PHASOR MEASUREMENT UNIT DATA VISUALIZATIONS AND THEIR ROLE IN IMPROVING OPERATION OF THE ELECTRIC GRID

BY

KATHLEEN GEGNER

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Thomas J. Overbye

ABSTRACT

In the energy industry, as in many other industries, data is growing faster than the tools needed to transform it into meaningful insights, decisions, and action. In the North American electric grid, many new sensors, called phasor measurement units, have been installed to better measure, assess, and provide situational awareness to operators of the electric grid. These devices can record measurements over 60 times more frequently than currents systems, meaning more data is available.

With the growth of electric grid data, much more work in the industry has focused on creating tools that can translate the wealth of phasor measurement unit data into improved operator system awareness and which can expedite and improve operator decision making. As such, this thesis documents the efforts of the University of Illinois to create visualizations of phasor measurement unit data for the purpose of improving operator situational awareness in electric grids. Specifically, the University of Illinois team created a large-scale electric grid model; created simulations, like earthquakes and ice storms, to strain the grid model; created a software tool that automates the process of going from simulation data to its analysis; and then created several data visualizations aimed at improving real time operation of the electric grid. Though the work was done for a specific region of the U.S. electric grid, the processes that were followed, the software that was built, and the visualizations that were made can be easily extended to any electric system, and, in fact, making sure they could be generalized was a key guiding principle of our work.

To the extraordinary village that has helped shaped me in to the person I am, this is for you.

ACKNOWLEDGMENTS

First, a big thank you to the Bonneville Power Administration for funding my research, through the TIP 359 program, and for providing such an engaging, challenging, and interesting project. These last two years have been a joy and have taught me so much.

To my adviser, Professor Overbye, who, coincidentally or not, came into my life through an email about what class I should take, thank you for taking me under your wing. Your kindness, encouragement, and constant nudging to push me outside my comfort zone have made me a better person and engineer. If I ever decide to do a PhD, you will be the first one I talk to. I am a proud $\frac{Team}{B}$ member.

To Dr. Esa Rantanen, and the entire Rochester Institute of Technology team, I have thoroughly enjoyed our collaboration for this project. Thank you for your patience with us power engineers, as we try to make our area of expertise more clear to you. Likewise, thank you for taking the time to share ideas from human factors with me. I always love learning new things and you provided an abundance of opportunities to do so, for which I am extremely grateful.

To my fellow graduate students, thank you for making it so much fun to come to work. It has been my pleasure being in the trenches with such smart, kind, and hardworking folks. My sincere wishes to you, for great successes!

To my teachers, from grade school through college, you helped get me to this point. Thank you is insufficient for all you do, but it is what I have to give. You truly are heroes.

To my family—Laurel, Mom, and Dad—thank you for being the wonderful people you are. It just takes thinking of you to turn hard days into good ones. Laurel, you are the source of so many fits of laughter, good music, great conversations, dance parties, and encouragement. I am proud to call you my sister, lucky to call you my friend, and blessed to be able to call you both. Thanks for showing me the ropes, Little Big. Dad, thank you for believing in me and for always being so supportive of whatever it is I am doing. Whether on a sports fields, in a classroom, or in the arena of life, it means the world to me to have my biggest fan cheering me on. Mom, thank you for role modeling generosity, kindness, hard work, and a fearless commitment to being exactly who you are. It is a gift to have learned how to walk in the world from such a remarkable teacher.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 GRID OF THE FUTURE

In 2003, the National Academy of Engineering recognized electrification in the United States—and the infrastructure that made it possible—as the greatest achievement of the 20th century [1]. While still remarkable, the U.S. electric grid of the 21st century looks almost exactly like that of the 20th century, and though the electricity still flows through the aging system, it is doing so less reliably and efficiently. In fact, among all developed nations, the United States has the highest number of outage minutes, which not only leads to annoyed customers, but also economic loss for businesses and utilities [2].

Less reliable service is not the only shortcoming of the aged 20th century electric grid. Cyber and physical attacks, the environmental impact of energy production, updating the digital and communication network, dealing with more extreme weather, and leveraging data for flexibility and controllability, are additional growing pains the future grid needs to address. So, though the challenges are daunting, the great minds of the 21st century must rise to the challenge, just as those of the 20th did. Fortunately, the pursuit is underway.

Just what is meant by the future electric grid (many refer to this as the “smart grid”)? It means an electric grid that is able to integrate a large amount of renewable energy sources, despite their intermittency. It means an electric grid that has the latest, greatest, and most secure communication and digital infrastructure. It means an electric grid that can handle the new paradigm of distributed generation—think solar panels on houses—rather than centralized generation—think large coal plant located in rural America sending power to cities. It means an electric grid that is more flexible and able to adapt to changes in generation and load more quickly. It means an electric grid that can bounce back from a problem with ease. It means an electric grid that provides more information, more quickly, to operators and equipment [3]. In short form, the above list could be summarized by the following bullet points. The future electric grid is:

- More environmentally friendly
- More diverse and flexible
- More reliable, secure, and resilient

- More intelligent

Each of the aforementioned bullet points deserves a lengthy dialogue, but for the sake of brevity and emphasis, my thesis will focus on the latter—the intelligence of the future electric grid. More specifically, I will focus on data and how it is used to improve operation of the electric grid. However, just to be sure we cover all of our bases, [4], [5], and [6] present a great overview of the smart grid, and address the areas of environmental responsibility, flexibility, intelligence, reliability, renewables, resilience, and security.

Grid intelligence requires that information about the state of the electric grid be collected and shared. For the 21st century, the added requirement for grid intelligence is that more information be shared, more quickly, and more securely. In order to collect more information, more sensors have been (or are being) deployed to collect measurements throughout the U.S. electric system. At the transmission system level, these sensors are called synchrophasors or phasor measurement units (PMUs). Similar sensors, like smart meters, are being deployed to homes at the distribution level. Additionally, the grid's communication infrastructure, both at the transmission and distribution levels, is being improved to ensure that all the new sensor data can reach its end user quickly and safely.

With many sensors already installed and a communication infrastructure mostly in place, the next step, and what I will focus on in my thesis, is what is being done with the data that is collected by all the new sensors. To truly claim intelligence, it is not enough for the grid just to collect and share data, but the data must enable new insights and drive better decisions. That is the challenge we find ourselves facing today. Utilities, whether at the transmission or distribution level, have access to more data, but lack the means to turn it into something really meaningful.

It is precisely that challenge that was posed to our team. Can you translate data collected from PMUs into something that helps our operators make better decisions and improves their understanding and awareness of the system they are operating?

1.2 GRID OPERATION AND PHASOR MEASUREMENT UNITS

Before I delve into the problem, I want to spend a little more time introducing just what PMUs are and how the grid is operated. From here on, when I refer to the electric grid, it is from the perspective of the transmission system, unless otherwise noted.

In the transmission system, operation of the grid requires maintaining the system frequency to be near 60 Hz, balancing tie-line flows (transmission lines that carry electricity between neighboring utilities), and ensuring line currents, equipment loading, and voltage levels are within their limits [7]. Duties of, and decisions made by, operators largely revolve around the above. Thus, to improve operator's performance, data must provide keener insights about one or more of those operational objectives.

Historically, operators have relied on information gleaned from supervisory control and data acquisition (SCADA) systems, which has been sufficient for normal, localized operation of the grid. But, when trouble arises, more precise, frequent, time synchronized, and widespread measurements can help. This is where PMU data comes in. Unlike SCADA measurements, which are collected every 2 to 4 seconds, PMUs record data at a minimum of 30 times per second, with each measurement time stamped using the global positioning system (GPS). This data includes voltage magnitude and angle, current magnitude and angle, and frequency, measured at the buses—conductive connection points between electrical equipment—where they are installed. The precise GPS timestamp allows data from many PMUs to be integrated to provide comprehensive visibility of an electric grid and its neighboring systems. That widespread visibility is referred to as wide-area situational awareness, and it is a key reason for using PMUs, because it allows operators to see problems as they develop in neighboring systems, before they reach their own. In fact, one of the main reasons for the Northeast Blackout of 2003 was because utilities did not have enough visibility and knowledge of their neighboring utilities' systems. PMUs circumvent that problem by delivering data to all parties who want it. Figure 1.1 plots the time scales for various power system events, and the monitoring capabilities of PMUs and SCADA systems.

PMUs are installed at substations and record measurements for multiple buses within a substation. A typical PMU installation is shown in Figure 1.2, where potential and current transformers connect to various buses in the substation to record voltage and current measurements. A single PMU has multiple channels through which measurements can be recorded, allowing one PMU to collect measurements for multiple buses. Once measurements are made by a PMU, their data is sent to a regional hub, called a phasor data concentrator (PDC). PDC's time-synchronize and aggregate data from many PMUs, using the timestamp assigned to the data by the PMU's GPS clock. Once aggregated, the data is then sent to another PDC, for further aggregation, or to a central control room for an operator to deal with.

As of 2015, more than 1,500 networked PMUs were installed in North America, and more than 225 PDCs, as a result of funding from the American Recovery and Reinvestment Act of 2009. This larger PMU infrastructure has given operators near "100% visibility of the entire U.S high-voltage transmission network", a better ability to "diagnose failing or mis-operating equipment", and has helped prevent outages [9]. The report of [9] goes on to say that, "As utilities become more familiar with (PMUs), they will increasingly use (their) data to inform a broader range of operating decisions" provided new software is developed to take advantage of the "rich inventory of synchrophasor data".

Because of that need, the intent of this thesis was to help build the software tools that will translate PMU data into actionable insights for operators.

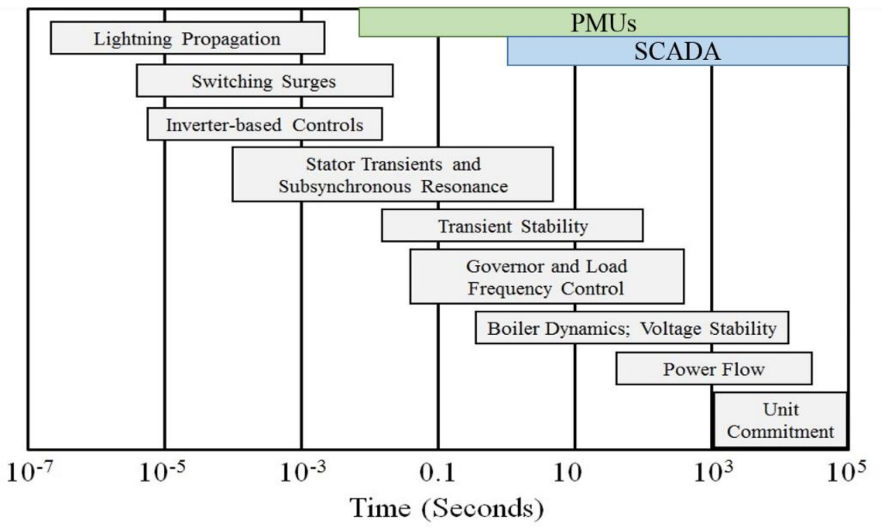


Figure 1.1: Power system time scales and measurement capabilities. [8]

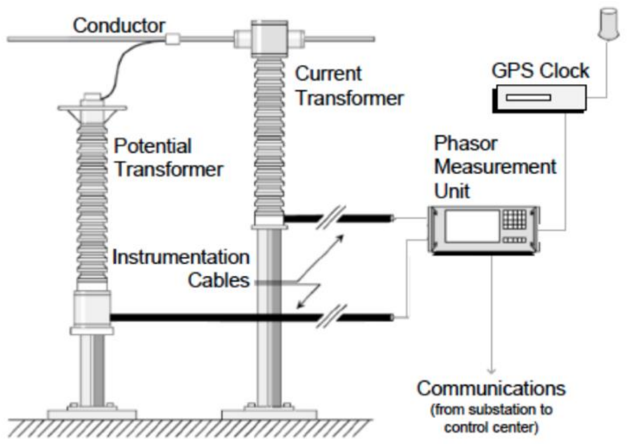


Figure 1.2: PMU substation installation diagram. [9]

1.3 PROJECT OVERVIEW

A team at the Rochester Institute of Technology and the University of Illinois at Urbana-Champaign is conducting research on how to better leverage synchrophasor data to help operators of the electric grid [10]. The project is being led by Dr. Esa Rantanen, Associate Professor of Psychology and Research Associate Professor of Industrial and Systems Engineering at the Rochester Institute of Technology, and co-led by Dr. Thomas Overbye, Emeritus Professor at the University of Illinois at Urbana-Champaign. Specifically, the scope of work states that new data visualizations will be created for PMU data, in order to improve operator situational awareness and operator decision making. Other elements of the statement of work, include (1) a cognitive work analysis of operators to better understand how they work and how they currently use their own data visualizations and (2) testing of each created visualization to determine how effective it is in improving situational awareness and decision making. However, this thesis will focus only on the University of Illinois's contribution to creating PMU data visualizations.

The creation of visualizations was a multifaceted process which included creating a power system model, creating problematic scenarios to replicate situations an operator would face, running simulations of the problematic scenarios and recording data as would be recorded by PMUs, cleaning and formatting that data, and then, finally, creating visualizations of PMU data. For clarification, we did not use data actually collected from PMUs, but instead created our own data to replicate what could be collected by real PMUs, so that the method was generalizable for other systems. This allowed us to test a variety of problem scenarios without the requirement that they had actually occurred in the real grid and allowed us to focus on creating visualizations instead of dealing with erroneous data from real PMUs. Error checking is another area of research being conducted, but not for this project. Figure 1.3 provides a simplified overview of our project as compared to the operation and data collection process of the real grid.

In the first step of the project, a power system model was created and simulated using the software package, PowerWorld. Data from each simulation was saved in csv files, before being sent to a software tool, created specifically for this project. In the second step, the software tool cleaned and formatted the data to resemble what would be collected by real PMUs. Third, clustering was performed to uncover hidden patterns in the data. Last, visualizations were created.

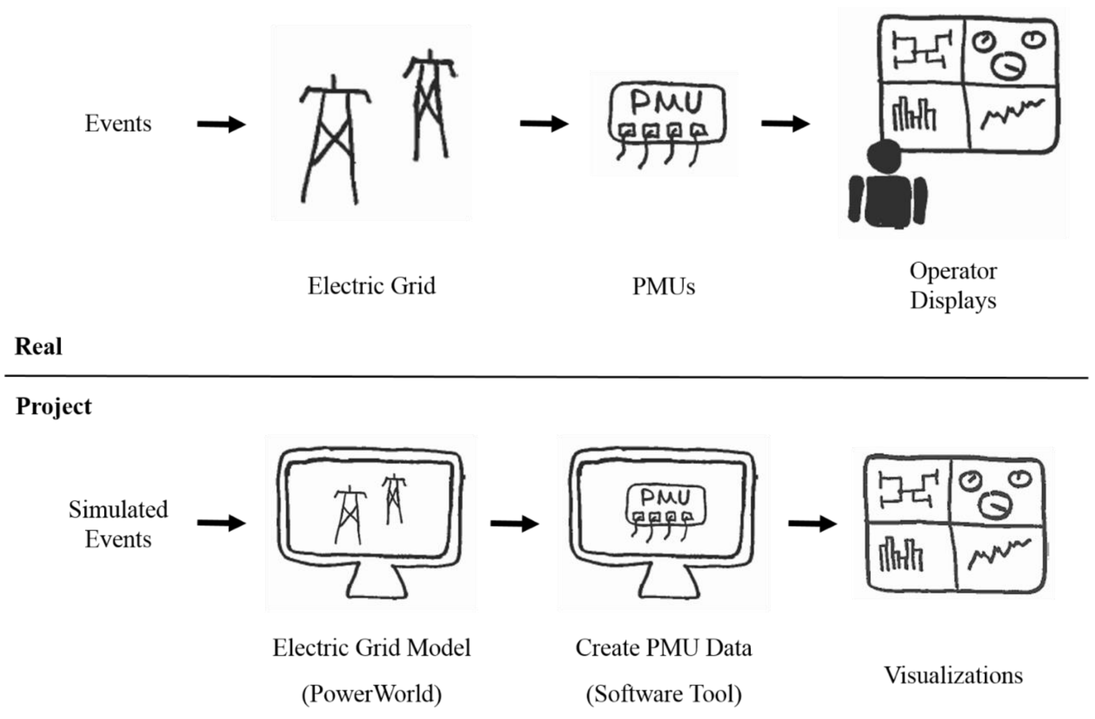


Figure 1.3: Comparison of the data and operational flow of a real electric grid and our project.

1.4 THESIS ORGANIZATION

The remaining document outlines the steps taken by the University of Illinois team to create the project deliverables. Chapter 2 outlines the processes used to make an equivalent power system model and to make scenarios to test the system. Chapter 3 describes the data cleaning and formatting process, performed to create data that represents what would be collected by real PMUs. Chapter 4 documents clustering techniques that were used to further classify and reduce the data. Chapter 5 discusses how each visualization was created. Chapter 6 concludes the thesis and offers suggestions for future work. The appendix provides a user guide for the software tool that automates and generalizes the content of Chapters 3, 4, and 5, so it can be used in projects beyond this one. Definitions and icons that will be used throughout the rest of the document are contained in Figure 1.4.

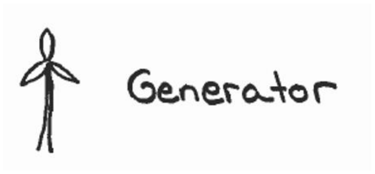
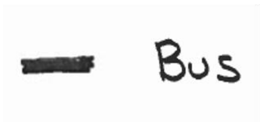
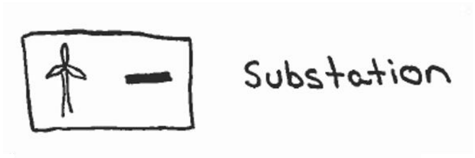
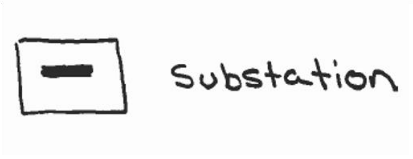
	This symbol is used to represent a generator of any fuel type.
	This symbol is used to represent electrical buses, which are an electrical connection point between conducting equipment.
	This symbol is used to represent substations that have a generator operating within them.
	This symbol is used to represent substations that do not have a generator operating within them.

Figure 1.4: Definitions and descriptions of icons that will be used throughout the text.

CHAPTER 2
POWER SYSTEM MODEL AND SIMULATIONS

The first step of the research process required creating a model for the power system under study, and then creating problematic simulations to run on it. This section documents both of those components.

2.1 EQUIVALENT POWER SYSTEM MODEL

The first requirement for the University of Illinois team was to create a large-scale power system model. This was done by creating an equivalent network from a model of the entire Western Interconnect. A separate equivalent model was created for several reasons. First, we wanted to focus on a particular operating area of the Western Interconnect. Second, we wanted the new model to look as similar to the original one as possible. Third, by using a smaller model, simulations and data processing could be completed more quickly. Finally, because we were focused on PMU data, it did not matter whether the total number of buses was 1,000 or 22,000, since the data would be reduced to only those buses that are included in a fixed set of buses being monitored by PMUs. The above justifications for creating an equivalent power system model hold true in other projects, as well, especially because doing so reduces the computation time required to conduct simulations and reduces the amount of data that needs to be processed.

To create an equivalent model, the equivalencing process of [11] was used, which requires that buses be split into study buses and external buses. Study buses are those that are being kept in the new model (for study), and external buses are all of those that are not being investigated or are out of the scope of the desired geographic footprint. The equivalencing process is somewhat like filtering, in that certain buses are kept and the rest discarded. Though unlike filtering, information from the discarded buses is embedded in the new model, via transmission line parameters—resistance, reactance, and susceptance—and loads that are assigned at the study system’s boundary buses. These boundary buses are buses in the study system that normally connect to the external system via transmission lines. Figure 2.1 illustrates the equivalencing technique. It shows that the resulting equivalent model consists of only buses that are in the study system, and that all traces of the external system are confined to transmission lines and loads.

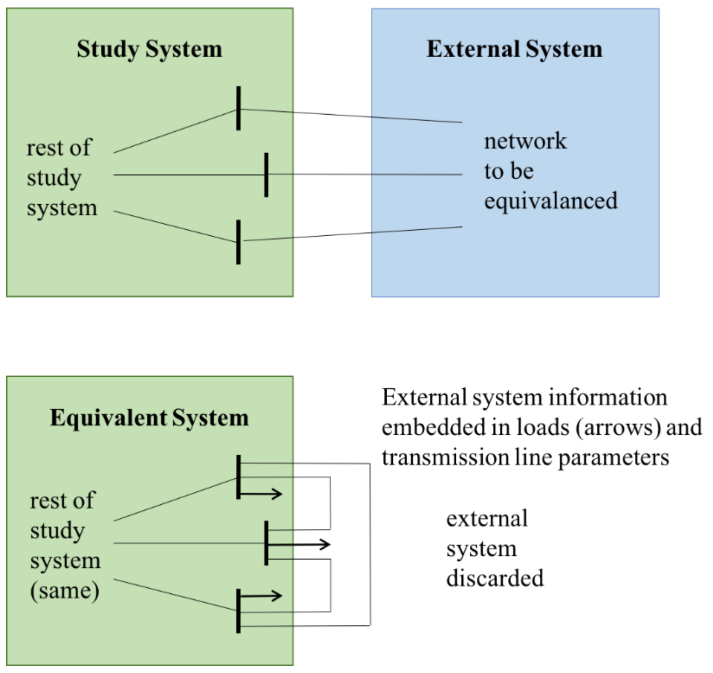

Study System
External System
rest of study system
network to be equivalanced
Equivalent System
External system information embedded in loads (arrows) and transmission line parameters
rest of study system (same)
external system discarded

Figure 2.1: Diagram of the power system equivalencing process. The top blocks are the original system, and the bottom, the result after equivalencing. See [11] for a more comprehensive description.

Buses that fit the bullet point criteria, below, were considered "study" buses, and the rest were treated as "external" buses. The resulting model was reduced from over 22,000 buses, for the entire Western Interconnect, to about 1,300 buses.

- All buses with generator's larger than 300 MW
- All DC lines and the buses they are connected to
- All buses in the operating area of interest, with voltages above 230 kV
- Several heavily used buses on the outskirts of the operating area of interest

2.2 SIMULATION DESCRIPTIONS

In order to create data visualizations and then test their effectiveness, power system data was needed. To generate data, situations that mimic those that an operator might face in real life were created. These scenarios were designed to replicate power system events that can arise from storms, attacks, or other physical equipment damage.

For each scenario, two simulations were created that lasted 60 seconds and 120 seconds, and were run in a power system software simulation package. Frequency, voltage angle, voltage magnitude, and current magnitude were measured at each bus in the 1,300 bus equivalent model, 30 times per second. These types of data values are the same as those that could be recorded by a PMU. Current angles were not considered, partly due to time constraints and partly because we felt they would not provide extensive information beyond what was already known from voltage angles. Data was saved in separate csv files corresponding to its type—frequency, voltage magnitude, voltage angle, and current magnitude. See Section A.4 of the appendix for more information about file format.

Within the software simulation package we used, there is a limited number of settings that can be adjusted to create problematic events for a power system model. For example, there is not a way to tell the program to induce an earthquake on a particular area of a system. So, to create a problematic event, like an earthquake, we had to choose from settings available in the package. These adjustable settings include increasing or decreasing the amount of demand or generation in a model; faulting (short circuiting) certain buses, lines, or transformers in a model; disconnecting one or more of a model's buses, generators, loads, lines, and transformers; or re-connecting one or more of a model's out-of-service buses, generators, loads, lines, and transformers. The combination of settings we used to create four problematic scenarios is briefly described in the following subsections, so they can be replicated in other systems.

2.2.1 Substation Attack

Motivated by the California Metcalfe incident, several transformers were taken out of service at one of the model's largest substations. Nothing else was done besides taking the transformers offline.

2.2.2 Earthquake

Buses and transmission lines were faulted and/or removed from service in order to mimic what could happen during an earthquake along Oregon and Washington's Pacific coast. Line faults were used to simulate real transmission lines being shaken to the ground or to another phased line. It was assumed that most of the faults could not be cleared because they were lying on the ground or touching another line. As would be done in the real grid for persistent faults, the faulted lines were opened to stop the short circuit. Transformers were opened to simulate physical damage caused by shaking. In Figure 2.2, the area to the left of the black line shows where the most damage was incurred.

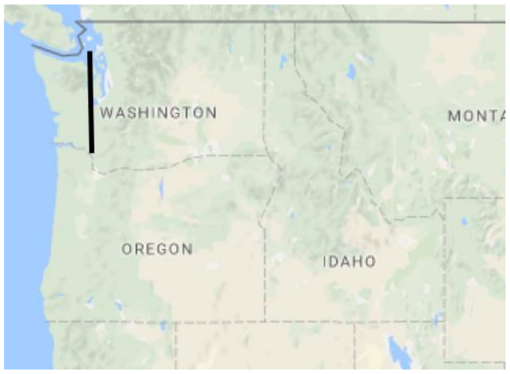


Figure 2.2: A map showing the region affected by a simulated earthquake, to the left of the line.

2.2.3 Ice Storm

Transmission line faults and openings were used to mimic what could happen during an ice storm. Wind and/or the weight of ice was assumed to weigh down transmission lines, such that they broke from their towers and fell to the ground, causing line faults. Just like in the earthquake simulation, for faults that could not be cleared, the affected transmission lines were opened. The area affected by the ice storm is shown in Figure 2.3.

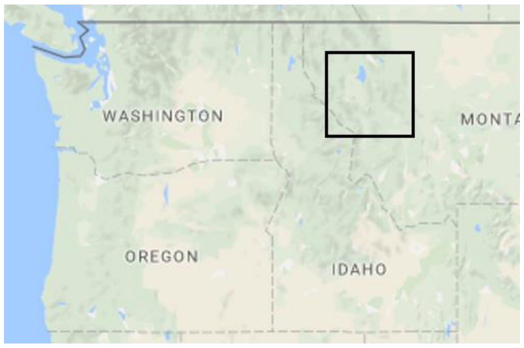


Figure 2.3: A map showing the region affected by a simulated ice storm, within the box.

2.2.4 Geomagnetic Disturbance

The software simulation package includes a tool to simulate geomagnetic disturbances, which was used to create a geomagnetic storm to act on the equivalent model. Additionally, several large transformers were opened during the simulation as if they had overheated from high currents induced by the geomagnetic storm. Several loads were shed too, as might occur during a real geomagnetic storm. Varying field strengths of 6 V/km at 60 degrees, 8 V/km at 75 degrees, and 10 V/km at 80 degrees were applied to create the storm. It was not paramount that the storm be totally realistic, but instead that it provide what an operator *could* see during a geomagnetic storm.

CHAPTER 3
DATA COLLECTION AND PROCESSING

Once the equivalent model and simulations were completed, the data collected from each simulation was processed. Data processing included (1) formatting the data to have uniform and concise identifiers and (2) extracting data that corresponded to what PMUs would collect in the real system. To automate data processing, a software tool was created by the University of Illinois team. The tool was built to work for other simulations or power models beyond this project, so as to streamline the process between simulation data and analysis. The data processing functionality of the software tool is described in this chapter, and a complete user guide for the tool is given in the appendix.

3.1 SIMULATION AND MODEL DATA AND FORMATTING

As mentioned in Chapter 2, measurements for frequency, voltage magnitude, voltage angle, and current magnitude were recorded at every bus in the model system, 30 times per second. Data for each measurement type —frequency, voltage magnitude, voltage angle, and current magnitude —was saved directly from the software simulation package into a csv file. Frequencies were measured in hertz, voltage and current magnitudes in per unit, and angles in degrees. The aforementioned software tool was used to adjust formatting of each dataset, so that bus numbers were used as the column names and the timestamp for each observation as the row name.

In addition to measurement data, information about the power system being modeled in the simulation package, including information for buses, substations, and generators was saved in separate csv files. This data, like the measurement data, was re-formatted and all extraneous information discarded. The pertinent information kept for each power system component was, for buses, the bus name, number, nominal voltage level, and substation it belonged to; for generators, the bus number it was connected to, the substation it belonged to, and its MW generating capacity; and for substations, the substation identifier (number), operating area it belonged to, and geographic latitude and longitude coordinate. Substation and bus information were merged to form one dataset that contained the bus name and number, substation identifier, nominal voltage, operating area name, and geographic latitude and longitude coordinates.

The goal of this step was to make more concise variable identifiers and a uniform format between measurement types and model information. The resulting data was considered the complete dataset, from which PMU data was derived, as is described in the next section. The appendix provides more information about the data input to and output from the software tool.

3.2 PMU DATA PROCESSING

As mentioned in the previous section, measurements for frequency, voltage magnitude, voltage angle, and current magnitude were recorded at every bus in the model system. However, for our research, we were only interested in data that would be provided by PMUs, which, in general, are not installed at every bus in a system. To reconcile this, an algorithm was created to reduce the number of buses from which data was collected, to more realistically represent what would be measured by PMUs. Figure 3.1 diagrams this idea where simulated data is shown at the top with larger boxes, representing the larger number of buses from which data was collected, and the filtered PMU data shown at the bottom with smaller boxes, to represent the fewer number of buses from which PMU data was collected. The PMU data processing algorithm is a two-step process, which begins by creating a list of substations and buses where PMUs are actually located or where they should fictitiously be assigned. The second part of the algorithm extracts data from the complete set of simulation data, for only the buses included in the list made in step 1. Step 1 is described in Section 3.2.1 and step 2 in Section 3.2.2.

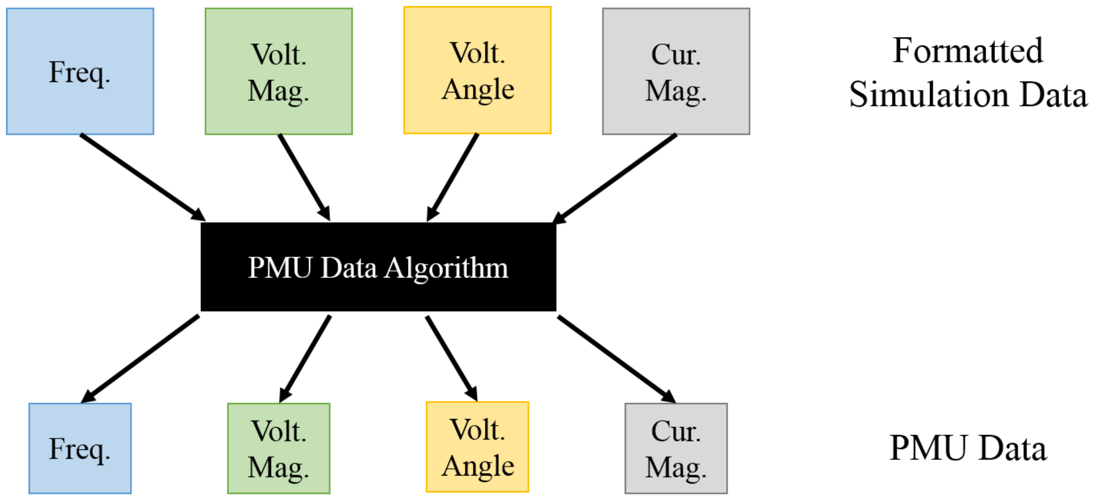


Figure 3.1: A diagram of the input data to and from our software tool's PMU data processing algorithm. Data from every bus in the simulation is input to the algorithm and a select portion of that is returned to represent what would be collected by actual PMUs.

3.2.1 Step 1: PMU Substation Identification and Assignment

In the first step of the two-part data processing algorithm, substations where PMUs were known, or were likely, to be located in the real system, were identified, and then a list of buses contained within those substations was generated.

To do this, the algorithm first looked for the user-defined number of substations where PMUs were actually installed in the real power system, or the number of PMUs desired by the user. That number is represented by N. Next, the algorithm checked if there was a csv file that contained information where PMUs were actually located. The contents of that csv file were the substation name, in the first column, and the nominal substation voltage, in the second column. If there was such a file, the software tool assigned the number of known PMUs, in that file, to a value, K, and the substation and its corresponding information—identifier, voltage level, and geographic coordinates— to a data structure, *pmu_subs*. Otherwise, K was set to zero and *pmu_subs* remained empty. Substation information was gathered by cross checking the known substation identifiers, with the model data from Section 3.1.

With K assigned, the program then checked if the number of known PMUs (K) was equal to the required number of PMUs (N). If so, the PMU substation identification and assignment process was complete. If not, the difference in known and needed PMUs was calculated, and the result was the number of PMUs that needed to be assigned (A) by the program, such that $N = K + A$. This process is shown in Figure 3.2, and the PMU substation assignment process is described more fully in the next paragraphs.

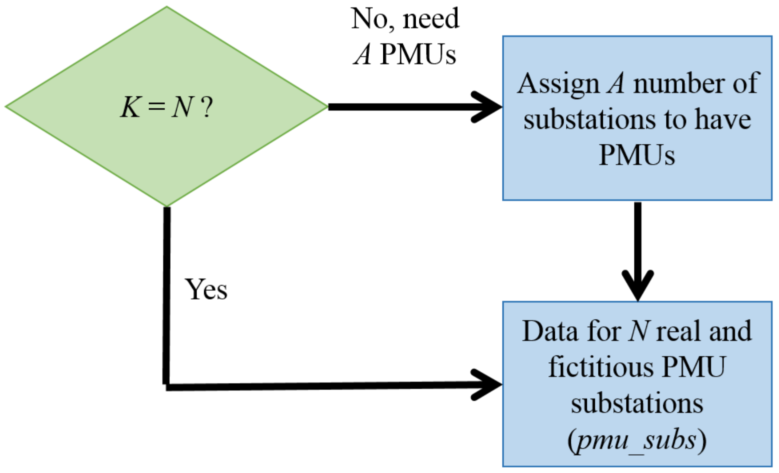

K = N ?
No, need A PMUs
Assign A number of substations to have PMUs
Yes
Data for N real and fictitious PMU substations (pmu_subs)

Figure 3.2: An overview of the process for identifying or fictitiously assigning PMUs to a substation, and then generating the required data for those substations.

When *A* was non-zero, the assignment process for unknown PMU locations was initiated. It identified substations that were most likely to have PMUs installed and fictitiously assumed there to be a PMU at each of those substations. The approach was based on the notions of [12] which suggests that PMUs are most often found at substations with a generator and which have buses operating at the highest and most widespread system voltages. By widespread, I mean that if there happens to be only a few buses at a high voltage, the program will ignore those, and instead use the next lesser voltage level that has a greater number of buses. In the model we used, there were a few buses at 345 kV, but a lot more at 230 kV and 500 kV, so those were the two voltage levels used for PMU assignments. We assume PMUs monitor only two voltage levels.

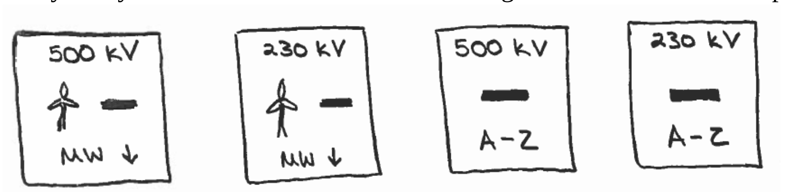
The PMU substation assignment approach is described by the numbered steps in Figure 3.3. At each step, the substation, where PMUs were fictitiously assigned, and its corresponding information, were saved to the data structure, *pmu_subs*. Note that in our assignment algorithm a PMU will only be assigned to one unique voltage level, so if a substation has buses at both 230 and 500 kV, two PMUs would be needed. The assignment process was complete once the number of substations assigned was equal to *A* and consequently the number of substations in *pmu_subs* was equal to *N*. For the system we studied, there were 126 PMUs installed [13]. Of those 126 PMUs, 56 substations were known to have PMUs from the public document of [14], but the remaining 70 had to be assigned, since their exact locations were not publicly known. Note, however, that the location of substations in the system we studied is public, as shown in [15].

Figure 3.4 summarizes the PMU substation assignment process thus far, where the data structure *pmu_subs* was created from known PMU substations, contained in a csv file, and fictitiously assigned substations, if applicable, created in the assignment process outlined in Figure 3.3. The data structure *pmu_subs* contained data—substation identifier, nominal voltage level, and geographic coordinates—for both known and fictitiously assigned PMU substations.

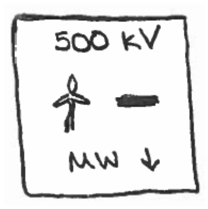
With the dataset of PMU substations complete, a more granular list of buses that should be monitored by each PMU needed to be created. To do this, we assumed that each PMU had enough channels to monitor all buses in a substation, for a unique voltage level, as exampled in Figure 3.5. Then, for each substation in *pmu_subs*, all buses within that substation that operated at either the highest and/or second highest voltage were saved to a list called *pmu_buses*. Figure 3.6 shows this list making process for the example of Figure 3.4. Once the list of buses was made, it was used in the next step of the PMU data processing algorithm, for data extraction.

STEP 1
Identify the two highest and most widespread voltage levels in the power system model. This is done automatically by the algorithm or can be defined by the user. In our model, the two highest and most widespread voltages levels are 500 kV and 230 kV.

STEP 2
Sort all substations into categories based on their voltage level and whether or not they have a generator. Once sorted, arrange the substations, with generators, in order of the largest MW generating capacity and all other substations alphabetically. Only substations with buses at the voltage levels identified in Step 1 are considered.

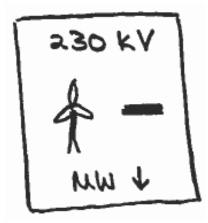


STEP 3



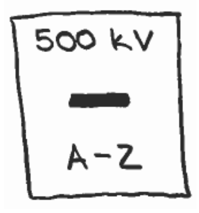
Assign PMUs to each substation that has both a generator and buses operating at the highest and most widespread system voltage, eg. 500 kV in our model.

STEP 4



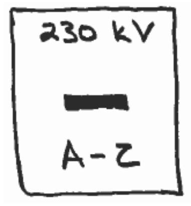
If all PMUs are assigned, the process is complete, and all substations with a real or fictitious PMU are contained in *pmu_subs*. If not, assign PMUs to each substation that has both a generator and buses operating at the second highest and most widespread system voltage, eg. 230 kV in our model.

STEP 5



If all PMUs are assigned, the process is complete, and all substations with a real or fictitious PMU are contained in *pmu_subs*. If not, assign PMUs to any remaining substations that contain buses at the highest and most widespread system voltage, eg. 500 kV in our model.

STEP 6



If all PMUs are assigned, the process is complete, and all substations with a real or fictitious PMU are contained in *pmu_subs*. If not, assign PMUs to any remaining substations that contain buses at the second highest and most widespread system voltage, eg. 230 kV in our model. At the conclusion of this step, all substations with a real or fictitious PMU are contained in *pmu_subs*.

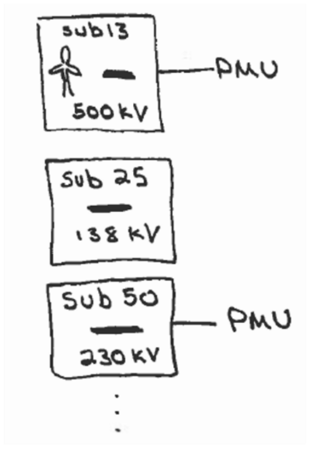
Note: In step 2, alphabetical ordering for non-generator substations was used to ensure repeatability between simulations, and is the default for the program. If repeatability is not desired, no ordering need be done, but modifications to the program would need to be made.

Figure 3.3: Steps of the PMU substation assignment process.

Known PMU Substations ***(K)***		**Assigned PMU Substations** ***(A)***		**All Real and Fictitious PMU Substations** ***(N)***

Sub. ID	Nominal Voltage (kV)
sub1	230
sub1	500
sub5	230
…	…

+



=

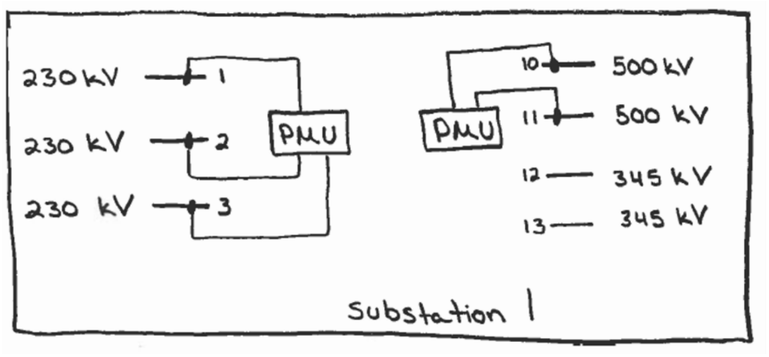
Sub. ID	Nominal Voltage (kV)	Latitude, Longitude
sub1	230	lat1, long1
sub1	500	lat1, long1
sub5	230	lat5, long5
sub13	500	lat13, long13
sub50	230	lat50, long50
…	…	…

from csv file

from process in Figure 3.3

resulting data structure *pmu_subs*

Figure 3.4: Overview of the process to gather information about the substations that are known to have, or are fictitiously assigned, a PMU.



Sub. ID	Nominal Voltage (kV)	PMU Bus Numbers
sub1	230	1, 2, 3
sub1	500	10, 11

Figure 3.5: An overview of a substation's layout and what buses would be monitored by a PMU.

Sub. ID	Nominal Voltage (kV)	PMU Bus Numbers
sub1	230	1, 2, 3
sub1	500	10, 11
sub5	230	15, 16, 17
sub13	500	23
sub50	230	64, 65
…	…	…

[1, 2, 3, 10, 11, 15,16,17, 23, 64, 65]

pmu_buses

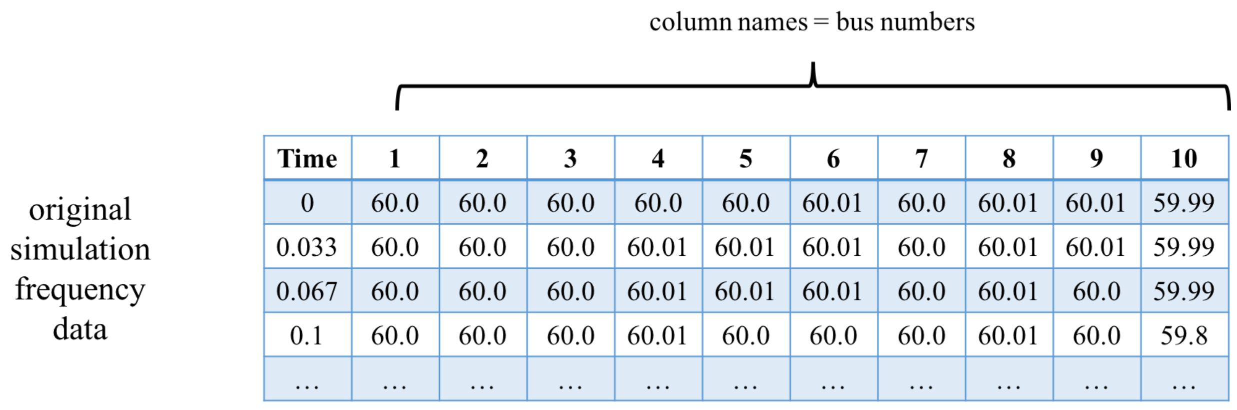
Figure 3.6: The process of generating a list of buses, *pmu_buses*, being monitored at each PMU substation.

3.2.2 PMU Data Extraction

As was mentioned previously, the main goal of the PMU algorithm was to generate realistic data like that collected from PMUs in an actual system. To do this, the number of buses from which measurements were collected had to be reduced to represent only what would be collected by PMUs. In step 1 of the PMU data processing algorithm, a list of buses that the PMUs were monitoring, called *pmu_buses*, was created. The second step is then to create the actual set of PMU data.

The *pmu_buses* list was used to select from which buses, of the complete set of simulation data, measurements should be extracted. In other words, any bus contained in both *pmu_buses* and the data set for frequency, voltage angle, voltage magnitude, or current magnitude, was saved in a new data structure to represent PMU data. Recall, from Section 3.1, that simulation data includes frequency, voltage angle, voltage magnitude, and current magnitude at every bus in the model.

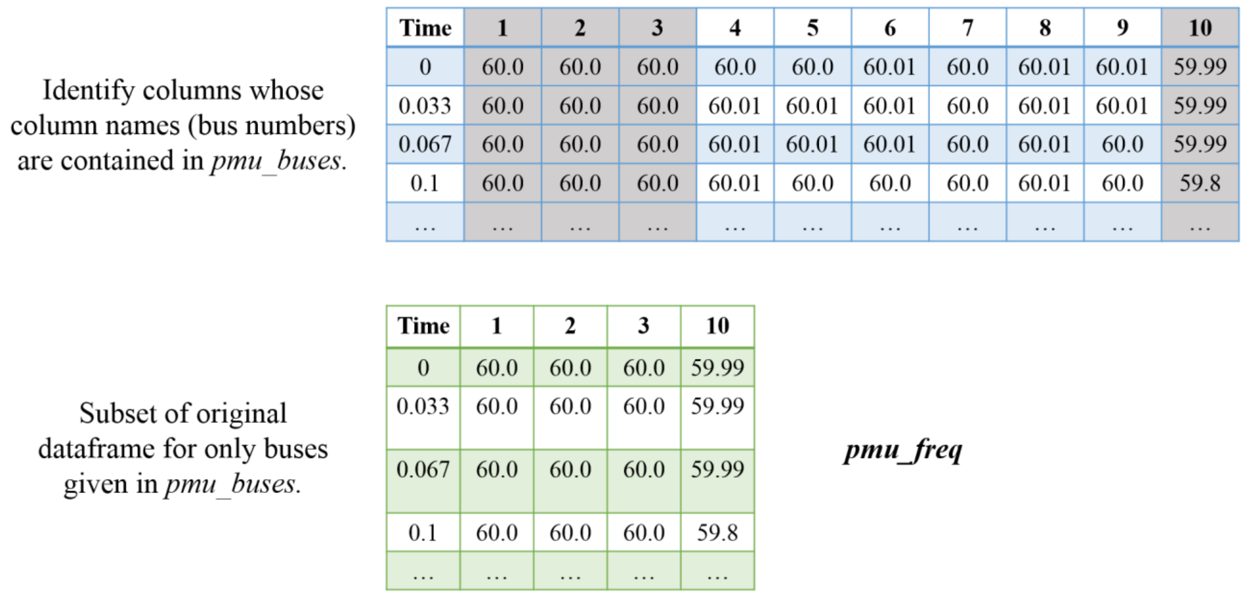
As an example of the data extraction process, see Figure 3.7, which shows fake frequency simulation data and a fake list of buses that are being monitored by PMUs. To extract only PMU data from the simulation data, the buses (columns) that are shared between the frequency simulation data and buses in *pmu_buses* are identified (gray). Data from those columns is extracted and saved to a new data structure, as shown in Figure 3.8. The resulting data structure is saved in a csv file with the naming convention of pmu_datatype, where datatype is one of "freq", "vang", "vmag", or "cmag". This data extraction process must be performed for each measurement type —frequency, voltage angle, voltage magnitude, and current magnitude.


column names = bus numbers
original simulation frequency data

Time	1	2	3	4	5	6	7	8	9	10
0	60.0	60.0	60.0	60.0	60.0	60.01	60.0	60.01	60.01	59.99
0.033	60.0	60.0	60.0	60.01	60.01	60.01	60.0	60.01	60.01	59.99
0.067	60.0	60.0	60.0	60.01	60.01	60.01	60.0	60.01	60.0	59.99
0.1	60.0	60.0	60.0	60.01	60.0	60.0	60.0	60.01	60.0	59.8
…	…	…	…	…	…	…	…	…	…	…

pmu_buses [1, 2, 3, 10]

Figure 3.7: Example data, including "fake" frequency simulation data and a "fake" list of buses being monitored by a PMU, *pmu_buses*.



Identify columns whose column names (bus numbers) are contained in *pmu_buses*.

Time	1	2	3	4	5	6	7	8	9	10
0	60.0	60.0	60.0	60.0	60.0	60.01	60.0	60.01	60.01	59.99
0.033	60.0	60.0	60.0	60.01	60.01	60.01	60.0	60.01	60.01	59.99
0.067	60.0	60.0	60.0	60.01	60.01	60.01	60.0	60.01	60.0	59.99
0.1	60.0	60.0	60.0	60.01	60.0	60.0	60.0	60.01	60.0	59.8
…	…	…	…	…	…	…	…	…	…	…

Subset of original dataframe for only buses given in *pmu_buses*.

Time	1	2	3	10
0	60.0	60.0	60.0	59.99
0.033	60.0	60.0	60.0	59.99
0.067	60.0	60.0	60.0	59.99
0.1	60.0	60.0	60.0	59.8
…	…	…	…	…

pmu_freq

Figure 3.8: An example of the process used to select which buses data should be saved from the original simulation data. Buses that are contained in the list *pmu_buses* will be saved.

The data processing and PMU identification, assignment and data extraction process, outlined in this section and Section 3.1, works for any power system, and is contained within the custom software tool we created. It is flexible enough to accommodate cases where PMU locations are known and cases when they are not. A quick summary of the data input and output from the entire data processing step is shown in Tables 3.1 and 3.2.

From here on, any time the terms PMU or PMU data are used, it is with regard to the *N* real and/or fictitiously assigned PMUs generated in this process, unless otherwise connoted by the context. From Tables 3.1 and 3.2, PMU data equates to the data saved in pmu_freq, pmu_vang, pmu_vmag, and pmu_cmag. Analysis of the PMU data is discussed in Chapters 4 and 5.

Table 3.1: Data files and descriptions for input to the data processing portion of the software tool.

File Name (.csv)	Type	Description
buses	Case	Contains bus name and number, voltage level, and substation identifier, for every bus in the system.
subs	Case	Contains substation identifier, operating area name, and geographic coordinates, for every substation in the system.
gen	Case	Contains bus number, MW generating capacity, and substation identifier for every generator in the system.
real_pmus	Case	If known, contains a list of substations where PMUs are actually located in a real system, and their corresponding voltage levels
bus_freq	Simulation	Frequency measurements, in Hz, recorded at every bus in the PowerWorld model
bus_vang	Simulation	Voltage angle measurements, in degrees, recorded at every bus in the PowerWorld model
bus_vmag	Simulation	Voltage magnitude measurements, in per-unit, recorded at every bus in the PowerWorld model
branch_cmag	Simulation	Current magnitude measurements, in per-unit, recorded coming in to every bus in the PowerWorld model

Table 3.2: Data files and descriptions for output from the data processing portion of the software tool.

File Name (.csv)	Type	Description
bus_info	Case	Contains bus name and number, voltage level, substation identifier, and geographic coordinates of the substation, for every bus in the system.
gens	Case	Contains list of all buses at substations with a generator, total MW generating capacity, substation identifier, for every generating substation in a system
pmu_info	Case	Contains bus name number, voltage level, substation identifier, operating area name, and geographic coordinates for every bus/substation being monitored by a real or fictitious PMU
pmu_freq	Simulation, PMU	Frequency measurements, in Hz, recorded at every bus being monitored by a PMU
pmu_vang	Simulation, PMU	Voltage angle measurements, in degrees, recorded at every bus being monitored by a PMU
pmu_vmag	Simulation, PMU	Voltage magnitude measurements, in per-unit, recorded at every bus being monitored by a PMU
pmu_cmag	Simulation, PMU	Current magnitude measurements, in per-unit, recorded coming in to every bus being monitored by a PMU

CHAPTER 4
CLUSTERING AND OUTLIER DETECTION

With PMU data generated, the work of turning it into actionable insights began. This work included clustering, described in this chapter, and visualizations, described in Chapter 5. Clustering was used for two reasons: (1) to reveal any patterns or features hidden in the data, and (2) as a way to condense a large amount of data into summaries of it. For operators, that big picture perspective is preferable to more detailed knowledge about every component in a system [16], which contributes to the common operational challenge of information overload.

Two clustering techniques were used: density-based spatial clustering of applications with noise (DBSCAN) and K-means. An overview of each technique is presented, and then a description of how each was used in our research is provided.

4.1 DBSCAN

DBSCAN clustering is a technique that sorts data into groups that are densely packed together, and those that are not. A detailed procedure and visualization of the DBSCAN algorithm is provided in [17], from which I have created the following summary. The average run time complexity for the DBSCAN algorithm is O(n*log(n)), where n is the number of data points being clustered [18].

Before the algorithm begins, the user must define two numbers. The first is a number that specifies how many points are required to be near a point, in order for that point to be considered a part of a cluster. This number is called *min_points*. The second number, *epsilon*, tells how far away from the original point the algorithm can look for other points to meet the *min_points* requirement.

With the parameters for *min_points* and *epsilon* defined, the algorithm begins by selecting an arbitrary data point from the dataset. From that data point, the algorithm looks for points within a radius of *epsilon*. If it finds at least *min_points* number of points within that radius, including itself, all those points are added to a cluster. If there is a point that does not have at least *min_points* around it, within a radius of *epsilon*, that point is considered noise.

Then, for each of the points that were just added to the cluster, the algorithm looks for at least *min_pts* number of points, within a radius of *epsilon*, just as was done for the initial arbitrary point.

This process repeats for each point iteratively added to a cluster, until there are not any more points that have at least *min_points* around them, within a radius of *epsilon*. In other words, once the cluster surrounds itself by noise points, the cluster is complete.

Upon completion of a cluster, a new arbitrary point is chosen from the data set to see if a new cluster can be formed. The process described in the previous paragraph is repeated for this new point and the subsequent ones that are grouped with it. It is possible that a point that was in a previous cluster can be assigned to a new cluster. That is not a problem. The formation of new clusters occurs until all the data has been classified. Figure 4.1 shows the results after DBSCAN has been performed on a random set of data, where three clusters were created. Data considered to be noise are shown as triangles (black), surrounding each cluster.

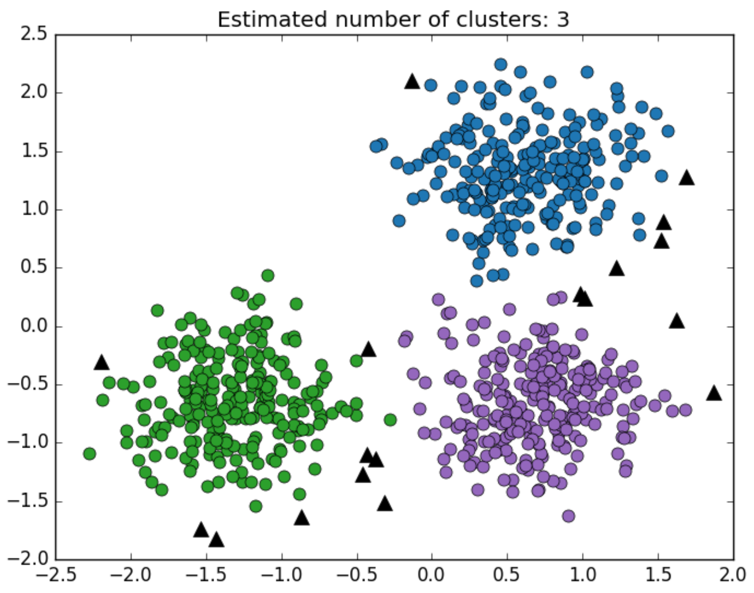


Figure 4.1: A visual representation of clusters formed using the DBSCAN algorithm of [19] and the data considered as noise, shown by black triangles.

For our application, we used DBSCAN as a way to determine the likely number of clusters that could be formed, not really for the cluster assignments it creates. Instead, K-means was used for classifying data, and is described in the next section.

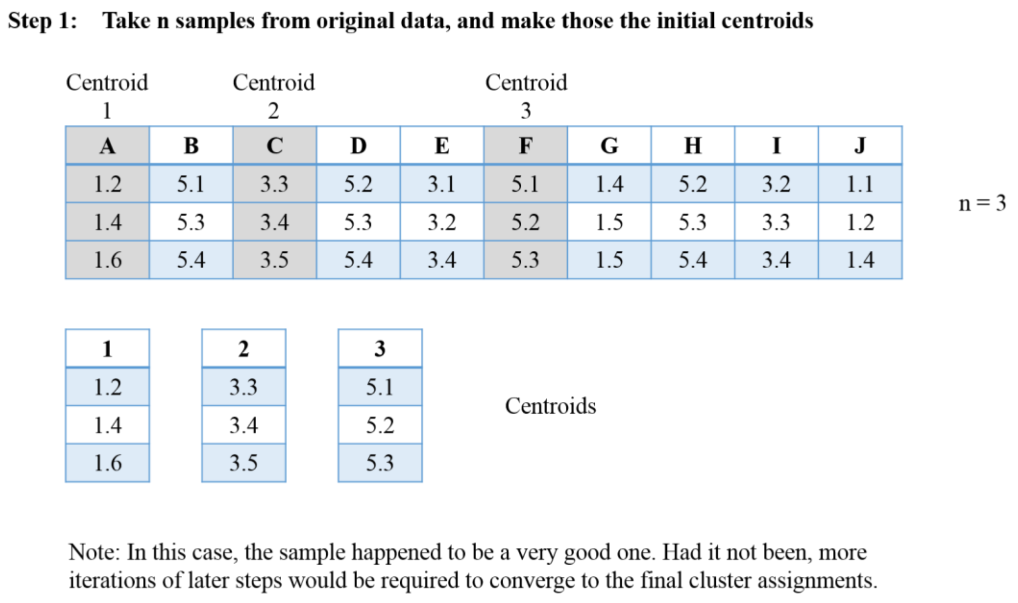
4.2 K-MEANS CLUSTERING

K-means clustering, also known as Lloyd's algorithm, is an iterative technique used to classify groups of similar data. It is a widely used clustering technique that works well for large amounts of data. One challenge with K-means is that the number of clusters, or groups, must be specified before running the algorithm. The documentation of [20] addresses that problem, and we provide a solution in the next section. The K-means algorithm is summarized in the remaining paragraphs of this section, and is fully documented in [20]. To explain the algorithm, the data given in Table 4.1 will be used, as well as the standard terminology of statistics and data science, where each row is known as an observation and each column as a variable.

Table 4.1: Example data that will be used to demonstrate the K-means algorithm.

A	B	C	D	E	F	G	H	I	J
1.2	5.1	3.3	5.2	3.1	5.1	1.4	5.2	3.2	1.1
1.4	5.3	3.4	5.3	3.2	5.2	1.5	5.3	3.3	1.2
1.6	5.4	3.5	5.4	3.4	5.3	1.5	5.4	3.4	1.4

First, *n* samples are chosen from the original data, where *n* is the number of clusters that should be formed. One sample is the same as one column (variable). These *n* samples are used as the initial centroids, around which clusters will be formed. The sampling process is shown in Figure 4.2, where the grayed columns have been selected as the samples to be the initial centroids. It should be noted that the choice of initial samples can affect the number of iterations and convergence of the algorithm, as is documented more fully in [20].



Step 1: Take n samples from original data, and make those the initial centroids

Centroid 1		Centroid 2			Centroid 3				
A	B	C	D	E	F	G	H	I	J
1.2	5.1	3.3	5.2	3.1	5.1	1.4	5.2	3.2	1.1
1.4	5.3	3.4	5.3	3.2	5.2	1.5	5.3	3.3	1.2
1.6	5.4	3.5	5.4	3.4	5.3	1.5	5.4	3.4	1.4

n = 3

1	2	3
1.2	3.3	5.1
1.4	3.4	5.2
1.6	3.5	5.3

Centroids

Note: In this case, the sample happened to be a very good one. Had it not been, more iterations of later steps would be required to converge to the final cluster assignments.

Figure 4.2: Example of the sampling process for generating the first set of K-means centroids.

Second, each column (variable) of the original data set is assigned to the nearest centroid. Nearness is determined by calculating the Euclidean distance between each column and centroid. The smaller the Euclidean distance, the nearer the column of data is to a centroid, and thus will ensure its assignment to that centroid. The Euclidean distance formula is given by:

$$d = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}\ ,$$

where d is the Euclidean distance, p_i is the value in row i, of a column of data, q_i is the value in row i, of the centroid column of data, i is the iterator specifying the row number, and n is the total number of rows of the data. In other words, each column of data is considered to be the vector, **p** = (p_1, p_2, p_3…, p_n), with the subscript identifying the row number of each data point (p) in the column. The same vector definition holds for the centroid, where the column of centroid data is the vector **q** = (q_1, q_2, q_3…, q_n).

To show this process, let us look at the data of column G. To determine which cluster column G belongs in, we need to determine the Euclidean distance between itself and each centroid. For centroid 1, the vectors are defined as

$$\mathbf{p} = (1.4, 1.5, 1.5) \qquad \mathbf{q} = (1.2, 1.4, 1.6)\ ,$$

and the Euclidean distance is calculated:

$$d = \sqrt{(1.4 - 1.2)^2 + (1.5 - 1.4)^2 + (1.5 - 1.6)^2} = \sqrt{0.04 + 0.01 + 0.01} = 0.245.$$

This process is repeated for centroid 2

$$\mathbf{p} = (1.4, 1.5, 1.5) \qquad \mathbf{q} = (3.3, 3.4, 3.5)$$

$$d = \sqrt{(1.4 - 3.3)^2 + (1.5 - 3.4)^2 + (1.5 - 3.5)^2} = \sqrt{3.61 + 3.61 + 4} = 3.35$$

and for centroid 3

$$\mathbf{p} = (1.4, 1.5, 1.5) \qquad \mathbf{q} = (5.1, 5.2, 5.3)$$

$$d = \sqrt{(1.4 - 5.1)^2 + (1.5 - 5.2)^2 + (1.5 - 5.3)^2} = \sqrt{13.69 + 13.69 + 14.44} = 6.47.$$

The Euclidean distance between column G and centroid 1 is the smallest, so column G is assigned to cluster 1. This process is carried out for each remaining column of data, until all columns have been assigned to the nearest centroid, as shown in Figure 4.3, where the gray shaded column is the cluster centroid.

1	A	G	J
1.2	1.2	1.4	1.1
1.4	1.4	1.5	1.2
1.6	1.6	1.5	1.4

2	C	E	I
3.3	3.3	3.1	3.2
3.4	3.4	3.2	3.3
3.5	3.5	3.4	3.4

3	B	D	F	H
5.1	5.1	5.2	5.1	5.2
5.2	5.3	5.3	5.2	5.3
5.3	5.4	5.4	5.3	5.4

Figure 4.3: Cluster assignments based on each column's proximity to the centroid shown in gray.

Third, new centroids are calculated by taking the mean of all samples, per row, in each newly formed cluster. For example, for the first cluster, the new centroid is formed by taking the mean of rows 1, 2, and 3 separately, as in:

Row 1: $\frac{1.2+1.4+1.1}{3} = 1.23$

Row 2: $\frac{1.4+1.5+1.2}{3} = 1.37$

Row 3: $\frac{1.6+1.5+1.4}{3} = 1.6$

The new centroids are shown in Figure 4.4.

1
1.23
1.37
1.5

2
3.2
3.3
3.43

3
5.15
5.28
5.38

Figure 4.4: New centroids formed for the clusters in Figure 4.4.

Fourth, the difference in the previous and new centroid is calculated, for each row. Figure 4.5 shows the old and new values for each centroid and the resulting difference between the two.

Old

1
1.2
1.4
1.6

New

1
1.23
1.37
1.5

Old

2
3.3
3.4
3.5

New

2
3.2
3.3
3.43

Old

3
5.1
5.2
5.3

New

3
5.15
5.28
5.38

Δ1
0.03
0.03
0.1

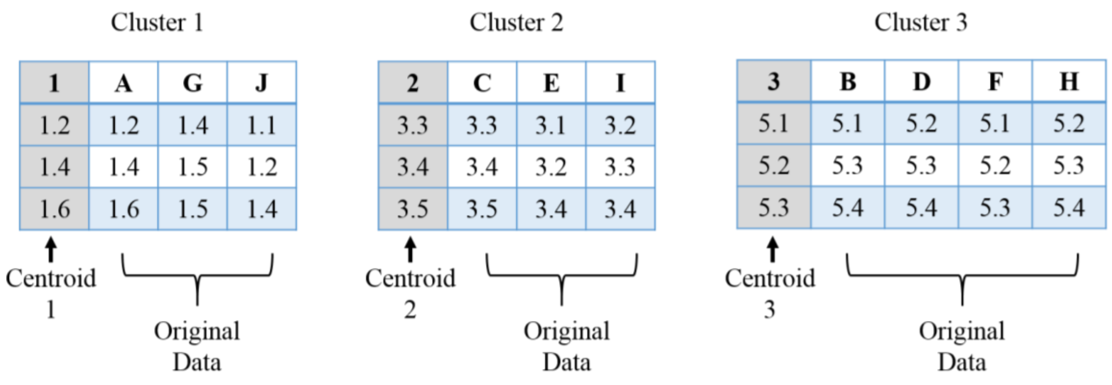
Δ2
0.1
0.1
0.07

Δ3
0.05
0.08
0.08

Difference in centroids

Figure 4.5: The difference between values of the original centroids and the new centroids.

Finally, the difference between the old and new centroids is compared to a pre-specified tolerance value. If the difference exceeds the tolerance, steps 2, 3, and 4 are repeated until the centroid difference is within the specification, or until a maximum number of iterations is reached. If the specification is met, the clustering process is complete, and the final clustered data is ready, as shown in Figure 4.6.



Cluster 1

1	A	G	J
1.2	1.2	1.4	1.1
1.4	1.4	1.5	1.2
1.6	1.6	1.5	1.4

Cluster 2

2	C	E	I
3.3	3.3	3.1	3.2
3.4	3.4	3.2	3.3
3.5	3.5	3.4	3.4

Cluster 3

3	B	D	F	H
5.1	5.1	5.2	5.1	5.2
5.2	5.3	5.3	5.2	5.3
5.3	5.4	5.4	5.3	5.4

Figure 4.6: The resulting data, after the entire K-means process is complete.

The average complexity required to perform K-means is given by O(k*n*l*T), where k is the number of clusters being formed, n is the number of variables (columns, buses), l is the number of observations per variable (measurements at each time step), and T is the number of iterations the algorithm takes to converge [20].

4.3 COMBINED CLUSTERING APPROACH

In order to consolidate data to provide a summary of the state of the electric grid, for each simulation, a two-staged clustering approach was used. Simulation data that was clustered included frequency, voltage angle, and voltage magnitude. Due to time constraints, clustering was not performed on current magnitude data.

Originally, various clustering techniques were tested to see which might be the most effective in identifying problematic buses and which provided the best overview of the system. It was found that DBSCAN was useful for identifying the number of clusters that could be formed and K-means did well classifying similar buses and highlighting the problematic ones. As such, a combination of the two techniques was used to leverage their respective strengths.

The clustering process was performed on a simulation by simulation basis, where the user specified the scenario they wanted to investigate. Once a scenario was selected, the data was cleaned and formatted as was described in Chapter 3, and the clustering process began. During clustering, DBSCAN and K-means were performed on each frequency, voltage magnitude, and voltage angle data set, individually. The DBSCAN and K-means algorithm were used from a machine learning Python package, called scikit-learn. For that package, the original data had to be transposed, such that each row held the measurement data for each bus, and each column specified the time each measurement was recorded. The entire process is described in more detail next, but is summarized in Figure 4.7, where the original transposed data is sent through the two clustering algorithms, and is output as data that has been sorted into clusters.

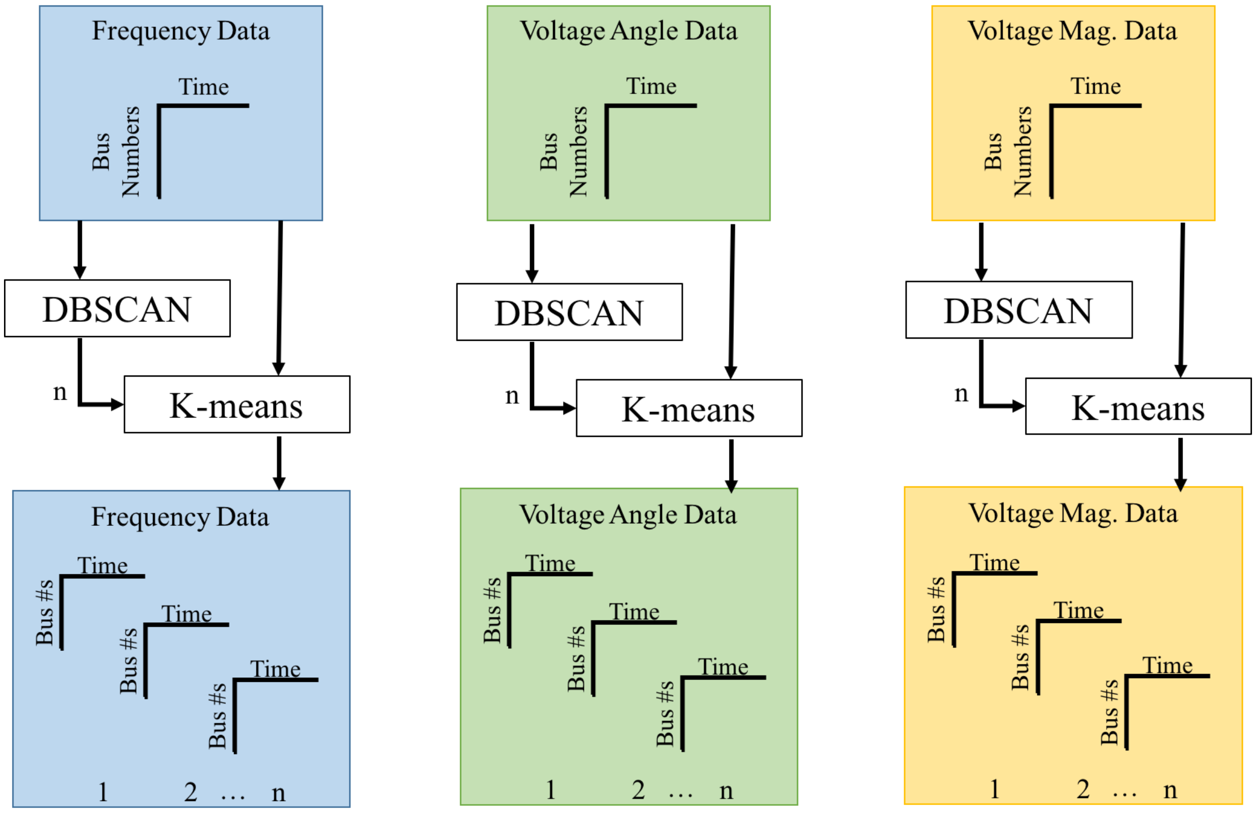


Figure 4.7: Overview of the two-tiered clustering approach, with input data to the clustering process shown at the top and the resulting data shown at the bottom, by cluster.

First, the transposed frequency, voltage magnitude, and voltage angle data were passed into the DBSCAN algorithm. The algorithm followed what was described in Section 4.1, where the term “data point” corresponded to each set of bus measurements. In other words, each “data point”, as coined in Section 4.1, was actually a full row of time stamped measurements for a bus. Then, just as previously described, the DBSCAN algorithm began by picking an arbitrary bus’s measurements, then looked for *min_points* number of other bus measurements within a distance of *epsilon*. The whole process outlined in Section 4.1 then continued, until all data was sorted into a cluster or identified as noise.

The values of *epsilon* and *min_points* were set for each measurement type, after experimenting with what gave the best cluster results, as is detailed in Table 4.2. In the future, an automatic and more sophisticated tuning approach should be used to find *epsilon* and *min_points*.

Table 4.2: Summary of DBSCAN input parameters for each measurement type.

	Frequency	**Voltage Angle**	**Voltage Magnitude**
epsilon	0.02	2	0.05
min_points	10	5	10

Next, K-means clustering was performed. The number of clusters formed in the DBSCAN algorithm, *n*, was used as an input to the K-means algorithm, along with the transposed frequency, voltage magnitude, and voltage angle data. As before, the K-means algorithm was used from the machine-learning Python package, scikit-learn, and followed the process described in Section 4.2. In our project, each column of data, described in Section 4.2, corresponds to measurements for each bus. However, due to the transposition required for the scikit-learn package, the example of 4.2 must also be transposed for direct comparison. Thus, the vectors **p** and **q** are formed from sampling rows of the transposed frequency, voltage magnitude, or voltage angle data, rather than columns as is described in Section 4.2. Nonetheless, the K-means algorithm functioned as expected, and sorted the original data into *n* clusters, with *n* centroids. The *n* centroids concisely summarize the state of the system.

The next chapter talks about how to visualize the data once it has been clustered. The idea behind visualization is to leverage the adage that a picture is worth a thousand words, by translating lots of information (a thousand words) into concise, summative, visual stories (pictures) that operators can easily draw insights from and then act upon.

4.4 SUMMARY AND OUTLIER DETECTION

In addition to clustering, the tool also performs a summary of the data to identify any buses that appear to be misbehaving more than others. To do so, the difference in measurements between time steps is calculated, and then the maximum change is saved for each bus. A histogram is created, with the x-axis designating the value of the change between time steps, and the y-axis the number of buses observing that difference. As an example, Figure 4.8 shows a visual of the histogramming process, for frequency, voltage angle, and voltage magnitude measurements. Colors correspond to the time window in which the maximum difference in time steps was observed.

Once the histogram is made, outlier buses are classified as the buses that fall into the last full histogram bin for frequency data, the last two histogram bins for voltage magnitude data, and the last three bins for voltage angle. As of now, the number of bins to use for identifying outliers is somewhat arbitrary, so in the future they would need to be more rigorously defined. However, for the simulations we created, the aforementioned bounds work well.

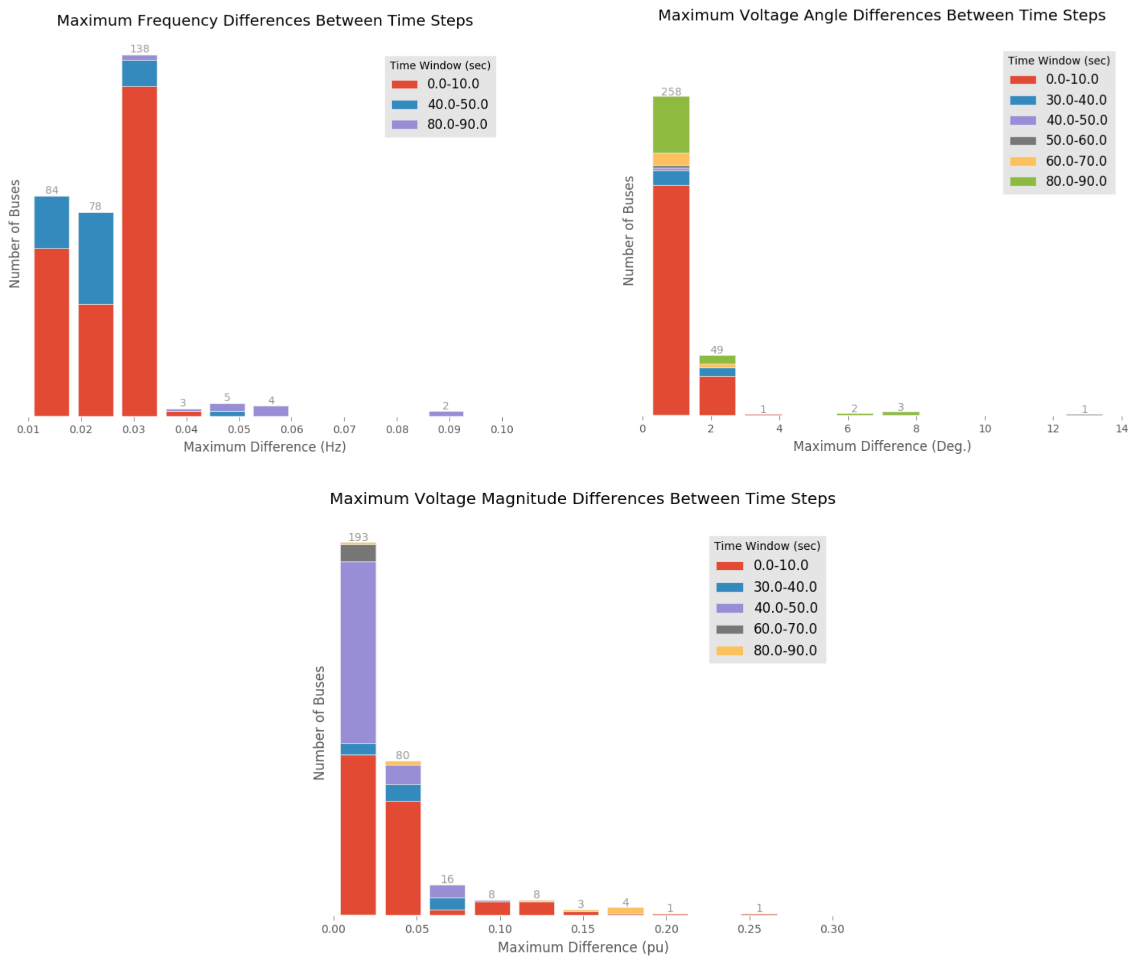


Figure 4.8: Histograms of the greatest difference observed between time steps for each bus. Colors correspond to the time window in which the largest time difference was observed.

CHAPTER 5
DATA VISUALIZATIONS

In order to discover insights from the PMU data, and thus enable more accurate and faster operator decision-making, visualizations were created. These visualizations sought to capture the physical footprint of the test system as well as the temporal component of the measurement data. Physical geography of the system was known from the location of substations where PMUs are actually or fictitiously installed, and temporal data from time-stamped frequency, voltage angle, and voltage magnitude measurements. The approach for creating spatial-temporal visualizations is discussed in this section, as well as the current state of power system visualizations.

5.1 POWER SYSTEM VISUALIZATION

With the advent of more data, power systems face the same challenges many industries do in making sense of all the data they have. One way to do this that has been used in power systems, as well as other industries, is to visualize it. Visualization makes it easier to understand what is happening in data by revealing patterns via colors, shapes, sizing or other graphics. In doing so, visualizations increase how well the data can be understood and how quickly. Thus, it is no surprise that industries, including the power and energy industry, are tapping into the power of visualizations to understand their data and consequently drive more effective decision-making.

The type of visualization to use depends on the application it is needed for, and this is true in power systems as well. A power system planner or an operator working on post-event analysis may prefer and require more detailed visualizations, because they have the time to carefully look through them. On the other hand, an operator who is dealing with real-time operation of the grid would want bite-sized pieces of information that they can quickly understand and act upon. They want to know as much as they can about their system (wide-area situational awareness) with as little effort as possible. Different visualizations are appropriate for each of those cases, though our research focuses on the latter one, which deals with events on the second and minute time scales.

To date, common visualization techniques used in power systems are pie charts showing how much electrical equipment is loaded, digital numeric displays providing exact measurement values, contours showing how frequency, voltage, or phase angles vary across a geographic area, and flow arrows specifying the direction of power or the gradient of phase angles [21]. However, with the new wealth of PMU data, utilities are seeking new techniques, to take full advantage of the information PMUs can provide.

Current PMU data displays include one-line diagrams, and maps with special icons and graphics to convey information about oscillations, voltage angles, megawatt flows, and frequency, throughout their system. These displays are relatively new, and so there is still room for improvement. In fact, in the cognitive work analysis conducted by the Rochester Institute of Technology team, several opportunities for improvement were identified. Namely, operators expressed a need to be able to simply identify where an oscillation is occurring in their system, how best to represent time data, and how to consolidate displays for oscillation and islanding detection, mode meters, angle alarms, MW flow, and frequency disturbance monitoring. In addition to the observations made by the Rochester Institute team, a few more shortcomings of power system visualizations include the use of too many colors, flashing graphics, too many numbers, too many mouse clicks between displays, and not providing enough summative information. All of these issues can overwhelm operators, distract, and ultimately, hinder them from making the decisions they need to make.

In creating our new visualizations, we sought to address as many of the identified challenges as possible, but we took an approach more focused on novelty and outside the box thinking in hopes of sparking ideas for future innovative visualization solutions. Our creativity was bounded only by what could realistically be implemented in a control room, mitigating as many of the issues previously addressed as possible, and following human factors' best practices. In large part, the emphasis of UIUC's work was to provide inspiration for new visualization directions, while the Rochester Institute of Technology team's visualizations were more researched and tested for their operational effectiveness. Both are important contributions.

It should be noted and re-emphasized that for normal operation of the grid, the current visualizations being used by power system operators are sufficient. Though, as in all things, there is always room for improvement. Our research aimed to make those improvements, and offer solutions for abnormal periods of operation, specifically regarding the information PMUs can provide in those instances.

5.2 GEOGRAPHIC (SPATIAL) VISUALIZATIONS

In order to make the visualizations as helpful and understandable as possible, we catered to operators' spatial familiarity of their system by providing visualizations that incorporated the system's physical layout. To do this, two visualization types were created, including an animation loop of frequency, voltage, and angle contours, and a geographic plot that classified substations according to the cluster they were assigned for each measurement type.

In general, contour mapping is used for spatial visualizations, though several challenges arise for this type of visualization in power systems. First, power system data is not spatially continuous, so some values have to be assumed to correctly form a contour line or area. Second, power system data is heavily time dependent, but contours can only be created for a single static time point. Lastly, voltage contour maps have to use a per-unit system or create maps for different voltage levels, separately. Despite these challenges, one key benefit to using contour maps is that they can clearly and concisely show data for a large number of buses. Because of that important benefit, we sought to build upon the contouring visualization idea by creating animated contour maps.

The animated contour maps were created by saving snapshots of contours for frequency, voltage angle, and voltage magnitude, in each simulation, at successive time steps. The images were then compiled into a video to create a looping animation that mimics Doppler weather radar displays. The thought behind this type of visualization is that it takes advantage of a contour's ability to concisely represent a large amount of spatially distributed information and allows for the time component to be captured via time stepped changing contours. In an operational setting, the animation loop would continue to be refreshed with new data, though data before a certain interval, say 60 minutes, would be discarded. An operator can pause or rewind the loop, if they see something developing in their system. In [22], I claimed that an animation loop may not have realizable real-time operational benefits, given the time required to watch the animation loop. However, if there is an operator who actively monitors the animation loops and/or if an event develops slowly enough to be detected in the animation loop (i.e. the event develops in a timeframe of minutes or hours, rather than seconds or sub-seconds), animation loops could be a helpful option for real-time operation of the grid. Sample animation loops can be viewed at [23].

The other approach we used to represent the geographical component of PMU data was to show the location of buses (spatial data) with markers on a map, and then color them according to which cluster they fell into, derived from measurement data (temporal data). Data points were shaped according to the nominal voltage level of each bus. The idea behind this was to visually show regions of the grid that behaved in a similar way. Figure 5.1 shows a plot of buses being monitored by PMUs, whose cluster and color assignments were determined from frequency simulation data that was sent through the clustering process described in Chapter 4. By using individual data points, rather than a contour, the requirement for spatial continuity was circumvented. Additionally, because individual buses were plotted, an operator may be able to more quickly identify the exact set of problematic buses, rather than just a general area of concern.

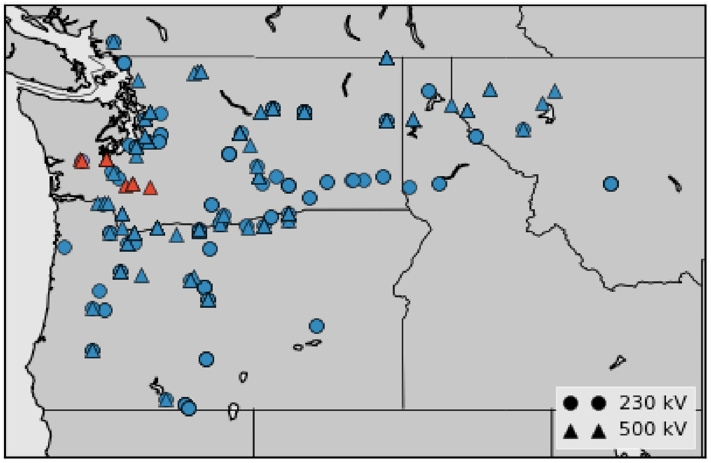


Figure 5.1: Geographic plot of bus locations. Marker colors specify the cluster to which each bus belongs, and the shape the voltage level each bus operates at.

In the visualization of Figure 5.1, and all visualizations we created, extraneous lines, colors, and data were excluded to focus the attention of the operator on the most important information. As you will notice in Figure 5.1, gray background colors were used, instead of colors like blue or green that might more easily denote water and land areas. However, those features are not what the operator should focus on, and the distinction between the two would already be well known by someone familiar with the area. Thus, gray was used for background colors, to ensure that more colors were available to represent and emphasize the most important data.

5.3 TEMPORAL VISUALIZATIONS

Temporal data is often expressed with line plots, because they easily display how data changes for each time step. In power systems, that tendency holds true as well. However, using line plots for tens or hundreds of buses quickly makes the plotted data cluttered and unhelpful. Because of this challenge, two techniques were used to limit the amount of data presented in line plots. First, clustering was used to reduce the data down to representative cluster centroids, which are mean values of the data in a cluster. Second, a drill-down technique was used, such that data was only shown for what is selected by the user. The latter technique will be described in Section 5.4.

In Figure 5.2, line plots are shown for each bus being monitored by PMUs (left) and the cluster centroids (right), for each measurement type. As you can see, when all PMU data is plotted, the result is a cluttered display of overlapping lines that do not provide very useful information. On the other hand, when just the cluster centroids are plotted (right side of Figure 5.2), they are able to capture the envelope response of the system for each measurement type, and in so doing provide a concise summary of the state of the system, using only a few lines, rather than hundreds. Unfortunately, using the clustered data does not provide detailed information about a specific bus or substation, but it does offer a summary that would allow an operator to narrow down the area they should look at next in their investigation.

Figure 5.3 provides a visual of a more granular view of the clustered measurement data, by plotting the original measurement data, within each cluster, alongside the cluster's centroid data points. The data shown in Figure 5.3 is for voltage magnitude clusters, though the same sort of plots could be formed for frequency or voltage angle. As before, you can see that the cluster centroid, for the most part, captures the general response of the system. The cluster centroid shows where events occurred via the sudden spikes or divots, and in so doing provides focus for further investigation by an operator.

As was mentioned in Chapter 4, misbehaving outlier buses can be identified for the data as a whole or for each individual cluster. This extra information can supplement the cluster centroid data to provide both a big picture understanding of the system, with the cluster centroids, and a more granular view of the system, with identification of outlier buses. The outlier buses are the ones that most likely need to be dealt with to mitigate power system problems. Figure 5.4 provides a visualization of outlier buses for cluster 4, shown in Figure 5.3. A display like Figure 5.4 could be used in combination with other temporal displays.

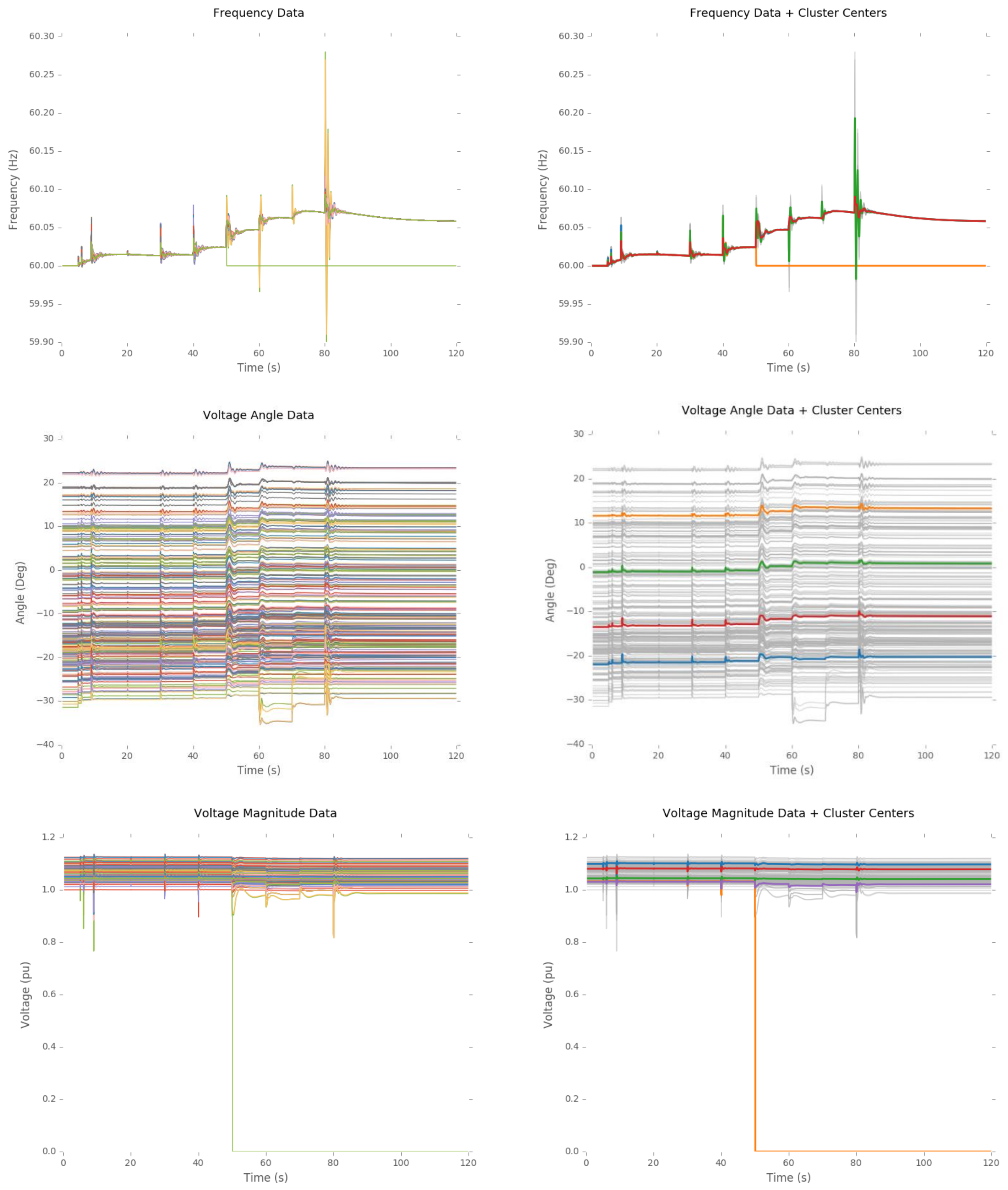


Figure 5.2: Temporal line plots of PMU data, for all buses being monitored (left) and summative cluster centroid line plots (right) for each measurement type.

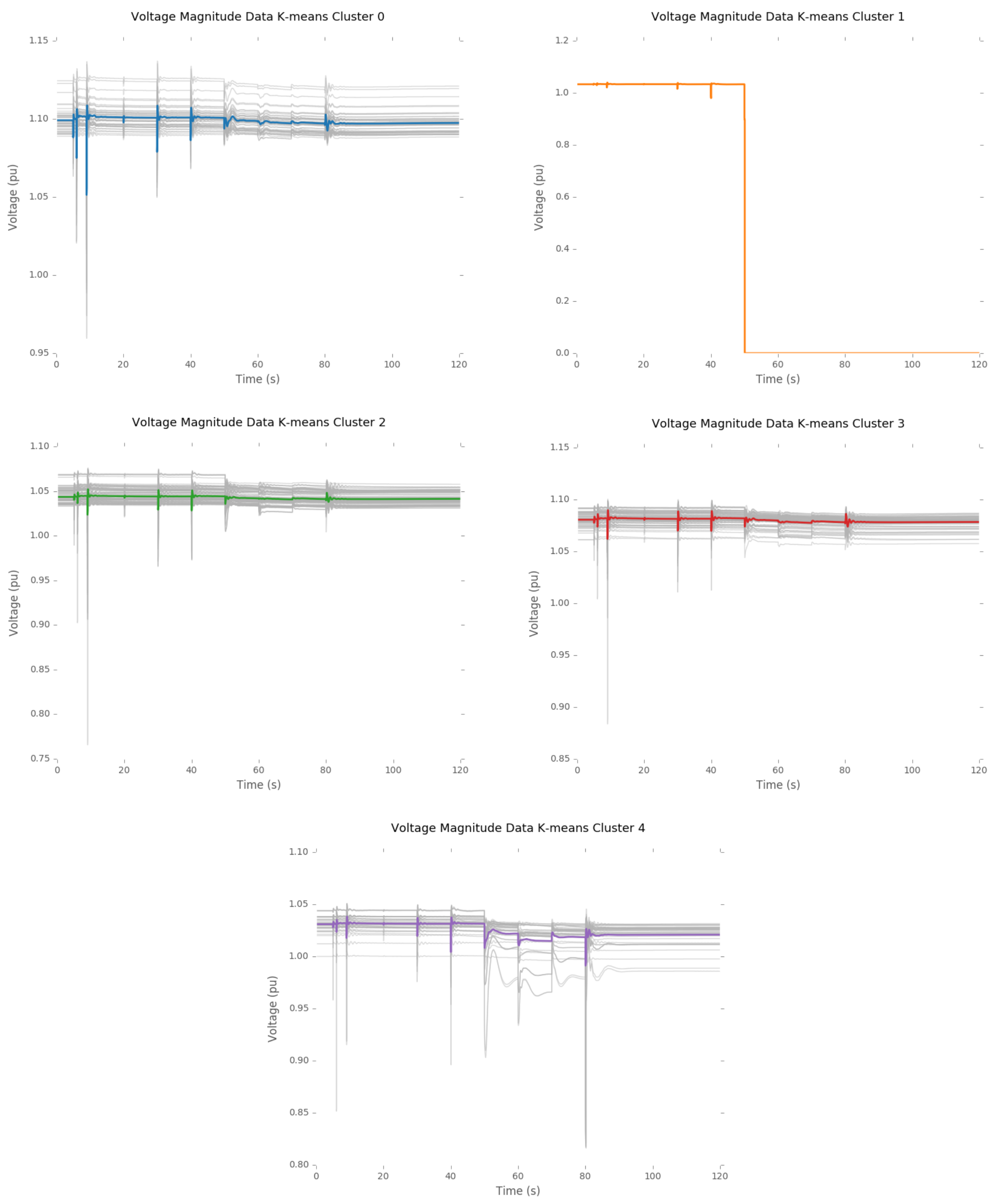


Figure 5.3: Temporal line plots of PMU data, for all buses being monitored (left) and summative cluster centroid line plots (right) for each measurement type.

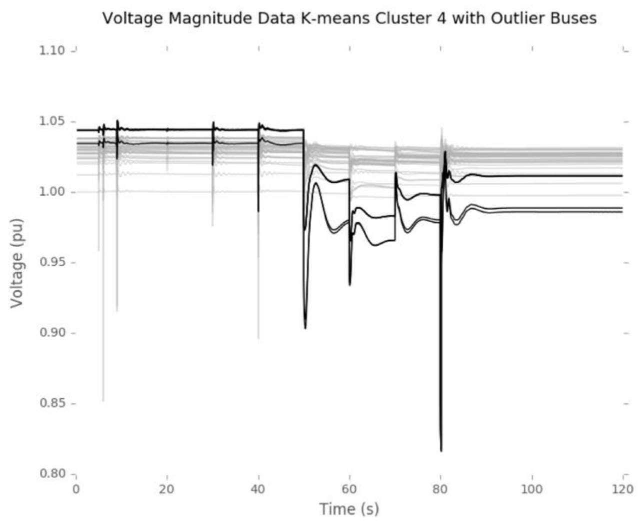


Figure 5.4: Line plots of data for a cluster of voltage magnitude PMU data, with outlier buses identified by the black dark lines. Substation names for the outlier buses can easily be provided in the plot, but were kept hidden for confidentiality.

In an operational setting, the plots in Figures 5.1-5.4, would not, by themselves, be helpful. Thus, the next section describes our ideas for how elements from the aforementioned spatial and temporal plots can be combined to provide improved wide area situational awareness and enough detailed information to help an operator pin point a problem and fix it.

5.4 USING VISUALIZATIONS TO PINPOINT PROBLEMS

Two interactive visualization panes were created to give operator's better situational awareness and the ability to more quickly identify problems. Each display uses visualizations that capture both the temporal and spatial elements of PMU data. The first interactive display is shown in Figure 5.5, and is made from data after it has been clustered using the process described in Chapter 4. The plots in the top row show the geographic location of each bus in the model. The markers are shaped according to their voltage level and colored according to the cluster they have been assigned. The second row of plots shows the cluster centroids, to summarize the overall response of the system, without plotting data from every bus. This ensures a concise, but complete, picture of a large electric grid. The last blank row of plots is reserved to show measurement data for buses that are selected from one of the geographic plots. For example, the user can draw a rectangle over a section of one of the plots in the first row, and then the data for the buses contained in the rectangle will be plotted in the last row. The user selection process is shown in Figure 5.6. Note that the colors in each *column* of plots are associated, though not necessarily the colors in plots of the same *row*.

There are several benefits for using this type of display. First, there are only as many colors used as there are clusters, which ensures the operator can focus on what the data is saying rather than how it is saying it through colors. Second, a concise and summative state of the system is provided by using cluster centroids, rather than plotting data for every bus in the network. However, if a more detailed view is desired, the display also allows for that, through user selection of buses on the geographic plots. This balances the need for a big picture perspective and more granular details, with only one mouse click. Finally, because this display incorporates the geographic locations of buses, outside information like weather radar could be overlaid on the plots to more quickly diagnose the cause of a problem, especially for weather related incidents.

The second interactive display is shown in Figure 5.7. The basis for the visualizations in this display is the underlying grouping of data based on the maximum differences between time steps for each bus, described in Section 4.4. Histograms are provided in the first row and show the maximum difference between adjacent time steps, for each bus. Sections of each bin are colored according to what time window the maximum difference is observed, for each bus. The second row of plots shows the measurement data for outlier buses contained in the last one or two full bins of the histogram above them. One bin is used for frequency and two bins for voltage angle and magnitude. The color of each line does not correspond to the histogram, but is uniquely colored to distinguish between various buses. The final row of plots shows the geographic locations of each outlier bus identified for each measurement type—frequency, voltage angle, and voltage magnitude. Note that the colors between plots are not necessarily associated.

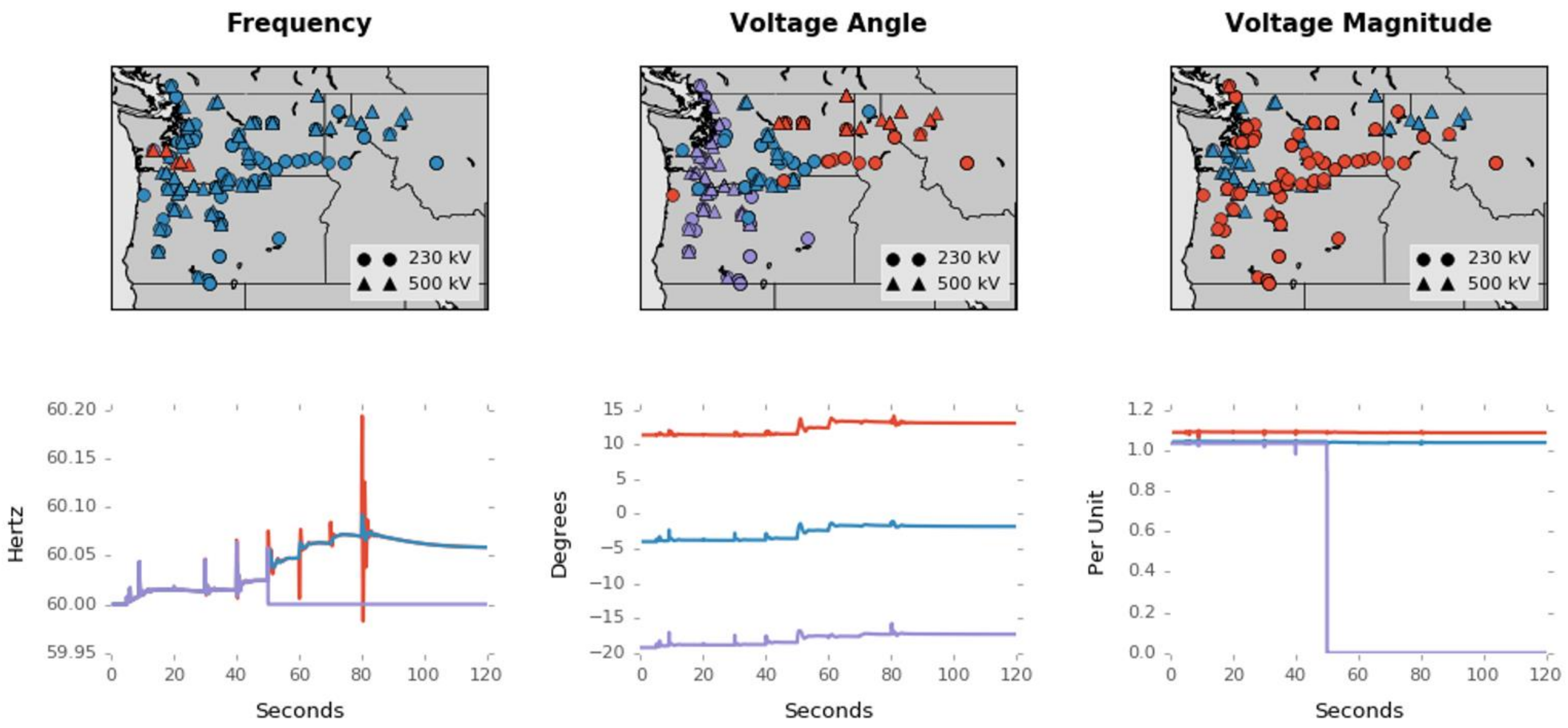
Just like the previous display, this display is also interactive. A user can select a bin in one of the histograms that they want to investigate further. By doing so, the middle row of line plots updates to reflect the data for the buses contained in the bin the user just selected, as does the geographic plot to show the correct buses that were selected. The new line plot also includes the corresponding bus number for the data, which allows an operator to quickly identify the problematic buses. Figure 5.8 shows the updated display, from Figure 5.7, after a user selected to view the very last bin of the frequency histogram. The two buses contained in that bin then have their data displayed in the second row of line plots and their location in the third row of geographic plots.

This type of display makes it easy to concisely summarize drastic changes in measurement values and to show in which 10 second window those changes are observed. The display allows an operator to quickly identify outlier buses and see the temporal nature of the data for those buses, as well as their geographic footprint. By doing so, third party information, like weather radar, can be incorporated into the display to help diagnose problems and then correct them. One problem is that there may be too many colors used in this display, which could distract or strain the operator from focusing on what is most important.

The difference in displays can be characterized by function, where the first display allows for interpretation and search based on geography, and the second allows for interpretation and search based on the most drastic changes between data points. As mentioned in Section 5.1, looping contour videos may provide a nice addition to each display.

The displays shown in Figures 5.5-5.8 are for the earthquake scenario, occurring on the western coast of Washington. In the first display, the blue colored markers of the frequency geography plot seem to indicate buses that are either the cause of a problem or are affected by one. In fact, the buses shown in blue are the buses that I manipulated in the simulation and/or those that are electrically close to the manipulated buses. In the second display, the largest deviations are observed in the region where the earthquake's damage was greatest. Similarly, the displays of Figures 5.9 and 5.10 correctly identify buses that were directly affected by an ice storm or the buses that are electrically close to those buses.

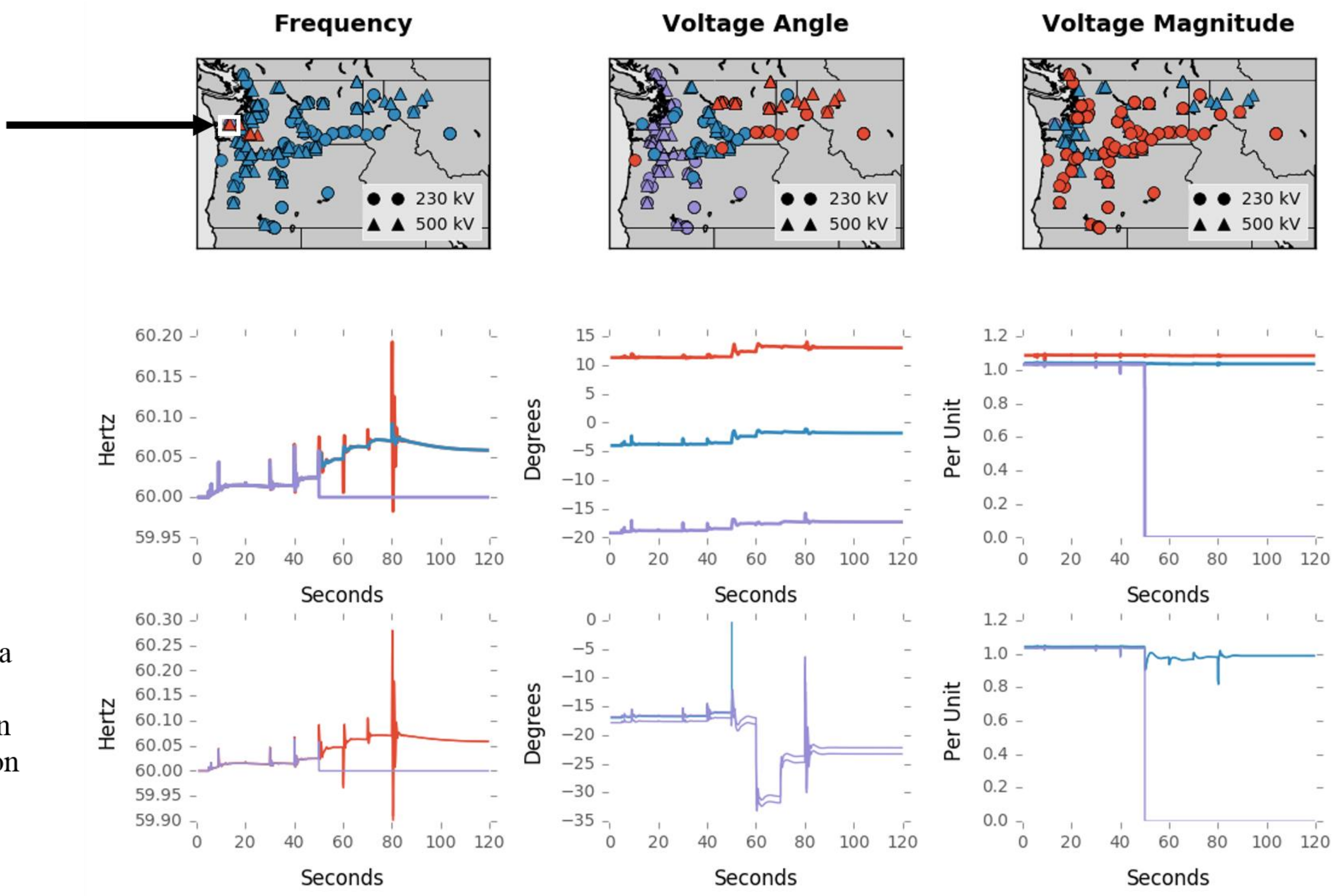
All in all, the displays discussed in this chapter are meant to inspire new ideas for visualizations, and are not necessarily meant to be final products. They may be beneficial in their own right for real-time operation of the grid, or could help in offline studies of events to help operators come up with new procedures, called remedial action schemes, that dictate what actions an operator should take when a certain event is observed. Indirectly, these visualizations would then help in future real-time operation of the electric grid. The next step of the research process is to determine the effectiveness of the displays via testing performed by the Rochester Institute of Technology team.



No plots appear in this row until the user makes a selection of buses they want to zoom in on, in one of the geographic maps.

Figure 5.5: Display of (top) geographic locations of buses and the clusters they belong to and (bottom) the cluster centroids, for the earthquake scenario.

User selection



Plots of data for buses contained in user selection box

Figure 5.6: Display of (top) geographic locations of buses and the clusters they belong to, (middle) the cluster centroids, and (bottom) the data for the buses selected by a user-drawn rectangle in the frequency geographic plot, for the earthquake scenario. Colors are associated in plots within each display column, not necessarily the row.

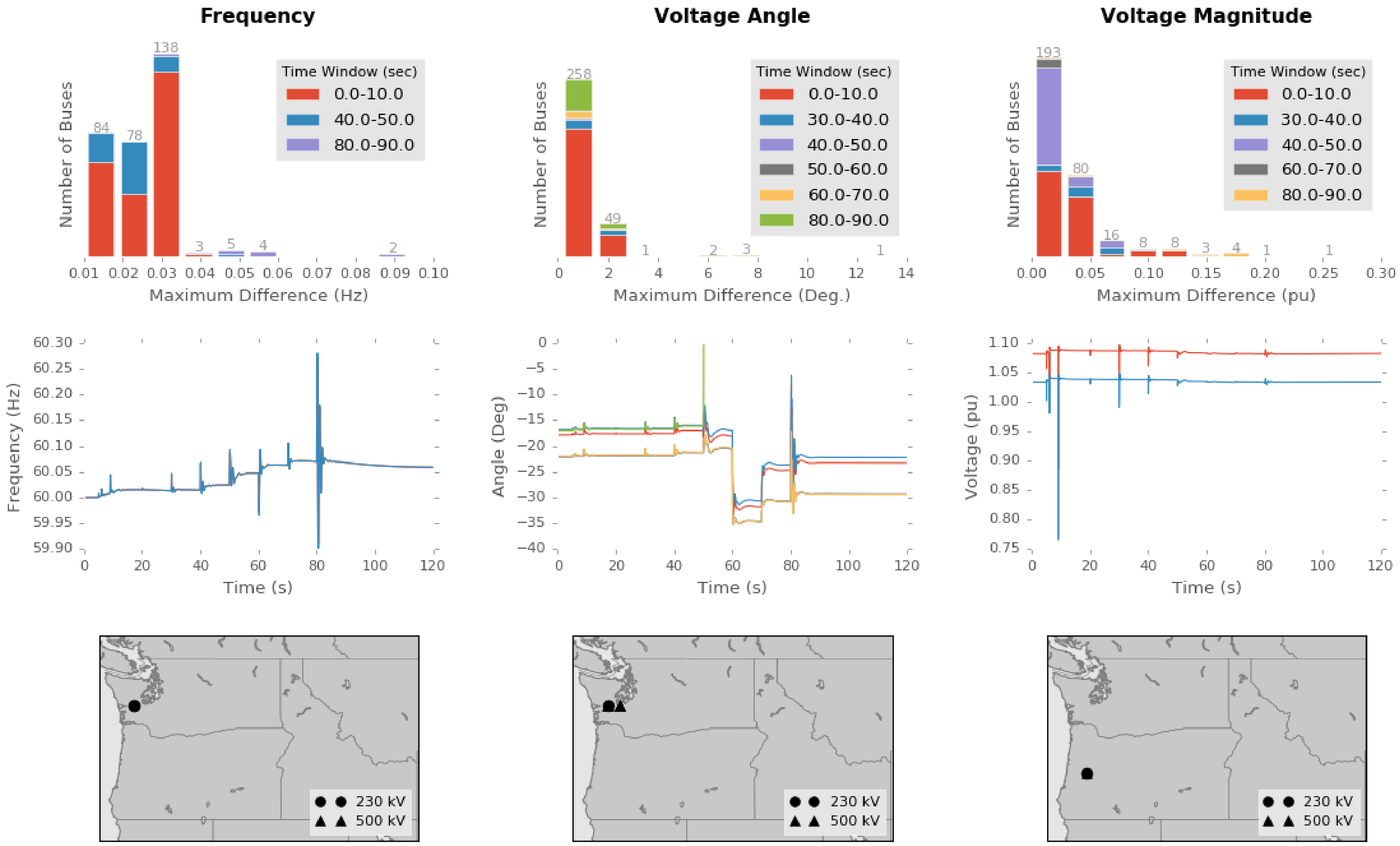


Figure 5.7: Display of (top) maximum changes between time steps observed for each measurement type and colored according to the time window when the maximum was observed; (middle) measurement data for each bus contained in the last bin, for frequency, and last two bins for voltage angle and voltage magnitude; and (bottom) geographic location of the buses identified as outlier buses in the last one or two bins, for the earthquake scenario.

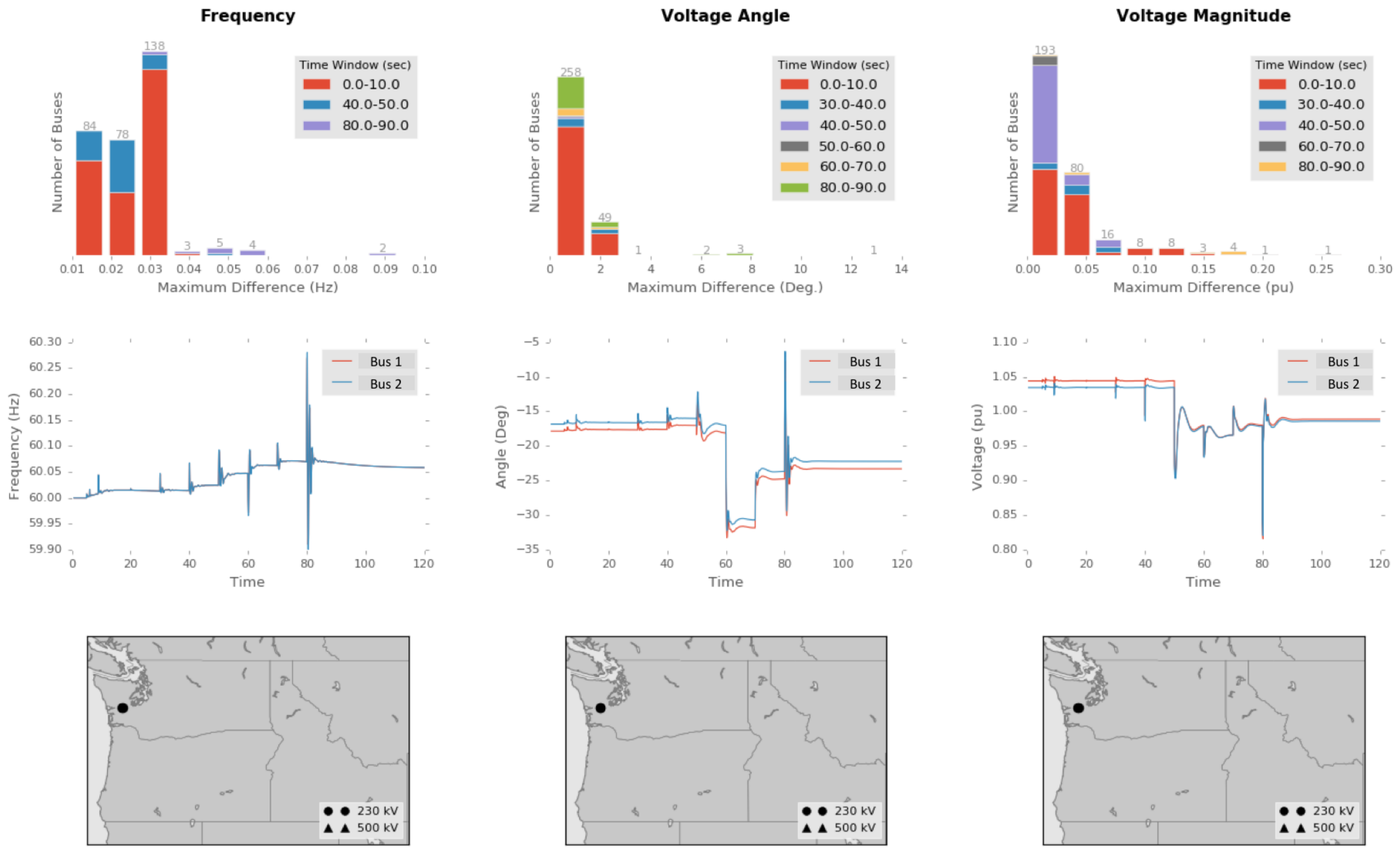


Figure 5.8: Display of (top) maximum changes between time steps observed for each measurement type and colored according to the time window when the maximum was observed; (middle) measurement data for the buses contained in the last bin, as selected by the user from the frequency diagram; and (bottom) geographic location of the two buses contained in the last bin of the frequency histogram, for the earthquake scenario. Note that bus numbers in the legends of the middle plots have been changed to maintain confidentiality.

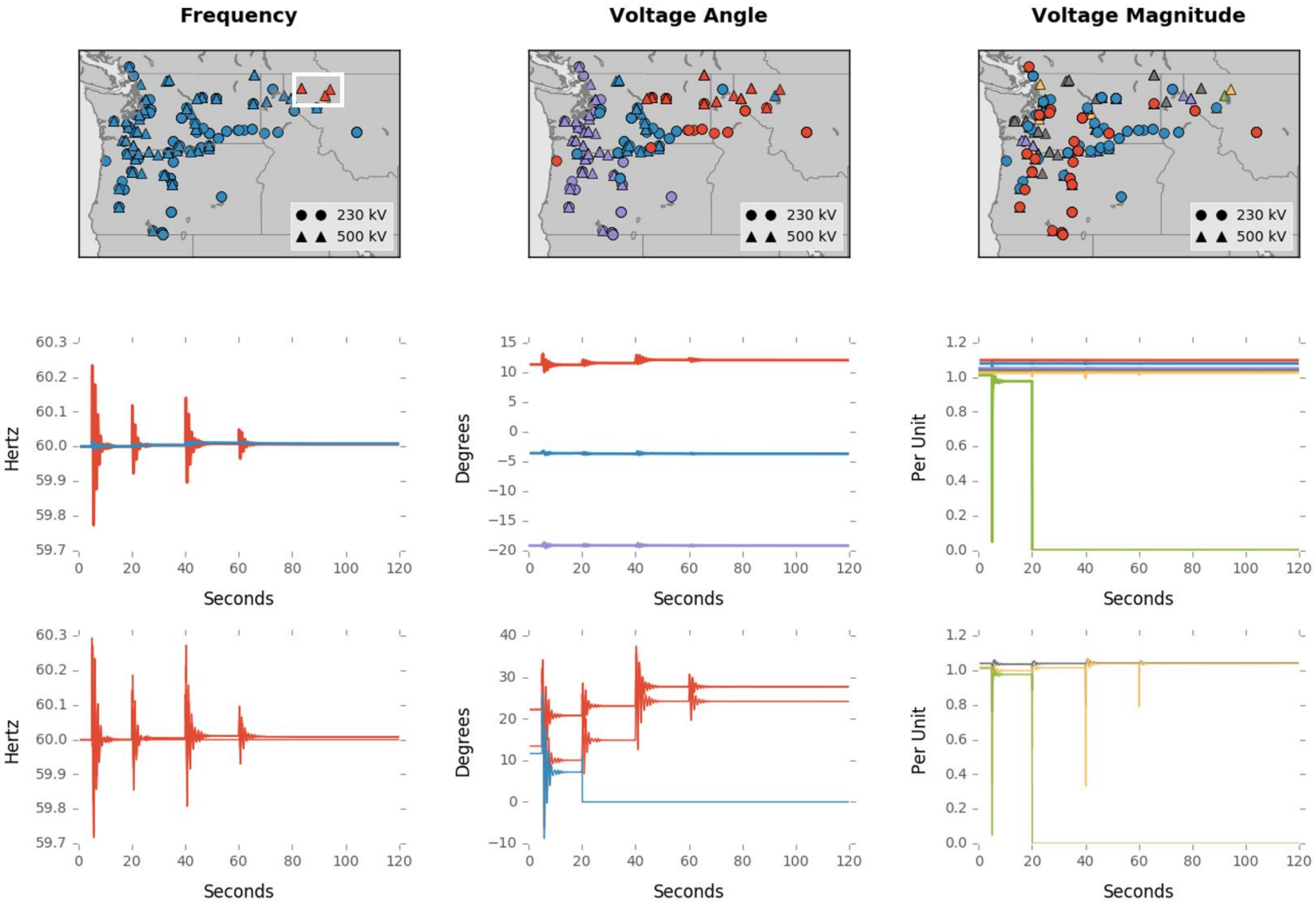


Figure 5.9: Display of (top) geographic locations of buses and the clusters they belong to, (middle) the cluster centroids, and (bottom) the data for the buses selected by a user drawn rectangle in the frequency geographic plot, for the ice storm scenario. Colors are associated in plots within each display column, not necessarily the row.

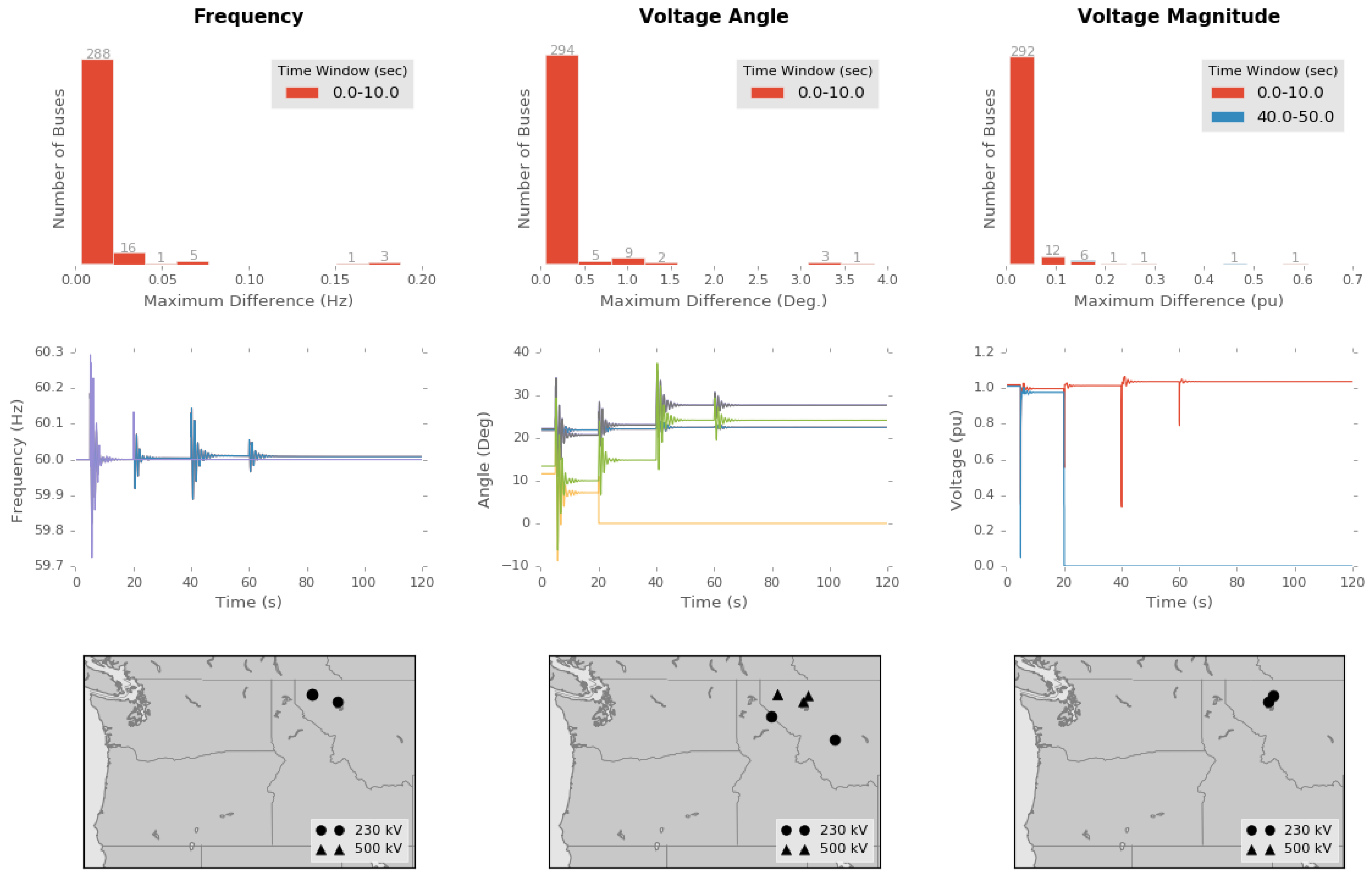


Figure 5.10: Display of (top) maximum changes between time steps observed for each measurement type and colored according to the time window when the maximum was observed; (middle) measurement data for each bus contained in the last bin, for frequency, and last two bins for voltage angle and voltage magnitude; and (bottom) geographic location of the buses identified as outlier buses in the last one or two bins, for the ice storm scenario.

CHAPTER 6
CONCLUSION AND FUTURE WORK

As in all research, the path toward the finish line in this project was rife with missteps and opportunities for learning. Of course, we had our share of successes, too. This chapter provides a summary of work done, lessons learned, and ideas for future work.

6.1 DISCUSSION AND SUMMARY

In the area of visualization, we sought to think outside the box, so spent many hours poring over visualizations of every shape and variety. We saw many beautiful visualizations that sparked ideas for our own research, but ultimately concluded, as others have, that visualizing spatio-temporal data is quite challenging, especially when the visualizations have to be interpreted quickly. Table 6.1 summarizes the strengths and weaknesses of our visualization solutions and research.

We noticed that previous research in power system visualization has largely been done in silos, where experts in power systems, statistics, human factors, or design have yet to come together to collaboratively tackle visualization problems. For our project, we changed that paradigm and brought together experts from both power systems and human factors. In doing so, we tapped into the power of collaboration and a diversity of skill sets. Though advantageous for creating impactful visualizations, it added a layer of complexity in trying to create data suitable for the human factors team to deal with and which was understandable. To ease that burden, we created a software tool to standardize and automate the process of transforming power system data into a well defined format for which visualization prototypes could quickly be developed. The idea was to create a bridge between power system data and data visualizations, so that visualization experts can focus on creating visualizations rather than waste time trying to create data and understand power system intricacies. As long as data input to our program is of the form given in Section A.4 of the appendix, and visualizations are created in Python, the tool should be able to get the user from file selection to visualization, within a few clicks and a few seconds.

Table 6.1: Summary of visualization techniques used and what their benefits and challenges are.

	Benefits	**Challenges**	**Our Solution**
Line Plots	Clearly show temporal characteristics of PMU data. Easy to understand quickly.	Plotting line plots for every bus makes cluttered and unhelpful displays. Spatial data is not organically included in line plots.	Use a summative line plot that represents many buses, like a cluster centroid. Then, associate each line with their corresponding buses marked on a geographic map, by color.
Looping Video	Allows for the plotting of spatial and temporal data simultaneously. Similar to weather radar loops, so would be intuitive to operator.	It requires time to create and watch the video.	Create a looping style video that can be sped up or slowed down. This may be a tool more suited for offline analysis.
Outlier Detection	Reduces area of focus to a handful of buses.	Some problematic buses could be missed by the outlier detection algorithm.	Create a histogram with observed maximums for all buses, so can double check that all outliers are accounted for, and so that the outliers are obvious. Show time windows when maximums were observed, via colors in the histogram.
Combine Plot Types in a Display	Leverage strengths of various spatial and temporal plot styles.	Takes up more display space.	Use line plots, histograms, and geographic maps together in a display. Associate the plots using color. If a display can provide information the operator needs, then it is worth using more display space.
Isolate Time Windows	By using a known time window, the dimension of data is reduced.	Looking at the wrong time window could be devastating.	In the histogram plots, partition data by the time window in which a measurement occurred. Use colors to distinguish the different time windows, so that the operator can go investigate that particular window further.

After exploring various clustering algorithms, we ultimately decided to use K-means and DBSCAN because of their speed and accuracy. Beyond that, DBSCAN was chosen for two reasons. First, because one of the outputs of the DBSCAN algorithm is the number of clusters that can be formed, we used that to initialize the number of clusters used in the K-means clustering step. Second, we thought the input parameters to DBSCAN, *epsilon* and *min_samples*, could be helpful for utilities to define based on what made sense for their system. As such, the clustering program could better cater to a utility, and thus make better summative representations of power system data. However, if the user does not want to specify either the number of clusters to use for K-means or the input parameters for DBSCAN, an alternate method can be used. This method runs the K-means algorithm for *n* number of clusters, where *n* is cycled from 2 to 10. A score representing how well the data fits in each cluster is created for each run, and then the program selects the number of clusters, *n*, which has the lowest score, or at least the first minimum score. Figure 6.1 shows a plot of scores for 10 cluster sizes. The first minimum is at *n* = 5, so this is the number of clusters that are used for K-means clustering, in that particular example. Though this approach eliminates the need for any input parameters, it is slower since the K-means algorithm has to be run at least two times in order to generate clustered data. In the software tool, a user can select whether they want to use DBSCAN or the K-means scoring method for initializing the K-means algorithm.

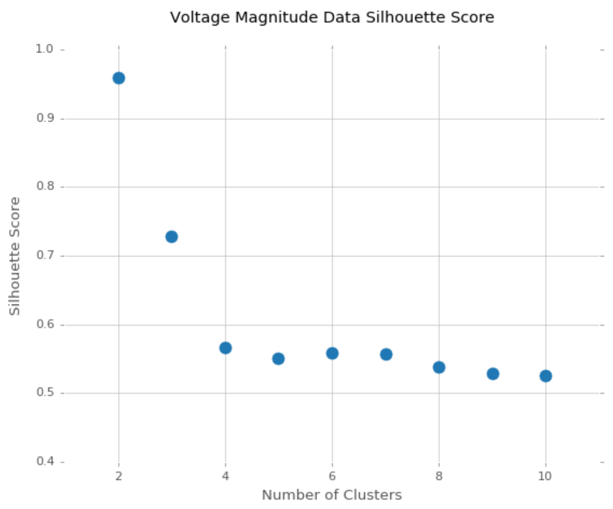


Figure 6.1: Plot of cluster scores, which assess how well data fits into clusters. The cluster scoring process is used to determine how many clusters should be used to initialize the K-means algorithm, when DBSCAN is not used.

Finally, by creating visualizations with geographic maps, non-power system data can easily be integrated to help operators diagnose problems in their systems. For example, adding weather radar to one of the geographic maps in Figures 5.5-5.10 could enable an operator to see if a storm is causing, or will cause, an event in their power system, so they can take the appropriate actions. Without PMU data and its geographic information, these sorts of displays would not be possible and the actions an operator should take less obvious.

6.2 IDEAS FOR THE FUTURE

Moving forward there are three primary areas of work that should be considered. The first is to improve our software tool to make it even more flexible and generalizable. This is something I will be working on after this thesis. Second, PMU data should be approached from the perspective of “big data”. That is to say that data scientists, statisticians, and visualization gurus working on “big data” could offer a great deal to the power systems area. The challenge will be for power systems experts to work collaboratively with these outside experts, while still maintaining confidentiality of grid data. Third, as power systems work to become more intelligent, cognitive computing should be explored. Cognitive computing expands upon the second future work direction, by using “big data” to allow computers to do much of the information processing, instead of operators. This allows operators to focus on the tasks that are most urgent and which they are best equipped to handle. Cognitive computing should not be used to replace operators, but instead to make operators more effective. Operators have a wealth of experience that algorithms cannot yet recreate.

PMU data will play a vital role in making the electric grid of North America smarter, more efficient, reliable, resilient, and flexible, but will require innovative software tools to transform that data into insights and action. There is much that must be done, but through collaboration and outside-the-box thinking, the new and improved 21st century grid is certainly within reach.

REFERENCES

[1] W. A. Wulf, “Great Achievements and Grand Challenges,” *The Bridge*, vol.30, no.3/4, p.6, Fall 2000.

[2] G. Bakke, *The Grid: The Fraying Wires between Americans and Our Energy Future*. New York, NY, United States: Bloomsbury Press, 2015, p. xiv.

[3] DOE Grid Tech Team, “Vision of the Future Grid,” in *U.S. Department of Energy*. [Online]. Available: https://energy.gov/under-secretary-science-and-energy/vision-future-grid.

[4] U.S. Department of Energy. *The Smart Grid: An Introduction*. U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability, 2008. [Online]. Available: https://energy.gov/oe/downloads/smart-grid-introduction-0

[5] Y. Aillerie, et al. *Smart Grid Cyber Security*. Intel Corporation, McAFee, ALSTOM, 2013.

[6] E. Dall'Anese, P. Mancarella and A. Monti, "Unlocking Flexibility: Integrated Optimization and Control of Multienergy Systems," in *IEEE Power and Energy Magazine*, vol. 15, no. 1, pp. 43-52, Jan.-Feb. 2017.

[7] R. H. Miller, J. H. Malinowski. *Power System Operation.* 3rd ed. Boston, MA: McGraw Hill, 1994, p.83.

[8] P. Sauer and M. Pai, *Power System Dynamics and Stability*, Champaign, IL: Stipes Publishing L.L.C., 1997, p. 4.

[9] U.S. Department of Energy. *Advancement of Synchrophasor Technology*. U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability, March 2016. [Online]. Available: https://www.smartgrid.gov/files/20160320_Synchrophasor_Report.pdf

[10] Bonneville Power Administration, *TIP 353: Improving Operator Situation Awareness by Phasor Measurement Unit (PMU) Data Visualization.* 2015. [Online]. Available: https://www.bpa.gov/Doing%20Business/TechnologyInnovation/TIPProjectBriefs/2016-TIP-353.pdf

[11] *Power System Equivalents Training*. PowerWorld, 2008. [Online]. Available: https://www.powerworld.com/files/TrainingI14Equivalents.pdf

[12] L. Beard et al. Guidelines for Siting Phasor Measurement Units. North American SynchroPhasor Initiative, 2011.

[13] "Synchrophasor Success Lands BPA Its First Platts Award". Bpa.gov, 2013. [Online] Available: https://www.bpa.gov/news/newsroom/Pages/Synchrophasor-success-lands-BPA-its-first-Platts-Award.aspx

[14] U.S. Department of Energy. *Environmental Clearance Memorandum.* U.S. Department of Energy, Bonneville Power Administration, 2011. [Online]. Available: https://www.bpa.gov/efw/Analysis/CategoricalExclusions/cxl/CX-ProgrammaticSynchrophasorGPSantennas_WEB.pdf

[15] https://www.bpa.gov/power/pl/columbia/graphics/regions2.jpg

[16] A. von Meier, "Occupational cultures as a challenge to technological innovation," in *IEEE Transactions on Engineering Management*, vol. 46, no. 1, pp. 101-114, Feb 1999.

[17] N. Harris, "Visualizing DBSCAN Clustering". Naftaliharris.com. 2015. [Online]. Available: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

[18] M. Ester, H. Kriegel, J. Sander, and X. Xu "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96) pp. 226-231 1996. [Online]. Available: http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[19] "Scikit-Learn 0.18.1 Documentation – 2.3.7 DBSCAN". Scikit-learn.org, 2016. [Online] Available: http://scikit-learn.org/stable/modules/clustering.html#dbscan

[20] "Scikit-Learn 0.18.1 Documentation – 2.3.2 K-means". Scikit-learn.org, 2016. [Online] Available: http://scikit-learn.org/stable/modules/clustering.html#k-means

[21] T. J. Overbye, "Visualization enhancements for power system situational assessment," *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, Pittsburgh, PA, 2008, pp. 1-4.

[22] K. M. Gegner, T. J. Overbye, K. S. Shetye and J. D. Weber, "Visualization of power system wide-area, time varying information," *2016 IEEE Power and Energy Conference at Illinois (PECI)*, Urbana, IL, 2016, pp. 1-4.

[23] http://www.powerworld.com/animated-transient-stability-contingencies-in-v19

APPENDIX
SOFTWARE TOOL OVERVIEW AND USER GUIDE

This appendix provides a user guide for the software tool that was created by the University of Illinois team to automate the process of translating data saved from PowerWorld, or a similar power system simulation package, into PMU like data and visualizations of it. The software tool is meant to provide a foundation for future researchers to easily develop new visualizations without having to worry about data formatting or processing. The data input and output from the software tool is well documented and standardized, such that the "wheel" does not need to keep being reinvented for each new visualization idea. In addition to providing a tool for future research, the software program also showcases the work completed by the University of Illinois team, with regards to visualization ideas. The following sections describe how to set up and use our open source software tool. Note that this user guide will likely change as improvements are made to the software tool, so refer to my GitHub repository, https://github.com/kgegner/analytics_tool, for the latest version.

A.1 COMPUTER AND SOFTWARE SETUP

In order for the software tool to work, several software packages and installations are required, including a data science package called Anaconda, which contains Python, a graphical user interface tool for Python called Kivy, and other packages for mapping and plotting in Python. All these files can be found at the links given in each step, or in my GitHub repository, in the folder 'packages'. The software tool was written in Python.

1. Install Anaconda3-2.3.0 from their archives, here.
   - Kivy only works with Python 3.4, which the above installation of Anaconda uses. If/when Kivy supports 3.5, you can use the latest Anaconda installation, here.
   - Anaconda contains all the Python modules needed for data science, such as numpy, pandas, scikit-learn, and more.
2. Open a command prompt (Windows) or terminal window (Linux and OS X), and check if Python 3.4 is the default Python version:

```
python --version
```

If you do not see something with Python 3.4 listed, like:

```
python 3.4.5 :: Anaconda 2.3.0 (x86_64)
```

Try typing:

```
python3 --version
```

From here on, use either python or python3, based on which one returned the correct version, Python 3.4.x, in the previous step.

3. Install Kivy for Python 3, from here.
   - Instructions for Linux, OS X, and Windows installations are found at the above hyperlink.
   - If ‘python3’ was the only successful option, in Step 2, replace ‘python’ with ‘python3’ and ‘pip’ with ‘pip3’ in the instructions given at the hyperlink.
4. Install the basemap package, which will be used for plotting on geographic maps.

   On OS X:

   ```
   conda install -c anaconda basemap=1.0.7
   ```

   After the basemap installation, you may need to set your bash profile to use UTF-8. To do this, type in a terminal window:

   ```
   nano ~./bash_profile
   ```

   Then, add the following statements, as two separate lines, anywhere in the text:

   ```
   export LC_ALL=en_US.UTF-8
   export LANG=en_US.UTF-8
   ```

   On Windows:

   Download the correct wheel for your computer from here, and save in a folder of your choice. If you have a 32-bit processor use the wheel that contains ‘win32’ and if you have a 64-bit processor use the wheel that contains ‘amd64’.

   eg. basemap-1.0.8-cp34-none-win32.whl
   basemap-1.0.8-cp34-none-win_amd64.whl

   Open an Anaconda Command Prompt window, and navigate to the folder where you have saved the basemap wheel, something like the following:

   ```
   cd C:\Users\Name\Downloads
   ```

Within the folder from above, make sure the basemap wheel is listed when you type:

```
ls
```

If you do, install the basemap wheel you downloaded by typing:

```
pip install basemap-1.0.8-cp34-none-win_amd64.whl
```

Note: If your wheel name is different than the above, substitute your specific wheel name, after "install"

Verify that basemap has been installed by checking if it is returned when you type:

```
pip list
```

5. Because we installed an older version of Anaconda, so that Kivy would work with it, several packages need to be updated within the Anaconda package, namely pandas. To make the updates, in a terminal window or command prompt type:

   `conda update all` or to just update pandas `conda update pandas`

   - If you have issues, consult: https://github.com/conda/conda/issues/1967

A.2 FILE STRUCTURE

With the software installed. The next step is setting up the computer to store the PowerWorld files and their corresponding data. To do this, a folder hierarchy, within which all data will be stored, needs to be created.

First, create a folder called "cases". This folder will hold the actual PowerWorld files that will be used to create simulations. Note that PowerWorld files are referred to as cases. Figure A.1, shows an example with three PowerWorld cases saved. Each PowerWorld case file has an extension of ".pwd".

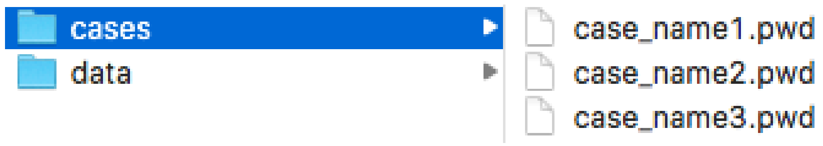


Figure A.1: Example file structure for storing PowerWorld case files.

Next, create a folder called "data". Within the "data" folder, create a folder for each case created and saved in the first step. For the example of Figure A.1, that means three folders titled "case_name1", "case_name2", and "case_name3" are needed. Then, within each case directory, create a folder for each simulation you have made or intend to make and a folder named "case_info". Each simulation folder will hold data generated when a simulation is performed and the “case_info” folder will contain data about the power system model being used (case). Figure A.2 illustrates the required file structure.

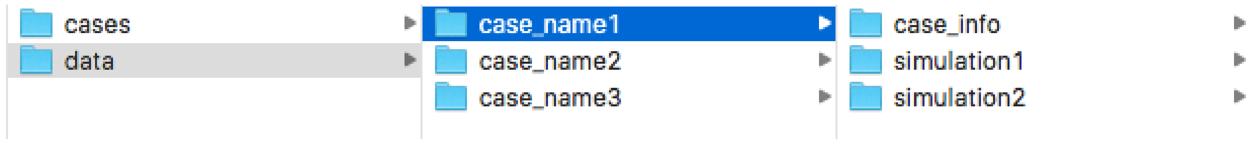


Figure A.2: Example file structure for storing data, from each PowerWorld case file listed in figure A.1.

Within each simulation folder and “case_info” folder, create a folder called “raw”. You can also create a folder called “formatted” in each of those folders, as is shown in Figures A.3 and A.4, but it is not necessary, since the software tool will automatically create the “formatted” folder when it runs. Data from PowerWorld will later be saved directly into the “raw” folders.

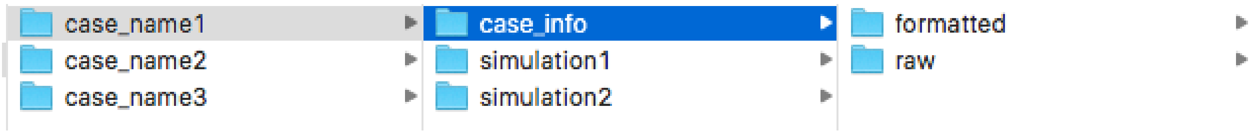


Figure A.3: Example file structure for storing case information from each PowerWorld case file listed in Figure A.1.

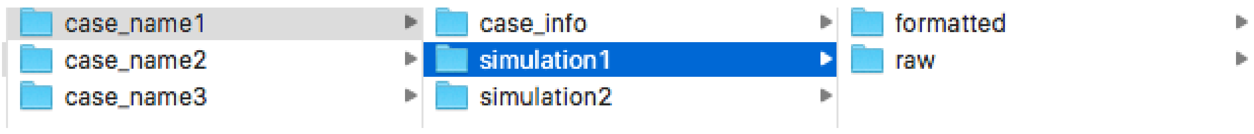


Figure A.4: Example file structure for storing simulation data, for each simulation created in a PowerWorld case file.

A.3 USING POWERWORLD AND SAVING DATA FROM IT

It is assumed that at least one electric grid model has already been created. You can find an example case in my GitHub repository, in the folder “cases”. The following instructions assume a basic working knowledge of PowerWorld. If you already have data saved in the format described in Section A.4, you can skip this section.

A.3.1 Saving Case Data

Save information about your case (power system model), including data for buses, substations, and generators, to the folder “case_info/raw”. This information is available by selecting “Model Explorer” from the “Case Information” tab in the ribbon, as shown in Figure A.5.

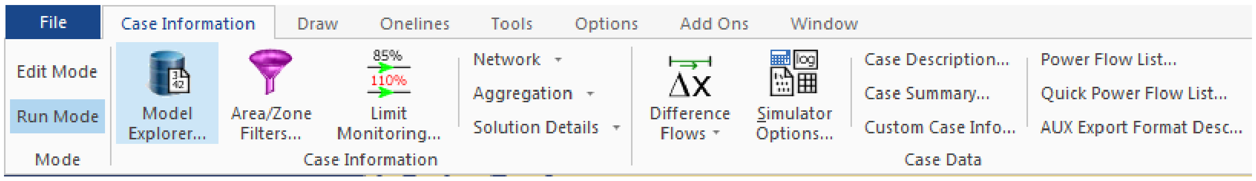


Figure A.5: Screenshot of how to open the Model Explorer to access case (power system model) information.

Once the dialog box opens, select "Buses" from the list of options in the left pane, as shown in Figure A.6. You will see lots of data in your version of PowerWorld, though I have whited out mine to ensure confidentiality. Make sure you have, at least, the columns shown in the selection box, towards the top of Figure A.6. If not, right click within the data, and select "Display /Column Options…". Select the missing column names from the dialog box that pops up. Then, to save the bus data select the icon shown in Figure A.7, and select "Send All to Excel" from the drop down menu. In Excel, save the spreadsheet as a csv file, named "buses.csv" and specify the directory to save it to as "data/case_nameX/case_info/", where you replace "case_nameX" with the name of your specific case. Repeat this process for "Generators" and "Substations", where the column names required are "Bus Number", "Area Name", and "Gen MW" and "Sub Num", "Area Name", "Nominal kV", "Latitude", and "Longitude", respectively. Name the data for generators "gens.csv" and for substations, "subs.csv". When all case data has been saved, the folder layout will look similar to Figure A.8.

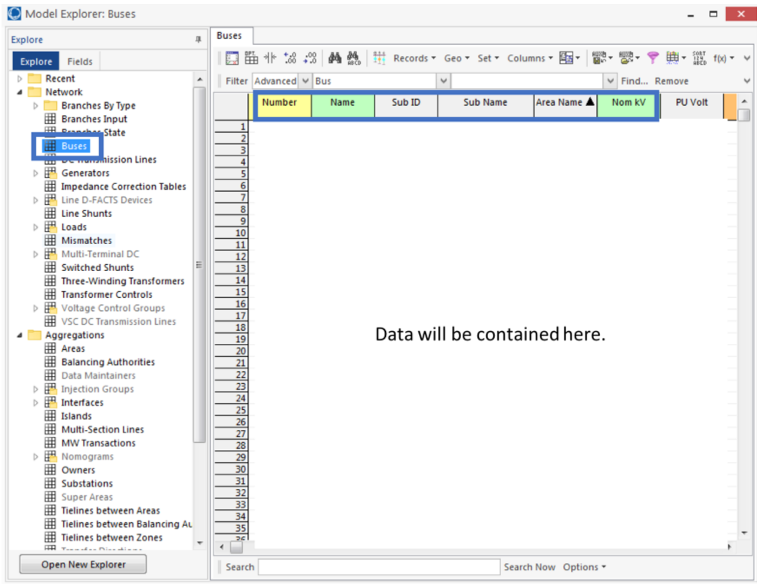


Figure A.6: Bus data and the required columns.

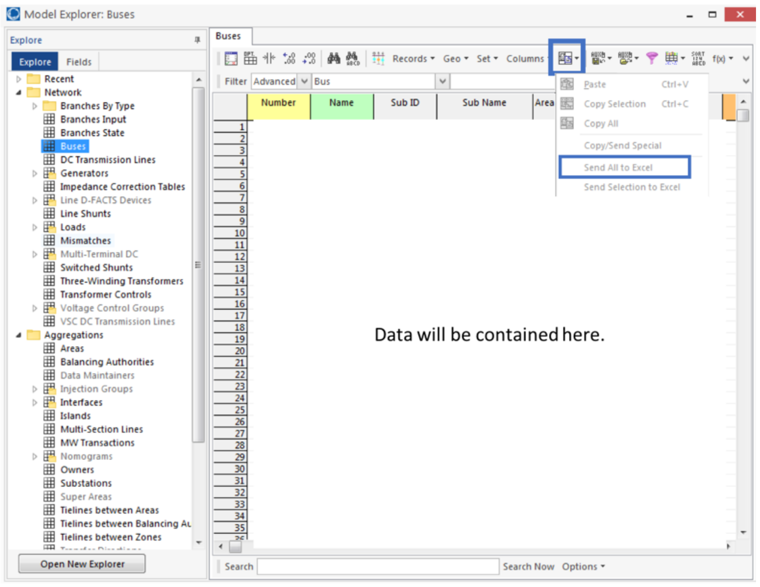


Figure A.7: Screenshot of how to send bus data, for a case file, to Excel for saving.

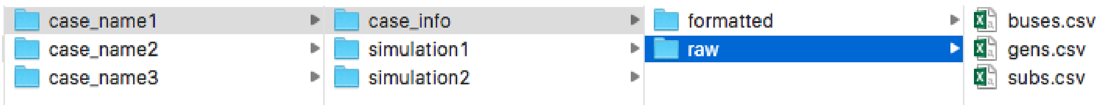


Figure A.8: Example of file structure for storing case (model) data for a PowerWorld case file.

A.3.2 Creating Transient Stability Simulations

This section describes how to create a transient stability simulation in PowerWorld. Within a PowerWorld case, open up a transient stability window. The transient stability tool is used to create the power system events for simulating earthquakes, physical equipment damage, etc. that will strain the power system model (case) and provide time stamped measurements at each bus.

To open the transient stability tool, select "Transient Stability" in the ribbon tab "Add Ons", as shown in Figure A.9.

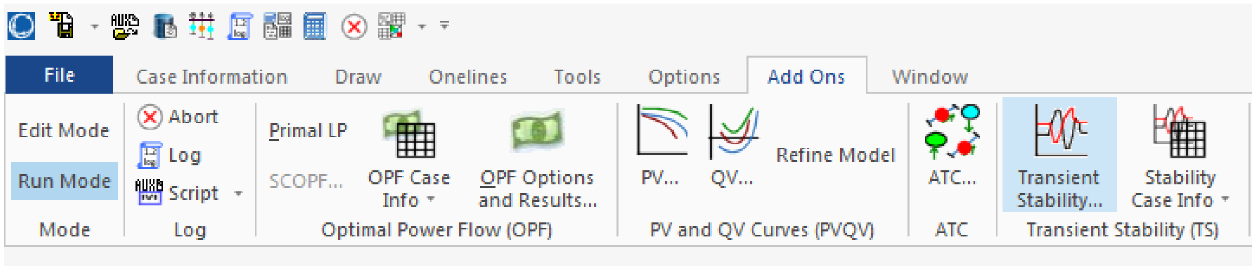
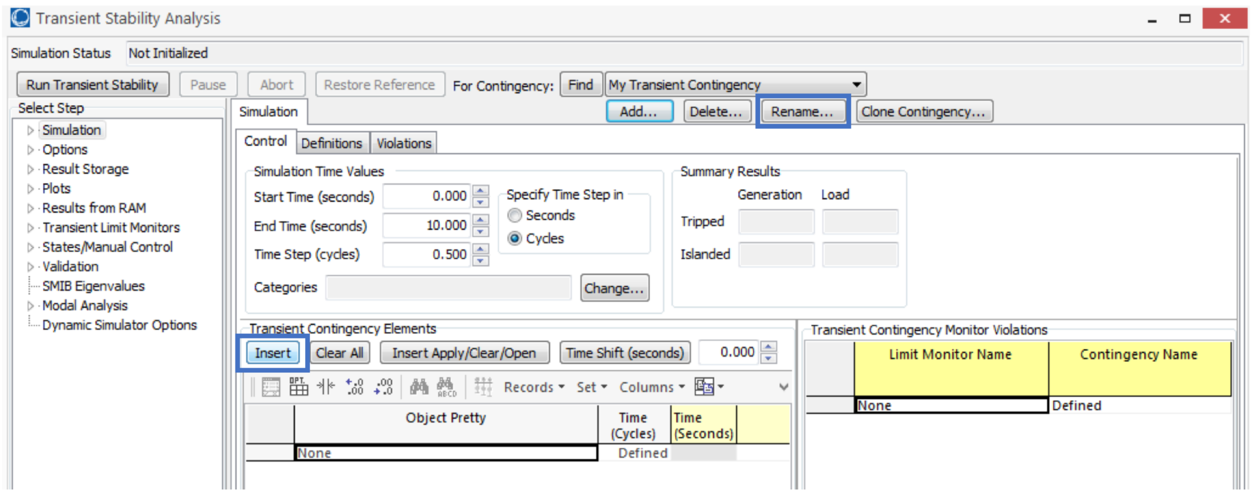


Figure A.9: Screenshot of how to open the Transient Stability tool in PowerWorld.

A dialog box, like the one shown in Figure A.10, will appear. In it, make a name for your new simulation, like “earthquake”, by selecting the button “Rename”, or if you already have a simulation and want to create another one, select “Add” and then “Rename”. Then, create a new transient stability definition by clicking on the button ‘Insert’.



Simulation Status: Not Initialized

For Contingency: My Transient Contingency

Start Time (seconds): 0.000
End Time (seconds): 10.000
Time Step (cycles): 0.500

Specify Time Step in: Seconds [✓ Cycles]

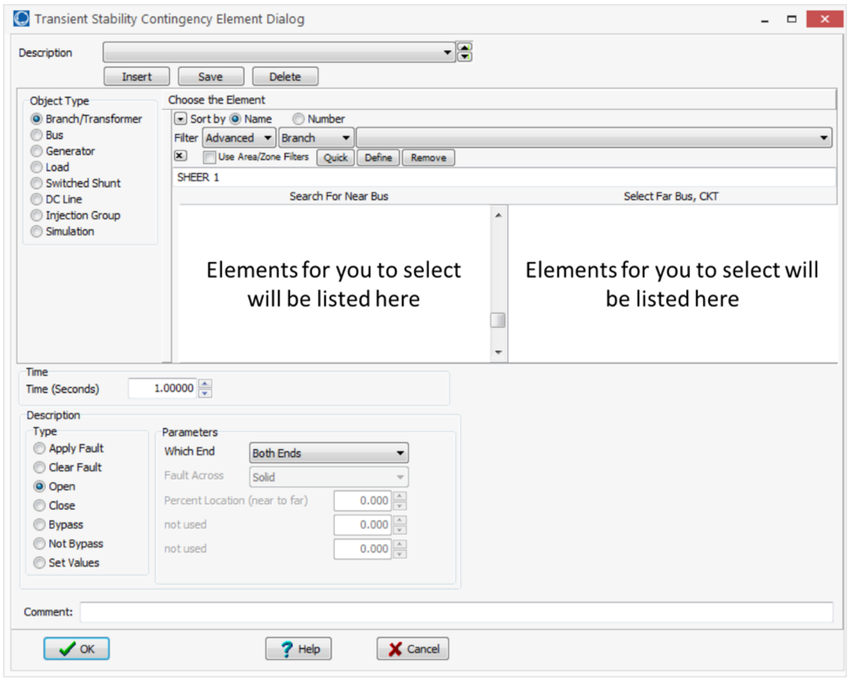
Time Shift (seconds): 0.000

Object Pretty	Time (Cycles)	Time (Seconds)
None	Defined	

Limit Monitor Name	Contingency Name
None	Defined

Figure A.10: Screenshot of a transient stability home dialog box, with buttons highlighted for renaming or starting a new transient stability definition.

A new dialog box, like the one shown in Figure A.11, will open. From this dialog box, you can select the type of equipment you want to affect in your simulation, in the far left pane, which exact element will be affected, in the central pane, and how it will be affected, in the bottom pane. For each piece of equipment you want to impact, repeat the process of pressing “Insert” and then completing the definition when the new dialog box opens. Note that a transient stability simulation will often have at least a few pieces of equipment that are being affected, though a simulation can be run without any equipment modifications. This can be especially helpful to get a baseline assessment of what is normal for the system.


Transient Stability Contingency Element Dialog

Description:

Insert | Save | Delete

Object Type: [✓ Branch/Transformer] Bus, Generator, Load, Switched Shunt, DC Line, Injection Group, Simulation

Choose the Element

Sort by: [✓ Name] Number

Filter: Advanced | Branch

Use Area/Zone Filters | Quick | Define | Remove

SHEER 1

Search For Near Bus	Select Far Bus, CKT
Elements for you to select will be listed here	Elements for you to select will be listed here

Time (Seconds): 1.00000

Description

Type: Apply Fault, Clear Fault, [✓ Open] Close, Bypass, Not Bypass, Set Values

Parameters

Which End	Both Ends
Fault Across	Solid
Percent Location (near to far)	0.000
not used	0.000
not used	0.000

Comment:

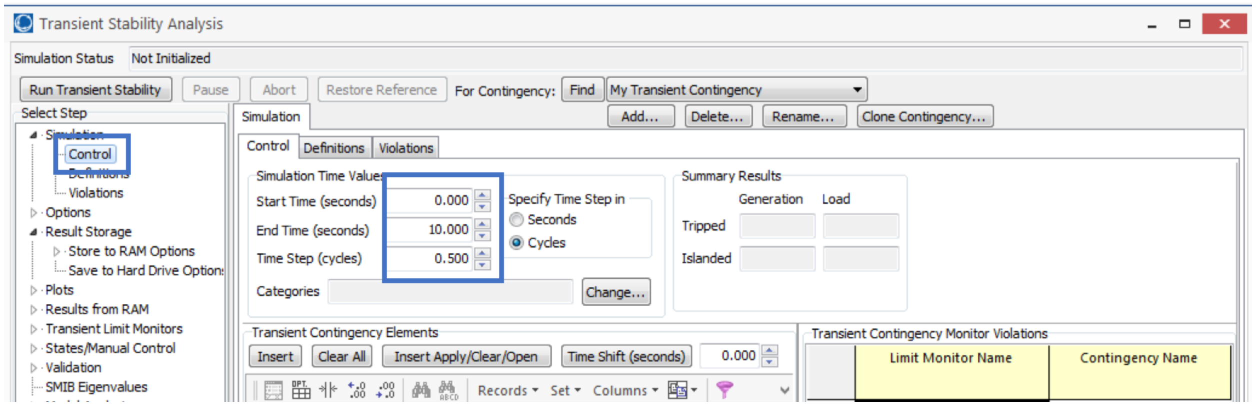
OK | Help | Cancel

Figure A.11: Screenshot of the dialog box used to create a transient stability event for a piece of equipment in the system.

When the transient stability simulation has been created, the next step requires configuring how the simulation should run and what data should be collected when it does. First, set up how long the simulation should run and what time step it should use. To do this, in the home dialog box of the Transient Stability tool, select “Simulation” → “Control”, and then make your settings for the items outlined in blue of Figure A.12. To collect data 30 times per second, as a PMU would, you can leave the time step at 0.5 seconds, for now.

Then, change how often data measurements are saved, by selecting “Results Storage” → ”Store to Ram Options”, and set the value for “Save Results Every n Time Steps”. To mimic PMUs, set this value to 8, as shown in Figure A.13.

Next, you need to make sure results are saved according to their bus number. To do this, select “Options” → “General” and then select the radio button “Number” on the far right, under “Identify Buses in Events and Results by”, shown in Figure A.14.



Simulation Status: Not Initialized

For Contingency: My Transient Contingency

Simulation / Control

Simulation Time Values
Start Time (seconds) 0.000
End Time (seconds) 10.000
Time Step (cycles) 0.500
Categories

Specify Time Step in: Seconds [✓ Cycles]

Summary Results

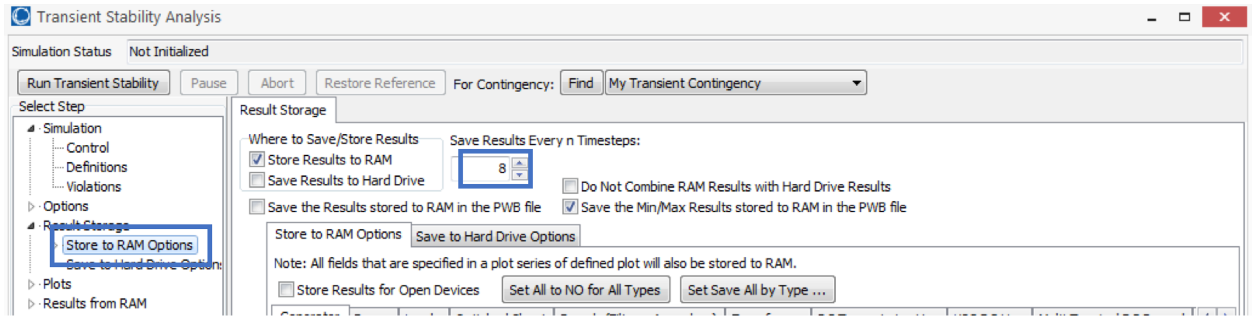
	Generation	Load
Tripped		
Islanded		

Transient Contingency Elements
Time Shift (seconds) 0.000

Transient Contingency Monitor Violations

Limit Monitor Name	Contingency Name

Figure A.12: Screenshot of which buttons and text boxes to adjust to change transient stability simulation run-time and time step.



Simulation Status: Not Initialized

For Contingency: My Transient Contingency

Result Storage

Where to Save/Store Results
☑ Store Results to RAM
☐ Save Results to Hard Drive

Save Results Every n Timesteps: 8

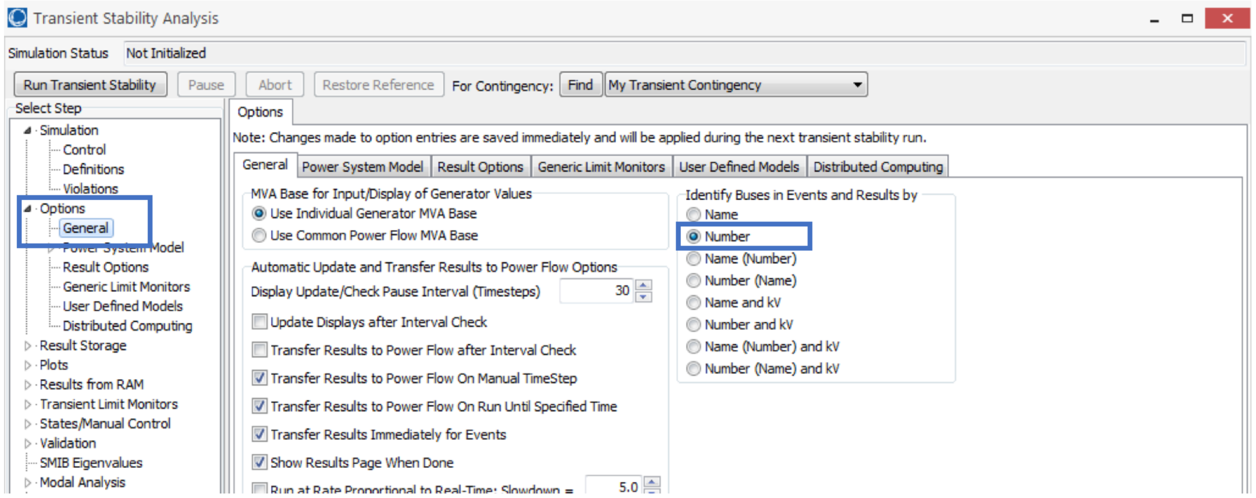
☐ Do Not Combine RAM Results with Hard Drive Results
☐ Save the Results stored to RAM in the PWB file
☑ Save the Min/Max Results stored to RAM in the PWB file

Store to RAM Options | Save to Hard Drive Options

Note: All fields that are specified in a plot series of defined plot will also be stored to RAM.

☐ Store Results for Open Devices

Figure A.13: Screenshot of which buttons and text boxes to adjust to change often to record data measurements.



Simulation Status: Not Initialized

For Contingency: My Transient Contingency

Options

Note: Changes made to option entries are saved immediately and will be applied during the next transient stability run.

General | Power System Model | Result Options | Generic Limit Monitors | User Defined Models | Distributed Computing

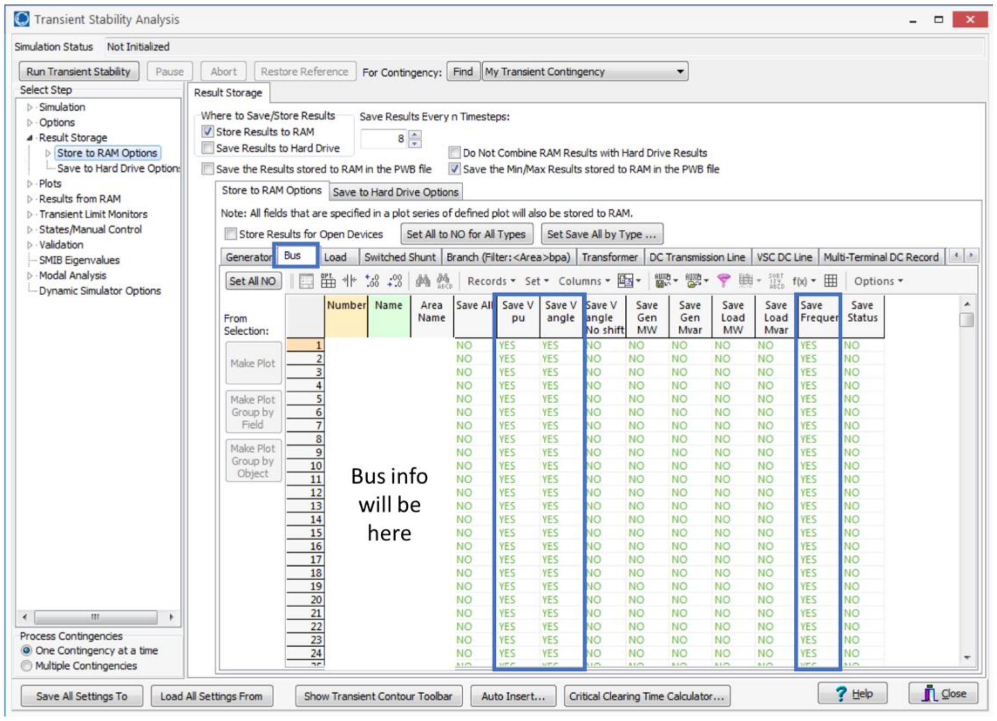
MVA Base for Input/Display of Generator Values
[✓ Use Individual Generator MVA Base]
Use Common Power Flow MVA Base

Automatic Update and Transfer Results to Power Flow Options
Display Update/Check Pause Interval (Timesteps) 30
☐ Update Displays after Interval Check
☐ Transfer Results to Power Flow after Interval Check
☑ Transfer Results to Power Flow On Manual TimeStep
☑ Transfer Results to Power Flow On Run Until Specified Time
☑ Transfer Results Immediately for Events
☑ Show Results Page When Done
☐ Run at Rate Proportional to Real-Time: Slowdown = 5.0

Identify Buses in Events and Results by
Name
[✓ Number]
Name (Number)
Number (Name)
Name and kV
Number and kV
Name (Number) and kV
Number (Name) and kV

Figure A.14: Screenshot of which buttons to press to make sure results are saved according to their bus number.

Lastly, you need to make sure that the correct measurements will be recorded. To replicate what PMUs would collect, we need measurements at buses for frequency, voltage angle, and voltage magnitude and in branches, the current magnitude. In the home transient stability dialog box, select "Results Storage" → "Store to Ram Options". First, select the "Bus" tab in the central pane, shown in Figure A.15. Then make sure all values in the columns “Save V pu”, Save V angle”, and “Save frequency” are set to “YES”. If not, right click somewhere within the column, and select “Set/Toggle/Columns” → “All YES”. Second, select the “Branch” tab in the central pane, shown in Figure A.16, and make sure all values in the “Save Current To” column are set to “YES”. All other columns can be left as they are. Though, if any extras are set to “YES” the simulation will run more slowly and the computer memory will fill more quickly. Thus, it is best to set any unnecessary columns to all have values of “NO”.


Transient Stability Analysis

Simulation Status Not Initialized

For Contingency: My Transient Contingency

Where to Save/Store Results
- [✓] Store Results to RAM
- [ ] Save Results to Hard Drive
- [ ] Save the Results stored to RAM in the PWB file

Save Results Every n Timesteps: 8

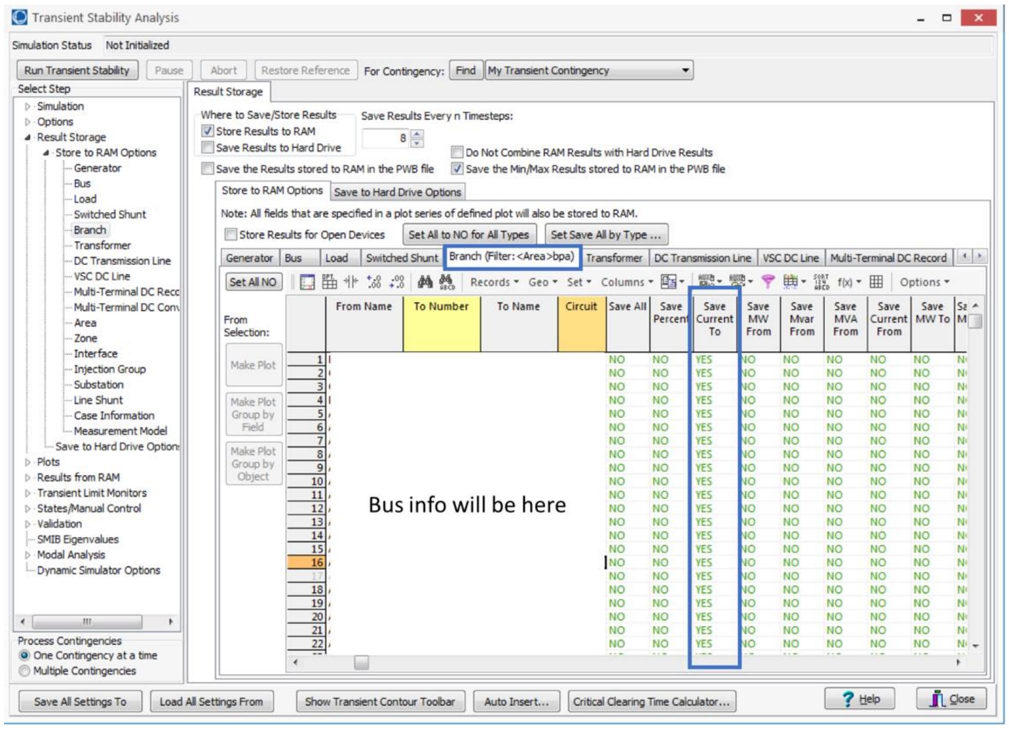
- [ ] Do Not Combine RAM Results with Hard Drive Results
- [✓] Save the Min/Max Results stored to RAM in the PWB file

Note: All fields that are specified in a plot series of defined plot will also be stored to RAM.

- [ ] Store Results for Open Devices

	Number	Name	Area Name	Save All	Save V pu	Save V angle	Save V angle No shift	Save Gen MW	Save Gen Mvar	Save Load MW	Save Load Mvar	Save Frequer	Save Status
1				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
2				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
3				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
4				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
5				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
6				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
7				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
8				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
9				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
10				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
11				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
12				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
13				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
14				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
15				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
16				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
17				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
18				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
19				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
20				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
21				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
22				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
23				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO
24				NO	YES	YES	NO	NO	NO	NO	NO	YES	NO

Figure A.15: Screenshot of how to specify which measurements should be recorded for buses, during each simulation.



Where to Save/Store Results
[✓] Store Results to RAM
[ ] Save Results to Hard Drive
Save Results Every n Timesteps: 8
[ ] Do Not Combine RAM Results with Hard Drive Results
[ ] Save the Results stored to RAM in the PWB file
[✓] Save the Min/Max Results stored to RAM in the PWB file

Note: All fields that are specified in a plot series of defined plot will also be stored to RAM.
[ ] Store Results for Open Devices
Set All to NO for All Types
Set Save All by Type ...

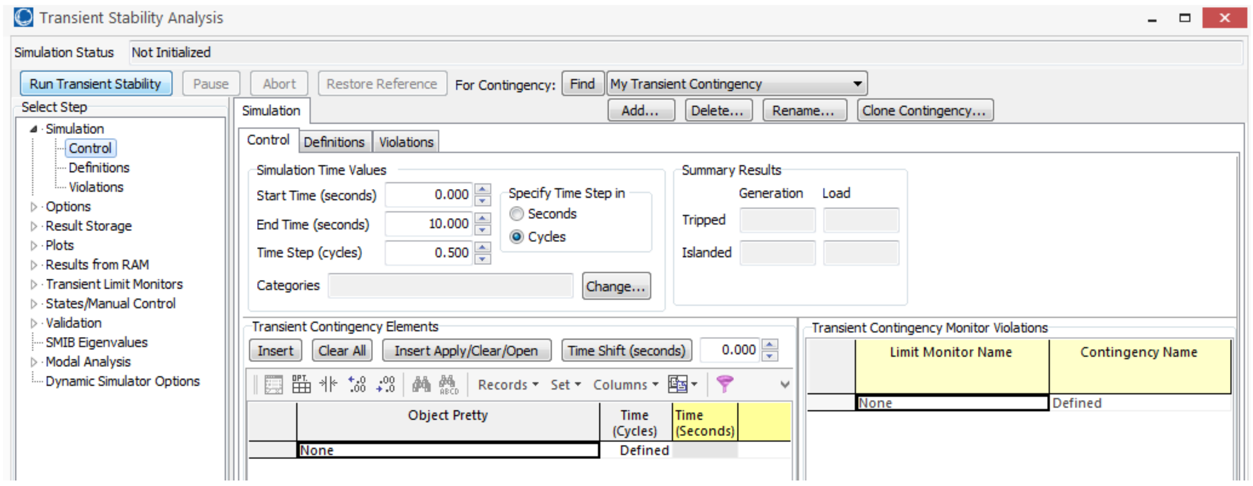
From Selection:

Process Contingencies: [✓ One Contingency at a time] Multiple Contingencies

	From Name	To Number	To Name	Circuit	Save All	Save Percent	Save Current To	Save MW From	Save Mvar From	Save MVA From	Save Current From	Save MW To
1					NO	NO	YES	NO	NO	NO	NO	NO
2					NO	NO	YES	NO	NO	NO	NO	NO
3					NO	NO	YES	NO	NO	NO	NO	NO
4					NO	NO	YES	NO	NO	NO	NO	NO
5					NO	NO	YES	NO	NO	NO	NO	NO
6					NO	NO	YES	NO	NO	NO	NO	NO
7					NO	NO	YES	NO	NO	NO	NO	NO
8					NO	NO	YES	NO	NO	NO	NO	NO
9					NO	NO	YES	NO	NO	NO	NO	NO
10					NO	NO	YES	NO	NO	NO	NO	NO
11		Bus info will be here			NO	NO	YES	NO	NO	NO	NO	NO
12					NO	NO	YES	NO	NO	NO	NO	NO
13					NO	NO	YES	NO	NO	NO	NO	NO
14					NO	NO	YES	NO	NO	NO	NO	NO
15					NO	NO	YES	NO	NO	NO	NO	NO
16					NO	NO	YES	NO	NO	NO	NO	NO
17					NO	NO	YES	NO	NO	NO	NO	NO
18					NO	NO	YES	NO	NO	NO	NO	NO
19					NO	NO	YES	NO	NO	NO	NO	NO
20					NO	NO	YES	NO	NO	NO	NO	NO
21					NO	NO	YES	NO	NO	NO	NO	NO
22					NO	NO	YES	NO	NO	NO	NO	NO

Figure A.16: Specify which measurements should be recorded for every branch.

With all the aforementioned steps complete, the simulation can be run by selecting "Run Transient Stability Simulation" from the top of the transient stability home dialog box, as shown in Figure A.17.



Simulation Time Values
Start Time (seconds) 0.000
End Time (seconds) 10.000
Time Step (cycles) 0.500
Specify Time Step in: Seconds [✓ Cycles]
Categories

Summary Results

	Generation	Load
Tripped		
Islanded		

Transient Contingency Elements
Time Shift (seconds) 0.000

Object Pretty	Time (Cycles)	Time (Seconds)
None	Defined	

Transient Contingency Monitor Violations

Limit Monitor Name	Contingency Name
None	Defined

Figure A.17: Screenshot of how to start a transient stability simulation.

A.3.3 Saving Transient Simulation Data

Once a simulation is complete, the data from it needs to be saved to csv files. To do this, within the home transient stability dialog box shown in Figure A.10, on the left side, select "Results from

RAM” → “Time Values”. Then, select the “Bus” tab. On the left side of the screen, select the measurement type you want to look at. First, select frequency. Then, click on the icon shown in Figure A.18, and select “Send All to Excel” from the drop down menu. In Excel, save the file as “bus_freq.csv” in the folder “/data/case_nameX/simulationX/raw/”. Repeat this for the measurement types “V angle” and “V pu”, and save the files, in the same folder, as “bus_vang.csv” and “bus_vmag.csv”, respectively. Note, if you want the actual kV value instead of the per-unit value, for voltage magnitude, simply select “V (kV)” instead of “V pu”.

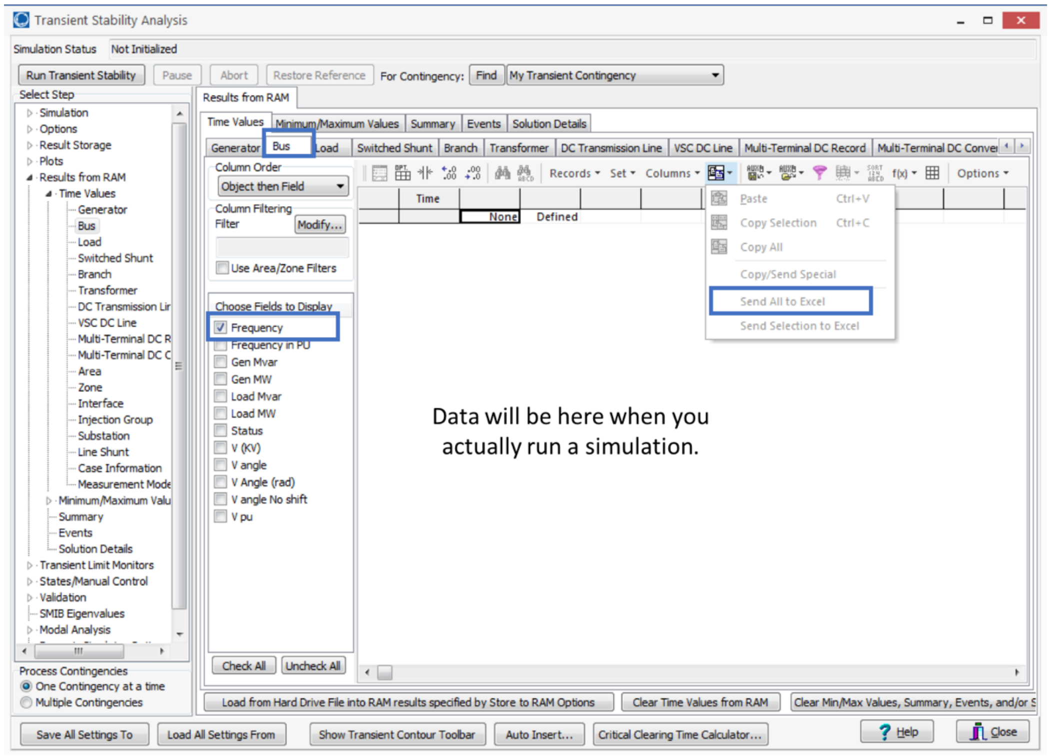
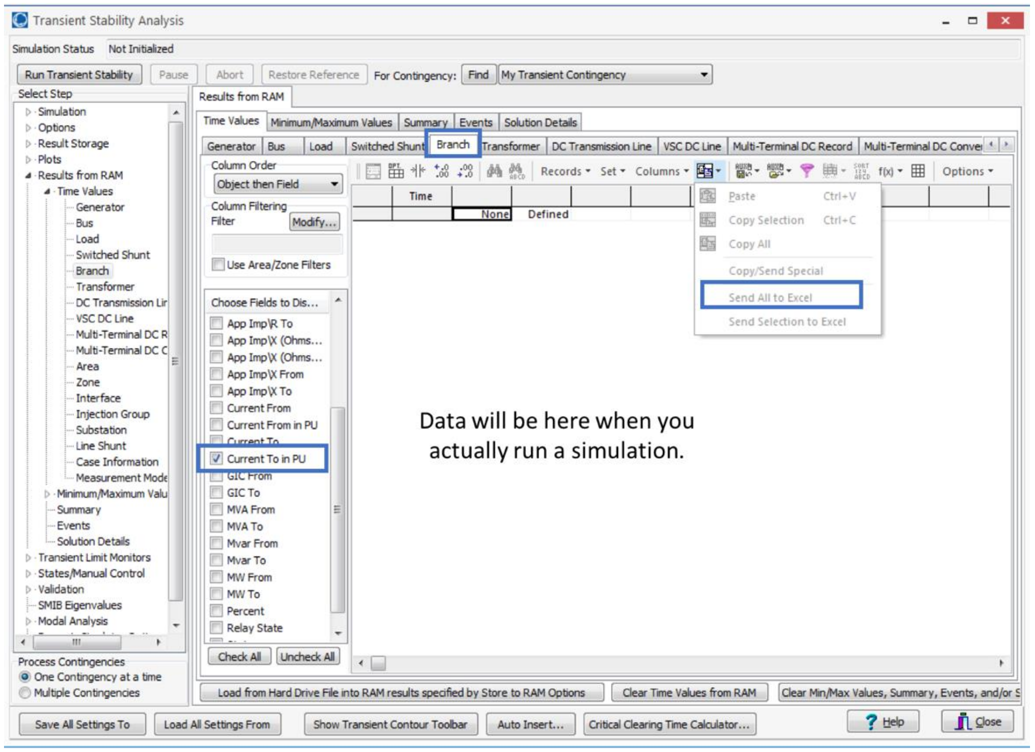


Figure A.18: How to save frequency data from the transient stability simulation. Follow the same process to save voltage angle and voltage magnitude data, simply be selecting their corresponding checkbox, instead of “Frequency”.

Then, in the same window, select the “Branch” tab. Select the measurement type of “Current To in PU”. Then, click on the icon shown in Figure A.19, and select “Send All to Excel” from the drop down menu. In Excel, save the file as “branch_cmag.csv” in the folder “/data/case_nameX/simulationX/raw/”. Note, if you want the actual ampere value instead of the per-unit value, for current magnitude, simply select “Current To” instead of “Current To in PU”. When all files have been saved, the data folder should look like the one shown in Figure A.20.



Data will be here when you actually run a simulation.

Figure A.19: How to save current magnitude data from a transient stability simulation.

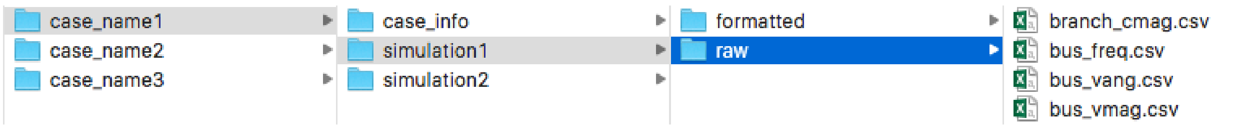


Figure A.20: File structure for simulation data.

A.4 DATA FORMATTING

This section provides an overview of the format for data that is input to the software tool, and data that is output from it. Note, if you have data already saved in the format required for input data, or can easily put it in that format, you may not need to use PowerWorld to generate any more data.

A.4.1 Input Data

This section describes the file format and location for data that is saved directly from PowerWorld and is the data that will be read in by the software tool. All data that is considered input data is stored in the “raw” folders, which are subdirectories of the “case_info” (Figure A.8) and “simulationX” (Figure A.20) folders. Input data is broken into two categories, Case Data and

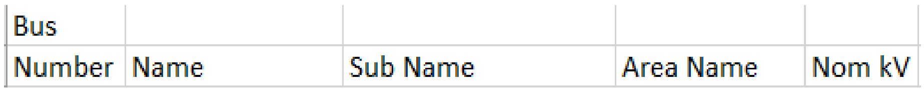
Simulation Data, as is described in Table A.1. A screenshot of the format of Case Data files is shown in Figure A.21 and Simulation Data files in Figure A.22. Note that no data is shown for any of the Case Data files, just the column headings, to ensure confidentiality.

Table A.1: Data input to the software tool. Case data describes the PowerWorld file and the power system model contained within it, and simulation data contains the measurements recorded during a simulation.

File Name (.csv)	**Type**	**Description**
buses	Case	Contains bus name and number, voltage level, and substation identifier, for every bus in the system.
subs	Case	Contains substation identifier, operating area name, and geographic coordinates, for every substation in the system.
gen	Case	Contains bus number, MW generating capacity, and substation identifier for every generator in the system.
real_pmus	Case	If known, contains a list of substations where PMUs are actually located in a real system, and their corresponding voltage levels
bus_freq	Simulation	Frequency measurements, in Hz, recorded at every bus in the PowerWorld model
bus_vang	Simulation	Voltage angle measurements, in degrees, recorded at every bus in the PowerWorld model
bus_vmag	Simulation	Voltage magnitude measurements, in per-unit, recorded at every bus in the PowerWorld model
branch_cmag	Simulation	Current magnitude measurements, in per-unit, recorded coming in to every bus in the PowerWorld model

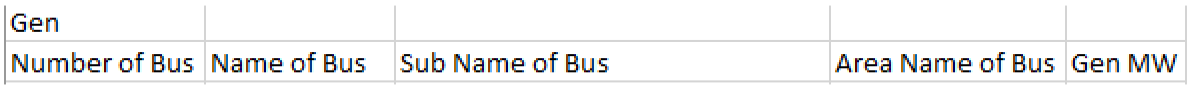
CASE DATA

buses.csv



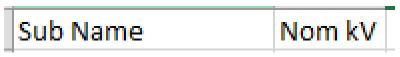
Bus				
Number	Name	Sub Name	Area Name	Nom kV

gens.csv



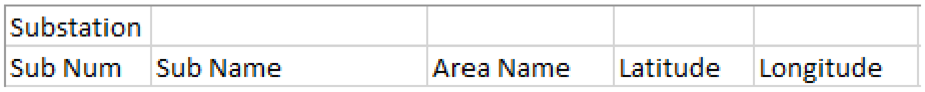
Gen				
Number of Bus	Name of Bus	Sub Name of Bus	Area Name of Bus	Gen MW

real_pmus.csv (if any PMU substations are known at the beginning)



Sub Name	Nom kV

subs.csv

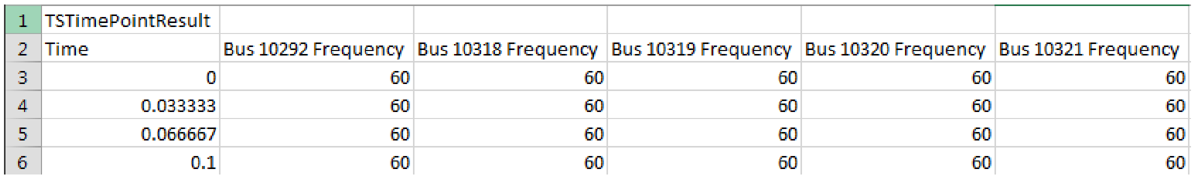


Substation				
Sub Num	Sub Name	Area Name	Latitude	Longitude

Figure A.21: Screenshot of what each case data csv file looks like. Note that the first row can be omitted, where it says "Bus", "Gen", or "Substations" and the software tool will still work. The column headings of the second row need to be the same.

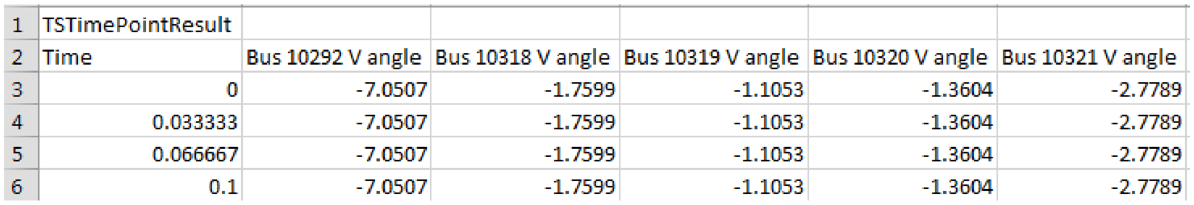
SIMULATION DATA

bus_freq.csv



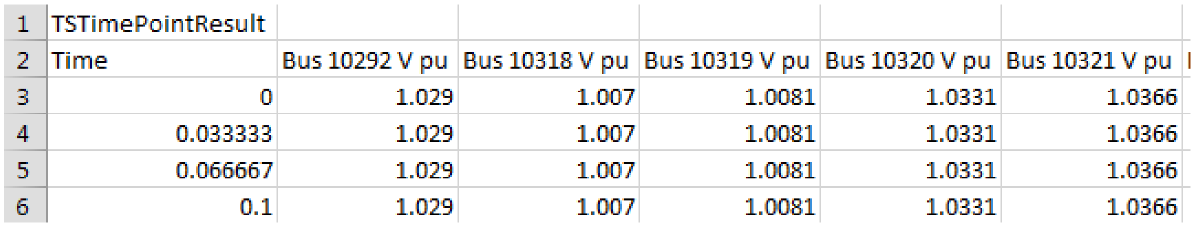
1	TSTimePointResult					
2	Time	Bus 10292 Frequency	Bus 10318 Frequency	Bus 10319 Frequency	Bus 10320 Frequency	Bus 10321 Frequency
3	0	60	60	60	60	60
4	0.033333	60	60	60	60	60
5	0.066667	60	60	60	60	60
6	0.1	60	60	60	60	60

bus_vang.csv



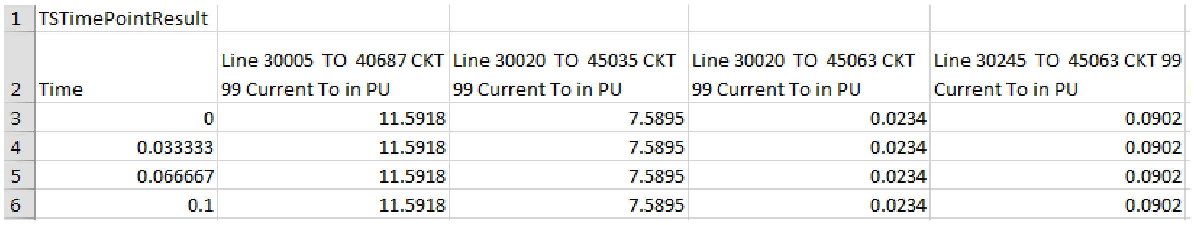
1	TSTimePointResult					
2	Time	Bus 10292 V angle	Bus 10318 V angle	Bus 10319 V angle	Bus 10320 V angle	Bus 10321 V angle
3	0	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789
4	0.033333	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789
5	0.066667	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789
6	0.1	-7.0507	-1.7599	-1.1053	-1.3604	-2.7789

bus_vmag.csv



1	TSTimePointResult					
2	Time	Bus 10292 V pu	Bus 10318 V pu	Bus 10319 V pu	Bus 10320 V pu	Bus 10321 V pu
3	0	1.029	1.007	1.0081	1.0331	1.0366
4	0.033333	1.029	1.007	1.0081	1.0331	1.0366
5	0.066667	1.029	1.007	1.0081	1.0331	1.0366
6	0.1	1.029	1.007	1.0081	1.0331	1.0366

branch_cmag.csv



1	TSTimePointResult				
2	Time	Line 30005 TO 40687 CKT 99 Current To in PU	Line 30020 TO 45035 CKT 99 Current To in PU	Line 30020 TO 45063 CKT 99 Current To in PU	Line 30245 TO 45063 CKT 99 Current To in PU
3	0	11.5918	7.5895	0.0234	0.0902
4	0.033333	11.5918	7.5895	0.0234	0.0902
5	0.066667	11.5918	7.5895	0.0234	0.0902
6	0.1	11.5918	7.5895	0.0234	0.0902

Figure A.22: Screenshot of what each simulation data csv file looks like. Note that the first row can be omitted, where it says "TSTimePointResult" and the software tool will still work. The column headings of the second row need to be of the same format, though the exact bus or circuit numbers will be different.

A.4.2 Output Data

This section describes the file format and location for data exported from the software tool. All data that is considered output data is stored in the "formatted" folders, which are subdirectories of the "case_info" (Figure A.8) and "simulationX" (Figure A.20) folders. If the "formatted" folder does not exist in either the "case_info" or "simulationX" folders, the software tool will automatically create it. Output data is broken into two categories, Case Data and Simulation, PMU Data, as is described in Figure A.2. A screenshot of the format of Case Data files is shown in

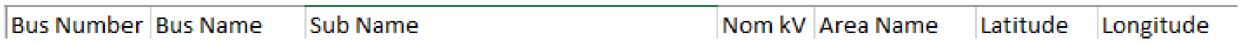
Figure A.23 and Simulation Data files in Figure A.24. Note that no data is shown for any of the Case Data files, just the column headings, to ensure confidentiality.

Table A.2: Data export from the software tool. Case data describes the PowerWorld file and the power system model contained within it, and Simulation, PMU data contains the measurements recorded during a simulation, but only for a select number of buses being monitored by real or fictitiously assigned PMUs.

File Name (.csv)	**Type**	**Description**
bus_info	Case	Contains bus name and number, voltage level, substation identifier, and geographic coordinates of the substation, for every bus in the system.
gens	Case	Contains list of all buses at substations with a generator, total MW generating capacity, substation identifier, for every generating substation in a system
pmu_info	Case	Contains bus name number, voltage level, substation identifier, operating area name, and geographic coordinates for every bus/substation being monitored by a real or fictitious PMU
pmu_freq	Simulation, PMU	Frequency measurements, in Hz, recorded at every bus being monitored by a PMU
pmu_vang	Simulation, PMU	Voltage angle measurements, in degrees, recorded at every bus being monitored by a PMU
pmu_vmag	Simulation, PMU	Voltage magnitude measurements, in per-unit, recorded at every bus being monitored by a PMU
pmu_cmag	Simulation, PMU	Current magnitude measurements, in per-unit, recorded coming in to every bus being monitored by a PMU

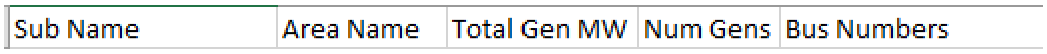
CASE DATA

bus_info.csv



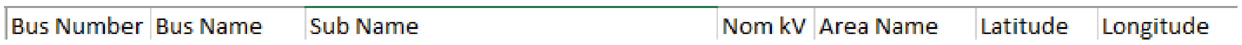
Bus Number	Bus Name	Sub Name	Nom kV	Area Name	Latitude	Longitude

gens.csv



Sub Name	Area Name	Total Gen MW	Num Gens	Bus Numbers

pmu_info.csv

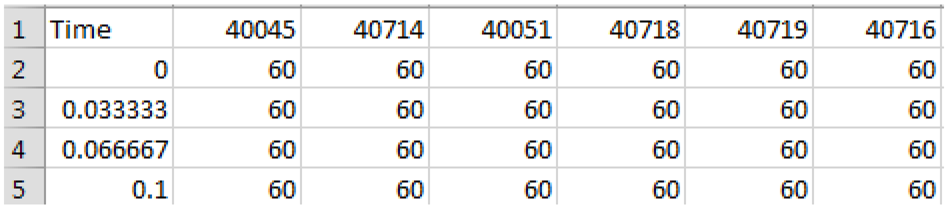


Bus Number	Bus Name	Sub Name	Nom kV	Area Name	Latitude	Longitude

Figure A.23: Screenshot of what each case data csv file looks like, after being export from the software tool.

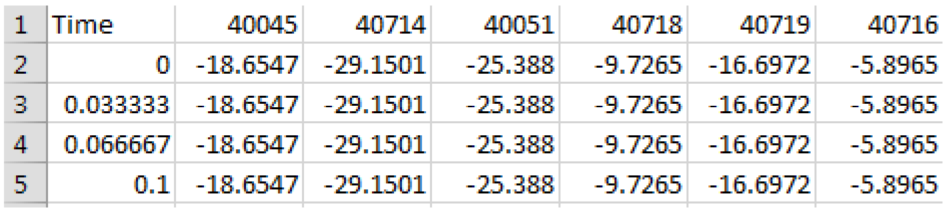
SIMULATION DATA

pmu_freq.csv



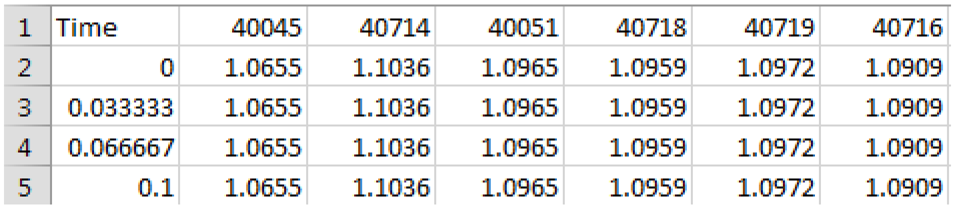
1	Time	40045	40714	40051	40718	40719	40716
2	0	60	60	60	60	60	60
3	0.033333	60	60	60	60	60	60
4	0.066667	60	60	60	60	60	60
5	0.1	60	60	60	60	60	60

pmu_vang.csv



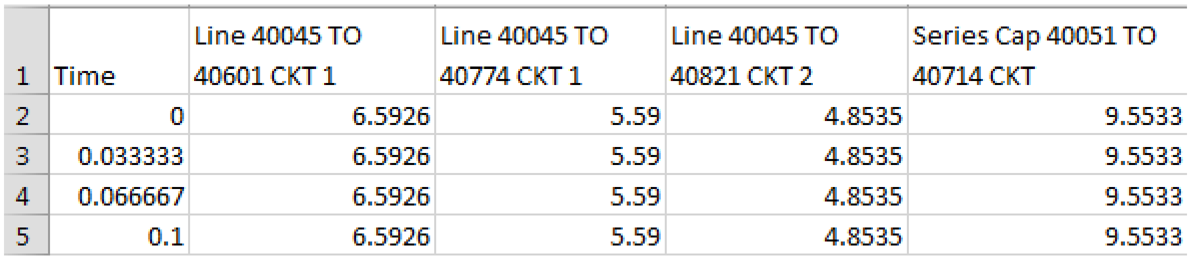
1	Time	40045	40714	40051	40718	40719	40716
2	0	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965
3	0.033333	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965
4	0.066667	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965
5	0.1	-18.6547	-29.1501	-25.388	-9.7265	-16.6972	-5.8965

pmu_vmag.csv



1	Time	40045	40714	40051	40718	40719	40716
2	0	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909
3	0.033333	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909
4	0.066667	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909
5	0.1	1.0655	1.1036	1.0965	1.0959	1.0972	1.0909

pmu_cmag.csv



1	Time	Line 40045 TO 40601 CKT 1	Line 40045 TO 40774 CKT 1	Line 40045 TO 40821 CKT 2	Series Cap 40051 TO 40714 CKT
2	0	6.5926	5.59	4.8535	9.5533
3	0.033333	6.5926	5.59	4.8535	9.5533
4	0.066667	6.5926	5.59	4.8535	9.5533
5	0.1	6.5926	5.59	4.8535	9.5533

Figure A.24: Screenshot of what each Simulation, PMU data csv file looks like, after it has been exported from the software tool. Columns are distinguished by their bus number.

A.5 DESCRIPTION OF SOFTWARE TOOL AND HOW TO USE IT

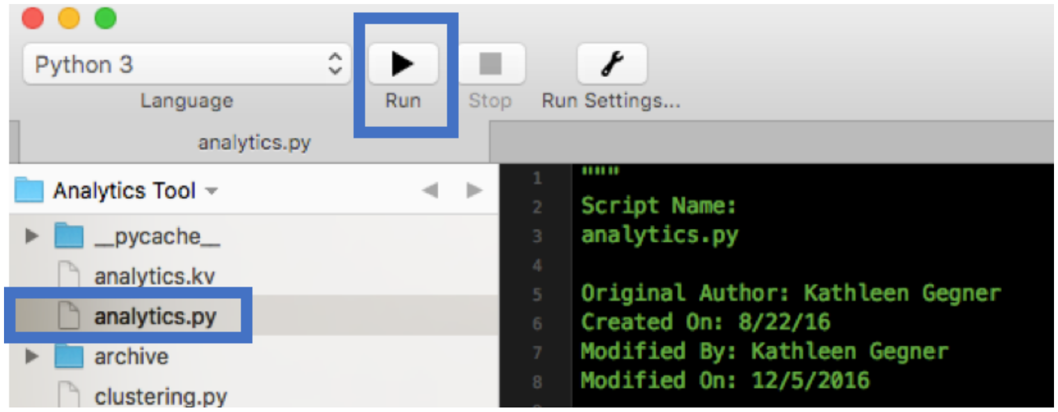
The software tool is a means to automate the process of turning PowerWorld simulation data into PMU data visualizations. Key functions of the software tool are as follows:

- If PMU locations are not known for a system, the program can create a list of substations that would likely have a PMU installed
- PMU data for frequency, voltage angle, and voltage magnitude can be processed

- The program uses a well-defined, documented, and unchanging format for data input and output from the program. This standardization allows for new modules to be built without having to reinvent the wheel each time.
- The program is written in Python, is all open source, and is hosted on GitHub.
- User interaction with the tool is pretty intuitive. More advanced settings can be changed in the actual code, and areas that lend themselves to those changes are well commented.
- K-means and DBSCAN clustering is built into the tool.
- A set of visualizations is displayed automatically, though the user can specify which other ones they may want to see.

As of this writing, the code is still being finished, so the following provides a basic overview of how the tool works. However, you are encouraged to consult the most up to date user guide, which can be found on GitHub, to make sure the below is still applicable. All instructions before this section should remain unchanged in any new code updates.

To launch the Data Analytics Tool, you may either use your choice of Integrated Development Environment (eg. PyCharm, Spyder, CodeRunner) or launch the tool from the command line. Using an Integrated Development Environment, open the file “analytics.py”, and then run the script. Figure A.25 shows how to do this using CodeRunner, by opening the “analytics.py” file and then hitting the “Run” button. Figure A.26 shows how to run the tool in PyCharm, where you need to open the file “analytics.py”, right click on the “analytics.py” tab, and then select “run analytics.py” from the drop down menu.



```
"""
Script Name:
analytics.py

Original Author: Kathleen Gegner
Created On: 8/22/16
Modified By: Kathleen Gegner
Modified On: 12/5/2016
```

Figure A.25: Screenshot of how to run the software tool using CodeRunner.

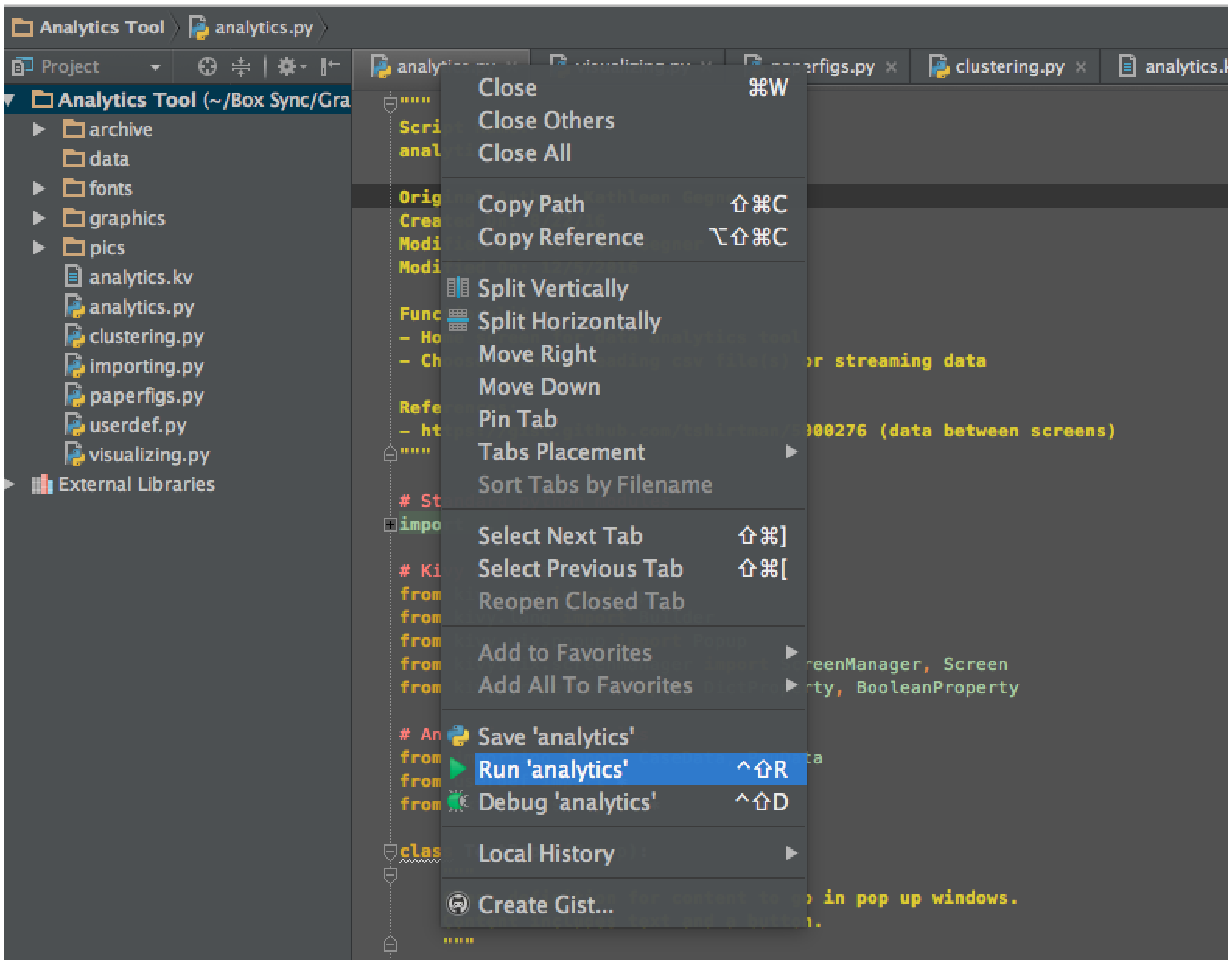
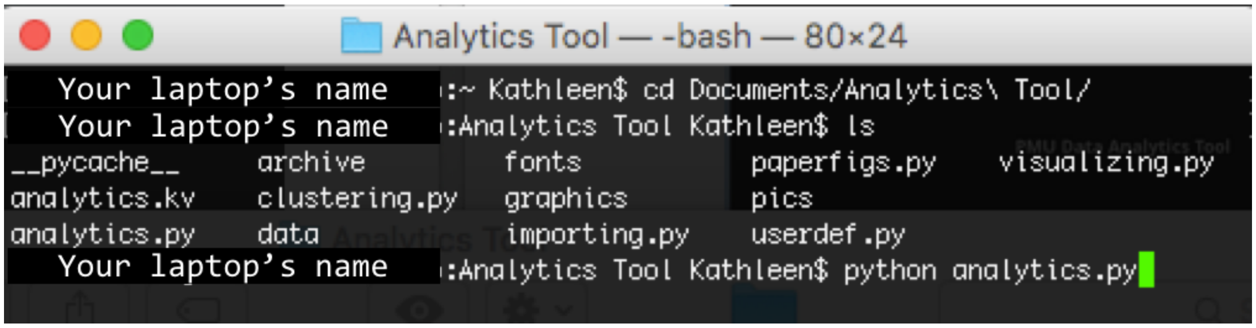


Figure A.26: Screenshot of how to run the software tool using PyCharm.

Alternatively, you can launch the software tool from a terminal window, by navigating to the folder where the file "analytics.py" is saved. As an example, see Figure A.27. Then, launch the software tool by typing:

```
python analytics.py
```



```
Analytics Tool — -bash — 80×24
[  Your laptop's name  :~ Kathleen$ cd Documents/Analytics\ Tool/
[  Your laptop's name  :Analytics Tool Kathleen$ ls
__pycache__     archive         fonts           paperfigs.py    visualizing.py
analytics.kv    clustering.py   graphics        pics
analytics.py    data            importing.py    userdef.py
   Your laptop's name  :Analytics Tool Kathleen$ python analytics.py
```

Figure A.27: Screenshot of how to navigate to the analytics.py file folder, and how to run the script.

Once the software tool has been launched, a screen similar to Figure A.28 will appear. Select the option “Load CSV File”. A new screen will pop up, like that in Figure A.29. Select the folder (one-click) that contains the simulation data you want to use. From there, the program should handle everything else, until interactive visualization screens appear, like Figures A.30-31.

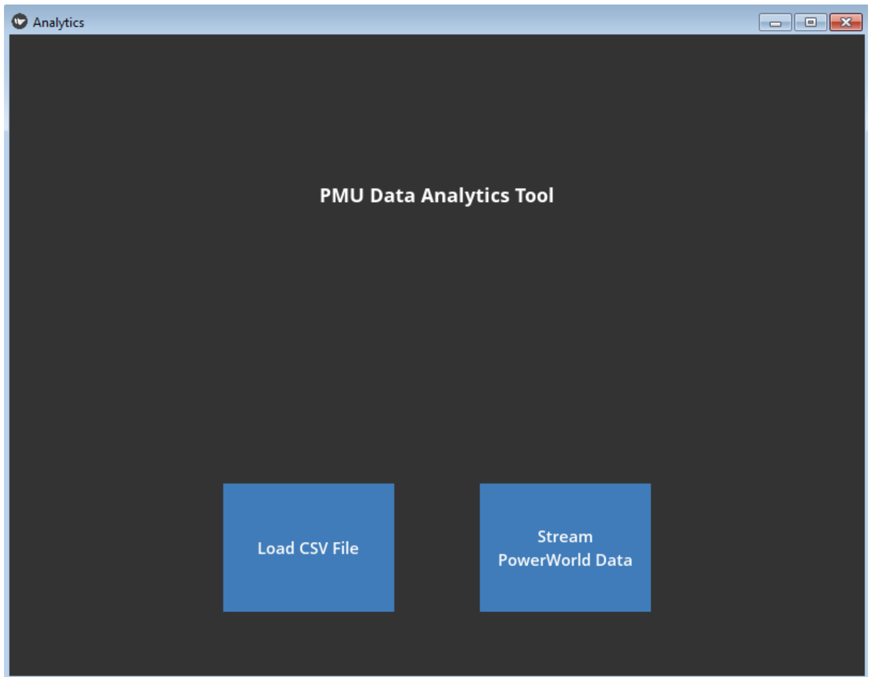


Figure A.28: Analytics tool home screen.

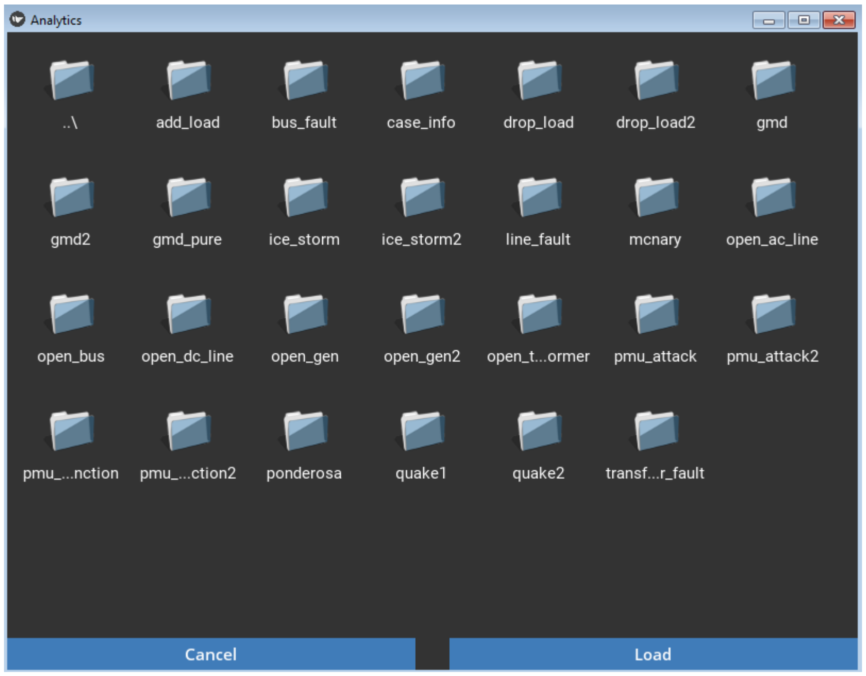


Figure A.29: Analytics tool simulation data folder selection.

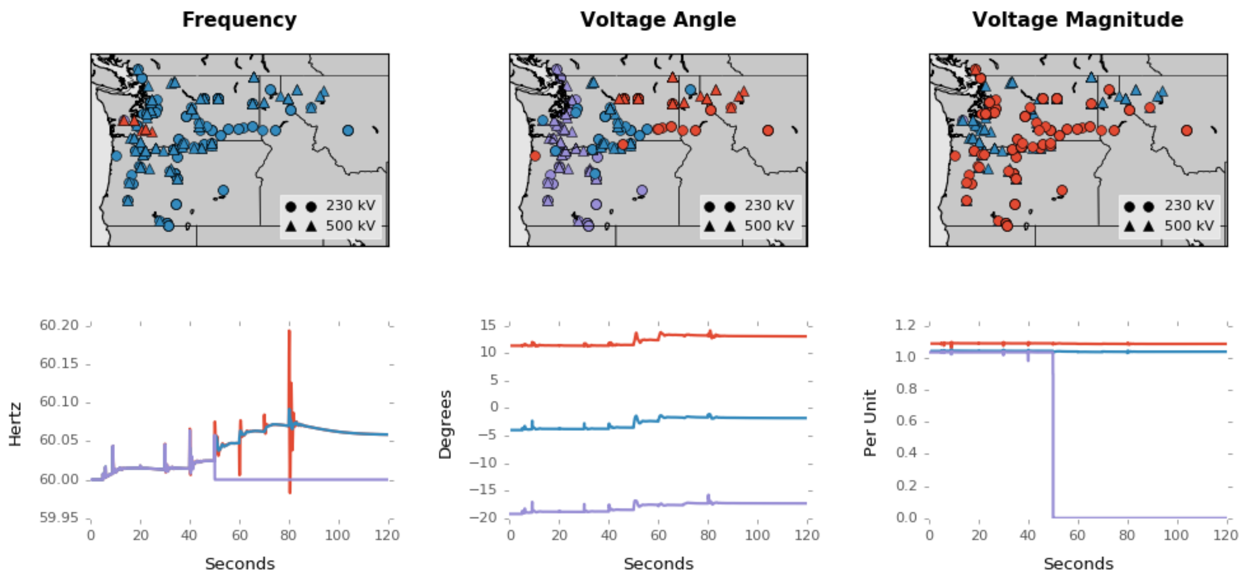


Figure A.30: Analytics tool interactive visualization pane for geographic searching.

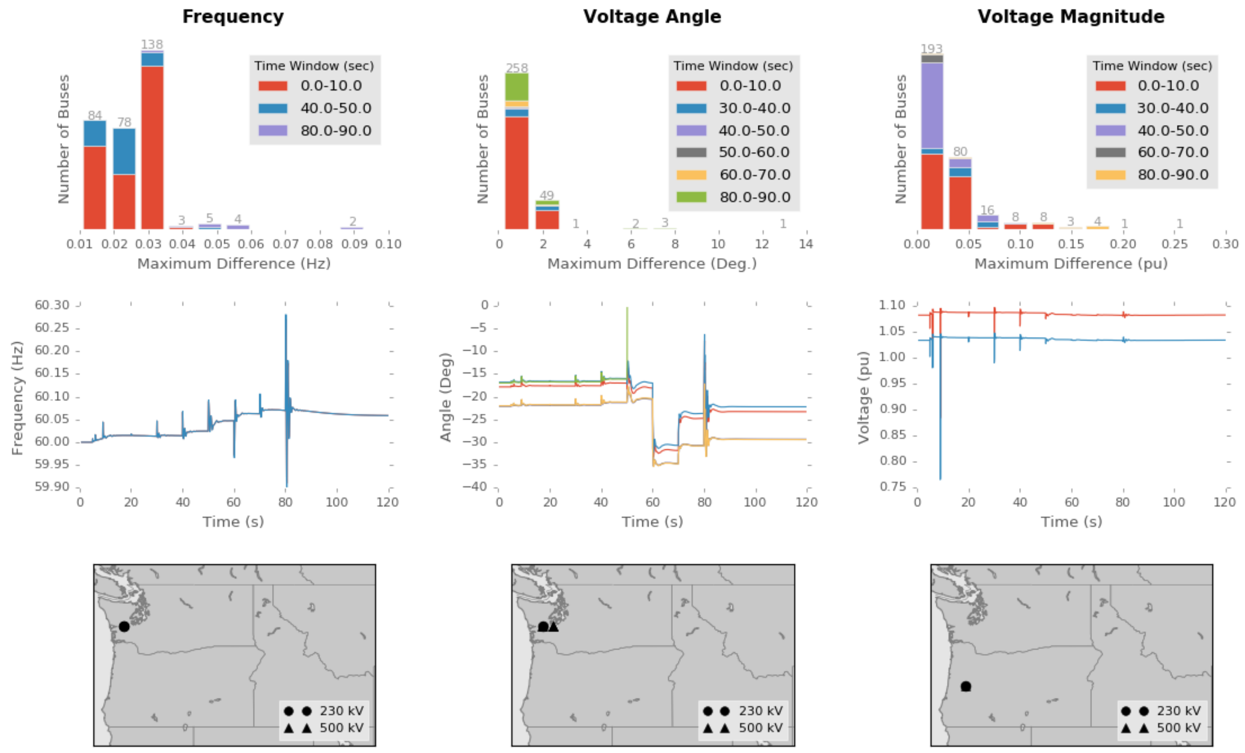


Figure A.31: Analytics tool interactive visualization pane for searching by greatest measurement variation.

Within the interactive visualization window, shown in Figure A.30, the user can draw a rectangle over an area in the geographic map to see specific line plots for the buses selected by the user. To draw a rectangle, click and hold the left mouse button and drag it to the final point you want and release. A rectangle will not be shown, but know that the program is working. Within a second or so, the line plots will appear in the last row of the visualization.

Within the interactive window, shown in Figure A.31, the user can select which bin of a histogram they want to investigate by clicking anywhere on the histogram plot, as long as it is within the x values of the bin. When they do so, the line plots and geographic plots will be updated to show the data for only the buses that are contained in the user selected bin.

To quit the program, simply press the red “X” button, at the top right corner of the application window.