Study on Impacts of Varying Weather Patterns upon Circuits with Photovoltaic (PV) Penetration

A Major Qualifying Project Report submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science in Electrical and Computer Engineering (ECE)

Worcester Polytechnic Institute,
Worcester, MA, 01609, USA

Submitted by: Barry Aslanian Jr, Christian Curll, Matthew Scherrer, Markus Zimmermann

Advisors: Professor Maqsood Mughal and Professor Sundari Ramabholta
Sponsor: Umair Zia (Eversource)

Table of Contents

List of Figures

Abstract

The goal of the project is to model and create a system to show the effects of cloud coverage on an installed PV system and those impacts on the electrical grid. When a cloud disrupts a PV array by blocking the sun, the electrical output varies in reference to the initial output. This variation ripples across the grid and can impact other systems along the power line. Initially, to adjust for this phenomenon a test is done at an output lower than initially expected. While this method is sufficient it does not consider all preventable scenarios. We analyzed irradiance data for areas provided by Eversource Energy with the electrical output of their system and made a model to indicate a flicker moment on the grid and compensate accordingly.

With more and more customers, including large businesses, turning to photovoltaics, as an additional source of electrical energy to power their homes and buildings, any disruption can cause issues on the grid. Disruption causes electrical instability and asset degradation, increasing operating costs. The disruption is caused by PV's susceptibility to weather conditions such as cloud cover. High PV penetration of distributed PV systems is causing technical problems such as reverse power flows and voltage instability in distribution feeders that affects the operation of the bulk transmission system.

Our sponsor, Eversource, wants to determine a better method for calculating a dip in electrical production when a cloud covers an array of solar panels for a period of time. By predicting the effects of flicker in a simulation, power distributors can take

preparatory measures to reduce the strain on the grid, lowering the operating cycles of corrective equipment and therefore lowering operating costs. Additionally, with more and more photovoltaic systems entering the grid, there is growing concern for the longevity of corrective assets. Proper knowledge of the lifecycle of equipment allows for timely replacements that prevent inconveniences to customers.

Acknowledgements

We'd like to show our appreciation to Kaveh Rahimi and Robert Broadwater of Virginia Polytechnic Institute for sharing their work on cloud motion simulations with us. Their paper titled Computation of Voltage Flicker With Cloud Motion Simulator was extremely helpful for us to begin our project.

We'd also like to thank Umair Zia, Everett Hall, Thaer Qunias, and Dan Lewis of Eversource Energy for providing us with the necessary materials and support to work on this project. Eversource Energy has been a generous sponsor for this project through their donation of funding and data.

Introduction

Eversource Energy, hereafter called Eversource, is a publicly traded electric utility company based in the New England region. Established in 2015 as Eversource, the company came out of a merger between Northeast Utilities and NSTAR to provide service to the greater Boston region. To reach their goal, Eversource has a mission to deliver "reliable energy and superior customer service, in community service and leadership, and in doing the right thing for their customers, co-workers, shareholders and the environment." Eversource not only has a stake in the electrical grid, the company also provides gas for heating and cooking to numerous homes, and more recently, water.

Goals and Objectives

The goal of the project is to implement a fully functional interface with Synergi, a distributed feeder simulation software that Eversource Energy uses to analyze load flow for their circuits. The current method used by eversource is inefficient and does not properly simulate voltage flicker. Our project goal is to make Synergi more impactful in its use by creating an add-on model that will compute PV output power and a voltage flicker severity index using IEEE 1453-2015 standard and high resolution (5 minute) data from a pyranometer located near the PV array.

Background

Photovoltaics

Photovoltaics (PV), commonly referred to as solar panels, are typically first thought of when it comes to renewable energy. To create a PV array, smaller photovoltaic modules are attached together to cover a larger area. PV arrays can replace traditional methods of power generation such as coal-fired and natural gas powered plants, by producing electricity with a much smaller carbon footprint (Evans). More recently, the efficiency of solar panels have reached up to 23%, converting 23 percent of the sun's solar energy into electricity. This whole process is dependent on one huge factor, the sun. Any disruption reduces the amount produced and transferable to the grid (Energy Sage). Renewable energy accounted for about 17% of the United States electricity generation in 2018, making flicker and equipment wear pressing issues (US Energy Information Administration).

Distributed PV power generation systems are being rapidly deployed, causing technical problems to utility sectors worldwide as reverse power flows and voltage fluctuations in distribution feeders, and real and reactive power transients that affect the operation of the bulk transmission system. Traditional voltage control devices such as line voltage regulators and switched capacitor banks can alleviate slow moving fluctuations, but these devices need to operate more frequently than usual when PV generation fluctuates due to cloud cover. The output of a PV system can vary from

100% to 20% in seconds. A concern of the utility sector is that such frequent operations will impact the life expectancy of these voltage control devices.

Tap Changers

Tap changers are a voltage control device that allow transformers to hold variable turn ratios in discrete steps. There are two types of tap changers, on-load and off-load tap changers. On-load transformers are used to make changes to the turn ratio while the circuit is in use, while off-load requires a disconnection from the circuit before changing. On-load tap changers are typically found in power circuits, as it allows for corrections to the voltage to occur throughout the day based on load needs(Pitt). A tap changer is only expected to have a limited number of tap changes in its lifetime, so more frequent corrections lower its life expectancy. To fully understand and address these problems, utility companies like Eversource are seeking solutions to this problem.

In a meeting including Eversource Distribution Engineering Director Umair Zia in October, 2019 concern was voiced in regards to the impact of PV penetration on the performance of the electrical grid. As PV penetration into the grid is increasing, concern arose for the longevity of the tap changers. If PV penetration increases the number of tap changing operations per day, the life span prediction of the equipment needs to be re-evaluated so replacement can happen before equipment failure, which can cause an electric power outage.

Eversource Energy

Eversource is a publicly traded company providing electrical, gas, and water services to houses and businesses in New Hampshire, Connecticut and Massachusetts. It is the largest utility in New England, delivering energy to approximately 3.7 million residents.

As an electric utility, Eversource has a power quality standard regarding the interconnection of distributed energy resources (DER). The standard is to keep voltage levels on the grid within ±2% of the nominal voltage level (Eversource Energy). High PV penetration into the grid is making it difficult for Eversource to meet this standard due to the intermittent nature of PV systems caused by cloud motion above arrays. As PV penetration increases, the severity of the problems associated with PV intermittency will also increase.

The renewable market is expected to grow as time goes on. With 80 financial incentives in the state of Massachusetts alone, Eversource customers are encouraged by the government to add more PV sources to Eversource's networks (N.C. Clean Technology Center). As of December 31, 2018, Eversource has nearly 700 MW of PV power interconnected with their network (Eversource Energy).

Currently, the engineers simulate flicker by running a simulation at either end of capacity, from 100% to 5% (Eversource Energy). These run times are two extremes, not what would actually happen if a cloud covers a PV array for an extended period of time.

This is not effective as a one runtime simulation, as more clouds could roll over and reduce the output of the PV system.

Literature Review

Voltage Flicker

Voltage flicker is defined by IEEE 1453-2015 as the subjective impression of fluctuating luminance caused by voltage fluctuations (*IEEE Recommended Practice for the Analysis of Fluctuating Installations on Power Systems.* 1-74). This phenomena can be described as a change in voltage quick enough to not be corrected by stabilizing systems, but noticeable to the human eye via an incandescent light bulb. Although flicker may not be blatantly visible, it can still cause irritation.

The relationship between irritability and visibility of voltage flicker is called the "GE curve" and was published by General Electric from a collection of previous flicker studies. The GE curve consists of two curves, one showing the borderline of irritation and the other showing the borderline of visibility. The perceptibility curves were developed by scientists using human subjects in front of various flickering lamps. The GE curve is still used in the power industry to impose flicker limits on industrial customers connected to the grid (*IEEE Recommended Practice for the Analysis of Fluctuating Installations on Power Systems.* 1-74). Eversource's voltage flicker limit, for example, is that voltage flicker should not exceed ±2% (Eversource Energy).

The most recent measurement method used to calculate voltage flicker is from the IEEE 1453-2015 standard, IEEE Recommended Practice for the Analysis of Fluctuating Installations on Power Systems. The following flicker computation method is adapted from the IEC 61000-4-15:2010 standard (*IEEE Draft Recommended Practice -- Adoption of IEC 61000-4-15:2010, Electromagnetic Compatibility (EMC)---Testing and Measurement Techniques---Flickermeter---Functional and Design Specifications.* 1-53). The computation method is done in five steps that are briefly described by a block diagram.

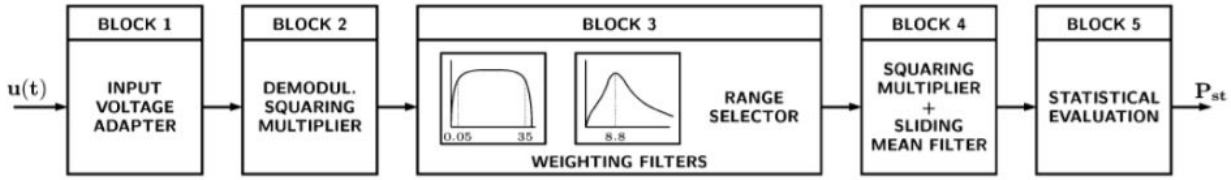

Figure 1: Block diagram of flicker meter from IEEE standard 1453-2015

Block 1 removes the dependence on the input voltage level by converting the value to a percent ratio. Blocks 2-4 simulate the lamp-eye-brain response. Block 2 simulates the behavior of an incandescent lamp by separating the low frequency voltage fluctuation from the main voltage signal using a squaring demodulator. Block 3 filters unwanted frequencies produced by the demodulator. In addition, it also weights the signal according to the lamp-eye-brain response. A 4^{th} order band-pass filter is used to represent the lamp-eye-brain response. Block 4 outputs the instantaneous flicker level by squaring the input voltage signal to simulate the non-linear eye-brain response, and a sliding mean filter is used to average the signal and simulate the short-term

storage effect of the brain. Block 5 processes the output of block 4 statistically to create a histogram. The output is separated into different classes or “buckets”, creating a probability density function (PDF). Then, from the PDF, a cumulative distribution function (CDF) can be formed. The CDF is the probability that the instantaneous flicker does not exceed a certain level (*IEEE Recommended Practice for the Analysis of Fluctuating Installations on Power Systems.* 1-74; U.S. Energy Information Administration).

Flicker severity is evaluated at two levels, short-term and long-term. Short-term flicker severity, P_{st}, is calculated over a 10-minute window of observation using equation (1):

$$P_{st} = \sqrt{0.0314P_{0.1} + 0.0525P_{1s} + 0.0657P_{3s} + 0.28P_{10s} + 0.08P_{50s}} \quad (1)$$

where $P_{0.1}$, P_{1s}, P_{3s}, P_{10s}, and P_{50s} are flicker levels that exceeded the percent of time specified in the subscript. For instance, P_{50s} represents flicker level that is exceeded 50% of the time. The flicker level values are taken from the CDF curve mentioned above. The suffix *s* represents smoothed values obtained from equations (2)-(5). For example, P_1 represents the flicker level that is exceeded 1% of the time.

$$P_{1s} = \frac{P_{0.7}+P_1+P_{1.5}}{3} \quad (2)$$

$$P_{3s} = \frac{P_{2.2}+P_3+P_4}{3} \quad (3)$$

$$P_{10s} = \frac{P_6+P_8+P_{10}+P_{13}+P_{17}}{5} \quad (4)$$

$$P_{50s} = \frac{P_{30}+P_{50}+P_{80}}{3} \quad (5)$$

Long term flicker severity is calculated from 12 successive short-term flicker severity values using equation (6) (*IEEE Recommended Practice for the Analysis of Fluctuating Installations on Power Systems.* 1-74).

$$P_{lt} = \sqrt[3]{\frac{1}{12}\sum_{j=1}^{12} P_{st_j}^3} \quad (6)$$

Methods

As voltage flicker was not a term known throughout the group, we were advised to read a few scholarly articles, case studies and the IEEE 1453 standard about it. One paper that was advised to read was centered around a Cloud Motion Simulator (CMS) to compute voltage flicker index. This CMS provided a way to simulate what would happen to the voltage if a cloud rolled over a PV system, and could be used to extrapolate the impacts on the grid. This paper closely aligned with the scope of our project and served as a point of reference to present to Eversoure as a better way to simulate a cloud covering PV panels. As a group, with our advisors, we presented this idea to our sponsor, Eversource, and received high praise and the go-ahead. To begin, we understood that analyzing the changes in transformers, tap changers, would be quick to analyze and focused on that. Then as the cloud simulator is our main goal, it was imperative that we then began that process.

A sub task assigned was to check if the addition of PV panels on a circuit lowered the stability of the circuit. A subset of the data provided held information on tap changes, which mark how often corrective action had to be taken to stabilize the circuit via a change in transformer windings. The data given showed repair dates on the transformers, and how many tap changes had occurred since the last repair. By dividing the number of tap changes in a repair period by the number of days in the repair period, the result is that the average amount of tap changes per day. By plotting the average tap changes per day as a function of time, as well as collecting meta-data with dates in regards to solar panel installation, the effects of solar panels on grid stability will become apparent.

Matlab proved to be a valuable tool in analyzing the data that Eversource provided to us. We created numerous plots to see impacts of cloud coverage over time, and use these plots to analyze the fluctuation on a given circuit. Figure XX is an example of four Cloud Motion Simulator parameters from a seven minute time window that shows constant photovoltaic output and a medium cloud covering the system.

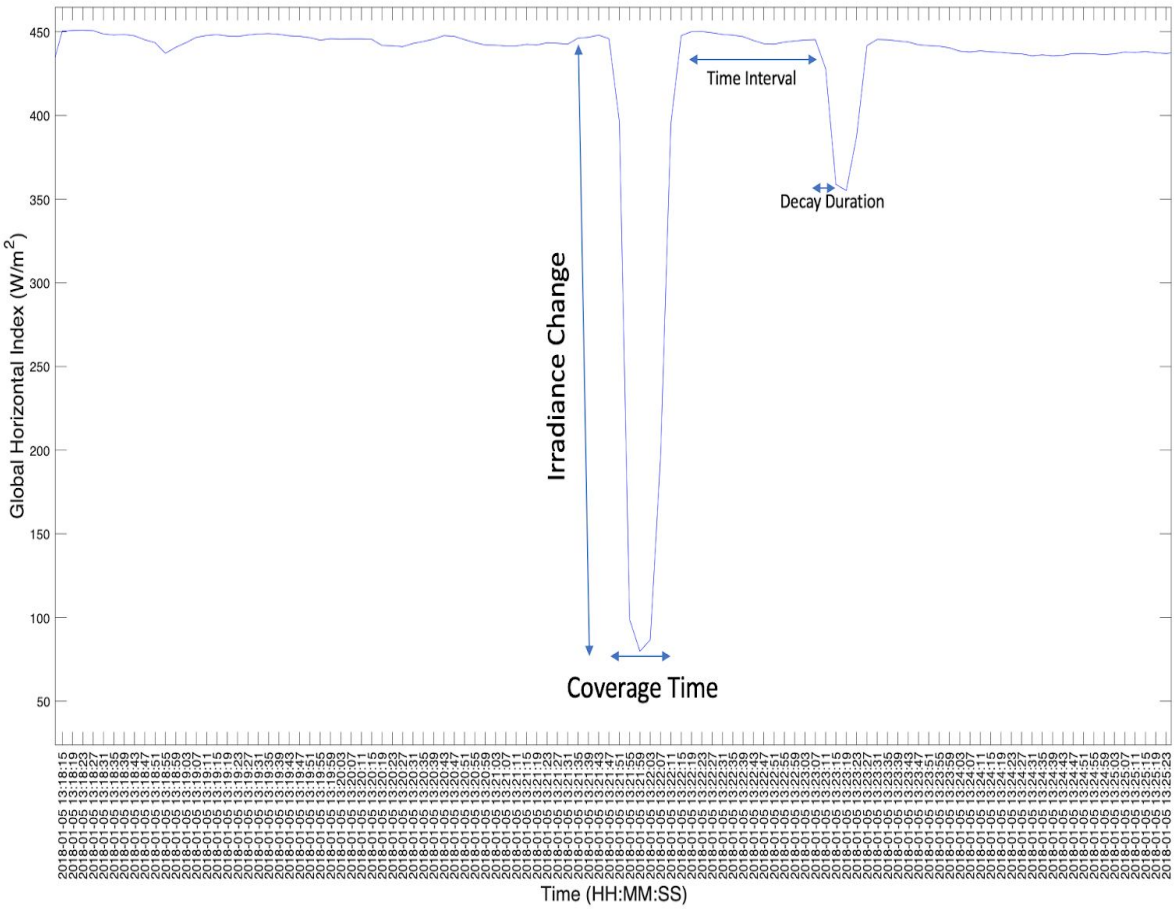

Figure 2: Demonstrating the use of irradiance values, the change of value, and the rate of change of values to find the CMS parameters.

The large drop in irradiance identifies a cloud large enough to significantly affect power output and cause a flicker event on the grid. The received irradiance drops from 450 W/m^2 to a measly 75 W/m^2, and on the second small tip, from the same 450 W/m^2 to a smaller 350 W/m^2.The initial drop lasted around 20 seconds, then 16 seconds for the second smaller drop. These two provide two numbers for the beginning of the CMS When the system returns to normal to then be covered by a cloud almost a minute later for a shorter period of time. The large irradiance change is the primary concern for Eversource. Figure 4 is a seven minute window on a sunny day experiencing

intermittent clouds; several events like the first described create the challenge of integrating PV into the grid.

Further analysis was done on different days. The graph seen below in Figure 3 is an example of what a typical PV output would look like when there are no clouds to block the sun.

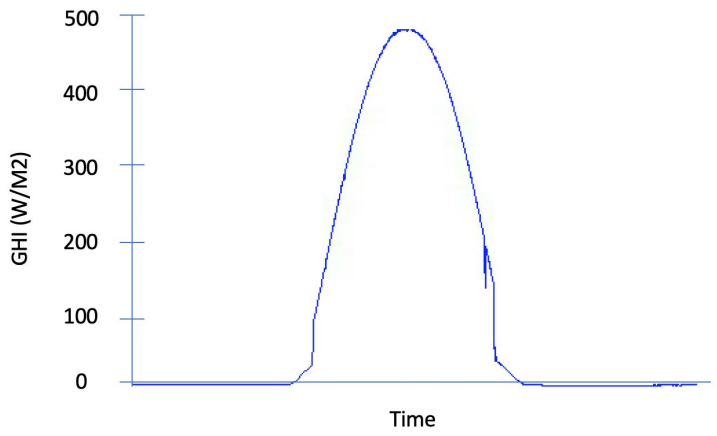

Figure 3: A typical Day, Without Clouds.

A normal sunny provides a bell curve like shape for the output of a system. This curve is predictable, allowing easy integration to the grid. The Power companies can easily predict what the expected load on the grid for a day will be. If PV power generation is also predictable, it is a fairly simple process to reduce traditional generation to account for the influx of power from PV. A seamless blend of PV and traditional generation will easily meet the load requirement of the grid.

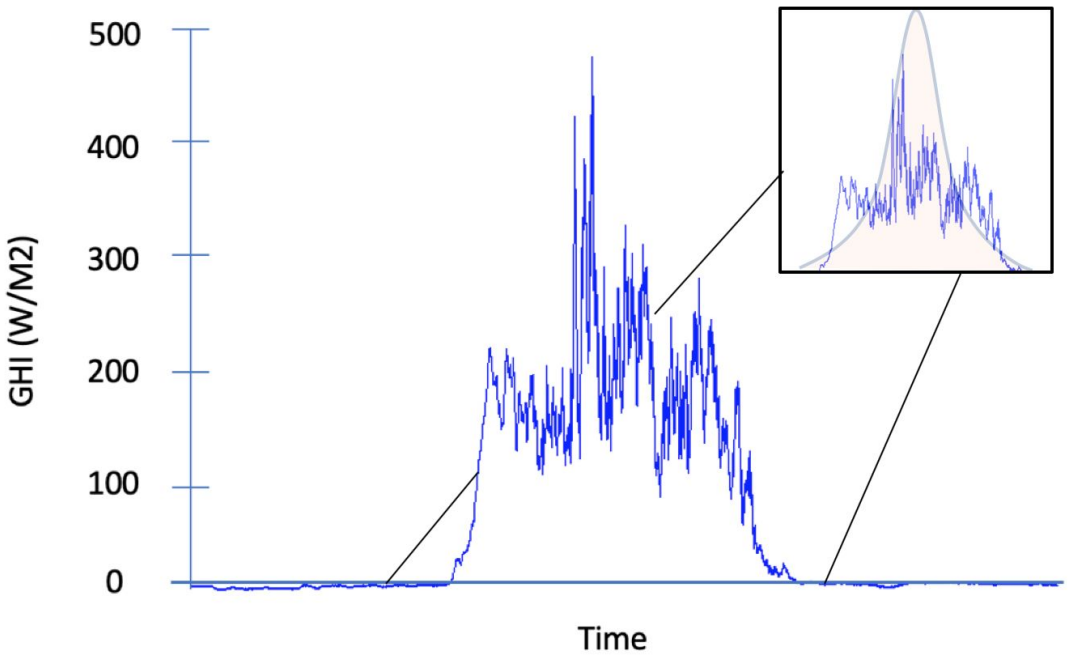

Figure 4: Cloudy Day Over PV Array

Figure 4 displays a figure of a very cloudy day with very little PV output. This was helpful in conducting analysis with the cloud coverage and calculating the worst case scenario in most situations. It is days like shown in figure XX that prevents the integration of solar into the grid. If the grid is compensated according to an expected bell curve of PV output, the eratic dips in irradiance due to clouds can create flicker events. If the grid is not compensated for PV generation, excess power is created and flows backwards along the power lines, reducing power quality. The data used was from Brookhaven Laboratories in New York and provided the results.

Cloud Motion Simulator (CMS)

There are several different ways to simulate the effects of cloud coverage on PV generators and the impact that it has on power systems. Some engineers simulate flicker by running a simulation at either end of capacity, from 100% to 5% (Eversource Energy). These run times are two extremes, not what would actually happen if a cloud rolls over for an extended period of time. This one runtime simulation is not effective, as more clouds could roll over and reduce the output of the PV system.

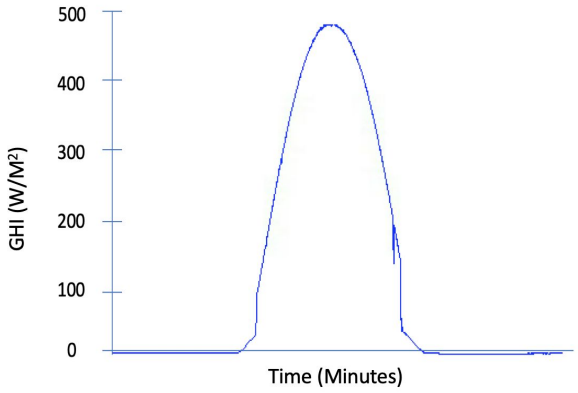

Figure 5: An ideal day with no clouds and consistent PV array output

Fig. 5 exhibits the expected solar irradiance of an ideal day. With no clouds blocking the sunlight, the curve generated by the measurement of power from the PV array is predictable and stable, allowing easy assimilation to the grid to help with load requirements (Nwaigwe, Mutabilwa, and Dintwa 629-633).

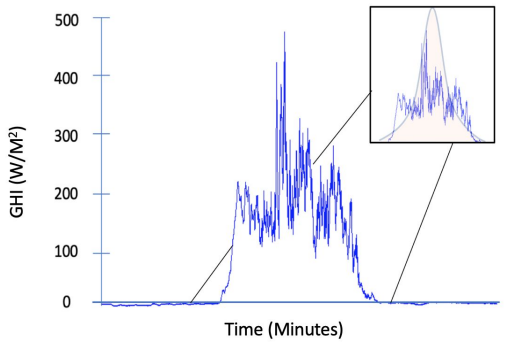

Figure 6: A cloudy day with inconsistent PV array output

Fig. 6 displays the irradiance received by a PV array on a day with clouds. With clouds intermittently covering the array, the power generated does not drop to 5% often from 100% but rather somewhere between. A quasi steady state (QSS) simulation can be used to perform simulations with higher accuracy than the aforementioned method to simulate flicker. One such method is to simulate how the effect of a cloud rolling over PV panels affect the output power over time, instead of the extremes (Rahimi et al. 2628-2636). By simulating a constant motion, opposed to two extremes, one can witness the effect of power quality over time.

In this study, to measure the impact of moving clouds upon large PV systems, we performed a simulation using irradiance data from a pyranometer located at Eversource solar site in East Springfield as shown in Figure 9. The line voltage of the circuit is 13.8 kV. The net DG current installed is 5,233.715 kW with 14,291.2 kW in progress. Parameters of cloud motion were determined by analyzing data from the pyranometer to gather measurable parameters regarding a cloud's impact on the system. Of the data to

collect, cloud speed, width of cloud, and time interval between clouds is determined. The direction of the moving cloud can also be determined with knowledge of the solar array's layout.

Cloud Motion Simulator Parameters

Figure 7 below shows how four of the parameters for the cloud motion simulator relate to irradiance values. Figure 8 is a table containing all the parameters of the simulation. Fig. 9 is a visual representation of how the parameters affect the simulation.

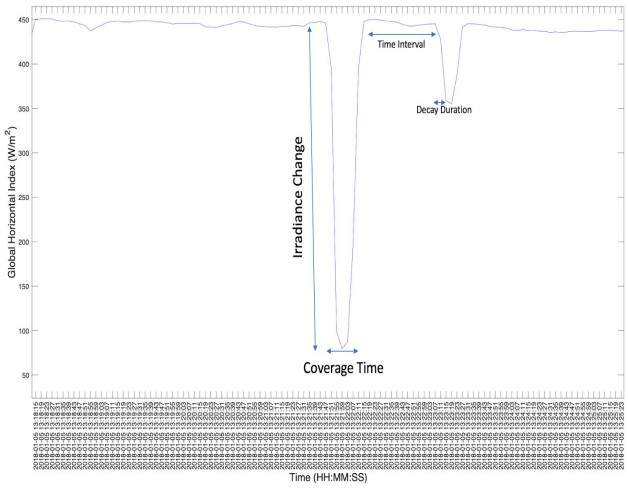

Figure 7: Demonstrating the use of irradiance values, the change of value, and the rate of change of values to find the CMS parameters.

The wind speed indicates how fast the cloud moves over the PV array, and determines how long the simulation performs the analysis of one cloud over the array. The time between successive clouds is a representation of how cloudy the day is, with a shorter interval indicating more tightly packed clouds. The cloud width parameter

determines how much of the array is covered as the cloud passes over. The final parameter is how fast a PV system can react to cloud coverage. A cloud may completely pass over an array, but still affect system output afterwards based on the systems rate of change.

Parameter	Parameter Description
P1	Number of clouds passing over
P2	Direction of cloud movement
P3	Speed of cloud over the system
P4	Time between successive clouds
P5	Width of clouds
P6	Rate of change PV generation as clouds move over the system

Figure 8: Cloud Motion Simulator Parameters

Number of Cloud: (P1) The first parameter is the number of clouds passing over the East Springfield array, which is how many instances the simulation will analyze.

Cloud speed: (P2) Throughout the calculations, the speed of the clouds was kept at a constant 7.6605 miles per hour to keep the coverage time longer than the decay time.

Width of Clouds: (P3) With increasing width of clouds the Pst also increases as long as the coverage time is less than decay time.

Number of clouds: (P4) With more clouds drifting over a PV system the number of fluctuations increase, increasing the Pst value.

Time interval between two successive clouds: (P5) By giving more time between cloud coverage, similar to the time it takes for the output to reach the minimum, the Pst value also increases.

Direction of Cloud Motion: (P6) The direction of cloud movement allows the simulation to adjust PV output based on the configuration and orientation of the PV array. This was assumed to be laterally over the array, covering more and more each second until it leaves

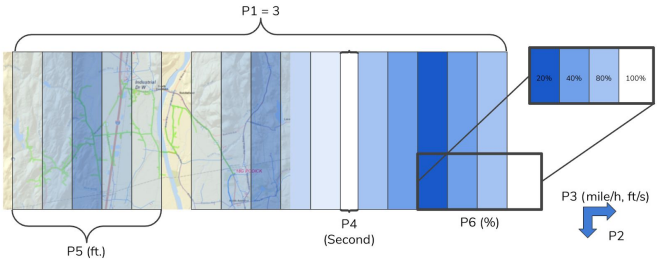

Figure 9: Visual representation of the cloud motion simulator parameters

Calculations

The P_{st} calculation is a calculation done in order to determine the flicker perceptibility of a distribution system. This has two elements: long- and short-term flicker. The short-term flicker is more applicable in this case due to the impacts that clouds can have in an instant over a solar array.

Using data given to the team by Eversource, P_{st} calculations were done using MATLAB. These values can be calculated using these equations:

$$d = \Delta S / S_{sc} \tag{7}$$

$$T_f = 2.3 * (100 * d * F)^3 \tag{8}$$

$$P_{st} = \sqrt[3]{\frac{\Sigma T_f}{window\ interval}} \tag{9}$$

Where d is a parameter that describes percentage voltage flicker and ΔS is the change in power output and S_{sc} is the maximum available fault current. F is a parameter that describes the fastest rate of change in the system and is set to 0.2 in this case. Using these equations, the data analyzed over the course of three days yields graphs of Pst over time.

GE Curve

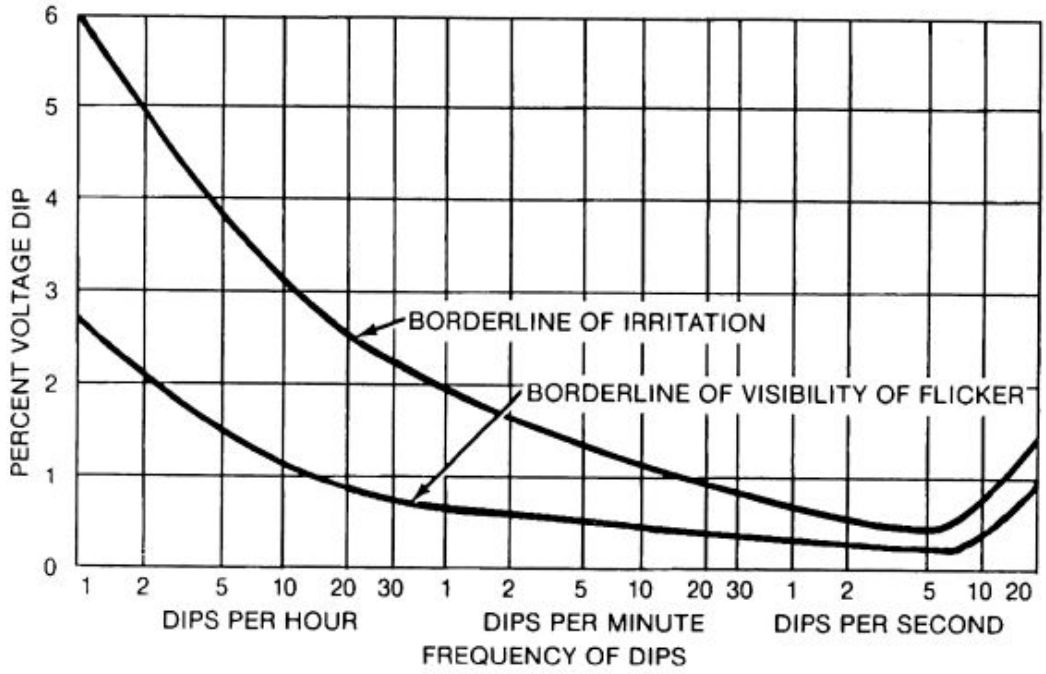

Figure 10: GE Curve

Desired by Eversource was a comparison between the IEEE 1453-2015 curve to the GE curve. Since the GE curve was the initial standard and is still in use today it is important that the flicker attributed on the grid meets all standards. After conducting a few MATLAB analysis, on average, the number of dips per minute from Eversource's data results in 5 dips. This number falls in between both the line of irritation and line of visibility. This is ideal since it does not further impact the end user, however, the larger dips, dips of 30% and more is an issue.

Results

Tap Changer

After obtaining information in regards to the tap changes over time, a plot of average tap changes per day was generated to observe any trends. Figure 11 below shows the average tap changes per day of the 18G-8x circuit.

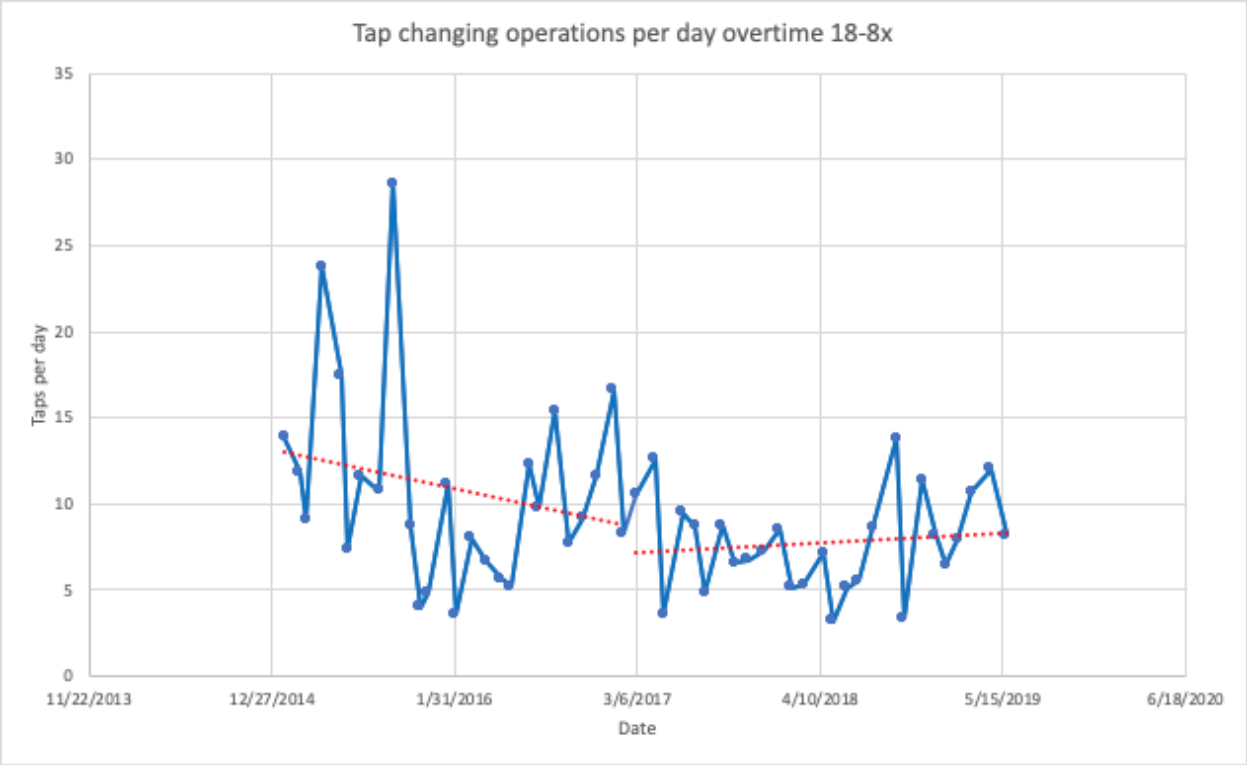

Figure 11: Average Tap Changing Operations for 18G-8x Circuit

After further analysis, it was decided that the graph be split up into two sections. The first, between 2014 and 2017 and the next between 2017 and 2019, the most recent data. The large spikes in mid 2015 years, is a result of UMass Amherst adding a

project to the Eversource grid and then a stabilization. However, the key thing to notice is that after March of 2017 there is a slight increase in tap changes over time. At first the line doesn't look steep however upon further investigation it is an increase of seven to almost ten changes a day which is significant. The conclusion is that with more and more PV systems added to the grid, there is an increase in voltage changes which means faster wear time and the possibility of failure and need of replacement.

Voltage Analysis

With Voltage and Irradiance data for the 18G8 circuit obtained, one could observe relationships between the two graphically. Irradiance and Voltage are both measured in respect to time, using one of two sensors, an ACLARA sensor or a pyranometer, so that the rates of change could also be compared. Another comparison made was the direct correlation of irradiance to measured voltage, which produced 9 horizontal lines when graphed. This most likely has to do with where the voltage measurement is taken, and what the ratio for the transformer was at the time of measurement.

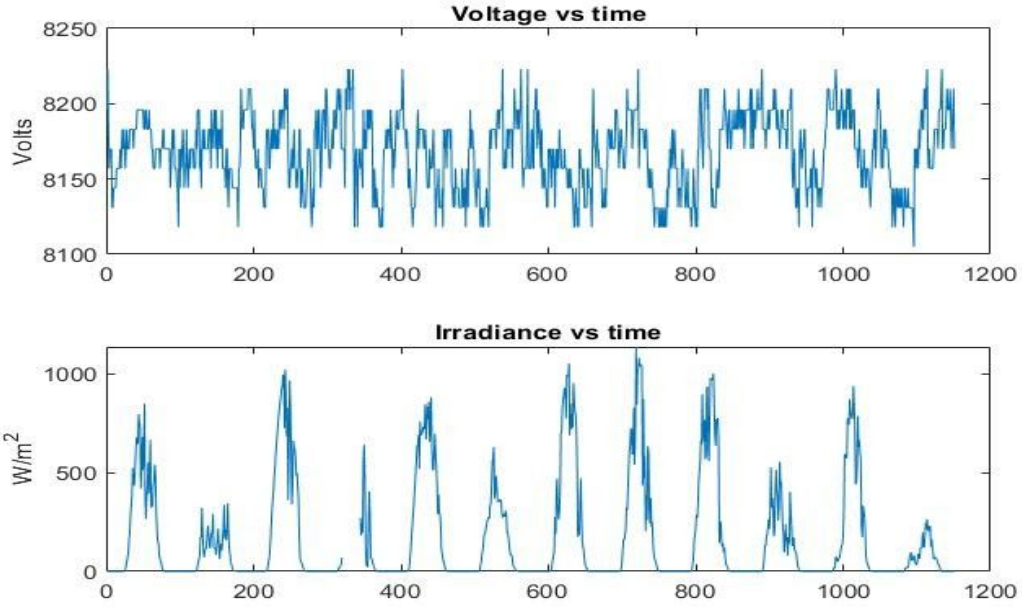

Figure 12: Aclara Data Over Three Days

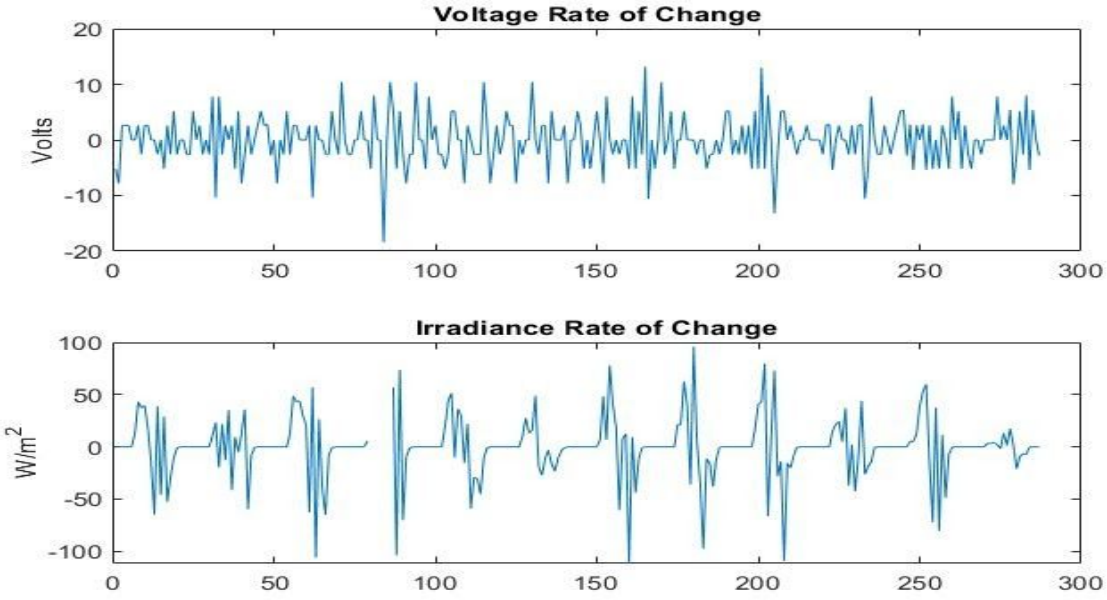

Figure 13: Aclara Data as Rate of Change Over Three Days

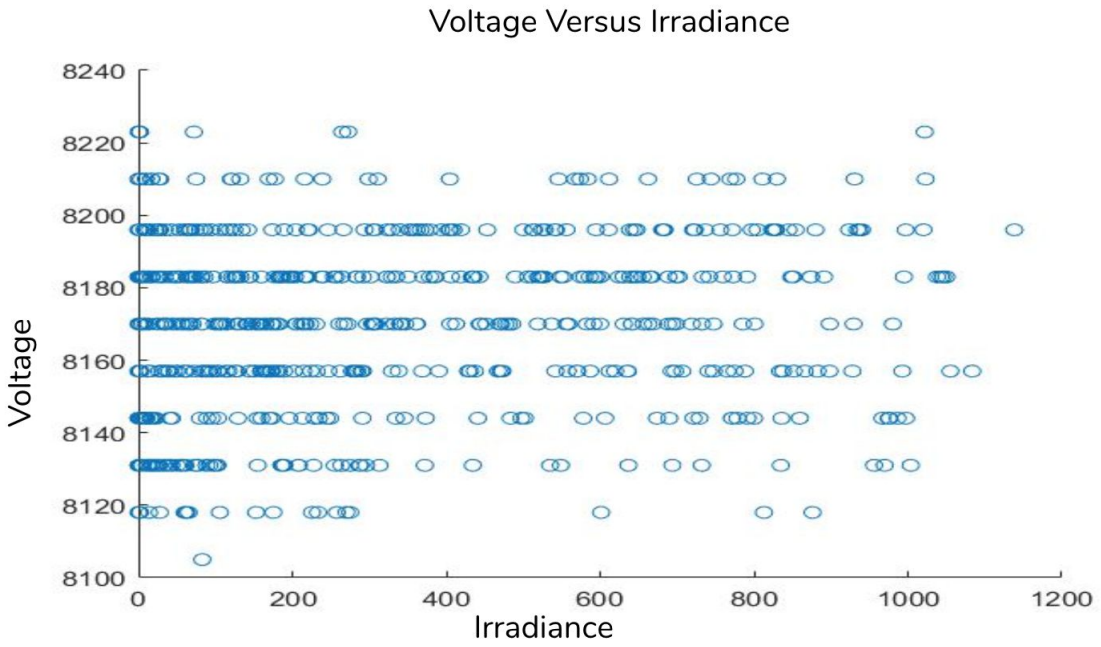

Figure 14: Aclara Data of Voltage Versus Irradiance Over Three Days

Short-Term Flicker Severity (P_{st})

The overarching goal for this project was to allow Eversource to analyze their grid in order to make better decisions about new DER entering the grid, and subsequently maintenance or upgrades that would be necessary to support these DERs. The P_{st} calculation is a calculation done in order to determine the flicker perceptibility of a distribution system. Due to the infrastructure that was present at the time the resolution of the irradiance data that was available was only a fifteen-minute interval. To accommodate for this the P_{st} calculations were calculated for an hour window instead of the normal ten-minute window.

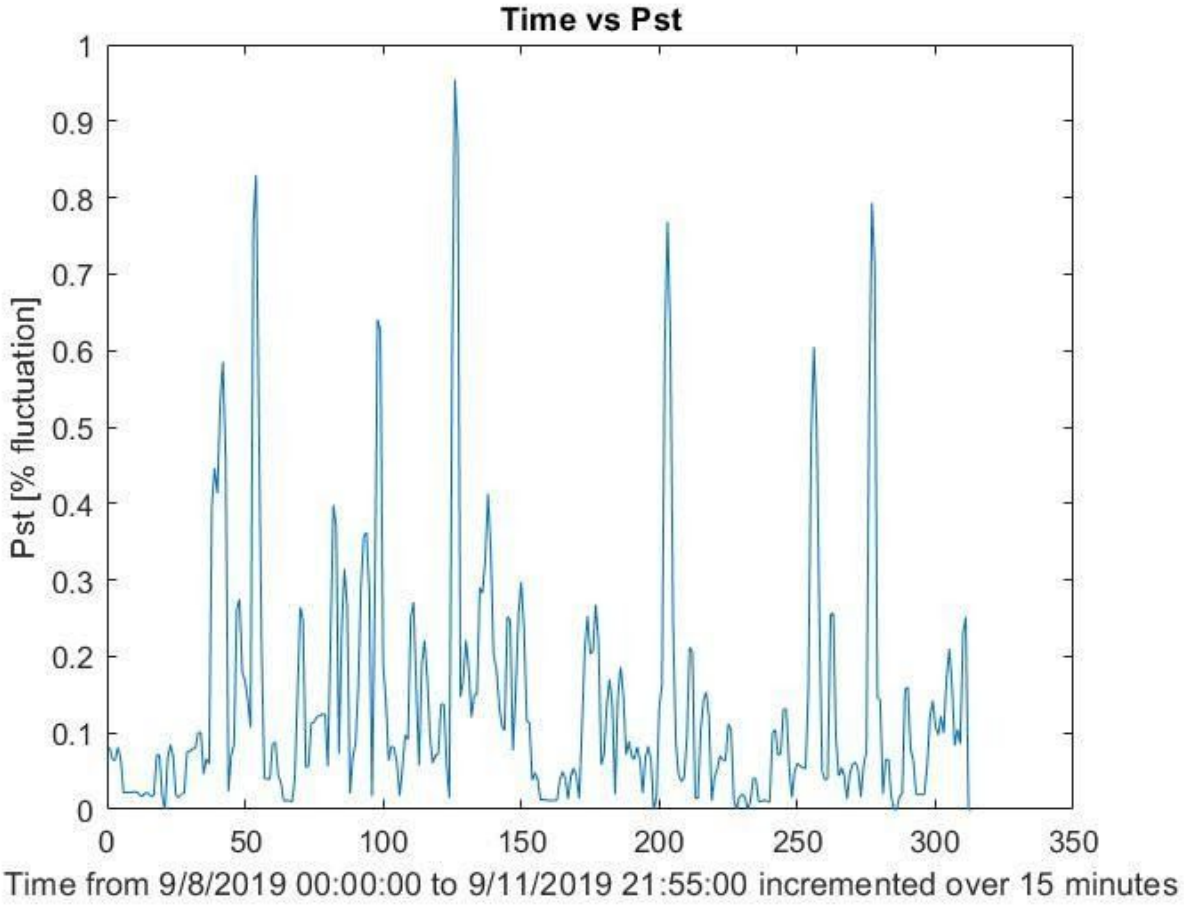

Figure 15: P_{st} Calculations Using Eversource Data

Figure 15 shows that the P_{st} values calculated all fall below the 2% fluctuation benchmark Eversource holds standard. With fluctuation within acceptable bounds, the relationship between line power and irradiance on line voltage was explored and is displayed in Figure 16.

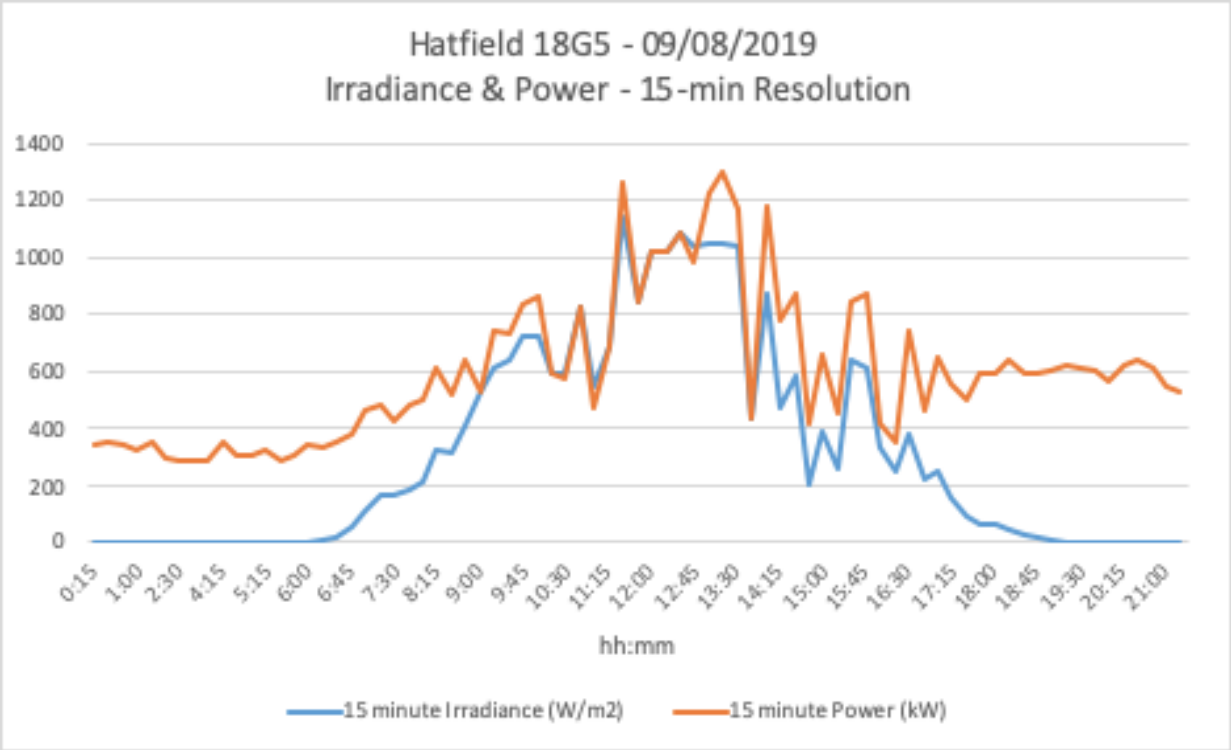

Figure 16: Irradiance Versus Line Power With Respect to Time

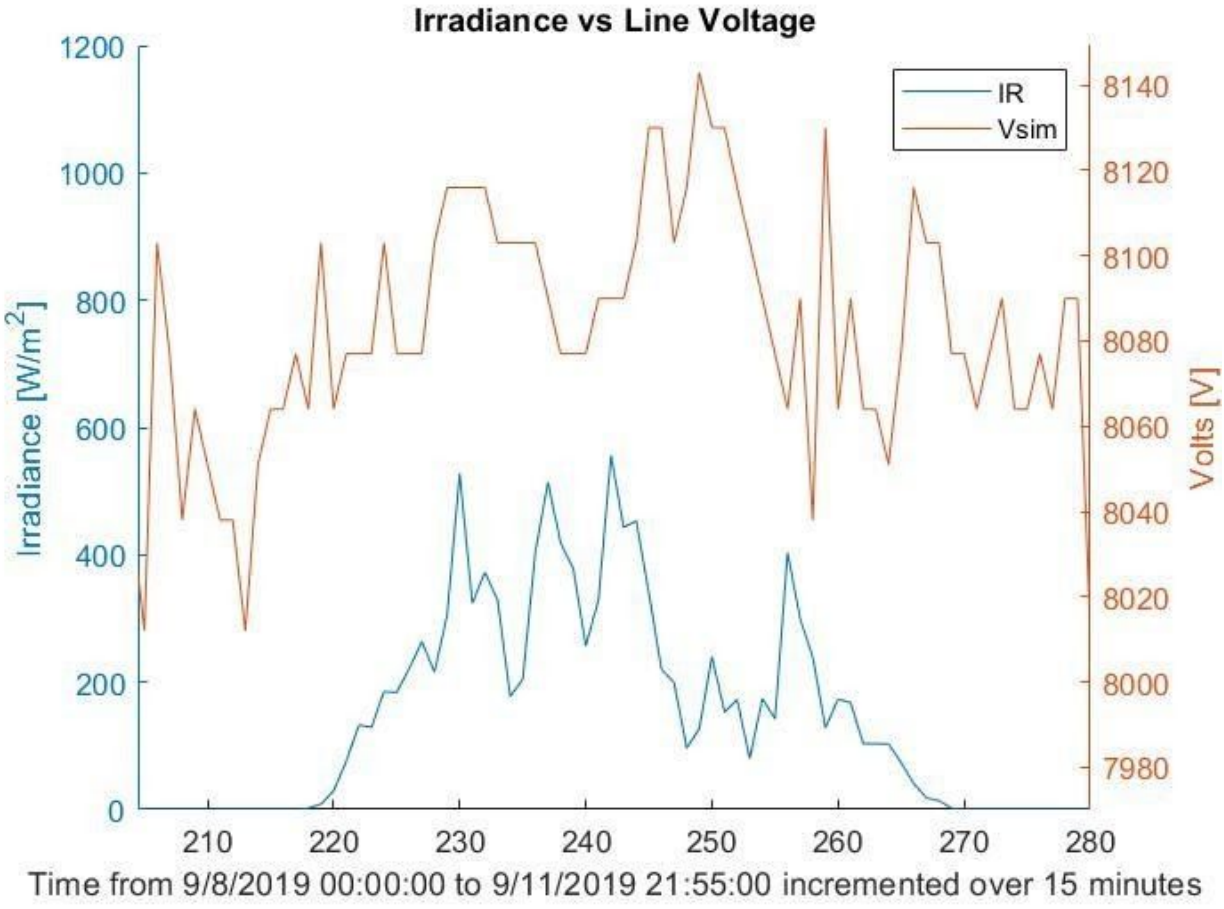

Figure 17: P_{st} Versus Line Voltage, Zoomed In

Due to how close irradiance and line power correspond, Figure 17 is a zoomed in view of Figure 16 to provide a better understanding of the relationship between the P_{st} and line voltage to see the full impact that a cloud coverage dip has on output and utility line transmission. The effect of irradiance on line voltage can be determined by the line power, as the current is kept constant so fluctuations in power result from a change in line voltage.

As seen in these figures, the impact on line voltage caused by irradiance is clear. There is in fact a lag in line voltage fluctuation due to irradiance. Unfortunately, with the

limited resolution of data available, this time cannot be accurately determined. The P_{st} was also compared against irradiance in order to see the relationship between the two.

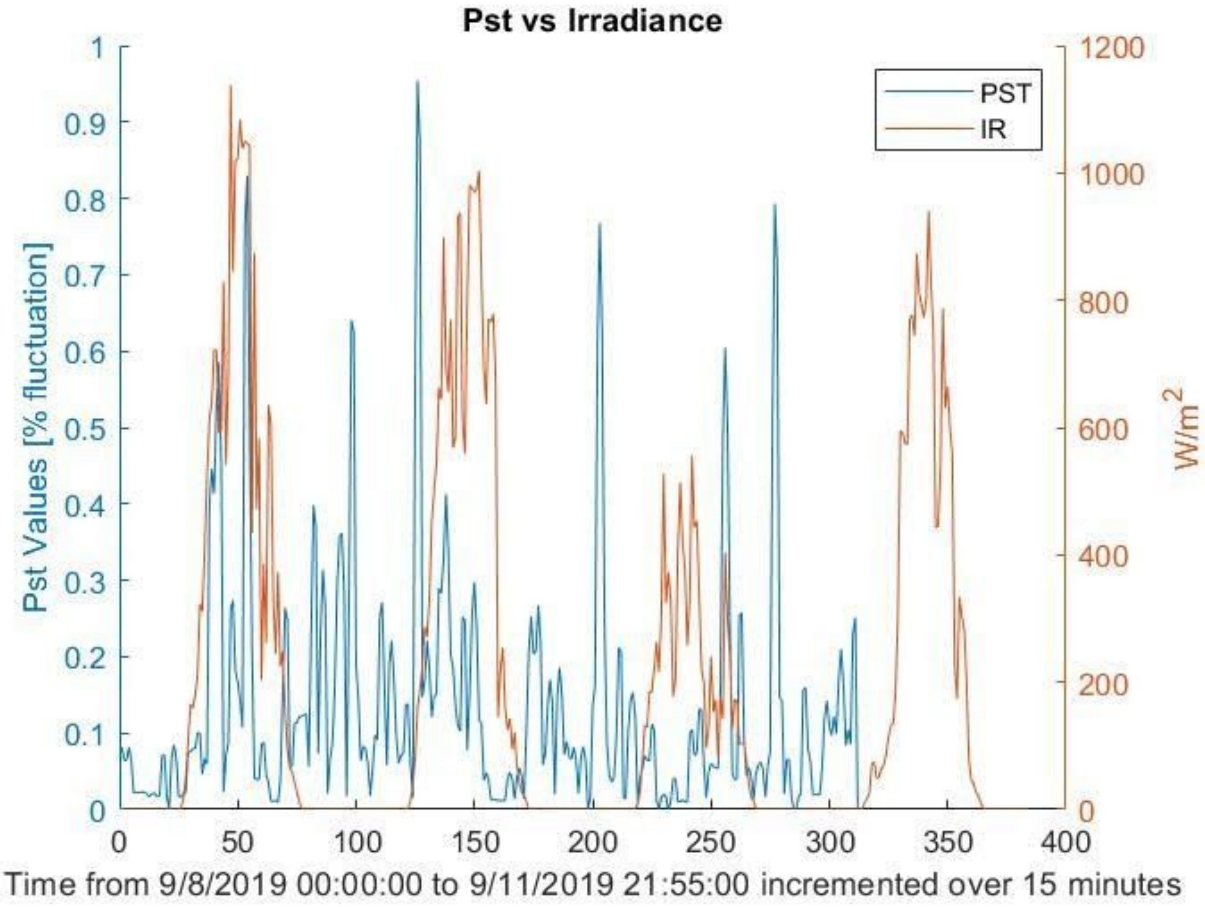

Figure 18: P_{st} Versus Irradiance

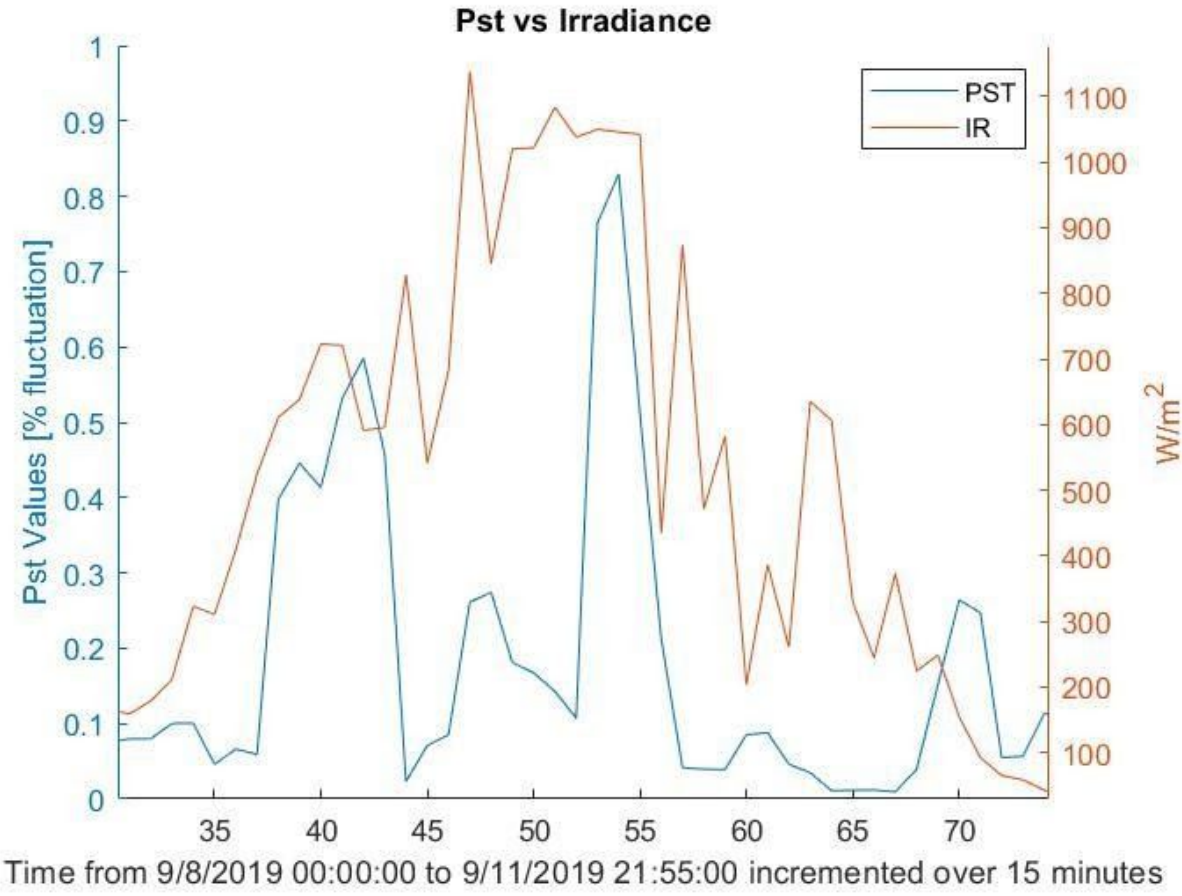

Figure 19: P_{st} Versus Irradiance, Zoomed In

As exemplified by the beginning of the zoomed in graph, the lobes of the irradiance and P_{st} fluctuation spike are matching after removing data. This shows the relationship between solar irradiance on a panel and both voltage fluctuation and line voltage. A big takeaway is the impact of cloud motion direction on Pst. The Pst graphs, specifically Figure 15, show that the direction of cloud motion can change the maximum value of Pst. The change is due to the fact that for different cloud motion directions, different PV generators are affected in the time sequence, and this results in different voltage changes, and consequently, different Pst values.

GUI

The purpose of the graphical user interface (GUI) is to provide a way to analyze irradiance data from a pyranometer for a given period of time in an efficient manner. Below is the current state of the GUI with an explanation of its functionality.

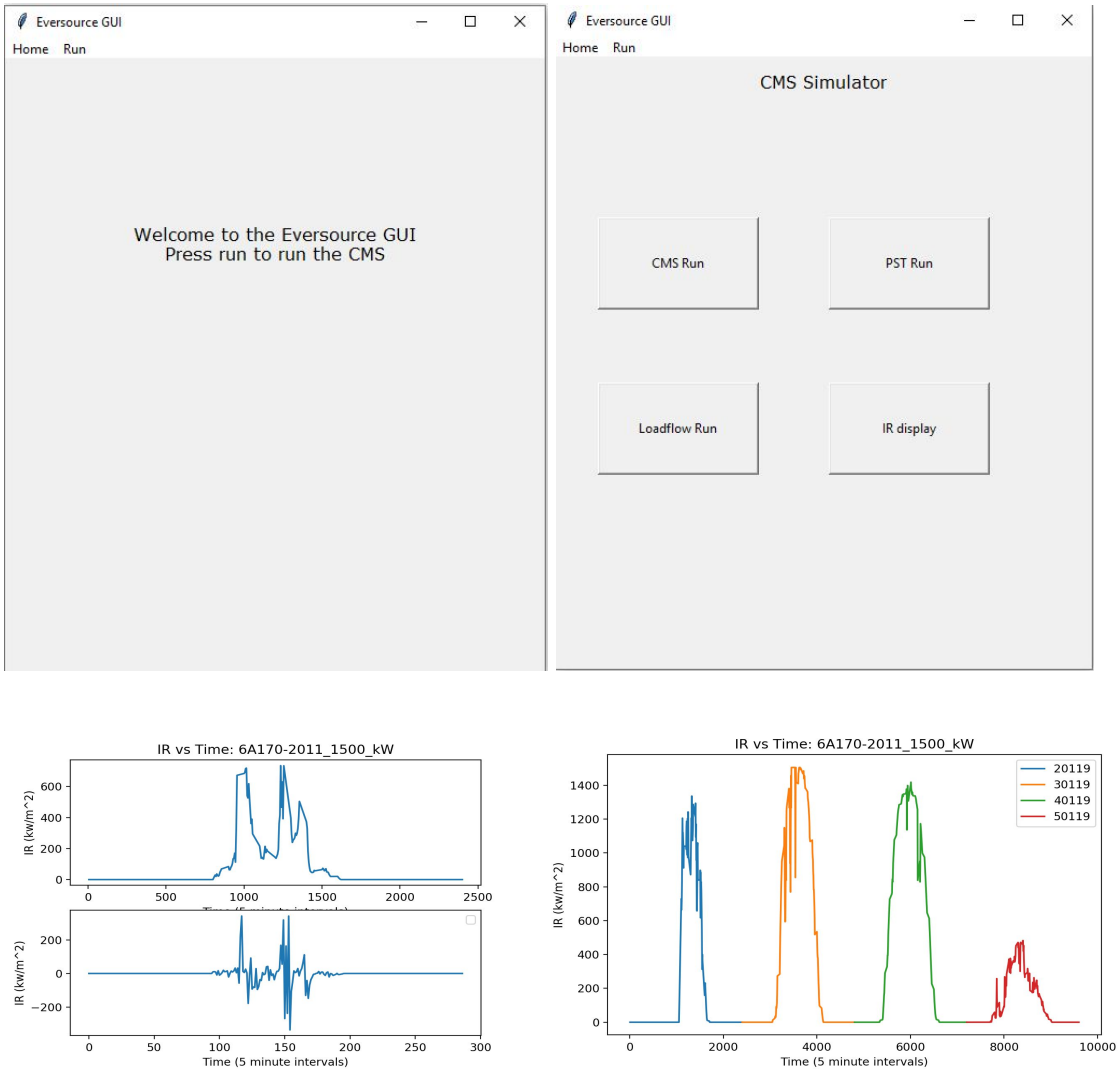

Figure 20: Current Functionality of GUI

Upon hitting run in Synergi, the Graphic User Interface displays. Hitting run in the toolbar shows the next screen with four button options. Additionally, a command window asks for the desired wind speed to calculate cloud coverage and length. Each button has a different functionality. The first runs a normal CMS calculation and displays graphs shown in Figure 7. The button below runs a load flow within Synergi to calculate the different loads at the moment. The third button displays PST data shown in Figure 19, for easy accessible references. Finally, the IR display button shows the two graphs seen in Figure 20, showing dips and changes over time in longer time spans. Further testing will be done to make sure this is smoothly added to Synergi and it accomplishes all the goals we set out to do. Due to the nature of the software that Eversource provides, quick numbers are provided without any analysis. Providing a Graphic User Interface (GUI) to see the effect of cloud coverage. With the additional input of cloud speed, the other Cloud Motion simulator parameters such as cloud width and rate of PV output so that a better understanding of what happens when a cloud rolls can be gathered. What we intend to do in terms of adjusting Transformer output based on irradiance data in “real time”. In our current situation hopefully every few seconds based on past IR data.

Conclusion

Considering PV accounts for over 9% of produced renewable energy in the United States, the ability to predict the effect of clouds passing over an array, the most likely cause of flicker, and its ability to influence the power grid, is of interest(U.S. Energy Information Administration). This paper introduces a cloud motion simulator that

simulates cloud shadows passing over a PV array. The CMS calculates the short-term flicker based on the IEEE 1453-2015 standard. In this work, effects of the cloud speed, width, number, direction, and interval between on short-term flicker are investigated. Results indicate that there is a relationship between flicker from a PV site and the voltage of the power lines it feeds.

Going Forward

The final end state would be the implementation and integration of code and begin testing to provide insight to Eversource. Initial analysis of the tap changers produced Information Eversource was Intrigued to receive and our initial voltage analysis, the same. The next step in this process is the full implementation of our analysis tool into synergi for eversource to utilize. Unfortunately, this aspect could not be realized at the time of writing due policy regarding the project team as an outside source to modify company software. Further expansion of this project would include real time analysis of irradiance data and automating the adjusting of generators to maximize efficiency based on the irradiance data.

References

Energy Sage. *Solar Panel Efficiency: What Panels are most Efficient? | EnergySage.* Energy Sage, 2020, https://news.energysage.com/what-are-the-most-efficient-solar-panels-on-the-market/.

Evans, Simon. "Solar, Wind and Nuclear have 'Amazingly Low' Carbon Footprints, Study Finds.", -12-08T17:58:33+00:00, 2017, https://www.carbonbrief.org/solar-wind-nuclear-amazingly-low-carbon-footprints.

Eversource Energy. *2018 Sustainability Report.*

---. "Information and Technical Requirements for the Interconnection of Distributed Energy Resources (DER).", Jan 21, 2020, https://www.eversource.com/content/docs/default-source/builders-contractors/der-information-technical-requirements.pdf?sfvrsn=ab2bfc62_8.

IEEE Draft Recommended Practice -- Adoption of IEC 61000-4-15:2010, Electromagnetic Compatibility (EMC)---Testing and Measurement Techniques---Flickermeter---Functional and Design Specifications. , 2011, doi:10.1109/IEEESTD.2011.5937014.

IEEE Recommended Practice for the Analysis of Fluctuating Installations on Power Systems. , 2015, doi:10.1109/IEEESTD.2015.7317469.

N.C. Clean Technology Center. "Dsire.", Feb 26, 2020, https://programs.dsireusa.org/system/program?fromSir=0&state=MA.

Nwaigwe, K. N., P. Mutabilwa, and E. Dintwa. "An Overview of Solar Power (PV Systems) Integration into Electricity Grids." *Materials Science for Energy Technologies*, vol. 2, no. 3, 2019, pp. 629-633. *CrossRef*, https://doaj.org/article/2a371a7ccbfc4a37bb9148b954ca7cd4, doi:10.1016/j.mset.2019.07.002.

Pitt, Jeson. "All You Need to Know about Tap Changers.", 2018-08-21T11:04:00.0000000-04:00, 2018-08-21T11:04:00.0000000-04:00, https://www.azom.com/article.aspx?ArticleID=16582.

Rahimi, Kaveh, et al. "Computation of Voltage Flicker with Cloud Motion Simulator." *IEEE Transactions on Industry Applications*, vol. 54, no. 3, 2018, pp. 2628-2636. *CrossRef*, https://ieeexplore.ieee.org/document/8240682, doi:10.1109/TIA.2017.2787621.

U.S. Energy Information Administration. "What is U.S. Electricity Generation by Energy Source?", Feb 27, 2020, https://www.eia.gov/tools/faqs/faq.php?id=427&t=3.

US Energy Information Administration. "How Much of U.S. Energy Consumption and Electricity Generation Comes from Renewable Energy Sources?", Jun 4, 2019, https://www.eia.gov/tools/faqs/faq.php?id=92&t=4.

Appendices

CMS Code [Matlab]

Used to produce CMS graphs and CMS parameters

```
%Solar Sort
clear Q N i L Data J;

Q = size(BNLSolarData);
N = Q(1);
i = 2;
L = 1;
Data = [];
while(i <= N)
      J = char(string(table2array(BNLSolarData(i,1))));
      if str2num(J(2)) == L
         if J(14) == '9' && J(16) == '3'
            d = table2array(BNLSolarData(i,2))
            Data = [Data, d]
         end
         if J(14) == '9' && J(16) == '4'
            L = L + 1;
            plot(Data)
         end
      end

   i = i + 1;
end


%BucketSort
%import data from previous script one day at a time.  Note call the imported data IR
%it will sort into buckets for ROC and regular values

dIRdt = diffby5(IR);
dIRdt = abs(dIRdt);
%differentiate in a discrete manner IR
B1 = 0; %%0-30
B2 = 0; %%30-40
B3 = 0; %%40-50
B4 = 0; %%50-60
B5 = 0; %%60-70
B6 = 0; %%70-80
B7 = 0; %%80-90
```

```
B8 = 0; %%90+
%IR data
A1 = 0; %%0-30
A2 = 0; %%30-40
A3 = 0; %%40-50
A4 = 0; %%50-60
A5 = 0; %%60-70
A6 = 0; %%70-80
A7 = 0; %%80-90
A8 = 0; %%90+
Size = size(dIRdt);
%%get size of IR
for i=(1:Size(1,2))
   %Bucket sort dIR/dt
   if dIRdt(i) < 30
      B1 = B1+1;
   elseif (dIRdt(i) > 30 && dIRdt(i) < 40)
      B2 = B2+1;
   elseif (dIRdt(i) > 40 && dIRdt(i) < 50)
      B3 = B3+1;
   elseif (dIRdt(i) > 50 && dIRdt(i) < 60)
      B4 = B4+1;
   elseif (dIRdt(i) > 60 && dIRdt(i) < 70)
      B5 = B5+1;
   elseif (dIRdt(i) > 70 && dIRdt(i) < 80)
      B6 = B6+1;
   elseif (dIRdt(i) > 80 && dIRdt(i) < 90)
      B7 = B7+1;
   else
      B8 = B8+1;
   end
end
S2 = size(IR)
for i=(1:S2(1,2))
   %Bucket sort IR
   if IR(i) < 30
      A1 = A1+1;
   elseif (IR(i) > 30 && IR(i) < 40)
      A2 = A2+1;
   elseif (IR(i) > 40 && IR(i) < 50)
      A3 = A3+1;
   elseif (IR(i) > 50 && IR(i) < 60)
      A4 = A4+1;
   elseif (IR(i) > 60 && IR(i) < 70)
      A5 = A5+1;
   elseif (IR(i) > 70 && IR(i) < 80)
      A6 = A6+1;
   elseif (IR(i) > 80 && IR(i) < 90)
```

```
        A7 = A7+1;
    else
        A8 = A8+1;
    end
end
    DataCheck = B1+B2+B3+B4+B5+B6+B7+B8
    %this loop sorts each discrete point into one bucket with ranges
    %designated above then chacks to ensure the buckets add to the total
    %data size
    B = [B1,B2,B3,B4,B5,B6,B7,B8]/Size(1,2);
    A = [A1,A2,A3,A4,A5,A6,A7,A8]/S2(1,2);
    %plot both functions as bar charts
    figure(1)
    bar(B)
    figure(2)
    bar(A)

function [output_data] = diffby5(input_data)
% find the Rate of change of input_data based on every fourth data
% entry(limits data set and creates a smoother curve generally
Q = size(input_data);
S = Q(1,2); %gets the actual size because size returns a table
j = 1;
i = 1;
while i < S
    if(i+5 < S)
        output_data(j) = ((input_data(i+4) - input_data(i))/5);
        j = j + 1; % iterate output data array counter
        i = i + 4; %jump and check new RoC for 5 seconds forward
    else
        i = i + 1;
    end
end
end
function [Sorted_data] = sortGreaterThanZero(Monthly_IR_Data)
%trim the IR data to be only greater than 0.4 and have a new index with the
%smallest possible value to allocate all data properly
j = 1;
for i = (1:size(Monthly_IR_Data))
   if Monthly_IR_Data(i) > 0.04
       Sorted_data(j) = Monthly_IR_Data(i);
       j = j + 1;
   end
end
end
```

Pst Code [Matlab]

```
clearvars -except combinedData
h = exist("combinedData",'var');
pause(3);
if(h)
   q = size(combinedData);
   q = q(1);
   if(q == 373)
      clearvars combinedData;
      clearvars q;
      h = 0;
   end
end
if(~h)

%% Import data from text file
% Script for importing data from the following text file:
%
%    filename: C:\Users\cmcurll\Desktop\18G3\combinedData.csv
%
% Auto-generated by MATLAB on 23-Mar-2020 13:49:55

%% Setup the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 5);

% Specify range and delimiter
opts.DataLines = [3, 1118];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["INSTANTANEOUSREALPOWER", "Sensor000616", "VarName3",
"START20190908", "Var5"];
opts.SelectedVariableNames = ["INSTANTANEOUSREALPOWER", "Sensor000616", "VarName3",
"START20190908"];
opts.VariableTypes = ["datetime", "double", "double", "double", "string"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, "Var5", "WhitespaceRule", "preserve");
opts = setvaropts(opts, "Var5", "EmptyFieldRule", "auto");
opts = setvaropts(opts, "INSTANTANEOUSREALPOWER", "InputFormat", "MM/dd/yyyy HH:mm");
```

```
% Import the data
combinedData = readtable("C:\Users\cmcurll\Desktop\18G3\combinedData.csv", opts);



%% Clear temporary variables
clear opts
end
p = exist("Solar",'var');
if(~p)
%% Import data from spreadsheet
% Script for importing data from the following spreadsheet:
%
%    Workbook: C:\Users\cmcurll\Desktop\Solar.xlsx
%    Worksheet: Sheet1
%
% Auto-generated by MATLAB on 30-Mar-2020 16:06:33

%% Setup the Import Options and import the data
opts = spreadsheetImportOptions("NumVariables", 3);

% Specify sheet and range
opts.Sheet = "Sheet1";
opts.DataRange = "A1:C384";

% Specify column names and types
opts.VariableNames = ["Sep2019000000", "VarName2", "VarName3"];
opts.VariableTypes = ["datetime", "double", "double"];

% Specify variable properties
opts = setvaropts(opts, "Sep2019000000", "InputFormat", "");

% Import the data
Solar = readtable("C:\Users\cmcurll\Desktop\Solar.xlsx", opts, "UseExcel", false);


%% Clear temporary variables
clear opts
end
% this data is in 5 minute intervals, I may add Gaussian white noise to
% simulate second data
i = 1;
j = 1;
q = size(combinedData);
q = q(1);

storeData = zeros((q/3)+1,4);
while(j < q)
```

```
    storeData(i,1) = i;%table2array(combinedData(1,j));
    storeData(i,2) = double(table2array(combinedData(j,2)));
    storeData(i,3) = double(table2array(combinedData(j,3)));
    storeData(i,4) = double(table2array(combinedData(j,4)));
    j=j+3;
    i=i+1;
end
    clearvars combinedData;
    combinedData = zeros((q/3)+1,4);
    combinedData = storeData;
    clearvars storeData;
Psimulated = (combinedData(:,2));
%Psimulated = resample(Pow,5,1);
IR_US = (Solar(:,2));
%IR_US = resample(IR,15,1);
Vsimulated = (combinedData(:,3));
%Vsimulated = resample(Vinst,5,1);

% L was calculated in synegri it represents the load down the circuit from
% the solar DG
j = sqrt(-1);
L = 0.059+j*0.165;
%volatage / current lead or lag
phi = tan(imag(L)/real(L));
%apparent power
i = size(Psimulated);
i = i(1);
j = 1;
Q = zeros(1,i);
while(j < i)
    %get reactive power
    Q(j) = tan(phi)*Psimulated(j,1);
    j = j + 1;
end
R = real(282.3256*L);
X = imag(282.3256*L);
j = 1;
k = 1;
maxV = 0;
minV = 50000;
maxCurrent = 0;
maxDeltaV = 0;
while(j < i)
    %find min and max IR of each hour window from current location in
    %the dataset
if(j + 4 < i)
  while(k < 4)
    if(Vsimulated(j+k,1) > maxV)
```

```
        maxV = Vsimulated(j+k,1);
    end
    if(minV > Vsimulated(j+k,1))
        minV = Vsimulated(j+k,1);
    end
    deltaV = maxV-minV;
    if(maxDeltaV < deltaV)
        maxDeltaV = deltaV;
    end
    current = Psimulated(j+k,1)/Vsimulated(j+k,1);
    if(maxCurrent < current)
        maxCurrent = current;
    end
    k = k + 1;
  end
end
    k = 1;

    F = maxDeltaV/maxV; % percentage fluctuation from max value within 30min seconds
    deltaS = maxV-minV; % max fluctuation
    Ssc = maxCurrent; %max available current from PCC
    d = deltaS/Ssc; %parameter for flicker
    Tf(j) = 2.3*(100*d*F)^3; % flicker time according to solar data
    if(j + 4 < i)
        deltaP = max(Psimulated(j:j+4)) - min(Psimulated(j:j+4));
        deltaQ = max(Q(j:j+4)) - min(Q(j:j+4));
        avgV = mean(Psimulated(j:j+4));
        d2 = ((R * deltaP) + (X * deltaQ))/(avgV^2);
        Tf2(j) = 2.3*(100*d2*F)^3; % flicker time according to solar data
    end
    maxV = 0;
    minV = 50000;
    maxCurrent = 0;
    maxDeltaV = 0;
    j = j + 1;
end
i = i -60;
j = 1;
k = 1;
q = size(Vsimulated);
q =q(1);
eightk = 8000 + zeros(q,1);
VsimulatedNorm = Vsimulated - eightk;
Pst = zeros(1,i);
while(j < i-1)
    if(j+10 < 5519 )
        Pst(j) = (sum(Tf(j:j+1))/60)^(1/3);
        Pst2(j) = (sum(Tf2(j:j+1))/60)^(1/3);
```

```
   end
   j = j + 1;
end
%based on data from 09/08/2019-09/11/2019
figure(1)
plot(Pst(1:313))
       title('Time vs Pst')
       xlabel('Time from 9/8/2019 00:00:00 to 9/11/2019 21:55:00 incremented over 15 minutes')
       ylabel('Pst [% fluctuation]')
%figure(2)
%plot(Pst2(20:5501))
 %       title('time vs Pst2')
  %      xlabel('Time in minutes')
   %     ylabel('Pst2')
%% notes
%d = (line resistance * active power change) + ((reactance * reactive power change)/nominal voltage^2)
%F = maxDeltaV/maxV; or percentage fluctuation from max value within 60 seconds
%Tf(i) = 2.3*(100*d*F)^3; % flicker time according to solar data
% Pst = cube root(sum(Tf)/10min)
% Plt = cube root(1/12*sum[1:12](Pst^3))
eightk = 8000 + zeros(i+60,1);
VsimulatedNorm = (Vsimulated - eightk);
IR_US = double(table2array(IR_US));
%Vsimulated = double(table2array(Vsimulated));
figure(3)
hold on
title("Irradiance vs Line Voltage")
yyaxis left
ylabel("Irradiance [W/m^2]")
ylim([0 1200])
plot(IR_US)
yyaxis right
ylabel("Volts [V]")
plot(Vsimulated(1:313));
legend("IR","Vsim");
xlabel("Time from 9/8/2019 00:00:00 to 9/11/2019 21:55:00 incremented over 15 minutes")
hold off

figure(6)
hold on
title("Pst vs Irradiance")
yyaxis left
ylabel("Pst Values [% fluctuation]")
plot(Pst)
yyaxis right
ylabel("W/m^2")
ylim([0 1200]);
plot(IR_US)
```

```
legend("PST","IR");
xlabel("Time from 9/8/2019 00:00:00 to 9/11/2019 21:55:00 incremented over 15 minutes")
hold off
```

Generator output Code

```
import mlreportgen.ppt.*
%% This file gets the IR change over a window so that we can change the transformer output percentages in Synergi
%Output table: durations,sizes,wind,
%% Import IR data needed if not already imported
%this takes approximately 2 minutes if executed
h = exist("secondData",'var');
if(~h)
    % Import data from spreadsheet
    % Script for importing data from the following spreadsheet:
    % Auto-generated by MATLAB on 24-Feb-2020 13:03:07
    % Setup the Import Options and import the data
    opts = spreadsheetImportOptions("NumVariables", 12);
    % Specify sheet and range
    opts.Sheet = "Sheet2";
    opts.DataRange = "A1:L1048576";
    % Specify column names and types
    opts.VariableNames = ["Column1", "Var2", "Var3", "Var4", "Var5", "Var6", "Var7", "Column8", "Var9", "Column10", "Var11", "Column12"];
    opts.SelectedVariableNames = ["Column1", "Column8", "Column10", "Column12"];
    opts.VariableTypes = ["string", "char", "char", "char", "char", "char", "char", "string", "char", "string", "char", "string"];
    % Specify variable properties
    opts = setvaropts(opts, ["Column1", "Var2", "Var3", "Var4", "Var5", "Var6", "Var7", "Column8", "Var9", "Column10", "Var11", "Column12"], "WhitespaceRule", "preserve");
    opts = setvaropts(opts, ["Column1", "Var2", "Var3", "Var4", "Var5", "Var6", "Var7", "Column8", "Var9", "Column10", "Var11", "Column12"], "EmptyFieldRule", "auto");
    % Import the data
    secondData = readtable("C:\Users\cmcurll\Downloads\second data.xlsx", opts, "UseExcel", false);
    % Clear temporary variables
    clear opts
end

% ask for wind speed or option for file input

x = input("Enter Wind Speed for day in <strong>single quotes </strong> or type file for a file import: \nfile import must match time scale and size of IR data\nPlease rename the file WIND pre import\n");
```

```
if(x ~= "file")
   windSpeed = str2double(x);
else
   uiopen
end

%% start CMS
%% start CMS
%Best to use if only one day of data is imported otherwise it will take
%forever and the data may not be that useful
   m = double(table2array(secondData(:,3))); %import all of column 3 for IR data, if you need a
different column change this
   highIR = 0;
   SSS = size(m); % gets the size of the passed parameter m
   cloud = zeros(1,SSS(1,2));
   IRdip = [];
   cloudsizes = [];
   starttimes = [];
   durations = [];
   starttime = 0;
   endtime = 0;
   windtotal = 0;
   windavg = 0;
   cloudsize = 0;
   j = 1;
   i = 2;
   lowIR = 900;
   eof = SSS(1,1);
while(i <= eof)
  if(m(i,1) > highIR - 19)
     if(m(i,1) > highIR)
     highIR = m(i,1);
     end
     %store highest IR number to this point
     cloud(i) = 0;
     if(cloud(i-1) == 1)
         endtime = i;
     end
  elseif(m(i) + 19 < highIR)
     cloud(i) = 1;
     if(m(i) < lowIR)
     lowIR = m(i);
     end
```

```
      %if there is a discrepancy of more than 20 W\m^2 then we assume
      %there is a cloud or at least the sun is setting
        if(cloud(i-1) == 0)
          starttime = i;
        end
    else
       cloud(i) = 0;
      if(cloud(i-1) == 1)
          endtime = i;
      end
    end
if(starttime ~= 0 && endtime ~= 0)
    starttimes = [starttimes,starttime];
    IRdip = [IRdip,highIR-lowIR];
    highIR = 0;
    lowIR = 900;
    if(x=="file")
    windavg = 0;
    windtotal = 0;
    while(j <= endtime-starttime)
       windtotal = windtotal+WIND(j+starttime);
    end
    windavg = windtotal/j;
    else
       windavg = windSpeed;
    end
    %convert windavg to cloud size
    %1mile/hr = 0.447m/s
    %endtime-starttime = coverage in minutes
    %m/s*(coverage in min)*60sec/min outputs cloud size in meters

    cloudsize = 0.447*windavg*(endtime-starttime)*60;
    duration = endtime - starttime;
    cloudsizes = [cloudsizes, cloudsize];
    durations = [durations, duration];
    %%store data from calculation in case its needed later
    %%we are assuming square clouds for worst case scenario analysis
    starttime = 0;
    endtime = 0;
    cloudsize = 0;
    duration = 0;
    windtotal = 0;
    %windavg = 0;
```

```
    j = 1;
end
i = i +1;
end
figure(1)
stem(starttimes,IRdip);
xlabel("times of cloud cover start in seconds since start")
ylabel("Size of irradiance change [W/m^2]")
i = 1;
s = size(durations);
s = s(1,2);
while(i<s)
if(durations(1,i) > 10000)
    durations(1,i) = 0;
end
i = i + 1;
end
figure(2)
stem(starttimes,durations);
xlabel("times of cloud cover in seconds since start")
ylabel("length of duration in seconds")
%clearvars -except m IRdip starttimes eof maxIR
generatorOutputs = zeros(1,eof);
j = 1;
i = 1;
maxIR = input("Enter IR value in Watts per Meter squared that leads to 100% generator
output<strong> in single quotes</strong>\n");
maxIR = str2double(maxIR);
while(i <= eof)
  generatorOutputs(i) = 100*(m(i)/maxIR);
  i = i + 1;
end
figure(3);
plot(generatorOutputs);
xlabel("time in seconds since start")
ylabel("percentage setting for generator output")

% create csv's for import to python and data analysis
outputs = [cloudsizes',durations',IRdip',starttimes'];
out = table(outputs(:,1),outputs(:,2),outputs(:,3),outputs(:,4));
out.Properties.VariableNames = ["cloud sizes[m]","duration of cover[s]","delta IR[W/m^2]","start
times"];
```

```
writematrix(generatorOutputs','genOutput.csv");
writetable(out,"cloudData.csv");
```

GUI Code [Python]

```
import tkinter as tk
import requests
from decimal import getcontext, Decimal
#import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import pandas as pd

month_dict = {"jan" : "January",
              "feb": "February",
              "mar": "March",
              "apr": "April",
              "may": "May",
              "jun": "June",
              "jul": "July",
              "aug": "August",
              "sep": "September",
              "oct": "October",
              "nov": "November",
              "dec": "December"}

#lat and long are based on an x-y plane coordinate system so if W, long needs to be negative
## need to start coding in the eversource stuff
HEIGHT = 700
WIDTH = 800

getcontext().prec = 5


labels = []




#try and use Text to print on each line
#the data is in kWh/m^2/day
def ir_format_response(weather):
    IR_data = pd.read_csv('/Users/MarkusZimmermann/Desktop/6A267-2012_2000_KW.csv', index_col =
"DATE")
    weather = str(weather)
    print(weather)
    r = 0
    counter = 0
    filtered = IR_data.loc[[weather], ["TIME", "kW"]]
    x = filtered.TIME
```

```
    y = filtered.kW

    for num in y:
        if num != 0:
            non_zero_IR = num
            labels.append(tk.Label(lower_frame, text=str(non_zero_IR), anchor="w", justify="left"))
            labels[r].place(relwidth=0.5, y=(r*20))
            r = r + 1
##          print(non_zero_IR)

    plt.plot(x, y)
    plt.ylabel('IR')
    plt.xlabel('Time (5 minute intervals)')
    plt.title('IR vs Time')
    plt.show()

##  print(filtered)




##    mylist = Listbox(lower_frame, yscrollcommand = scrollbar.set)
##    for key in month_dict.keys():
##        temp = Decimal(weather["outputs"]["avg_dni"]["monthly"][key])*(730)
##        mylist.insert(END, month_dict[key] + ": "+ str(temp))



##
##    r = 0
##    del labels[:] #remove any previous labels if the callback was called before
##    for key in month_dict.keys():
##        temp = Decimal(weather["outputs"]["avg_dni"]["monthly"][key])*(730)
##        labels.append(tk.Label(lower_frame, text=month_dict[key] + ": "+ str(temp), anchor="w",
justify="left"))
##        labels[r].place(relwidth=0.5, rely=0.2, y=(r*20))
##        r = r + 1

def city_format_response(weather):
    try:
        name = weather["name"]
        desc = weather["weather"][0]["description"]
        final_str = "City: %s\nDescription: %s\n\nAverage Irradiance in W/m^2:" % (name, desc)
    except:
        final_str = "There was a problem with the input"

    return final_str

##def get_ir_data(coordinate):
```

```
##    ir_weather_key = "3sQIGVGw9civsBSL7uI50SispEWBcx1g4gLomFvx"
##    #url =
"https://developer.nrel.gov/api/alt-fuel-stations/v1.json?fuel_type=E85,ELEC&state=CA&limit=2&api_key=
3sQIGVGw9civsBSL7uI50SispEWBcx1g4gLomFvx&format=JSON"
##    ir_url =
"https://developer.nrel.gov/api/solar/solar_resource/v1.json?api_key=3sQIGVGw9civsBSL7uI50SispEWB
cx1g4gLomFvx"
##    coor_string = coordinate.split(", ")
##    print(coor_string)
##    ir_params = {"APPID": ir_weather_key, "lat": float(coor_string[0]), "lon": float(coor_string[1]), "units":
"imperial"}
##    ir_response = requests.get(ir_url, params=ir_params)
##    weather = ir_response.json()


   ##ir_format_response(weather)
   ##print(weather)

def get_city(coordinate):
   city_weather_key = "8897cde892da72541a4c2b7baed2289b"
   city_url = "https://api.openweathermap.org/data/2.5/weather?"
   coor_string = coordinate.split(", ")
   print(coor_string)
   city_params = {"APPID": city_weather_key, "lat": float(coor_string[0]), "lon": float(coor_string[1]),
"units": "imperial"}
   city_response = requests.get(city_url, params=city_params)
   weather = city_response.json()

   label_1["text"] = city_format_response(weather)
   print(weather)

#key for weather:
#8897cde892da72541a4c2b7baed2289b
# api.openweathermap.org/data/2.5/forecast?q={city name},{country code}

#key for irradiance
#3sQIGVGw9civsBSL7uI50SispEWBcx1g4gLomFvx
#https://developer.nrel.gov/api/alt-fuel-stations/v1.json?fuel_type=E85,ELEC&state=CA&limit=2&api_key
=3sQIGVGw9civsBSL7uI50SispEWBcx1g4gLomFvx&format=JSON

root = tk.Tk()

canvas = tk.Canvas(root, height = HEIGHT, width = WIDTH)
canvas.pack()

#top frame
frame = tk.Frame(root, bg="#80c1ff", bd = 5)
frame.place(relx = 0.5, rely = 0.1, relwidth = 0.7, relheight = 0.1, anchor = "n")
```

```
entry = tk.Entry(frame, font=40)
entry.place(relwidth = 0.68, relheight = 1)
                                                #get_city(entry.get())          #this is the city
button = tk.Button(frame, text = "Get IR data", font=40,
command=lambda:[ir_format_response(entry.get())])
button.place(relx = 0.7, relwidth = 0.3, relheight = 1)

lower_frame = tk.Frame(root, bg="#80c1ff", bd = 10)
lower_frame.place(relx = 0.5, rely=0.25, relwidth=0.75, relheight=0.6, anchor="n")

label = tk.Label(lower_frame, anchor="nw", justify="left")
label.place(relwidth=1, relheight=0.1)

##label_1 = tk.Label(lower_frame, anchor="nw", justify="left")
##label_1.place(relwidth=0.5, relheight=0.01)

label_2 = tk.Label(lower_frame, anchor="e", justify = "right", text="This is a placeholder\nfor the graph")
label_2.place(relx=0.5, relheight = 0.81, relwidth = 0.5)

##scrollbar = tk.Scrollbar(lower_frame)
##scrollbar.pack(side = RIGHT, fill = Y)
##mylist.pack(side = LEFT, fill = BOTH)
##scrollbar.config(command = mylist.yview)

root.mainloop()
```

Extracted CMS Parameter

AVG Wind Speed = 12mph (User selected)

cloud sizes[m]	duration of cover[s]	delta IR[W/m^2]	start times[s]	since "2018-01-01 00:00:00"
55678.32	173	137.4	128346	January 2, 2018, 11:39:06 AM
12229.92	38	43.6	128531	
5471.28	17	24	128593	
1609.2	5	28.4	128885	
3218.4	10	206.2	128913	
10620.72	33	391.3	128937	
16735.68	52	42.2	129147	
182805.12	568	57.8	129514	January 2, 2018, 11:58:34 AM
3540.24	11	232.2	130119	
38620.8	120	382	130134	
250713.36	779	422.5	130336	
85609.44	266	43.7	131161	
321.84	1	19.9	131505	
7724.16	24	24	131646	
37333.44	116	166.3	131838	
11586.24	36	257.4	132071	
5793.12	18	205.5	132124	
1609.2	5	20.9	132482	January 2, 2018, 12:48:02 PM
9655.2	30	28.9	132550	

14482.8	45	71.4	133476	
15126.48	47	216.6	133775	
643.68	2	21.2	134023	
321.84	1	19.1	134711	
17379.36	54	22.3	135707	
26712.72	83	24.7	136266	
2896.56	9	20	136966	
53103.6	165	32.9	137552	
8046	25	26.7	138155	
2574.72	8	19.3	138273	
965.52	3	19.2	138446	
3218.4	10	20.7	138557	
10620.72	33	23.2	138657	
129701.52	403	74.9	139089	
55678.32	173	48.6	139677	
12873.6	40	22.3	140069	
20863921.68	0	277.429	140576	January 2, 2018, 03:02:56 PM

Second Based Generator Outputs

Start time 2018-01-01 0:00:00
Max expected irradiance = 1200 W/m^2
Used to inject into Synergi electric and fulfil the goal of getting eversource a more accurate flicker model