Spring 7-24-2013

Solar Data Analysis

Mike C. T. Ray
Portland State University

Solar Data Analysis

by

Mike C.T. Ray

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Thesis Committee:
Robert Bass, Chair
Yih-Chyun Jenq
Martin Siderius

Portland State University
2013

ABSTRACT

The solar industry has grown considerably in the last few years. This larger scale has introduced more problems as well as possibilities. One of those possibilities is analyzing the data coming from the sites that are now being monitored, and using the information to answer a variety of questions.

We have four questions which are of prime importance identified in this thesis:

1. Can data from customers be trusted?
2. Can we use data from existing sites to determine which sites need the most improvement?
3. Can we implement a location-based algorithm to reduce the amount of false positives for performance, or other alarms?
4. Can we improve upon the current predicted power algorithm?

We find that not only can we answer these questions definitively, but the improvements found are of significant value. Each of these items represents an important question that either directly or indirectly translates into increased revenue and engineering improvements for the solar industry as a whole.

ACKNOWLEDGEMENTS

First and foremost, I'd like to thank my wife for putting up with long nights working on this thesis. I'd also like to thank DECK monitoring for their data dump – and in particular Ben Weintraub for being particularly helpful. Thank you Dr. Bass for going back/forth on revisions quicker than could have been expected. Also, thank you Dr. Jenq and Dr. Siderius for being willing to sit on my committee with such short notice. Without any of you, none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

Installation of solar PV is expanding at a very rapid rate, and its rate of growth has only been increasing with time as is shown in Figure 1. In 2007, there were 3 GWp (peak gigawatts of electrical power) of solar installed in the US. In 2011 there were 23.5 GWp installed with a staggering 6.1 GWp installed that year alone (Navigant Energy, 2012).

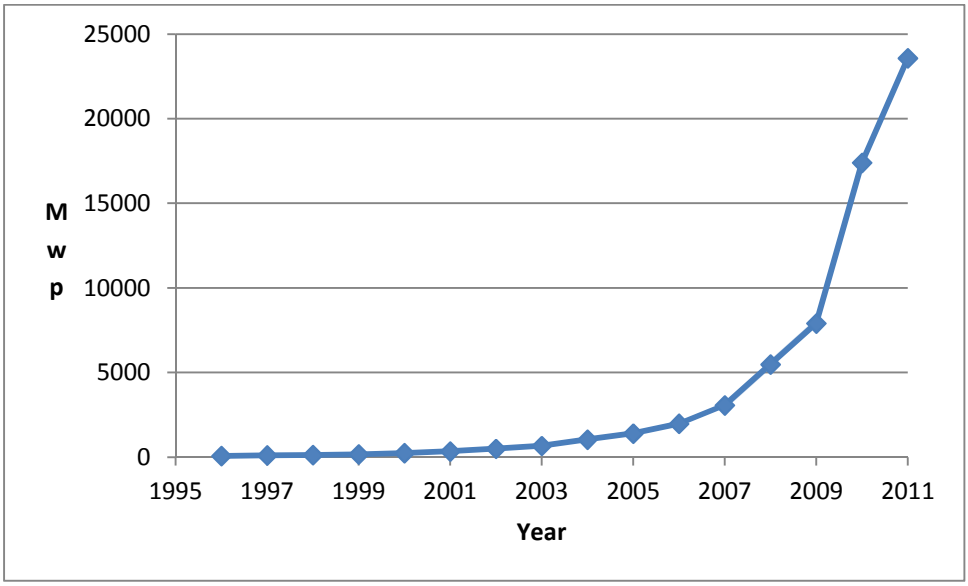

Figure 1: Chart showing solar installations as measured by total nameplate rating (Navigant Energy, 2012).

Solar has been presented with particular challenges for the return on investment. Mainstream solar adoption is not currently possible without government subsidy, although it is getting closer with each day (The Economist, 2012). In the past 8 years alone, average solar panel efficiency has increased from 12.4% in 2004 to 15.4% in 2011 – a rise of almost 25% (IEEE Spectrum, 2012). Experimental cells in 1980 were typically

obtaining under 15% efficiency (depending on the company), with most being well under, while experimental cells today are achieving as high as 43.5% efficiency (NREL, 2010). Cost per watt of solar panels has decreased from $286/watt in 1954 (Perlin, 1999) to current inventory sales of $0.65/watt in today's market (Navigant Energy, 2012).

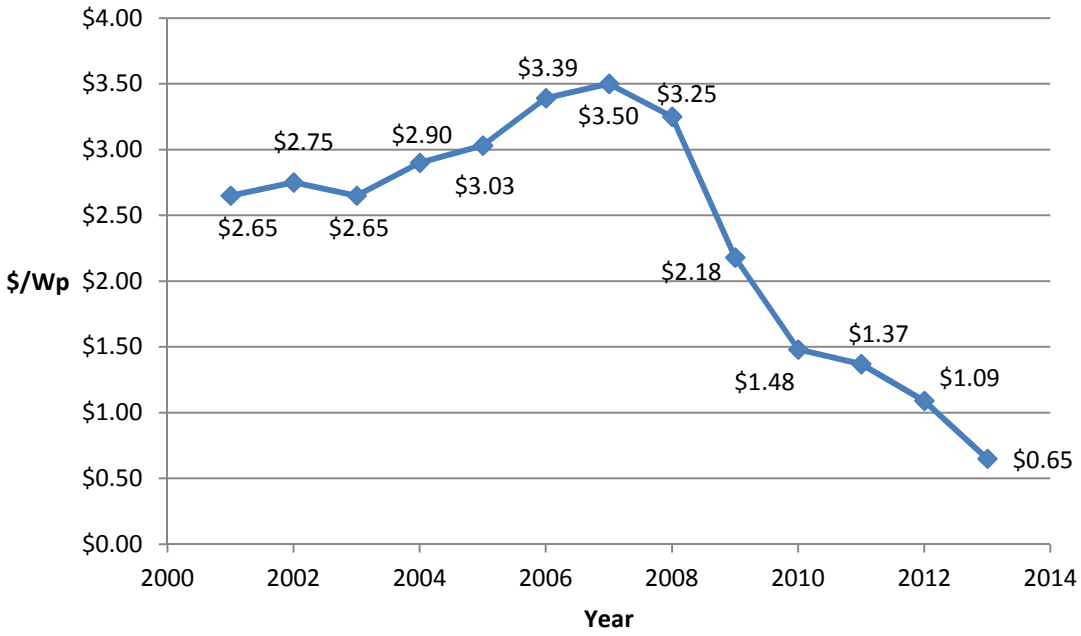

Figure 2: Dramatic decline of solar panel prices (Navigant Energy, 2012). The cost of solar panels used to be the primary cost for solar installations. Now, it is sitting at $0.65 per watt, which represents a smaller fraction of the current $4/Wpk installed.

Due to the dramatic increase in efficiency, as well as the even more dramatic decline in prices, solar panels no longer make up the majority of the cost of a given solar system. Solar panels require inverters, racking, and labor which bring installed costs to around $2-3/watt in 2012's utility-scale solar market. Total residential materials and labor prices are currently at roughly $4-5/watt (Gwinner, 2012), and commercial/industrial

installations are somewhere between these the residential and large utility scale prices. The US department of energy has created a program, the Sunshot initiative, with the aim to decrease the cost of installed solar to $1/watt by 2017, and $0.73/watt by 2030 (Kanellos, 2011). We can see from Figure 3 that as the cost decreases, the amount of capacity that would become economical increases dramatically. At $1/watt, solar power would become one of the most economical forms of energy with a payback period of less than 10 years in most areas of the US. Once grid parity is reached, solar subsidies may no longer be necessary – and this day is approaching quickly. For comparison, a new coal plant costs about $3/W to build while a Simple Cycle Combustion Turbine (SCCT) gas plant costs around $1/W to build. However, both of these also require fuel thereby adding to the cost per MWh, while the "fuel" for PV is free (The Economist, 2012). Worth noting however is that natural gas and coal plants have high capacity factors – generally above 0.8. Solar PV installations have capacity factors of roughly 0.2 – though most of that energy is produced during peak hours when electricity is the most expensive.

Swanson's law is a similar version of Moore's law. It suggests that with each doubling of worldwide production of solar panels, there is a 20% decrease in price per watt. As solar catches on, solar cells (which are already not the primary cost component of most solar farms) will continue to decline in price (The Economist, 2012). It would be reasonable to assume that with larger volume, the inverter, racking, labor, and other associated costs would decrease as well.

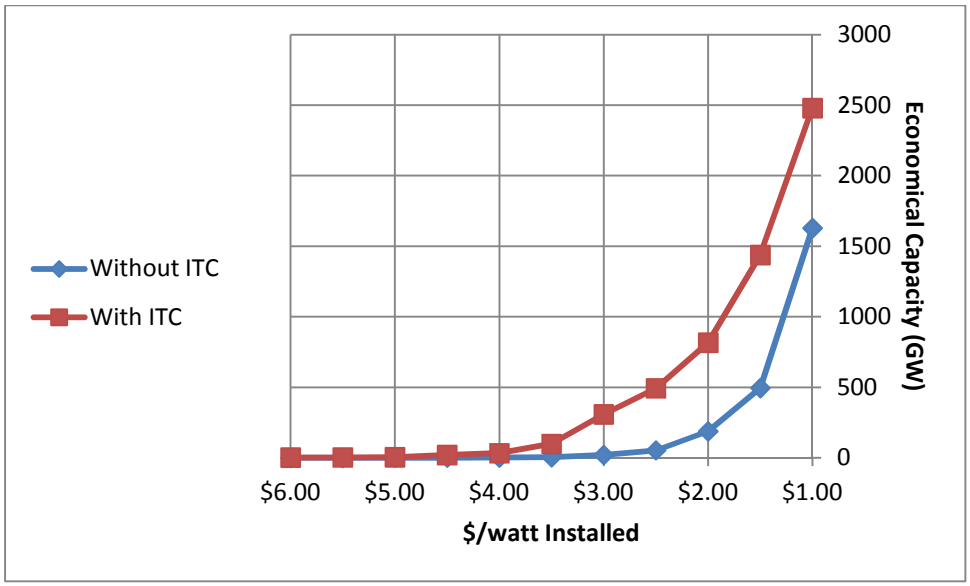

Figure 3: Chart showing cost of solar vs. economical amount to install. ITC is the investment tax credit (federal subsidy) (Keiser). It shows that when solar installations are cheap (whether augmented by subsidy or not), it makes economical sense to install them in more places. For example, if solar installations were expensive, it would only make sense to install them in places where power could not otherwise be obtained cheaply (such as remote cabins, or other rural areas without electrical grids). However, as the price declines, it begins to make sense to put them in many more places – non-south facing roofs, roofs partially shaded, unused land, etc…

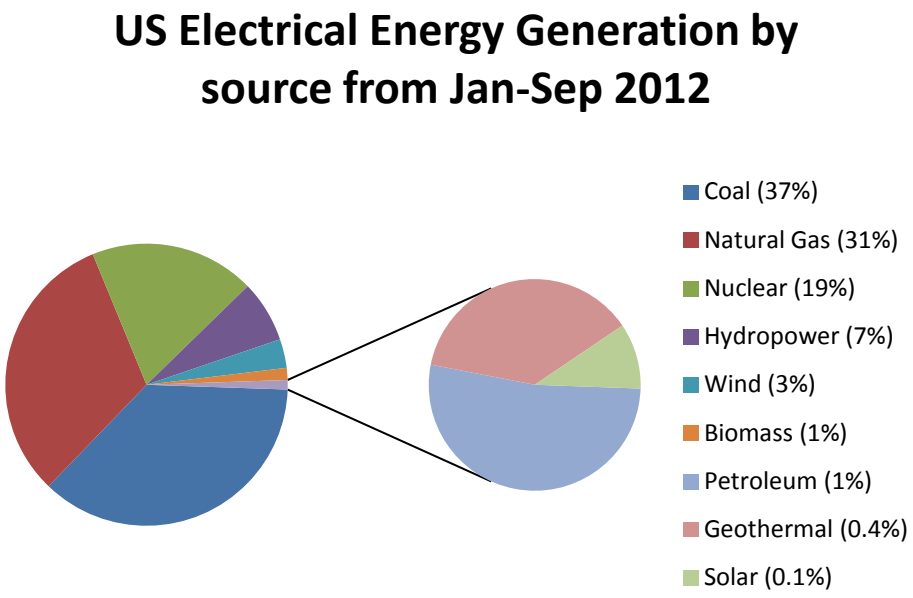

Figure 4: Chart showing breakdown of electrical energy coming from each energy source (EIA, 2012). Notice the currently small contribution of solar.

One way to increase the economic viability of solar power is to increase the amount of power and energy generated per panel. There are a variety of common issues which can be resolved by system owners. These issues can be identified by analyzing the production data and other data coming from each site, and comparing these data with past performance and performance by similar systems. With the rise of monitoring companies, these data are now more available than ever. As solar power increases its penetration into the power grid, utilities and system operators will need to be able to reasonably predict the real-time power output from these sites. As such, it is of critical value for the utility to have as good of a prediction of power production as possible. This thesis focuses on the development of algorithms that facilitate these two categories of ideas.

Background

Currently, there are a variety of firms performing monitoring and analyzing the collected data. Three of the largest players in this field are detailed below along with their respective patents and public-facing information. Unfortunately, the industry in general is secretive and does not divulge much information to people who are not potential customers. It is likely that current data analysis has gone far beyond what is detailed in the following background section.

2.1 DECK Monitoring

DECK currently has no patents, nor do they directly do their own data analysis. Clean Power Research currently does the analysis in a partnership with DECK. Together, they provide data analysis-based services.

2.2 Clean Power Research

Clean Power Research holds several relevant patents, detailed below.

Computer-Implemented System and Method for Determining Point-To-Point Correlation of Sky Clearness for Photovoltaic Power Generation Fleet Output Estimation (#8,165,811) & Computer-Implemented System and Method for Estimating Power Data for a Photovoltaic Power Generation Fleet (Hoff, 2011)

Summary: Using the data feeds coming from weather stations, satellite feeds, other nearby sites, or from any other sources, put together a given ‘clearness index’ vs. time. The scale of this index goes from 0 (totally obscured) to 1 (totally clear). The model predicts the amount of power that should be generated, and then this power figure is multiplied by the cloudiness index to obtain the estimated real-time power. In practice, even on a cloudy day, the cloudiness index is still greater than 0.5. The primary benefits of this method are:

- Getting real-time production data from the entire photovoltaic fleet
- Predicting power for a future site under investigation
- Supporting time resolution of power in real-time, not necessarily with a lag of the sample time
- Providing real-time current results to power systems operators for their use – without a lag of the sample time
- This sample time lag is a key reason why direct measurements will inherently be inferior to using a data aggregation method such as the one in this patent. By predicting the power being generated, one can even look into the future with reasonable accuracy – which obviously is impossible with measured data. In the larger public measurement networks, the delay can be 20 seconds to 1 minute at the very least, which is very significant for power systems operators.

Although not directly along the same lines as what is covered in this thesis, it goes along the same lines of trying to use data to predict the power and derive information for the various interested parties.

Computer-Implemented System and Method for Efficiently Performing Area-To-Point Conversion of Satellite Imagery for Photovoltaic Power Generation Fleet Output Estimation (Hoff, 2011)

Summary: One of Clean Power Research's major goals is to input various data streams into their model-building algorithm for solar sites. This requires the processing of these data streams – but more importantly, the efficient processing of those streams. This patent covers how to efficiently parse satellite solar irradiance feeds into input for the algorithm. As such, it is only tangentially relevant to this thesis and will not be covered in further detail.

Method and Apparatus for Distributed Generator Planning (Lee & Zazueta-Hall, 2010)

Summary: Details software to plan solar power systems. Predicts the power that they will generate, and from that, estimates the return on investment. Likely is using the models which are generated from their data analysis work to drive the numbers. Accounts for a variety of design factors such as orientation, microinverter model, panel model, etc… This patent is only tangentially relevant to this thesis and will not be covered in detail.

2.3 Draker + Solar Power Technologies (Merged to one company in 2012)

Draker/Solar Power Technologies provide their own data analysis services. Draker does not have any patents, while Solar Power Technologies has one patent which is unrelated to data mining of solar data.

2.4 Locus Energy

Locus does their own data analysis and provides it as a software service that they sell. They currently hold two relevant patents, detailed below.

1. Comparable Diagnostics for Renewable Energy Power Systems (Peleg, Herzig, & Kerrigan, 2010)

Summary: Estimate generation from data collected from multiple solar hot water and solar PV sites. A model of the system can be built to estimate performance. Once this is built, it's possible to find deviations from the model that would be due to temporary issues such as soiling. Then, the performance gain can be estimated from resolving the issue. Locus can also determine the value of the system and show it to the customer to reinforce the positive aspects of solar power production. The model includes system parameters such as:

- Roof Pitch/Building Orientation/General Orientation
- Expected Sunlight
- Expected Generation
- System Size
- System Technology
- System Tolerances (accuracy of measurements)
- Shading
- Adjustment Factor (to reduce false alarms if the sites' model is problematic)

With location-based data, the customer can also be notified if their site is underperforming compared to nearby sites, and by how much. If a given site is underperforming due to a resolvable issue (such as leaves falling on the array), and if the

problem is flagged, then a technician can be dispatched to fix the problem. Rapidly addressing problems in this manner results in more energy production per installed power unit (capacity factor).

Locus also points out that such a system is essentially of negligible cost to duplicate once the framework has been implemented since it is implemented in software on servers. The alternative would be to use expensive sensors at each site that would likely not perform as well.

In this patent, Locus picks 40 random sites from a pool of all sites within a given radius. I have found this to be suboptimal, as not all nearby sites are correlated sites. When there is a surplus of sites to choose from, a methodical selection of 'most correlated sites' should be used (in line with this thesis).

2. Estimating Solar Irradiance Components From Plane of Array Irradiance and Global Horizontal Irradiance (Kerrigan, Williams, & Herzig, 2012)

Summary: Estimating global irradiance using aggregated location-tagged data coming from solar PV systems and solar hot water systems. They filter and adjust their results to account for the following factors:

- Fixed system vs. system w/ tracker
- Systems that are limited during their peak production time by the inverter rating
- Effect of cloudy vs sunny days on different types of solar PV systems (thin film, crystalline, etc…)
- Effect of temperature on different types of solar PV systems
- Flat plate vs. West-tilted
- Inclination of the earth at that location
- Partial shading at specific locations

Once they determine this irradiance data vs. location, they can create a model to predict the output and the value of a solar installation at a given location. Compared with

current methods, they can obtain real-time results that are more accurate, more granular and more detailed.

A specific relevant example of how they can be more detailed is by breaking apart the solar irradiance value into normal irradiance, ground-reflection and diffuse irradiance. Solar panels that are oriented horizontally pick up diffuse irradiation far better than solar panels that are tilted to receive maximum solar exposure for that site. Also, satellite feeds are particularly prone to ground-reflection, such as when there is snow on the ground. For example, if a satellite feed is used and it gives a value of '800W/m^2', this is far inferior to the breakdown of 10% ground reflection, 80% normal irradiation and 10% diffuse irradiation.

They can also use satellite data and other measurement data as additional data points in their model – so through aggregating all measurements and combining it with their other data, they can outperform a given measurement.

This particular paper details methods and ideas that are along the same lines as the ideas presented in this thesis. One major improvement made in this thesis, however, is a more sophisticated selection algorithm for determining correlated sites used for the alarm false positive reduction, but which could easily be applied to generate a more correlated set of sites being used as input to the Locus algorithm. The Locus algorithm accounts for uncorrelated noise by increasing the variance, which is likely a suboptimal approach in cases where they are many sites nearby, but is an optimal approach for areas where there is a smaller density of solar sites.

The patent also covers topics that are not relevant to this thesis but are good ideas nonetheless. Ideas such as automatic site categorization, wind system modeling and usage

in combination with building energy and solar data, identifying correlated components (such as sunrise, or a potentially infinite list of fields) and profiling of a business based on their consumption data are covered.

Verifying customer data input

One of the assumptions which make up the foundation of this thesis is that customer-provided data can be trusted. Otherwise, the DECK dataset, which contains large amounts of customer-supplied data, would not be useful. Although no dataset is perfect, we need to measure the degree of that imperfection and change our analysis approach if necessary.

3.1 Introduction to the California Solar Initiative (CSI) dataset

The principle objective of the California Solar Initiative is to promote the installation of solar hot water heaters and photovoltaics. A secondary objective is to provide public data which will further the cause of these installations both inside and outside of California as well. (California Solar Initiative)

Below is a subset of the fields available in the dataset which are relevant to this thesis:

- Nameplate of the installation
- CEC PTC rating
- Design Factor
- Residential, Commercial, Government, or other type of installation
- Address
- PV module manufacturer
- PV module model
- PV module quantity
- Inverter manufacturer
- Inverter model
- Inverter quantity
- Completion date
- CSI Number

There is also a month-by-month summary of the generation from each of these sites. We source the data used in this thesis from the California Solar Statistics website (California Solar Initiative). In verifying our customer data inputs, we treat the CSI data as the 'gold standard.' We treat errors in the DECK data as legitimate errors.

3.2 Why is it necessary to verify, and how do we verify customer-sourced information?

One of the key problems faced in the data analysis going forward is assuming the data is reliable. Although we can create algorithms to weed out bad data, there simply is only so much we can determine from looking at the raw data.

One field, which is in both the DECK dataset as well as the CSI dataset is the nameplate rating of the site. This is one of the few pieces of data that we can verify in our database against a known good source. If we can crosslink the value given to CSI (known good data since that is how their incentives are calculated) versus the value provided to us, we can see how reliable customer-provided information really is. The CSI database also has CECPTC ratings, so both typical ratings are covered. The CECPTC rating is devised by the California Energy Commission as a more realistic 'actual' nameplate. The default nameplate rating (STC) is typically considered optimistic in most installations.

It is important to point out that many times, these projects do have licensed professional engineers who do understand the design they created. However, many times, the person with whom the monitoring company would contact is a project manager who may not fully understand electrical terms (such as nameplate).

3.3 Verification

We look at the data using the following algorithm, expressed here in psuedocode:

Loop for all sites

If customer nameplate is within 2% of the CSI standard nameplate Then

Put it in the ‘standard nameplate’ bin

Elseif customer nameplate is within 2% of the CSI CECPTC nameplate Then

Put it in the ‘CECPTC nameplate’ bin

Otherwise

Put it in the ‘neither’ bin

To summarize, for each site, we check to see if the nameplate is within 2% of the CSI standard nameplate. If it is, we put it into the ‘standard nameplate’ bin. Otherwise, if the nameplate is within 2% of the CSI CECPTC nameplate, we put it into the ‘CECPTC nameplate’ bin. If it doesn’t meet either of these criteria, we put it into a ‘neither’ bin.

3.4 Results

Below, you can see the results of running our algorithm on the data. The majority of the data falls into one of the two categories we’ve identified, but a significant 24% do not fall into either category. Average percent error of those in the ‘Neither’ category: 13%[1]

[1] Note: DECK is very clear in asking for ‘standard’ DC nameplate information or CECPTC – there is no ambiguity, so any discrepancy between CSI data and DECK data is either procedural error, or more likely customer error.

Table 1: Customer error versus various nameplates

Standard Nameplate	65%
CECPTC Nameplate	11%
Neither	24%

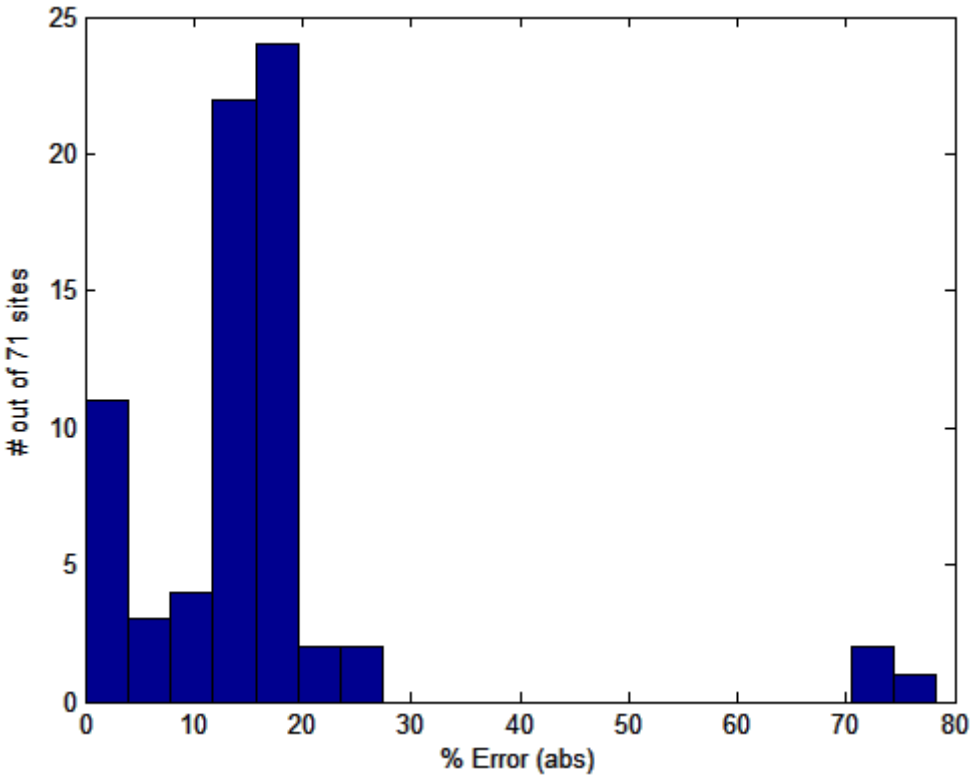

Figure 5: Figure shows how far customer-provided numbers are from the CECPTC nameplates. Some of the values are considerably inaccurate (70-80% error).

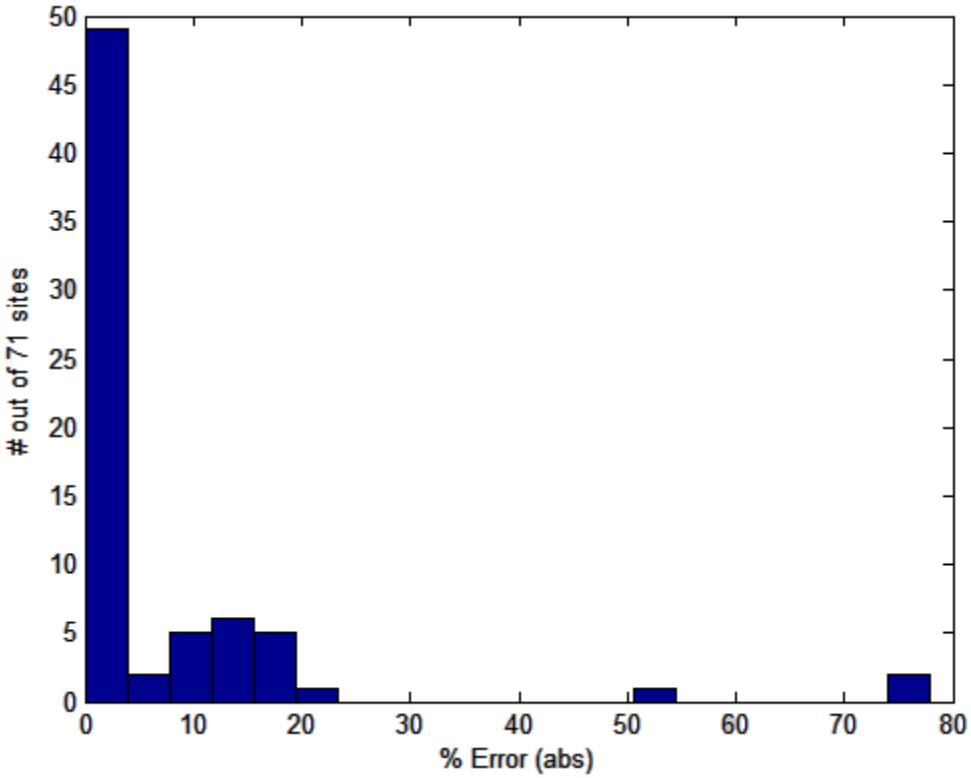

Figure 6: Figure shows how far customer-provided numbers are from the standard nameplates. Some of the values are considerably inaccurate (70-80% error).

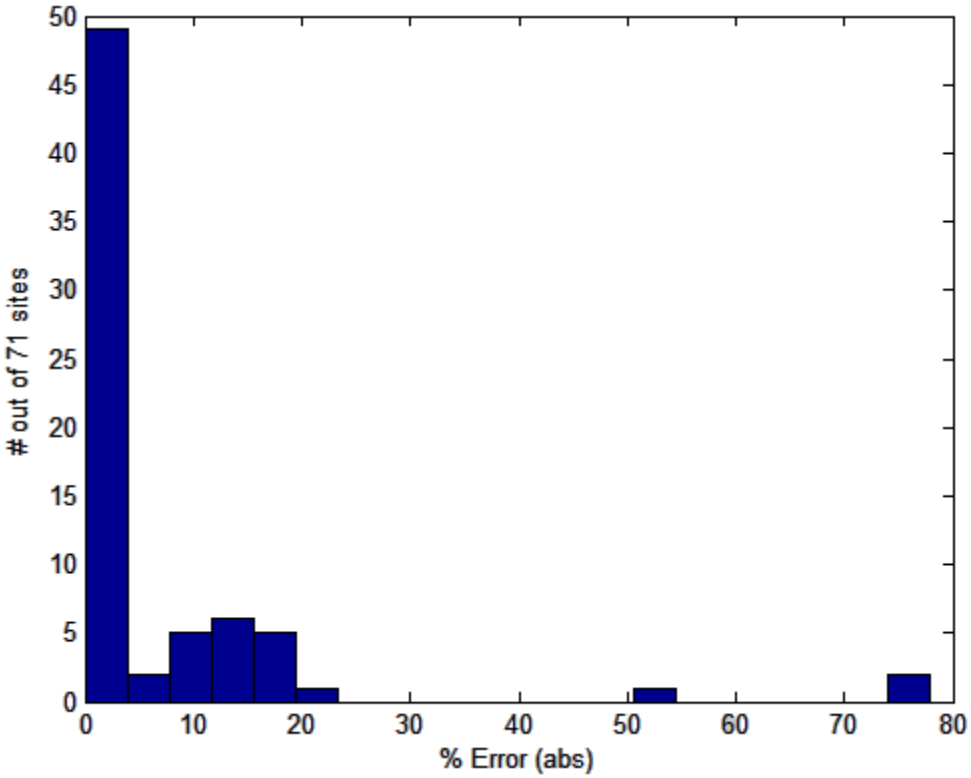

Figure 7: Figure shows the minimum error between the provided nameplate and the CECPTC nameplate. Some of the values are considerably inaccurate (70-80% error).

3.5 Analysis of Results

The first thing that we notice from the customer data is that most customers can be trusted to provide at least a ballpark estimate of the nameplate, although some customers may not be precise about it. There are about a quarter of customers who either do not understand, or don't care to provide the accurate nameplate rating.

Nameplate rating is particularly important since it is a number that in many ways defines a site. If the customer cannot provide accurate nameplate data, they certainly cannot be assumed to provide any other numbers or information accurately.

If we are to draw conclusions upon the data based on differences of (sometimes) as little as a percent, then it is critical to identify the users who do not provide reliable data and eliminate their information from the dataset and/or identify erroneous data and eliminate it from the dataset. One potential method of doing this is to take site information only from stamped electrical drawings, and have a qualified party review the drawings and provide the correct figures. This would at least guarantee a qualified person with attention to detail would get the correct data from the engineering drawings.

It would not eliminate the possibility of the installation being different from the drawings, or the drawings being incorrect. However, one would logically assume that at least most of the errors would be eliminated with this method.

Use data to find out which sites need most improvement

We seek to investigate the use of data analysis algorithms to determine which sites are not performing like their similar peers. We can then suggest design changes for both the current site(s) as well as future designs to optimize the power generated.

4.1 Why are data cleaning functions necessary?

We wrote a data cleaning function since the data was noisy. Inverters frequently fail, or there are communication issues. A ‘bad’ model of inverter can be offline 10% of the time. In calculating capacity factor for a given site, obviously one should not penalize for data acquisition during system downtime, or perhaps should not penalize for inverter downtime either. In Figure 8: Shows a segment of time which has zero power generation, probably due to a maintenance event or equipment failure., we can see that without a data cleaning function, the output for several months is not as cyclical and predictable as we would like. Figure 9 and Figure 10 show additional problems with the data that have to be ‘cleaned out.’

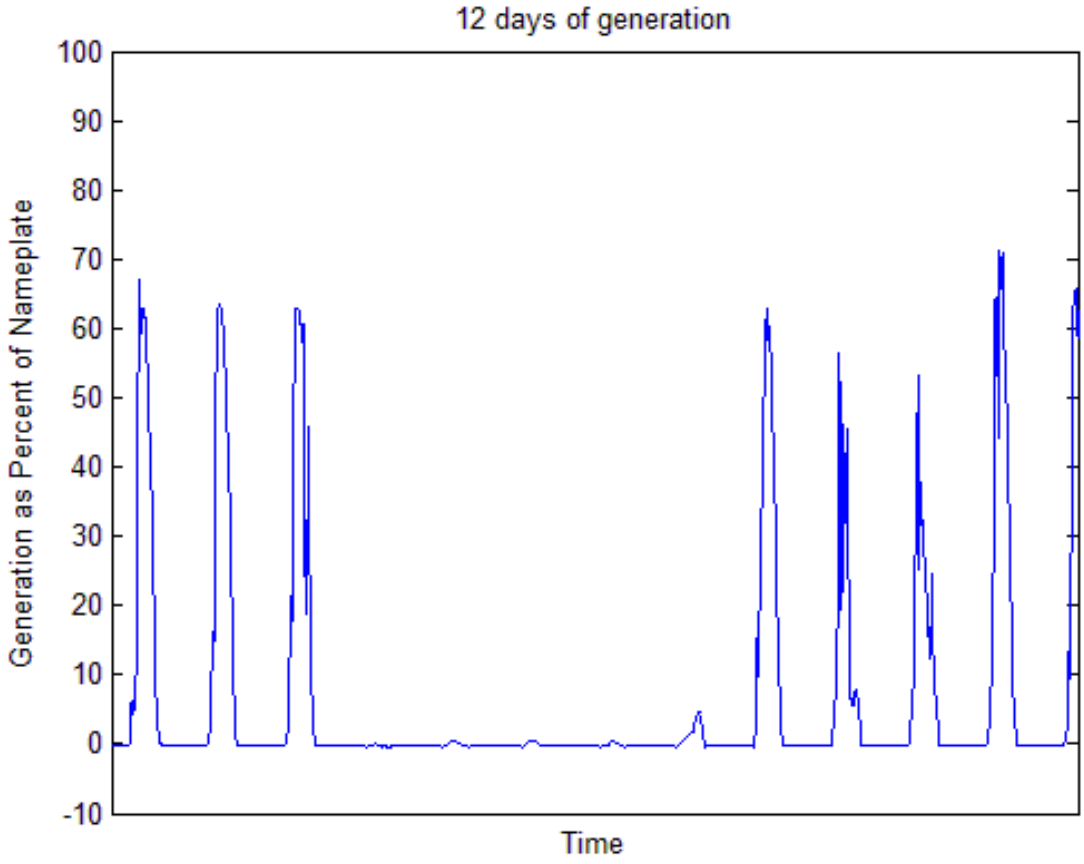

Figure 8: Shows a segment of time which has zero power generation, probably due to a maintenance event or equipment failure.

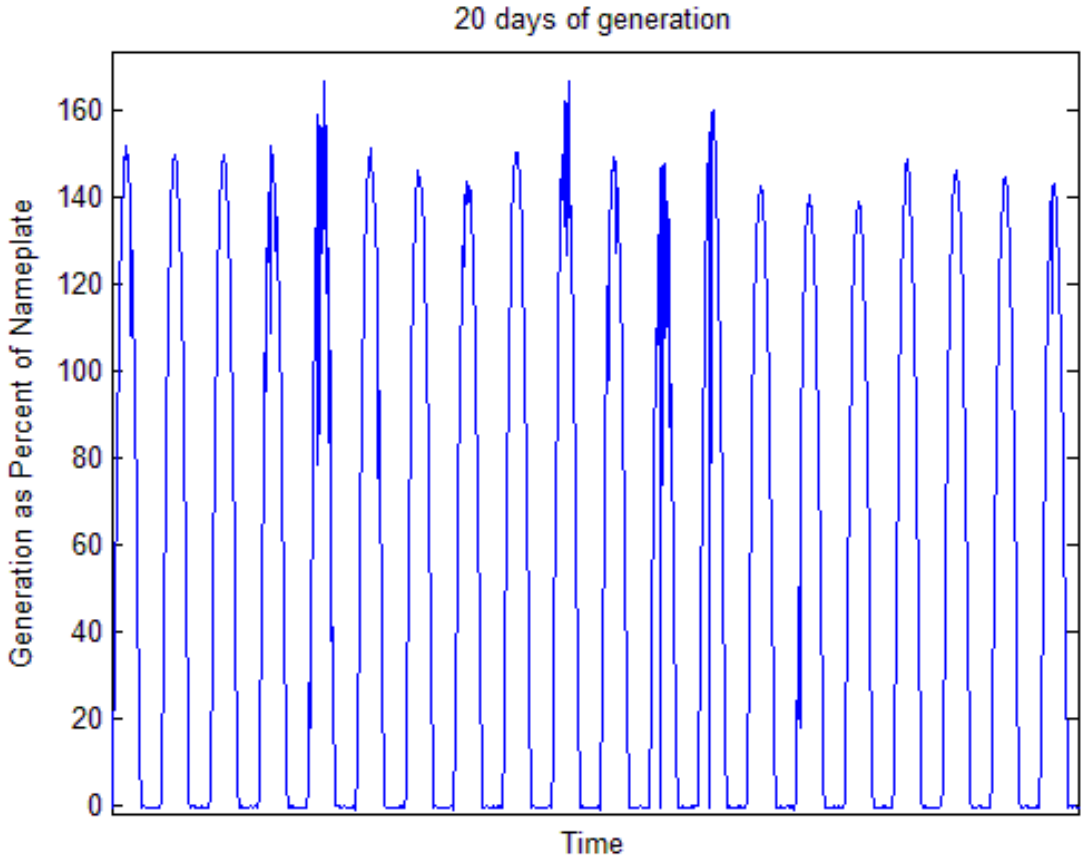

Figure 9: Data error showing peaks of 140-160 percent of nameplate generation every day (each peak occurs at approximately noon of that day). This site probably has an incorrect nameplate rating and should be removed from our dataset.

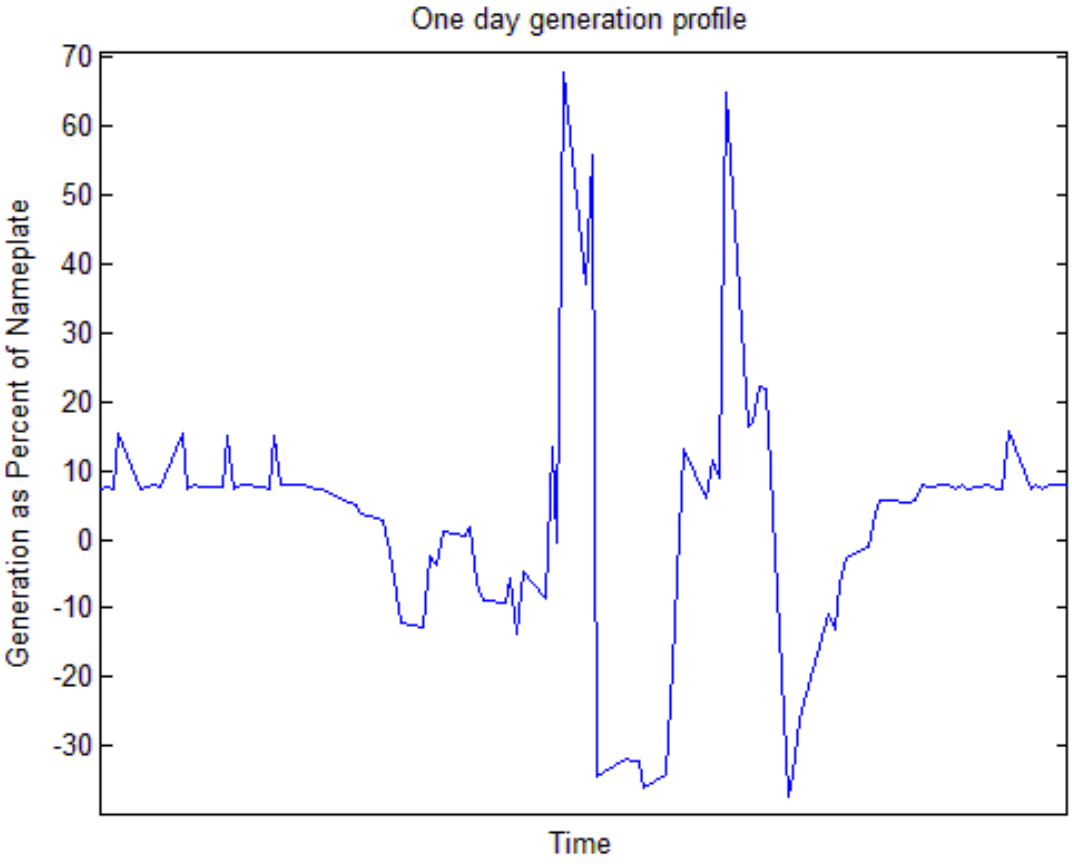

Figure 10: Example of one day at a site with a building energy component which consumes net power (as shown by the large negative 'production' values.

4.2 What data cleaning algorithms are there in place?

We created a cleaning algorithm, the 'CleanData' algorithm. The algorithm does the following, again in psuedocode:

Loop for each site

Loop for each day

If the day had less than 1% of nameplate generation, remove day

// Purpose: If the day has less than 1% of the nameplate generation, this means that there was probably an equipment failure or maintenance event. For determining the capacity factor for a site which operates normally, we don't want to include days with equipment failures/maintenance events.

If the day had more than 130% of nameplate generation, remove day

// Purpose: If the site had more than 130% of nameplate generation that day, it is probably a site where the nameplate information does not match the equipment installed.

If the lowest datapoint had less than -10% of nameplate generation, remove day

// Purpose: Some sites are miscategorized building energy sites. We don't want to be doing solar analysis on sites with building energy systems for this thesis due to unnecessary complication.

If more than 25% of the days were removed, remove this site from the database

// Purpose: If the site had so many days removed, it is probably not an 'ordinary site' that we want to investigate.

4.3 What can we learn from this data?

One way in which the data can be useful in a broader sense is identifying problematic sites. These are sites which generate less power than their neighbors, or less than they should according to an established metric.

Sites that have very low capacity factors right next to sites that have very high capacity factors are obviously sites of interest. These sites might have issues which can be corrected economically.

We came up with the nameplate rating by pulling the maximum value of the generation during the full data period. This is a much closer estimate of the nameplate information than what the customer provides us.

After cleaning the data, we then calculate the capacity factor by totaling generation for every 15 minutes, and averaging it over the year. Then, we plot the results on a lat/long graph. Refer to Figure 11 below. We have been asked by the company that provided us with data, DECK Monitoring, to not show this map for concern of their customer's privacy. Figure 11 is a fictitious map demonstrating the visual value such a map can provide.

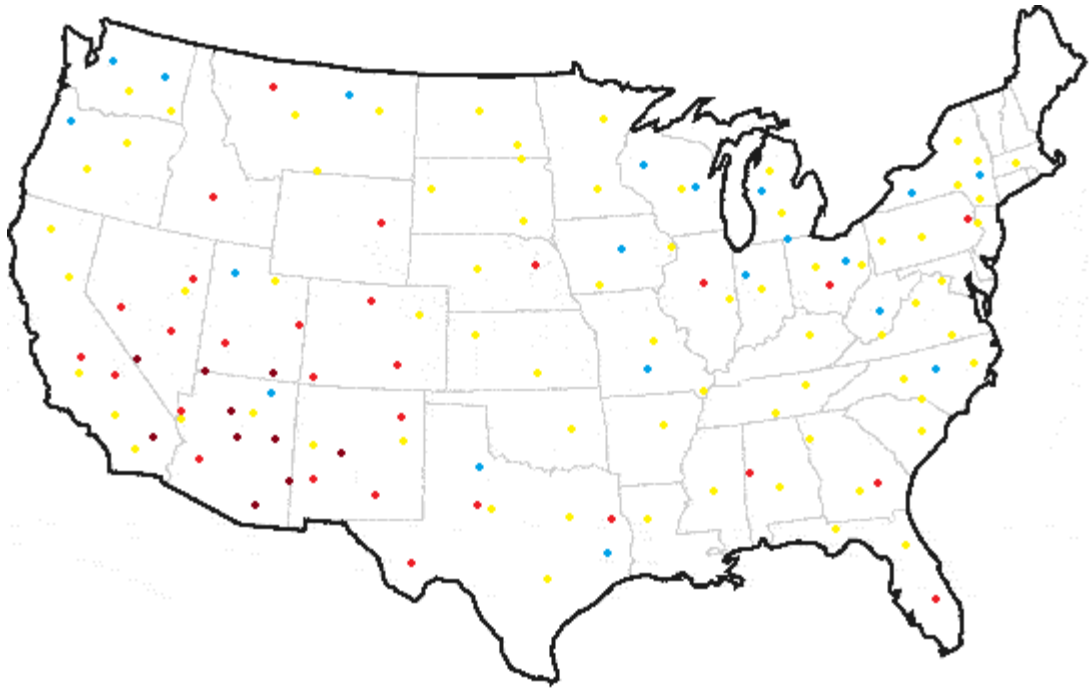

Figure 11: Example of various sites represented by docs whose color corresponds to their capacity factor. The closer the color is to dark red, the higher the capacity factor.

What we find from the graphical representation of capacity factor is very interesting. For one thing, there is the obvious dependence of capacity factor on latitude. As we move from New Jersey to California, we see a general shift from 10-15% capacity factor to 15-20+% capacity factor.

Worth noting is that a difference between 15% and 20% is a difference of 33% improvement. This has implications for how many panels should be paired per inverter of a given rating for example. It also means that the 'bang for the buck' of solar is far higher in a sunny area like California than New Jersey. This is particularly important as solar gets closer to grid parity in the US; we may see unsubsidized solar installations being built in areas which traditionally do not have large solar industries due to lack of subsidy. We can see that there are areas of the southwestern US which are optimal for solar

installations, but lack large numbers of solar PV installations.

A key conclusion to draw from this graph is that in a given geographical area, not all installations are equal. We see numerous sites in California for example where there are extremely high performing sites directly adjacent to high performing sites (15-20% vs 20+%). Although a 5% difference does not seem huge, it represents a 20% increase in energy generation for essentially identical site placement. We see the same thing in New Jersey, albeit with a greater spread. Based on this data, it is entirely possible that there are some sites in New Jersey that are producing twice the energy of other sites nearby relative to their nameplates.

This case demonstrates the value of this algorithm; a visual tool allows for quick analysis of multiple sites, revealing information that can be used to improve energy production. Perhaps in further analysis, we may find that there are some installers that perform far worse than others – installers who would be open to consulting services to correct their mistakes.

4.4 Verifying that there are not problems with the data cleaning and algorithm

As a test of the data cleaning algorithm, a built-in plotting function is used. This plotting function plots valid data points in blue and highlights points slated for removal in red to show which points are being removed. We can then spot-check some of the graphs to make sure that the common data problems are indeed being removed correctly, and that no data points are erroneously being removed.

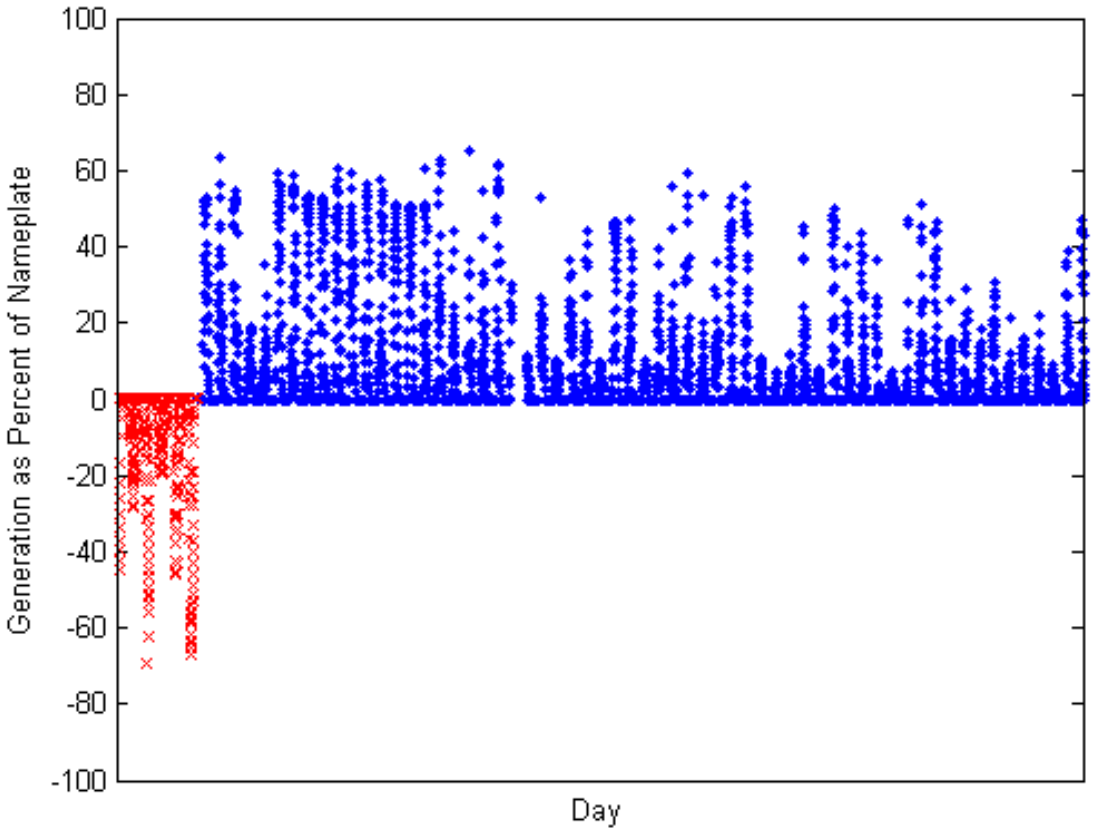

Figure 12: A segment of time is shown which has large negative power generation, probably due CTs facing the wrong direction – a common mistake. This problem has been shown to be corrected after a few weeks. We want to remove the erroneous data points for when the CTs were flipped.

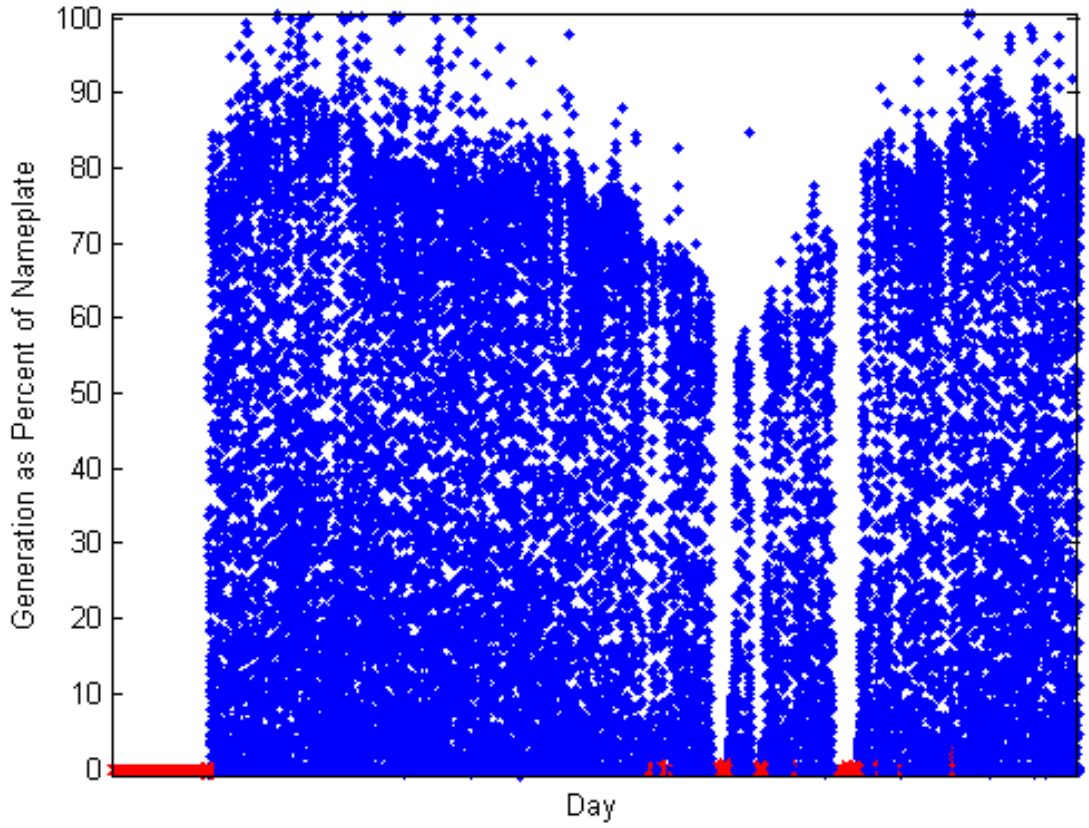

Figure 13: We can see periods of minimal generation for entire days are flagged for removal. They are treated on a day-by-day basis, rather than as individual 15 minute data points. This is correct behavior. These long periods of 0 generation indicate either downtime for the power generation, or downtime for the metering equipment. We do not want to be using this data.

4.5 What is the extent of the improvement that can be seen at selected sites?

Example 1:

We are going to examine two sites which have similar locations, but one site performs worse than the other. We are going to try to determine what could be causing this difference.

Let's take a look at Site 38 and Site 41. They are about a mile apart. See Figure 14 for Site 38's power graph and Figure 15 for Site 41's power graph. We can see something strange about the graph for Site 38, namely that it does not appear to follow the same smooth seasonal pattern that Site 41 does. It seems to just stop at an arbitrary level around ~83% of nameplate capacity. If we overlay the two production graphs, as is done in Figure 16, we can see this difference even more clearly.

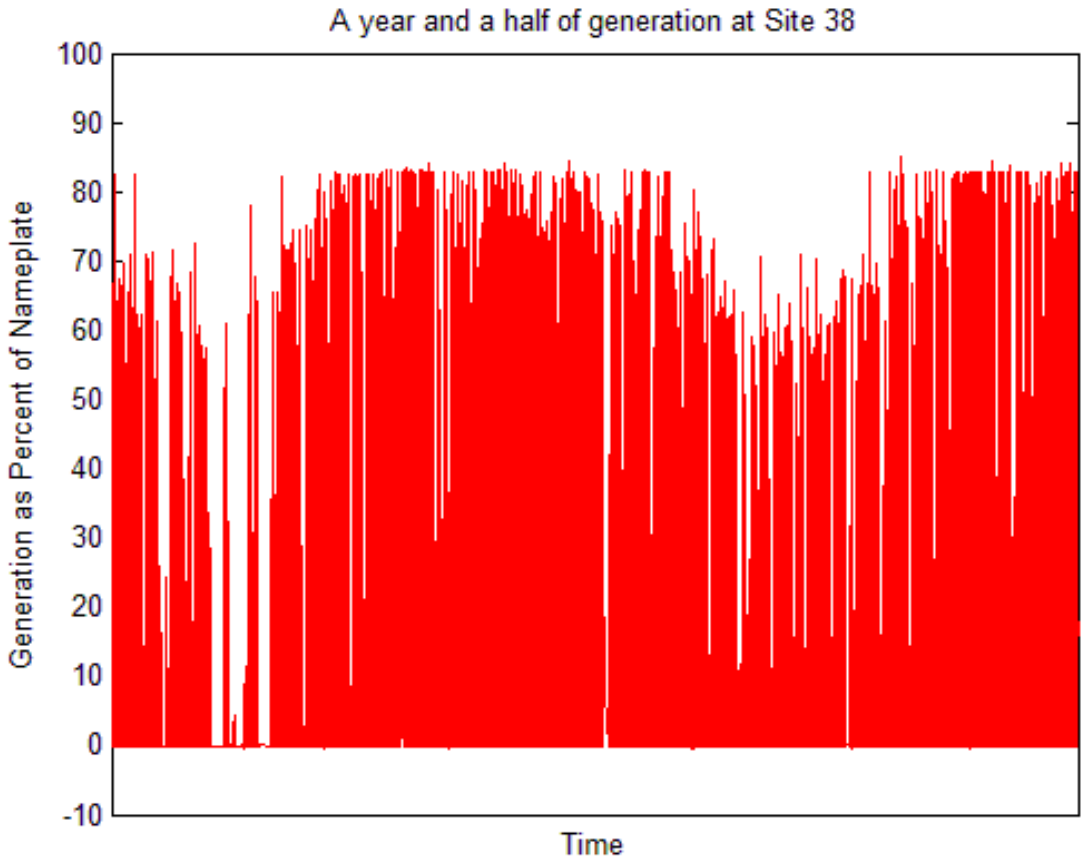

Figure 14: Site 38's power over time as a percentage of nameplate. Something seems strange about this site's power output; the site has an artificial limit at ~83% or so of nameplate generation. This is probably due to an undersized inverter which is unable to produce more power than its nameplate despite the solar cells being capable.

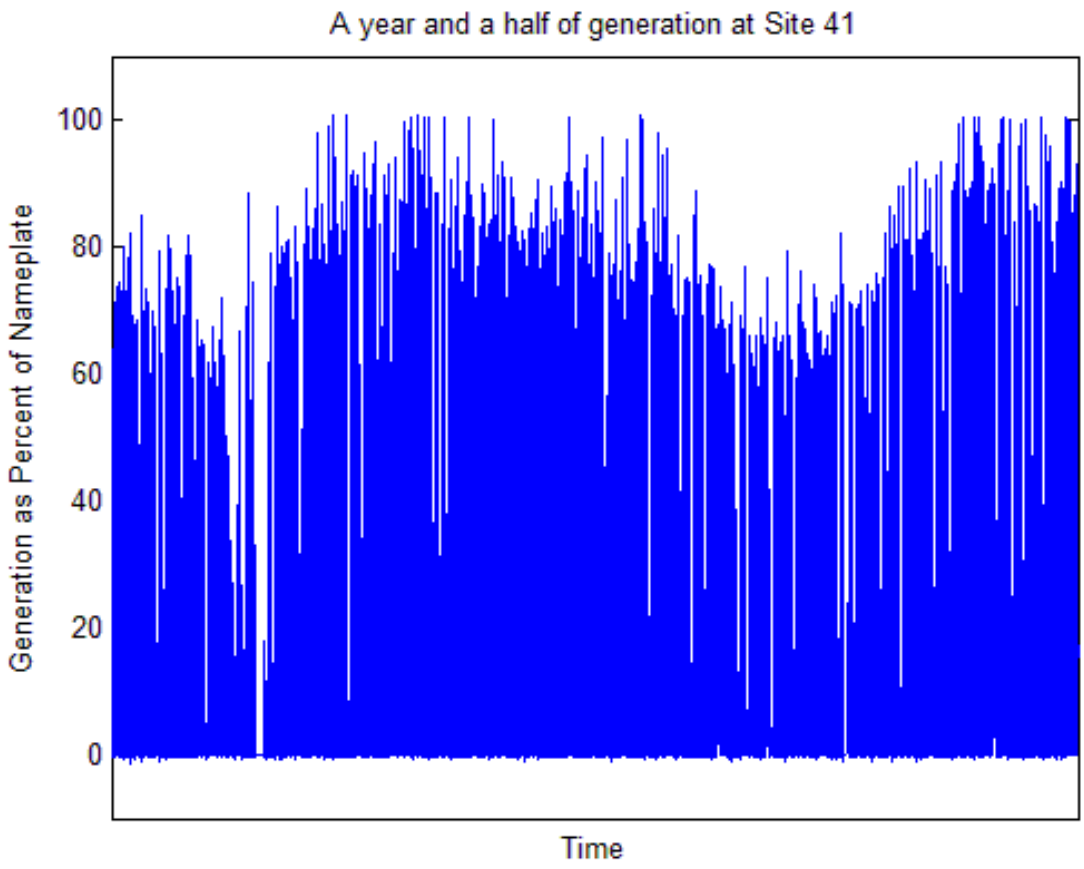

Figure 15: Site 41's power over time as a percentage of nameplate. We can see here that the site varies over the course of the year (lower generation in winter, higher generation in summer). We can also see that it appears to not be artificially limited in any way. This is exactly the type of power graph that we would expect from a 'normal' solar PV installation.

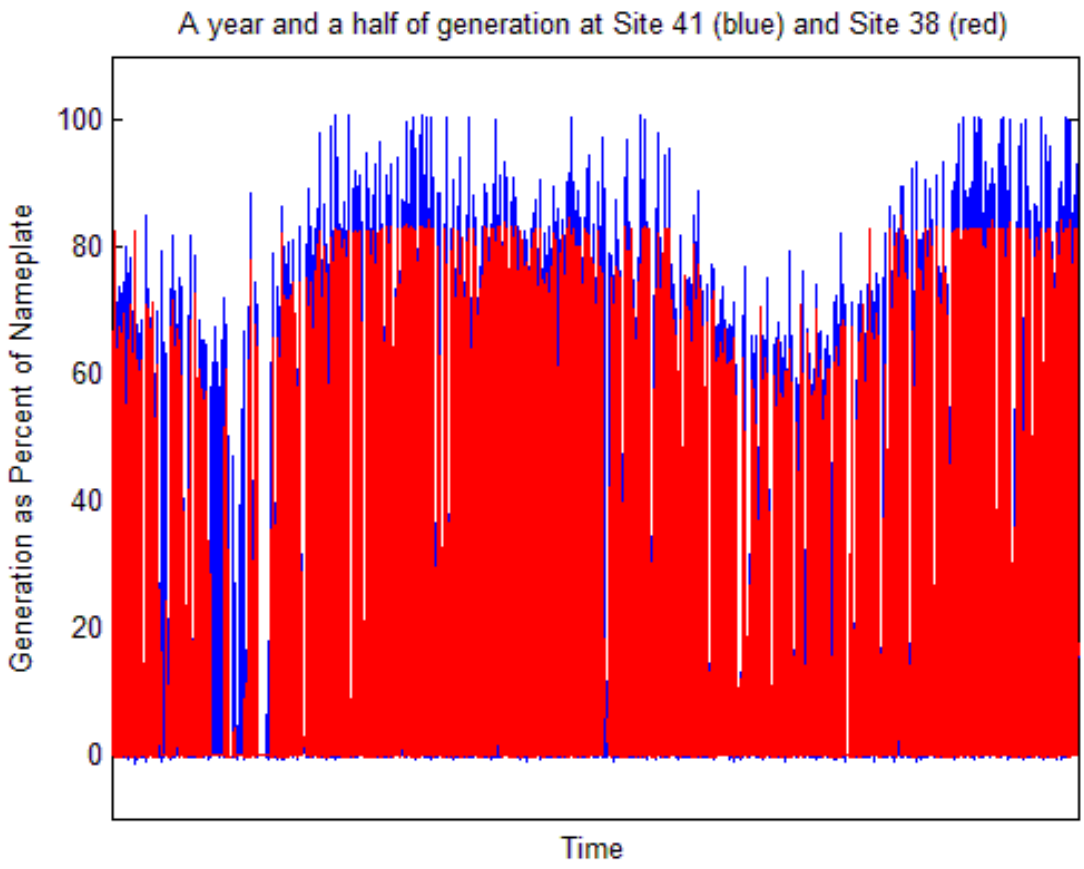

Figure 16: Overlay of Site 38 and Site 41 production data. Notice how the sites track each other more or less through the year, but for a good portion of the year, during the peak solar season, Site 38 caps out at ~83%.

In conclusion, we can see that if their site was not limited by an inverter (or other) rating, their site could produce more power with the same amount of panels and other materials. This may or may not be desired for the gain that they would receive from those peaks.

Example 2:

This example shows three sites with similar locations, but again with different energy outputs over time. We will try to determine what can be done to improve these sites.

Let's take Site 20 as an excellent performing site. Site 21 and 35 are nearby. Table 2 shows the capacity factors of the sites:

Table 2: Site # vs. Capacity Factor

Site #	Capacity Factor
20	20.98%
21	19.32%
35	16.77%

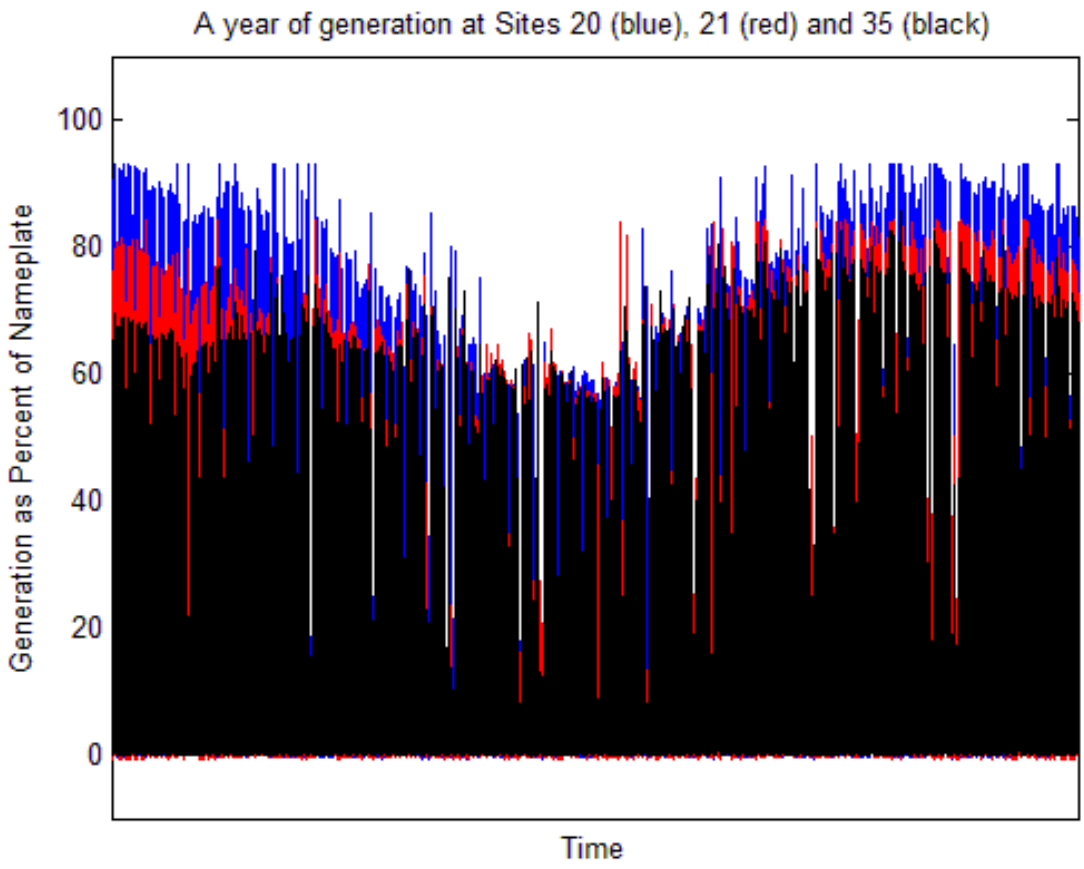

Figure 17: We see a year's worth of generation at Sites 20, 21 and 35. We can see immediately that Site 20 outperforms Site 21, and significantly outperforms Site 35. Notice that during the Winter (low) months, all the sites seem to be producing similar amounts of power, yet during the summer months, Site 20 outperforms Site 21, and significantly outperforms Site 35.

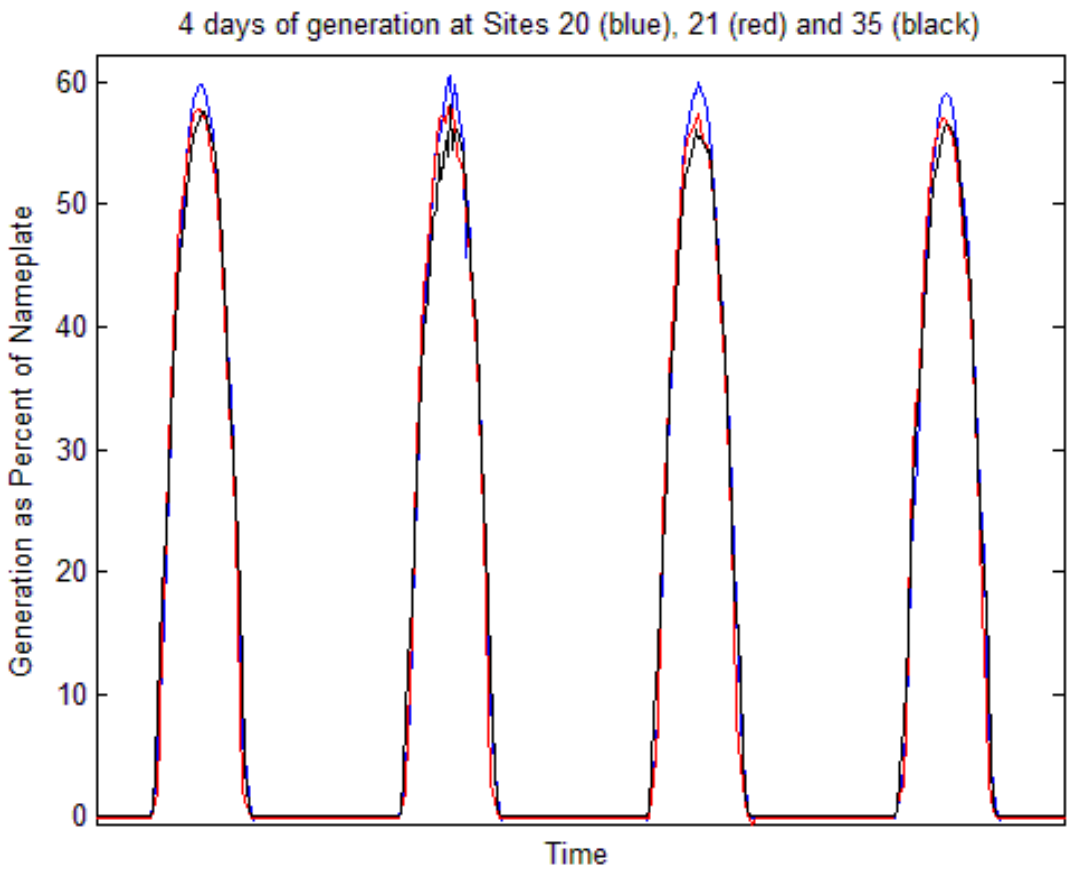

Figure 18: Close-up of Winter production sample week for the three sites. Site 20 is consistently out producing sites 21 and 35 during winter, however marginally.

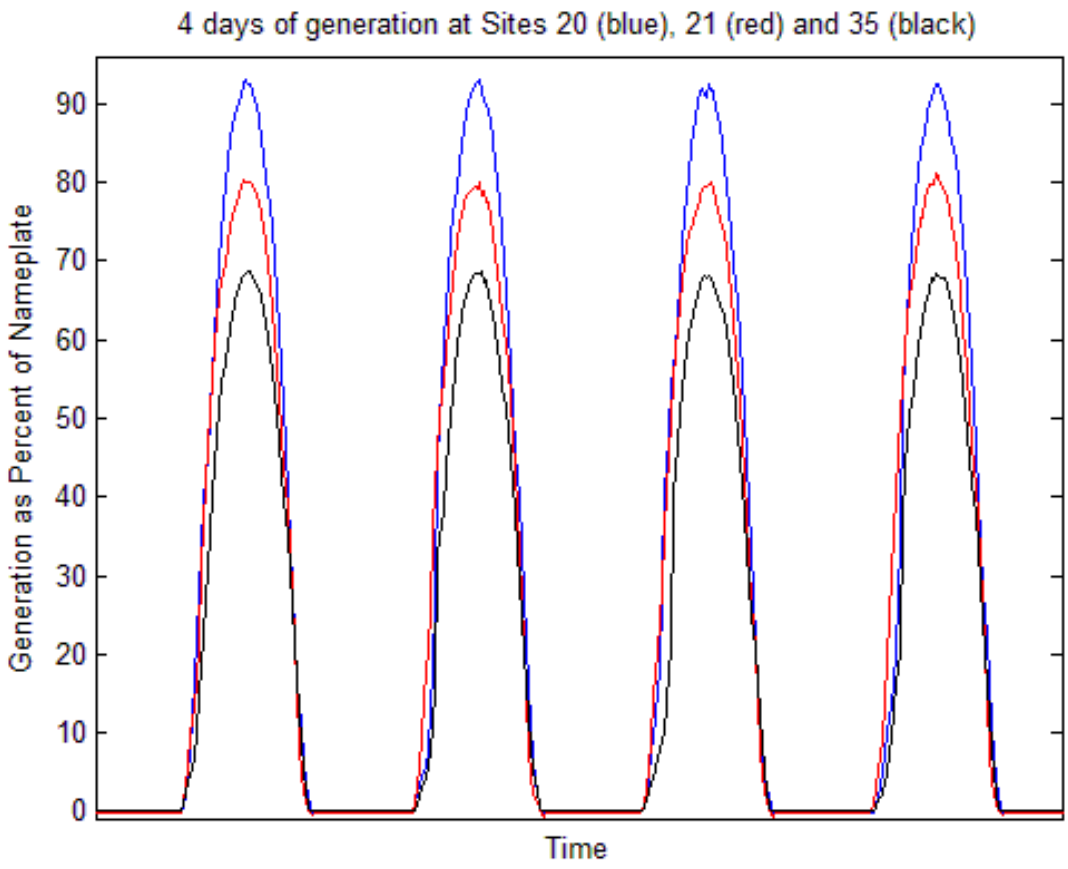

Figure 19: Close-up of Summer production sample week. Site 20 outperforms both Site 21 and Site 35 by a large margin. Site 21 also handily outperforms Site 35.

Table 3: Inverter and solar panel information for the sites listed in this section

Site #	Inverter Manufacturer	Inverter Model	Solar Panel Manufacturer	Solar Panel Model
20	SatCon Technology	PVS-110-S-MT (480V)	NexPower Technology	NH-100UX 3A
21	SatCon Technology	PVS-75 (208 V)	Kenmec Mechanical Engineering	TKSA-23001
35	SMA America	SB7000US (208V)	Powercom (70%)	PPV-230M6L
		SB4000US (208V)	Green Energy Technology (30%)	GET-090A
		SB6000US (208V)		

Due to the performance being so different during the Summer vs. the Winter, we propose a hypothesis that this might be due to differences in how heat affects the panels and/or the inverters.

Table 4: Heat characteristics of the panels used:

Panels	**Technology**	**Performance decrease per °C of cell temperature increase**
NH-100UX	Thin-Film	-0.200%/°C
TKSA-23001	Polycrystalline	-0.459%/°C
PPV-230M6L	Polycrystalline	-0.509%/°C
GET-090A	Thin-Film	-0.259%/°C

In this area of the country, the daily high temperatures can swing between Winter to Summer by about 30°C. However, this is not the only way in which solar panels heat up. The losses incurred through the PV conversion process are lost as heat – so the actual cell temperature is generally significantly higher than the ambient air temperatures. During Winter, not only are ambient temperatures lower, the angle at which the panels are relative to the sun are such that there are less photons incident upon the panel (which in turn means less heat losses). As a result, cell temperatures in Summer are generally higher than the ambient temperature difference would suggest.

The temperature is especially relevant since the performance of a solar cell degrades as it increases in temperature.

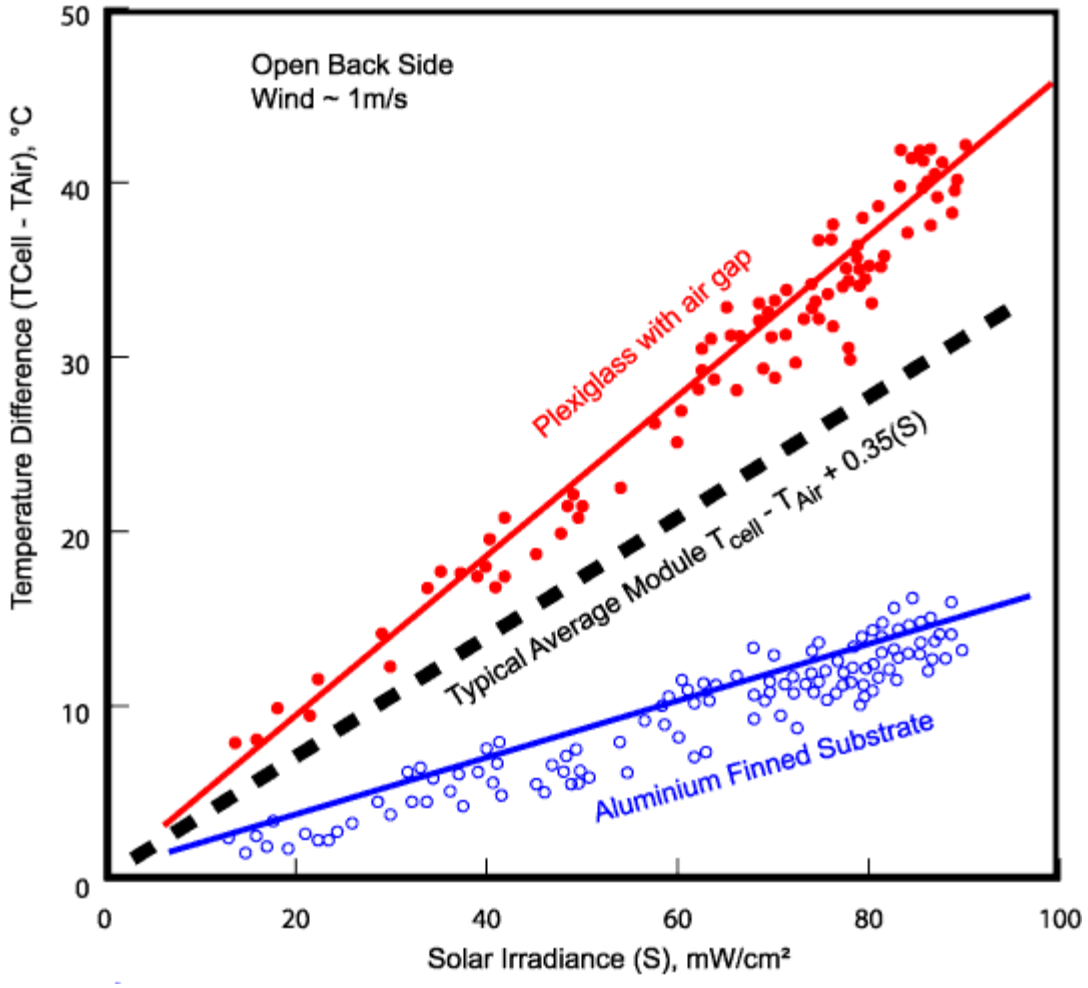

Figure 20: Demonstration of the difference between cell temperature and ambient temperature vs. irradiance for different types of solar panel constructions (PV Education).

This is not the only effect however. As we can see from Figure 20, the design of the panels themselves also contributes significantly how hot the cells get relative to the ambient air temperature. 'Plexiglas with air gap' construction has the cells essentially insulated by an air gap, so they will get hotter than aluminum-finned substrate

construction where the cells can radiate heat via the aluminum heat sink fins connected to the back of the cells.

And finally, there is yet another factor that can account for large differences between the solar panel cell temperature and the ambient temperature. The orientation of the solar panels can be such that the racking is installed flush with a tilted roof; this will enable the heat to rise and the air will circulate. If the air circulation is poor, the cell temperature can rise significantly more. Also, it makes a difference for air circulation if the panels are installed in a field four feet off the ground, or on a residential roof, less than 6" away from the roof.

The best way to measure the performance decrease would be to have a cell temperature sensor mounted to the back of multiple solar panels in the site. However, we do not have this data for any of these three sites. As such, we can only take an educated guess as to the magnitude of the difference between the cell temperatures and the ambient temperatures. If we use Winter temperatures as a baseline, and 25°C as a heating baseline (below which heat degradation does not play an effect), we arrive at the following calculations:

Site 20:

Summer Ambient – 25°C ~= 15°C

Cell temperature at 80mW/cm^2 irradiance = Ambient + 28°C = 43°C

Performance degradation:

$$43°\text{C} * -0.2\ \frac{\%}{°\text{C}} = -8.6\% \quad (1)$$

Expectation based on installation: Installed on a commercial style flat roof with good spacing on tall racking.

Site 21:

$$Summer\ Ambient - 25°\text{C} = 15°\text{C} \quad (2)$$

$$Cell\ Temperature\ @\ \frac{80mW}{cm^2}\ irradiance = Ambient + 28°\text{C} = 43°\text{C} \quad (3)$$

$$Performance\ Degradation = 43°\text{C} - 0.459\ \frac{\%}{°\text{C}} = -19.737\% \quad (4)$$

Expectation based on installation: Installed on a commercial style flat roof with good spacing on tall racking. The only difference we'd expect to see is the effect of the higher cell temperature coefficient. We wouldn't expect to see much of a difference in winter since the cell temp would only briefly get into the range where derating is even necessary.

Conclusion: The actual performance difference during peak heat times of the year are about 10%. In Winter, there is minimal difference. This is almost exactly in-line with our expectations of ~11%.

Site 35:

$$Summer\ Ambient - 25°\text{C} = 15°\text{C} \tag{5}$$

$$Cell\ Temperature\ @\frac{80mW}{cm^2}\ irradiance = Ambient + 28°\text{C} = 43°\text{C} \tag{6}$$

$$Performance\ Degradation = \tag{7}$$

$$43°\text{C} - {}^{(70\% * 0.509 + 30\% * 0.259)}/_{°\text{C}} = -17.65\%$$

Expectation based on installation: Installed on carports. Part of the installation (it looks like the 30% piece from the picture) faces southwest and the other part is installed facing southwest. We would therefore expect lower production during summer when there is more direct sunlight, and not as much of an effect during cloudy days and/or winter when there is more diffuse light. Also, the racking is installed on a slanted carport with minimal spacing (less than 6 inches). We would therefore expect there to be more of an effect of heat during the summer.

Conclusion: During Winter, we see minimal differences because there is more diffuse light (so the orientation angle matters less), and the cell temperature does not increase sufficiently to require derating (so the airflow problems are not as impactful). However, during Summer, we see not only the effect of the worse cell temperature coefficient, but the lack of airflow also increases the cell temperature and decreases output by an additional ~5%. Working the numbers backwards, this means that Site 35's cell temperature would have to be 14.5°C above Site 20/21. This seems like a very reasonable temperature difference. This example demonstrates how the data may be examined to diagnose problems with PV systems.

4.6 What is the dollar amount of an improvement at each site?

Example 1:

Let's consider two hypothetical adjacent 100kW sites for the purpose of analyzing revenue and system upgrades. A key note is that we are not factoring in days of extended downtime, so the sites are not being penalized for this downtime. Obviously, a site with an inverter down 10% of the time will have a significantly lower capacity factor than a site with an inverter down 1% of the time. These are the range of typical inverter downtime values.

The capacity factor of Site 38 is 13.73% while the capacity factor of Site 41 is 14.93%. If we analyze the portion above 83% of nameplate, we can see that this accounts for 0.39% of the total power for Site 41. If we add this to the Site 38 output, we would raise the capacity factor from 13.73% to 14.12%.

Let's assume that this site is a 100kW site selling power at 20 cents/KWh. This would mean that the annual gain in production would be $683, or 3416KWh. If the owner wants 10 year payoff for improvements, this would limit their budget to $6830. This would be enough to perhaps install an additional small inverter. A 20kW inverter at today's prices would cost around $10,000 just for the inverter itself (not counting installation and possible foundation work), so most likely, it would not be worth the cost to modify the site.

Looking at this from another perspective, Site 41 has an inverter that, if analyzed in economic terms, may be oversized. An additional 20kW of capacity at today's prices costs around $10,000, so they may have been able to buy a slightly smaller inverter, and just accept the loss of power generation during peak hours at peak times of the year. This

may have been the economical choice. Inverters are sized like this in less than 10% of the sample data. This represents some cost savings which could be incorporated into future designs – although it is too late for Site 41.

These types of analysis, as well as many other types, are enabled by having a dataset where the user can easily access the capacity factors and power data of nearby sites.

Example 2:

Consider three 100kW sites selling power at 20 cents/KWh.

Recommendation for Site 20:

Site 20 is a star performer with a capacity factor of 20.98%. Its annual revenue is $36,757.

Recommendation for Site 21:

Site 21 is also a relatively good performer with a 19.32% capacity factor, with an annual revenue of $33,848. The only thing to say with regards to this site is that if they had used panels with a lower cell temperature coefficient such as Site 20, they could get the capacity factor up to those levels. However, after installation, it is obviously too late to be switching solar panels.

Recommendation for Site 35:

Site 35 is not such a great performer with a 16.77% capacity factor, with annual revenue of $29,381. It's too late at this point to switch solar panel models, but if the airflow problem could be improved, as much as 5% additional production could be gained. This has a dollar value of ~$2500 a year. Perhaps a forced air solution could be explored for this price.

Conclusion

We see here the value in having the ability to analyze data from nearby sites, as well as having a larger dataset to pull from in the background to make suggestions on site improvements.

Location-based correlation algorithm for false alarm reduction

There are a variety of solar alarms which are used in industry today. The most basic idea behind the alarm is that a given customer cannot monitor every site by themselves. Many customers have dozens of large sites, and they do not have anyone on staff who's job description is solely to monitor sites for performance issues. By having alarms, it saves the customer time that could be better spent doing other tasks. Also, there are some alarms which are inherently impractical for a customer to do on their own (on-the-run regression analysis to detect long-term performance issues for example).

One of the simplest alarms is the so called 'performance alarm' whereby an arbitrary threshold and time range can be set such that if the site does not produce above that threshold during that time period, an alarm would be triggered. Figure 21, Figure 22, and Figure 23 below are examples to illustrate when the alarm would and would not be triggered.

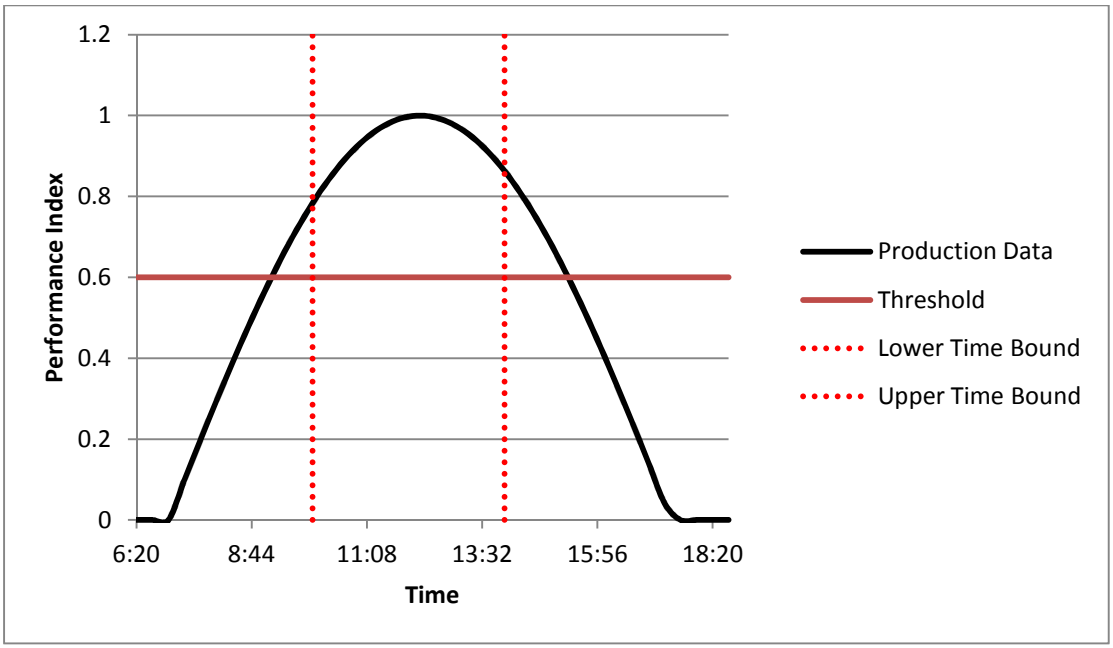

Figure 21: Normal operation, so no alarms are triggered. The alarm should trigger whenever the performance drops below the 'Threshold' line between the two dotted lines. In this case, the alarm would never trigger since the Production Data is never below the 'Threshold' line during the time span defined by the two dotted lines.

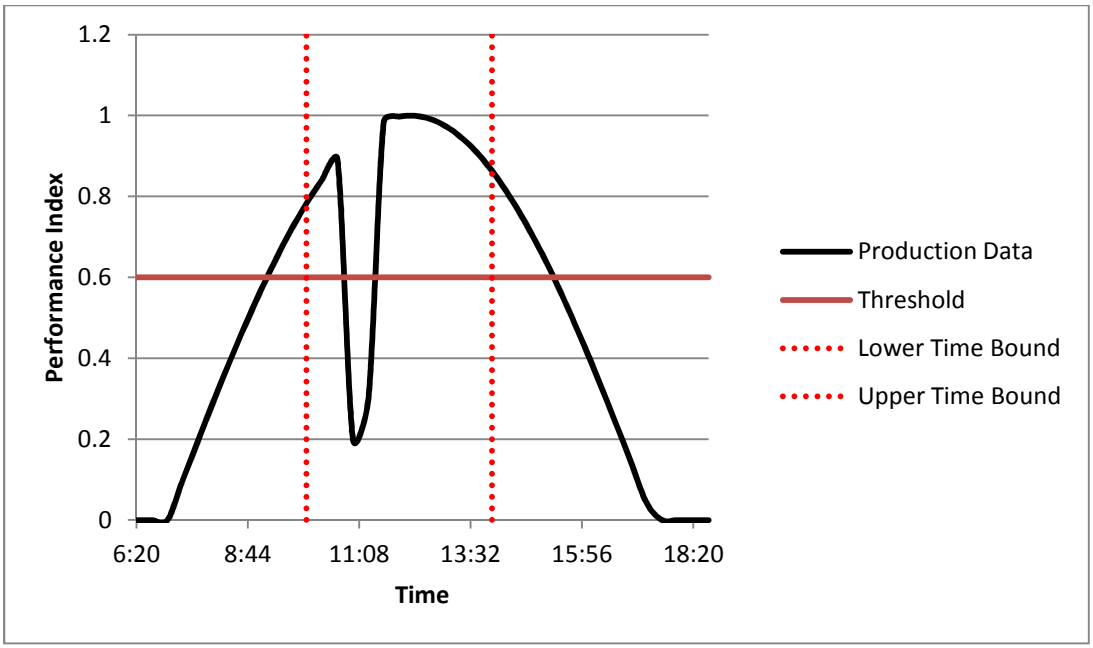

Figure 22: Performance decreased below the threshold during the defined time period, so the alarm is triggered. In this case, the alarm is triggered because of the brief excursion that the production data makes into the area beneath the threshold curve. It will trigger every 15 minutes during this time.

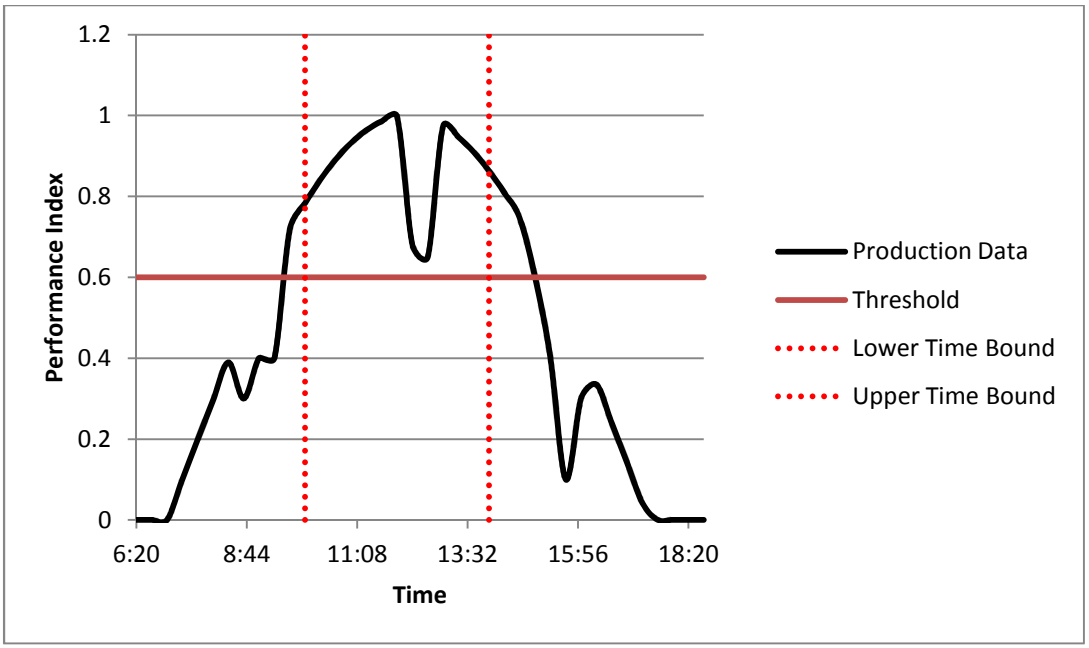

Figure 23: 1) Low production outside of alarm range. Alarm is not triggered (since it is outside of time range). 2) The performance decrease during the defined time range is also not detected since it is not below the threshold.

One of the benefits of this alarm is that it does not require irradiance data in any form. If for any reason at all, the site drops below the threshold for generation, the alarm will be triggered. This method is fairly crude. However it is easy to understand and implement.

This alarm has problems for both selectivity and sensitivity. These problems are very interrelated. The main problem with these types of alarms is that if a cloud rolls in, it can decrease the output of the site dramatically. Unfortunately, the odds of a single cloud being over the site at a given time during the day is very high, so this alarm triggers very often.

5.1 What is the false positive rate of the basic algorithm?

Consider a performance alarm with parameters as follows:

Threshold: Trigger alarm if current production is under 50% of nameplate

Time range alarm is active: 10am-2pm

Note that the threshold for performance decrease is 50% of nameplate. This is by no means an aggressive threshold, and the time range is only 4 hours from 10am to 2pm, which are peak generation hours (typically). Even with this very lax set of alarm criteria, we find that at some point during the day on 72% of days across sites, an alarm is triggered. We find by manually looking through the data that the vast majority of these alarms are false positives. This is unacceptable for customers, and defeats the entire purpose of making it unnecessary for customers to look at their site production data on a daily basis.

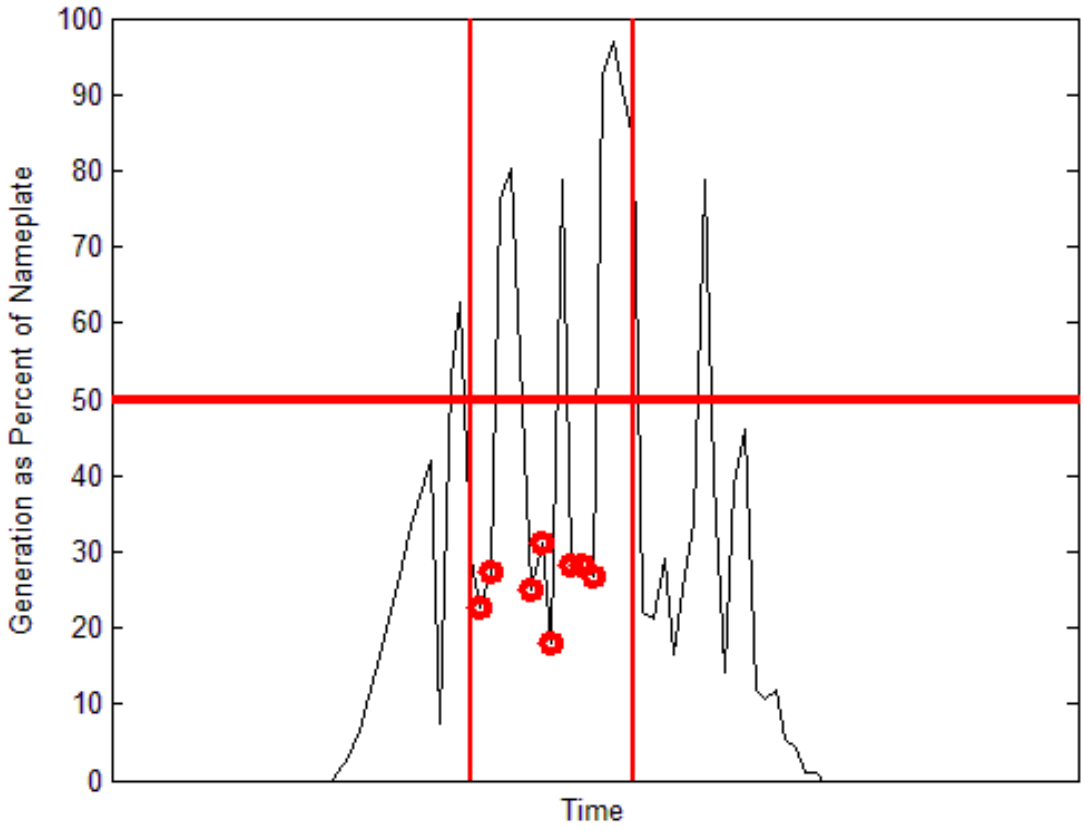

Figure 24: Verification (red dots) of alarm trigger points being correct for a given site. Each point below the threshold between the two alarm bounds triggers an alarm. These points for which an alarm is triggered are noted with red circles. Notice that they only occur in the correct alarm area. Points for which the performance is below the threshold, but not within the time bounds are not counted as alarm points.

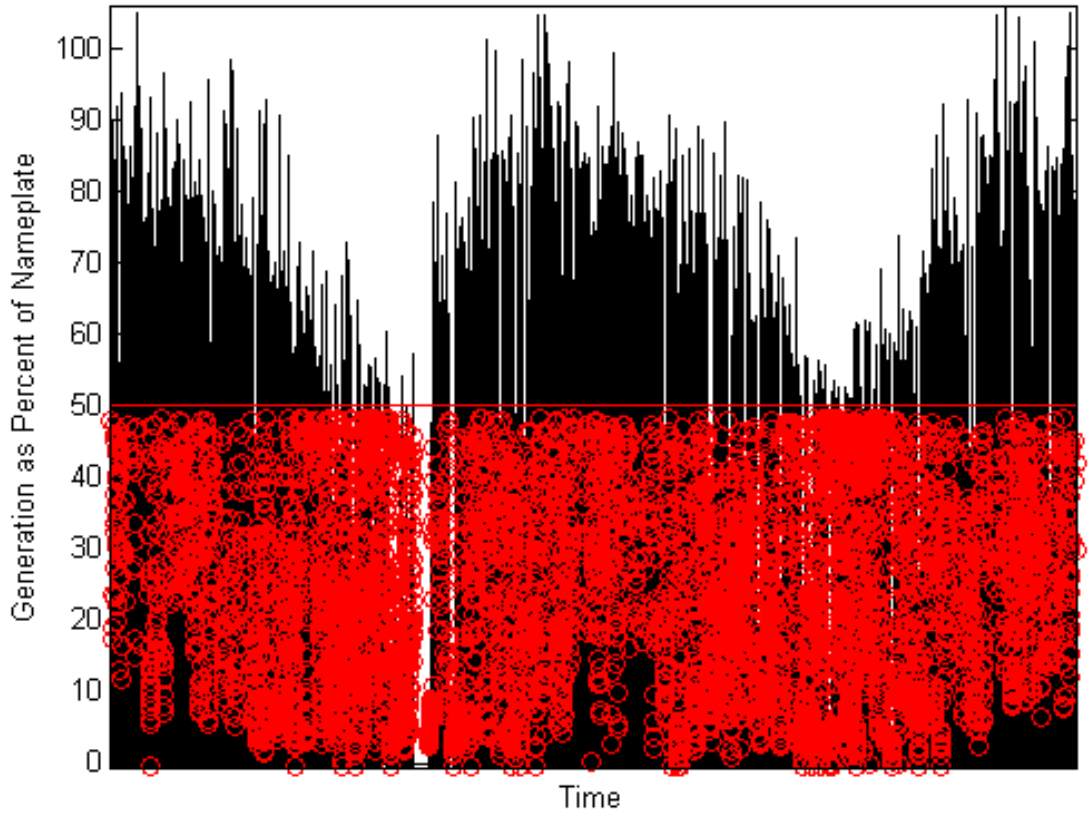

Figure 25: Verification that the alarm does in fact trigger correctly (no triggers above 50% nameplate). Notice all triggers stop at 50% of nameplate (denoted by the red horizontal line).

5.2 Algorithm improvement description

Our original hypothesis is that the main cause of the intermittent performance problems is cloud cover. One thing to note about clouds is that they are generally part of storm fronts of other larger weather patterns. There are not as many sunny days where there is literally just one small cloud floating in the sky passing above a single solar array. From this hypothesis, we can deduce that if a given site has a performance

decrease, nearby sites should also have performance decreases.

We first must test to make sure that our hypothesis is actually correct. The following sites show the correlation in time of several nearby, correlated sites. We can see that on a sunny day (Figure 26), all sites show good smooth performance. On a cloudy day (Figure 27), we can see that the sites are all correlated as well. On a partially sunny, then partially cloudy day (Figure 28), we can again see that the sites are all correlated as well.

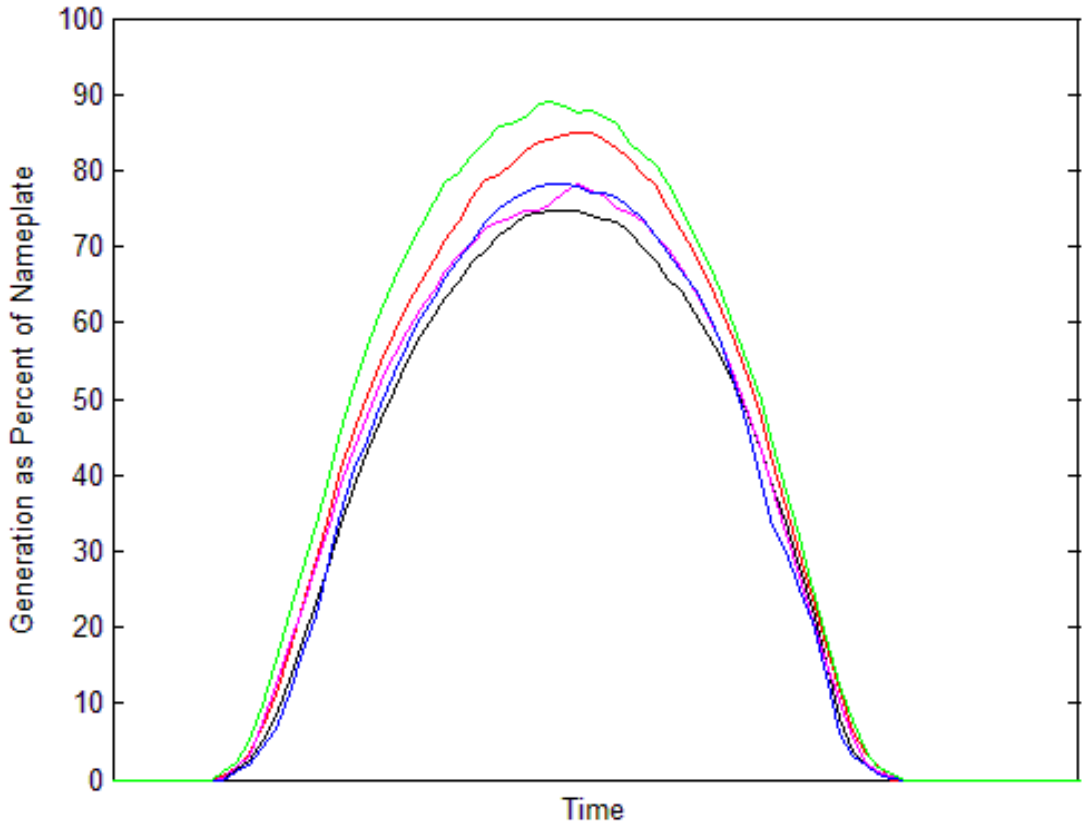

Figure 26: Performance profile of a given sunny day for nearby sites. They are all fairly correlated – although not perfect.

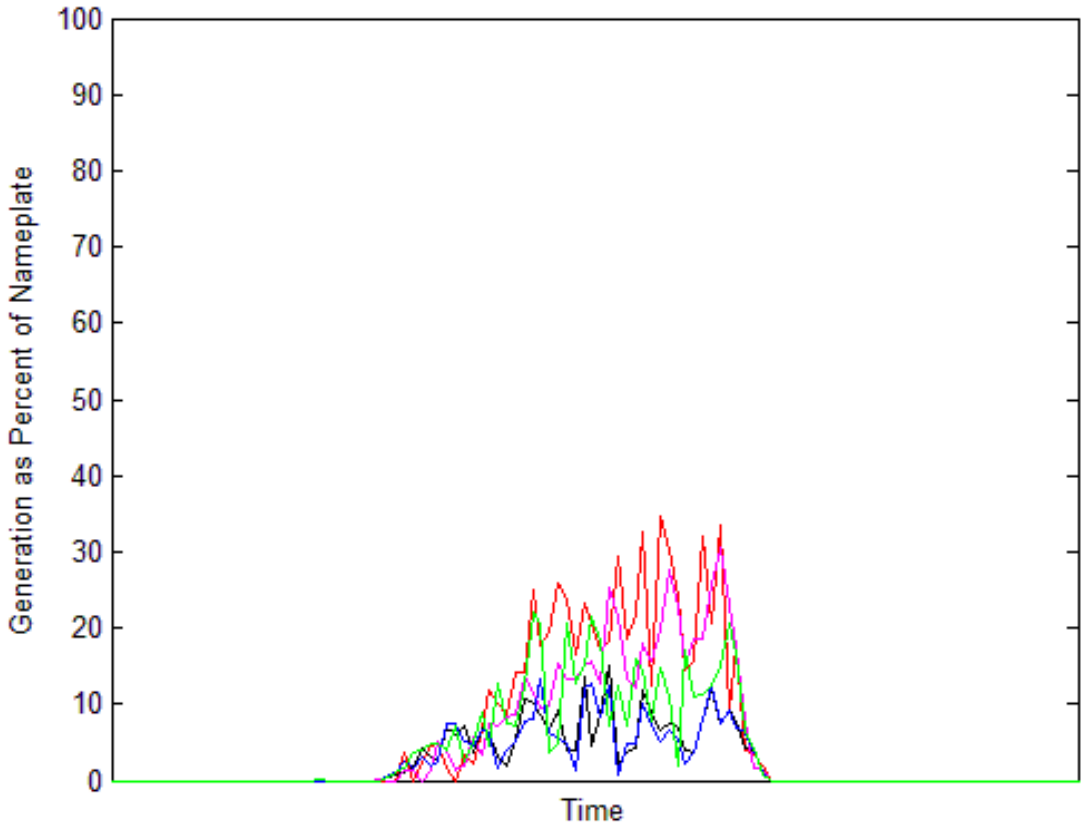

Figure 27: Performance profile of a given overcase day for neaby sites. This is an example of sites which previously all would have tripped the performance alarm every 15 minutes, all day.

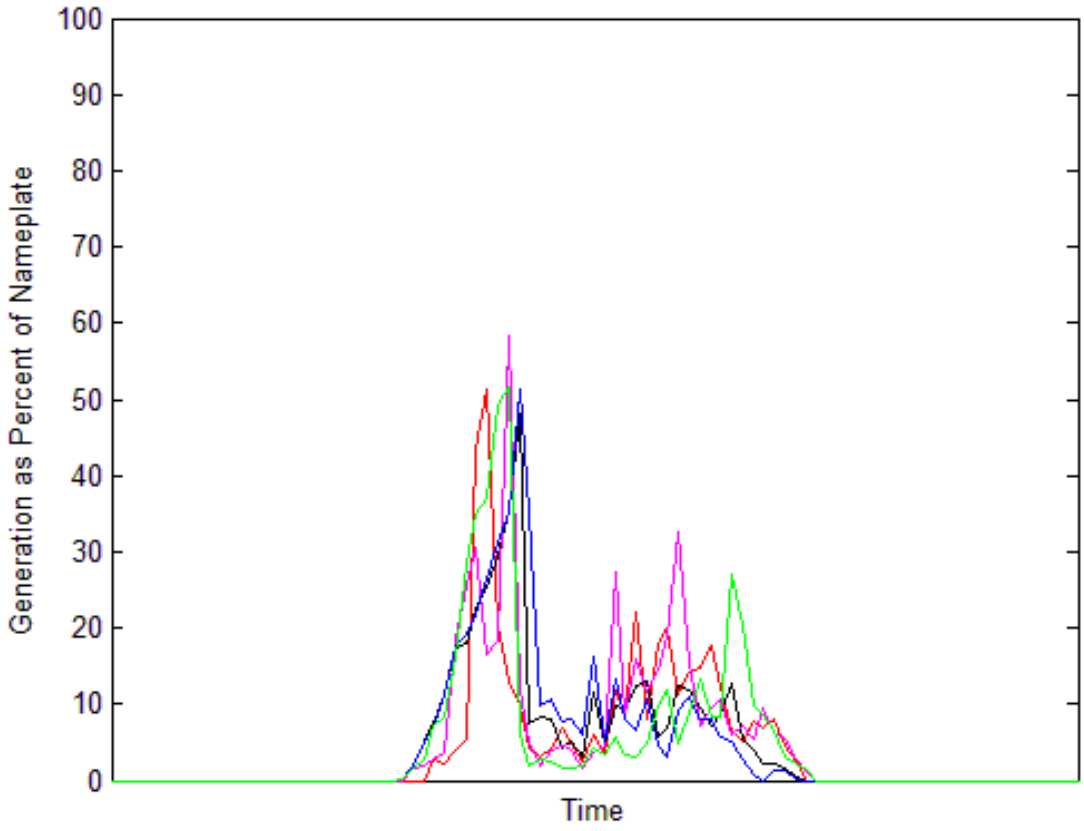

Figure 28: Performance profile of an intermittently cloudy day for nearby sites. We see that during the day, when performance does down on one site, the rest more or less follow. We do notice some minor variations, however it is close as a whole.

From this idea, we can draw a square of a given dimension around the site (we choose a square, and not a circle due to complexities in calculating circle dimensions in lat/long, which is actually fairly complicated and unnecessary for our initial proof-of-concept purposes).

In order to determine the size of the square that we will draw around the site, we want to correlate production at various sites. In Figure 29, we pick a given site and correlate

the production data with those of the neighboring sites and plot the correlation coefficient based on the last 30 days with respect to distance. We do this for every site and plot the results, Figure 29.

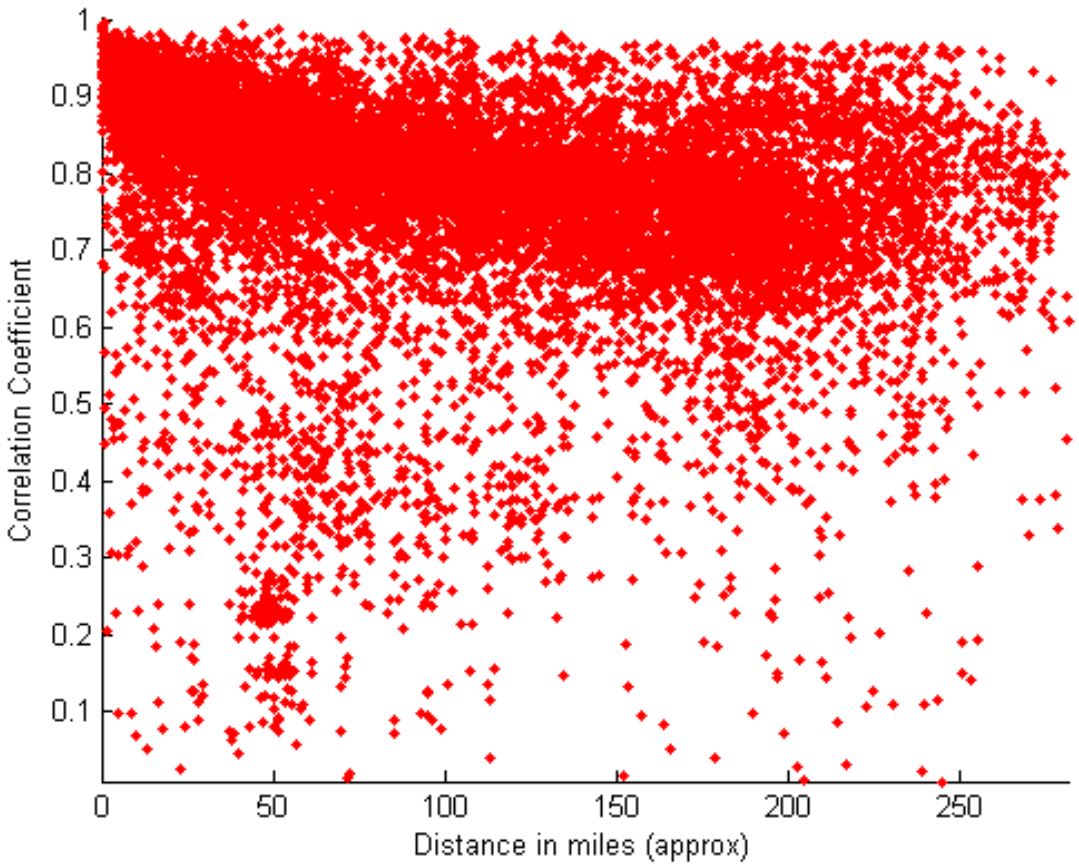

Figure 29: Correlation coefficient vs distance for sites in the database.

Correlation coefficients range from 1 (perfect correlation) to -1 (perfect inverse correlation). The selection of a theshold is somewhat arbitrary, and depends on the application.

There are three main takeaways from Figure 29:

1. The correlation coefficient goes down as distance from the site increases as expected.
2. The correlation between sites is not determined completely by distance from the site. There are very close sites which are in the 0.5 correlation range. It is important to note that not all sites that are nearby are correlated sites. For example, one site could be at the base of a hill, 2 miles away, and perhaps get more clouds than the site on the top of the hill. Also, a site could be 10 miles away, but be on very flat terrain with very uniform weather and be highly correlated.
3. Correlation numbers are subjective. A 0.8 number may be very significant for some fields, and a 0.8 may be very inadequate for other fields. For our application, we have chosen to set the correlation coefficient to be very high (~0.9) in order to ensure good matches.

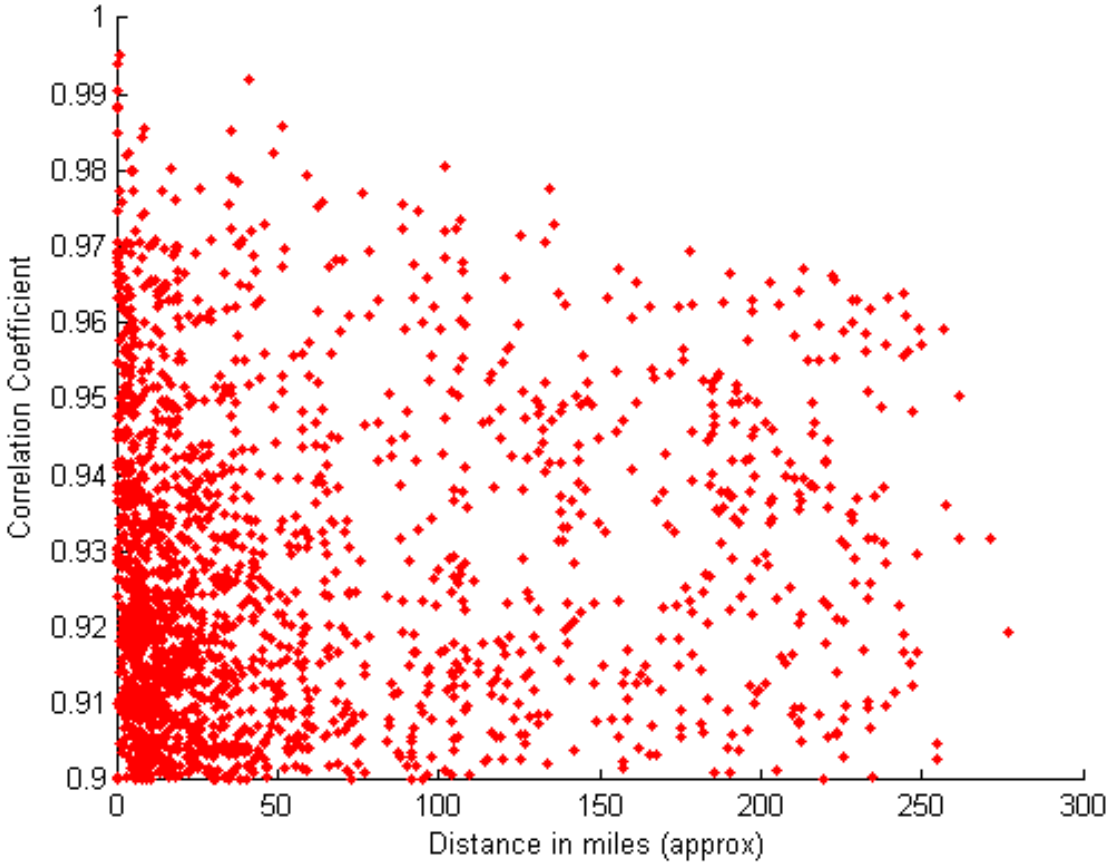

Figure 30: Close-up of Figure 29 showing only sites with correlation coefficients of 0.9 or higher vs distance.

From examining Figure 30, we deduce that the square that we should draw is roughly 35 miles by 35 miles. Beyond 35 miles, there are fewer highly-correlated sites. However, if we draw the square too small, we may find that there are no sites at all within that area if the site is located in an area without many solar sites.

Due to the relative abundance of correlated sites, we decided to set our correlation coefficient threshold to 0.9; sites with correlation coefficients of less than 0.9 are not considered correlated with the site in question.

To make sure that we are not throwing out too many sites, we verify whether each site had at least one correlated site. 73% had at least one correlated match. If the correlation coefficient minimum was lowered 0.8 (from 0.9), then 90% had at least one correlated match. It's very likely that there are a lot of sites out there without sufficient neighbors – correlated or not.

Algorithm pseudocode for a given site:

Find sites that are within the distance specified

Out of those sites, find the sites that have correlation coefficients above 0.9

Pair the sites up (associate them with each other in the data structure)

5.3 What is the improvement provided by the location-based correlation addition to the performance alarm?

Overall, we verified that the current algorithm throws an alarm on 72% of days. For sites with nearby correlated sites, false positives are down to 12%. For sites without nearby correlated sites, the results from the new algorithm are identical to those of the old algorithm. When these sites are factored in, there is at least one false positive on 24% of days overall.

With a reduction in false alarm rate of 83% for sites with correlated sites nearby, we find that this alarm goes from a nuisance alarm to a useful alarm. Instead of the alarm going off every 15 minutes all day for more than two out of three days, it now goes off a

couple times in one day out of two weeks.

5.4 What is one way in which this alarm could be further improved to produce fewer false positive alarms?

This description does not tell the whole story. The findings show most sites that are triggering this alarm are triggering them due to very momentary gaps in production – say from an isolated cloud.

If we apply a smoothing function to the data, let's say two sample periods, we could remove these momentary 15 minute lapses in production. A small cloud would be seen typically by only one site. This would further decrease the false positive rate. The estimate for the false positive rate would be under 10%, and potentially under 5% based on what was seen from the data. The tradeoff would be to reduce reaction time to these incidents from 15 minutes to 30 minutes, but this is not significant compared to the signal-to-noise ratio improvement.

This transforms the alarm from one that would alarm on more than two out of every three days, to one that would alarm once every two weeks or less.

Augmented Predicted Power algorithm

A Power Purchase Agreement (PPA) is one type of subsidy program. There are varying types, but in general the idea is to not give a subsidy per watt installed, but to instead give a subsidy per watt-hour of energy produced. For many installers who install PPA systems, their contracts can be tied to the predicted power algorithm if they are not also the final owners.

If the installer is using such a contract and the power that is predicted is higher than the actual, they could get penalized. However, this is not the typical application of this type of algorithm.

A more common application of a predicted power algorithm would be to see the predicted power and use this information to understand whether or not the site is performing like it should. For example, if we have a site that, according to the predicted power algorithm, should be producing 10MW but it is only producing 3MW, we can infer that there may be a problem with the site. There are a variety of predicted power equations out there, but below we have a basic example:

$$Predicted\ Power = NI(I - CT) \quad (8)$$

Individual variables are described in detail below.

(N) Nameplate

N = Nameplate in DC watts. This is the peak power that the site will generate under standard test conditions (for whichever applicable standard – CEC PTC, PTC, etc…)

Standard method for obtaining: Nameplate is a constant that can be retrieved from the design drawings.

Flaws: Nameplate data from datasheets are not the nameplate of the actual individual panels. No two panels are exactly alike. The nameplates should ideally be estimated from production data. That being said, many panels are guaranteed to be within +/- 5%, and many panels are "plus-sorted," meaning they are guaranteed to be within +0 to +5%.

One of the key missing components of this equation are the losses inherent in the system due to cabling, terminations, inverter losses and other areas. For example, if the installer uses 120ft of #10 wire, and an inverter with an efficiency rating of 96.3%, then the total efficiency could be 95%. In practice, if the wire varied from its datasheet slightly, and the inverter efficiency varied slightly, efficiency could range between 94-98%. There are other unknowns, such as how much the installer torqued down the connections, or if part of the wire is in sunlight, which increases its resistance for that segment.

We propose that not only is it very complex to come up with estimated efficiency ratings, but it is impossible to predict what will actually happen in the field better than using a month (or more) of actual collected data.

We have meter data, so we can determine the 'actual nameplate' measured at the meter. This will be different than the DC nameplate rating given since we would be accounting for losses in this method. This method also removed the human error associated with entering nameplate ratings – some of which we found were very significantly inaccurate (100+% error).

(I) Irradiance

I = Irradiance, measured in [W/m2]

Standard method for obtaining: Irradiance is a feed which can be taken from one or

multiple weather stations or satellite feeds. We get ours from weather station data.

Flaws: Satellite feeds of irradiance data can have problems. For example, snow on the ground can reflect sunlight back up and cause erroneous readings at the satellite. Also, satellite feeds are generally given as one value for a 40km by 40km square (dimensions of this square vary). As such, it is very possible that on a sunny day with intermittent clouds, it is very sunny on one side of the square, and completely shaded and raining on another side. As such, the resolution is not very precise.

Weather station feeds are far more precise for the area. However, they have their own problems. Many customers do not install weather stations. The weather stations that are installed are typically very inaccurate, with up to +/- 20% error. This presents a challenge as this is one of the key components of the predicted power equation. However, it has been our experience that the variation in irradiance sensors is not as extreme as the manufacturer's datasheets suggest.

That being said, we estimate irradiance vs. actual power output in this thesis as well.

(T) Cell temperature

T = cell temperature in °C

Standard method for obtaining: Cell temperature is a feed which can be taken from one or multiple weather station sensors.

Flaws: Where the cell temperature sensor is located is of prime importance. If the sensor is located at the bottom of a slanted installation, due to airflow, it will have a lower temperature than the panels at the top of the rack. Also, for a large scale solar installation spanning multiple acres, the temperature and amount of sunlight at one spot

could be significantly different than the temperature at another location. However, in general, these sensors are installed in representative locations. The cell temperature sensors themselves are actually fairly accurate, and are probably one of the most accurate sensors on a standard, (relatively) low-cost weather station.

The combination of these two points means that we will not be concerned with their inaccuracy in this thesis.

(C) Coefficient of power output dependence on cell temp

C = coefficient of power output dependence on cell temp (%/°C)

Standard method for obtaining: The coefficient of dependence on cell temp is a constant that can be retrieved from the datasheets of the solar cells.

Flaws: Manufacturer warranties are typically tied to both nameplate and the output's dependence on cell temperature not changing more than a certain percentage over time. As such, the output dependence on cell temp listed in the datasheet is more of a worst-case coefficient – not a mean of the panels produced. We hypothesize that this means that the cells may vary significantly – and not vary about the value given by the manufacturer.

We estimate this value from the data instead of taking it from the datasheet. This value is impossible to obtain through any other means. This also has the benefit of removing the human error associated with entering this value. There were significant errors associated with this value, spanning from $1/10^{th}$ of the value to 10x the value.

Important Note Regarding Regression Model

It is technically incorrect to say that we will be estimating the coefficients and correlations as individual units. One of the key problems with estimating the various components of the model is that of co-linearity.

Co-linearity is what happens when inputs to the system are correlated with each other as well as the output. Take Cell Temperature and Irradiance for example. If we make a model using only a coefficient for cell temperature, we will find that as cell temperature increases, power output increases as well. We could therefore conclude that we should heat the solar panels in order to get higher output.

One big problem with this assumption is that cell temperature increases as irradiance increases – so we must set up a multi-variable regression model. As such, we do not estimate singular coefficients for each component separately. We set up a coefficient matrix to estimate the output with a higher degree of accuracy – taking into effect the co-linearities of the input data.

The algorithm uses ambient temperature, cell temperature feeds, and irradiance data feeds to find the coefficient matrix for the regression model. The algorithm uses the Partial Least Squares (PLS) Regression algorithm, which is built into MATLAB's statistics toolbox.

6.1 The proposed predicted power algorithm

The algorithm for the coefficient estimation is as follows, in psuedocode[2]:

Find all sites with non-zero irradiance and cell temperature

Remove sites that have less than 100 data points (Some are very recent)

Remove all datapoints that are zero or below (this removes night generation, which we are not concerned with).

Calculate out original predicted power using these values.

Fit the data to a multi-variable PLS linear regression model with cell temperature as the variable.

If the data points are below 0 for the new algorithm, set them to 0. Obviously, less than 0 generation makes no sense.

Compare the MSE of the original model with the MSE of the new model

[2] One critical advantage of the new algorithm is that it does not require any information about the nameplate of the site. It auto-detects and configures appropriately. This is because it finds the actual peak power production of the site and sets that to the nameplate. This can be more reliable than relying on customer-provided nameplate information.

6.2 How accurate is the original predicted power algorithm?

Table 5: Mean Squared Error of the original predicted power formula

Site #	Original Predicted Power (kW) Formula MSE	Average Percent Error
1	12.7460	22.1%
2	12.4966	17.4%
3	11.3053	16.2%
4	11.7634	14.8%
5	11.3451	15.8%
6	25.2535	12.1%
7	11.8785	13.5%
8	10.4291	15.8%
9	15.8025	35.0%
10	13.4376	19.0%
11	11.2812	12.7%
12	4.7730	11.7%

6.3 What is the quality of the fit compared to the current formula?

Table 6: MSE of current vs. new formula. Notice that there is a 35% improvement on the MSE on average across the sites. This represents a significant improvement over the current formula.

Site #	Current Predicted Power (kW) Formula	New Predicted Power (kW) Formula	Improvement %
1	12.7460	8.1577	36%
2	12.4966	8.8199	29%
3	11.3053	8.4141	26%
4	11.7634	8.0332	32%
5	11.3451	7.6125	33%
6	25.2535	16.9608	33%
7	11.8785	8.4227	29%
8	10.4291	7.2280	31%
9	15.8025	4.4258	72%
10	13.4376	9.2024	32%
11	11.2812	8.2444	27%
12	4.7730	4.3754	8%
Avg	12.7093	8.3247	35%

Table 7: Average percent error for current vs. new formula. Notice that there is a large improvement in the error percentage (28% on average).

Site #	Current Predicted Power Formula	New Predicted Power Formula	Improvement %
1	22.1%	12.5%	43%
2	17.4%	14.9%	14%
3	16.2%	16.2%	0%
4	14.8%	14.2%	4%
5	15.8%	11.4%	28%
6	12.1%	9.6%	21%
7	13.5%	13.0%	4%
8	15.8%	12.0%	24%
9	35.0%	8.0%	77%
10	19.0%	15.8%	17%
11	12.7%	11.6%	9%
12	11.7%	9.7%	17%
Avg	17.2%	12.4%	28%

As we can see, there is a dramatic improvement in the MSE, with a typical improvement of MSE being 35%. There is also a dramatic improvement in the overall percentage accuracy across the board, with an average of 28%. We conclude that in every case, the new algorithm decreases the large inaccuracies (much smaller MSE), but it also improves upon the non-extremes (better average percent error).

6.4 Analysis of the error reduction

Below, we examine the two predicted power equations and find the differences between the two. We expect that the new equation is better than the one currently used because by definition, the new algorithm reduces the mean squared error of the learning set to the minimal value. This, combined with our knowledge that the current algorithm has very high variability in its coefficients, help reassure us that it will be more accurate.

In Figure 31, the ‘Current Predicted Power’ equation (Equation 8) is being used. The ‘New Predicted Power’ equation is using the algorithm described above. We can see the error reduction graphically in the figures below.

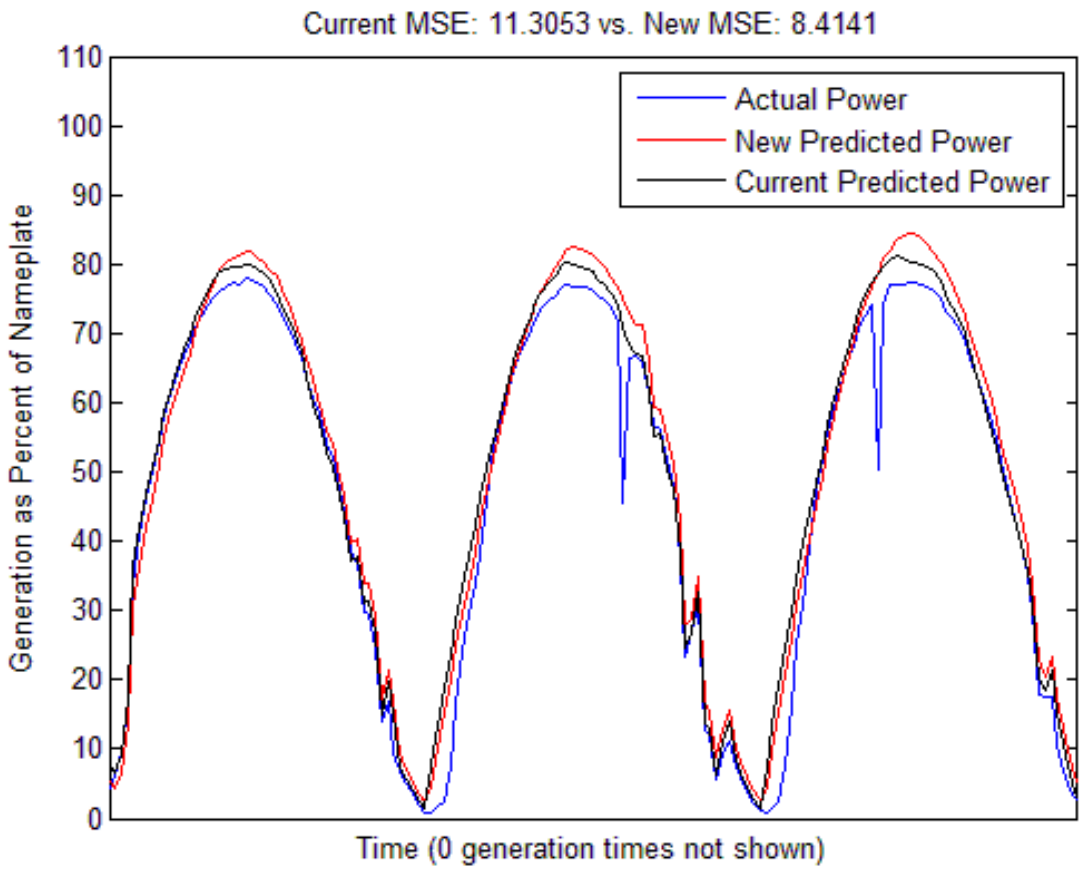

Figure 31: Both algorithms track the actual power pretty well. Notice how there is a slight decrease for actual power production on the second and third days. This could be due to a cloud that floated by that did not shadow the weather station, and therefore was not detected in either algorithm. Unfortunately, this is an inherent weakness in reliance on weather station data. That being said, this is not a daily occurrence, and the energy reduction due to these types of events is relatively small.

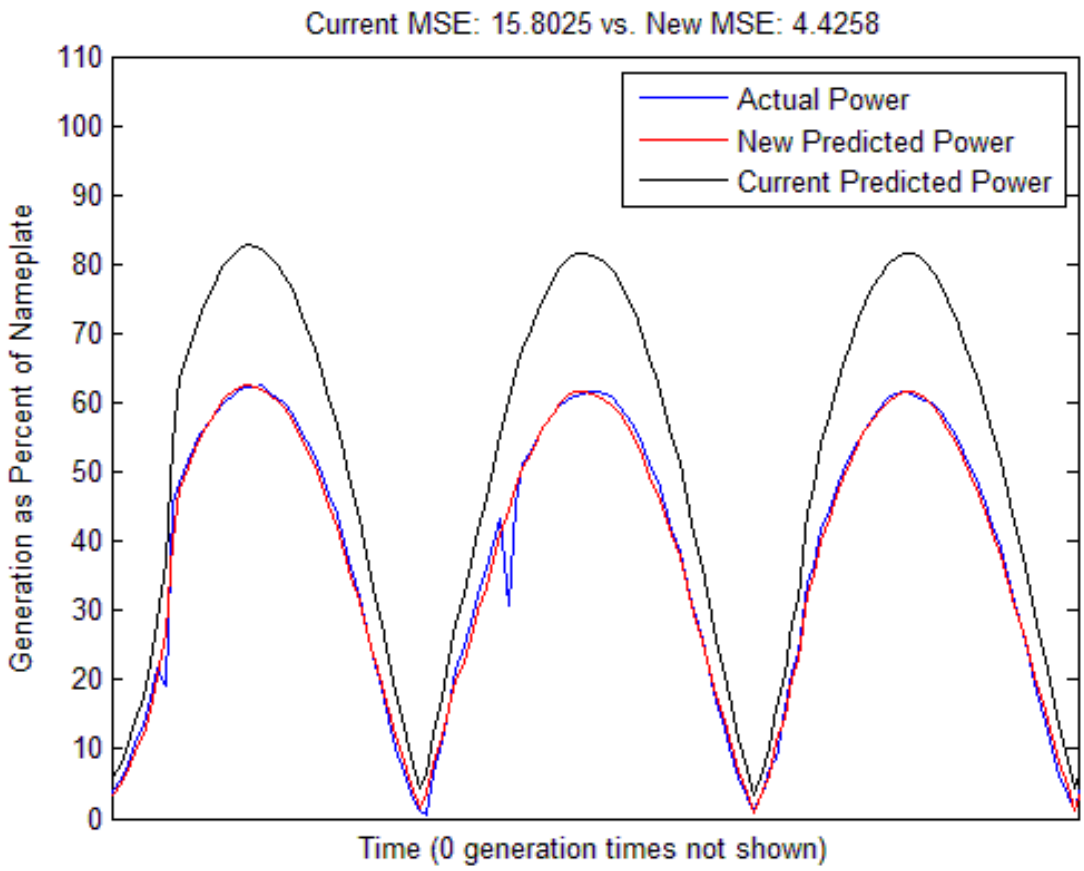

Figure 32: That the new algorithm tracks the actual power very closely while the current formula is completely wrong. This may be due to a mis-reported nameplate rating. One would expect a large MSE for the current equation, and a much smaller MSE for the new equation – which is indeed the case (15.8 vs. 4.4).

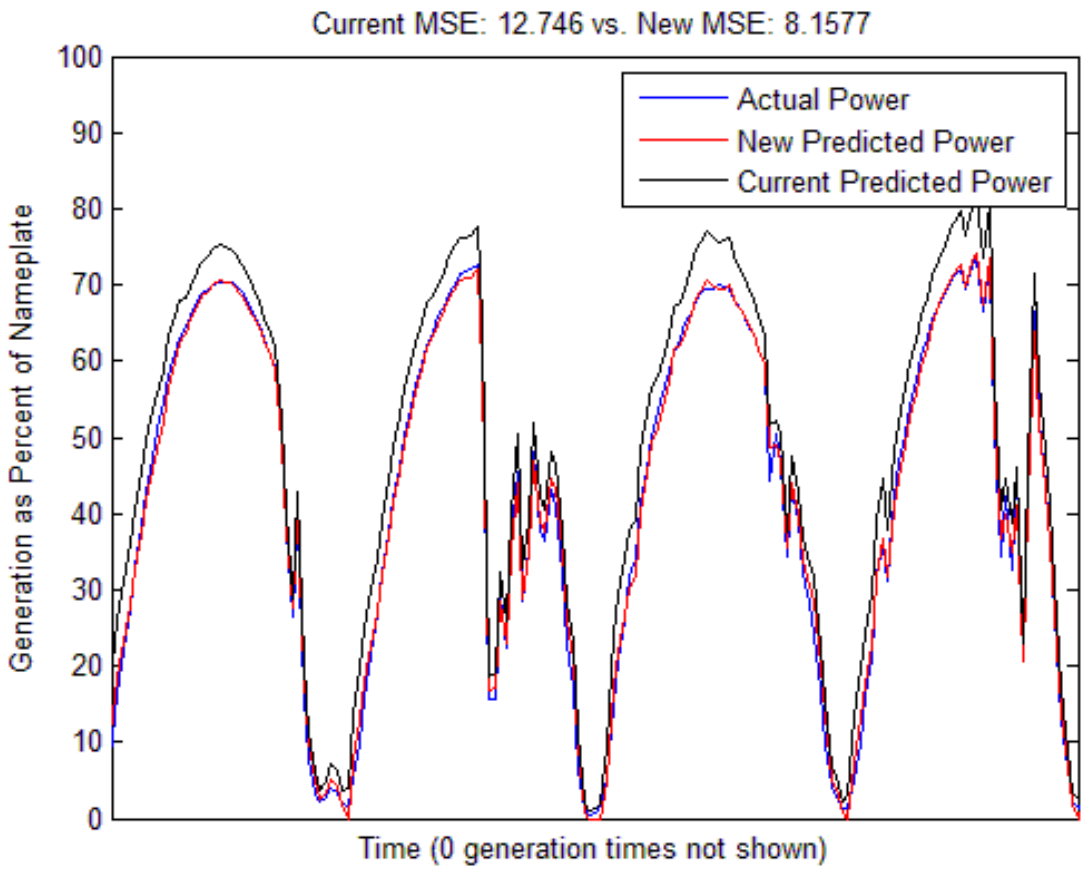

Figure 33: Even for periods of low generation, the new algorihtm tracks very well. We can see that there is a period of clouds on two of the days, and the predicted power algorithm predicts the reduced power output very accurately using the sensor information and data history.

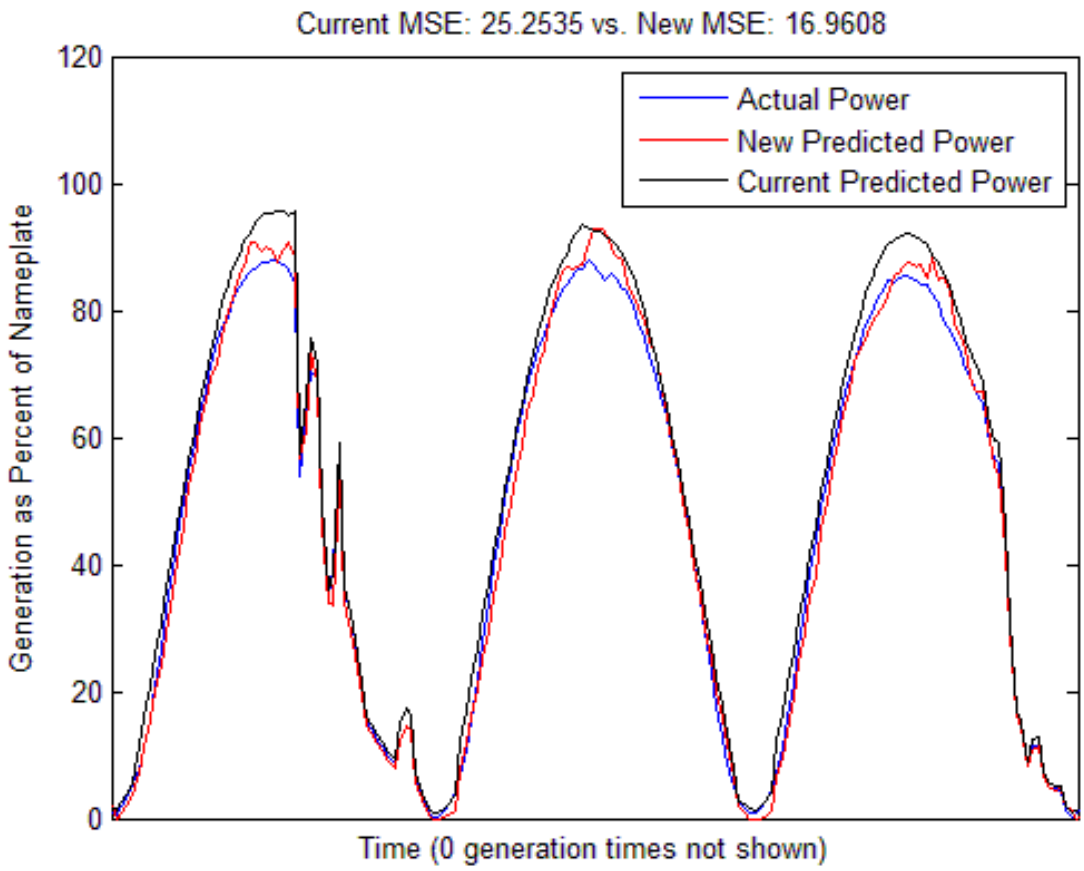

Figure 34: Example of a period where the difference is subtle, but the new algorithm tracks actual power more accurately. This is not a case of mis-applied nameplate ratings – for the rest of the period, the nameplate appears correct.

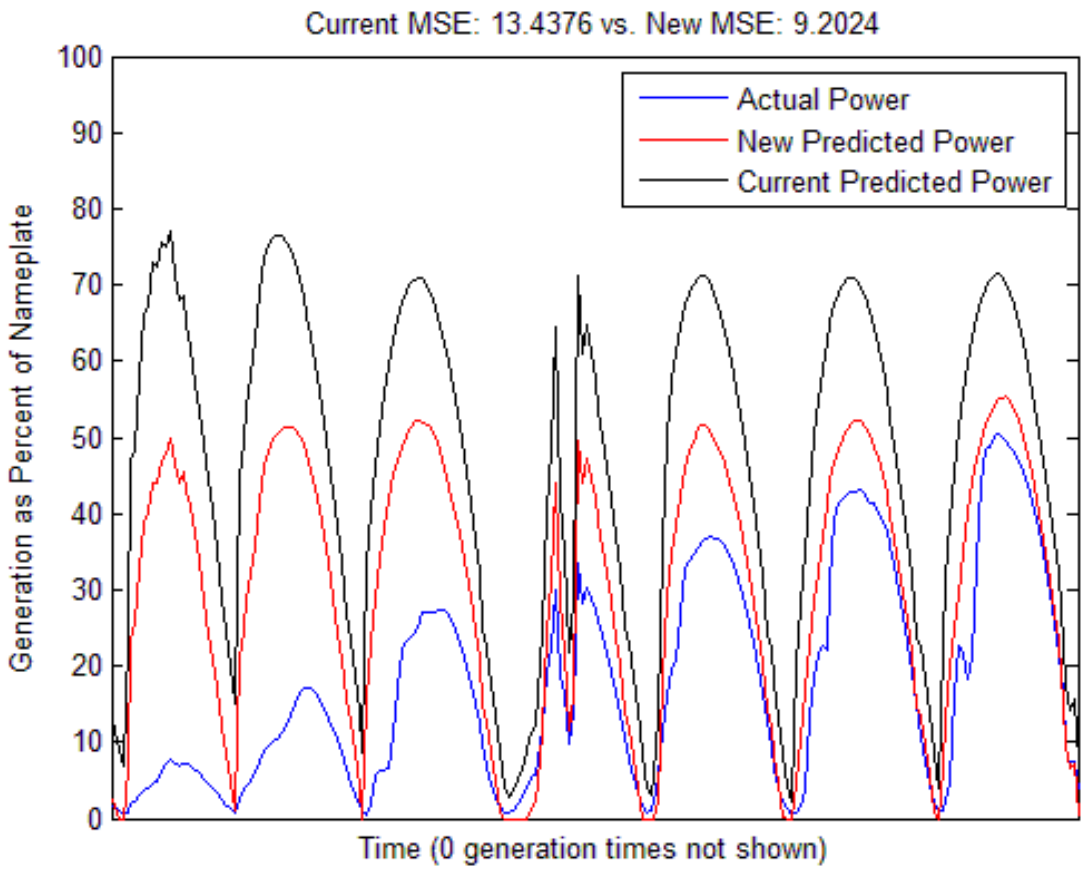

Figure 35: Example of a period where neither algorithm tracks accurately. Actual production is very low, indicating a problem with generation. This should trip an alarm.

The first part of the error reduction comes from the elimination of human error. As seen in the sample, in 15% of cases, the coefficient for performance dependence on cell temperature was very incorrect due to a misplaced decimal point. We did not include the erroneous coefficients in the anlaysis; we changed them all to their correct values.

The second way in which errors are reduced is by actually estimating the losses in the system rather than going by a default rating. Some modules are plus sorted, not all wires

are equal in losses, and there are any number of real-world scenarios which are impossible to predict.

An additional point that we would like to verify is that the correlation between the cell temperature and the power output has been accounted for. Figure 36 and Figure 37 show the percentage errors versus the cell temperature for the current as well as the new algorithms.

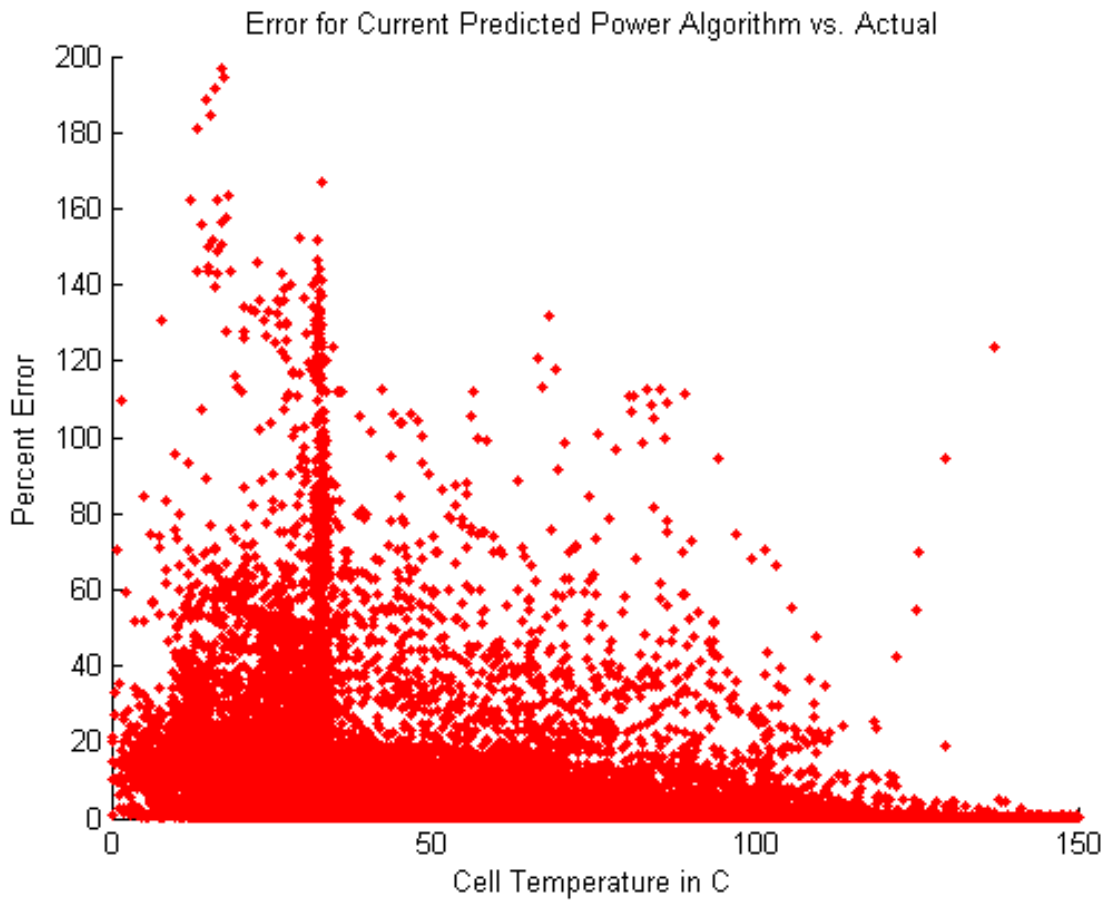

Figure 36: Graph of error versus cell temperature for current algorithm. This graph can be a bit misleading since there are so many more points around the 0-50C range (as we would expect) versus the 50-150 range. It appears that error decreases as temperature increases. This is not the case as we will see in another figure. However, this figure does give a sense of the spread of the data points.

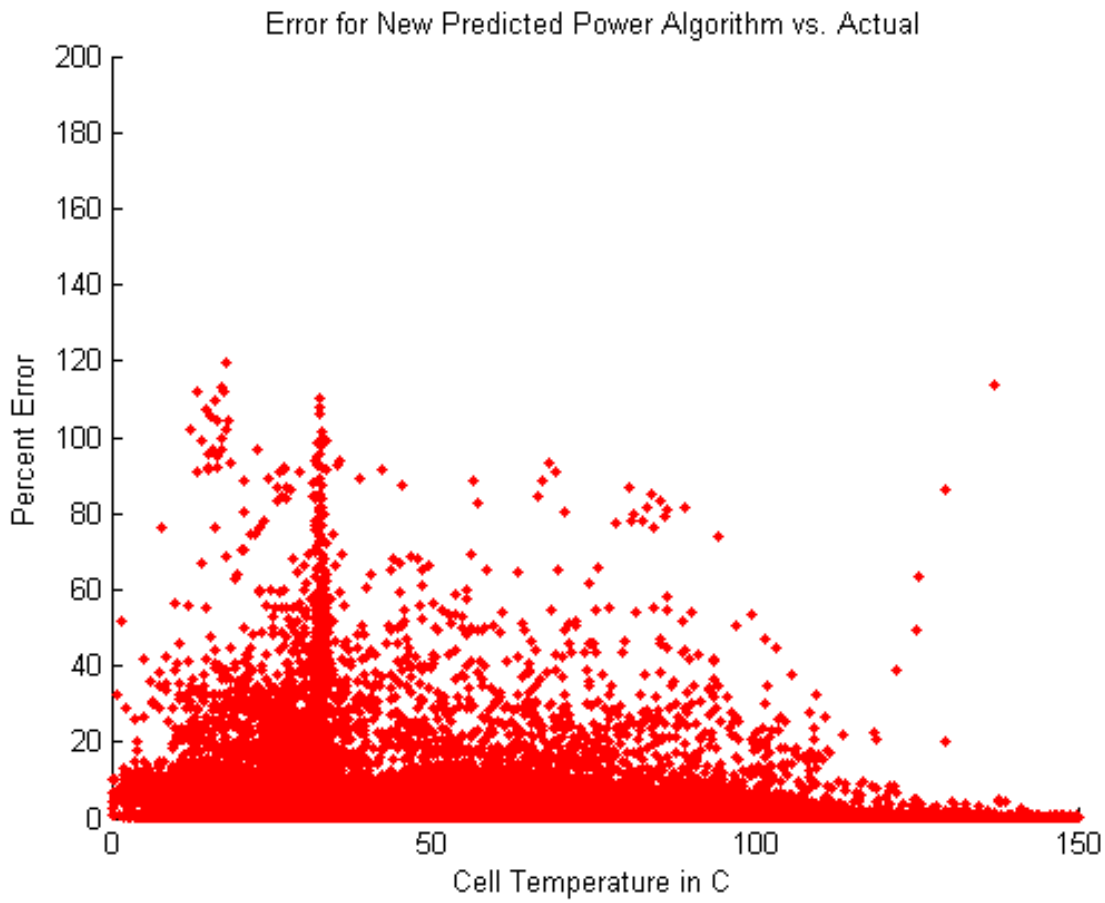

Figure 37: Graph of error versus cell temperature for new algorithm. The maximum error values have been significantly reduced with the new formula. This confirms our earlier findings that there is less error in the new model. Peak error is around 120 instead of more than 200% error. [3]

[3] Something that is apparent from both graphs is that there is a spike around 25-30 °C for cell temp. This means that when the day is beginning, a large period of time is spent in that temperature range. Unfortunately, it is difficult to see the average error in that range since the area is saturated with color. This will be explained below in other graphs.

Although it is interesting to note that the errors have declined, we wanted to see if the cell temperature dependence has been removed.

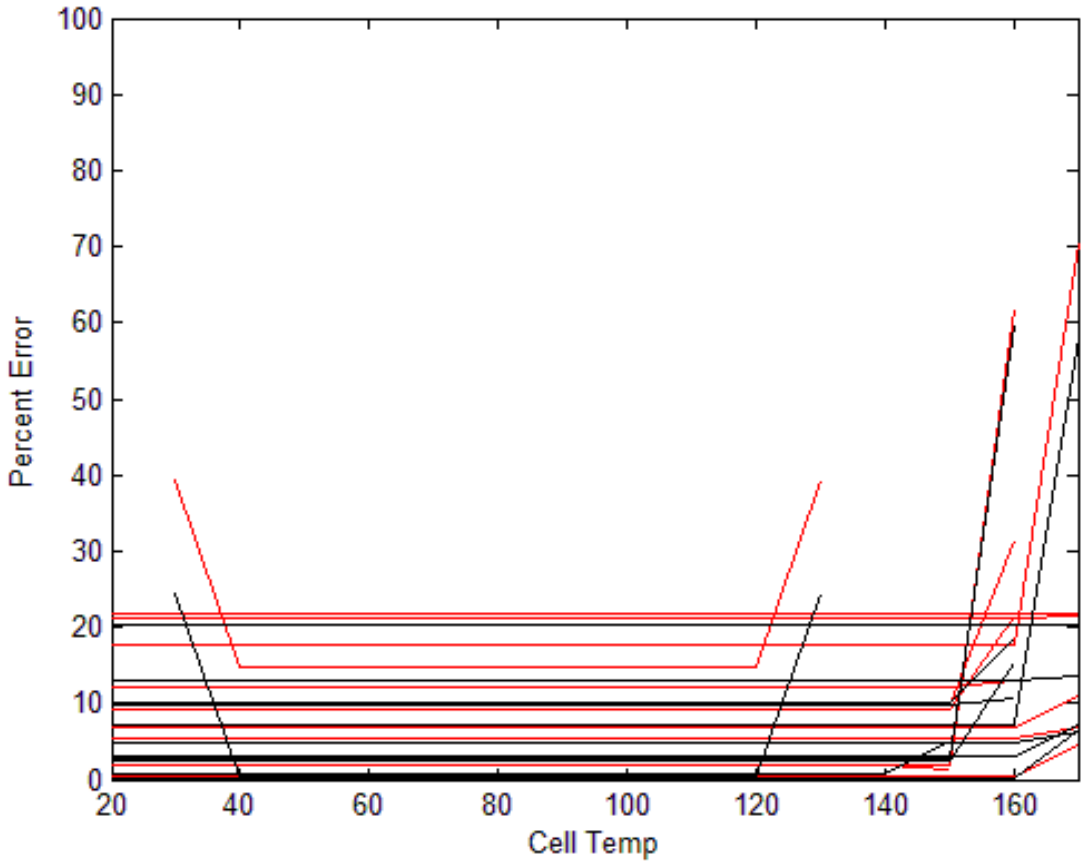

Figure 38: (Individual sites) Although the total of all elements shows that the new formula (red) has less error at every interval of cell temp than the original formula (black), there is no evidence of dependence of error on cell temp for either algorithm. This explains somewhat why the contribution of the cell temp did not (generally) have a meaningful impact on the accuracy of the predicted power equation.

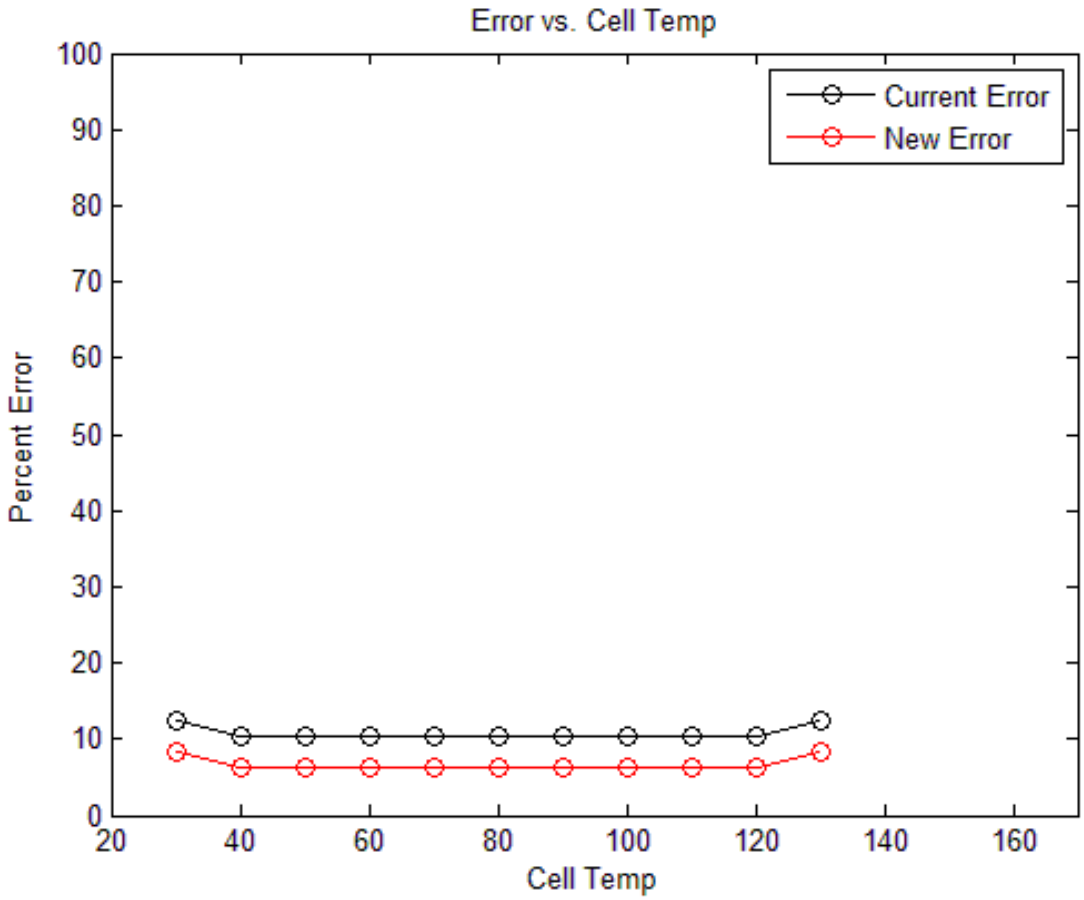

Figure 39: (Averaged across all sites) Although the total of all elements shows that the new formula (red) has less error at every interval of cell temp than the original formula (black), there is no evidence of dependence of error on cell temp for either algorithm. This explains somewhat why the contribution of the cell temp did not (generally) have a meaningful impact on the accuracy of the predicted power equation.

6.5 Can lower-quality sensors be used, and better predicted power still be obtained if we use this new predicted power algorithm?

Figure 40 shows the percent of variance explained by each of the variables. We can see clearly that the first PLS component explains the vast majority of the explained quantity. This first component is irradiance. The second component is cell temperature and the third component is ambient temperature. The other components help, but do not provide large decreases in error except for one particular example where the ambient temperature's contribution to the algorithm provides a very large decrease in error. What this means is that if cost is an issue, perhaps having the cell temperature and the ambient temperature sensors is not that important for the accuracy.

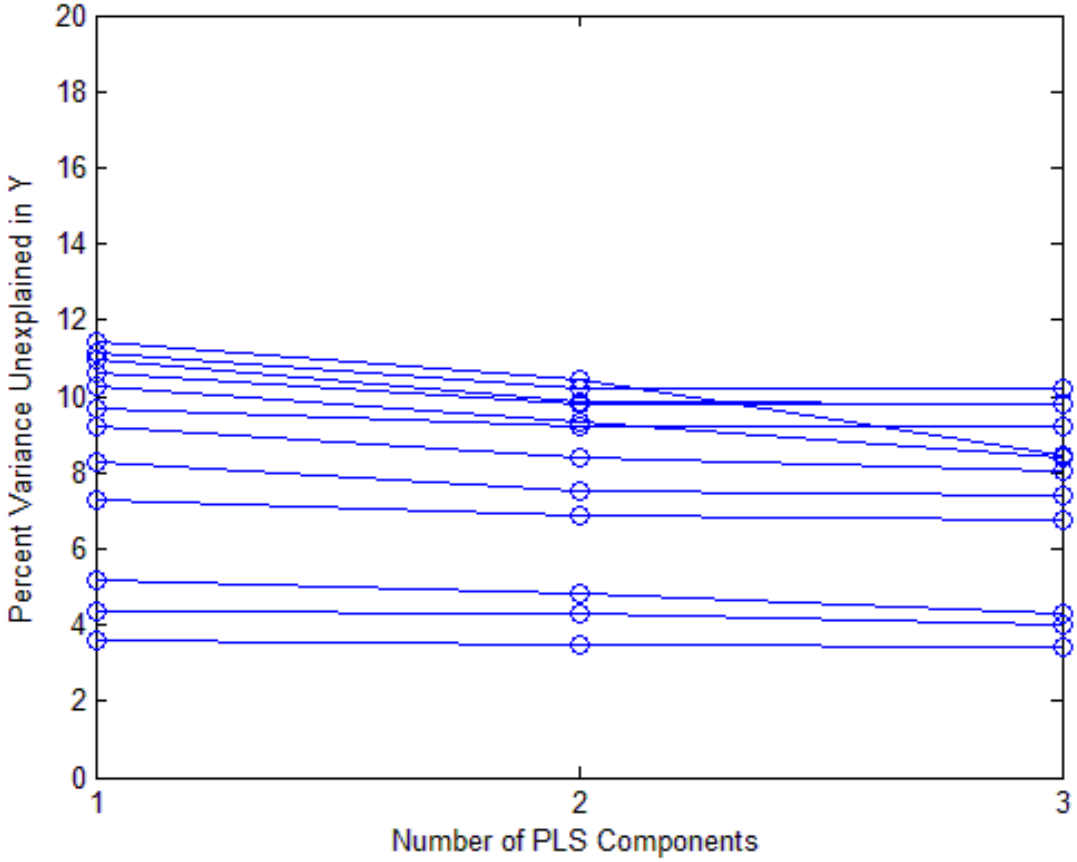

Figure 40: The number of PLS components included in the model. As the number of components in the model is increased, the unexplained variance goes down. However, we can see that the first element is generally the most important, and the rest of the components do not decrease the unexplained variance by more than a few percent.

If the ambient temperature and cell temperature sensors are removed, most of the variance can still be explained. This means that the new algorithm, with only a very good irradiance sensor, could match or exceed the accuracy of the current algorithm with cell temperature, irradiance, and ambient temperature sensors.

Conclusion

The solar industry appears to be growing larger every day, with a mostly positive future outlook. With the growing presence of solar monitoring companies, the possibilities exist to mine the data for value for the customers.

Four objectives were outlined in the abstract of this thesis. These objectives have all been satisfied, as discussed below.

Objective one was to verify whether customer-provided data could be trusted. Although customers cannot always be relied upon to provide accurate data, if procedures are in place to ensure accurate data streams, it represents a meaningful value to the solar industry. With the foundation of a good, reliable dataset, we can begin to do meaningful data analysis.

Objective two was to use solar data to improve sites. Using the techniques that we've outlined in this thesis, we can model sites and benchmark their performance against other 'real-world' sites as well as theoretical sites. If their performance lags, then we can determine which areas of their site most likely are the cause. We can also automatically assign dollar values to these problems. Using minimal engineering time, specific solutions can be proposed which maximize the return on investment of the site. At minimum, identifying these problems can be a bonus for future sites since the engineers are now aware of real-world examples of how to improve upon their designs.

Objective three was to determine if we could improve existing alarms with solar data. Using our data analysis, we used a location-based algorihtm for false alarm reduction. This decreased the number of false positives noticeably. There are other alarms that could

similarly be improved with the use of relevant solar data. Using these types of techniques would translate to better-maintained sites, operated closer to their true potential. This would also decrease the manpower required to keep those sites performing optimally.

Objective four was to improve the predicted power equation using solar data available from other sites. By having a very accurate predicted power equation, we can determine in real-time when a given site has a problem. We improved the existing equation using an augmented model of the solar site. Using this automated algorithm, without engineering intervention, we could determine when it was appropriate to send a field tech out to investigate the problem. In the future, we may even be able to arm them with the top three most likely causes to investigate first.

In this thesis, we have outlined only four of the ways in which solar data could be used to improve solar sites. This will help the industry to achieve higher returns for each site, which will help take the solar industry to the next level.

Bibliography

California Solar Initiative. (n.d.). Retrieved 1 14, 2013, from Solar Energy Research: http://www.gosolarcalifornia.ca.gov/professionals/research.php#csi

California Solar Initiative. (n.d.). Retrieved 01 14, 2013, from Download Currrent CSI Data: http://www.californiasolarstatistics.org/current_data_files/

EIA. (2012, 12 1). *December EIA Monthly Energy Review 2012*. Retrieved 1 7, 2013, from http://www.eia.gov/totalenergy/data/monthly/pdf/sec7_5.pdf

Gwinner, B. (2012, 08 21). Sales Account Manager at SunModo. (M. Ray, Interviewer)

Hoff, T. E. (2011). *Patent No. 8165812*. US.

Hoff, T. E. (2011). *Patent No. 8165813*. US.

IEEE Spectrum. (2012, 06 01). *The Solar Efficiency Gap*. Retrieved 01 15, 2013, from IEEE Spectrum: http://spectrum.ieee.org/green-tech/solar/the-solar-efficiency-gap

Kanellos, M. (2011, 03 17). *How to drop Solar to $1 a Watt*. Retrieved 15 01, 2013, from Greentech Solar: http://www.greentechmedia.com/articles/read/how-to-drop-solar-to-1-a-watt-try-diamond-saws-says-dick-swanson/

Keiser, R. (n.d.). *WREF2012: Calculating a Nation's "Economic" Solar Potential.* Retrieved 01 15, 2013, from https://ases.conference-services.net/resources/252/2859/pdf/SOLAR2012_0630_full%20paper.pdf

Kerrigan, S., Williams, M., & Herzig, M. (2012). *Patent No. 0191351*. US.

Lee, L., & Zazueta-Hall, I. (2010). *Patent No. 12/925705*. US.

Navigant Energy. (2012). *Solar Going Forward.* Retrieved 2013, from IREC USA: http://www.irecusa.org/wp-content/uploads/Mints.pdf

NREL. (2010, 01 01). Retrieved 01 15, 2013, from NREL: http://www.nrel.gov/docs/fy12osti/51847.pdf

Peleg, A., Herzig, M., & Kerrigan, S. (2010). *Patent No. 8190395*. US.

Perlin, J. (1999). From Space to Earth (The Story of Solar Electricity). Harvard University Press.

PV Education. (n.d.). *Nominal Operating Cell Temperature*. Retrieved 01 19, 2013, from PV Education: http://www.pveducation.org/pvcdrom/modules/nominal-operating-cell-temperature

The Economist. (2012, 11 21). *Sunny Uplands*. Retrieved 01 15, 2013, from The Economist: http://www.economist.com/news/21566414-alternative-energy-will-no-longer-be-alternative-sunny-uplands

Appendix: Matlab Code

```
% ActualNameplateComparison.m
% Find the actual percent generated vs nameplate generation
% ver 1.00 MR

Percentages = zeros(length(Sites),1);
for count = 1:length(Sites)
  Percentages(count) = mean(Sites(count).Power)/Sites(count).Nameplate*100;
end

% Delete errors (defined as efficiencies above 30%)
Percentages(find(Percentages > 30)) = []

hist(Percentages,10);
```

```
% AddCurrentPredictedPower.m
% Script to calculate current predicted power
% ver 1.00 MR

%Diagnostic Plots
PlotPower       = 0;
PlotPLSComp     = 0;
PlotRawCellTemp  = 0;
PlotCellTemp    = 0;
PlotAverageError = 1;

% Clean up the data
MarkedForDeletion = [];
for count = 1:length(Sites)
  % Remove times where power, irradiance and cell temp are 0
  IA = find(Sites(count).Power > 0.5 & Sites(count).Irradiance > 0.05 & Sites(count).CellTemperature >
0.05);
  Sites(count).Timestamps      = Sites(count).Timestamps(IA);
  Sites(count).Timestrings     = Sites(count).Timestrings(IA);
  Sites(count).Timenum         = Sites(count).Timenum(IA);
  Sites(count).Power           = Sites(count).Power(IA);
  Sites(count).Irradiance      = Sites(count).Irradiance(IA);
  Sites(count).CellTemperature  = Sites(count).CellTemperature(IA);
  Sites(count).CellTemperature2 = Sites(count).CellTemperature2(IA);
  Sites(count).Temperature     = Sites(count).Temperature(IA);

  if length(Sites(count).Power) < 100
    MarkedForDeletion = [MarkedForDeletion count];
  end
end
Sites(MarkedForDeletion)=[];

% Calculate Predicted Power using current formula
```

```
for count = 1:length(Sites)
  Temperature = (Sites(count).Temperature-32)*5/9; %Convert to Celcius
  Nameplate   = Sites(count).Nameplate;
  Irradiance  = Sites(count).Irradiance./1000;
  Coefficient = Sites(count).Coefficient;
  PredictedPower = Nameplate.*Irradiance.*(1-Coefficient.*Temperature);
  Sites(count).CurrentPredictedPower = PredictedPower;

  %Give Mean Squared Error
  PredPower = Sites(count).CurrentPredictedPower;
  Power     = Sites(count).Power;
  %Using unbiased mean squared error formula
  Sites(count).CurrentMSE          = sqrt(mean((PredPower - Power).^2));
  Sites(count).CurrentPercentError = mean(abs(PredPower - Power))/mean(Power)*100;
end

% Use Principle Least Squares regression to build model for each site
for count = 1:length(Sites)
  % Build relevant input matrices
  X = [Sites(count).Irradiance Sites(count).CellTemperature Sites(count).Temperature];
  y = Sites(count).Power;

  [Xloadings,Yloadings,Xscores,Yscores,betaPLS,PLSPctVar] = plsregress(X,y,3);
  NewPredictedPower = [ones(length(X),1) X]*betaPLS;
  Indeces = NewPredictedPower < 0;
  NewPredictedPower(Indeces) = 0;
  Sites(count).NewPredictedPower = NewPredictedPower;

  %Give Mean Squared Error
  PredPower = Sites(count).NewPredictedPower;
  Power     = Sites(count).Power;
  %Using unbiased mean squared error formula
  Sites(count).NewMSE          = sqrt(mean((PredPower - Power).^2));
  Sites(count).NewPercentError = mean(abs(PredPower - Power))/mean(Power)*100;

  if PlotPower == 1
    figure;
    title(['Current MSE: ' num2str(Sites(count).CurrentMSE) ...
      ' vs. New MSE:' num2str(Sites(count).NewMSE) ...
      ' Current % Error:' num2str(Sites(count).CurrentPercentError) ...
      ' vs. New % Error:' num2str(Sites(count).NewPercentError)]);
    hold on
    plot(1:length(Sites(count).NewPredictedPower)    ,Sites(count).NewPredictedPower    ,'r');
    plot(1:length(Sites(count).Power)                ,Sites(count).Power                ,'b');
    plot(1:length(Sites(count).CurrentPredictedPower),Sites(count).CurrentPredictedPower,'k');
    axis tight
    xlabel('Sample #')
    ylabel('Power as Percent of Nameplate')
  end

  if PlotPLSComp == 1
    plot(1:3,100-cumsum(100*PLSPctVar(2,:)),'-bo');
    hold on
```

```
    xlim([1 3]);
    ylim([0 20]);
    xlabel('Number of PLS Components');
    ylabel('Percent Variance Unexplained in Y');
    set(gca,'XTick',[1:3])
    set(gca,'XTickLabel',['1';'2';'3'])
  end

  clear X
  clear y
  clear yfitPLS
  clear Indeces
  clear NewPredictedPower
end

%Plot percent error with respect to cell temp
if PlotRawCellTemp == 1
  figure
  for count = 1:length(Sites)
    PredPower = Sites(count).CurrentPredictedPower;
    Power     = Sites(count).Power;
    PctErr    = abs(PredPower-Power);

    %Plot errors vs cell temp
    plot(Sites(count).CellTemperature,100*PctErr/Sites(count).Nameplate,'r.');
    hold on
    xlabel('Cell Temp')
    ylabel('Percent Error')
    xlim([20 170]);
    ylim([0 100]);
  end

  figure
  for count = 1:length(Sites)
    PredPower = Sites(count).NewPredictedPower;
    Power     = Sites(count).Power;
    PctErr    = abs(PredPower-Power);

    %Plot Percent Error vs cell temp
    plot(Sites(count).CellTemperature,100*PctErr/Sites(count).Nameplate,'k.');
    hold on
    xlabel('Cell Temp')
    ylabel('Percent Error')
    xlim([20 170]);
    ylim([0 100]);
  end
end

% Plot bins of percent error vs cell temp
if PlotCellTemp == 1 || PlotAverageError == 1
  TotalError = 0;
  figure
  for count = 1:length(Sites)
```

```
  PredPower = Sites(count).CurrentPredictedPower;
  Power     = Sites(count).Power;
  PctErr    = 100*abs(PredPower-Power)/Sites(count).Nameplate;
  %Calculate Bins
  Increment = 10;
  counter = 0;
  for bins = 10:Increment:170
    counter = counter + 1;
    Indeces = INTERSECT((Sites(count).CellTemperature < bins+Increment/2), ...
                        (Sites(count).CellTemperature > bins-Increment/2));
    CellTemp(counter) = bins;
    if sum(Indeces) == 0
      Error(counter) = NaN;
    else
      Error(counter) = PctErr(Indeces);
    end
  end
  TotalError = TotalError + Error;

  if PlotCellTemp == 1
    %Plot errors vs cell temp
    plot(CellTemp,Error,'r');
    hold on
    xlabel('Cell Temp')
    ylabel('Percent Error')
    xlim([20 170]);
    ylim([0 100]);
  end
  clear Error
end

if PlotAverageError == 1
  % Plot the average error of the current algorithm
  figure
  plot(CellTemp,TotalError/12,'-ko');
  hold on
  xlabel('Cell Temp')
  ylabel('Percent Error')
  xlim([20 170]);
  ylim([0 100]);
  clear CellTemp
  clear TotalError
end

 TotalError = 0;
%  figure
 for count = 1:length(Sites)
  PredPower = Sites(count).NewPredictedPower;
  Power     = Sites(count).Power;
  PctErr    = 100*abs(PredPower-Power)/Sites(count).Nameplate;
  %Calculate Bins
  Increment = 10;
  counter = 0;
  for bins = 10:Increment:170
```

```
    counter = counter + 1;
    Indeces = INTERSECT((Sites(count).CellTemperature < bins+Increment/2), ...
                (Sites(count).CellTemperature > bins-Increment/2));
    CellTemp(counter) = bins;
    if sum(Indeces) == 0
      Error(counter) = NaN;
    else
      Error(counter) = PctErr(Indeces);
    end
  end
  TotalError = TotalError + Error;

  if PlotCellTemp == 1
    %Plot errors vs cell temp
    plot(CellTemp,Error,'k');
    hold on
    xlabel('Cell Temp')
    ylabel('Percent Error')
    xlim([20 170]);
    ylim([0 100]);
  end
  clear Error
 end

 if PlotAverageError == 1
  % Plot the average error of the new algorithm
  plot(CellTemp,TotalError/12,'-ro');
  hold on
  xlabel('Cell Temp')
  ylabel('Percent Error')
  xlim([20 170]);
  ylim([0 100]);
  title('Error vs. Cell Temp')
  legend('Current Error','New Error')
  clear CellTemp
  clear TotalError
 end
end
```

```
% rev.m
% Code taken from MATLAB Central

function [reversed] = rev(x)
% Reverses x

[rows cols] = size(x);
reversed = zeros(length(x),1);
revcount = 0;
for count = length(x):-1:1
  revcount = revcount + 1;
  reversed(revcount) = x(count);
end
```

```
[revrows revcols] = size(reversed);
if revrows ~= rows
  reversed = reversed';
end
```

```
% SetNearbySites.m
% ver 2.00 MR

function [Sites] = SetNearbySites(Sites,numDays,radius,significance)
% Finds nearby sites, runs the site comparison and assigns both the site
% index as well as the score to the newly created Nearby and Scores
% variables.

  ProgressBar = waitbar(0,'Analyzing Nearby Sites');
  plotresults = true;
  if plotresults
    figure
  end

  for count = 1:length(Sites)
    Scores    = [];
    Nearby    = [];
    Distances = [];
    Sites(count).Scores    = [];
    Sites(count).Nearby    = [];
    Sites(count).Distances = [];
    [Nearby Distances]= GetNearbySites(Sites,count,radius);
    waitbar(count/length(Sites),ProgressBar,['Analyzing - ' num2str(count) ' out of ' num2str(length(Sites)) ' '
num2str(length(Nearby)) ' sites nearby']);
    for count2 = 1:length(Nearby)
      Scores(count2) = CompareSites(Sites(Nearby(count2)),Sites(count),numDays);
    end
    if plotresults
      hold on; plot(Distances,Scores,'r.');
    end
    for count2 = length(Nearby):-1:1
      if Scores(count2) < significance
        Scores(count2)    = [];
        Distances(count2) = [];
        Nearby(count2)    = [];
      end
    end
    Sites(count).Scores    = Scores;
    Sites(count).Distances = Distances;
    Sites(count).Nearby    = Nearby;
  end
  close(ProgressBar);
end
```

```
% GetNearbySites.m
```

```
% ver 1.00 MR

function [ SiteIndeces,Distances] = GetNearbySites(Sites,Sitenum,radius)
% SiteIndeces: Draws a circle with radius (in miles) of the input 'radius'
%              around the lat/long of the input sitenum and returns the
%              indeces of the sites which are in the circle.
% Distances:   Distances to those respective sites in approximate miles.

  % Convert radius to long/lat. Note that this is very approximate. This
  % function (for now) creates a rectangle around the site, not a circle.
  % Also, distance measurements are not exact since spacing between
  % lattitude and longitude depends on where you are on the planet. Also,
  % the earth is not a sphere either, which adds to the complication.
  % However, this is a good approximate square for our analysis purposes.
  SiteIndeces = [];
  Distances   = [];
  RadiusLat   = radius/35; % One degree Lat is approximately 35 miles in the US
  RadiusLong  = radius/35; % One degree Long is approximately 35 miles in the US
  Latmin  = Sites(Sitenum).Lat - RadiusLat;
  Latmax  = Sites(Sitenum).Lat + RadiusLat;
  Longmin = Sites(Sitenum).Long - RadiusLong;
  Longmax = Sites(Sitenum).Long + RadiusLong;
  for count = 1:length(Sites)
    TempLat  = Sites(count).Lat;
    TempLong = Sites(count).Long;
    if TempLat < Latmax && TempLat > Latmin && TempLong < Longmax && TempLong > Longmin
      if count ~= Sitenum
        SiteIndeces = [SiteIndeces count];
        Distances = [Distances ...
          sqrt(((TempLat-Sites(Sitenum).Lat)*35)^2+((TempLong-Sites(Sitenum).Long)*35)^2)];
      end
    end
  end
end
```

```
% CompareSites.m
% ver 1.00 MR

function [R] = CompareSites(Site1,Site2,numDays)
% Compares site generation. If the two sites follow each other, then they
% are deemed a match. It will keep going starting from the first valid day
% that both sites have until it hits numDays. Then, it decides whether or
% not the two have generation profiles that basically match each other.
%
% Inputs:
%  Site1: First Site Structure
%  Site2: Second Site Structure
% Output:
%  R: R-value ranges from -1 to 1. The higher the better, but it remains to
%     be seen how high is 'significant.'

  % Find the times that are shared by both sites
```

```
[Times,I1,I2] = intersect(Site1.Timenum,Site2.Timenum);

% Get the associated power numbers for those times that are shared
previous1 = 1; % Make the search faster by not searching entries twice
previous2 = 1;

Power1 = Site1.Power(I1);
Power2 = Site2.Power(I2);

if length(Times) < 4*24*numDays
  R = 0;
  return
end
Times  = Times(1:4*24*numDays);
Power1 = Power1(1:4*24*numDays);
Power2 = Power2(1:4*24*numDays);

% Cycle through until the number of days is reached
% Normalize the two datasets by their maximum generation during the
% numDays period. Call this the nameplate for that site.
Nameplate1 = max(Power1); % 15 minute data = 4 samples per hour, 24 hours per day
Nameplate2 = max(Power2);

% Normalize the Power Data
Power1 = Power1 ./ Nameplate1;
Power2 = Power2 ./ Nameplate2;

% Replace 0 with 0.0001 to avoid divide by zero errors
Temp = find(0 == Power1);
Power1(Temp) = 0.00001;
Temp = find(0 == Power2);
Power2(Temp) = 0.00001;

 % Calculate the quality of the match using Pearson's correlation coefficient
%  R = sum((X-mean(X)).*(Y-mean(Y)))./sqrt(sum(((X-mean(X)).^2).*((Y-mean(Y)).^2)));
 r = corrcoef([Power1 Power2]);
 R=r(1,2); % Get the cross correlation R value

%     For debug purposes
% figure
% plot(Times,Power1,'rx');
% hold on
% plot(Times,Power2,'bx');
end
```

```
% CleanData.m
% Removes extended periods (more than 1 day) of zero generation
% Removes days with spikes above 130% nameplate generation. These are most
% likely errors
% ver 1.00 MR
```

```
ProgressBar = waitbar(0,'Cleaning Data'); tic
numsites = length(Sites);
RemoveSite = zeros(length(Sites),1);
plotresults = false;

for sitenum = 1:numsites
  if sitenum > 10
    plotresults = false;
  end
 if iscell(Sites(sitenum).Lat)
   Sites(sitenum).Lat  = str2num(Sites(sitenum).Lat{:} );
   Sites(sitenum).Long = str2num(Sites(sitenum).Long{:});
 end
 if plotresults
   figure
 end
 Power = Sites(sitenum).Power;
 Timestamps = Sites(sitenum).Timestamps;
 DayTimestamps = datenum(datestr(Timestamps,'yyyymmdd'),'yyyymmdd');

 UniqueDays = unique(DayTimestamps);
 ValidDay = ones(length(UniqueDays),1);
 for count = length(UniqueDays):-1:1
   Indeces = find(DayTimestamps == UniqueDays(count));
   % For each day, check if there was zero generation
   if sum(Power(Indeces))/Sites(sitenum).Nameplate < 0.01
     ValidDay(count) = 0;
     if plotresults
       plot(Timestamps(Indeces),Power(Indeces),'rx');
       hold on
     end
   % For each day, check if there was higher than rated generation
   elseif max(Power(Indeces))/Sites(sitenum).Nameplate > 1.3
     ValidDay(count) = 0;
     if plotresults
       plot(Timestamps(Indeces),Power(Indeces),'go');
       hold on
     end
   elseif min(Power(Indeces))/Sites(sitenum).Nameplate < -0.1
     ValidDay(count) = 0;
     if plotresults
       plot(Timestamps(Indeces),Power(Indeces),'ro');
       hold on
     end
   else
     ValidDay(count) = 1;
     if plotresults
       plot(Timestamps(Indeces),Power(Indeces),'b.');
       hold on
     end
   end
   % Remove day's data points if generation is invalid
   if ValidDay(count) == 0
     Sites(sitenum).Timestamps(Indeces)  = [];
```

```
    Sites(sitenum).Timestrings(Indeces) = [];
    Sites(sitenum).Timenum(Indeces)     = [];
    Sites(sitenum).Power(Indeces)       = [];
  end
 end

 % If more than 25% of the days were removed, then the site should be
 % removed from the dataset
 if mean(ValidDay) < 0.75
  disp(['Percent Valid:' num2str(mean(ValidDay))]);
  RemoveSite(sitenum) = 1;
 elseif length(Sites(sitenum).Power) < 3000
  RemoveSite(sitenum) = 1;
 % For this dataset, remove all values which haven't reported since May
 % 1st 2012. Going forward, this will not be this arbitrary date. It would
 % probably be something pretty close to the actual date.
 elseif max(Sites(sitenum).Timenum) < 201205010000
  RemoveSite(sitenum) = 1;
 end
 if plotresults
  % Format Axes
  axis tight
  xData = linspace(min(UniqueDays),max(UniqueDays),length(UniqueDays));
  set(gca,'XTick',xData)
  datetick('x','yyyymmdd','keepticks')
  ylabel('Average Power Generation')
  xlabel('Day')
 end
 waitbar(sitenum/length(Sites),ProgressBar,[num2str(sitenum) ' out of ' num2str(length(Sites))]);
end

% Remove sites with more than 25% of days flagged
for count = numsites:-1:1
 if RemoveSite(count) == 1
  Sites(count) = [];
 end
end

disp([num2str(sum(RemoveSite)) ' Sites Removed']);

clear Indeces
clear Power
clear Timestamps
clear plotresults
clear ValidDay
clear RemoveSite
clear numsites
clear count
clear sitenum
clear DayTimestamps
clear UniqueDays
close(ProgressBar); toc
```

```
% PlotNearby.m
% ver 1.00 MR

function PlotNearby(Sites,sitenum)
% Plots the nearby site's locations. This is useful during debugging.
  Nearby = Sites(sitenum).Nearby;

  figure
  plot(Sites(sitenum).Long,Sites(sitenum).Lat,'rx');
  hold on
  for count = 1:length(Nearby)
    plot(Sites(Nearby(count)).Long,Sites(Nearby(count)).Lat,'bx');
  end
end
```

```
% TraditionalDECKAlarms.m
% ver 1.00 MR

function [Sites] = RangeTimeAlarm(Sites, option)
% Run Tradional DECK Alarm analysis
% option:   User can select 'DECK' or 'MCR'. MCR is an upgraded version of
%           the DECK algorithm.
% Sites:    Input Sites are used to calculate the alarms. Results are
%           appended to the structure and returned as Sites as well.

if strcmp(option,'DECK')
  ProgressBar = waitbar(0,'Running Standard DECK Analysis');
  for count = 1:length(Sites)
    [AlarmTimes,TotalDays] = TimeGenerationAlarm(Sites(count));
    Sites(count).AlarmTimes = AlarmTimes;
    Sites(count).TotalDays  = TotalDays;
    waitbar(count/length(Sites),ProgressBar,[num2str(count) ' out of ' num2str(length(Sites))]);
  end
  close(ProgressBar);
elseif strcmp(option,'MCR')
  ProgressBar = waitbar(0,'Running Upgraded DECK Analysis');
  for count = 1:length(Sites)
    [AlarmTimes,TotalDays] = TimeGenerationAlarm(Sites(count));
    Sites(count).AlarmTimes = AlarmTimes;
    Sites(count).TotalDays  = TotalDays;
    waitbar(count/length(Sites),ProgressBar,[num2str(count) ' out of ' num2str(length(Sites))]);
  end
  close(ProgressBar);
else
  disp('Invalid option selected');
end
```

```
% TimeGenerationAlarm.m
% ver 1.00 MR
```

```
function [AlarmDays,AlarmIndex,TotalDays] = TimeGenerationAlarm(Site)
% Site:       Input structure for a given site
% AlarmDays:  Days that the alarm goes off
% TotalDays:  Total number of days in the dataset
% AlarmIndex: Indeces when the alarms are being set off

 Times       = Site.Timenum;
 Power       = Site.Power;
 plotresults = false;

 % Set Dates to yyyymmdd
 Dates = zeros(length(Times),1);
 for count = 1:length(Times)
  temp = num2str(Times(count));
  temp = temp(1:8);
  Dates(count) = str2num(temp);
 end

 % Set Times to HHMM
 for count = 1:length(Times)
  temp = num2str(Times(count));
  temp = temp(9:12);
  Times(count) = str2num(temp);
 end

 % The test alarm criteria is that generation is 50% below nameplate
 % during 10am to 2pm
 [Indeces1] = find(Power < Site.CalcNameplate*0.5);
 [Indeces2] = find(Times > 1000);
 [Indeces3] = find(Times < 1400);
 AlarmIndex = intersect(intersect(Indeces1,Indeces2),Indeces3);

 if plotresults
  % Plot Alarm Points
  plot(Site.Timestamps,Power./Site.Nameplate*100,'k');
  hold on; plot(Site.Timestamps(AlarmIndex),Power(AlarmIndex)./Site.Nameplate*100,'ro','Linewidth',1);
%   [Indeces] = union(find(Times == 1400),find(Times == 1000));
%   for count = 1:length(Indeces)
%     plot([Site.Timestamps(Indeces(count)) Site.Timestamps(Indeces(count))],[0 100],'r','LineWidth',2);
%   end
  plot([Site.Timestamps(1) Site.Timestamps(end)],[50 50],'r','LineWidth',1);
  ylim([-10 100]);
  ylabel('Generation as Percent of Nameplate');
  set(gca, 'XTick', []);
  xlabel('Time');
  axis tight
 end

 TotalDays  = length(unique(Dates));
 Dates      = Dates(AlarmIndex);
 [AlarmDays,Rows] = unique(Dates);
end
```

```
% RangeTimeAlarm.m
% ver 3.05 MR

function [Sites] = RangeTimeAlarm(Sites, option, significance)
% Run Tradional DECK Alarm analysis
% option:       User can select 'DECK' or 'MCR'. MCR is an upgraded version of
%               the DECK algorithm.
% significance: Uses all sites which have correlation above 'significance'.
%               A good number to use is 0.9 or above.
% Sites:        Input Sites are used to calculate the alarms. Results are
%               appended to the structure and returned as Sites as well.
tic
if strcmp(option,'DECK')
  ProgressBar = waitbar(0,'Running Standard DECK Analysis');
  for count = 1:length(Sites)
    [AlarmDays,AlarmIndex,TotalDays] = TimeGenerationAlarm(Sites(count));
    Sites(count).AlarmIndex = AlarmIndex;
    Sites(count).AlarmDays = AlarmDays;
    Sites(count).TotalDays = TotalDays;
    waitbar(count/length(Sites),ProgressBar,[num2str(count) ' out of ' num2str(length(Sites))]);
  end
  close(ProgressBar);
elseif strcmp(option,'MCR')
  ProgressBar = waitbar(0,'Running Upgraded DECK Analysis');
  for count = 1:length(Sites)
    % Get dumb alarm times
    [AlarmDays,AlarmIndex,TotalDays] = TimeGenerationAlarm(Sites(count));
    Sites(count).AlarmIndex = AlarmIndex;
    Sites(count).AlarmDays  = AlarmDays;
    Sites(count).TotalDays  = TotalDays;

    % Verify if nearby sites are also going down by a similar amount
    Index1     = count;
    AlarmIdx   = Sites(count).AlarmIndex;
    AlarmTimes = Sites(count).Timenum(AlarmIdx);
    AlarmPower = Sites(count).Power(AlarmIdx)./Sites(count).CalcNameplate;
    Scores     = zeros(length(AlarmTimes),1);
    Numbers    = zeros(length(AlarmTimes),1);
    if length(Sites(count).Nearby) ~= 0
      for count2 = 1:length(Sites(count).Nearby)
        if Sites(count).Scores(count2) > significance
          Index2 = Sites(count).Nearby(count2);
          [C,IA,IB] = intersect(Sites(Index2).Timenum,AlarmTimes);
          Scores(IB)  = Scores(IB) + Sites(Index2).Power(IA)./Sites(Index2).CalcNameplate;
          Numbers(IB) = Numbers(IB)+ ones(length(IB),1);
        end
      end
      NormScores = Scores./Numbers;

      % Find points where the site is generating 20% of nameplate less than
      % its peers and flag that as an actual alarm.
      % If the power is above its neighbors, then that's not really a problem
```

```
    % and even though it's below the alarm threshold, it should not trigger
    % an alarm.
    Indeces1 = find(AlarmPower < NormScores - 0.2);
    Indeces2 = find(NormScores == 0);
    ReducedAlarmTimes = AlarmTimes(union(Indeces1,Indeces2));

    % Also save the percentage that the alarms were reduced
    Sites(count).ReducedAlarmTimes = ReducedAlarmTimes;
   else
    % If there are no comparable sites, set the reduced alarm times to
    % the previously found alarm times.
    Sites(count).ReducedAlarmTimes = Sites(count).Timenum(Sites(count).AlarmIndex);
   end
   waitbar(count/length(Sites),ProgressBar,[num2str(count) ' out of ' num2str(length(Sites)) ' Remaining
Time:' num2str(toc/60/count*(length(Sites)-count)) ' minutes']);
  end
  close(ProgressBar);
else
  disp('Invalid option selected');
end
toc
```

```
% PlotPredictedPower.m
% ver 1.00 MR

function PlotPredictedPower(Site)
% Plots an individual site's current and new predicted power algorithm
% results
  plot(1:length(Site.Timestamps),Site.Power./Site.Nameplate*100,'b');
  hold on
  plot(1:length(Site.Timestamps),Site.NewPredictedPower./Site.Nameplate*100,'r');
  plot(1:length(Site.Timestamps),Site.CurrentPredictedPower./Site.Nameplate*100,'k');
  legend('Actual Power','New Predicted Power','Current Predicted Power');
  title(['Current MSE: ' num2str(Site.CurrentMSE) ' vs. New MSE: ' num2str(Site.NewMSE)]);
  set(gca, 'XTick', []);
  xlabel('Time (0 generation times not shown)')
  ylabel('Generation as Percent of Nameplate')
  ylim([0 100]);
end
```

```
% PlotNearbyGeneration.m
% ver 1.00 MR

function PlotNearbyGeneration( Sites,sitenum )
% Plots generation profiles of correlated sites near sitenum

  % Matlab colors: ymcrgbwk
  NearbySites = Sites(sitenum).Nearby;
  figure
  plot(Sites(sitenum).Timestamps,Sites(sitenum).Power./Sites(sitenum).Nameplate*100,'r');
  hold on
```

```
  if length(NearbySites)>0

plot(Sites(NearbySites(1)).Timestamps,Sites(NearbySites(1)).Power./Sites(NearbySites(1)).Nameplate*100
,'k');
  end
  if length(NearbySites)>1

plot(Sites(NearbySites(2)).Timestamps,Sites(NearbySites(2)).Power./Sites(NearbySites(2)).Nameplate*100
,'m');
  end
  if length(NearbySites)>2

plot(Sites(NearbySites(3)).Timestamps,Sites(NearbySites(3)).Power./Sites(NearbySites(3)).Nameplate*100
,'b');
  end
  if length(NearbySites)>3

plot(Sites(NearbySites(4)).Timestamps,Sites(NearbySites(4)).Power./Sites(NearbySites(4)).CalcNameplate
*100,'g');
  end
  if length(NearbySites)>4

plot(Sites(NearbySites(5)).Timestamps,Sites(NearbySites(5)).Power./Sites(NearbySites(5)).CalcNameplate
*100,'c');
  end
  if length(NearbySites)>5

plot(Sites(NearbySites(6)).Timestamps,Sites(NearbySites(6)).Power./Sites(NearbySites(6)).CalcNameplate
*100,'y');
  end

  ylim([0 100]);
  ylabel('Generation as Percent of Nameplate');
  xlabel('Time');
  set(gca, 'XTick', []);

end
```

```
% PlotErrorVsCellTemp.m
% ver 1.00 MR

function PlotErrorVsCellTemp(Sites)
%Plots error vs cell temp for either the old or the new algorithms for
%predicted power

  % Plot raw total error vs. cell temperature
  % For new algorithm
  figure
  for count = 1:length(Sites)
    Error = abs((Sites(count).Power - Sites(count).NewPredictedPower)./Sites(count).Power);
    hold on
    plot(Sites(count).CellTemperature,Error,'r.');
```

```
  end
  title('Error for New Predicted Power Algorithm vs. Actual')
  xlim([0 150]);
  ylim([0 200]);
  xlabel('Cell Temperature in C')
  ylabel('Percent Error')

  % Plot raw total error vs. cell temperature
  % For current algorithm
  figure
  for count = 1:length(Sites)
    Error = abs((Sites(count).Power - Sites(count).CurrentPredictedPower)./Sites(count).Power);
    hold on
    plot(Sites(count).CellTemperature,Error,'r.');
  end
  title('Error for Current Predicted Power Algorithm vs. Actual')
  xlim([0 150]);
  ylim([0 200]);
  xlabel('Cell Temperature in C')
  ylabel('Percent Error')

  % Plot average total error vs. cell temperature

end
```

```
% PlotAvgPowerByLocation.m
% ver 1.00 MR
% Plot generation by lat/long location

for count = 1:length(Sites)
  if Sites(count).AvgPower < 0.05
    plot(Sites(count).Long,Sites(count).Lat,'wo');
  elseif Sites(count).AvgPower < 0.10
    plot(Sites(count).Long,Sites(count).Lat,'bo');
  elseif Sites(count).AvgPower < 0.15
    plot(Sites(count).Long,Sites(count).Lat,'yo');
  elseif Sites(count).AvgPower < 0.20
    plot(Sites(count).Long,Sites(count).Lat,'ro');
  else
    plot(Sites(count).Long,Sites(count).Lat,'ko');
  end
  hold on
end
title('Generation by location')
```

```
% PerMonthAnalysisBarGraph.m
% ver 1.00 MR
% Find average power per month
ProgressBar = waitbar(0,'Running Per Month Analysis'); tic

TempSites = Sites; TempSites(78) = []; TempSites(20) = [];
```

```
Timestamps = zeros(12,1); Timestamps(1) = 734869;  Timestamps(2) = 734900;
Timestamps(3) = 734929;   Timestamps(4) = 734960;  Timestamps(5) = 734990;
Timestamps(6) = 735021;   Timestamps(7) = 735051;  Timestamps(8) = 735082;
Timestamps(9) = 735113;   Timestamps(10) = 735143; Timestamps(11) = 735174;
Timestamps(12) = 735204;

clear Bins
% Gather data into Bin structure
% The data structure will look like this:
%  Bins.Valid        - Is this site a valid site?
%  Bins.Power()      - Cumulative Power generated from the data normalized by nameplate
%  Bins.Timestamps() - Unique Timestamps in the dataset

for sitenum = 1:length(TempSites)
  % Convert timestamps to a datenum, but only sorted by month
  TempTimestamps = datenum(datestr(TempSites(sitenum).Timestamps,'mm'),'mm');
  Bins(sitenum).Valid = 1;
  Bins(sitenum).Timestamps = unique(TempTimestamps);

  % Only include data with 12 months of data
  if length(TempTimestamps) < 12
    Bins(sitenum).Valid = 0;
  else
    % Match the power generated with the timestamps
    for count = 1:12
      % Find the values that match the timestamps
      TimestampIndex = find(Timestamps(count) == TempTimestamps);
      Bins(sitenum).Power(count) =
mean(TempSites(sitenum).Power(TimestampIndex))/TempSites(sitenum).Nameplate*100;
      if Bins(sitenum).Power(count) <= 2 || Bins(sitenum).Power(count) > 130 ||
isnan(Bins(sitenum).Power(count))
        Bins(sitenum).Valid = 0;
      end
    end
  end

  % Fix the row/column being swapped issue. Not sure why this happens.
  [row col] = size(Bins(sitenum).Power);
  if col > 1
    Bins(sitenum).Power = Bins(sitenum).Power';
  end

  % Only plot valid entries
  if Bins(sitenum).Valid == 0
    plot(Bins(sitenum).Timestamps,Bins(sitenum).Power);
    fprintf('%d Name:%s Nameplate:%d Max
Power:%d\n',sitenum,TempSites(sitenum).Name{:},TempSites(sitenum).Nameplate,max(Bins(sitenum).Po
wer)*TempSites(sitenum).Nameplate);
    hold on
  end
  waitbar(sitenum/length(TempSites),ProgressBar,[num2str(sitenum) ' out of '
num2str(length(TempSites))]);
```

```
end

%Format Axes
axis tight
ylim([0 50])
xData = linspace(datenum('01','mm')+2,datenum('12','mm')+2,12);
set(gca,'XTick',xData)
datetick('x','mmm','keepticks')
ylabel('Average % of Nameplate Generation')
xlabel('Month')

% Add all valid power readings and average them
ValidBins = 0;
FinalBins = zeros(12,1);
for count = 1:length(Bins)
  ValidBins = ValidBins + Bins(count).Valid;
  if Bins(count).Valid
    FinalBins = FinalBins + Bins(count).Power;
  end
end

bar(Timestamps,FinalBins/ValidBins);

disp([num2str(ValidBins) ' out of ' num2str(length(TempSites)) ' are valid']);

% Format Axes
% I have to add 2 so that it spaces these out correctly. Otherwise, it goes
% Jan 1, Jan 31, March 1, etc... when it should be Feb on the second month.
xData = linspace(datenum('01','mm')+2,datenum('12','mm')+2,12);
set(gca,'XTick',xData)
datetick('x','mmm','keepticks')
ylabel('Average % of Nameplate Generation')
xlabel('Month')
ylim([0 50])

close(ProgressBar); toc
```

```
% PerMonthAnalysis.m
% ver 1.00 MR
% Find average power per month

ProgressBar = waitbar(0,'Running Per Month Analysis');
tic

TempSites = Sites;
TempSites(78) = [];
TempSites(20) = [];

% Get unique timestamps
for sitenum = 1:length(TempSites)
  Timestamps = zeros(length(TempSites(sitenum).Timestamps),1);
```

```
  Timestamps = datenum(datestr(TempSites(sitenum).Timestamps,'mm'),'mm');

  Power = zeros(length(TempSites(sitenum).Power),1);
  Power = TempSites(sitenum).Power/TempSites(sitenum).Nameplate;

  UniqueTimestamps = unique(Timestamps);
  AveragePower = zeros(length(UniqueTimestamps),1);

  for count = 1:length(UniqueTimestamps)
    AveragePower(count) = mean(Power(find(Timestamps == UniqueTimestamps(count))))*100;
  end

  %Plot the data
  if max(AveragePower) <= 50 %Don't plot results with more than 50% Nameplate generation
    plot(UniqueTimestamps,AveragePower);
  end
  % Format the axes according to dates
  axis tight
  % I have to add 2 so that it spaces these out correctly. Otherwise, it goes
  % Jan 1, Jan 31, March 1, etc... when it should be Feb on the second month.
  xData = linspace(datenum('01','mm')+2,datenum('12','mm')+2,12);
  set(gca,'XTick',xData)
  datetick('x','mmm','keepticks')
  ylabel('Average % of Nameplate Generation')
  xlabel('Month')
  hold on
  waitbar(sitenum/length(TempSites),ProgressBar,[num2str(sitenum) ' out of '
num2str(length(TempSites))]);
end

close(ProgressBar);

toc
```

```
% PerDayAnalysis.m
% ver 1.00 MR
% Find the amount of power generated per day

ProgressBar = waitbar(0,'Running Per Day Analysis');
tic

TempSites = Sites;
TempSites(78) = [];
TempSites(20) = [];

% Get unique timestamps
for sitenum = 1:length(TempSites)
  Timestamps = zeros(length(TempSites(sitenum).Timestamps),1);
  Timestamps = datenum(datestr(TempSites(sitenum).Timestamps,'HH:MM:SS'));

  Power = zeros(length(TempSites(sitenum).Power),1);
```

```
  Power = TempSites(sitenum).Power/TempSites(sitenum).Nameplate;

  UniqueTimestamps = unique(Timestamps);
  AveragePower = zeros(length(UniqueTimestamps),1);

  for count = 1:length(UniqueTimestamps)
    AveragePower(count) = mean(Power(find(Timestamps == UniqueTimestamps(count))))*100;
  end

  % Format the axes according to dates
  startDate = datenum(min(UniqueTimestamps));
  endDate = datenum(max(UniqueTimestamps));
  xData = linspace(startDate,endDate,96);

  %Plot the data
  if max(AveragePower) <= 120 %Don't plot results with more than 120% Nameplate generation
    plot(xData,AveragePower);
  end
  axis tight
  set(gca,'XTick',xData)
  datetick('x','HH','keepticks')
  ylabel('Average % of Nameplate Generation')
  xlabel('Time of day (hours)')
  hold on
  waitbar(sitenum/length(TempSites),ProgressBar,[num2str(sitenum) ' out of '
num2str(length(TempSites))]);
end
close(ProgressBar);

toc
```

```
% CompareCSINameplateToDECK.m
% ver 1.00 MR
% Compare CSI Nameplate to DECK Nameplate
hold on
for count = 1:length(Sites)
  disp([num2str(Sites(count).Nameplate) ' ' num2str(Sites(count).CSINameplate) ' '
num2str(Sites(count).CECPTC) ' ' num2str(ErrorPercent(count))]);
  if Sites(count).CSINameplate * 0.95 < Sites(count).Nameplate && Sites(count).Nameplate <
Sites(count).CSINameplate * 1.05
    ErrorPercent(count) = 100*(Sites(count).Nameplate-
Sites(count).CSINameplate)/max(Sites(count).Nameplate,Sites(count).CSINameplate);
    plot(Sites(count).Nameplate,Sites(count).CSINameplate,'bo');
  elseif Sites(count).CECPTC * 0.95 < Sites(count).Nameplate && Sites(count).Nameplate <
Sites(count).CECPTC * 1.05
    ErrorPercent(count) = 100*(Sites(count).Nameplate-
Sites(count).CECPTC)/max(Sites(count).Nameplate,Sites(count).CECPTC);
    plot(Sites(count).Nameplate,Sites(count).CECPTC,'go');
  else
    ErrorPercent(count) = 100*(Sites(count).Nameplate-
Sites(count).CSINameplate)/max(Sites(count).Nameplate,Sites(count).CSINameplate);
    plot(Sites(count).Nameplate,Sites(count).CECPTC,'ro');
```

```
  end
end

% Remove Sites that are not within 5% of either CECPTC or DC Nameplate
for count = length(Sites):-1:1
  if abs(ErrorPercent(count)) > 5
    Sites(count) = [];
  end
end

ylabel('CSI Nameplate');
xlabel('DECK Nameplate');
title('Nameplate customer provided to DECK vs. CSI');
hold on
plot([1:1200],[1:1200]);
plot([1:1200],[1:1200].*1.1);
plot([1:1200],[1:1200]./1.1);
```

```
% RemoveIncorrectNameplates.m
% ver 1.00 MR

% Remove incorrect nameplates
SiteIndeces = zeros(1,length(Sites));
for count = 1:length(Sites)
  if ~isempty(Sites(count).CSINameplate)
    SiteIndeces(count) = 1;
  end
end

CSISites = Sites;
CSISites(~SiteIndeces) = [];

NameplateError = zeros(length(CSISites),1);
CECPTCError = zeros(length(CSISites),1);
TotalError = zeros(length(CSISites),1);
VerifiedIndeces = zeros(length(CSISites),1);

for count = 1:length(CSISites)
  NameplateError(count) = (CSISites(count).Nameplate -
CSISites(count).CSINameplate)./CSISites(count).CSINameplate*100;
  CECPTCError(count)    = (CSISites(count).Nameplate -
CSISites(count).CECPTC)./CSISites(count).CECPTC*100;
  TotalError(count)     = min(NameplateError(count), CECPTCError(count));
  if abs(TotalError(count)) < 5
    CSISites(count).Nameplate = CSISites(count).CSINameplate;
    VerifiedIndeces(count) = 1;
  end
end

VerifiedSites = CSISites(~~VerifiedIndeces);

plot(Sites(20).Timestamps-734500,Sites(20).Power/Sites(20).Nameplate*100)
```

```
hold on; plot(Sites(21).Timestamps-734500,Sites(21).Power/Sites(21).Nameplate*100,'r')
hold on; plot(Sites(35).Timestamps-734500,Sites(35).Power/Sites(35).Nameplate*100,'k')
```

```
% NameplateCSIVerification.m
% ver 1.00 MR

% Compare sites nameplates
DECK_bin     = [];
CECPTC_bin   = [];
Neither_bin  = [];
ErrorPercent = [];
for count = 1:length(Sites)
  CSI    = Sites(count).CSINameplate;
  DECK   = Sites(count).Nameplate;
  CECPTC = Sites(count).CECPTC;

  if     CSI    > 0.98*DECK && CSI    < 1.02*DECK
    DECK_bin     = [DECK_bin DECK];
  elseif CECPTC > 0.98*DECK && CECPTC < 1.02*DECK
    CECPTC_bin   = [CECPTC_bin DECK];
  else
    Neither_bin  = [Neither_bin DECK];
    ErrorPercent = [ErrorPercent (DECK-CSI)/max(DECK,CSI)*100];
  end
end

disp(['Match DECK Nameplate: ' num2str(length(DECK_bin))]);
disp(['Match CECPTC        : ' num2str(length(CECPTC_bin))]);
disp(['Match Neither       : ' num2str(length(Neither_bin))]);
disp(['Average Error       : ' num2str(mean(ErrorPercent))]);
```

```
% ImportSolarData.m
% ver 1.00 MR

% Import Solar Data
clear all

ImportRawData
CleanData
AddFields
```

```
% ImportRawData.m
% ver 1.00 MR

% Import Raw Data
% This function imports data from the data files to an organized MATLAB
% data structure. This makes it easier to work with the data later on.

RootDir = ['D:\Research\2012_06_12_DataDump\'];
```

```
%RootDir = ['D:\Research\2012_05_10_DataDump\'];

% Get directory contents
device_contents = ls([RootDir 'inverter_data\']);

% Get device file contents
fid = fopen([RootDir device_contents 'locations.txt']);
Temp = textscan(fid, '%s%s%s%s%s%s%s%s%s%s%s%s','Delimiter','|');
fclose(fid);

for count = 1:length(Temp{1})
  Sites(count).Id              = str2num(Temp{1}{count});
  Sites(count).Name            = Temp{2}{count};
  Sites(count).Device_id       = str2num(Temp{3}{count});
  Sites(count).Lat             = str2num(Temp{4}{count});
  Sites(count).Long            = str2num(Temp{5}{count});
  Sites(count).State           = Temp{7}{count};
  Sites(count).Device_type     = Temp{8}{count};
  Sites(count).Device_type_id  = str2num(Temp{9}{count});
  Sites(count).NumPoints       = str2num(Temp{10}{count});
  Sites(count).Nameplate       = str2num(Temp{11}{count});
  Sites(count).CSI_ID          = Temp{12}{count};
end
Sites(1) = [];

% Remove any sites with less than 5760 points (60 days) or less than 1kW
% DC Nameplate rating
TempRemovalSites = [];
for count = 1:length(Sites)
  if Sites(count).NumPoints < 5760 || Sites(count).Nameplate < 1
    TempRemovalSites = [count TempRemovalSites];
  %elseif strcmp(Sites(count).CSI_ID,'NA') % Comment this out if you need all data
  %  TempRemovalSites = [count TempRemovalSites];
  end
end
Sites(TempRemovalSites) = [];

% Import data into data structure
ProgressBar = waitbar(0,'Importing Data');
for count = 1:length(Sites)
  TempData = importdata([RootDir 'device_data\' num2str(Sites(count).Id) '_'
num2str(Sites(count).Device_id) '.csv'],',');
  Sites(count).Timestamps = datenum(TempData.textdata(2:end,1),'yyyy-mm-dd HH:MM');
  Sites(count).Timestrings = datestr(Sites(count).Timestamps,'yyyymmddHHMM');
  for count2 = 1:length(Sites(count).Timestrings)
    Sites(count).Timenum(count2) = str2num(Sites(count).Timestrings(count2,:));
  end
  Sites(count).Power = TempData.data(:,1);
  waitbar(count/length(Sites),ProgressBar,[num2str(count) ' out of ' num2str(length(Sites))]);
  clear TempData
end

clear Temp
```

```
clear TempRemovalSites
clear sitenum
clear count
clear device_contents
clear RootDir
clear Temp
close(ProgressBar);
```

```
% ImportPredictedPowerData.m
% ver 3.00 MR

% Import Predicted Power Data
% This function imports data from the data files to an organized MATLAB
% data structure. This makes it easier to work with the data later on.
tic
RootDir = ['D:\Research\2012_07_20_DataDump\'];

% Get directory contents
device_contents = ls([RootDir 'power_data\']);

% Get device file contents
fid = fopen([RootDir 'locations.txt']);
Temp = textscan(fid, '%s%s%s%s%s%s%s%s','Delimiter','|');
fclose(fid);

for count = 1:length(Temp{1})
  Sites(count).Location_id  = str2num(Temp{1}{count});
  Sites(count).Name         = Temp{2}{count};
  Sites(count).Lat          = str2num(Temp{3}{count});
  Sites(count).Long         = str2num(Temp{4}{count});
  Sites(count).Address      = Temp{5}{count};
  Sites(count).State        = Temp{6}{count};
  Sites(count).Coefficient  = str2num(Temp{7}{count});
  Sites(count).Nameplate    = str2num(Temp{8}{count});
end
Sites(1) = [];

ProgressBar = waitbar(0,'Importing Data');
MarkedForDeletion = [];
for count = 1:length(Sites)
  % Import power data into data structure
  TempData = importdata([RootDir 'power_data\' num2str(Sites(count).Location_id) '.csv'],',');
  Sites(count).Timestamps = datenum(TempData.textdata(2:end,1),'yyyy-mm-dd HH:MM');
  Sites(count).Timestrings = datestr(Sites(count).Timestamps,'yyyymmddHHMM');
  for count2 = 1:length(Sites(count).Timestrings)
    Sites(count).Timenum(count2) = str2num(Sites(count).Timestrings(count2,:));
  end
  Sites(count).Power = TempData.data(:,1);

  % Import weather data into data structure
  % Get device file contents
  fid = fopen([RootDir 'weather_data\' num2str(Sites(count).Location_id) '.csv']);
```

```
Temp = textscan(fid, '%s%s%s%s%s%s','Delimiter',',');
fclose(fid);

clear WeatherTimes
if length(Temp{2}) > 1
 for count2=2:length(Temp{1})
  WeatherTimes(count2-1) = datenum(Temp{1}{count2},'yyyy-mm-dd HH:MM');
 end

 Sites(count).Irradiance        = zeros(length(Temp{2})-1,1);
 Sites(count).CellTemperature   = zeros(length(Temp{3})-1,1);
 Sites(count).Temperature       = zeros(length(Temp{4})-1,1);
 Sites(count).CellTemperature2  = zeros(length(Temp{5})-1,1);
 if strcmp(Temp{2}(2),"""")==1 || strcmp(Temp{3}(2),"""")==1 || strcmp(Temp{4}(2),"""")==1
  MarkedForDeletion = [MarkedForDeletion count];
 else
  for count2=2:length(Sites(count).Irradiance)
   if strcmp(Temp{2}{count2},"""")==0
    Sites(count).Irradiance(count2-1)        = str2num(Temp{2}{count2});
   else
    MarkedForDeletion = [MarkedForDeletion count];
   end
  end
  for count2=2:length(Sites(count).CellTemperature)
   if strcmp(Temp{3}{count2},"""")==0
    Sites(count).CellTemperature(count2-1)   = str2num(Temp{3}{count2});
   else
    MarkedForDeletion = [MarkedForDeletion count];
   end
  end
  for count2=2:length(Sites(count).Temperature)
   if strcmp(Temp{4}(count2),"""")==0
    Sites(count).Temperature(count2-1)       = str2num(Temp{4}{count2});
   else
    MarkedForDeletion = [MarkedForDeletion count];
   end
  end
  % If there's no second cell temp, that's fine
  for count2=2:length(Sites(count).CellTemperature2)
   if strcmp(Temp{5}(2),"""")~=1
    Sites(count).CellTemperature2(count2-1)  = str2num(Temp{5}{count2});
   end
  end
 end
else
 WeatherTimes = [];
 MarkedForDeletion = [MarkedForDeletion count];
end
waitbar(count/length(Sites),ProgressBar,[num2str(count) ' out of ' num2str(length(Sites))]);
clear TempData
clear Temp

% Consolidate the various time frames
[C,IA,IB] = intersect(Sites(count).Timestamps,WeatherTimes);
```

```
  Sites(count).Timestamps       = Sites(count).Timestamps(IA);
  Sites(count).Timestrings      = Sites(count).Timestrings(IA);
  Sites(count).Timenum          = Sites(count).Timenum(IA);
  Sites(count).Power            = Sites(count).Power(IA);
  Sites(count).Irradiance       = Sites(count).Irradiance(IB);
  Sites(count).CellTemperature  = Sites(count).CellTemperature(IB);
  Sites(count).CellTemperature2 = Sites(count).CellTemperature2(IB);
  Sites(count).Temperature      = Sites(count).Temperature(IB);
end
Sites(MarkedForDeletion) = [];

clear Temp
clear TempRemovalSites
clear sitenum
clear count
clear device_contents
clear RootDir
clear Temp
close(ProgressBar);
toc
```

```
% ImportCSISiteInfo.m
% ver 1.00 MR

% Import CSI Data
ProgressBar = waitbar(0,'Clearing Variables');
clear CSI_Data_Raw
clear CSI_Data

waitbar(0,ProgressBar,'Importing Raw Data');

%Import Raw Data from file
CSI_Filepath = 'C:\Research\CSI_Data\WorkingDataSet_6-13-2012.txt';
fid = fopen(CSI_Filepath);
CSI_Data_Raw = textscan(fid,
'%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s','Deli
miter','|');
fclose(fid);

% Add data for each site
for count = 1:length(CSI_Data_Raw{1})
  CSI_Data(count).CSI_Number        = CSI_Data_Raw{1}{count};
  CSI_Data(count).ProgramAdmin      = CSI_Data_Raw{2}{count};
  CSI_Data(count).Program           = CSI_Data_Raw{3}{count};
  CSI_Data(count).IncentiveDesign   = CSI_Data_Raw{4}{count};
  CSI_Data(count).IncentiveType     = CSI_Data_Raw{5}{count};
  CSI_Data(count).IncentiveStep     = CSI_Data_Raw{6}{count};
  CSI_Data(count).IncentiveAmount   = str2num(CSI_Data_Raw{7 }{count});
  CSI_Data(count).TotalCost         = str2num(CSI_Data_Raw{8 }{count});
  CSI_Data(count).Nameplate         = str2num(CSI_Data_Raw{9 }{count});
```

```
CSI_Data(count).CECPTC            = str2num(CSI_Data_Raw{10}{count});
CSI_Data(count).DesignFactor      = str2num(CSI_Data_Raw{11}{count});
CSI_Data(count).CSIRating         = str2num(CSI_Data_Raw{12}{count});
CSI_Data(count).ApplicationStatus = CSI_Data_Raw{13}{count};
CSI_Data(count).Contractor        = CSI_Data_Raw{45}{count};
CSI_Data(count).Seller            = CSI_Data_Raw{47}{count};
CSI_Data(count).ThirdParty        = CSI_Data_Raw{48}{count};
CSI_Data(count).InstalledStatus   = CSI_Data_Raw{100}{count};
CSI_Data(count).PBI               = CSI_Data_Raw{101}{count};
CSI_Data(count).MonitoringProvider= CSI_Data_Raw{102}{count};
CSI_Data(count).PDPProvider       = CSI_Data_Raw{103}{count};

% Add Solar Panel Information
CSI_Data(count).Panels(1).Manufacturer = CSI_Data_Raw{49}{count};
CSI_Data(count).Panels(1).Model        = CSI_Data_Raw{56}{count};
CSI_Data(count).Panels(1).Number       = CSI_Data_Raw{63}{count};
if ~isempty(CSI_Data_Raw{50}{count})
  CSI_Data(count).Panels(2).Manufacturer = CSI_Data_Raw{50}{count};
  CSI_Data(count).Panels(2).Model        = CSI_Data_Raw{57}{count};
  CSI_Data(count).Panels(2).Number       = CSI_Data_Raw{64}{count};
  if ~isempty(CSI_Data_Raw{51}{count})
    CSI_Data(count).Panels(3).Manufacturer = CSI_Data_Raw{51}{count};
    CSI_Data(count).Panels(3).Model        = CSI_Data_Raw{58}{count};
    CSI_Data(count).Panels(3).Number       = CSI_Data_Raw{65}{count};
    if ~isempty(CSI_Data_Raw{52}{count})
      CSI_Data(count).Panels(4).Manufacturer = CSI_Data_Raw{52}{count};
      CSI_Data(count).Panels(4).Model        = CSI_Data_Raw{59}{count};
      CSI_Data(count).Panels(4).Number       = CSI_Data_Raw{66}{count};
      if ~isempty(CSI_Data_Raw{53}{count})
        CSI_Data(count).Panels(5).Manufacturer = CSI_Data_Raw{53}{count};
        CSI_Data(count).Panels(5).Model        = CSI_Data_Raw{60}{count};
        CSI_Data(count).Panels(5).Number       = CSI_Data_Raw{67}{count};
        if ~isempty(CSI_Data_Raw{54}{count})
          CSI_Data(count).Panels(6).Manufacturer = CSI_Data_Raw{54}{count};
          CSI_Data(count).Panels(6).Model        = CSI_Data_Raw{61}{count};
          CSI_Data(count).Panels(6).Number       = CSI_Data_Raw{68}{count};
          if ~isempty(CSI_Data_Raw{55}{count})
            CSI_Data(count).Panels(7).Manufacturer = CSI_Data_Raw{55}{count};
            CSI_Data(count).Panels(7).Model        = CSI_Data_Raw{62}{count};
            CSI_Data(count).Panels(7).Number       = CSI_Data_Raw{69}{count};
          end
        end
      end
    end
  end
end

% Add Inverter Information
CSI_Data(count).Inverters(1).Manufacturer = CSI_Data_Raw{70}{count};
CSI_Data(count).Inverters(1).Model        = CSI_Data_Raw{80}{count};
CSI_Data(count).Inverters(1).Number       = CSI_Data_Raw{90}{count};
if ~isempty(CSI_Data_Raw{71}{count})
  CSI_Data(count).Inverters(2).Manufacturer = CSI_Data_Raw{71}{count};
  CSI_Data(count).Inverters(2).Model        = CSI_Data_Raw{81}{count};
```

```
    CSI_Data(count).Inverters(2).Number       = CSI_Data_Raw{91}{count};
    if ~isempty(CSI_Data_Raw{72}{count})
      CSI_Data(count).Inverters(3).Manufacturer = CSI_Data_Raw{72}{count};
      CSI_Data(count).Inverters(3).Model        = CSI_Data_Raw{82}{count};
      CSI_Data(count).Inverters(3).Number       = CSI_Data_Raw{92}{count};
      if ~isempty(CSI_Data_Raw{73}{count})
        CSI_Data(count).Inverters(4).Manufacturer = CSI_Data_Raw{73}{count};
        CSI_Data(count).Inverters(4).Model        = CSI_Data_Raw{83}{count};
        CSI_Data(count).Inverters(4).Number       = CSI_Data_Raw{93}{count};
        if ~isempty(CSI_Data_Raw{74}{count})
          CSI_Data(count).Inverters(5).Manufacturer = CSI_Data_Raw{74}{count};
          CSI_Data(count).Inverters(5).Model        = CSI_Data_Raw{84}{count};
          CSI_Data(count).Inverters(5).Number       = CSI_Data_Raw{94}{count};
          if ~isempty(CSI_Data_Raw{75}{count})
            CSI_Data(count).Inverters(6).Manufacturer = CSI_Data_Raw{75}{count};
            CSI_Data(count).Inverters(6).Model        = CSI_Data_Raw{85}{count};
            CSI_Data(count).Inverters(6).Number       = CSI_Data_Raw{95}{count};
            if ~isempty(CSI_Data_Raw{76}{count})
              CSI_Data(count).Inverters(7).Manufacturer = CSI_Data_Raw{76}{count};
              CSI_Data(count).Inverters(7).Model        = CSI_Data_Raw{86}{count};
              CSI_Data(count).Inverters(7).Number       = CSI_Data_Raw{96}{count};
              if ~isempty(CSI_Data_Raw{77}{count})
                CSI_Data(count).Inverters(8).Manufacturer = CSI_Data_Raw{77}{count};
                CSI_Data(count).Inverters(8).Model        = CSI_Data_Raw{87}{count};
                CSI_Data(count).Inverters(8).Number       = CSI_Data_Raw{97}{count};
                if ~isempty(CSI_Data_Raw{78}{count})
                  CSI_Data(count).Inverters(9).Manufacturer = CSI_Data_Raw{78}{count};
                  CSI_Data(count).Inverters(9).Model        = CSI_Data_Raw{88}{count};
                  CSI_Data(count).Inverters(9).Number       = CSI_Data_Raw{98}{count};
                  if ~isempty(CSI_Data_Raw{79}{count})
                    CSI_Data(count).Inverters(10).Manufacturer = CSI_Data_Raw{79}{count};
                    CSI_Data(count).Inverters(10).Model        = CSI_Data_Raw{89}{count};
                    CSI_Data(count).Inverters(10).Number       = CSI_Data_Raw{99}{count};
                  end
                end
              end
            end
          end
        end
      end
    end
  end
  if mod(count,1000)==0
    waitbar(count/length(CSI_Data_Raw{1}),ProgressBar,[num2str(count) ' out of '
num2str(length(CSI_Data_Raw{1}))]);
  end
end

%Delete the first element since it is just column headers
CSI_Data(1) = [];

clear CSI_Data_Raw

% Add CSI Data to the Sites variable so that now the data is
```

```
% cross-referenced between CSI and DECK
for SiteCount = 1:length(Sites)
  found = 0;
  % Find the site in the CSI Data
  for CSICount = 1:length(CSI_Data)
    if strcmp(CSI_Data(CSICount).CSI_Number,Sites(SiteCount).CSI_ID)
      found = 1;
      Sites(SiteCount).Inverters          = CSI_Data(CSICount).Inverters;
      Sites(SiteCount).Panels             = CSI_Data(CSICount).Panels;
      Sites(SiteCount).ProgramAdmin       = CSI_Data(CSICount).ProgramAdmin;
      Sites(SiteCount).Program            = CSI_Data(CSICount).Program;
      Sites(SiteCount).IncentiveDesign    = CSI_Data(CSICount).IncentiveDesign;
      Sites(SiteCount).IncentiveType      = CSI_Data(CSICount).IncentiveType;
      Sites(SiteCount).IncentiveStep      = CSI_Data(CSICount).IncentiveStep;
      Sites(SiteCount).IncentiveAmount    = CSI_Data(CSICount).IncentiveAmount;
      Sites(SiteCount).TotalCost          = CSI_Data(CSICount).TotalCost;
      Sites(SiteCount).CSINameplate       = CSI_Data(CSICount).Nameplate;
      Sites(SiteCount).CECPTC             = CSI_Data(CSICount).CECPTC;
      Sites(SiteCount).DesignFactor       = CSI_Data(CSICount).DesignFactor;
      Sites(SiteCount).CSIRating          = CSI_Data(CSICount).CSIRating;
      Sites(SiteCount).ApplicationStatus  = CSI_Data(CSICount).ApplicationStatus;
      Sites(SiteCount).Contractor         = CSI_Data(CSICount).Contractor;
      Sites(SiteCount).Seller             = CSI_Data(CSICount).Seller;
      Sites(SiteCount).ThirdParty         = CSI_Data(CSICount).ThirdParty;
      Sites(SiteCount).InstalledStatus    = CSI_Data(CSICount).InstalledStatus;
      Sites(SiteCount).PBI                = CSI_Data(CSICount).PBI;
      Sites(SiteCount).MonitoringProvider = CSI_Data(CSICount).MonitoringProvider;
      Sites(SiteCount).PDPProvider        = CSI_Data(CSICount).PDPProvider;
    end
  end
  if found == 0
    disp([Sites(SiteCount).CSI_ID ' ' Sites(SiteCount).Name]);
  end
end

close(ProgressBar);
```

```
% ImportCSIData.m
% ver 1.00 MR

% Import CSI Data
ImportSolarData
ImportCSISiteInfo
```

```
% AddFields.m
% ver 1.00 MR

% Add fields to the data so that they do not have to be calculated later
% Specifically, add:
%  Average percent generation per day
```

```
for count = 1:length(Sites)
  Sites(count).AvgPower = mean(Sites(count).Power)/Sites(count).Nameplate;
  X = sort(Sites(count).Power,'descend');
  Sites(count).CalcNameplate = median(X(1:ceil(length(X)*.001)));
end
```
