Visual Analysis of PMU Data Project Report

Reet Chatterjee, Ziming Dong, Venkat Charan Guduru, Neel Jepaliya, Cody Schierbeck, Vivek Tiwary

***Abstract*—PMUs, or phasor measurement units, are sensors that are placed in the power grid network, which measure various attributes associated with electricity (voltage, frequency, etc.). PMUs provide very fast measurements (usually at least 30 measurements per second) and the data is highly noisy, which makes analysis a non-trivial task. PMU data is therefore time series data that is linked through the network structure of the power grid network. When events occur in the power grid such as a blackout or oscillation, the effects of the event can propagate through the network and be sensed by PMUs.This project develops a visual analytics interface to analyze the PMU dataset, with an emphasis on analyzing bus data, events and identifying anomalous behavior using time and frequency domain data.**

I. Introduction

A. Functionality of System

Our project aims to develop a visualization tool to analyze PMU datas and find events of interest using time and frequency domain dataset. The major functionalities of the system are as follows:

1) PMUs and its neighbors network map
2) Zoom and view buses based on selected region from the dropdown.
3) Click and analyze time-series and frequency domain data of a bus and its neighbors

B. Background

A PMU, or Phasor Measurement Unit, is a device used to estimate in real-time the magnitude and phase angle of electrical signals. They can collect measurements at up to 120 measurements per second which is an improvement over traditional SCADA devices. This lends to PMUs being an excellent tool for electrical engineers to analyze electrical grid systems. Engineers can utilize PMUs to obtain overviews of both a local scope as well as a global status, allowing them to safely, accuratly, and rapidly monitor the state of a system.

For a visualization system to be be useful for our target users, it must be able to plot signals against frequencies, query and visualize data for a given substation, as well as data for neighboring substations. Time-variant data needs to be processed in such a way that allows for frequency measures to be plotted against amplitude values.

C. Team Members Contributions

The project task can be split into several parts, each team member has same weight contribution.
Reet Chatterjee: Database Schema Design and Development. Database Migration to Cloud. Flask REST based API development. Bus Details UI fixes. Sanity Testing.
Ziming Dong: Data pre-processing. Set up flask server. Buses detail interface page design, including time series charts, and FFT vs Frequency charts.
Venkat Charan Guduru: Data pre-processing, development of Flask REST API, Anomaly detection.
Neel Jepaliya: Data formatting and linking, database schema design, map user interface.
Cody Schierbeck: Interactive neighbor-network map and Flask.
Vivek Tiwary: Setup & configured Google Cloud Platform for Cloud SQL DB and App Engine usage for development. Flask server setup. User Interface design. Bus Details page UI fixes, Integrated flask server API calls on the client side using async fetch calls. Developed REST API.

II. Data Preprocessing

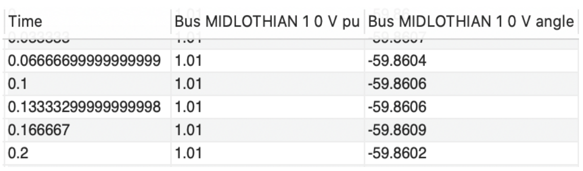
Data Preprocessing in this project was done with using Python. The primary purpose of data preprocessing in this project is to transform raw PMU data into network data in temporal and frequency domain. There are some difficulties for preprocessing dataset, we go through the research paper [1] to generate the solution. We divide the work into multiple smaller tasks to achieve our goal. The tasks are: 1) Divide the data into voltage, angle and frequency data, 2) Calculate phasor data, 3) Convert phasor data into frequency domain. To complete the first task, we filter the data based on the headers of the file. Regular Expressions along with Pandas library are used to perform this task.We use the linear algebra functionality of the Numpy library to calculate the phasor representations of the data. The following formula is used to compelete this task:

$$Va = pu * \cos(A + \delta)$$

where pu is the per unit voltage, A in the angle and δ is the phase difference. At the end of this task, we will be have the three phasor representations of the PMU data i.e. Va, Vb and Vc. Finally,each phasor representation will be converted into frequency domain. Numpy library is used to perform this task as well. Along with the phasor representation, this task requires the start and end timestamp of the required subset of data.

III. Data Storage

The data is obtained from the Synthetic PMU Data website https://electricgrids.engr.tamu.edu/electric-grid-test-cases/synthetic-pmu-data/ and can be classified into two types: i) Raw Sensor values data, ii) Network data. The sensor values data consists of Voltage, Angle, and Frequency values for each timestamp. The scope of the data is limited to the state of Texas but has been divided into several parts according to the location of the buses. To make queries according to the parameters passed by the user, each region data has been further broken down into several components so that the size of each component satisfies the constraint of the database.



Time	Bus MIDLOTHIAN 1 0 V pu	Bus MIDLOTHIAN 1 0 V angle
0.033333	1.01	-59.8607
0.06666699999999999	1.01	-59.8604
0.1	1.01	-59.8606
0.13333299999999998	1.01	-59.8606
0.166667	1.01	-59.8609
0.2	1.01	-59.8602

Fig. 1. HLD of the system

Due to the column number and row size constraints of MySQL databases, each region data has been further broken down into parts before transferring them into a table in the database. The number of parts each region dataset is broken down into depends on the length of the columns or the number of buses presented in the file of the region. Some regions like "Coast", having approximately 2000 columns/650 buses information, have been divided into 10 parts with respect to the columns so that each division consists of 200 columns. These divisions are numbered and named by appending the number to the name of the region. To determine the division that a certain bus belongs to, a mapping is maintained with the first value of each record having the name of the bus and the second value indicating the sub-division the bus belongs to. Each sub-division of a region dataset is stored in separate tables in the database. The first column of each such table indicates the timestamp and the subsequent columns have each bus's Voltage, Angle, and Frequency values. Hence the values of each row in the tables represent the respective bus's Voltage, angle, and Frequency values of a certain timestamp. The sampling of these values is done every 0.033 seconds by the sensors. Hence, the difference in the timestamp between adjacent rows in each table is 0.033. For making the queries efficient, the mapping table is used to retrieve the table name a particular bus belongs to, and the returned table is queried according to the parameters, such as timestamps, that are provided by the users. MYSQL database engine hosted on Google Cloud Platform is used to deploy the data onto the cloud for making queries through the APIs.

IV. Challenges and Limitations for visualizing PMU Data

1) Uploading and fetching PMU data to database: We handled this in our project by uploading data to Cloud SQL DB having configurable amount of storage.
2) Retrieving PMU data from databases quickly: We were able to fetch data from Cloud DB within a reasonable amount of time. It is not the quickest of response due to the amount rows being queried for processing raw data. This is a limitation of the system.
3) Plotting all PMUs and creating links on the User Interface: This has been handled by generating a map of all the PMU data using their x, y coordinates. However, the limitation is on the zoom event as there are many points to connect causing latency and unresponsiveness. Hence, zoom feature is handled in a specific way as discussed under interface section.
4) Visualizing time domain and frequency domain data over long period of time quickly: The FFT transformation is done on the flask server by fetching all rows of the data within the selected time-window. This affects the response time of this service as a lot of data is being handled simultaneously. This is a limitation which can be handled with a better hardware compute configuration.
5) Linking events/anomalous behaviors to visualized data: The huge amount of data for a small period of time makes it very difficult to visualize all data simultaneously and finding events of interest. We handled this by providing user with the ability to analyze time-series voltage data and FFT data of a bus and its neighboring buses effectively. The limitation is on the time period for which this can be visualized as hardware configuration needed to support all timestamps is not feasible due to our limited resources.
6) Handling and analyzing data in real-time: Our application doesn't support real-time data handling due to the amount of data that will be pushed onto our system which will require a much better infrastructure in place. But this feature can be extended on top of our current tool.

V. Data Management in Backend

A. Flask Server Set Up

The backend of the visualization platform has been handled using the light-weight **flask** framework. The flask is a micro-framework which is written in python and suitable for hosting server side code of a web application. Our flask server will listen to http requests to end-points exposed via REST API and return http response accordingly. The Flask server was setup on our local machine and Google App Engine importing flask framework in our main python file. The Google App Engine needed a separate app.yaml and requirements.txt file to deploy the flask server on cloud. Due to the processing time taken by our FFT API, we are currently hosting flask server on localhost as Google App Engine kills the process taking too much of memory and time to complete a request.

B. Backend design

The architecture of our system followed **Model-View-Controller** pattern. The REST end-points were created on the main file which acted as a controller layer. The Google cloud MySQL DB was utilized as a Model layer and we used a separate data access layer in python to fetch the PMU data from database. The data visualization page acted as the view layer.

A **high level design** of our overall system is shown in fig 2. The service end-points were exposed to the front-end application via REST protocol. The services then made queries to the Cloud database and applied processing algorithm on the fetched data to generate meaningful data. This processed data was then sent to the fronted application where it was visualised by the end-user to find patterns and generate insights from it. Our application followed a modular structure segregating database, server and front-end from each other. The beauty of this modular design is that instances can be churned up to scale accordingly for applications that are experiencing heavy traffic. The url pattern of our API end-points followed the structure of **api/version/servicename** where api is the module name, version denoting the version of API and servicename is the name of REST service for handling a particular feature. In our system, we have just used ’api’ as the module name due to the presence of just one module. The server reads requests and sends responses in JSON format using http.

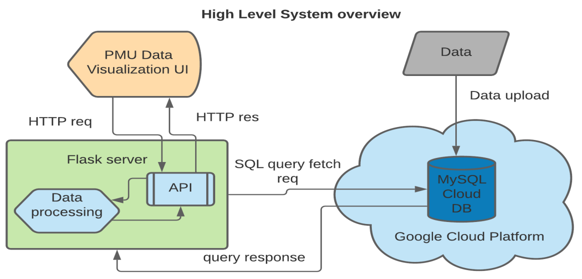


Fig. 2. HLD of the system

VI. Interface & Case study

A. Map interface Features

The map interface serves the dual purpose of landing page and point of control for visual analysis of PMU data in the synthetic bus network of Texas. Upon application initiation, all users will be routed to the map interface. There, the users can interact with the map to select a bus they would like to analyze, which routes them to the bus detail interface.

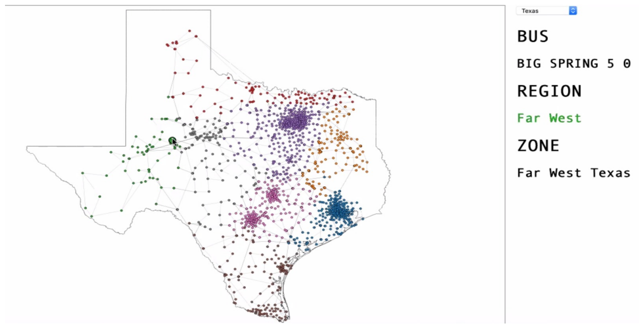


Fig. 3. Bus Map Interface Page

The map interface layout and interactions are focused on simplicity. The buses and lines are represented with point and line marks respectively. Additionally, the bus point marks use the color channel to represent the region the bus is associated with along with a black outline to assist in differentiating different buses, especially in crowded sections of the bus network. The lines are represented with near-transparent line marks between nodes in order to avoid visual overload and a general loss of discernible structure within the synthetic network due to overcrowding.

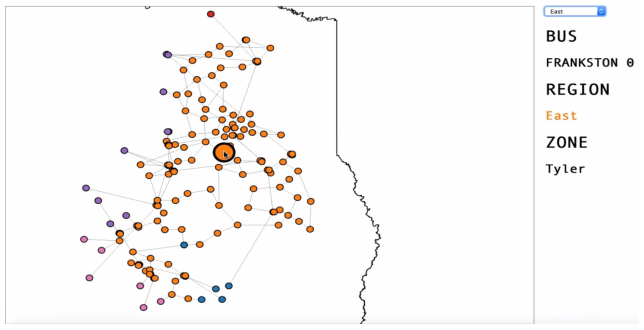
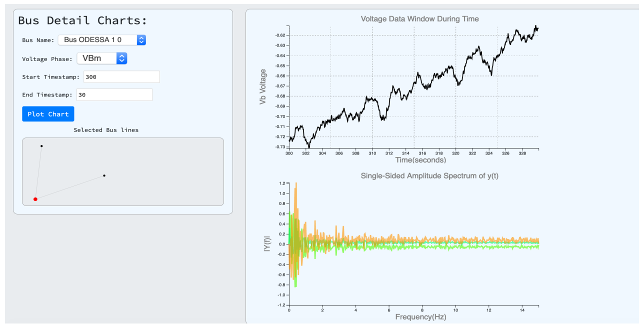


Fig. 4. Bus Zoom Interface

Hovering over a certain bus increases the point mark size and displays the bus’s name, region and zone on the side. The map display can be further filtered and zoomed by selecting a region from the drop-down menu. This feature provides no panning functionality to the user and makes use of a static scale and translate once the buses are filtered to the selected region and its direct connections to buses in neighboring regions. The filtered zones are pre-computed in order to save execution time when frequently changing focus between regions.



Bus Detail Charts:

Bus Name: Bus ODESSA 1 0

Voltage Phase: VBm

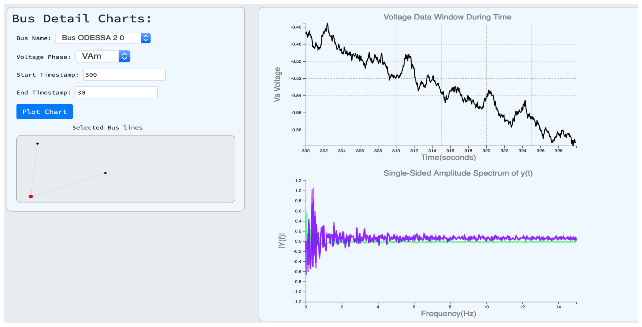
Start Timestamp: 300

End Timestamp: 30

Plot Chart
Selected Bus lines
Voltage Data Window During Time
Vb Voltage
Time(seconds)
Single-Sided Amplitude Spectrum of y(t)
|Y(f)|
Frequency(Hz)

Fig. 5. Case study peak detection 1

The decision to remove user-controlled panning was made due to the map display being rendered using a SVG (scalable vector graphic), which does not perform well with more than 4000 individual elements that need to be scaled and translated. This would lead to significant stutters in user experience and needless complexity due to the map scaling and translating to contain all points possible for each region filter. The decision to only preserve direct connections was made due to the sheer density of bus nodes on the map display, which would often distract the user from the region they are trying to focus on. Therefore, preserving direct connections still shows the regional network while still preserving information on inter-regional connections in the synthetic bus network of Texas.


Bus Detail Charts:

Bus Name: Bus ODESSA 2 0

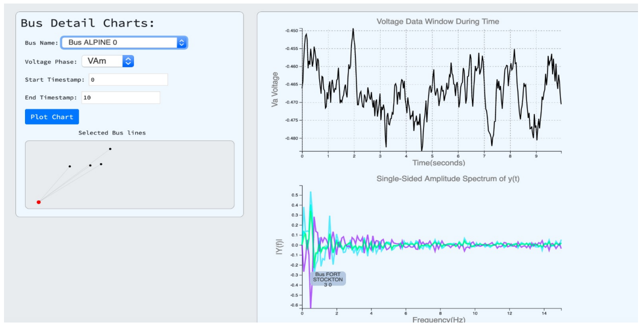
Voltage Phase: VAm

Start Timestamp: 300

End Timestamp: 30

Plot Chart
Selected Bus lines
Voltage Data Window During Time
Va Voltage
Time(seconds)
Single-Sided Amplitude Spectrum of y(t)
|Y(f)|
Frequency(Hz)

Fig. 6. Case study peak detection 2


Bus Detail Charts:

Bus Name: Bus ALPINE 0

Voltage Phase: VAm

Start Timestamp: 0

End Timestamp: 10

Fig. 7. Bus Detail Interface Page

B. Bus Detail Interface Features

Before working on the user interface design, some research paper helped the team get a better understanding of PMU domain knowledge and main goal of PMU data analysis system [2] [3]. The bus detail interface page can be divided into two parts. In the left side container, there are some parameter input control buttons and input boxes, when frontend receive the response JSON file, the bus name drop-down selection will be created automatically based on number of buses in the JSON file. Then the voltage phase drop-down selection contains VAM, VBm, and VCm options. Timestamp input box shows below the voltage phase selection, user can control the timestamp parameter to plot the time series chart and FFT vs frequency chart by click the ”Plot Chart” button. The small container shows under the click button, it contains a partion of select PMU line as graph network, in the graph, the nodes represent buses in the response file. When user click selects a particular node, the size of the selected node will increase. On the right side of the page, there will be two plotting charts,they are plotted by D3 library https://d3js.org/ in JavaScript, D3 is the powerful data-driven library to help building visualizations. In the chart container, the top one is the time series chart which represent the Voltage vs Time seconds variation. It will be hard to find abnormal event in this time series chart. Below the time series chart, the phasor data will be represented in frequency domain. Power systems engineers primarily require data in frequency domain to find anomalies. In 8, We can find the purple value line has sharp peak value at 0.33 frequency, it means the abnormal behavior happened in the Bus FORT STOCKTON 30. The bus detail interface page shows in the figure 7, the abnormal event detection finding shows in the figure 8.

C. Finding Anomalies

There are multiple methods of finding anomalies with emphasis on different statistics in each. We are using a Gaussian based method to find unusual peaks in data. In this method, we assume that the frequency representation of the data is in the form of a Gaussian distribution. Using the mean and standard deviation of data, we can the find the probability density function on any value. We use the Scipy library to find values which have a probability of less than 0.1. To verify this method, we have selected the value having the lowest probability in a particular PMU's data record. As can be seen in 5 and 6, We found that the particular bus as well as its neighboring buses experienced a peak at that timestamp. This proves that the method of finding anomalies is valid and helpful.

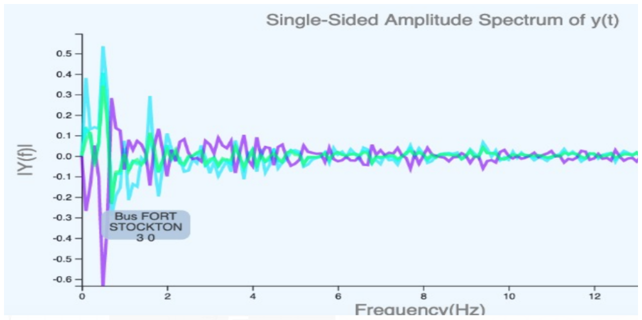


Fig. 8. Bus Abnormal Event

VII. Discussion

The current feature we have in the PMU data analysis system is the map landing page and bus detail interface page, we can fetch the data from database easily and get response JSON file to plot the line chart, users can use parameter controller to set up parameters and plot the line charts. The current problem will be the long loading time when fetching the data from database, for example, if we input long period of timestamp, the system take a while to loading the data, user will wait for a long time until frontend receive the response JSON data. Thus, we will be working on accelerating the response speed of fetching data in the future. Furthermore, there can be more features in the FFT vs Frequency line chart, we can draw the valuelines for different PMU lines, then we can compare the abnormal event between different PMU lines, there should be some new findings.

VIII. Conclusion

In this project we aim to provide a simple interface for Analysts to analyse anomalies on Electric GRID Power Systems. The project involves the use of PMU sensor data for detecting events in the grids. The tool implemented can help target-users to identify and isolate points of interests that is not possible without an abstract representation of the data through visual plots. The interface offers several features like Map View, Detailed Visual Dashboards and parameterized search queries that provide a platform for the users to inspect areas of concerns efficiently. To support the Interfaces back-end APIs are created that handle the client requests and retrieves the required data from the Database. The APIs also enables execution of pre-processing functions on the retrieved data before passing the values to the front-end. The modular separation of the Front-end allowed the teams to work on dependant tasks from remote environments. Although all the objective of the projects have been successfully implemented, some room of improvements exists in term of efficiency. The pre-processing and storing of FFT values for each bus in the database can allow the Bus Details page to be more responsive to user queries and can be considered as a direction of extension for the project.

References

[1] Bo Yang, June Yamazaki, Nao Saito, Yutaka Kokai, and Da Xie. Big data analytic empowered grid applications—is pmu a big data issue? In 2015 12th International Conference on the European Energy Market (EEM), pages 1–4. IEEE, 2015.

[2] Jesmin Khan, Sharif Bhuiyan, Gregory Murphy, and Johnathan Williams. Pmu data analysis in smart grid using wpd. In 2014 IEEE PES T&D Conference and Exposition, pages 1–5. IEEE, 2014.

[3] Sharif MA Bhuiyan, Jesmin F Khan, and Gregory V Murphy. Big data analysis of the electric power pmu data from smart grid. In SoutheastCon 2017, pages 1–5. IEEE, 2017.