FINAL REPORT

SYNCHROPHASOR BASED TRACKING THREE-PHASE STATE ESTIMATOR AND ITS APPLICATIONS

DOE Award Number: DE-OE0000118

Name of Recipient: Virginia Tech, Blacksburg, Virginia.

Principal Investigator: A.G. Phadke

Project Team, Virginia Tech: A.G. Phadke, James Thorp, Virgilio Centeno

Dominion Virginia Power: Matthew Gardner

Quanta Technology: Damir Novosel, Yi Hu, David Elizondo

Date: August 31, 2013

1.0 EXECUTIVE SUMMARY

Electric power infrastructure is one of the critical resources of the nation. Its reliability in the face of natural or man-made catastrophes is of paramount importance for the economic and public health wellbeing of a modern society. Maintaining high levels of security for the high voltage transmission back bone of the electric supply network is a task requiring access to modern monitoring tools. These tools have been made particularly effective with the advent of synchronized phasor measurement units (PMUs) which became available in late 1990s, and have now become an indispensable for optimal monitoring, protection and control of the power grid.

The present project was launched with an objective of demonstrating the value of the Wide Area Measurement System (WAMS) using PMUs and its applications on the Dominion Virginia Power High Voltage transmission grid. Virginia Tech is the birth place of PMUs, and was chosen to be the Principal Investigator of this project. In addition to Dominion Virginia Power, Quanta Technology of Raleigh, NC was selected to be co-Principal Investigators of this project.

The project has been divided into 6 parts. The rationale for each part and goals achieved after its completion are described next:

(1) **Installation of a Wide Area Measurement System on the Dominion Virginia Power (DVP) High Voltage (HV) transmission grid.** During this phase, PMUs have been installed at all the HV substations and individual phase voltages at the network buses as well as individual phase currents in all the connected feeders are being monitored. The measurements are transferred to the DVP control center, where applications of these measurements are installed for real-time use by the DVP engineers. The entire WAMS architecture has been checked out, and is functioning as designed. Measurements from the substations are received at the control center at the rate of 30 times per second. A communication link between the DVP control center and the Pennsylvania-Jersey-Maryland (PJM) Independent System Operator (ISO) has been set up and selected measurements from the DVP system are being transmitted to the PJM facility.

(2) **Development and deployment of a three-phase state estimator on the DVP system.** This is unique feature of the current project. A state estimator provides accurate real-time picture of the voltages and currents on the entire network. Because of the unbalance of currents between the three-phases of the DVP system, it became necessary to develop a three-phase state estimator. The existing state-of-the-art in this field is to assume balanced current flows, and create an estimate of the balanced voltages and currents on the network. For the DVP system, this is clearly in error. Of particular concern is the effect of unbalanced current flows on generators of the system which can cause overheating of the generator rotors. The three-phase estimator has been developed and installed on the DVP system, and is providing state estimates at the rate of 30 times per second.

(3) **Unbalance monitoring on the DVP system.** This application has been developed and is operational. It uses real-time data to determine the distribution of negative sequence current (which is a measure of the degree of unbalance). From this unbalance monitoring system it becomes possible to track negative sequence currents in the generators, and provide an alarm when a threshold (set by system operators) is reached. If excessive unbalance currents in the generators are detected, the operator has the ability to respond to the alarm and take corrective measures to bring the generator negative sequence currents to acceptable level.

(4) **Calibration of the current and voltage transformers**. All measurements in a power system are made with reduced level signals provided by current and voltage transformers. These instrument transformers are reasonably accurate, but for improved accuracy of measurements they need to be calibrated. The technique for calibrating these transformers is based on using the output of one precisely calibrated three-phase voltage transformer at one of the buses. This method was developed and checked out by simulations, and has been implemented in the DVP WAMS system.

(5) **Intelligent islanding system**. Upon occasion, the power system breaks up into islands which continue to function in a somewhat degraded fashion, and it is incumbent upon system operators to re-synchronize the islanded system to the rest of the transmission grid. One of the problems facing the operators is not knowing that an island has actually formed. In particular, this is the case when the island is on the lower voltage system which is not being monitored by the WAMS system. A technique has been developed and implemented on the DVP system which detects the formation of an island on the low voltage system by utilizing measurements made on the HV system.

(6) **Visualization system**. The measurements provided by the WAMS system and the results of the application functions described in (2)-(5) above must be displayed to the system operators for them to make use of the knowledge gained from the applications. This is a critical part of the entire project, since the raw data gained from the WAMS system is of no direct use to the operator. The displays generated by the visualization system place engineering data in appropriate units on a geographical representation of the DVP system, and as appropriate the generated alarms are also presented to the operator with proper characterization of the urgency of the alarm. This system has been developed for the present project, and has been installed on the DVP system.

2.0 TABLE OF CONTENTS

3.0 ACTUAL ACCOMPLISHMENTS AND PROJECT GOALS

3.1 Dominion Virginia Power

Table 1: Planned vs. Actual Substation Activities

Planned PMU Installations:	Actual PMU Installations:
North Anna Generating Station (2010)	North Anna – PMU/PDC[1] panels installation in both 500 & 230kV Houses done. PMU quantities all tied in except 575 line voltages and currents. Outage for 575 line submitted and accomplished in Q2 2013.
Ladysmith Generating Station (2010)	Surry – PMU/PDC panels installed both 500 & 230 kV Houses. PMU quantities all tied in. Field construction reported complete for both houses Q3 2011.
Carson (2011)	Valley – PMUs & PDC installed. PMU quantities all tied in. Field construction reported complete Q4 2011.
Bristers (2011)	Possum Point 500 – PMUs & PDC installed in Q4 2012.
Clover (2011)	Chesapeake Energy Center (CEC) - PMUs & PDC installed. PMU quantities all tied in. Field construction reported complete Q4 2011.
Fentress (2012)	Suffolk – PMUs & PDC installed. PMU quantities all tied in. Field construction reported complete Q4 2011.
Cunningham (2012)	Midlothian - PMUs & PDC installed. PMU quantities all tied in. Field construction reported complete Q2 2012.
Chancellor (2012)	Chancellor – PDC scheduled to be installed and completion planned for Q4 2012.
Septa (2012)	Septa – PMUs & PDC installed. PMU quantities all tied in. Field construction reported complete Q2 2012.

Table 2: Planned vs. Actual IT Activities

Planned IT Activities:	Actual IT Activities:
Purchase Synchrophasor Vector Processor (2010)	Purchased Server Hardware (2010)
Purchase Database Software(2011)	Established basic data flow to the R&D server (2012)
Purchase Server Hardware (2011)	Have a running linear state estimator on the R&D node (2012)
Purchase Phasor Data Visualization Software (2012)	Have state estimator output going back to the central PDC for archival (2012)
	All code and documentation completed and placed it in the IT repository (2012)

[1] PDC – Phasor Data Concentrator or Synchrophasor Data Concentrator

	The State Estimator data is flowing to the Historian (2012)
	Completed all phasor data validation of the currently installed PMUs (2012)
	Tested preliminary version of Quanta visualization software (2012)

3.2 Virginia Tech

Table 3: Planned vs. Actual State Estimation Activities

Planned Activities	**Actual Activities**
Phase I: Initial development of the state estimator, calibration, unbalance monitoring, and intelligent islanding detection algorithms in Matlab	Three-phase unbalanced data set developed from simulated data and approximated network impedances in the absence of real phasor data and impedance data.
	State estimator algorithms developed and tested successfully. Out of necessity, the topology processor algorithms also developed and tested.
	Instrument transformer calibration algorithms developed and tested successfully.
	Islanding detection algorithms developed and tested successfully.
	Algorithms were tested using the real network impedances once available but real phasor data was not available during this phase of the project.
Phase II: Migration of the Matlab algorithms to the openPDC platform; translation into C# language.	Learning development for the openPDC platform. Gaining proficiency with openPDC and the C# language.
	Direct translation of algorithms into C# language
	Algorithms modeled to fit the design patterns of the openPDC adapters. Each algorithm still dependent on individual data sets for testing. Algorithms modular do not interact yet.
	Successful testing of each of the applications in the openPDC platform against the simulated data set.
Phase III: Final implementation of algorithms on	Successful merging and testing of the algorithms

site on Dominion's hardware using streaming synchrophasor data from the newly installed PMUs.	under a unified data set.
	Restructuring of the algorithms to follow better object oriented design practices.
	Installation of state estimator, topology processor, negative sequence current monitor on Dominion hardware with real synchrophasor data.
	Development of the Network Model Editor GUI
	Thorough documentation of algorithms, workflow, user manuals, compiled help files, etc.
	Addition of new features to state estimator for a more robust design. (see section 4.2)
	Upon completion of PMU installation, the installation of the instrument transformer calibration algorithm and islanding detection algorithm on Dominion hardware with real synchrophasor data.

Table 4: Planned vs. Actual Calibration Tasks

Objectives:	Actual Accomplishments:
Get the diagrams of system with PMU installations identified.	After determining the locations of PMU, obtained the system diagram for the project.
Get the system topology information with line and device parameters.	DVP provided system topology, PSS/E savecase and unbalanced line parameters generated by ASPEN.
Get PMU measurements in 500kV system from DVP	Simulate PMU measurements in 500kV system by MATLAB code.
Develop the algorithm of three-phase instrument transformer calibration, and determine the requirements for the algorithm	The algorithm is to calibrate all the instrument transformers in power system with at least one set of pre-calibrated voltage transformer.
Program three-phase instrument transformer calibration algorithm	Develop the calibration algorithm code and run the MATLAB code on DVP system with simulated PMU measurements.
Check the calibration results	Compare the ratio correction factors of instrument transformers calculated by the code with the real ratio correction factors. The differences are close to zero.

Table 5: Planned vs. Actual Unbalance Monitoring Tasks

Objectives:	Actual Accomplishments:
Get the diagrams of system relative to the project, with PMU installation marked.	Determine the optimal location of PMUs.
Get PMU measurements.	Simulate PMU measurement in relative system by three-phase state estimation or power flow codes.
Get and monitor the negative current flow in the system	Change the three-phase power flow into three-sequence power flow and monitor the negative current.
Set the threshold of negative current flowing into generators.	Monitor the negative current flowing into generators in DVP 500kV system.

Table 6: Planned vs. Actual Substation Activities in Intelligent Islanding

Objectives:	Actual PMU Islanding Studies:
Develop an islanding scheme to detect and identify islanding contingencies in Dominion system with the usage of synchrophasor measurements	Created an islanding database using study of historical events, simulation results and Dominion system characteristics. Developed an islanding detection and identification scheme using decision tree algorithm. The tree is capable of detecting and verifying islanding scenarios. Estimate islanding effective area and stability. Developed an Dominion online Islanding detection, classification, and severity estimation module utilizing PMU data; and tested it with real PMU measurements and simulated cases.
Study DVP system islanding characteristics; develop potential islanding detection scheme improvements in future applications.	Apply analytics on the Dominion islanding simulation results and DT module, including DT size and attributes, DT variables and Dominion PMU installation policies, PMU contributions from inside the island, the influences to the DTs with variations in system models, and assessment of the Fisher's method in DT operation.

3.3 Quanta Technology

Quanta Technology goals and objectives in the project are summarized by the development of software code for the visualization of the PMU applications developed by Virginia Tech. The following table provides a comparison of the actual accomplishments with the goals and objectives of the project.

Table 7: Planned vs. actual PMU application visualization activities.

Objectives	Actual Activities
Phase 1. Analytical Studies QT participation in various tasks of Phase 1. Subtask – Participate in Management and Planning meetings and review results for Study System Data Base, Tracking Three-phase State Estimator, and Tracking Three-phase State Estimator Applications.	As planned. No deviations
Three-phase Estimator Display Development. SubTask - Requirements and Design, Interface from 3 Phase State Estimator to Display Applications, Three-phase Estimator System Display, Three-phase Estimator System Display, SubTask - Archive of Estimator Calculation Results and Trending Display.	Dominion geographical display includes a geographical display of Dominion's main bulk transmission system with real time monitoring of operations variables: - Directly measured from the field: a) Voltage Magnitude (three phases) and b) frequency. - Results of Three-phase State Estimation PMU application: a) Voltage Magnitude (three phases) and b) frequency. - Directly measured from the field: Current magnitude, negative to positive sequence ratio (I2/I1 ratio) and current magnitude, zero to positive sequence ratio (I0/I1 ratio). **Deviations:** Three-phase state estimator displays were developed internally by Dominion. Trending displays are commercial tools bought by Dominion based on Quanta Technology demonstration tool.
Phase 2. Prototype Demonstration PMU/PDC/Server System Integration and Architecture. Subtask-- Functional, performance, and other technical requirements for PMU, PDC, application server, and the data interfaces among them. Subtask - Technical and a physical architecture for the integrated PMU/PDC/Application Server system	As planned. No deviations

Phase 3. Full Scale Demonstration PMU/PDC/Server System Verification. Subtask-Test all displays functionality with real time data in the OpenPDC platform. Site acceptance tests at control center.	Delivery of final software and installation performed in October 2013.

4.0 APPENDIX I: PROJECT ACTIVITIES

4.1 Dominion Virginia Power

Phase I, 1st Quarter Activities:

Legal Actions

The company established Critical Energy Infrastructure Information (CEII) non-disclosure agreements (NDAs) between all parties involved in the grant that may require access to the company's infrastructure information.

Data Sharing

The company designed an appropriate "Documentum eRoom" for the exchange of CEII between all parties in the grant. An "eRoom" is an extranet environment (from the company's perspective) that allows for collaboration between internal and external users. It employs authentication and authorization so that information protection and granular access rights management can be achieved. The company had (and still has) several "eRooms" used for various legal matters. External users connect via SSL (HTTPS) to ensure secure collaboration over the internet. The company's existing infrastructure supported this project with no additional hardware or license costs.

Substation Activity Scoping

The company executed a scoping process for both PMU locations scheduled for installation in 2010. The scoping process included a meeting between key project management, engineering, and technical consulting individuals. The initial meeting focused on which currents and voltages within each of the stations can be practically monitored by the SEL 487E. The following constraints were noted at the time of scoping:

1. One panel per substation to be devoted to synchrophasor measurement
2. Each panel can accommodate two SEL 487Es.
3. Each SEL 487E has a compliment of 18 (six three-phase) currents.
4. Each SEL 487E has a compliment of six (two three-phase) voltages.
5. While multiple digital inputs are available on each SEL 487E a concern exists as to the availability of all requested breaker status signals.
6. In many cases, only one panel per substation could be devoted to synchrophasor measurement.
7. In every case, only two PMUs can be accommodated per panel. Additional PMUs in a panel are prohibited by a lack of terminal block space.

Phasor Data Concentrator (PDC) Design, PMU Education, and Project Collaboration

SEL visited the company on January 25, 2010 and held a seminar on the programming of SEL PMUs and data concentrators. Quanta Technology visited the Company on January 28 to discuss PDC architecture and design as well as to begin the process of PMU and state estimator visualization.

Phase I, 2nd Quarter Activities:

Modeling

Dominion continued to gather information on its system model for the project partners. All parties signed a critical energy infrastructure information non-disclosure agreement. Dominion is cooperating with project partners to identify possible islanding risks and scenarios.

Equipment

A substation PDC was determined to be a necessary component of the synchrophasor data architecture. The appropriate solution was selected. A preliminary effort to conduct end-to-end testing in a laboratory environment for both the PMUs and PDCs occurred May 2010.

Collaboration

Dominion established bi-weekly conference call comprising key personnel at Quanta Technology (Dr. David Elizondo, Dr. Yi Hu, and Mr. Don Morrow) and key individuals within Dominion (Dr. Matthew Gardner, Mr. Tom Reitz, Mr. Martin Quinlan, Dr. Tao Xia, Mr. Mike Kalichman, and Mr. Gus Johnson). The biweekly calls aided in the execution of project tasks, especially those tasks surrounding power system visualization and power system data concentration.

Phase I, 3rd Quarter Activities:

Equipment

Dominion identified multiple electric transmission substations containing metering accuracy three-phase CCVTs and one station with accessible three-phase PTs. These stations were scheduled to be instrumented with PMUs for calibration purposes. Metering accuracy CCVTs are assumed to have 1% error bounds.

Task Update

Dominion committed to installing PMUs at two stations in calendar year 2010 as a cost-sharing effort for this project.

Central to the timing of Dominion's synchrophasor installations was the availability of synchrophasor data concentration equipment. A preliminary effort to create procedures for and conduct basic testing in a laboratory environment for both the PMUs and PDCs was undertaken.

It came to Dominion's attention that a central PDC was necessary before obtaining the planned synchrophasor vector processor. Therefore, Dominion proposed delaying the purchase of the synchrophasor vector processor until 2011 and advancing the procurement of central PDC hardware and software into 2010. This change in plans did not change the overall amount of project funding from the Department of Energy

Collaboration

On May 25, 2010, a technical workshop between team members was held at Virginia Tech. The focus of this workshop was the central data concentration and display requirements. Dominion continued to work directly with Quanta on display development.

Phase I, 4th Quarter Activities:

Equipment

Figure 1 shows the basic parts and pieces of Dominion's EMS system. We evaluated the different methods of interfacing a central PDC with the existing EMS structure. One option was a direct interface using ALSTOM's proprietary "ISD" communications protocol. Another option was treating the PDC as another RTU and use an open DNP protocol.

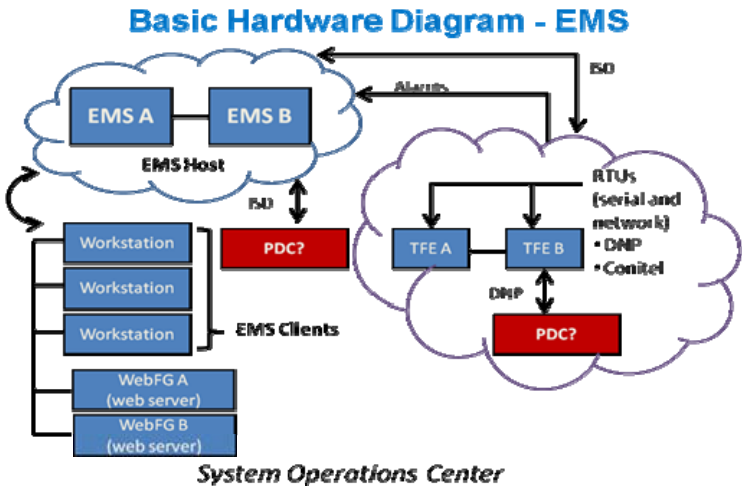


Figure 1: Dominion EMS and PDC considerations.

Dominion worked to better understand system operators might interface with synchrophasor data – see Figure 2. System operators could interface with the synchrophasor data through either a built-in layer that could be coded into the existing EMS software. Another option which could stand alone or exist in tandem with a built-in application layer is the use of a web server that could run on the central PDC server. This option would allow for the creation of any number of "web-based" options. Direct linking between an in-house central PDC-based web service and Dominion's EMS was also considered as a possibility.

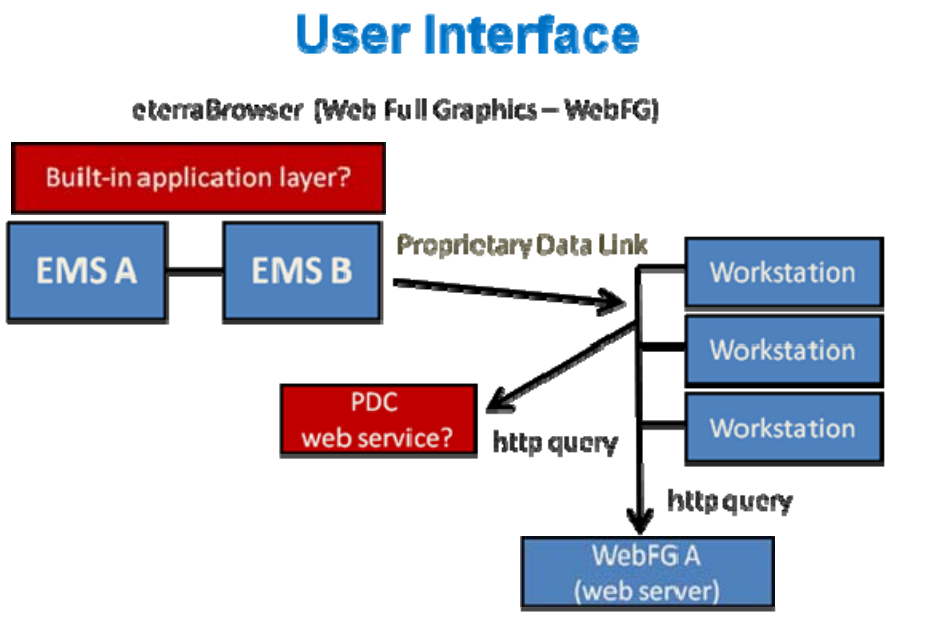


Figure 2: Dominion's user interface possibilities.

Further Dominion worked to develop a high-level functional understanding of each of the components that would be used in its central synchrophasor data concentration architecture. Figure 3 presents a relationship of currently commercially available software and hardware (items highlighted in red are pertinent to this DOE project). Figure 3 leaves functionality gaps that are addressed in the diagram of Figure 4. Dominion worked to develop and deploy a workable synchrophasor data concentration architecture that would facilitate the operation of the three-phase PMU-based state estimator and contain the related displays. At this point in the project, we anticipated a final solution looking like a mixture of the two diagrams.

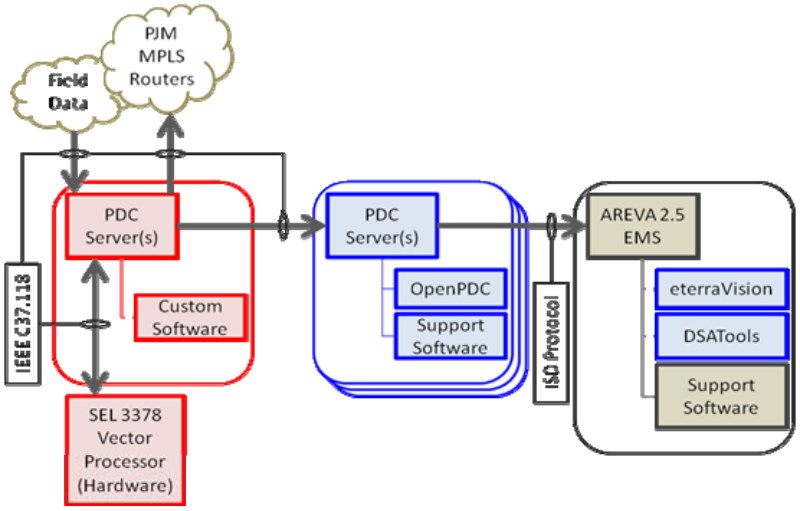


Figure 3: Functional synchrophasor data concentration possible with commercially available off-the-shelf products.

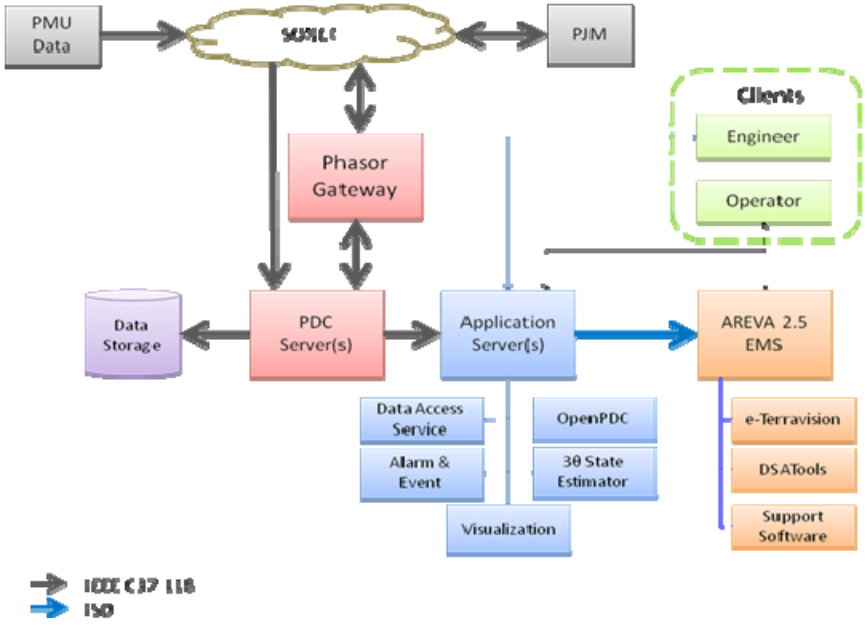


Figure 4: Conceptual functional components to accommodate future needs.

Task Update

Sub-award spending and engineering activities for Phase I were on schedule. Central to the timing of Dominion's synchrophasor installations was the availability of synchrophasor data concentration equipment. Dominion also ordered and received server hardware for use as a preliminary central data concentrator. While initial plans called for the use of the SEL 5073 software as a first attempt for central data concentration, Dominion began interacting with both the non-profit Grid Protection Alliance (GPA) and ALSTOM to investigate the use of the "OpenPDC" as a more permanent central data concentration solution. It was expected that the hardware and software architecture for central data concentration and applications would mature over the next year (Phase II).

Collaboration

At this point, it became unclear whether the SEL 3378 hardware was the optimal processor for use in this project. The use of the SEL 3378 vector processor was ultimately abandoned. Dominion decided not to perpetuate the annual contract with SEL for consulting services. We worked with other vendors (namely GPA) more native to synchrophasor data concentration to develop a work package to assist us with our central IT needs.

Phase II, 1st Quarter Activities:

Task Update

Dominion committed to installing PMUs at three stations in calendar year 2011 as a cost-sharing effort for this project (not including the ongoing work to be completed as a part of last year's work package).

Collaboration

Dominion worked with Grid Protection Alliance (GPA) and grant partners for the development of detailed software and hardware architectures for the central PDC. A statement of work was released to GPA. Upon developing a concise architecture for the PDC, Dominion planned to work with the company's EMS vendor for deployment of the architecture, although the Company eventually deployed the developed architecture with GPA.

At this point Dominion advised Quanta to discontinue PDC architecture work and focus on visualization tasks. Dominion decided to seek input on PDC architecture from consultants more familiar with the technical intricacies of this space (GPA).

Phase II, 2nd Quarter Activities:

Task Update

Engineering for the deployment of PMUs at three stations in Q4 2011 was underway as planned.

Dominion worked with Grid Protection Alliance (GPA) and grant partners for the development of detailed software and hardware architectures for the central PDC. A preliminary architecture document was drafted by GPA. Upon completion of the PDC documentation, Dominion planned to work with the company's EMS vendor for deployment of the architecture but eventually decided to implement the architecture with assistance from GPA.

Since Dominion had selected the OpenPDC to serve as the central data concentrator for synchrophasor data, Dominion IT reviewed impacts in terms of networking, risk management, and infrastructure, and worked with GPA to finalize the optimal implementation architecture.

Collaboration

Dominion continued collaboration with team members via bi-weekly conference calls.

Phase II, 3rd and 4th Quarter Activities:

IT Milestones Established for SOC PDC:

- Procurement of Open PDC: 12/2011
- Build Network Infrastructure & Configure SOC PDC: 3/2012
- Data Integration Central PDC: 5/2012
- Data Integration with Historian: 7/2012
- System Integration VT & Quanta: 8/2012

Outage availability remains a constant source of possible delays for future PMU deployments. Dominion makes reasonable attempts to mitigate these types of delays. However, some outage-related delays are inevitable, especially in the case of natural disasters.

Phase III, 1st Quarter Activities:

Task Update

PDC Update:

Dominion worked with EMS vendor ALSTOM and OpenPDC purveyor, GPA, for the deployment of an OpenPDC to serve the grant's data processing needs. Plans were on schedule to have the central PDC online in a basic mode of operation by the end of summer 2012. This task was accomplished.

Collaboration

At this point in the project, Dominion had discontinued biweekly calls with Quanta. Consultation from Quanta had led to an unworkable PDC architecture. Dominion had resorted to consulting with GPA for all further PDC architecture needs. At this point, to help Quanta control costs, Dominion suggested severely scaling back the scope of the visualization deliverables from Quanta. Conference calls to implement the basic visualization package would be scheduled closer to project close.

Phase III, 2nd Quarter Activities:

Task Update

PDC Update:

The company planned to have the central PDC online in a basic mode of operation by the end of summer 2012.

- Detailed physical architecture complete with GPA
- Build of physical architecture and infrastructure was in progress
- OpenPDC product implementation was in progress with GPA
- ISD link product SOW finalized with Alstom and was ready to implement

Collaboration

Bi-weekly conference calls have remained suspended during this time.

Phase III, 3rd Quarter Activities:

Task Update

PDC Update:

The company continues to make strides towards having the central PDC online in a basic mode of operation by the end of summer 2012.

- Build of physical architecture and infrastructure is 95% complete
- OpenPDC product validation was in progress with GPA
- Surry PMU data is streaming and validation is in progress
- Dominion planned and executed a "Go Live" with PJM at the end of July

Collaboration

Dominion contracted with Virginia Tech to have Mr. Kevin Jones join the Dominion team in Richmond on a temporary basis in the summer of 2012. Mr. Jones aided in deploying in state estimation and calibration code developed on the OpenPDC.

Dominion began the process of re-engaging Quanta Technology for the purpose of resuming display integration efforts.

4.2 Virginia Tech

4.2.1 Tracking state estimator:

As with the rest of the project, the tasks accomplished with the development of the three-phase linear state estimator and its applications can be divided into three phases. Each of the phases approximately encompasses one year of time. The first phase of the development includes the development and testing and the algorithms in Matlab. The second phase of the project encompasses those tasks which involved the migration of the Matlab algorithms into the openPDC platform by translating the algorithms into C#. During the final phase of the project, the modularized C# algorithms were synthesized into a unified structure which can properly serve Dominion Virginia Power in the long run. In many ways, the final results are beyond research grade software and prepare Dominion to fully utilize the capabilities of an EMS system which is driven by synchrophasors.

4.2.1 Phase I

The first phase of the software development included the development and testing of the three-phase linear state estimator and its applications in a Matlab environment. The Matlab environment served as an excellent starting point for development for several reasons.

1. It was readily available, as it is provided to engineering students by the university, to the three students working on the algorithm development.
2. It allowed students to work concurrently without having to worry during the initial development about how the algorithms would interact with each other and the synchrophasor data in the final implementation.
3. Coding algorithms in Matlab is a skill possessed by most electrical engineering graduate students and was therefore very easy to get started.

Algorithms that were developed in MATLAB include the three-phase linear state estimator, a topology processor, the instrument transformer calibration algorithm, and the islanding detection algorithm.

4.2.1.1 Three-phase Linear State Estimation Algorithm

The three-phase linear state estimator Matlab algorithm can be divided into three distinct pieces. First, it leveraged a set of *.m files that contained the system data. This includes network impedances and measurement information such as location of the measurement in the network. Second, functions were written which populated each of the pertinent matrices used in the calculation of the system matrix which represents the measurement footprint of the network and relates the synchrophasor measurements to the state of the system. The third piece of the Matlab algorithms for the three-phase linear state estimator was a script which testing the algorithm against the data set described below. The script was used to perform many iterations of the estimator on a single data set where the random errors that were added to the measurement set were different at each iteration.

One of the biggest challenges in the development and testing of the three-phase linear state estimation algorithm was the provision of the simulated data set. First, because the installation of the phasor measurement units had not yet begun, no real synchrophasor data was available. Secondly, a three-phase simulation program was not readily available and costly to develop from scratch. Finally, there was still no impedance data for the Dominion network available to Virginia Tech. Because of this, simplifications had to be made in order to compute a proper data set for testing of the state estimation algorithms. Positive sequence load flow results were extrapolated to three-phase and synthesized with three-phase unbalanced network impedances scaled to reflect the size of the impedances on the Dominion EHV transmission lines. This yielded a data set which reflected the unbalanced nature of the three-phase EHV network. This data set served as the true state of the system for the testing of the state estimation algorithms. Properly sized Gaussian errors were added to this data set to serve as the measurement set for the testing of the algorithms. The above processes were repeated when the true impedances of the Dominion network were made available to Virginia Tech.

4.2.1.2 Topology Processor

In addition to the development of the three-phase linear state estimation algorithms, it was deemed necessary to develop a topology processor algorithm for assessing the current topology of the network based on synchrophasor measurements. It was not felt that the breaker statuses provided to the current EMS were arriving fast enough to be used in a system with data resolution of 30 frames per second. Because of this, Dominion added many breaker statuses to the list of measurements provided by the PMUs that were to be installed. This was accomplished by using the digital input channels of the PMUs and dual-use line relays to monitor the breaker statuses in the substations. The breaker statuses are bitmapped into the digital word of the C37.118 stream and are therefore available at 30 frames per second as well.

The topology processor takes as input current phasors and breaker statuses from across the measured network. It uses an algorithm to determine whether the current phasors are 'zero' or not. A rough approximation of this algorithm is that if the value of the measurement falls below a percentage of the charging current of the transmission line in question then it is considered to be 'zero'. This information is synthesized with breaker statuses and the topology processor algorithm watches for changes in the states of both the current measurements and the breaker statuses. If something changes, breaker statuses are cross-referenced against the current measurements which measure the lines the breakers serve. If the current measurement in question is deemed 'zero' by the algorithm and the associated breaker status also show 'open' then the line in question is considered to be out of service. The branch will not be considered out-of-service unless all the states of the corresponding current measurements and breaker statuses agree.

4.2.2 Phase II

The second phase of the development of the software for this project can be summarized as the migration of the Matlab algorithms to the openPDC environment. This involves translating the algorithms into the C# language and organizing them to fit the design patterns required to implement the applications in what is called in openPDC jargon as 'Action Adapters'. Action Adapters are user defined phasor data concentration algorithms that run inside of the openPDC platform and can be written to perform tasks as simple as mapping phasor measurements from one naming convention to another and as complex as three-phase linear state estimation, topology processing, instrument transformer calibration, and islanding detection. openPDC served as a very powerful and flexible solution for the final implementation of the algorithms.

Because the openPDC was a still growing open source project there was not a tremendous amount of documentation available to Virginia Tech, a significant portion of time during this phase of the project was dedicated to the understanding of the openPDC and gaining proficiency in the use of the software. This roughly accounted for 40-45% of the time during this phase of the project. However, as typical with steep learning curves, the benefits were many as implementation of the algorithms in C# and openPDC were much easier to accomplish was Virginia Tech was proficient with both C# and openPDC.

Accomplishments during this phase of the project included the successful translation of each of the algorithms into C# and successfully running the algorithms in the openPDC environment. However, at the culmination of this phase of the project, the algorithms were still very much a modular solution and depended on data sets that had been generated in a simulated environment and read in to the openPDC in a different way than real measurements would be fed into the applications in the final implementation.

4.2.3 Phase III

In many ways, the final phase of the project was the most important and productive phase of the entire project for Virginia Tech. This phase can be divided up into three stages of progress. The first was the unification of the Virginia Tech application under a single data set which serves as the first step in the full integration of the algorithms.

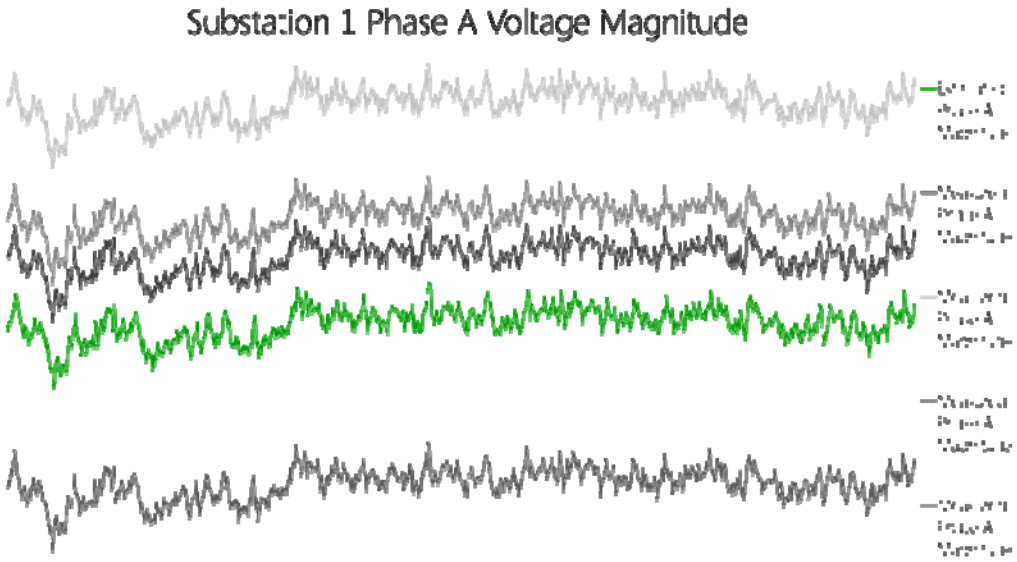
The unbalance monitoring application was not developed until the first stage of the third phase of the project. It was a simple algorithm and didn't require development in Matlab. It was directly developed in C# and openPDC. The unbalance monitoring application monitors the negative sequence current flows in the Dominion EHV network and provides the amperage of the negative sequence current, the ratio of the negative sequence current to the positive sequence current, and an alarm that is generated if the ratio of negative sequence current to positive sequence current goes above a user defined threshold.

This stage could have, if needed, resulted in a fully functional suite of synchrophasor applications. However, the next two stages of the third phase of the project were made possible by a deadline extension due to delays in PMU installation. This situation was fully taken advantage of by both Virginia Tech and Dominion.

The second stage of Phase III was a restructuring of the algorithms into one which follows better object oriented design practices and will be a more nimble and powerful solution that can grow and change as the demands of Dominion's changing EMS infrastructure grows and changes. The final stage of phase 3 of the software development was one of polishing and finalization of the algorithms as well as the integration of many great features of the state estimator. The latest build of the state estimator and its applications approaches a production grade implementation of the algorithms with the intent of a long term usage by Dominion Virginia Power. The functionality of the state estimator and its applications includes the following features:

1. A fully function synchrophasor-only three-phase state estimator.
2. A state estimator which employs a set of XML configuration files for storing system data including network impedances, measurement location information, names, and descriptions, measurement variances, openPDC Historian IDs, etc.
3. The ability to schedule system snapshots and take them on demand, preserving not only the state of the system but the measurements and configuration as well.
4. A state estimator which has the ability to automatically remove measurements based on topology changes, the loss of entire devices, or the loss of a whole substation.
5. A state estimator which has the ability to remove measurements that are temporarily or permanently loss due to communication problems.
6. A state estimator which has the ability to automatically remove measurements which have been flagged by the STATUS Word in the C37.118 frame as being invalid, out-of-sync, with error, etc.
7. A state estimator which has the ability to adapt to changes in observability.
8. A state estimator which has the ability to monitor islands of measurements; the measurement footprint need not be a contiguous network (physically or just in terms of measurements)
9. A state estimator which can perform the reverse operations of 3, 4, 5, 6, & 7.
10. A state estimator which leverages the results of a fully functional transformer calibration algorithm.
11. A fully functional calibration algorithm which computes instrument transformer errors every 24 if the EHV network is fully in service.
12. A fully functional islanding detection algorithm which detects the presence, geographic location, and severity of the islanding condition.
13. A fully function negative sequence current monitor which measures negative sequence amps, the ratio of negative sequence to positive sequence current, and alarms based on a user configurable setting.
14. Capability to similar functionality for zero sequence monitoring.
15. Capability to implement negative and zero sequence monitoring of voltages.
16. Capability to implement calculation of real and reactive power flows based on state estimator output.

17. A graphical user interface for maintaining the network model used by all of the synchrophasor applications.
18.



4.2.3.1 Network Model Editor GUI

It was strongly felt that providing a graphical user interface for maintaining the network model and measurement information would be a better solution that manually editing the configuration files. Therefore, a tabular GUI was developed to provide an improved workflow for managing the network model used by the state estimator and its applications. Additionally, there are automatic checks that are built into the table to prevent users from making simple mistakes when creating new or editing existing entries in the network model configuration files.

4.2.3.2 C# Algorithm Documentation & User Documentation

Documentation for all of the algorithms developed by Virginia Tech has been written. It exists in several forms. For the three C# libraries containing the Virginia Tech application algorithms and the Virginia Tech openPDC adapters, proper API documentation has been provided in the form of a Microsoft Help File. This HTML based format documents the classes in each of the libraries and provides information for using the algorithms for developing further applications based on the existing code and algorithms. The open source Sandcastle Help File Builder was used for using the XML markup in the source code to generate this searchable help file. In addition, to the documentation provided regarding the libraries developed by Virginia Tech, guides for implementing the applications and navigating the openPDC have been provided in the same compiled help file. Finally, all of the source code contains clear and abundant commenting in addition to the XML markup used to generate the compiled help files. Now that the projected is hosted open source, all of these resources can be located at the URL http://phasoranalytics.codeplex.com

4.3 Quanta Technology

This report provides a summary of the contributions of Quanta Technology that can be used for the 4600 Federal Asst. Reporting Checklist document. To facilitate report integration, the numbering from the 4600 Federal Asst. Reporting Checklist is included in this document.
From the PMU system architecture and PMU application visualizations perspective, a fundamental original hypothesis was that the displays to develop would be standalone applications, completely separated from the existing EMS/SCADA system already in place at Dominion's control center.

Probably one of the most significant deviations from the planned work/methodology was the inclusion of the PMU data in to the existing EMS/SCADA system already in place at Dominion's control center. This was a good decision by Dominion and radically changed the initially conceived project architecture and the visualization structure, design, look and feel, etc., as the visualization requirements were elevated to production grade levels. Another departure was related to the delays in which the real data would come available for the display visualization final testing.

Quanta Technology kept constant communications with its project partners Virginia Tech and Dominion, by bi-weekly project team conference calls and webinars. Together, the three entities successfully discussed the project deviations and agreed to a win-win solution that considered the overall project objectives and the allocated funds to execute the work.

The modifications and departure from planned methodology were always focused on the end customer: Dominion Virginia Power. The team was aware that modifications over the three year project duration would come and they were handled in such a way that they always have positive impact on the project results.

From the visualizations perspective, the objectives have been fulfilled as the visualization tools were demonstrated and the displays developed by Quanta Technology helped Dominion Virginia Power to select production grade and commercially available visualization tools. The delivery of final software and installation instructions is planned for October 2013.

The main results and the capabilities of the visualization software development were shared with the electric power industry in the North American Synchrophasor Initiative (NASPI) meeting of February 2012. A presentation was delivered by Dominion Virginia Power staff (Thomas Retiz) within the Operators Task Team in NASPI and served to guide interesting discussions regarding the balance that must be achieved between the complexity of the PMU application and visualization displays that are proposed by academia, versus the simplicity and slim requirements from the electric power system operators that are in charge of controlling the electric power grid. There is a fundamental issue that the industry is currently recognizing and the presentation and discussion by Dominion staff reminded the electric power industry the fact that most of the

electric power grid operators may not be engineers and as such the visualization tools must be designed accordingly. The visualization display developed under this grant is a great example of a useful display that was developed in close collaboration with the power system operators and which may be used in the real time control center environment.

A summary of the synchrophasor based tracking three-phase state estimator and its applications project in the context of the Smart Grid investments and US DOE grants for the application and use of PMUs was included in a paper co-authored by Dominion and Quanta Technology staff. The paper was presented in the IEEE General meeting in July 2012. The paper discusses the industry transformation and the initial use of PMUs information mainly by the transmission department of electric power utilities for disturbance analysis. Now the electric power industry is facing the entrance of PMU information and applications to the real time control center environment. Given the amount of investments in PMU related activities by US DOE, it is expected that technology transfer activities take place inside and outside the USA. The paper provides an example of how the application of PMU technology is crossing the frontiers of the USA by describing a PMU based initiative in Colombia.

The visualization applications are stored in a computer server within the security perimeter of Quanta Technology. Additionally, the initial software code that supports the visualization applications is stored in another computer server within the security perimeter of Dominion Virginia Power. The final version of the software will be transferred to Dominion in a trip to the control center in October 2013.

A number of industry collaborations fostered during the project include the participation of Quanta Technology, Dominion and Virginia Tech staff in a number the NASPI meetings. Quanta Technology staff strategically selected the NASPI meetings organized on February 2011, September 2011, and February 2012 in which different specific project technical topics were discussed. In the NASPI 2012 meeting in February there was a particularly opportunity to share with the electric power industry the visualization applications developed.

The project applied the latest technologies from web-based visualization applications and the software tools utilized to develop the applications are listed below:

- Open PDC
- SQL server
- Microsoft Visual Studio
- Expression Blend
- Resharper.
- SQL Server 2008 R2
- Silverlight
- Windows Server 2008 R2
- Infragistics data visualization tool for map display

a. Inventions/Patent Applications, licensing agreements;

The source code and all related software developed are to be provided to Dominion Virginia Power in October 2013 as the application may be part of the visualizations in the control center environment.

The main contribution is the source code which supports the visualization of the PMU applications. All visualization code was developed with funds of US DOE and all codes will be transferred to Dominion Virginia Power.

The software code that supports the PMU application visualizations was executed in C sharp (C#), Silverlight and .NET 4.0. The software pulls the data from the OpenPDC database and displays it in the visualization screens at a rate of one new data sample per second.

This is not applicable for the software code developed.

The verification, validation, and testing of the visualization applications is currently being finalized with the use of Dominion field data and will be delivered to Dominion by October 2013 with the final source code.

The visualization application is currently in a server at Dominion. The hardware requirements and characteristics are specified by Dominion Virginia Power.

The visualization applications user guide is currently in progress and has been provided to Dominion with the final source code delivery.

The main results and the capabilities of the visualization software development were shared with the electric power industry in the North American Synchrophasor Initiative (NASPI) meeting of February 2012. A presentation was delivered by Dominion Virginia Power staff (Thomas Retiz) within the Operators Task Team in NASPI and served to guide interesting discussions regarding the balance that must be achieved between the complexity of the PMU application and visualization displays that are proposed by academia, versus the simplicity and slim requirements from the electric power system operators that are in charge of controlling the electric power grid.

There is a fundamental issue that the industry is currently recognizing and the presentation and discussion by Dominion staff reminded the electric power industry the fact that most of the electric power grid operators are not engineers and as such the visualization tools must be designed accordingly. The visualization display developed under this grant is a great example of a useful display that was developed in close collaboration with the power system operators and which will be used in the real time control center environment. A copy of the presentation is included in Appendix to this document.

A summary of the Synchrophasor based tracking three-phase state estimator and its applications project in the context of the Smart Grid investments and US DOE grants for the application and use of PMUs was included in a paper co-authored by Dominion and Quanta Technology staff. The paper was presented in the IEEE General meeting in July 2012. The paper discusses the industry transformation and the initial use of PMUs information mainly by the transmission department of electric power utilities for disturbance analysis. Now the electric power industry is experimenting the entrance of PMU information and applications to the real time control center environment. Given the amounts of investments in PMU related activities by US DOE, it is expected that technology transfer activities take place inside and outside the USA. The paper provides an example of how the application of PMU technology is crossing the frontiers of the USA by describing a PMU based initiative in Colombia. A copy of the paper and presentation is included in Appendix to this document

The visualization applications are stored in a computer server within the security perimeter of Quanta Technology. Additionally, the software code that supports the visualization applications is stored in another computer server within the security perimeter of Dominion Virginia Power.

A number of industry collaborations fostered during the project include the participation of Quanta Technology, Dominion and Virginia Tech staff in a number the NASPI meetings. Quanta Technology staff strategically selected the NASPI meetings organized on February 2011, September 2011, and January 2012 in which different specific project technical topics were discussed. In the NASPI 2012 meeting in January there was a particularly opportunity to share with the electric power industry the visualization applications developed.

The project applied the latest technologies from web-based visualization applications and the software tools utilized to develop the applications are listed below:

- Open PDC
- SQL server
- Microsoft Visual Studio
- Expression Blend
- Resharper.
- SQL Server 2008 R2
- Silverlight
- Windows Server 2008 R2
- Infragistics data visualization tool for map display

The source code and all related software development are to be provided to Dominion Virginia power as the application are to be part of the visualizations in the control center environment.

The main contribution is the source code which supports the visualization of the PMU applications. All data and databases are proprietary information of Dominion Virginia Power

The software code that supports the PMU application visualizations was executed in C sharp (C#), Silverlight, and .NET 4.0. The software pulls the data from the OpenPDC and displays it in the visualization platform.

The visualization application is currently in a server at Dominion. The hardware requirements and characteristics are specified by Dominion Virginia Power.

The visualization applications user guide is currently in progress and will be provided to Dominion and Virginia Tech with the final source code.

5.0 APPENDIX II: PRODUCT DEVELOPMENT AND TECHNOLOGY TRANSFER

[1] “Three-Phase Linear State Estimation Using Phasor Measurements” Kevin D. Jones, Student Member, IEEE, James S. Thorp, Life Fellow, IEEE, and R. Matthew Gardner, Member, IEEE

[2] Link to Thesis of Kevin Jones: http://scholar.lib.vt.edu/theses/available/etd-05022011-141649/unrestricted/Jones_KD_T_2011.pdf

[3] Rui Sun, Zhongyu Wu, Virgilio A. Centeno, “Power System islanding Detection and Identification Strategy using Topology Approach and Decision Tree”, *Power & Energy Society General Meeting, 2011*. PES ‘11. IEEE, July 24-29, 2011

[4] R. Sun and V. A. Centeno, “Wide Area System Islanding Contingency Detecting and Warning Scheme with the Implementation of Synchrophasor Measurements”, North American SynchroPhasor Initiative Work Group Meeting, 2012, Feb 29 – March 1, 2012

[5] Rui Sun, “Wide Area Power System Islanding Detection, Classification and State Evaluation Algorithm”, Ph. D Dissertation, Virginia Polytechnic Institute and State University, December, 2012.

[6] “Synchronized Phasor Measurement Applications in Power Systems”; Jaime De La Ree, Senior Member, IEEE, Virgilio Centeno, Senior Member, IEEE, James S. Thorp, Life Fellow, IEEE, and A. G. Phadke, Life Fellow, IEEE

[7] “Dynamic State Prediction Based on Auto- Regressive (AR) Model Using PMU Data” ;Fenghua Gao, James. S. Thorp, Life Fellow, IEEE, Anamitra Pal, Student Member, IEEE, and Shibin Gao

[8] “Recent Developments in State Estimation with Phasor Measurements”; A.G. Phadke Life Fellow IEEE , J.S. Thorp Life Fellow IEEE, R.F. Nuqui Member IEEE, M. Zhou Student Member IEEE

[9] COMMUNICATION NEEDS FOR WIDE AREA MEASUREMENT APPLICATIONS A.G. Phadke, Life Fellow IEEE, and J.S. Thorp, Life Fellow, IEEE

[10] David Elizondo, R. Matthew Gardner and Ramón Leon, Synchrophasor Technology: The Boom of Investments and Information Flow from North America to Latin America. IEEE General Meeting 2012.

[11] Plans for the integration of new synchrophasor based information to Dominion control room environment—an operators perspective. Thomas Reitz and David Elizondo. North American Synchrophasor Initiative, February 2012.

6.0 APPENDIX III: CALIBRATION CODE FILES

State Estimation documentation is hosted on tha Codeplex site where the source code is listed. The link to the web-site is http://phasoranalytics.codeplex.com

1. Instrument Transformer calibration code: See Appendix IV.

File multicases_cali.m

```
clear all;
CaseNum=16; %the number of cases. Redundancy

Branch = load('Branch_1P.txt');% get the branch and bus information
NodeNum = max(max(Branch(:,1)),max(Branch(:,2)));
BranchNum = size(Branch,1);

tol=10e-16; %tolence for iteration
max_it=30; % maximum iteration


P_vt=[40,41,42]; %% [14*3-2:14*3]. the bus with perfect PT. Here, PT is installed at bus 14.
P_VT=[P_vt,3*NodeNum+P_vt];%% calibrate RCF(magnitude) PACF(angle)
PerfNum=length(P_vt); %% the number of pre-cali PT
% wind load
LoadPercent=[1.10 1.20 0.28 0.32 0.33 0.37 0.40 0.43 0.48 0.50 0.62 0.78 0.86 0.92 0.96 0.98 1.00];
% light load
%LoadPercent=[0.10 0.12 0.13 0.16 0.19 0.20 0.23 0.26 0.28 0.29];

V3_average_unbal=zeros(3*NodeNum,CaseNum); %initial V without error

for i1=1:CaseNum
   loadfile=sprintf('V1b_load%0.2f.txt',LoadPercent(i1));
   [V3_average_unbal(:,i1),yA_P_y0A_0_P,yA_N_y0A_0_N]=GetVoltUnbalance(loadfile);
end

YY=[yA_P_y0A_0_P;yA_N_y0A_0_N]; % the branch matrix
%% measurement
E_I_measure=zeros(3*NodeNum+6*BranchNum,CaseNum); % initial. V and I with error

load r0_vector; % get the error model for CVT
load r0_ct_vector;%get the error model for CT

r0_vector=r0_vector(1:3*NodeNum);
r0_ct_vector=r0_ct_vector(1:6*BranchNum);


r0=diag(r0_vector);
r0_ct=diag(r0_ct_vector);

for i8=1:PerfNum % set the perfect PT's error as 1
   r0(P_vt(i8),P_vt(i8))=1;
```

```
   r0_vector((P_vt(i8)))=1;
end

BB1=r0_ct*YY;
BB0=[r0;BB1];

for i3=1:CaseNum
   E_I_measure(:,i3)=BB0*V3_average_unbal(:,i3);% get the V and I with errors.
   yt(:,i3)=[r0_vector',r0_ct_vector']'; % the real correction factor of CT and VT
end

V3=zeros(3*NodeNum,CaseNum);
for i6=1:CaseNum
   V3(:,i6)=E_I_measure(1:3*NodeNum,i6);
end

I3=zeros(6*BranchNum,CaseNum);
for i7=1:CaseNum
   I3(:,i7)=E_I_measure(3*NodeNum+1:3*NodeNum+6*BranchNum,i7);
end
load r_pmu_magn; % add PMU error to I3 and V3
load r_pmu_ang;

r_pmu_vector=1+complex(r_pmu_magn.*cosd(r_pmu_ang),r_pmu_magn.*sind(r_pmu_ang));

% comment out the 2 line below, if assume PMU has no error
I3=r_pmu_vector(1:6*BranchNum,1:CaseNum).*I3;
V3=r_pmu_vector(1:3*NodeNum,1:CaseNum).*V3;
%% ////////////////////////////////////////////////////////////////


I3_r=real(I3);
I3_i=imag(I3);

YY_r=real(YY);
YY_i=imag(YY);

V3_r=real(V3);
V3_i=imag(V3);
```

File multicases_cali.asv

```
clear all;
CaseNum=16; %the number of cases. Redundancy

Branch = load('Branch_1P.txt');% get the branch and bus information
NodeNum = max(max(Branch(:,1)),max(Branch(:,2)));
BranchNum = size(Branch,1);

tol=10e-16; %tolence for iteration
max_it=30; % maximum iteration


P_vt=[40,41,42]; %% [14*3-2:14*3]. the bus with perfect PT. Here, PT is installed at bus 14.
P_VT=[P_vt,3*NodeNum+P_vt];%% calibrate RCF(magnitude) PACF(angle)
PerfNum=length(P_vt); %% the number of pre-cali PT
% wind load
LoadPercent=[1.10 1.20 0.28 0.32 0.33 0.37 0.40 0.43 0.48 0.50 0.62 0.78 0.86 0.92 0.96 0.98 1.00];
% light load
%LoadPercent=[0.10 0.12 0.13 0.16 0.19 0.20 0.23 0.26 0.28 0.29];

V3_average_unbal=zeros(3*NodeNum,CaseNum); %initial V without error

for i1=1:CaseNum
   loadfile=sprintf('V1b_load%0.2f.txt',LoadPercent(i1));
   [V3_average_unbal(:,i1),yA_P_y0A_0_P,yA_N_y0A_0_N]=GetVoltUnbalance(loadfile);
end

YY=[yA_P_y0A_0_P;yA_N_y0A_0_N]; % the branch matrix
%% measurement
E_I_measure=zeros(3*NodeNum+6*BranchNum,CaseNum); % initial. V and I with error

load r0_vector; % get the error model for CVT
load r0_ct_vector;%get the error model for CT

r0_vector=r0_vector(1:3*NodeNum);
r0_ct_vector=r0_ct_vector(1:6*BranchNum);


r0=diag(r0_vector);
r0_ct=diag(r0_ct_vector);

for i8=1:PerfNum % set the perfect PT's error as 1
   r0(P_vt(i8),P_vt(i8))=1;
   r0_vector((P_vt(i8)))=1;
end

BB1=r0_ct*YY;
BB0=[r0;BB1];

for i3=1:CaseNum
   E_I_measure(:,i3)=BB0*V3_average_unbal(:,i3);% get the V and I with errors.
end
```

```
V3=zeros(3*NodeNum,CaseNum);
for i6=1:CaseNum
   V3(:,i6)=E_I_measure(1:3*NodeNum,i6);
end

I3=zeros(6*BranchNum,CaseNum);
for i7=1:CaseNum
   I3(:,i7)=E_I_measure(3*NodeNum+1:3*NodeNum+6*BranchNum,i7);
end
load r_pmu_magn; % add PMU error to I3 and V3
load r_pmu_ang;

r_pmu_vector=1+complex(r_pmu_magn.*cosd(r_pmu_ang),r_pmu_magn.*sind(r_pmu_ang));

I3=r_pmu_vector(1:6*BranchNum,1:CaseNum).*I3;
V3=r_pmu_vector(1:3*NodeNum,1:CaseNum).*V3;
%% ////////////////////////////////////////////////////////////////
```

File GetYVI.m

```
function [V_balance,I_balance_P,I_balance_N,y_P_branch,y_N_branch]=GetYVI(filename)
%% Load branch data
Branch = load('Branch_1P.txt');
%% Get number of buses in system
NodeNum = max(max(Branch(:,1)),max(Branch(:,2)));
%% Get number of branches in system
BranchNum = size(Branch,1);
%% Form Impedance Matrix
for i = 1 : BranchNum
   Branch(i,5) = j*Branch(i,5);
   Branch(i,6) = 1/(Branch(i,3)+j*Branch(i,4));
end
Y= zeros(NodeNum);
y_branch=zeros(BranchNum);
y_0branch=zeros(BranchNum);

for i = 1 : BranchNum
   StartNode = Branch(i,1);
   EndNode = Branch(i,2);
   y_0branch(i,i) = Branch(i,5)/2;
   y_branch(i,i) = Branch(i,6);
   %Yii
   Y(StartNode,StartNode) = Y(StartNode,StartNode) + y_branch(i,i) + y_0branch(i,i);
   Y(EndNode,EndNode) = Y(EndNode,EndNode) + y_branch(i,i) + y_0branch(i,i);
   %Yij
   Y(StartNode,EndNode) = Y(StartNode,EndNode) - y_branch(i,i);
   Y(EndNode,StartNode) = Y(StartNode,EndNode);
end

A_P=zeros(BranchNum,NodeNum);
A_N=zeros(BranchNum,NodeNum);
A_0_P=zeros(BranchNum,NodeNum);
A_0_N=zeros(BranchNum,NodeNum);

for i1= 1:BranchNum
   A_P(i1,Branch(i1,1))=1;% phase A. Begin-end
   A_P(i1,Branch(i1,2))=-1;
   A_N(i1,Branch(i1,2))=1;% phase A, end-begin
   A_N(i1,Branch(i1,1))=-1;
   A_0_P(i1,Branch(i1,1))=1;
   A_0_N(i1,Branch(i1,2))=1;

end
y_P_branch=y_branch*A_P+y_0branch*A_0_P;
y_N_branch=y_branch*A_N+y_0branch*A_0_N;

V_balance=zeros(NodeNum,1);
V1b=load(filename);
for i=1:NodeNum
   V_balance(i,1)=complex(V1b(i,1)*cosd(V1b(i,2)),V1b(i,1)*sind(V1b(i,2)));
end
```

```
I_balance_P=y_P_branch*V_balance;
I_balance_N=y_N_branch*V_balance;
```

File GetYUnbalance.m

```
function
[Y_unbalance_matrix,yA_P_y0A_0_P,yA_N_y0A_0_N,NodeNum,BranchNum,y_self,y_0_self,Zs_branch,Bs
_branch]=GetYUnbalance()

Branch = load('Branch_3.txt');

NodeNum = max(max(Branch(:,1)),max(Branch(:,2)));
BranchNum = size(Branch,1);

Y_unbalance=cell(NodeNum);
for i = 1:NodeNum
   for ii=1:NodeNum
      Y_unbalance{i,ii}=zeros(3);
   end
end

Zs_branch=cell(BranchNum,1);% z
Bs_branch=cell(BranchNum,1);% bus B

y=cell(BranchNum); % branch y
y_0=cell(BranchNum);% branch y0


%% Get unbalance Y cell from unbalance branch data

for i = 1:BranchNum
   for ii=1:BranchNum
      y{i,ii}=zeros(3);
   end
end

for i = 1:BranchNum
   for ii=1:BranchNum
      y_0{i,ii}=zeros(3);
   end
end
%% Form Zs_branch and Bs_branch
for i= 1: BranchNum
   for i1=1:6
      Branch_impendance(i,i1)=complex(Branch(i,1+i1*2),Branch(i,2+i1*2));
   end
   for i2=1:6
      Branch_suspent(i,i2)=complex(0,Branch(i,14+i2));
   end
   Zs_branch{i}=[Branch_impendance(i,1) Branch_impendance(i,2) Branch_impendance(i,4)
            Branch_impendance(i,2) Branch_impendance(i,3) Branch_impendance(i,5)
            Branch_impendance(i,4) Branch_impendance(i,5) Branch_impendance(i,6)];

  % Bs_brach_temp=Branch_suspent(i,1)+Branch_suspent(i,3)+Branch_suspent(i,6);
   Bs_branch{i}=[ Branch_suspent(i,1) Branch_suspent(i,2) Branch_suspent(i,4)
            Branch_suspent(i,2)  Branch_suspent(i,3) Branch_suspent(i,5)
```

```
            Branch_suspent(i,4) Branch_suspent(i,5) Branch_suspent(i,6)];
end
%% Get unbalance Y cell from unbalance branch data
y_self=zeros(BranchNum,1);
y_0_self=zeros(BranchNum,1);

for i = 1 : BranchNum
   StartNode = Branch(i,1);
   EndNode = Branch(i,2);
   y{i,i}=Zs_branch{i}^(-1);
   y_self(i)=sum(diag(y{i,i}))/3;
   y_0{i,i}=Bs_branch{i}/2;
   y_0_self(i)=sum(diag(y_0{i,i}))/3;

   for ii = 1:3
      for jj = 1:3
        % Yii
        Y_unbalance{StartNode,StartNode}(ii,jj) = Y_unbalance{StartNode,StartNode}(ii,jj) +...
           y{i,i}(ii,jj) + Bs_branch{i}(ii,jj)/2;
        Y_unbalance{EndNode,EndNode}(ii,jj) = Y_unbalance{EndNode,EndNode}(ii,jj) + ...
           y{i,i}(ii,jj) + Bs_branch{i}(ii,jj)/2;
```

File GetYbalance.m

```
function Y_balance_matrix=GetYbalance()

[Y_unbalance_matrix,yA_P_y0A_0_P,yA_N_y0A_0_N,NodeNum,BranchNum,y_self,y_0_self]=GetYUnbala
nce();


Y_balance=cell(NodeNum);

for i = 1:NodeNum
   for ii=1:NodeNum
      Y_balance{i,ii}=zeros(3);
   end
end

Branch = load('Branch_1P.txt');
y_pos = zeros(BranchNum,1);
y_0_pos = zeros(BranchNum,1);
for i1=1:BranchNum
   y_pos(i1)=1/(Branch(i1,3)+j*Branch(i1,4));
   y_0_pos(i1)=j*Branch(i,5)/2;
end
y_mutual=y_self-y_pos;
y_0_mutual=y_0_self-y_0_pos;

y_branch_cell = cell(BranchNum);
y_branch_0_cell=cell(BranchNum);

for i2=1:BranchNum
   y_branch_cell{i2}=[y_self(i2) y_mutual(i2)  y_mutual(i2);
                y_mutual(i2) y_self(i2) y_mutual(i2);
                y_mutual(i2) y_mutual(i2) y_self(i2);];
   y_branch_0_cell{i2}=[y_0_self(i2) y_0_mutual(i2)  y_0_mutual(i2);
                y_0_mutual(i2) y_0_self(i2) y_0_mutual(i2);
                y_0_mutual(i2) y_0_mutual(i2) y_0_self(i2);];
end

for i = 1 : BranchNum

   StartNode = Branch(i,1);
   EndNode = Branch(i,2);

   for ii = 1:3
      for jj = 1:3
        % Yii
         Y_balance{StartNode,StartNode}(ii,jj) = Y_balance{StartNode,StartNode}(ii,jj) +...
            y_branch_cell{i}(ii,jj) + y_branch_0_cell{i}(ii,jj);
         Y_balance{EndNode,EndNode}(ii,jj) = Y_balance{EndNode,EndNode}(ii,jj) + ...
            y_branch_cell{i}(ii,jj) + y_branch_0_cell{i}(ii,jj);
        %Yij
         Y_balance{StartNode,EndNode}(ii,jj) = Y_balance{StartNode,EndNode}(ii,jj) - y_branch_cell{i}(ii,jj);
         Y_balance{EndNode,StartNode}(ii,jj) = Y_balance{StartNode,EndNode}(ii,jj);
```

```
      end
   end
end
%%
Y_balance_matrix=zeros(3*NodeNum);
for ii=1:3*NodeNum
   for jj=1:3*NodeNum
      r3cell=floor((ii-1)/3)+1;
      c3cell=floor((jj-1)/3)+1;
      r3matrix=rem(ii-1,3)+1;
      c3matrix=rem(jj-1,3)+1;
      Y_balance_matrix(ii,jj)=Y_balance{r3cell,c3cell}(r3matrix,c3matrix);
   end
end
```

File GetVoltUnbalance_i_no_0.m

```
function [V3_average_unbal,yA_P_y0A_0_P,yA_N_y0A_0_N,I3_unbal_V3]=
GetVoltUnbalance_I_no_0(filename)
%% E_I_measure: three phase voltage and current with error
%% (3*NodeNum+6*BranchNum,1) [V,I]
%% yt: true vector of three phase voltage and error model value
%% (6*NodeNum+6*BranchNum-3,1) [V,r_vt,r_ct]

%%[Y_unbalance,yA_P_y0A_0_P,yA_N_y0A_0_N]=GetYUnbalance()
%clear all;
%filename='V1b_load0.99.txt';

%% run GetYUnbalance sub-function to get unbalance Y matrix(93*93)
[Y_unbalance_matrix,yA_P_y0A_0_P,yA_N_y0A_0_N,NodeNum,BranchNum]=GetYUnbalance();
%% load 3-phase voltage value
V1b=load(filename);
V3_balance=zeros(3*NodeNum,2);
for i =1:NodeNum
   V3_balance(i*3-2,:)=V1b(i,:);
   V3_balance(i*3-1,1)=V1b(i,1);
   V3_balance(i*3-1,2)=V1b(i,2)-120; % degree
   V3_balance(i*3,1)=V1b(i,1);
   V3_balance(i*3,2)=V1b(i,2)+120;
end

for i=1:3*NodeNum

V3_bal_complex(i,1)=complex(V3_balance(i,1)*cosd(V3_balance(i,2)),V3_balance(i,1)*sind(V3_balance(i,
2)));
end
%% Calculate I balance

Y_3phase_bal=GetYbalance();
%Y_3phase_bal=zeros(NodeNum);
%for i=1:NodeNum
%    for j =1:NodeNum
%        Y_3phase_bal(i*3,j*3)=Y_balance_1phase(i,j);
%        Y_3phase_bal(i*3-1,j*3-1)=Y_balance_1phase(i,j);
%        Y_3phase_bal(i*3-2,j*3-2)=Y_balance_1phase(i,j);
%    end
%end
I3_balance=Y_3phase_bal*V3_bal_complex;
%% polar of I balance
I3_balance_Polar(:,1)=abs(I3_balance);
I3_balance_Polar(:,2)=angle(I3_balance)*180/pi;
%% Calculate I unbalance
I3_unbalance=Y_unbalance_matrix*V3_bal_complex;
%% calculate I unbalance average
I3_average_unbal=zeros(NodeNum,1);
for i=1:3*NodeNum
   I3_average_unbal(i)=(I3_unbalance(i)+I3_balance(i))/2;
end
```

```
%% get 3-phase Zp
Z_3phase=Y_unbalance_matrix^(-1);
%% calculate V unbalance
V3_unbalance=Z_3phase*I3_balance;
%%
V3_average_unbal=zeros(NodeNum,1);
for i=1:3*NodeNum
   V3_average_unbal(i)=(V3_bal_complex(i)+V3_unbalance(i))/2;
end
%% change into polar
I3_average_unbal_Polar(:,1)=abs(I3_average_unbal);
I3_average_unbal_Polar(:,2)=angle(I3_average_unbal)*180/pi;
V3_average_unbal_Polar(:,1)=abs(V3_average_unbal);
V3_average_unbal_Polar(:,2)=angle(V3_average_unbal)*180/pi;
%% divide into real imaginary parts
I3_average_unbal_real_imaginary(:,1)=real(I3_average_unbal);
I3_average_unbal_real_imaginary(:,2)=imag(I3_average_unbal);
I3_average_unbal_real_imaginary(:,1)=real(V3_average_unbal);
I3_average_unbal_real_imaginary(:,2)=imag(V3_average_unbal);
%% check whether I3_average and V2_average are Z
I3_unbal_V3=Y_unbalance_matrix*V3_average_unbal;
check=I3_unbal_V3-I3_average_unbal;
%%get line current
```

File GetVoltUnbalance.m

```
function [V3_average_unbal,yA_P_y0A_0_P,yA_N_y0A_0_N,I3_unbal_V3]= GetVoltUnbalance(filename)
%% E_I_measure: three phase voltage and current with error

%% yt: true vector of three phase voltage and error model value

%clear all;
%filename='V1b_load0.92.txt';

%% run GetYUnbalance sub-function to get unbalance Y matrix(93*93)
[Y_unbalance_matrix,yA_P_y0A_0_P,yA_N_y0A_0_N,NodeNum,BranchNum]=GetYUnbalance();
%% load 3-phase voltage value
V1b=load(filename);
V3_balance=zeros(3*NodeNum,2);
for i =1:NodeNum
   V3_balance(i*3-2,:)=V1b(i,:);
   V3_balance(i*3-1,1)=V1b(i,1);
   V3_balance(i*3-1,2)=V1b(i,2)-120; % degree
   V3_balance(i*3,1)=V1b(i,1);
   V3_balance(i*3,2)=V1b(i,2)+120;
end

for i=1:3*NodeNum

V3_bal_complex(i,1)=complex(V3_balance(i,1)*cosd(V3_balance(i,2)),V3_balance(i,1)*sind(V3_balance(i,
2)));
end
%% Calculate I balance

Y_3phase_bal=GetYbalance();

%%
Noinject_Bus=load('noinjectbus.txt');
Num_noinject=length(Noinject_Bus);
Noinject_3bus=zeros(3*Num_noinject,1);
for ii1=1:Num_noinject
   Noinject_3bus(ii1*3)=Noinject_Bus(ii1)*3;
   Noinject_3bus(ii1*3-1)=Noinject_Bus(ii1)*3-1;
   Noinject_3bus(ii1*3-2)=Noinject_Bus(ii1)*3-2;
end
Allbus=1:3*NodeNum;
%% a: load_gen bus ; b:no inject bus
Load_Gen_bus=setdiff(Allbus,Noinject_3bus);
Yaa_bal=Y_3phase_bal(Load_Gen_bus,Load_Gen_bus);
Yab_bal=Y_3phase_bal(Load_Gen_bus,Noinject_3bus);
Yba_bal=Y_3phase_bal(Noinject_3bus,Load_Gen_bus);
Ybb_bal=Y_3phase_bal(Noinject_3bus,Noinject_3bus);

E_gen_load_bal=V3_bal_complex(Load_Gen_bus);
E_noinj_bal=-inv(Ybb_bal)*Yba_bal*E_gen_load_bal;

V3_bal_complex(Noinject_3bus)=E_noinj_bal;
```

```
I3_balance=Y_3phase_bal*V3_bal_complex;
%%



%% polar of I balance
I3_balance_Polar(:,1)=abs(I3_balance);
I3_balance_Polar(:,2)=angle(I3_balance)*180/pi;

%% Calculate I unbalance
I3_unbalance=Y_unbalance_matrix*V3_bal_complex;

%% calculate I unbalance average
I3_average_unbal=zeros(NodeNum,1);
for i=1:3*NodeNum
   I3_average_unbal(i)=(I3_unbalance(i)+I3_balance(i))/2;
end

%% get 3-phase Zp
Z_3phase=Y_unbalance_matrix^(-1);

%% calculate V unbalance
V3_unbalance=Z_3phase*I3_balance;
%%
V3_average_unbal=zeros(NodeNum,1);
for i=1:3*NodeNum
   V3_average_unbal(i)=(V3_bal_complex(i)+V3_unbalance(i))/2;
end
%% change into polar
I3_average_unbal_Polar(:,1)=abs(I3_average_unbal);
I3_average_unbal_Polar(:,2)=angle(I3_average_unbal)*180/pi;

V3_average_unbal_Polar(:,1)=abs(V3_average_unbal);
V3_average_unbal_Polar(:,2)=angle(V3_average_unbal)*180/pi;

%% divide into real imaginary parts
I3_average_unbal_real_imaginary(:,1)=real(I3_average_unbal);
I3_average_unbal_real_imaginary(:,2)=imag(I3_average_unbal);

V3_average_unbal_real_imaginary(:,1)=real(V3_average_unbal);
V3_average_unbal_real_imaginary(:,2)=imag(V3_average_unbal);

%% check whether I3_average and V2_average are Z
I3_unbal_V3=Y_unbalance_matrix*V3_average_unbal;
check=I3_unbal_V3-I3_average_unbal;



%% a: load_gen bus ; b:no inject bus
Load_Gen_bus=setdiff(Allbus,Noinject_3bus);
Yaa=Y_unbalance_matrix(Load_Gen_bus,Load_Gen_bus);
Yab=Y_unbalance_matrix(Load_Gen_bus,Noinject_3bus);
Yba=Y_unbalance_matrix(Noinject_3bus,Load_Gen_bus);
```

```
Ybb=Y_unbalance_matrix(Noinject_3bus,Noinject_3bus);

E_gen_load=V3_average_unbal(Load_Gen_bus);
E_noinj=-inv(Ybb)*Yba*E_gen_load;

V3_average_unbal(Noinject_3bus)=E_noinj;

I3_noinj_gen_load=Y_unbalance_matrix*V3_average_unbal;

%%get line current

%% I3_line is the matrix of line current, the first column is from start to
%% end, the second is end to start buses

I3_line_P=yA_P_y0A_0_P*V3_average_unbal;
%I3_line_N=yA_N_y0A_0_N*V3_average_unbal;
```