
```

```
apb5=alikl6_fin(kaop,:);
apb6=mean(apb5);
apb7=apb2*apb6;

% uniform and near-uniform sky condition
if apb3≠0 && apb4==1
  temporary=zeros(atb1000,1);
  unc_fin_irr=[unc_fin_irr temporary];
end

% non-uniform sky condition
if apb3≠0 && apb4≠1
    [temporary,temporary1]=nocskyun(fin_pos,apb1,max_clust,apb4,ct,apb6);
    unc_fin_irr=[unc_fin_irr temporary1];
end

end

%calling a function to estimate the measurement uncertainty of proposed
%pyranometers configurations in one year data
unc_v_irr=temporal_unc(unc_fin_irr,confd);

fin_unc_irr(:,rep)=unc_v_irr;

stepu1=rep+(hug1-1)*efe1+(ct-1)*(no_random+1)*efe1;
waitbar(stepu1/stepus1)
end
avg_fin_unc=mean(fin_unc_irr,2);

temp_avg_unc=[temp_avg_unc avg_fin_unc];
fin_temp_unc(:,:,hug1)=fin_unc_irr;
end

%below is data array of complete final uncertainty values with several different ...
    pyranometer configurations
final_arr_uncert(:,:,:,ct)=fin_temp_unc;

%below is data array of final uncertainty averages over several different pyranometer ...
    configurations
final_avg_uncert(:,:,ct)=temp_avg_unc;
end
close(hintul1)


clearvars hintul1 stepu1 stepus1 confd max_clust repetation apb1 apb2 apb3 apb4 apb5 ...
    apb6 apb7 fin_pos_rep atb kaop rep ct atb atb1000 hug1 zoom_fin_data atb1 atb1000 ...
    efe efe1 fin_unc fin unc_v_irr avg_fin_unc temp_avg_unc fin_temp_unc temporary ...
    temporary1
```

A.30. MATLAB SCRIPT NOCSKYUN.M

```
%this function calculates deviation between the average of irradiance
%values measured on certain active cells and the true average value of all
%active cells

%the only difference between this function and nocskyun2 function is, this
%model is able to process the input with multiple number of pyranometer
%instead of just a single number of pyranometer like nocskyun2 does

function ...
    [temporary,temporary1]=nocskyun(fin_pos,zoom_fin_dat,max_clust,true_mean,ct,mean_cs)
temporary=zeros((max_clust-1),1);
temporary1=zeros((max_clust-1),1);
true_mean_irr=true_mean*mean_cs;
zoom_fin_dat_irr=zoom_fin_dat*mean_cs;


for tg=2:max_clust %loop to repeat the process on different number of pyranometer
```

```
    tyon=tg-1;
    tempo=zeros(tg,1);
    tempo1=zeros(tg,1);

    %following loop retrieves irradiance values on the cells that have been
    %assigned as measurement points
    for ty=1:tg
       aku=fin_pos(ty,tyon);
       aku1=zoom_fin_dat(aku);
       aku2=zoom_fin_dat_irr(aku);
       tempo(ty)=aku1;
       tempo1(ty)=aku2;
    end

    %complete explanation about how the uncertainty is defined is presented
    %on the report

    %deviation calculation based on absolute irradiance value, it could be
    %CSI(dimensionless) or irradiance (W/m2)
    if ct==1
    mean_centr=mean(tempo);
    mean_centr_irr=mean(tempo1);
    uncen=mean_centr-true_mean;
    uncen1=mean_centr_irr-true_mean_irr;
    temporary(tyon)=uncen;
    temporary1(tyon)=uncen1;
    else
    %deviation calculation based on relative irradiance value
    mean_centr=mean(tempo);
    mean_centr_irr=mean(tempo1);
    uncen=mean_centr-true_mean;
    uncen1=mean_centr_irr-true_mean_irr;
    temporary(tyon)=(uncen/mean_centr);
    temporary1(tyon)=(uncen1/mean_centr_irr);
    end

end
```

A.31. MATLAB SCRIPT MAP_POINTS.M

```
%Script to map out measurement positions
clc
clst=3;

aty=size(range_furt);
dies2=reshape(dies1,[aty(1) aty(2)]);
dies3=flipud(dies2);

rel_pos=final_pos_rep(:,:,:,2);
aty1=size(rel_pos);
aty2=aty1(3);
clst1=clst-1;

%converting 1D array coordinate into 2D Cartesian
cum_os=[];
for tr=1:aty2
    te_pos=nonzeros(rel_pos(:,clst1,tr));
    aty3=numel(te_pos);
    mep_os=[];

    for tr1=1:aty3
        te_pos1=te_pos(tr1);
        te_cor=zeros(1,2);
        x_pos=(floor(te_pos1/aty(1)))+1;
        y_pos=aty(1)+1-rem(te_pos1,aty(1));
        te_cor(1)=x_pos;
        te_cor(2)=y_pos;
        mep_os=[mep_os;te_cor];
```

```
    end

    cum_os=[cum_os;mep_os];
end

%Plotting all suggested measurement points under a certain number of
%cluster
c=[1 0 0];
scatter(cum_os(:,1),cum_os(:,2),[],c,'+');
hold on
im = image(dies3,'CDataMapping','scaled');
im.AlphaData = 0.7;

clearvars clst aty aty1 aty2 aty3 dies2 dies3 tr te_pos te_cor mep_os x_pos y_pos ...
    te_pos1 clst1 cum_os
```

B

APPENDIX-B

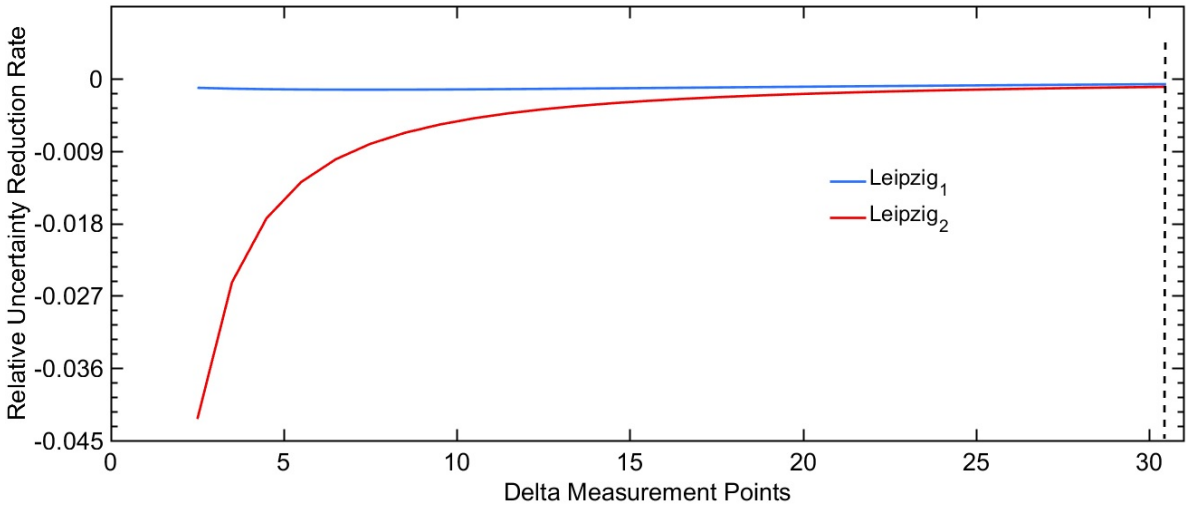


Figure B.1: Uncertainty reduction rate comparison between two different size solar parks on Leipzig

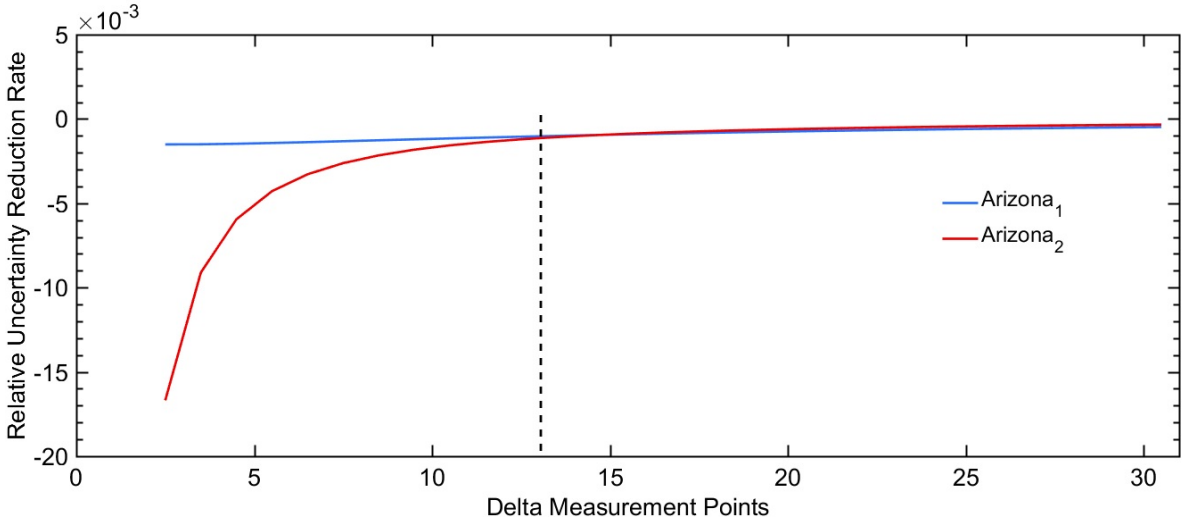


Figure B.2: Uncertainty reduction rate comparison between two different size solar parks on Arizona

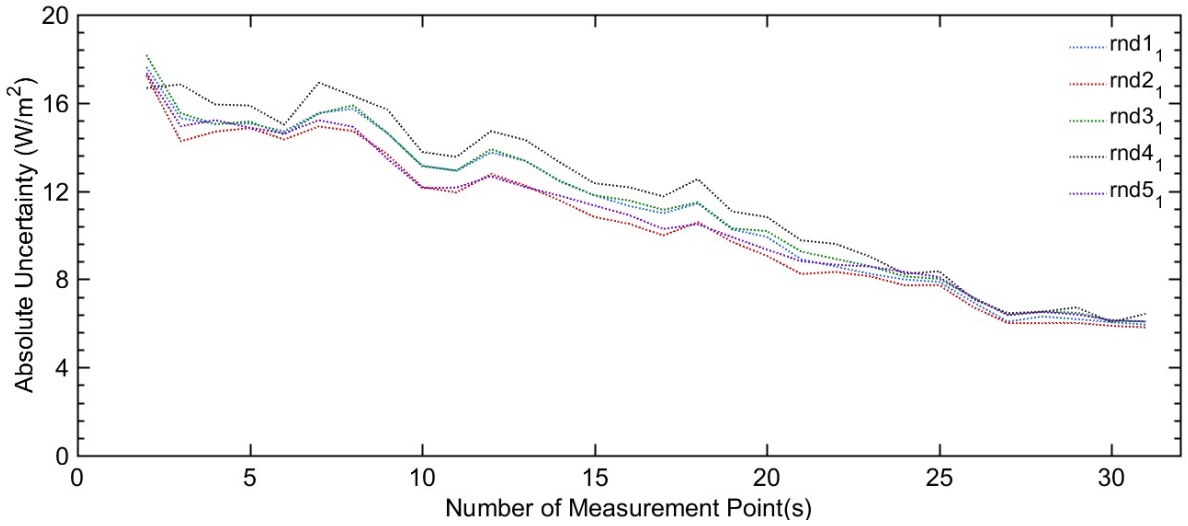


Figure B.3: Absolute Uncertainty of Normal Size Solar Park on Leipzig

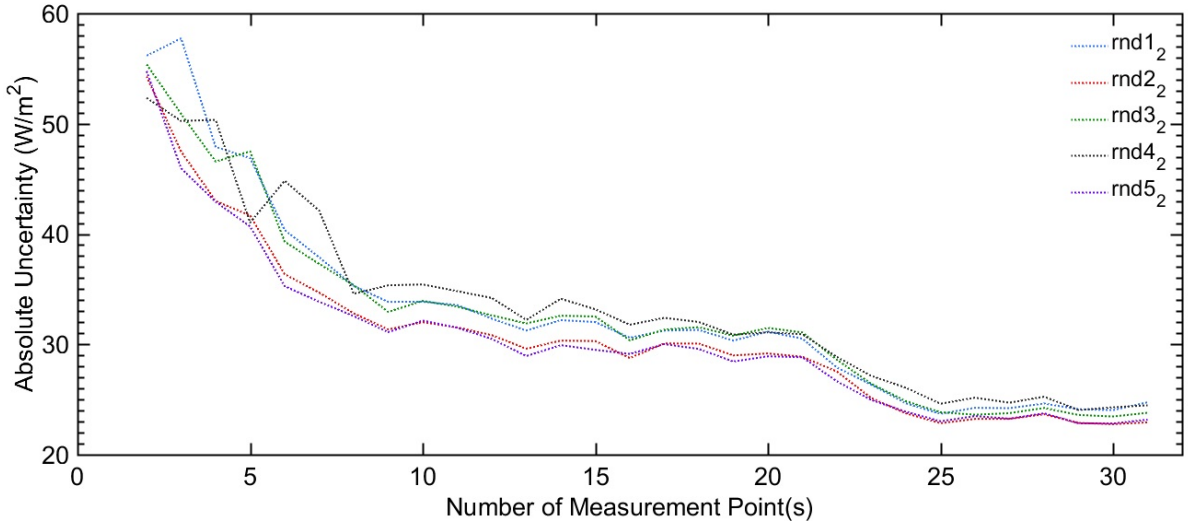


Figure B.4: Absolute Uncertainty of Large Size Solar Park on Leipzig

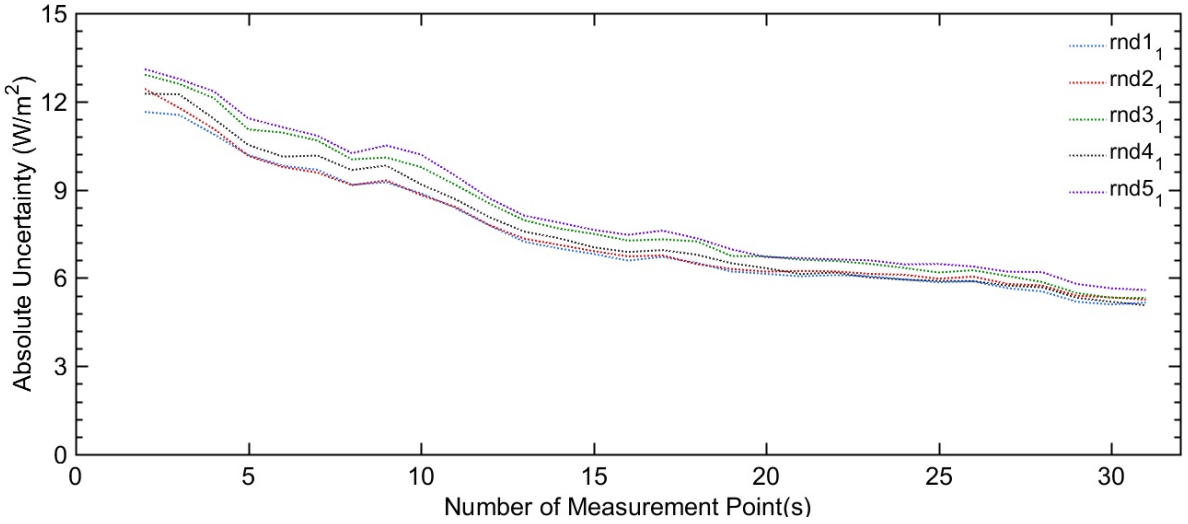


Figure B.5: Absolute Uncertainty of Normal Size Solar Park on Arizona

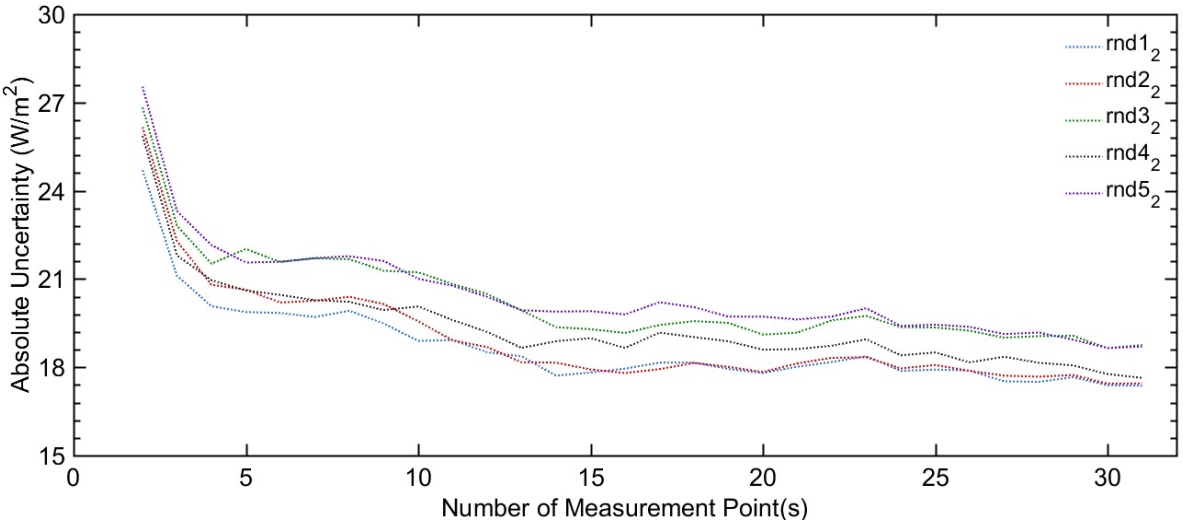


Figure B.6: Absolute Uncertainty of Large Size Solar Park on Arizona

C

APPENDIX-C

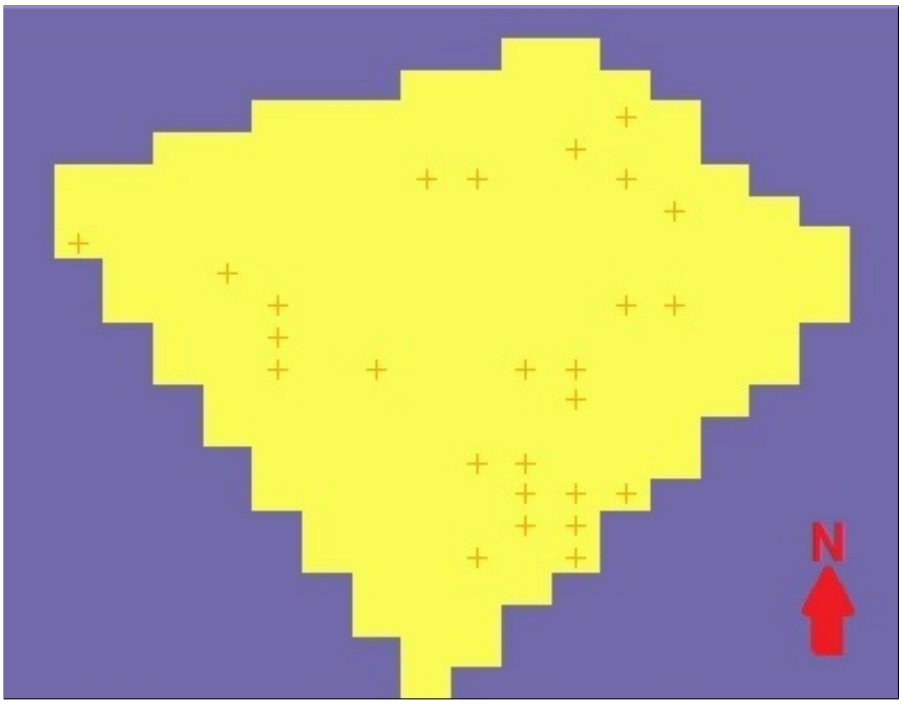


Figure C.1: Measurement points of 10 different configurations on normal size Leipzig solar park; N_c=3

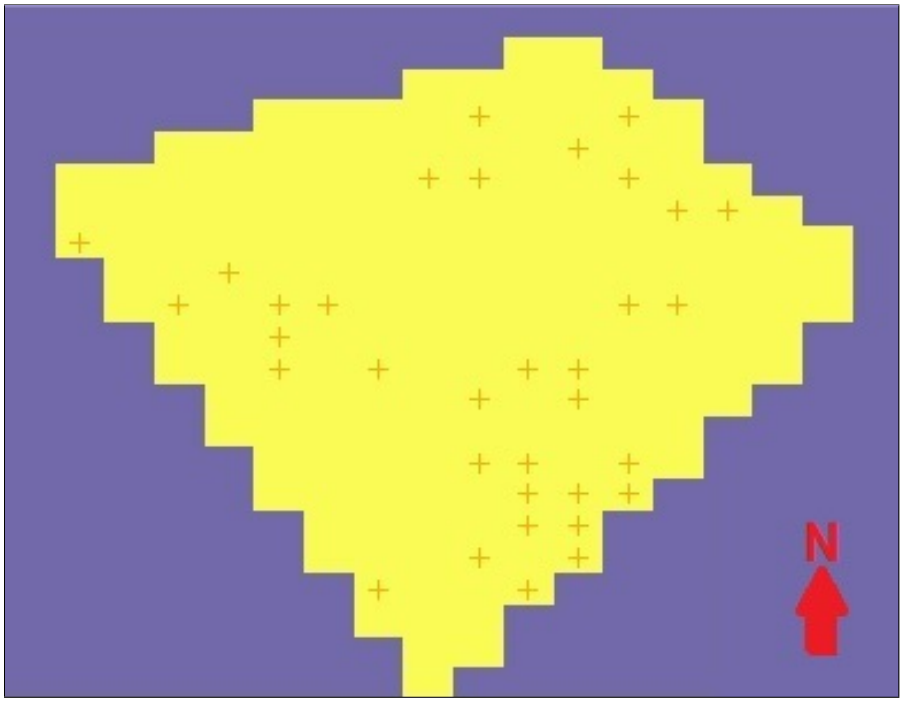


Figure C.2: Measurement points of 10 different configurations on normal size Leipzig solar park; N_c=4

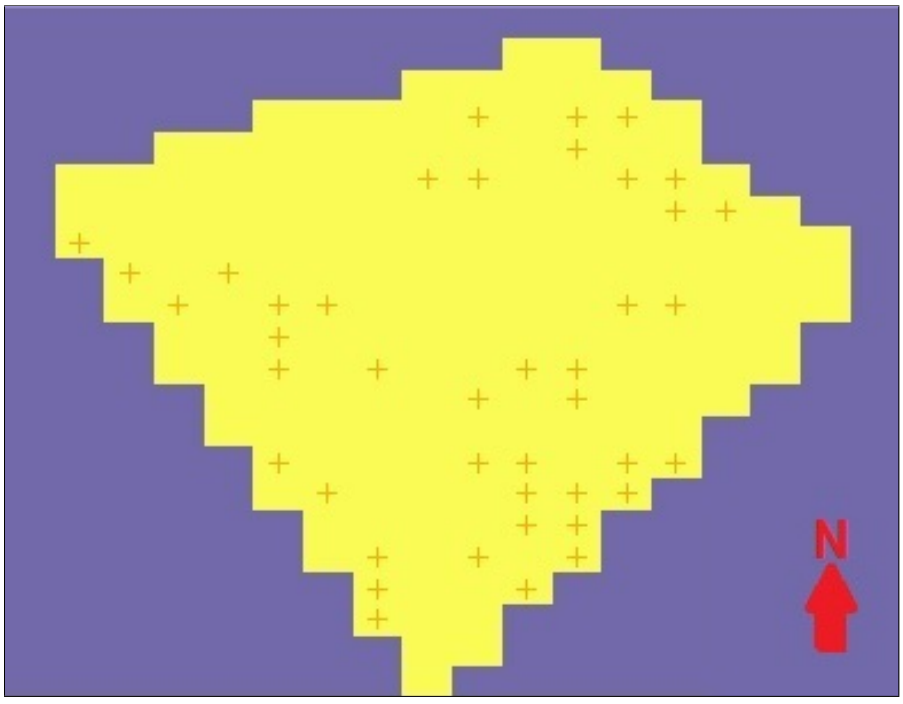


Figure C.3: Measurement points of 10 different configurations on normal size Leipzig solar park; N_c=5

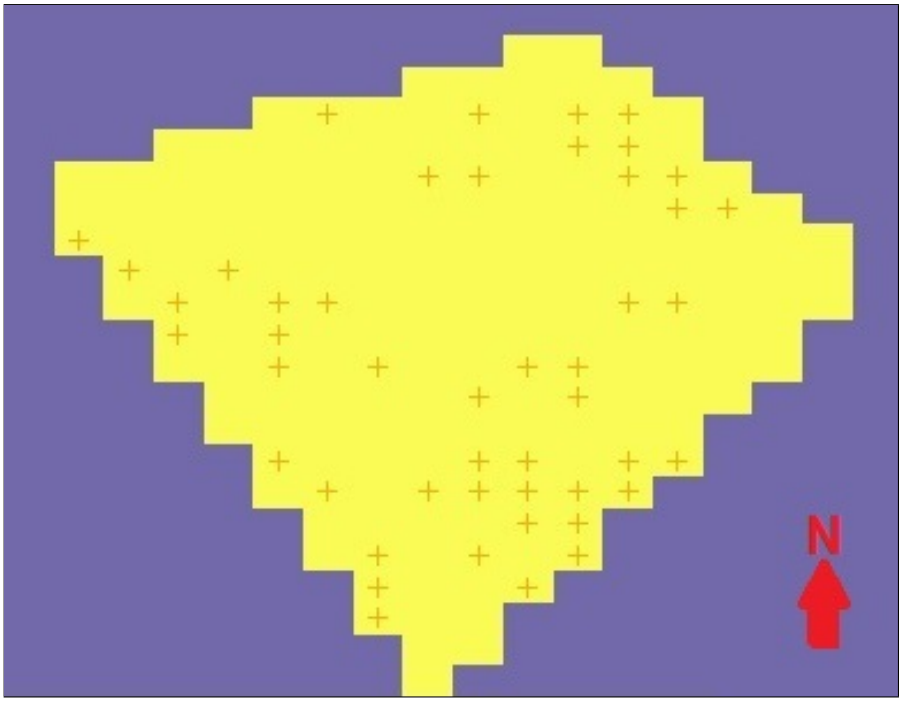


Figure C.4: Measurement points of 10 different configurations on normal size Leipzig solar park; N_c=6

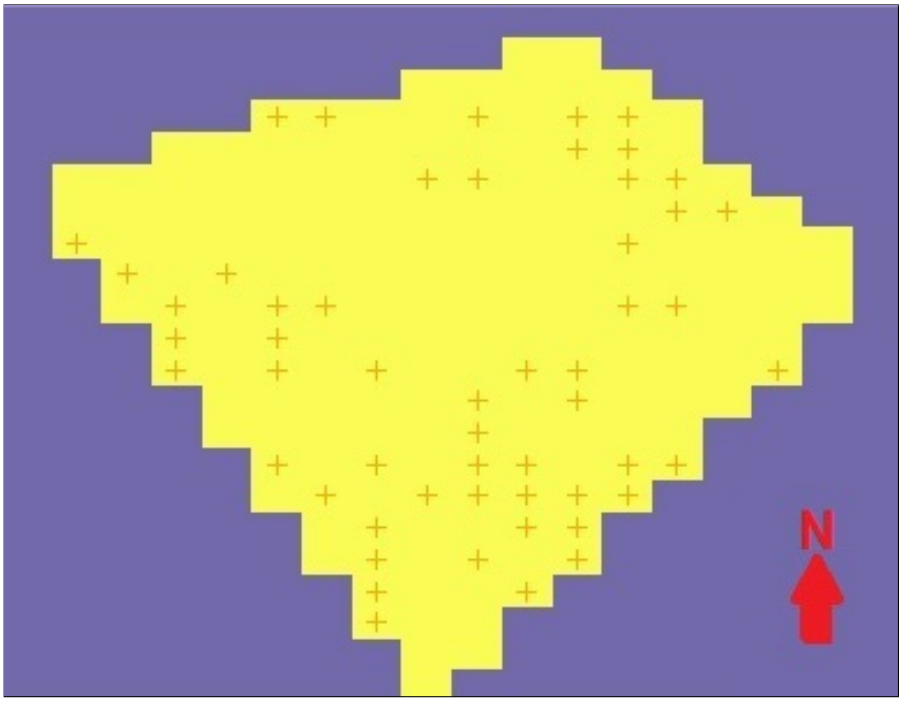


Figure C.5: Measurement points of 10 different configurations on normal size Leipzig solar park; N_c=7

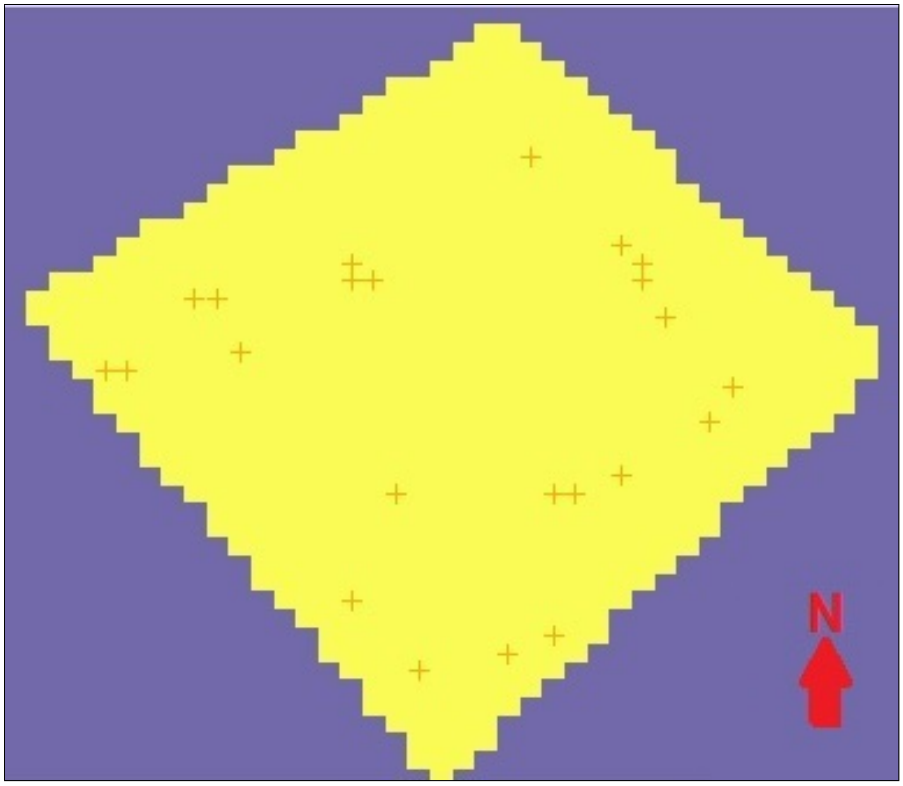


Figure C.6: Measurement points of 10 different configurations on large size Leipzig solar park; N_c=3

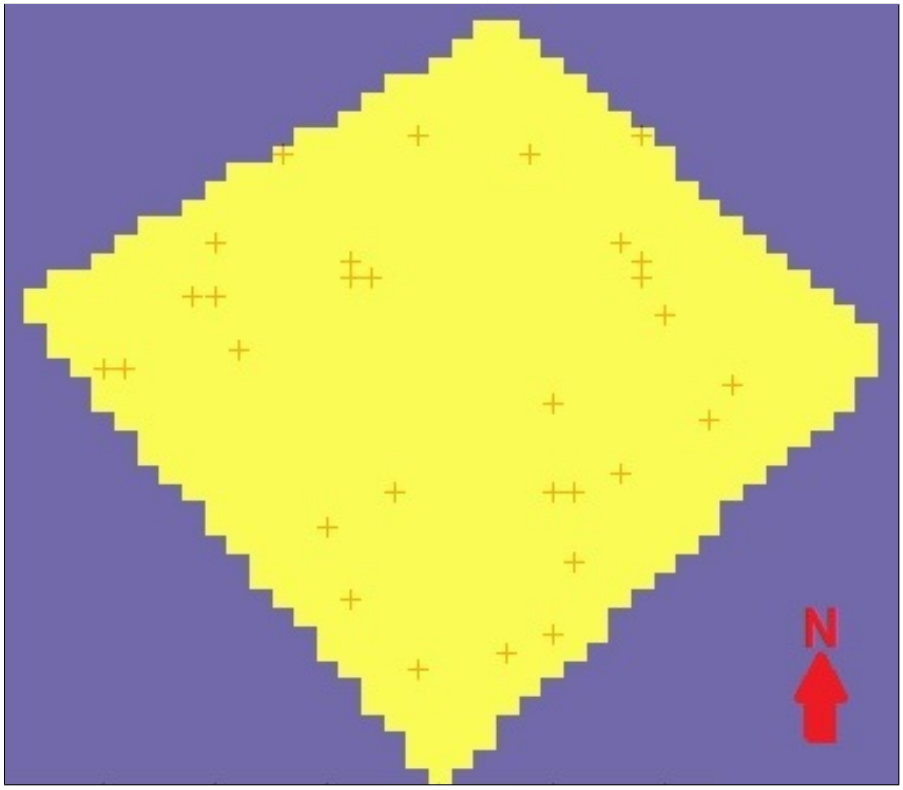


Figure C.7: Measurement points of 10 different configurations on large size Leipzig solar park; N_c=4

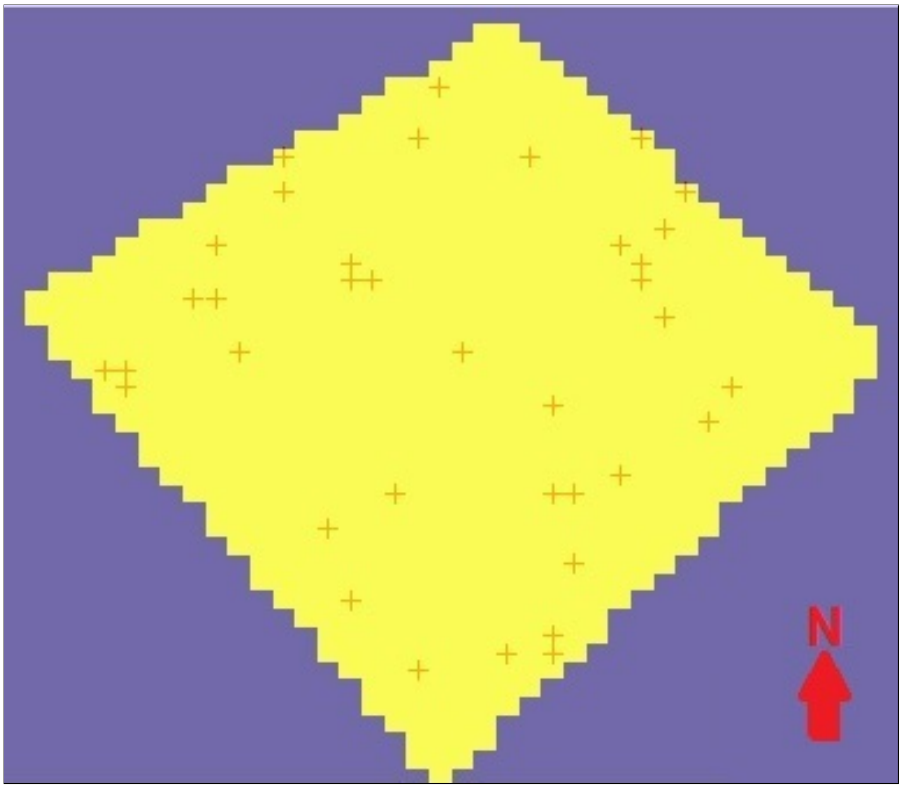


Figure C.8: Measurement points of 10 different configurations on large size Leipzig solar park; N_c=5

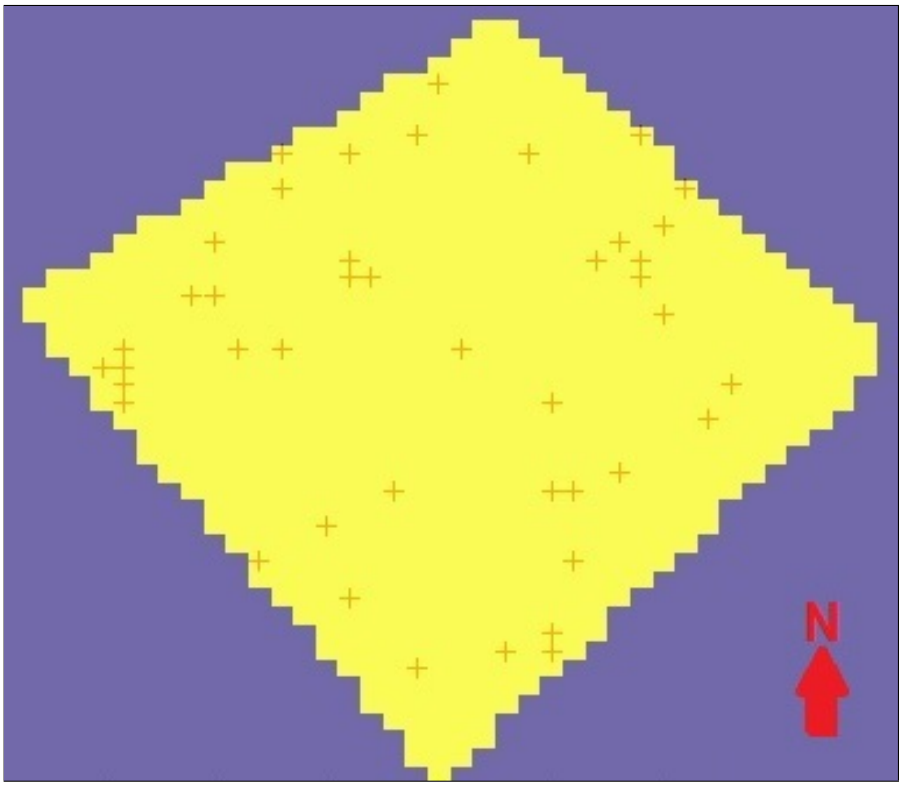


Figure C.9: Measurement points of 10 different configurations on large size Leipzig solar park; N_c=6

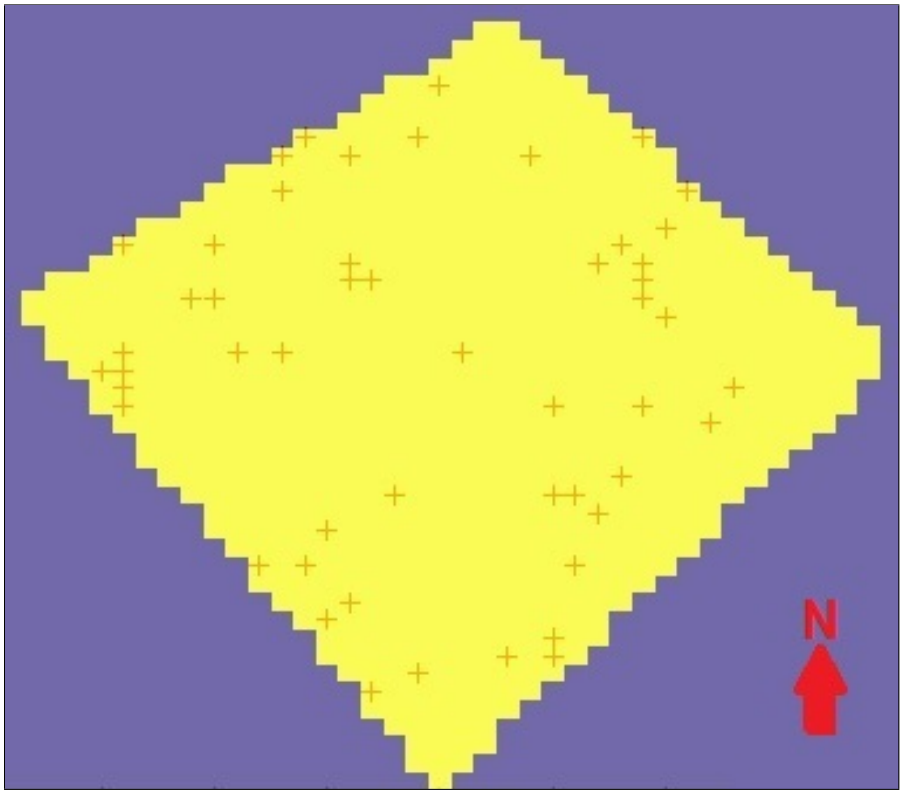


Figure C.10: Measurement points of 10 different configurations on large size Leipzig solar park; N_c=7

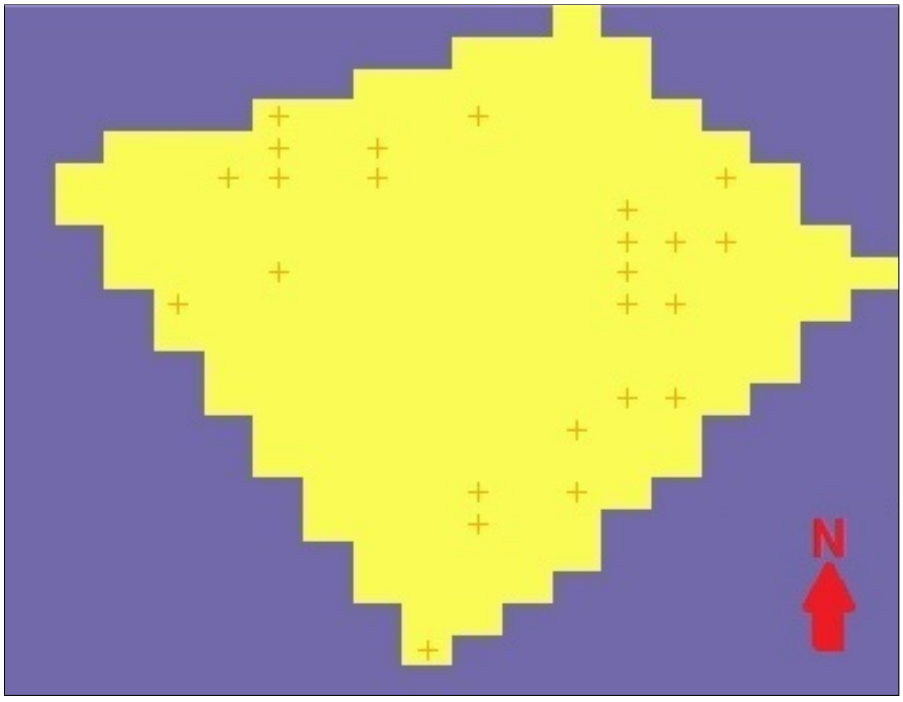


Figure C.11: Measurement points of 10 different configurations on normal size Arizona solar park; N_c=3

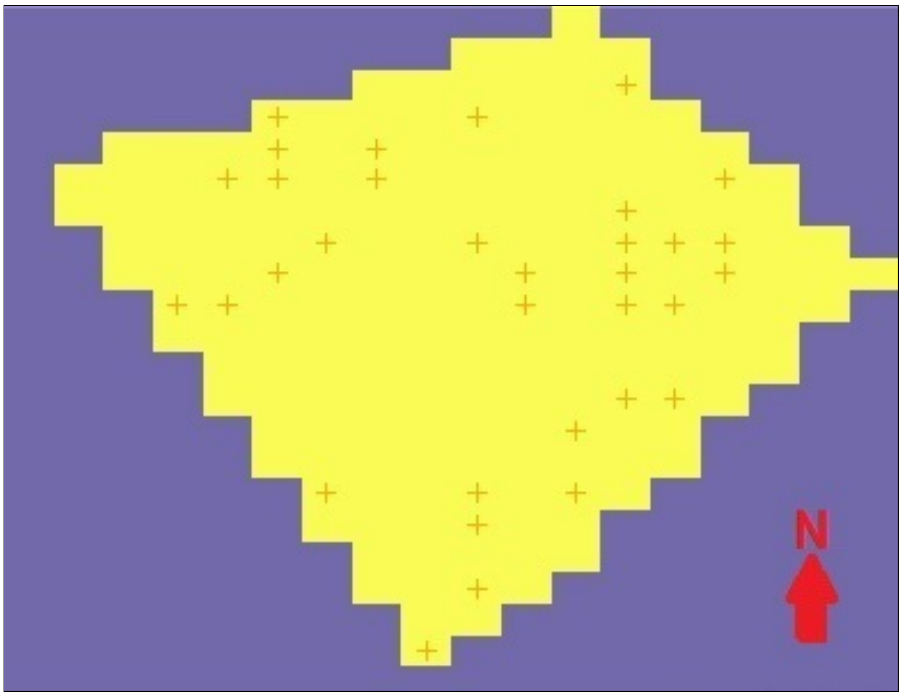


Figure C.12: Measurement points of 10 different configurations on normal size Arizona solar park; N_c=4

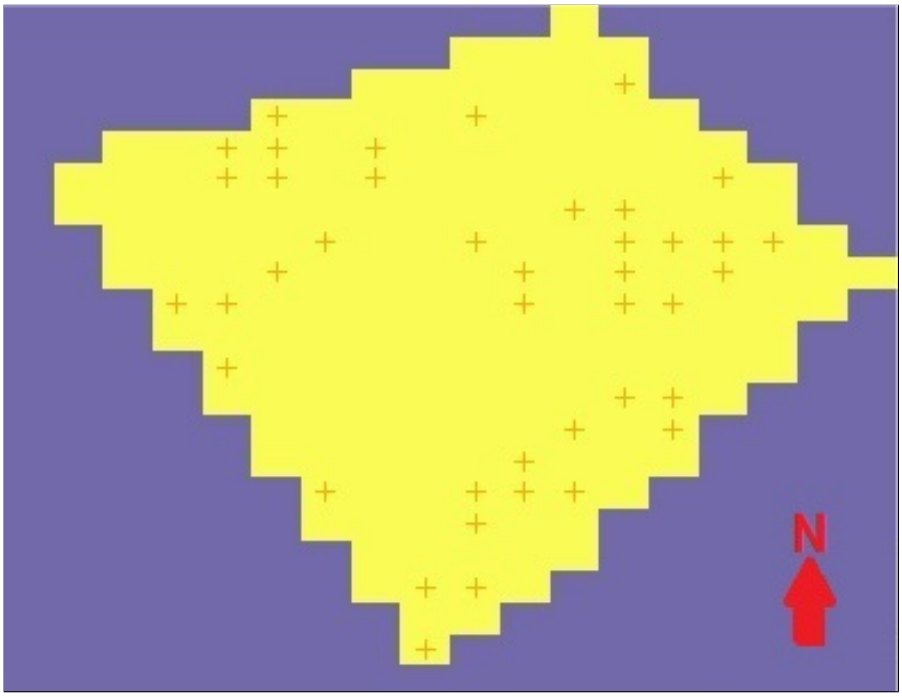


Figure C.13: Measurement points of 10 different configurations on normal size Arizona solar park; N_c=5

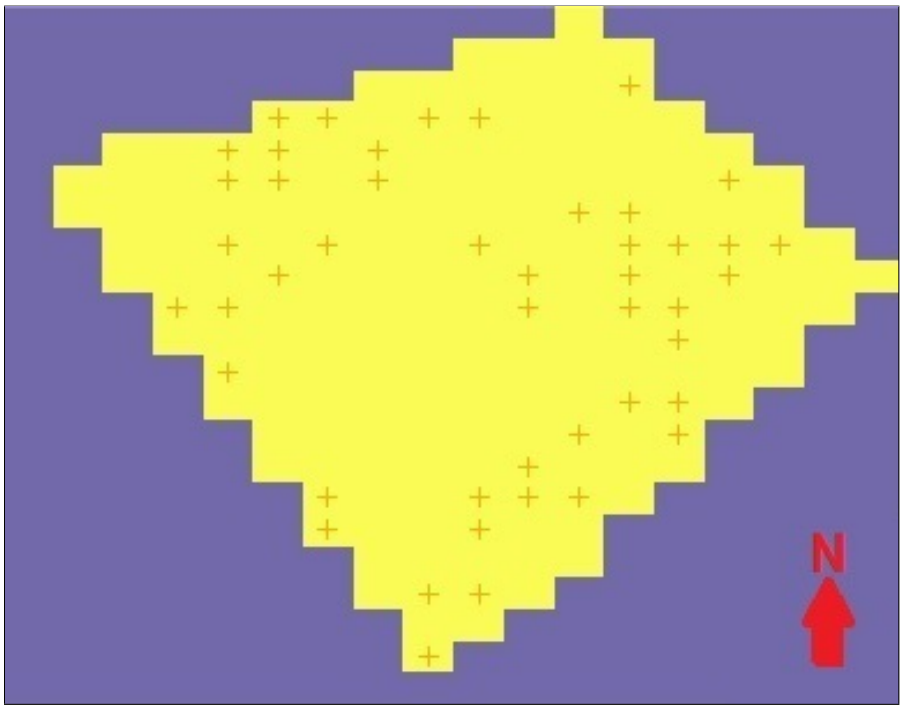


Figure C.14: Measurement points of 10 different configurations on normal size Arizona solar park; N_c=6

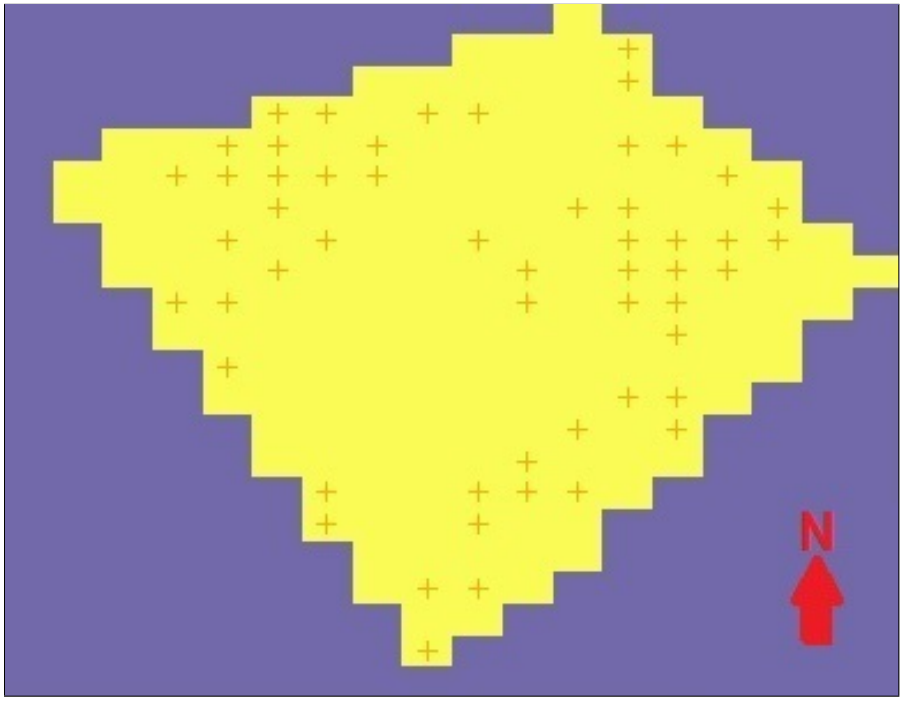


Figure C.15: Measurement points of 10 different configurations on normal size Arizona solar park; N_c=7

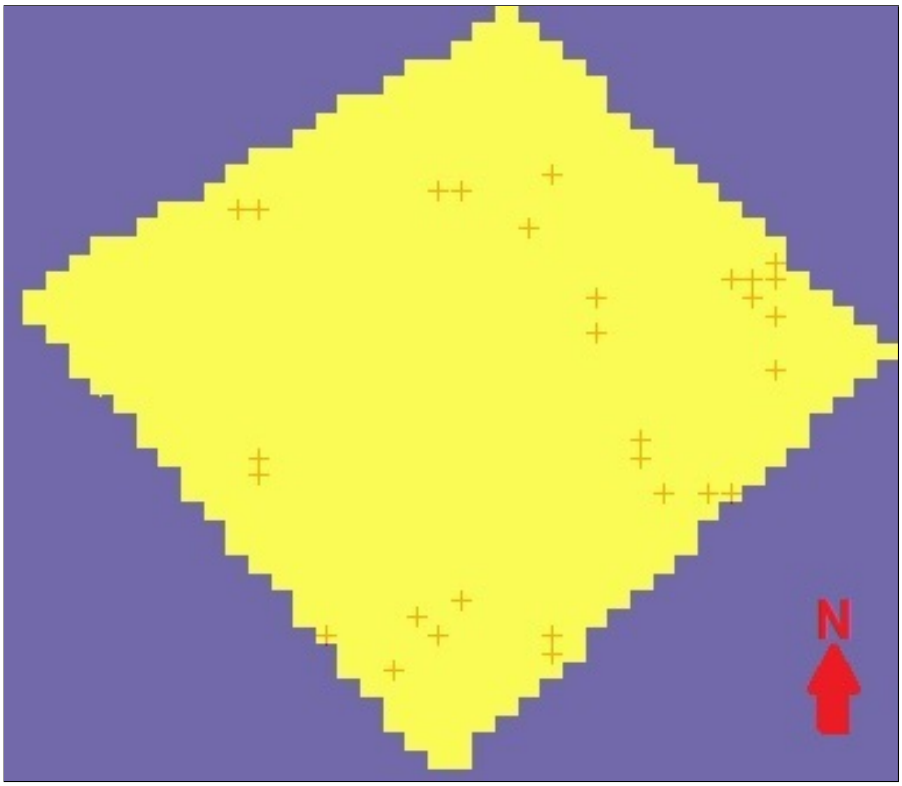


Figure C.16: Measurement points of 10 different configurations on large size Arizona solar park; N_c=3

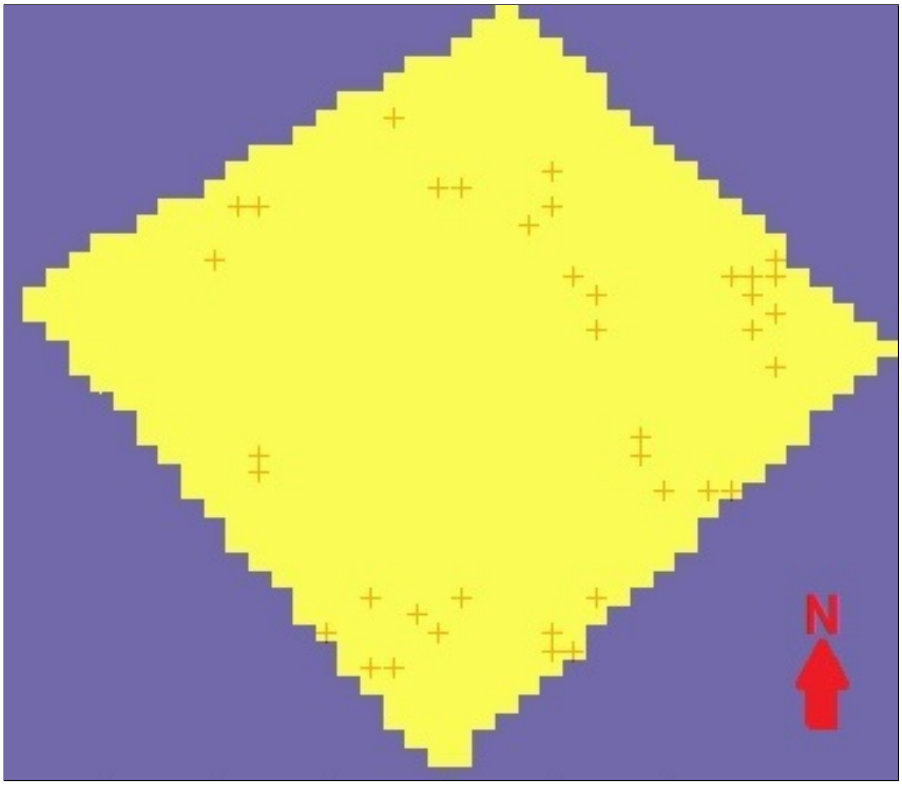


Figure C.17: Measurement points of 10 different configurations on large size Arizona solar park; N_c=4

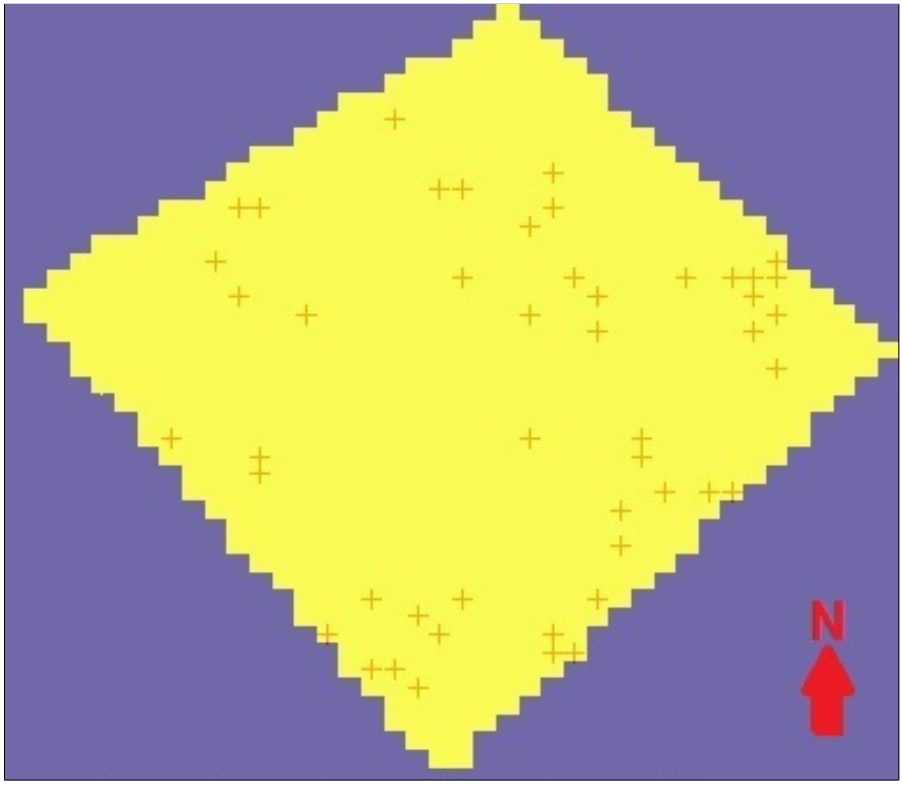


Figure C.18: Measurement points of 10 different configurations on large size Arizona solar park; N_c=5

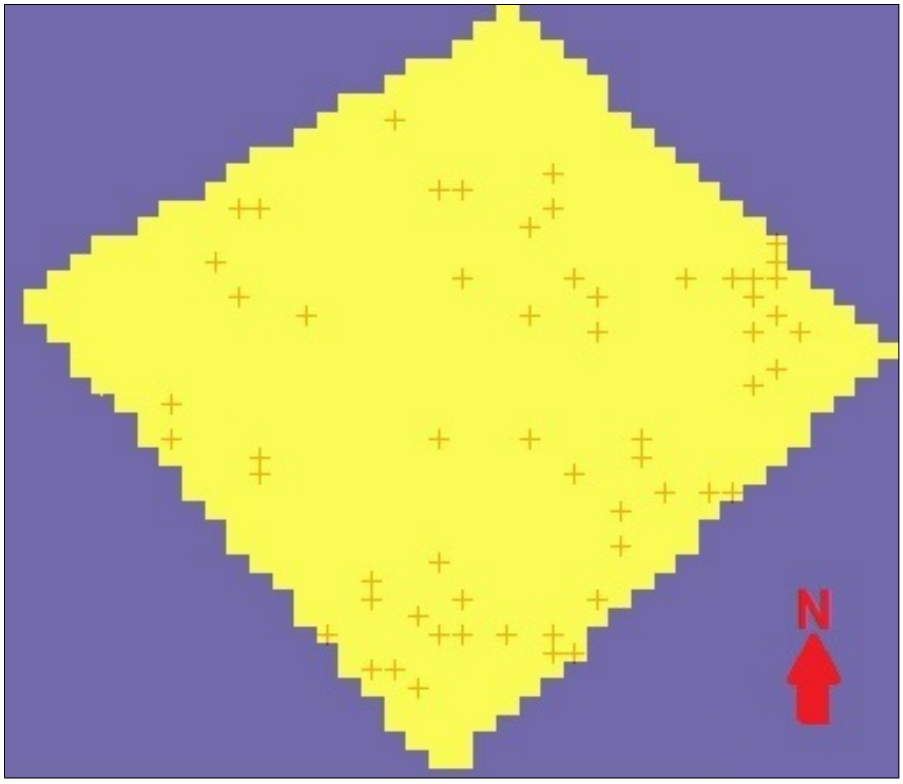


Figure C.19: Measurement points of 10 different configurations on large size Arizona solar park; N_c=6

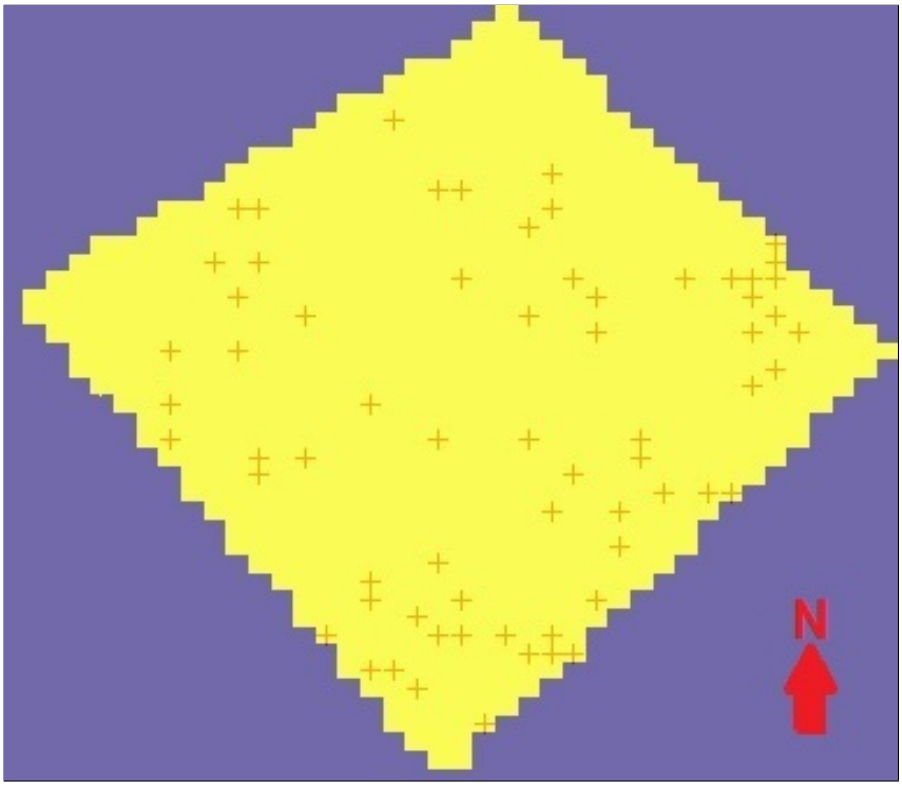


Figure C.20: Measurement points of 10 different configurations on large size Arizona solar park; N_c=7

BIBLIOGRAPHY

[1] Solar Radiation Data(SoDa), *HelioClim , overview,* (2014).

[2] M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational Geometry*, 3rd ed. (Springer, 2008) pp. 191–218.

[3] ArcGIS, *How Natural Neighbor works,* (2011).

[4] G. Bohling, *INTRODUCTION TO GEOSTATISTICS And VARIOGRAM ANALYSIS* (Kansas Geological Survey, Kansas, 2005) pp. 1–20.

[5] P.-N. Tan, M. Steinbach, and V. Kumar, *Cluster Analysis: Basic Concepts and Algorithms,* in *Introduction to Data Mining* (2005) pp. 487–568.

[6] D. Kernler, *A visual representation of the Empirical Rule based on the normal distribution,* (2014).

[7] J. Schmid, *The SEVIRI instrument,* , 1 (2000).

[8] Japan Meteorological Agency, *New geostationary meteorological satellites Himawari 8/9,* (2015).

[9] G. Masson and M. Brunisholz, *2015 Snapshot of global photovoltaic markets*, Tech. Rep. (Photovoltaic Power Systems Programme(PVPS), IEA, 2016).

[10] K. Komoto, H. Ehara, F. Lv, S. Wang, P. Sinha, E. Cunow, A. Wade, D. Faiman, K. Araki, M. J. Perez, K. Megherbi, N. Enebish, C. Breyer, and D. Bogdanov, *Energy from the Desert: Very Large Scale PV Power Plants for Shifting to Renewable Energy Future*, Tech. Rep. IEA PVPS T8-01:2015 (Photovoltaic Power Systems Programme(PVPS), IEA, 2015).

[11] National Centers for Environmental Prediction, National Weather Service, NOAA, U.S. Department of Commerce, *NCEP/NCAR Global Reanalysis Products, 1948-continuing,* (1994).

[12] Renewable Energy Unit, Institute for Energy and Transport (IET), Joint Research Centre, European Commission, *PVGIS Solar Radiation 1981-2011,* (2001-2012).

[13] NREL, *National Solar Radiation Database (NSRDB),* (1998-2017).

[14] MINES ParisTech, *TIME SERIES OF SOLAR RADIATION DATA FROM THE HELIOCLIM-3 DATABASE,* (2005).

[15] DLR, German Aerospace Center, *TIME SERIES OF SOLAR RADIATION DATA FROM THE CAMS RADIATION SERVICE,* (2007-2015).

[16] T. Mieslinger, F. Ament, K. Chhatbar, and R. Meyer, *A New Method for Fusion of Measured and Model-derived Solar Radiation Time-series,* Energy Procedia **48**, 1617 (2014).

[17] R. Meyer, N. Geuder, E. Lorenz, A. Hammer, C. V. Ossietzky, U. Oldenburg, and H. G. Beyer, *Combining solar irradiance measurements and various satellite-derived products to a site-specific best estimate,* in *Proc. of the 2008 ...* (2008) pp. 1–8.

[18] K. Schumann, H. G. Beyer, K. Chhatbar, and R. Meyer, *Improving Satellite-Derived Solar Resource Analysis with Parallel Ground-Based Measurements,* in *ISES Solar World Congress* (2011).

[19] C. A. Gueymard and S. M. Wilcox, *Assessment of spatial and temporal variability in the US solar resource from radiometric measurements and predictions from models using ground-based or satellite data,* Solar Energy **85**, 1068 (2011).

[20] J. A. Ruiz-Arias, S. Quesada-Ruiz, E. F. Fernández, and C. A. Gueymard, *Optimal combination of gridded and ground-observed solar radiation data for regional solar resource assessment,* Solar Energy **112**, 411 (2015).

[21] J. Polo, S. Wilbert, J.A.Ruiz-Arias, R. Meyer, C. Gueymard, M. Šúri, L. Martín, T. Mieslinger, P. Blanc, I. Grant, J. Boland, P. Ineichen, J. Remund, R. Escobar, A. Troccoli, M. Sengupta, K. P. Nielsen, D. Renne, and N.Geuder, *Integration of ground measurements to model-derived data*, Tech. Rep. (Solar Heating & Cooling Programme (SHC), IEA, 2015).

[22] T. Kato, T. Inoue, and Y. Suzuoki, *Estimation of total power output fluctuation of high penetration photovoltaic power generation system,* in *2011 IEEE Power and Energy Society General Meeting* (2011) pp. 1–7.

[23] T. E. Hoff and R. Perez, *Modeling PV fleet output variability,* Solar Energy **86**, 2177 (2012).

[24] A. R. Dyreson, E. R. Morgan, S. H. Monger, and T. L. Acker, *Modeling solar irradiance smoothing for large PV power plants using a 45-sensor network and the Wavelet Variability Model,* Solar Energy **110**, 482 (2014).

[25] P. Gotseff, J. Cale, M. Baggu, D. Narang, and K. Carroll, *Accurate power prediction of spatially distributed PV systems using localized irradiance measurements,* in *IEEE Power and Energy Society General Meeting* (2014) pp. 14–18.

[26] J. Marcos, L. Marroyo, E. Lorenzo, and M. García, *Power output fluctuations in large PV plants,* in *International Conference on Renewable Energies and Power Quality* (Santiago de Compostela, Spain, 2012).

[27] H. Kobayashi and J. ARAI, *Short-term Forecast for Photovoltaic Power Generation with Correlation of Solar Power Irradiance of Multi Points,* in *RECENT RESEARCHES in ELECTRICAL ENGINEERING* (Lisbon, Portugal, 2014) pp. 89–94.

[28] G. Chicco, V. Cocina, P. Di Leo, and F. Spertino, *Weather forecast-based power predictions and experimental results from photovoltaic systems,* in *2014 International Symposium on Power Electronics, Electrical Drives, Automation and Motion, SPEEDAM 2014* (Ischia, Italy, 2014) pp. 342–346.

[29] G. Blaesser and D. Munro, *Report EUR 16339 EN*, Tech. Rep. 4.1 (Institute for Systems Engineering and Informatics, 1995).

[30] IEC 61724:1998, *Photovoltaic system performance monitoring - Guidelines for measurement, data exchange and analysis*, Standard (International Electrotechnical Commission, Geneva, CH, 1998).

[31] M. Gostein, *Update on Edition 2 of IEC 61724: PV System Performance Monitoring,* (2014).

[32] IEC, *TC 82 Working Documents since 2016-01-04,* .

[33] T. Oozeki, T. Izawa, K. Otani, and K. Kurokawa, *An evaluation method of PV systems,* Solar Energy Materials and Solar Cells **75**, 687 (2003).

[34] S. Firth, K. Lomas, and S. Rees, *A simple model of PV system performance and its use in fault detection,* Solar Energy **84**, 624 (2010).

[35] W. van Sark and B. M. N.H. Reich, *Review of Pv Performance Ratio Development,* in *World Renewable Energy Forum (WREF) Congress XII*, 6 (Denver, CO, 2012) pp. 1–6.

[36] T. Nordmann, C. Luzi, and G. Mike, *Analysis of Long - Term Performance of PV Systems. Different Data Resolution for Different Purposes*, Tech. Rep. (Photovoltaic Power Systems Programme(PVPS), IEA, 2014).

[37] Y. Ueda, K. Kurokawa, K. Kitamura, M. Yokota, K. Akanuma, and H. Sugihara, *Performance analysis of various system configurations on grid-connected residential PV systems,* Solar Energy Materials and Solar Cells **93**, 945 (2009).

[38] C. Ventura and G. M. Tina, *Utility scale photovoltaic plant indices and models for on-line monitoring and fault detection purposes,* Electric Power Systems Research **136**, 43 (2016).

[39] B. Schulz, T. Glotzbach, C. Vodermayer, G. Wotruba, M. Mayer, and S. Grünsteidl, *Evaluation of Calibrated Solar Cells and Pyranometers Regarding the Effective Irradiance Detected by PV Modules,* in *25th EU-PVSEC* (Valencia, Spain, 2010) pp. 4797–4800.

[40] S. Reinhardt, C. Eggers, S. Grünsteidl, and C. Vodermayer, *INFLUENCE OF TECHNOLOGY DIFFERENCES IN PERFORAMNCE RATIO CALCULATIONS,* in *27th European Photovoltaic Solar Energy Conference and Exhibition* (Frankfurt, Germany, 2012) pp. 4025–4026.

[41] BHE Renewables, *JUST THE FACTS TOPAZ SOLAR FARMS,* (2015).

[42] SunSpec Alliance, *Best Practices in Solar Performance Monitoring,* (2014).

[43] A. Zagouras, A. Kazantzidis, E. Nikitidou, and A. A. Argiriou, *Determination of measuring sites for solar irradiance, based on cluster analysis of satellite-derived cloud estimations,* Solar Energy **97**, 1 (2013).

[44] R. J. Davy and A. Troccoli, *Continental-scale spatial optimisation of a solar irradiance monitoring network,* Solar Energy **109**, 36 (2014).

[45] D. Yang and T. Reindl, *Solar irradiance monitoring network design using the variance quadtree algorithm,* Renewables: Wind, Water, and Solar **2**, 1 (2015).

[46] A. Zagouras, R. H. Inman, and C. F. M. Coimbra, *On the determination of coherent solar microclimates for utility planning and operations,* Solar Energy **102**, 173 (2014).

[47] Boeing Satellite Development Center, *GOES N Databook,* Tech. Rep. Revision 5.2.2 (2006).

[48] E. S. Agency, *Meteosat second generation: The satellite development* (ESA Publications Division, 1999) pp. 1–55.

[49] M. Sengupta, A. Habte, S. Kurtz, A. Dobos, S. Wilbert, E. Lorenz, T. Stoffel, D. Renné, C. Gueymard, D. Myers, S. Wilcox, P. Blanc, and R. Perez, *Best Practices Handbook for the Collection and Use of Solar Resource Data for Solar Energy Applications*, Tech. Rep. (NREL, 2015).

[50] C. Thomas, E. Wey, P. Blanc, L. Wald, and M. Lefèvre, *Validation of HelioClim-3 Version 4, HelioClim-3 Version 5 and MACC-RAD Using 14 BSRN Stations,* Energy Procedia **91**, 1059 (2016).

[51] Z. Qu, A. Oumbe, P. Blanc, G. Espinar, B., Gesell, B. Gschwind, L. Klüser, M. Lefève, L. Saboret, M. Schroedter-Homscheidt, and L. Wald, *Fast radiative transfer parameterisation for assessing the surface solar irradiance: The Heliosat4 method,* Meteorologische Zeitschrift , 1 (2016).

[52] M. Sengupta, A. Habte, P. Gotseff, A. Weekley, and A. Lopez, *A Physics-Based GOES Satellite Product for Use in NREL ' s National Solar Radiation Database,* in *Solar* (San Francisco, California, 2014).

[53] M. Sengupta, A. Weekley, A. Habte, A. Lopez, and C. Molling, *Validation of the National Solar Radiation Database (NSRDB) (2005-2012),* in *the European PV Solar Energy Conference and Exhibition* (Hamburg, Germany, 2015).

[54] A. Nottrott and J. Kleissl, *Validation of the NSRDB–SUNY global horizontal irradiance in California,* Solar Energy **84**, 1816 (2010).

[55] R. E. Bird and R. L. Hulstrom, *A Simplified Clear Sky Model for Direct and Diffuse Insolation on Horizontal Surfaces*, Tech. Rep. (Solar Energy Research Institute, 1981).

[56] C. Rigollier, O. Bauer, and L. Wald, *On the clear sky model of the ESRA — European Solar Radiation Atlas with respect to the heliosat method,* Solar Energy **68**, 33 (2000).

[57] P. Ineichen, *A broadband simplified version of the Solis clear sky model,* Solar Energy **82**, 758 (2008).

[58] M. Lefèvre, A. Oumbe, P. Blanc, B. Espinar, B. Gschwind, Z. Qu, L. Wald, M. Schroedter-Homscheidt, C. Hoyer-Klick, A. Arola, A. Benedetti, J. W. Kaiser, and J. J. Morcrette, *McClear: A new model estimating downwelling solar radiation at ground level in clear-sky conditions,* Atmospheric Measurement Techniques **6**, 2403 (2013).

[59] J. Davies, D. McKay, G. Luciani, and M. Abdel-Wahab, *Validation of Models for Estimating Solar Radiation on Horizontal Surfaces*, Tech. Rep. (International Energy Agency, 1988).

[60] D. Shepard, *A two-dimensional interpolation function for irregularly-spaced data,* 23rd ACM national conference , 517 (1968).

[61] W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry, *Algorithm XXX: SHEPPACK: Modified Shepard Algorithm for Interpolation of Scattered Multivariate Data,* (2009).

[62] C.-s. Yang, S.-p. Kao, F.-b. Lee, and P.-s. Hung, *Twelve Different Interpolation Methods : a Case Study,* in *Proceedings of the XXth ISPRS Congress*, Vol. 35 (2004) pp. 778–785.

[63] G. Bohling, *Kriging* (Kansas Geological Survey, Kansas, 2005) pp. 1–20.

[64] J. Munkhammar, J. Widen, and L. M. Hinkelman, *A copula method for simulating correlated instantaneous solar irradiance in spatial networks,* Solar Energy **143**, 10 (2017).

[65] F. V. Gutierrez-Corea, M. A. Manso-Callejo, M. P. Moreno-Regidor, and J. Velasco-Gómez, *Spatial estimation of sub-hour Global Horizontal Irradiance based on official observations and remote sensors,* Sensors (Basel, Switzerland) **14**, 6758 (2014).

[66] M. Franklin, *Solution to Ordinary and Universal Kriging Equations,* , 1 (2014).

[67] M. Eriksson and P. P. Siska, *Understanding anisotropy computations,* Mathematical Geology **32**, 683 (2000).

[68] S. Arlot, *A survey of cross-validation procedures for model selection,* Statistics Surveys **4**, 40 (2010).

[69] E. Pebesma and B. Graeler, *Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simulation*, Tech. Rep. (2017).

[70] T. M. Hansen, *mGstat*, Tech. Rep. (2011).

[71] D. J. Bora and A. K. Gupta, *Effect of Different Distance Measures on the Performance of K-Means Algorithm : An Experimental Study in Matlab,* International Journal of Computer Science and Information Technologies **5**, 2501 (2014).

[72] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Greedy algorithms,* in *Algorithms* (McGraw-Hill Education, 2006) 1st ed., pp. 139–161.

[73] S. Bell, *Measurement*, Tech. Rep. 2 (The National Physical Laboratory, 1999).

[74] Harvard University, *A Summary of Error Propagation,* (2007).

[75] F. Pavlovcic, J. Nastran, and D. Nedeljkovic, *Non-Gaussian Probability Distributions,* in *XIX IMEKO World Congress* (Lisbon, Portugal, 2009).

[76] H. Sommer, *Polygeom: Polygonal Approximation of Boundary Integrals for Area, Centroid and Area Moment of Inertia,* (1998).

[77] H. Wirth, *Recent facts about photovoltaics in Germany*, Tech. Rep. (Fraunhofer ISE, 2017).

[78] EIA, *Levelized Cost and Levelized Avoided Cost of New Generation Resources in the Annual Energy Outlook 2016*, Tech. Rep. (2016).