DOE FOA 970

open ECA

open and Extensible Control & Analytics platform for synchrophasor data

Platform and Analytics Alpha Test Results

Grid Protection Alliance
1206 Broad Street
Chattanooga, Tennessee 37402

Prepared for
U.S. Department of Energy

April 2017

T&D Consulting Engineers

Platform and Analytics Alpha Test Results

CONTENTS

1 SUMMARY

The objective of the Open and Extensible Control and Analytics (openECA) Platform for Phasor Data project is to develop an open source software platform that significantly accelerates the production, use, and ongoing development of real-time decision support tools, automated control systems, and off-line planning systems that (1) incorporate high-fidelity synchrophasor data and (2) enhance system reliability while enabling the North American Electric Reliability Corporation (NERC) operating functions of reliability coordinator, transmission operator, and/or balancing authority to be executed more effectively.

The openECA platform will provide a Common Analytics Interface (CAI) for integration of a diverse set of platform analytics along with structured integration of platform configuration, display and storage systems.

The platform will include an open-source Linear State Estimator (LSE) as a core component of the openECA platform to enable the results from the LSE to be easily incorporated into other openECA analytical components. The openECA platform will enable the secure, high-performance exchange of synchrophasor data with external entities through publish/subscribe protocols and will include a local historian to archive openECA performance statistics. The openECA platform will provide an alarming engine that can raise alarms based on high and low data set points and will also provide a common set of visualization displays optimized for testing and verification of analytic results that can be also used to simplify information presentation for decision support.

This project will develop and/or refine to pre-commercial status nine analytic packages (some open source and some proprietary) that can be deployed using the openECA platform CAI. These nine analytic packages are divided into the three classes of real-time decision support, control, and off-line analytics as follows:

Real-Time Analytics

1. Oscillation Detection Monitor (ODM)
2. Oscillation Mode Meter (OMM)
3. Topology Estimation

Control Analytics

4. Regional Volt-Ampere-Reactive (VAR) Control
5. Local VAR Control
6. Phasor Measurement Unit (PMU) Synchroscope

• **Off-Line Analytics**

7. Dynamic PMU Transducer Calibration (Automated, Periodic Use Case)
8. Line Parameter Estimation (Ad-Hoc Use Case)
9. Synchronous Machine Parameter Estimation (Automated, Periodic Use Case)
10. Acceleration Trend Relay (ATR) Improvement (Research Use Case)

The openECA project completed its Phase 1 Design efforts in 2016. During the first quarter of calendar year 2017, the openECA team conducted bench testing of alpha versions of the openECA platform and analytics.

This report presents the results of this testing.

2 OPENECA PLATFORM

The Initial Alpha Release of openECA was issued on Jan 2, 1017. It included metadata and data structure definitions for returning values to the openECA server components and enhanced UI components.

Major architectural elements of the openECA platform include:

- Data Integration Services
- Common Analytics Interface
- Data Conditioning and Alarming
- Electric Network Model
- Shared Platform Services

OpenECA defines a unified environment for modeling an analytic's:

- Configuration
- Data Structures, and
- Measurement Mapping,

The Data Modeling Manager Tool allows the analytic developer to define two classes of data structures:

- The domain input, called SourceData
- The analytic product, called ResultData

The contents of these data structures are under the complete control of the analytic developer.

Alpha version testing has demonstrated that the openECA platform allows developers to easily create new analytics by creating data structures and mapping to streamed data sources. Developers can select a target language for the analytic and then use the tool to create a new analytic project.

Based on the results of Alpha testing, development of the Beta version of openECA is on schedule for completion by the end of May and installation at demonstration sites in June and July.

3 LINEAR STATE ESTIMATOR (LSE)

Progress Overview:

Given the need for a LSE by a broad set of openECA platform analytics, the Virginia Tech/Dominion-developed LSE will be included as a core GSF component and therefore a core component of the openECA platform. The associated goals and related progress are outlined as follows:

1. Merging of the LSE code base with GPA's Project Alpha – this task was completed and tested as part of the Alpha Development Phase. However, after use and experimentation with the new openECA features, it was decided to rescope how LSE was structured as a standalone component. For Beta and for future releases, the LSE will be structured as an openECA Client Analytic to take advantage of all of the advances with the openECA platform that would not be available in use with a Project Alpha template.
2. Compatibility with openECA – The original LSE adapter has been migrated to an openECA Client host and tested against both an early Alpha version of openECA as well as an early Beta version of openECA.
3. Improvements to Modeling and Testing Tools – There have been substantial updates to the modeling and testing tools as part of the Apha-Beta Development Phase. These include the merging of the Network Model Editor Tool and the Offline Module Tool into a single unified tool. This is helpful for being able to test and troubleshoot modeling changes in the LSE in a much more streamlined manner. Additionally, many automation features have been added for preparing models from GE/Alstom EMS systems and subsequently pruning those models for use with the LSE. These improvements have been implemented and tested successfully on EMS data from two companies.
4. Built-in Sample Data and Sample Model – We have not yet established a sample data and sample model set that will be packaged with the LSE. However, through work at Virginia Tech and Dominion, there is plenty of material available to put this together. It will be included at a later point in the testing and demonstration.
5. Additional Topology Awareness – In partnership with Virginia Tech, work on a Topology Estimator which provides phasor measurement based awareness of substation topology to the LSE has been studied and is in development. While scoped as a separate analytic, it will be a native component of the LSE. Therefore, during Alpha-Beta development phase, the major updates to the LSE that are required to accommodate this new feature have been implemented. They have been tested in isolation with simulated data in the lab but have not been tested in concert as part of the overall LSE implementation. This testing will be done at a later point in the testing and demonstration.
6. Updates to the LSE Core – Additional updates have been added to the LSE Core. These included provided key performance metrics from inside the LSE as time-series-measurement output to openECA. This new feature will help with troubleshooting in real-time.
7. Ancillary Components – The Measurement Sample Adapter and the Snapshot Manager are simple adapters from the older GSF versions of the LSE Library. Given their ease of implementation and low complexity, they will not be migrated to openECA analytics until further in the Beta development process.

Updates to Test/Demonstration Configuration:

Dominion determined that the previously scoped test bench setup would not be sufficient for testing all features at scale. Therefore, Dominion has procured a phasor domain simulator (OPAL-RT ePHASORSIM) hardware to install in the Dominion RTDS lab as well as two servers that will host the Beta and Final Release of the openECA. The procurement for the ePHASORSIM is completed and ready to install. Installation and training for employees and the Virginia Tech students are scheduled in early June. The procurement of the servers is also scheduled to be completed around the same time. Dominion will also utilize its existing PDC Data Architecture as the data source for the openECA Servers for testing purposes.

Demonstration Procedures:

1. Dominion will install and configure the hardware for the demonstration environment. This includes the ePHASORSIM simulator and the server hardware for hosting openECA + LSE.
2. Dominion will install and configure the openECA software on the servers.
3. Dominion will use reduced power system models in the ePHASORSIM to generate real-time phasor domain simulations that will produce real-time streaming synchrophasor data.
4. Dominion will stream synchrophasor data from the ePHASORSIM and from our central PDC system to the servers hosting openECA.
5. Dominion will install the LSE alongside openECA on one of the two servers.
6. Dominion will use the latest features in the Network Model Editor Tool to create LSE Network Models that represent the test system in the ePHASORSIM as well as an LSE Network Model which represents Dominion's EMS Network Model.
7. Dominion will execute modeling tasks to map measurements from openECA to these two models.
8. Dominion will test the LSE (and its new topology estimation features) with these two models with data from each respective system (simulation and real-time). This will need to be done incrementally by starting with a single substation and adding modelling data one substation at a time to make sure the LSE is tuned properly.
9. Demonstration of this will consist of loading the LSE with as many measurements as are either available or as many as extends the full capacity of the hardware and displaying the resulting streams on openECA Grafana displays.
10. Two data sources will be demonstrated:
    a. Real-Time Data from the field will be sent to the LSE, processed, and displayed.
    b. Simulated data from ePHASORSIM will be sent to the LSE, processed, and displayed. Multiple scenarios will be tested/demonstrated which would exercise the LSE.

Report Deliverables:

Dominion will provide evidence of demonstration in the report in the form of:

1. Narratives of completed activities and milestones
2. Screenshots from testing and demonstration
3. Output data from testing/demonstration scenarios will be presented in appropriate formats in the report.

Major Remaining Milestones:

1. Preparation of Sample Data and Sample Model Package for Installation.
2. Advanced Testing of Topology Estimator Plumbing in LSE Core.
    a. Testing with Simulator (necessary to fully test features)
    b. Testing with Real-Time Data (a validation step)
3. Migration of Ancillary Components to openECA Client Analytics.
4. Deployment and Testing at Scale in Demonstration Environment at Dominion against simulated and real-time data. (As described above)
5. Finalize inclusion in openECA installer.

4 OSCILLATION DETECTION MONITOR

Test Approach

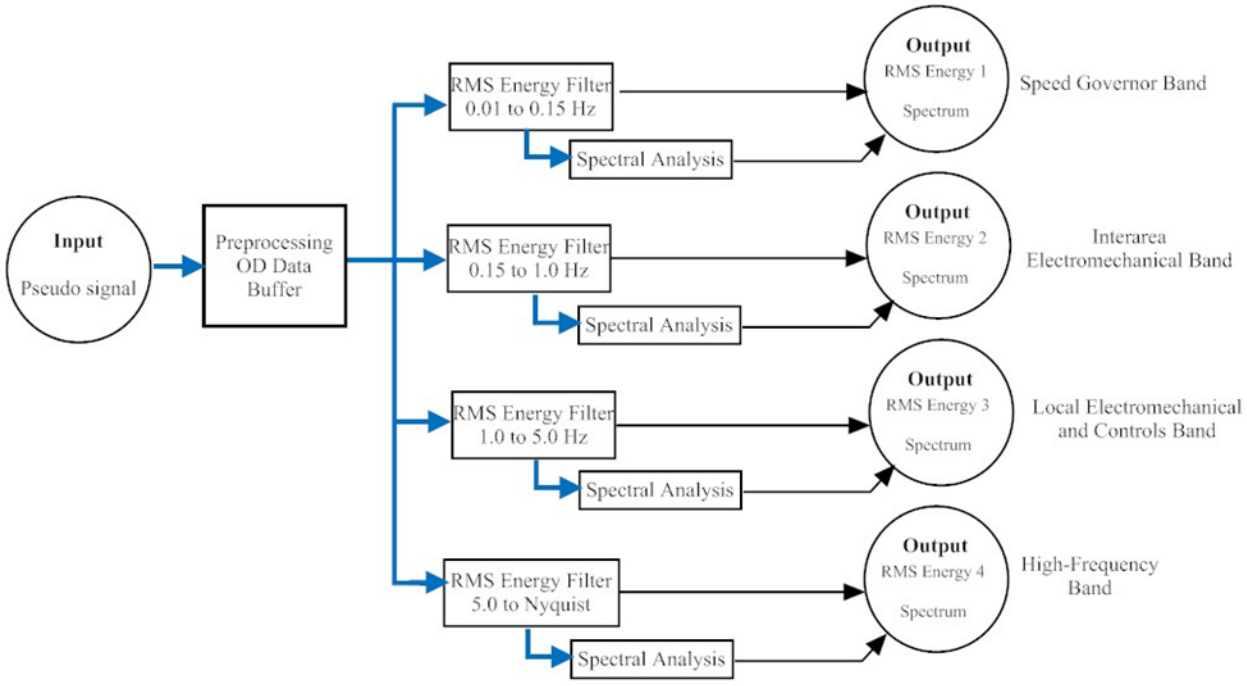
Two types of tests will be executed: "Comparison tests" will be developed to test the accuracy of the analytic's results when exposed to a known data set with known solution. "Unit tests" will be developed to test the repeatability of the analytic and to insure consistency of results when the analytic undergoes a future modification or upgrade. In addition to the fixed or simulated data sets, applied respectively to the "comparison" and "unit" tests, each test case requires parametric metadata for proper functionality. The parameters will be embedded within the analytic's code base for testing purposes. When "settings" facilities become available in openECA beta the test suite will be modified accordingly.

Test Environment

The test platform will be 64-bit Windows 7 configured as a workstation running on an Intel i7-3770 with 16Gb of memory and 8 cores. The software will be compiled under Visual Studio as "Any CPU" targeting the .NET Framework compatible with the openECA client compilation.

Analytics Overview

The OD analytic signal flow diagram is shown below. As the desired pseudo signal is formed, it is added to a data buffer with pre-processing in one-second blocks. When OD results are requested by the controlling application, 4 different RMS energies are output; each with a unique frequency band. The second output for each band is the spectrum of the RMS energy band-passed signal.



For each RMS energy band, parameters to be passed to the output include:

1. RMS energy in units of the pseudo signal.
2. Percent of invalid data used to calculate the RMS energy.

Pre-Beta Features to be Tested

The oscillation detector delivers the following outputs.

Feature 1: RMS energy in each of four frequency bands specified by user settings.

Feature 2: Percent of the data tested and considered by the analytic's bad data detector logic to be unusable by the analytic.

Feature 3: An array of frequency/energy pairs from which a partial spectrum of results can be obtained by the end user. This "spectrum" feature provides more granularity in the frequency domain than is possible by examining the energy content in the four primary frequency bins alone.

Test Configuration

No change to the bench setup described above.

Tests Conducted

Test 1: "Interpolate through invalid data within a one-second block" This test determines whether the analytic properly interpolates through invalid data when the invalid data is wholly contained within a one-second data block. This will be a unit test. With analytic loaded with valid configuration process a block of input signals containing flagged data points. Compare result with known solution.

Implementation: A unit test was created to test this requirement. The unit test instantiates an OscillationDetector object with simulated setup parameters. Consecutive blocks of data with varying "bad data" conditions are passed to the OD through the Load method.

Status: Complete. Test Passed.

Test 2: "Interpolate through invalid data spanning a one-second block" This test determines whether the analytic properly interpolates through invalid data when the invalid data spans two or more one-second blocks. This is a unit test. With analytic loaded with valid configuration process a block of input signals containing flagged data points at the end of the block followed by another block beginning with bad data. Compare result with known solution.

Implementation: A unit test was created to test this requirement. The unit test instantiates an OscillationDetector object with simulated setup parameters. Consecutive blocks of data with varying "bad data" conditions spanning the block boundaries are passed to the OD through the Load method.

Status: Complete. Test Passed.

Test 3: "Time gap less than 30 minutes fed to analytic should fill with invalid data" This test determines if the analytic properly handles a small time gap. A gap is caused by a timestamp greater than current time. This is a unit test. With analytic loaded with valid configuration process a block of input signals. Then send the next data block with a timestamp 60 seconds greater than previous. Analytic's timestamp should advance one minute and results containing bad data percentages should be reported.

Implementation: A unit test is under development. The unit test instantiates an OscillationDetector object with simulated setup parameters. The OD is pre-loaded with data, then gaps of durations varying from one second to 29 minutes are created.

Status: In process.

Test 4: "Time gap greater than 30 minutes fed to analytic should reset all buffers" This test determines if the analytic properly handles a large time gap. A gap is caused by a timestamp greater than current time. This is a unit test. With analytic loaded with valid configuration process a block of input signals. Then send the next data block with a timestamp 60 minutes greater than previous. Analytic's timestamp should advance one hour, no results should be reported. Subsequent data blocks input to the analytic should result in reports showing "Buffer not full" flag indicating buffers were emptied.

Implementation: A unit test is under development. The unit test instantiates an OscillationDetector object with simulated setup parameters. The OD is pre-loaded with data, then gaps of durations greater than 30 minutes are created.

Status: In process.

Test 5: "CurrentTime from analytic should be timestamp of last sample in the input block" This test determines if analytic properly updates current time. This is a unit test. With analytic loaded with valid configuration send an input data block with valid timestamp. Analytic's timestamp should be last sample of input timestamp.

Implementation: A unit test is under development. The unit test instantiates an OscillationDetector object with simulated setup parameters. The OD is pre-loaded with data. Timestamps of the incoming data are recorded. A result is retrieved from the OD, and timestamps are compared.

Status: In process.

Test 6: "OD estimate accurate and consistent" This tests OD accuracy. This is a fixed dataset test. Configure OD with setup parameters appropriate for a known solution. Run analytic and compare results to known solution.

Implementation: Create a dataset having a known solution. Inject the known solution dataset into the openECA framework as a data channel. Configure the OD analytic to consume the known dataset. Run the tests.

Status: In process. The dataset has been created. Efforts are underway to format the dataset in a format readable by openECA.

Test 7: "OD estimate for 30 sps derived signal accurate and consistent" This tests the OD downsampling filters. This is a fixed dataset test. Form analytic input signals from Test6 dataset at 30 sps sample rate. Compare results to known solution.

Implementation: Create a dataset having a known solution with sample rate 30sps. Inject the known solution dataset into the openECA framework as a data channel. Configure the OD analytic to consume the known dataset. Run the tests.

Status: In process. The dataset has been created. Efforts are underway to format the dataset in a format readable by openECA.

Test 8: "OD estimate for 60 sps derived signal accurate and consistent" This tests the OD downsampling filters. This is a fixed dataset test. Form analytic input signals from TestCase2 dataset at 60 sps sample rate. Compare results to known solution.

Implementation: Create a dataset having a known solution with sample rate 60sps. Inject the known solution dataset into the openECA framework as a data channel. Configure the OD analytic to consume the known dataset. Run the tests.

Status: In process. The dataset has been created. Efforts are underway to format the dataset in a format readable by openECA.

5 OSCILLATION MODE METER

Test Approach

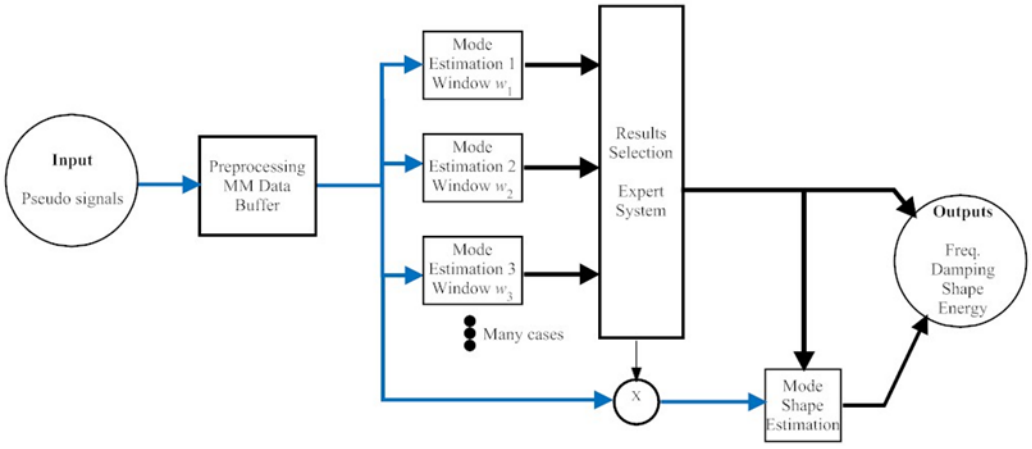
Two types of tests will be executed: “Comparison tests” will be developed to test the accuracy of the analytic’s results when exposed to a known data set with known solution. “Unit tests” will be developed to test the repeatability of the analytic and to insure consistency of results when the analytic undergoes a future modification or upgrade. In addition to the fixed or simulated data sets, applied respectively to the “comparison” and “unit” tests, each test case requires parametric metadata for proper functionality. The parameters will be embedded within the analytic’s code base for testing purposes. When “settings” facilities become available in openECA beta the test suite will be modified accordingly.

Test Environment

The test platform will be 64-bit Windows 7 configured as a workstation running on an Intel i7-3770 with 16Gb of memory and 8 cores. The software will be compiled under Visual Studio as “Any CPU” targeting the .NET Framework compatible with the openECA client compilation.

Analytics Overview

The signal flow diagram for the MM analytic is shown below. As pseudo signals are formed, they are added to a data buffer with pre-processing in one-second blocks. When mode estimates are scheduled to be provided to the openECA client, modes are estimated using several window sizes (w1, w2, …) and/or different algorithm settings; these are termed the Mode Estimation functions. A Results Selection function then analyzes the outputs of the parallel Mode Estimation functions to obtain the optimal mode damping and frequency estimation result . The estimated mode and pseudo data are then passed to a Mode Shape Estimation function which estimates the mode shape.



For each RMS energy band, parameters to be passed to the output include:
1. RMS energy in units of the pseudo signal.
2. Percent of invalid data used to calculate the RMS energy.

Pre-Beta Features to be Tested

The mode meter delivers the following outputs.

Feature 1: An estimate of the frequency of the most lightly damped mode identified in the range specified in the configuration.
Feature 2: An estimate of the damping for the mode identified in Feature 1.
Feature 3: The rms energy associated with the mode identified in Feature 1.
Feature 4: A mode shape vector describing the magnitude and angle of the oscillation energy associated with mode identified in Feature 1 at each of several buses specified in the configuration.
Feature 5: The percent of the data tested and considered by the analytic's bad data detector logic to be unusable by the analytic.

Test Configuration

No change to the bench setup described above.

Tests Conducted

Test 1: "Interpolate through invalid data within a one-second block" This test determines whether the analytic properly interpolates through invalid data when the invalid data is wholly contained within a one-second data block. This will be a unit test. With analytic loaded with valid configuration process a block of input signals containing flagged data points. Compare result with known solution.

Implementation: A unit test was created to test this requirement. The unit test instantiates a ModeMeter object with simulated setup parameters. Consecutive blocks of data with varying "bad data" conditions are passed to the MM through the Load method.

Status: Complete. Test Passed.

Test 2: "Interpolate through invalid data spanning a one-second block" This test determines whether the analytic properly interpolates through invalid data when the invalid data spans two or more one-second blocks. This is a unit test. With analytic loaded with valid configuration process a block of input signals containing flagged data points at the end of the block followed by another block beginning with bad data. Compare result with known solution.

Implementation: A unit test was created to test this requirement. The unit test instantiates a ModeMeter object with simulated setup parameters. Consecutive blocks of data with varying "bad data" conditions spanning the block boundaries are passed to the MM through the Load method.

Status: Complete. Test Passed.

Test 3: "Time gap less than 30 minutes fed to analytic should fill with invalid data" This test determines if the analytic properly handles a small time gap. A gap is caused by a timestamp greater than current time. This is a unit test. With analytic loaded with valid configuration process a block of input signals. Then send the next data block with a timestamp 60 seconds greater than previous. Analytic's timestamp should advance one minute and results containing bad data percentages should be reported.

Implementation: A unit test is under development. The unit test instantiates a ModeMeter object with simulated setup parameters. The MM is pre-loaded with data, then gaps of durations varying from one second to 29 minutes are created.

Status: In process.

Test 4: "Time gap greater than 30 minutes fed to analytic should reset all buffers" This test determines if the analytic properly handles a large time gap. A gap is caused by a timestamp greater than current time. This is a unit test. With analytic loaded with valid configuration process a block of input signals. Then send the next data block with a timestamp 60 minutes greater than previous. Analytic's timestamp should advance one hour, no results should be reported. Subsequent data blocks input to the analytic should result in reports showing "Buffer not full" flag indicating buffers were emptied.

Implementation: A unit test is under development. The unit test instantiates a ModeMeter object with simulated setup parameters. The MM is pre-loaded with data, then gaps of durations greater than 30 minutes are created.

Status: In process.

Test 5: "CurrentTime from analytic should be timestamp of last sample in the input block" This test determines if analytic properly updates current time. This is a unit test. With analytic loaded with valid configuration send an input data block with valid timestamp. Analytic's timestamp should be last sample of input timestamp.

Implementation: A unit test is under development. The unit test instantiates a ModeMeter object with simulated setup parameters. The MM is pre-loaded with data. Timestamps of the incoming data are recorded. A result is retrieved from the MM, and timestamps are compared.

Status: In process.

Test 6: "Mode estimate accurate and consistent. This test considers mode meter accuracy. This is a fixed dataset test. Monte Carlo tests will be performed outside of the openECA environment. Within the openECA, one test on one dataset will be conducted. If the result matches the offline test using the same dataset, and the Monte Carlo tests pass the accuracy threshold, then the tests passes.

Implementation: Run the Monte Carlo tests in Matlab. Create a dataset representing one of the Monte Carlo test sets. Inject the known solution dataset into the openECA framework as a data channel. Configure the MM analytic to consume the known dataset. Run the tests.

Status: In process. The Monte Carlo tests are complete. The ModeMeter analytic passed the accuracy tests and met the design parameters. The single dataset, picked from the Monte Carlo test set, has been selected. Efforts are underway to format the dataset in a format readable by openECA.

Test 7: "Mode shape accurate and consistent. This test considers mode shape accuracy. This is a fixed dataset test. Monte Carlo tests will be performed outside of the openECA environment. Within the openECA, one test on one dataset will be conducted. If the result matches the offline test using the same dataset, and the Monte Carlo tests pass the accuracy threshold, then the tests passes.

Implementation: Run the Monte Carlo tests in Matlab. Create a dataset representing one of the Monte Carlo test sets. Inject the known solution dataset into the openECA framework as a data channel. Configure the MM analytic to consume the known dataset. Run the tests.

Status: In process. The Monte Carlo tests are complete. The ModeMeter analytic passed the accuracy tests and met the design parameters. The single dataset, picked from the Monte Carlo test set, has been selected. Efforts are underway to format the dataset in a format readable by openECA.

Test 8: "Mode estimate accurate for ambient data at 30 sps raw rate" This tests the mode meter downsampling filters. This is a fixed dataset test. Form analytic input signals from Monte Carlo baseline case at 30 sps sample rate. Compare results to known solution.

Implementation: Run the Monte Carlo tests in Matlab. Create a dataset representing one of the Monte Carlo test sets at 30sps. Inject the known solution dataset into the openECA framework as a data channel. Configure the MM analytic to consume the known dataset. Run the tests.

Status: In process. The Monte Carlo tests are complete. The ModeMeter analytic passed the accuracy tests and met the design parameters. The single dataset, picked from the Monte Carlo test set, has been selected. Efforts are underway to format the dataset in a format readable by openECA.

Test 9: "Mode estimate accurate for ambient data at 60 sps raw rate" This tests the mode meter downsampling filters. This is a fixed dataset test. Form analytic input signals from Monte Carlo baseline case at 60 sps sample rate. Compare results to known solution.

Implementation: Run the Monte Carlo tests in Matlab. Create a dataset representing one of the Monte Carlo test sets at 30sps. Inject the known solution dataset into the openECA framework as a data channel. Configure the MM analytic to consume the known dataset. Run the tests.

Status: In process. The Monte Carlo tests are complete. The ModeMeter analytic passed the accuracy tests and met the design parameters. The single dataset, picked from the Monte Carlo test set, has been selected. Efforts are underway to format the dataset in a format readable by openECA.

6 TOPOLOGY ESTIMATOR

Program Details and User Manual for Alpha Version Testing

The whole process for implementation of the Topology Estimator algorithm is based upon two modules:

1. An offline module using PSS/E and Python to calculate the Delta threshold by running numerous power flow simulations based upon different configurations for each substation topology.
2. An online module which would use the pre calculated Delta Threshold value and use it to actually determine the topology of the substations in real time.

For the Alpha testing we had tested our algorithm for a IEEE 118 bus branch model and had run studies to calculate the Delta threshold value (for the offline module). (Note: The files and folders have been uploaded in Github account earlier.)

Initialization and Running

Note: This program is written in Python 2.5 and compatible with PSS/E version 32

There are several actions that need to be taken before running the program to insure that the program is configured properly.

1. Ensure that the PSS/E .sav file is within the same folder as the python code files

2. Rename '.sav' file within the main function to the name of the file you wish to use

```
12  #----------------------------------------------------------------------------#
13  # MODULES                                                                    #
14  #----------------------------------------------------------------------------#
15  import pssepy, numpy as np
16  import operator, pdb
17
18  Sav = 'ieee118bus_PSSEcorrected_rq.sav'
19  #Sav = 'IEEE14bus.sav'
20
```

3. Change the path name within the name function to the name of the folder containing the python files

```
21  #----------------------------------------------------------------------------#
22  # USER PATHS                                                                 #
23  #----------------------------------------------------------------------------#
24  PathSav = installPath + '\\TopologyEstimator'       # Path for .sav
25
```

4. If PSS/E is installed in a location other than C:\Program Files (x86), modify the variable: pssebindir to direct the program to the correct location(the \PTI\PSSE32\\PSSBIN)

```
5   #----------------------------------------------------------------------------#
6   # NEEDED PATH APPENDICES                                                     #
7   #----------------------------------------------------------------------------#
8   sys.path.append(Library)
9   pssebindir = "C:\Program Files (x86)\PTI\PSSE32\\PSSBIN"
10  os.environ['PATH'] = pssebindir + ';' + os.environ['PATH']
11  sys.path.insert(0,pssebindir)
```

Once the program has been configured to your system, it can be run.

Note: Depending on the UI used for running python code, after the program has started running it may ask for a command simply saying "Yes?". If this is the case, simply enter a blank command.

The program outputs the list of voltage angle and magnitude lists to a .csv file named "mycsv.csv". From this data, the delta and magnitude threshold can be determined and the percentage error can be seen. An example can be seen in the section 4 for the IEEE 118 bus system model.

The following sections detail the algorithm behind each function for the program.

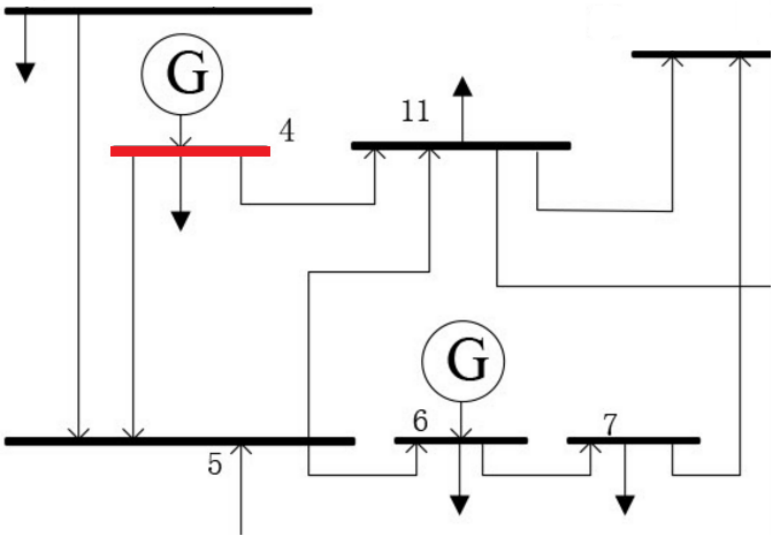
Main Function

The main function inside which we first start the analysis of delta threshold from a particular bus number to the desired bus number. At each bus number, we find the number of circuits/elements connected by calling the bus connections function and then calculate all the possible configurations of disconnection combinations possible by calling the configuration matrix function. Then for each element in the configuration matrix list (i.e. each possible combination) we calculate the outage buses and their groupings which is the outage matrix list and then we update the network by rearranging the loads/Generators/Lines/2 winding Transformers as required, creating an additional fake bus as required. This is achieved by calling the update network function. Finally load flow is rum for each case and Difference in angles and V_mag is calculated between the current bus number and the disconnected fake bus, thus leading to the study of a suitable Delta threshold which can guarantee minimum number of failed cases based on this methodology.

Bus Connections

The purpose of this function is to calculate the number of objects connected to a specific input bus and to determine each of those objects. The output will give you a number of connections and a list contained each bus connected. Additionally, if the bus has a generator and additional connection of its own bus number is added. If there is a load present at the bus, an additional connection with the value of 100000000 is added.

For example, when looking at the following bus number 4.

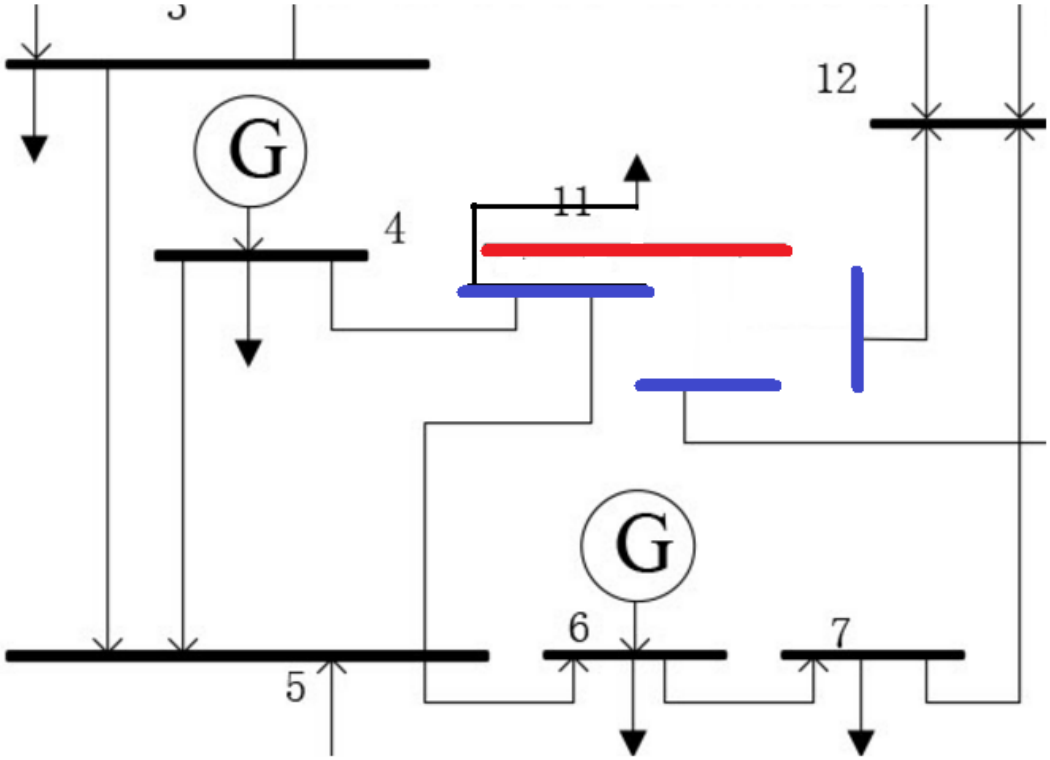


It can be seen that, bus 4 is connect to two other buses, one generator and one load. Therefore, there are 4 connections for this bus and the list would be [5,11,4,10000000].

Update Network

This function modifies the current system depending on the current outage scenario given to the function. Outage matrix provides the current connection list of the buses surrounding the current bus to the current bus and between each other. The network is updates so that each bus within the same outage group is connected to each other through a newly created bus. Any bus in outage group 0 is connected to a newly created bus individually. Branches connecting the current bus and the buses connected to this bus that are in the outage group are taken out of service and new branches are created connecting the outside buses and the newly created buses.

For example, when looking at Bus 11 in the previous figure, if buses 4, 5, and the load are in group two and all other buses are in group 0 the new system created will appear like this:



Any buses connected to the current bus which are not in the outage matrix remain connected to the current bus.

Truthtable

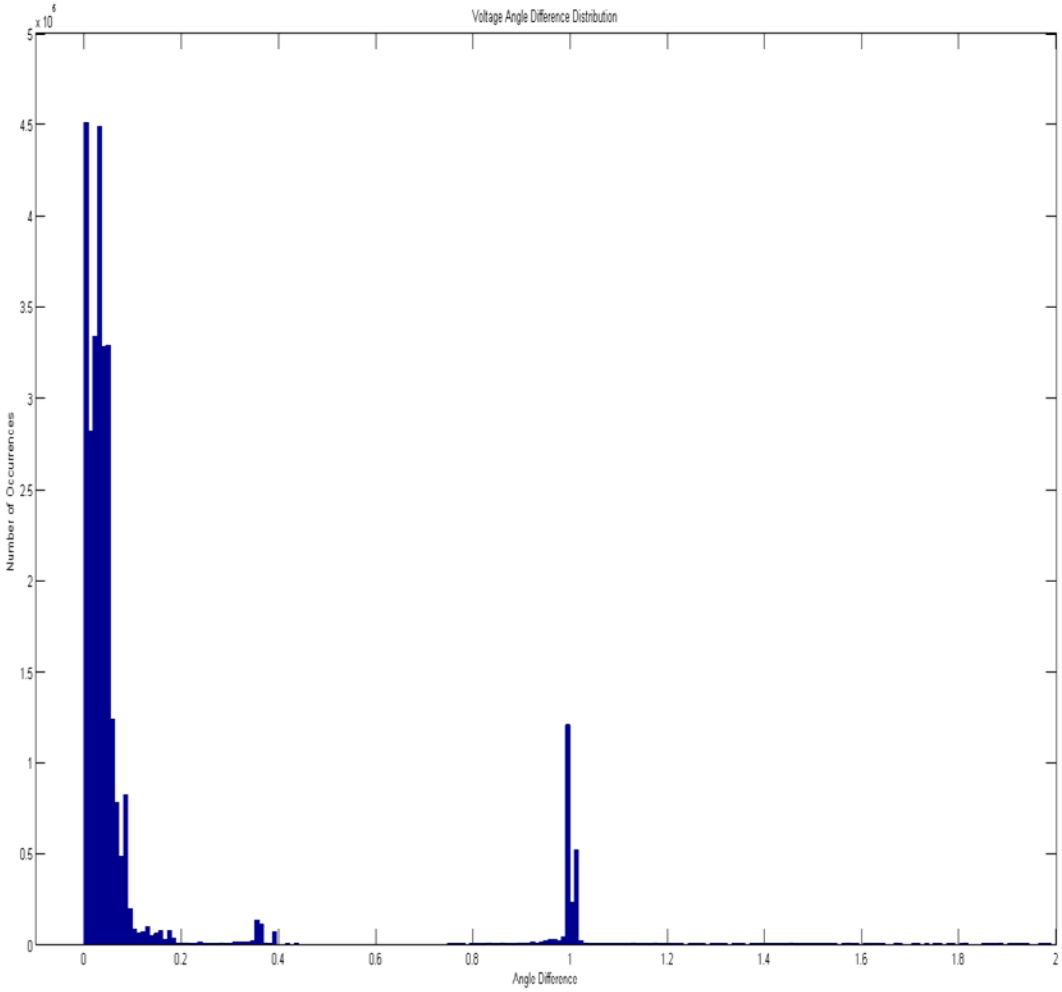
The purpose of this function is to create a truth table consisting of 0's and 1's which will be used in the configuration function later. This provides the possible number of configurations based on the variable n in a list format consisting of 2^n elements inside the final temp list.

Configuration

The purpose of this function is to create all possible configurations of the disconnected elements or circuits (branch, Gen, loads) which were initially attached to our current substation. Now the original configuration list would have been simply the list as provided by the truth table function. But as the disconnected (from the substation) circuits or elements can be regrouped among each other we have to cater all the possibilities for our study and thus come up with a generic yet comprehensive Delta threshold. This function is to generate all those possible configurations (much greater than simply 2^contingency).

Demonstration

All the possible configurations (with a maximum disconnection of 6 elements at a particular bus) were analyzed and the following distribution of Voltage Angle difference distribution is plotted but based upon a success rate of near about 96% of all cases, Delta angle threshold was calculated to be 0.75 degrees (with max V_mag difference of 0.02 pu).



From Alpha to Beta

In the Beta version, the analytic will be running on openECA platform while the input measurements will be simulated using real time PMU simulator. This section presents the preliminary results without real time simulator and using simulated data as real time phasors from a IEEE 118 nodal test system (Files uploaded in the GitHub Folder).

The test system that was used for these results is the IEEE 118-Bus system which has been converted from a bus/branch model to a nodal model. Each bus in the model was transformed into a substation with one of four standard topologies: Double Breaker-Double Bus, Ring Bus, Breaker and a Half, and Single Bus. This is a significant modification from the Alpha testing as we are catering to actual practical substation topologies in this fashion. The original system buses were assigned specific substation topologies sequentially, starting with Double Breaker-Double Bus continuing sequentially through Ring Bus, Breaker and a Half, and then Single Bus, finally looping back. Every connection to the original bus (branches/loads/generators) was connected to a separate node within the substation.

Different Modules:

CreateConfigFile.py
To make the analytic more general, many functions make use of a configuration file which provides details of the nodes contained within each individual substation and the breakers present connecting the nodes. This can be done manually, or if the system was created following the specified format assumed for the remaining functions, the function CreateConfigFile.py can be run. The systems were created in PSS/E such that each node in the system is a bus, where all nodes within the substation follow the naming convention of SUBSTATIONNAME_ID, where ID is what identifies the nodes within the substation. Nodes within the substation are connected with system switches. The format of the configuration file is shown below and is stored in **ConfigFile_118.csv:**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	8	1	3	1	1	4	1	1	5	1	1	6	1	2	3	1
2	3	7	8	1	7	9	1	8	9	1						
3	6	10	12	1	10	14	1	11	13	1	11	15	1	12	13	1
4	4	16	17	1	16	18	1	16	19	1	16	20	1			

Where each row details an individual substation. The first column indicates the number of breakers within the substation. Each of the following set of 3 columns gives information on the breaker in the form of [From Node, To Node, Breaker ID] e.g. the first breaker of the second substation is from Node 7 to Node 8 with ID 1. The second breaker is from node 7 to node 9 with ID 1.

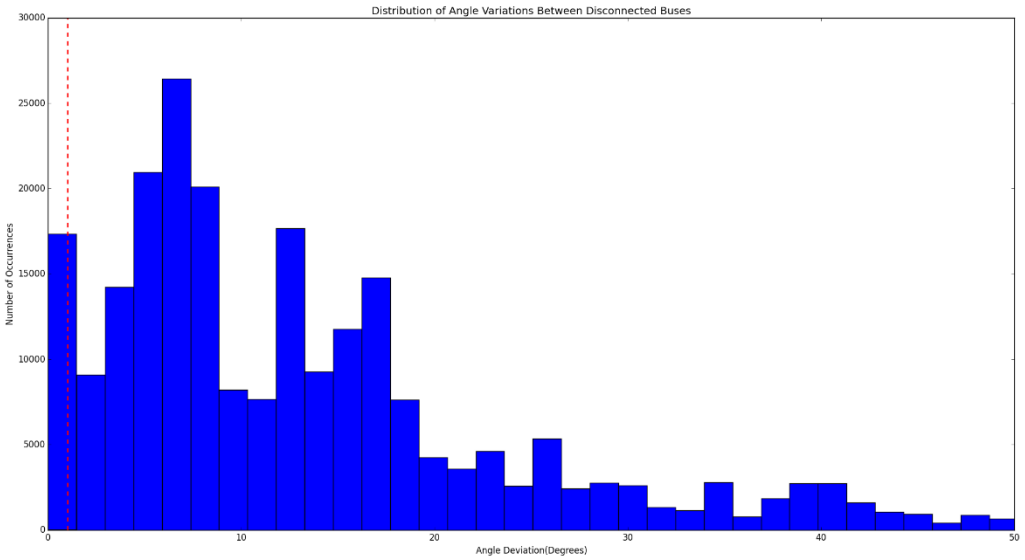
VarThreshCalc.py:
Offline Results for Threshold Benchmarking:

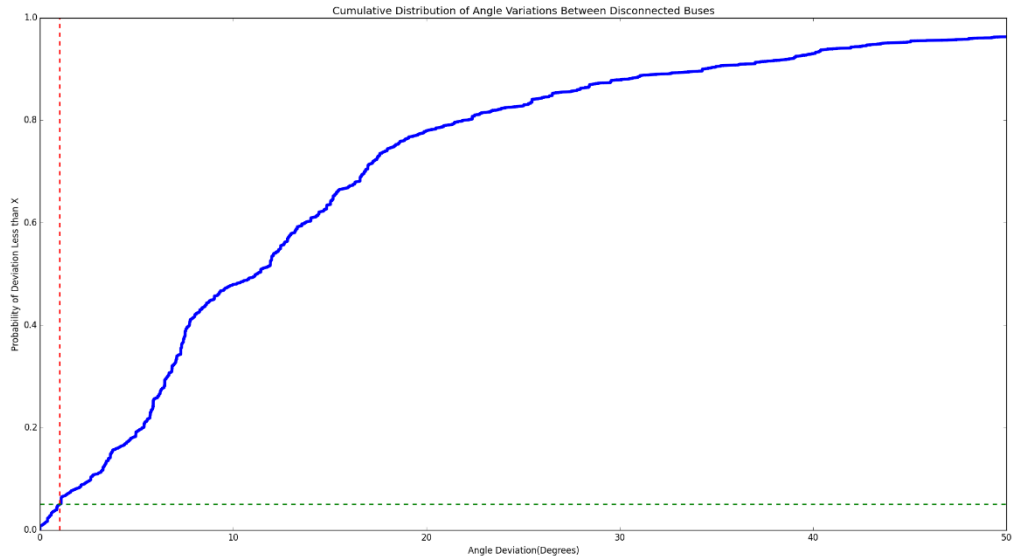
This portion of the analytic is implemented in the VarThreshCalc.py function. The purpose of this function is to empirically determine the voltage value difference between nodes at which the nodes can be considered disconnected. Every possible breaker on/off configuration was tested for the entire system and the values across each disconnected breaker were recorded. In order to obtain an accurate, but still practical threshold value the threshold was selected for the case when 95% of the disconnected values were above the threshold. The algorithm calculates and outputs in batches as to avoid memory overflows. The batch size can be tuned to fit best fit your computer specifications.

The function outputs a file called Outputs_Disc#.csv, where # is the current batch number.

28	[2, 'SERENITY_B2 ', 5, 'SERENITY_C ']	(1, 1, 1, 1, 0, 0, 0, 1)	0	0
29	[1, 'SERENITY_B1 ', 6, 'SERENITY_D ']	(1, 1, 1, 0, 1, 1, 1, 1)	0	0
30	[1, 'SERENITY_B1 ', 6, 'SERENITY_D ']	(1, 1, 1, 0, 1, 1, 1, 0)	6.427395	0.056267
31	[2, 'SERENITY_B2 ', 6, 'SERENITY_D ']	(1, 1, 1, 0, 1, 1, 1, 0)	6.427395	0.056267
32	[1, 'SERENITY_B1 ', 6, 'SERENITY_D ']	(1, 1, 1, 0, 1, 1, 0, 1)	0	0

The first column the contains information about which breaker and substation the data is from. The second column details the current breaker status on/off configuration of the substation. The 3rd and 4th column show the voltage angle and magnitude variation across the breaker respectively. For the test system, if this data is plotted the results can be seen below in the form of a histogram of angle differences and an empirical CDF.





The results for the test system indicate an angle threshold of about 1.0 degrees. This result is large enough such that most measurement errors will not have significant impacts on the estimation. There are still many cases were disconnected breakers will have a variation less than this value, but most cases are captured correctly.

The Core Algorithm for Determining Bus-Branch Topology:

As of now, the algorithm has been implemented in python, but is still being worked upon the conversion of the code into C# for use with the openECA platform. The algorithm currently makes use of simulated power flows to check the accuracy of the algorithm with the help of following modules in python.

GenerateData.py
Data is generated through the GenerateData.py function, this function generates many power flow cases with separate configurations and applies measurement errors to the data. The outputs of this function are Voltages.csv,

Voltages_1.csv, Voltages_3.csv, and gendata_outputs.csv. Voltages contain the true voltage measurements, where the 1 and 3 indicates the total vector error percentages applied to the data following a normal distribution to emulate real time phasors with errors. gendata_outputs.csv contains system information including the breaker information and the system load and which substations configuration was altered.

DetConnec.py
The core algorithm of this analytic makes use of the threshold previously calculated to determine the topology in the system. The functions which determine the connections within a substation are implemented as functions in DetConnec.py, which are called in the main function of

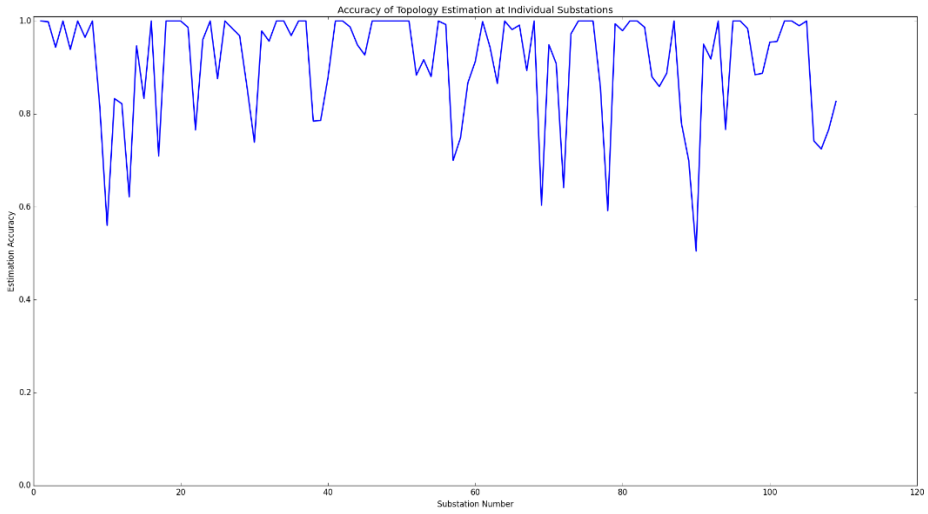
TopologyEstimation.py.
DetConnec.py contains two functions, DetConnec_Real which determines the actual connections in the system and DetConnec_Est which is the algorithm that estimates the value.
The algorithm follows these follow steps:

Step 1: Check if the node is energized or not based on the Voltage at the node
Step 2: Check for available breaker telemetry and assigned breaker statuses
Step 3: Use Voltage values to infer missing breaker statuses or to check existing breaker status based on threshold previously calculated
Step 4: Construct bus/branch model from nodes using resultant breaker statuses

The results are output into EmpiricalEstimation.csv. The results are displayed for each row, where a 1 indicates a successful estimation and 0 indicates an unsuccessful topology estimation. The accuracies of the algorithm for the test system are:

Standard Accuracy = 0.927620708647
1% Error Accuracy = 0.918097117679
3% Error Accuracy = 0.572861208004

The plots for each individual substation's errors are shown below:



7 REGIONAL VOLTAGE CONTROL

The detailed concept of this analytic is illustrated in Figure 1. As shown, the analytic is divided into two parts: online and offline. The offline adapter is implemented to create/update decision trees based on EMS data snapshot. When the tree is created/updated, they are mapped into the online adapter which is running as a module in openECA. Synchrophasor measurements will fall into the tree and provide VSA for each control combinations.

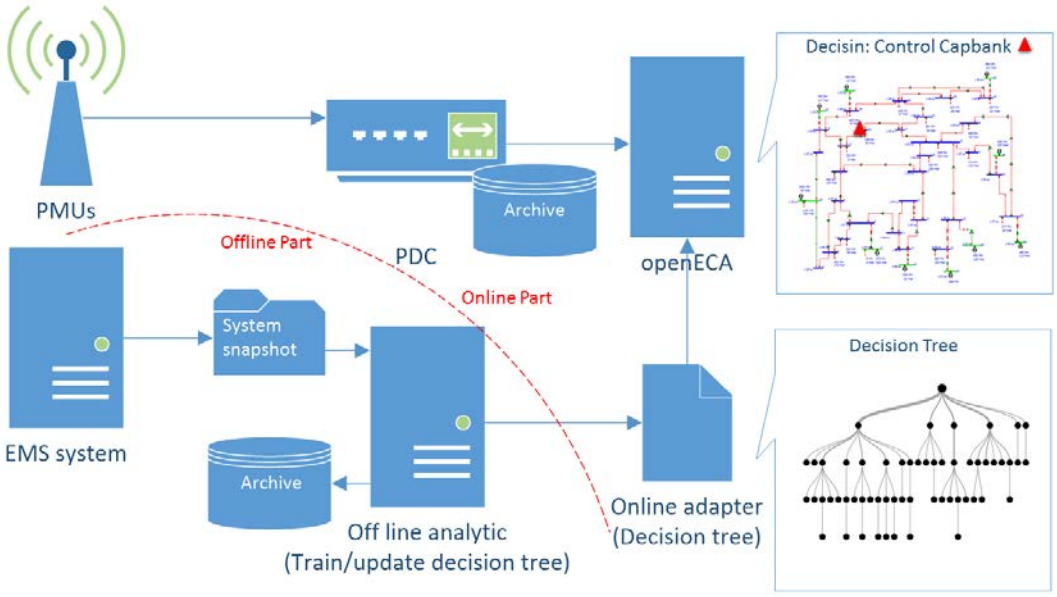


Figure 1 Regional Voltage Controller Analytic Concept

Test 1: Parallel Trees

Code

*Under folder **Test1\ TrainingCaseCreation***
*Run **IEEE118_voltage_violation_load_genchange_measurements.py** to create measurements: **databaseMeasurements***

*Run **IEEE118_voltage_violation_load_genchange_LabelsOC_CapbankSwitch.py** to create labels for all 64 control combinations (index 0 represents there is no control)*
*For example: **database_OC_PostControl_0.csv***

*Copy 64 **database_OC_PostControl_#.csv** files to folder : **Test 1\ParallelDecisionTrees***

*Under Folder: **Test 1\ParallelDecisionTrees***
*Change **dir** for both MATLAB script **Main_CreateDatabase.m** and **Main_CrossValidationDecisionTree.m***
*Run **Main_CreateDatabase.m** to create OCs*
*Run **Main_CrossValidationDecisionTree.m** to do crossvalidation*

Base Case and OC Generation

The IEEE 118 bus system is used for case study. The system is divided into 3 areas. Load buses within each area are assumed to have the same loading pattern that the load is scaled up and down in the same percentage. The generator is re-dispatched the same amount of the load as the load changed within the same area. The base case is generated by scaled down 5 percent of the total load.

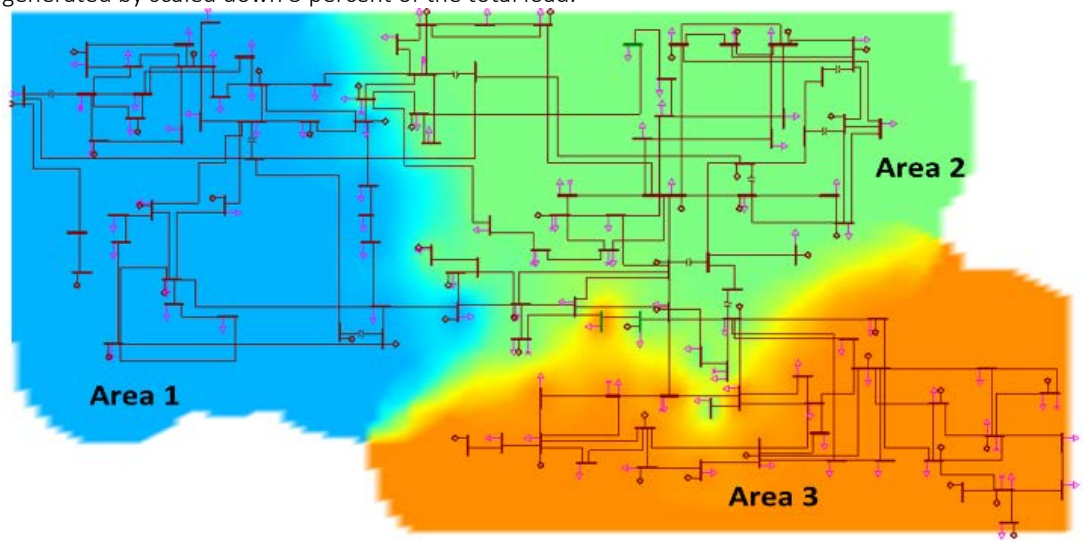


Figure 2 IEEE 118 bus system

Voltage magnitudes at all buses are selected for learning database generation. In this case, the control options are fixed capacitor banks only which are located at buses 34, 44, 45, 48, 74, and 105. In the initial condition, all selected fixed capacitor banks are switched off.

Table 1 Capacitor Bank Available for Control

Bus Number	Capacity of Capacitor Bank (MVar)
34	50
44	50
45	50
48	100
74	100
105	20

Overall, 25000 OCs are generated by scaling up the loads within 100% - 150% of their base case value for each area. The outputs of generators re-dispatch the same amount of load in the same area. VSA is implemented to determine the secure and insecure OCs. In this work, it is assumed that the load capacity limit and the secure operation limit mentioned in section 2 are overlapping. The unstable OCs are removed from the initial database, since they cannot provide useful information about the system condition. For all of these secure/insecure OCs, 60% of them are used for training while the rest of them are reserved for periodic update and testing. The initial trees are trained offline. For example, in the database for switching cap bank at 44 on, the number of secure and insecure OCs are shown in Table 2.

Table 2 Number of secure/insecure OCs

OC	Training	Testing and Update
Secure	9948	6615
Insecure	2199	1483

Their cross validation accuracies for all 63 control combinations are shown Figure 3. As it can be seen, the low error rates of cross validation indicate that the trained tree is able to provide accurate VSA for each control decision.

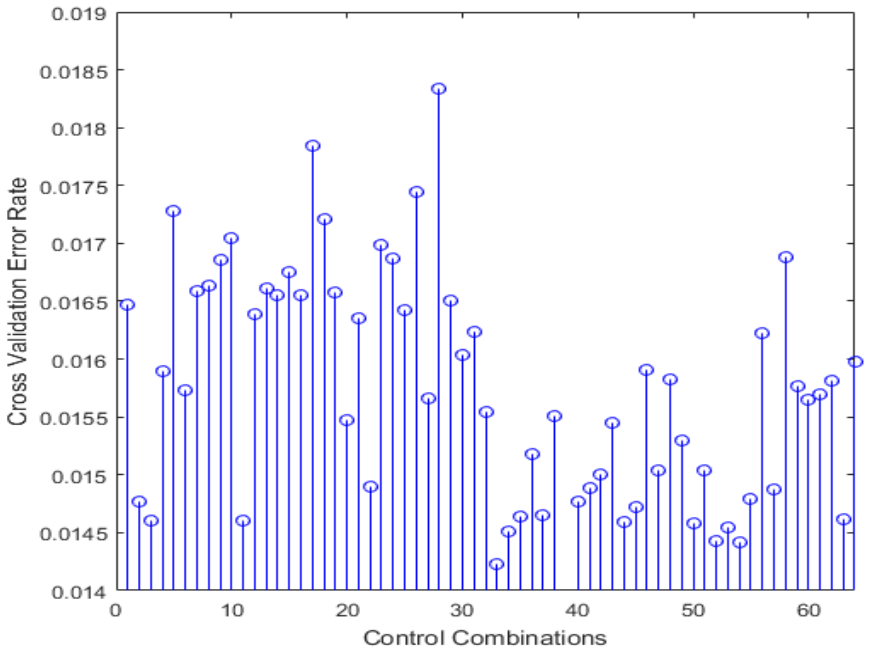


Figure 3 Cross-validation error rate

Test 2: Online Boosting

Code

Under folder ***Test2\ TrainingCaseCreation***
Run ***IEEE118_voltage_violation_load_genchange_measurements.py*** *to create measurements*

Run ***IEEE118_voltage_violation_load_genchange_LabelsOC_CapbankSwitch.py*** *to create labels for all 64 control combinations (index 0 represents there is no control)*

Copy the csv files: ***database_OC_PostControl_16.csv*** *and* ***databaseMeasurements_topologyChange.csv*** *to folder:* ***Test2\OnlineBoosting***

Copy the .mat files: ***xtrain.mat, xtest.mat,ytrain.mat,and ytest.mat*** *from folder* ***Test1\ParallelDecisionTrees\case_16*** *to folder* ***Test2\OnlineBoosting***

Under folder ***Test2:OnlineBoosting***
Run ***Main_CreateDatabase.m*** *to create post-topology change test and training data as* ***xtest_afterTopologyChange.mat, xtrain_afterTopologyChange.mat, ytest_afterTopologyChange.mat,*** *and* ***ytrain_afterTopologyChange.mat***

Run ***Main_initializeTreeStumps.m*** *to create* ***TreeStumps.mat***
Run ***Main_onlineBoost.m*** *to update the Adaboost trees*
(To shorten the simulation time, the variable in ***Main_onlineBoost.m****:* ***NumberofDataforTreeUpdate*** *is set as 1000, The maximum you can set is 4000)*

Periodic Update Using Online Boosting

In this section, control decision by switching on capacitor bank at bus 44 is selected for classifier performance evaluation. The initial tree is trained based on the offline Adaboost method [9] incorporated with 30 weak learners, and the number of selectors is also 30.

Transmission line between bus 15 and 33 is tripped on the test system. New training cases and test cases are created using the proposed approach in the previous sections but with a different system topology. 4000 of these new cases are used for the periodic update, and another 4000 of them are reserved for online validation. Among these new cases, 82% of them are secure OCs while the rest of them are insecure OCs. The performance of online boosting approach is evaluated by comparing it with single decision tree training using default MATLAB tree training. The computation time and misclassification error rate are recorded and illustrated in Figure . The online boosting scheme turns out to be more accurate than single DT training while the computation time spent by online boosting

for tree update is much less than re-training tree from scratch. The computation is run under the environment of MATLAB on a workstation with Intel Core i7-4790 3.6 GHz CPU and 32 GB memory.

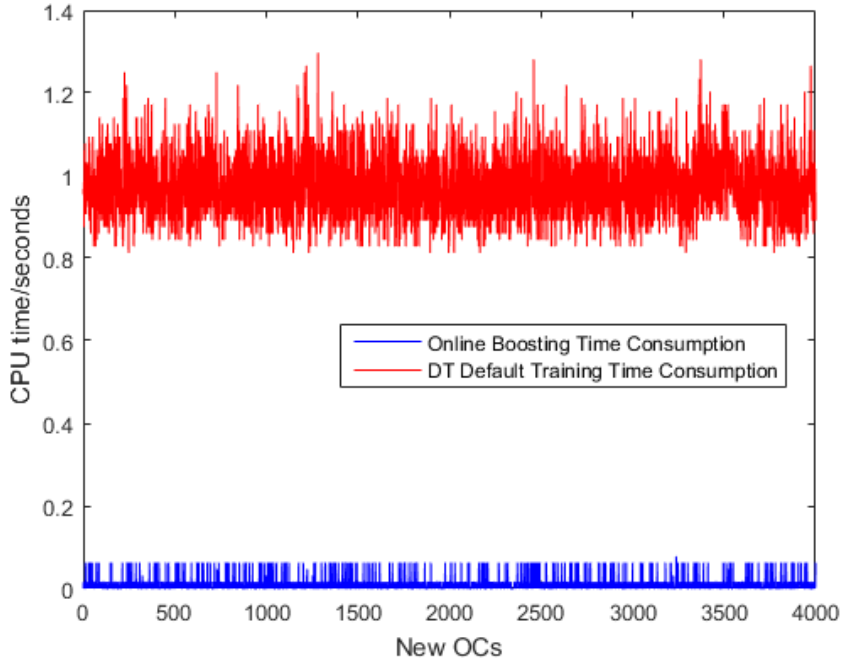


Figure 4 (a) Computation time for tree update

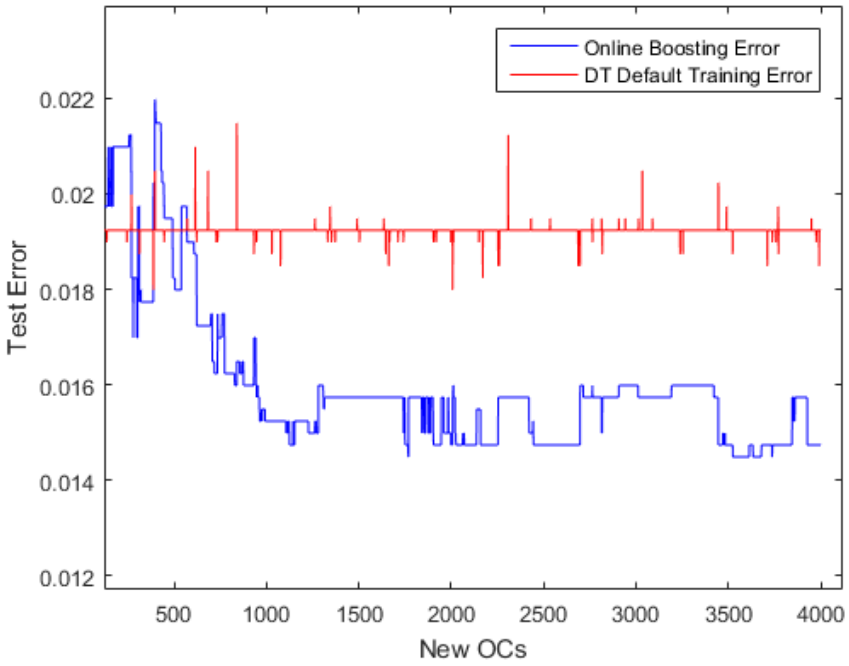


Figure 4 (b) Test error rate for online boosting and single DT training

From Alpha to Beta

In the Beta version, the Regional Voltage Controller analytic will be running on openECA platform while the input measurements will be simulated using real time PMU simulator. This section presents one of the test cases to be conducted in the Beta version and the preliminary test result without real time simulator (the real time simulator will be available soon). The objectives of this test case is to verify if the controller can **identify** the insecure voltage operating condition and switch **ON** the **minimal** capacitor banks to control the system.
In this test, the regional voltage controller analytic is completely converted into C# code. According to the control logic of the analytic, the control decision is generated based on the measurement streamed from the openECA platform. Each the control decision for one frame is logged as an *.xml file in the "Log" folder in the main analytics solution directory.

Figure 5 (a) – (c) show the result of voltage magnitude on the buses of capacitor banks using PSS/E to simulate the regional voltage control under a load-increasing circumstance of the system. All the capacitor banks are switched **off** and are available for further control. For each frame, the regional voltage controller adapter initially assess the security status based on the voltage measurement values from all buses. When an insecure voltage status is detected, the controller returns a control decision to switch on the necessary capacitor bank; for instance, the "CtrlDecisionMessage" shows that the CapBank #6 is switched on at the frame 2 in Figure 5 (a), so on and so forth for the following 2 control decisions.

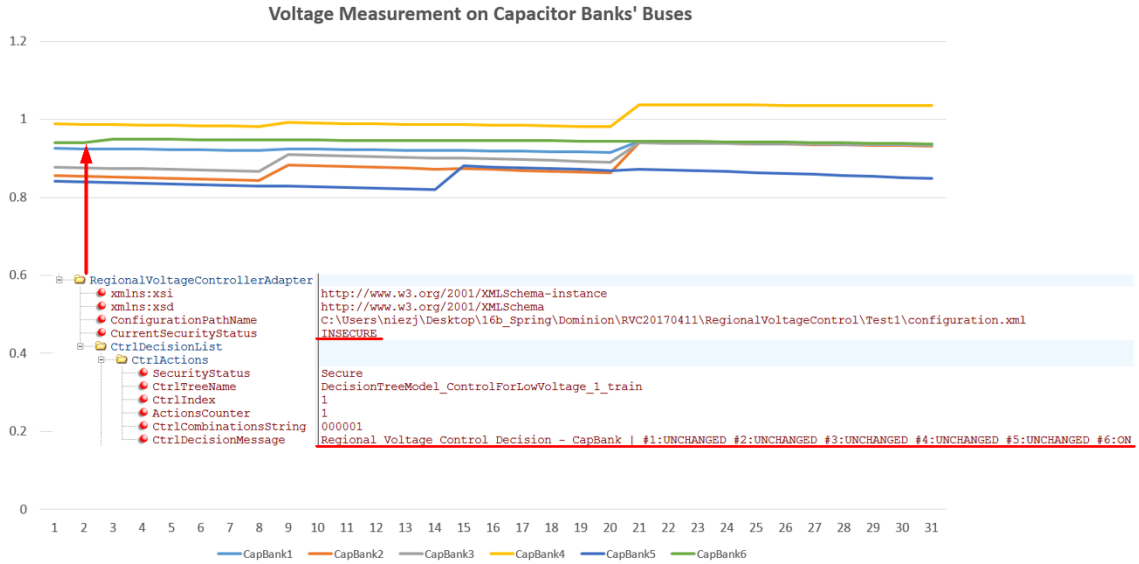


Figure 5 (a) Test Case of Regional Voltage Controller – Control Decision I

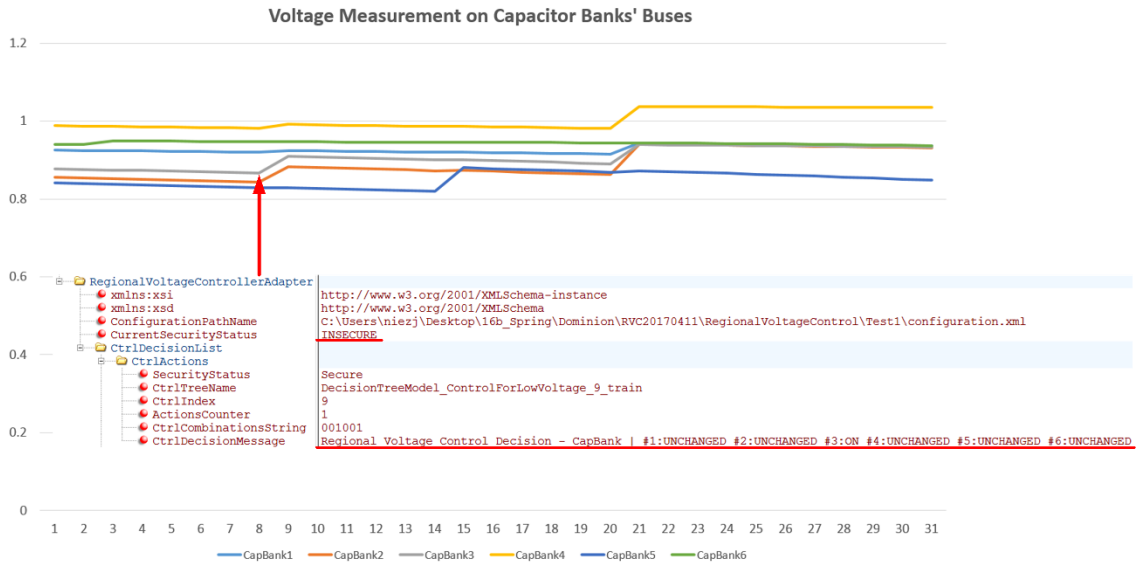


Figure 5 (b) Test Case of Regional Voltage Controller – Control Decision II

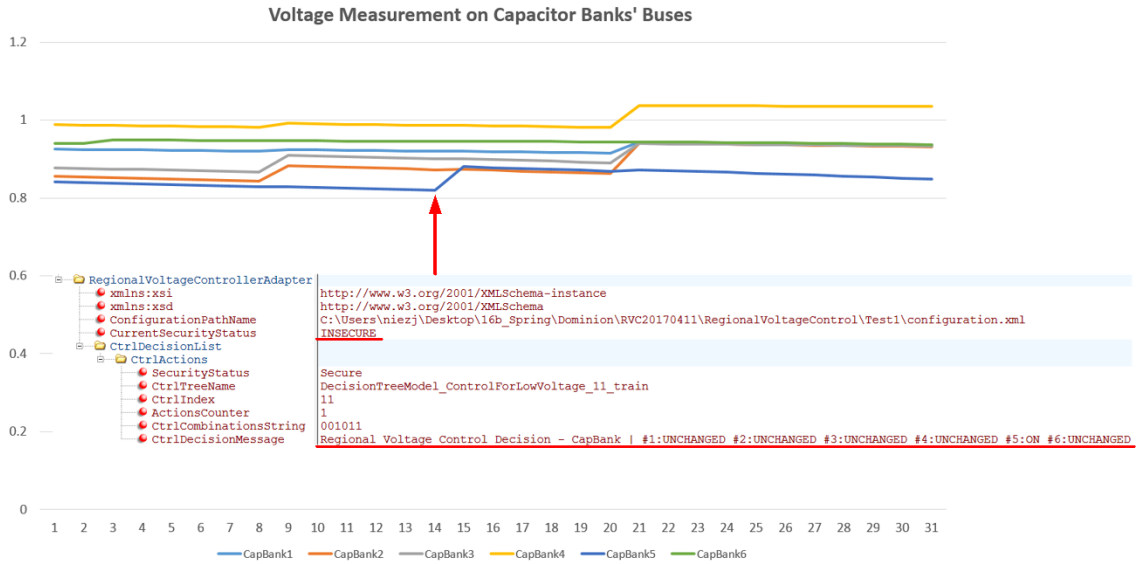


Figure 5 (c) Test Case of Regional Voltage Controller – Control Decision III

Reference

[1] E. E. Bernabeu, J. S. Thorp and V. Centeno, "Methodology for a Security/Dependability Adaptive Protection Scheme Based on Data Mining," *IEEE TRANSACTIONS ON POWER DELIVERY* , vol. 27, no. 1, pp. 104-111, 2012.

[2] L. Chengxi, "A systematic approach for dynamic security assessment and the corresponding preventive control scheme based on decision trees," *Power Systems, IEEE Transactions,* vol. 29, no. 2, pp. 717-730, 2014.

[3] D. Ruisheng, S. Kai, V. Vijay, O. R. J, R. M. R, N. Bhatt, S. Dwayne and S. S. K, "Decision Tree-Based Online Voltage Security Assessment Using PMU Measurements," *IEEE TRANSACTIONS ON POWER SYSTEMS*, vol. 24, no. 2, pp. 832-839, 2009.

[4] P. Kessel and H. Glavitsch, "Estimating the Voltage Stability of a Power System," *IEEE Transaction on Power Delivery,* Vols. PWRD-1, no. 3, pp. 346-354, 1986.

[5] S. Shukla and L. Mili, "A hierarchical decentralized coordinated voltage instability detection scheme for SVC," in North American Power Symposium , Charlotte, NC, USA, 2015.

[6] B. Leo, J. Friedman, C. Stone and R. Olshen, Classification and regression trees, CRC press, 1984.

[7] H. a. H. B. Grabner, "On-line boosting and vision," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.
[8] M. He, Z. Junshan and V. Vijay, "Robust online dynamic security assessment using adaptive ensemble decision-tree learning," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4089-4091, 2013.
[9] D.-J. Kroon, "MathWorks," 01 Jun 2010. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/27813-classic-adaboost-classifier. [Accessed 1 11 2016].

8 LOCAL VOLTAGE CONTROL

The demonstration is conducted based on two platforms: PSSE and C# as shown in Figure 2. The PSSE model is simulating the power system that provides measurements as input signals for the voltage controller written in C#. When the logic is triggered inside the voltage controller, the control signal will be sent back to PSSE and execute the control decision.

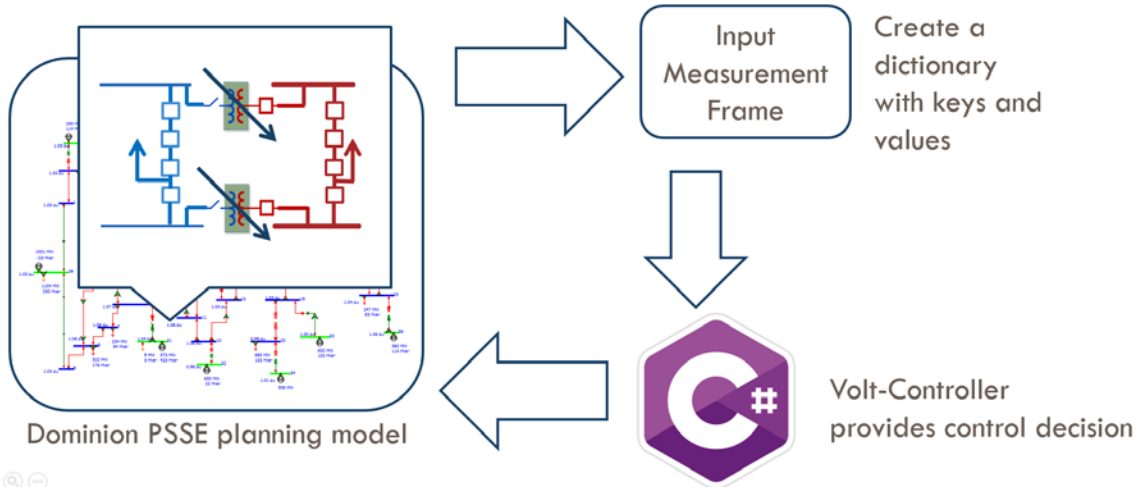


Figure 2 Cross-Validation

Note:

1. The python programs are written based on PSSE 34 API. If you are using PSSE 33, please change the PSSE settings: *import psse34* to *import psse*.
2. To test a specific case, please open the C# code and navigate to the main adapter: VoltVarControllerAdapter. Change the path name to a specific test folder. The figure below shows the test path for Test4.

```
// set the  directory
string testCase = (@"C:\Users\Duotong\Desktop\2019SUM\LocalVoltageControl20161109\Test4\\");
string inputdatafolder = (@"C:\Users\Duotong\Desktop\2019SUM\LocalVoltageControl20161109\Test4\Data\\");
string logsfolder = (@"C:\Users\Duotong\Desktop\2019SUM\LocalVoltageControl20161109\Test4\Logs\\");
string pythonCmdPath = (@"C:\Users\Duotong\Desktop\2019SUM\LocalVoltageControl20161109\Test4\pythonCode\\");
```

3. Click run, then you are supposed to see the program is continuously generating xml files in the data folder and logs folder. The program is designed to run 30 time instance only, so there will be 30 xml files generated in total.

4. To check the measurements, please open the csv files inside the data folder for a specific test. For example, the voltage measurement for 115 kV bus in Farm substation is stored in the 19th column of the transformer1.csv with a name called VoltsV.

Transformer Tap Changing

In this section, the simulation is conducted to demonstrate the control logic for the load tap changers. The tap position for both transformers are initialized as 0 while the tap limit position is ±16. The low and high voltage limits of both transformers is set to 114 kV and 116 kV respectively.

Test 1: Both Transformers' Voltages Reach Lower Limits

Step 1: Run DVPScaleLoad_CreateBenchMarkModel.py

The script will scale up the load at buses: 314691, 314692, 314693, 314694, 314695 for 350% and thus create a benchmark model with voltage at bus 314691 less than 115 kV.

Step 2: Set **"TapV**" for both transformers in configuration and csv files into 0

If the tap positions are initialized as 0, both transformers are unable to reach the highest tap position settings, which is 16. As load increasing continuously, the voltage controller is capable to regulate to a preferable voltage magnitude with tap changers' operations from both transformers.

Step 3: Navigate to the directory of C# scripts of Voltage Controller, run the solution file VoltController4.sln under the Microsoft Visual Studio environment.

It generates a series of operation condition frames according to different load settings during the voltage control stage.

Comment: From the results in transformer1.csv, plot the voltage magnitudes and the values for tap changer for this transformer. Figure 5 indicates the changes of voltage magnitude and tap position. As the load demand kept rising, the figure has shown two times of touches of the lower limit 114 kV at time instances 6 and 21, each of which has triggered tap changing in both transformers due to the sufficient spare amount to the highest tap position.

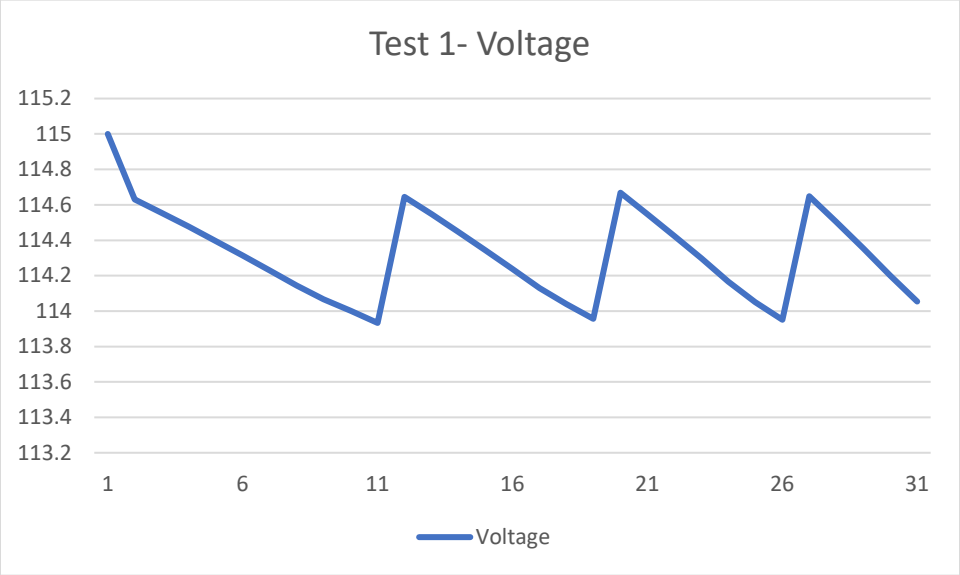


Figure 3 (a) Test 1: Both Transformers' Voltages Reach Lower Limits

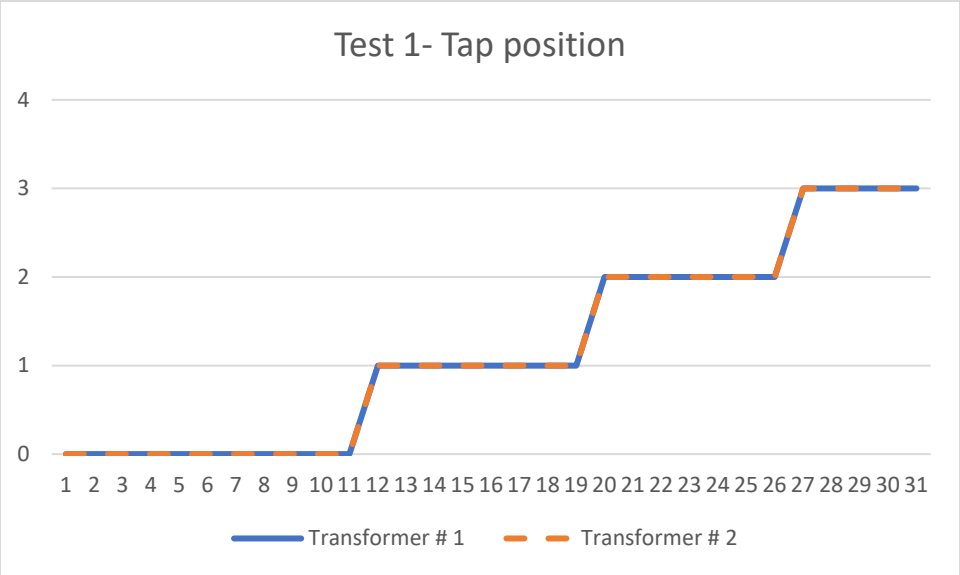


Figure 5 (b) Test 1: Both Transformers' Voltages Reach Lower Limits

Test 2: One Transformer's Tap Reaches the Limit

Step 1: Set "TapV" for transformers in configuration and csv files into 14 and 15

The second transformer is able to reach the highest tap position 16 first, then regulate to a preferable voltage magnitude coordinated by both transformers' tap changers.

Step 2: Navigate to the directory of C# scripts of Voltage Controller, run the solution file VoltController4.sln under the Microsoft Visual Studio environment. It generates a series of operation condition frames according to different load settings during the voltage control stage.

Comment: From the results in transformer2.csv, plot the voltage magnitudes and the values for tap changer for this transformer. Figure 6 indicates the changes of voltage magnitude and tap position. As the load demand kept rising, the figure has shown two times of touches of the lower limit 114 kV at time instances 6 and 21, and the second transformer changed its tap position at the time instance 6. However, at the time instance 25, even if the voltage has dropped below the lower limit, due to insufficient tap changing at this time, the voltage continued to drop, which reveals the unavailability of tap changings to maintain the voltage level at a preferable range.

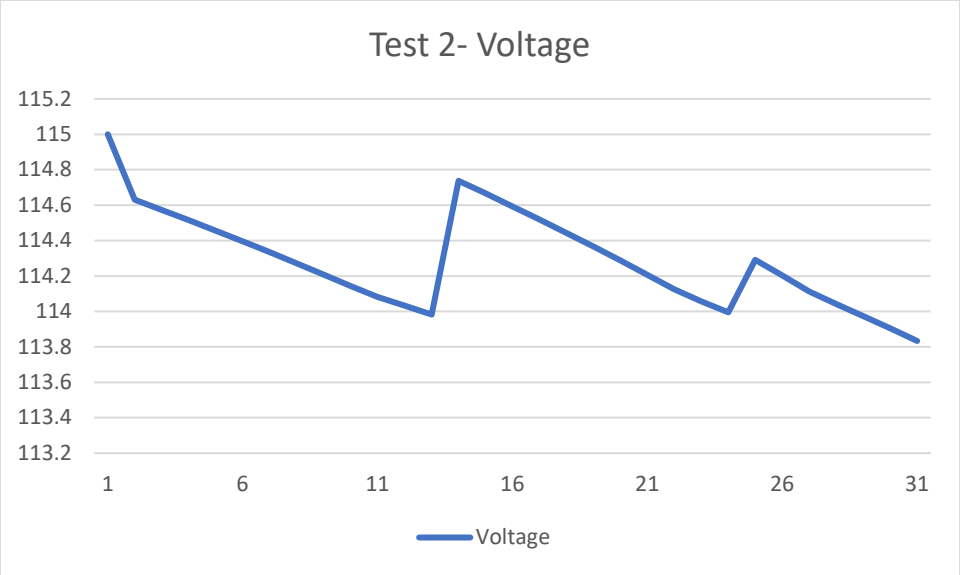


Figure 4 (a) Test 2: One Transformer's Tap Changer Reaches the Limit

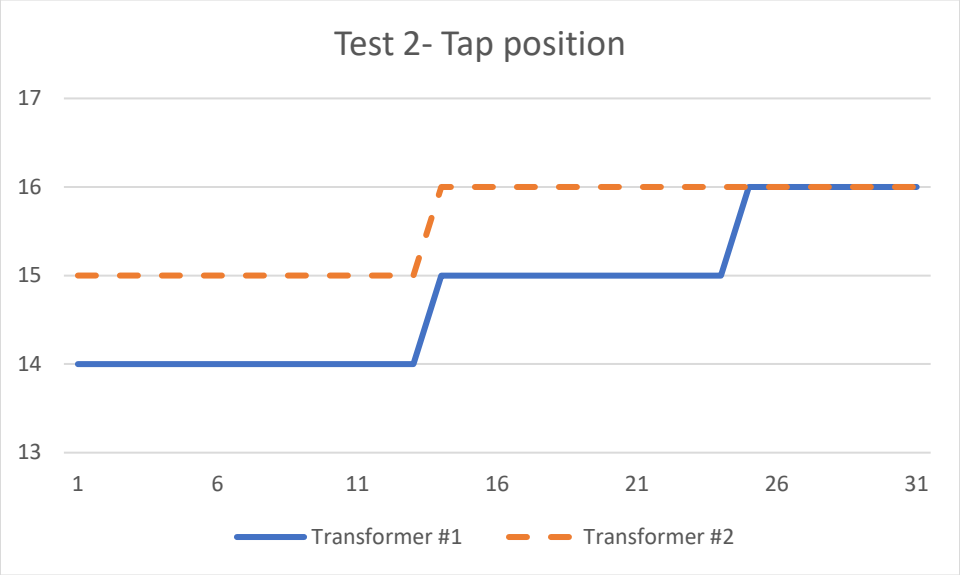


Figure 6 (b) Test 2: One Transformer's Tap Changer Reaches the Limit

Capacitor Bank Switching

In this section, the simulation is conducted to demonstrate the control mechanism for two capacitor banks. While the load is being increased by 3% in each step, the control decisions of these two capacitor banks are achieved when the voltages at their related buses reach the lower limit, which is 113.5kV.

Test 3: Capacitor Bank Switch On when Load Increase

Step 1: Run DVPScaleLoad_CreateBenchMarkModel.py

The script will switch off both capbanks at buses 314521 and 314519.

Step 2: Initialize the capacitor bank breaker configuration (CapBkrV) for both capacitor banks as “TRIP” in CapBank1.csv and CapBank2.csv files.

Both capacitor banks are currently on standby status. As load increasing continuously, the voltage controller is capable to regulate to a preferable voltage magnitude with operations of capacitor banks’ breakers to put capacitor banks online.

Step 3: Navigate to the directory of C# scripts of Voltage Controller, run the solution file VoltController4.sln under the Microsoft Visual Studio environment. It generates a series of operation condition frames according to different load settings during the voltage control stage.

Comment: From the results in CapBank1.csv and CapBank2.csv, plot the voltage magnitudes values for the capacitor banks, as shown in Figure 7 (a). In (b), “1” indicates the capacitor bank’s breaker is closed, and “0” indicates otherwise. As the load demand kept rising, the figure has shown that at the time instance 2, due to the high-load setting, the voltage at the controlled bus of the capacitor bank 1 has significantly dropped to 111.11kV, then the voltage controller decided to close one of the capacitor bank breaker and raised up the voltage at the time instance 3. Such process occurred again at the time instance 29, the voltage controller closed the capacitor bank 2’s breaker, after the voltage at the controlled bus of capacitor bank 2 dropped to 113.49kV (< 113.5kV). In addition, at the time instance 6, because the tap-changing operation occurred after a certain amount of time delay, the voltages are dropped subtly at both controlled buses.

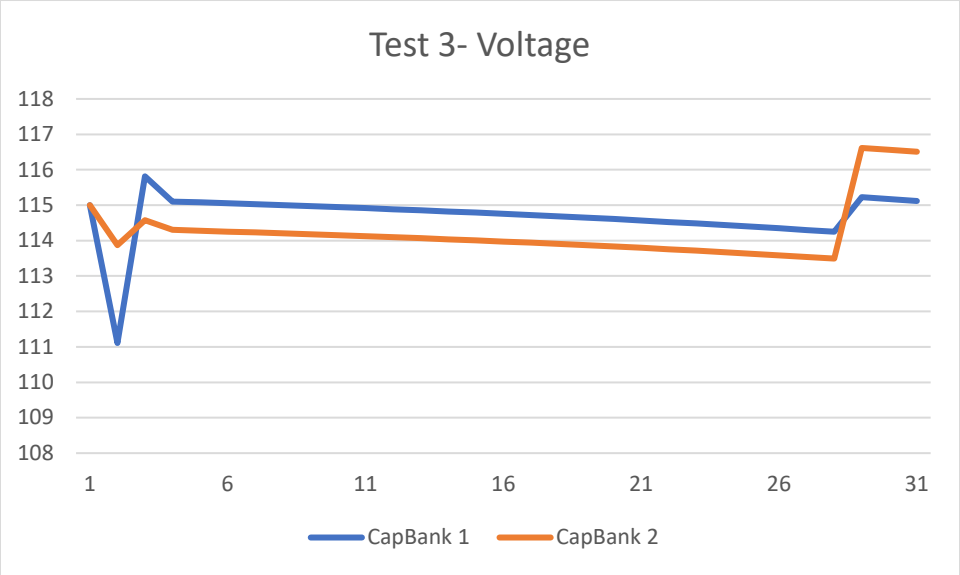


Figure 5 (a) Test 3-Voltage

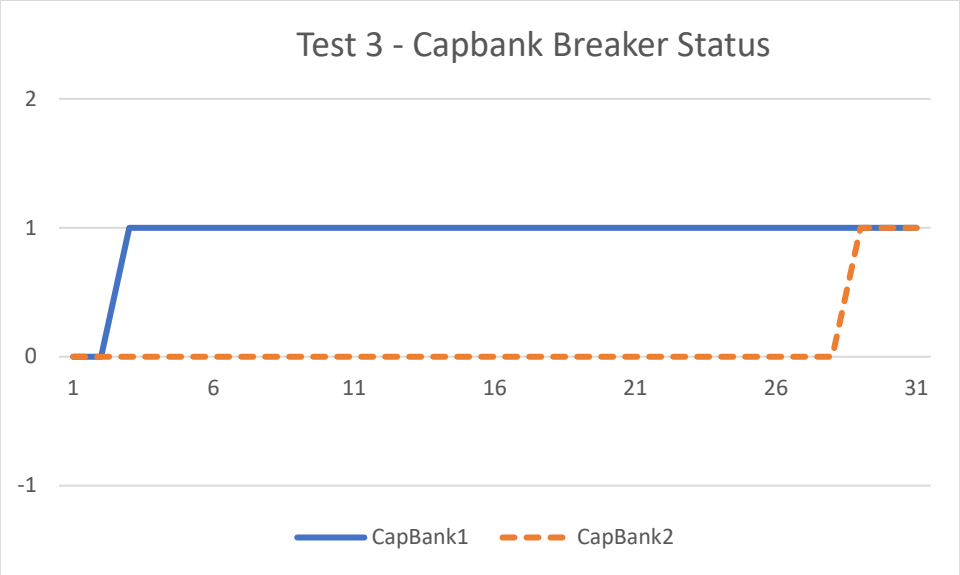


Figure 7 (b) Test 3- Capbank Breaker Status

Test 4: Capacitor Bank Switch Off when Load Decrease

Step 1: Run DVPScaleLoad_CreateBenchMarkModel.py

The script will scale down the load at buses: 314691, 314692, 314693, 314694, 314695 for 10%, such that create a benchmark model with voltage at bus 314519 higher than 117 kV. Both capbanks' breakers are set as closed.

Step 2: Initialize the tap changers configurations for both transformers as 0, the original tap position for LTC in transformer1.csv and transformer2.csv files,

Both transformers are unable to reach the highest tap position setting, which is 16. Besides, initialize the capacitor bank breaker configuration (CapBkrV) for both capacitor banks as "CLOSE" in CapBank1.csv and CapBank2.csv files. As load decreasing continuously, the Voltage Controller is capable to regulate to a preferable voltage magnitude with the comprehensive operations of transformers' tap changing and closing/tripping capacitor banks.

Step 3: Navigate to the directory of C# scripts of Voltage Controller, run the solution file VoltController4.sln under the Microsoft Visual Studio environment. It generates a series of operation condition frames according to different load settings during the voltage control stage.

Comment: From the results in transformer1.csv, plot the voltage magnitudes and the values for tap changer for this transformer. Figure 8 indicates the changes of voltage magnitude, the tap positions, and the status of capacitor banks' breakers. At the beginning, a significant load drop occurred at the time instance 2, which led to a considerable voltage increased to 117.4kV, then intermediately triggered the operation of tripping one capacitor bank according to the voltage controller mechanism. As the load demand kept dropping, the figure has shown a touch of the upper limit 116.1kV at the time instance 5, which has triggered tap changing to a lower position in both transformers.

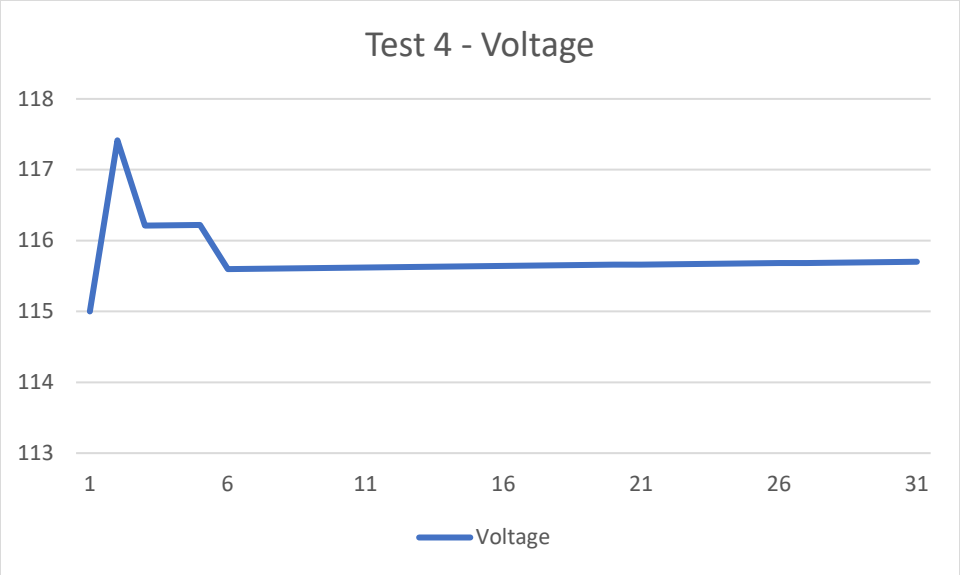


Figure 6 (a) Test 4 – Voltage

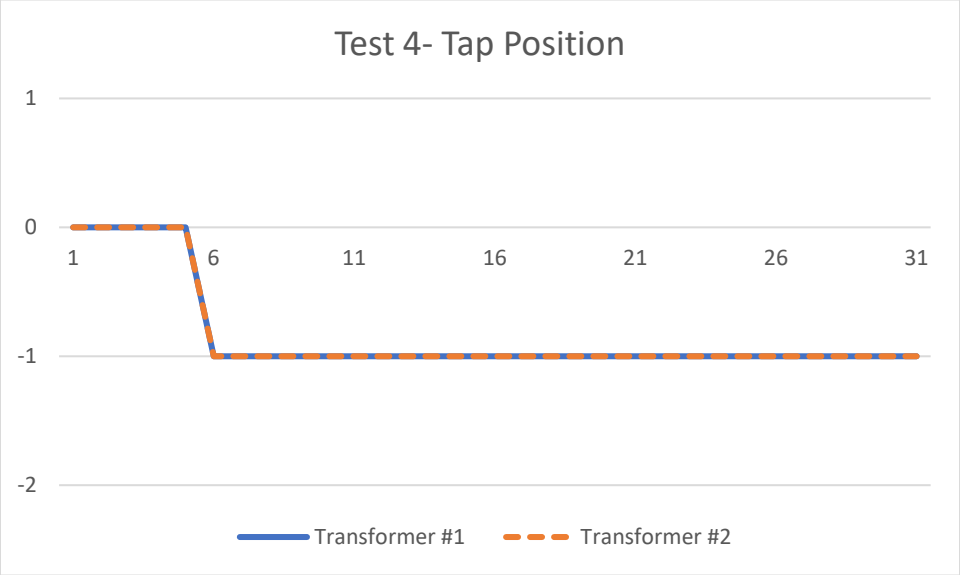


Figure 8 (b) Test 4 – Tap position

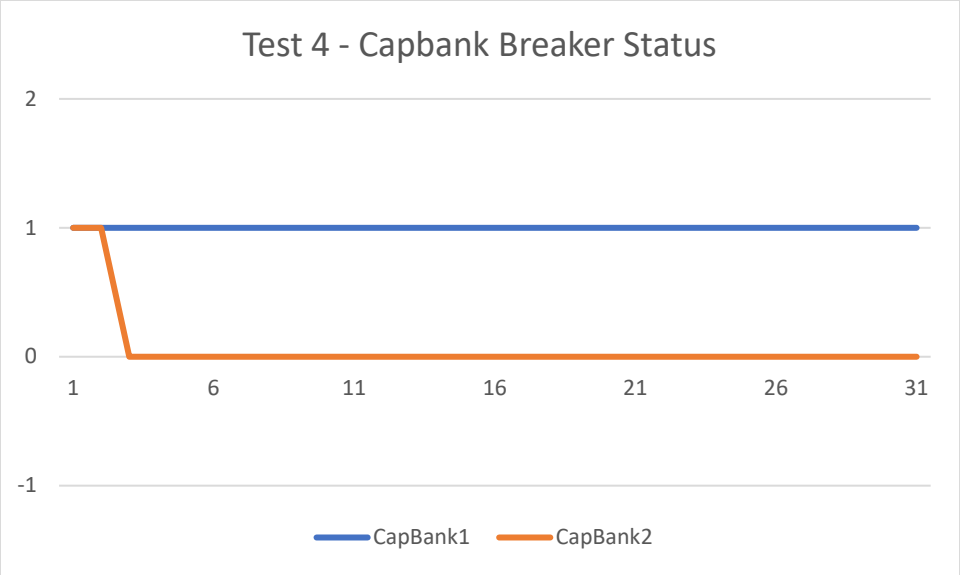


Figure 9 (c) Test 4 – Capbank Breaker Status

From Alpha to Beta

In the Beta version, the analytic will be running on openECA platform while the input measurements will be simulated using real time PMU simulator. This section presents the tests to be conducted in the Beta version and the preliminary test results without real time simulator (the real time simulator is arriving soon). The simulation result includes four tests for the Local Voltage Controller, driven by the synchrophasor streams from openECA platform. The objectives and the configuration files for these tests are listed in the following table.

Test	Objectives
1	Verify if the controller can RAISE both transformers' taps when voltages on both buses are lower than the limit (VLLIM = 114.5kV)
2	Verify if the controller is still able to operate (VLLIM = 114.5kV), when the other transformer's tap has reached the highest tap position (16)
3	Verify if the controller can switch ON the capacitor bank when the voltage in Pamplin substation reach the lower limit (Clov = 113.5kV)
4	Verify if the controller can switch OFF the capacitor bank when the voltage in Crewe substation reach the higher limit (Chiv = 119.7kV)

According to the control logic of the Local Voltage Controller, the analytics is going to decide based on the measurement streamed from the openECA platform. All the control decisions generated by the Local Voltage Controller are logged in the "CtrlDecisionLog_***_2017****_ *********.xml" files under the "Logs" folder. To fully reveal the process regarding to the control analytics, the program's main window monitors the tap positions of transformers, the breaker status of capacitor banks, and the voltage values on the buses of both transformers and capacitor banks. Besides, the Local Voltage Controller's decisions under the control logic are outputted to the main window of OpenECA Client Application.

Test 1: Operation of Load Tap Changers on Both Transformers

In this section, the simulation is conducted to demonstrate the control logic for the load tap changers (LTCs). The tap position for both transformers are initially configured as 0 in the Configurations_test1.xml file, while the tap limit position is ±16. The lower and higher voltage limits of both transformers are set as 114.5 kV (VLLIM) and 116 kV (VHLIM) respectively, under the tab of *VoltVarController>>SubstationAlarmDevice* in the *.xml file.

Scenario A: Both Transformers' Voltages Reach Lower Limits

1. There is a low-voltage incident is going to occur on the buses of the Farmville substation. After running the program, on the main window of OpenECA Client Application in Fig.1(a), we can see there is a key frame (highlighted) that the Local Voltage Controller decides to raise the load tap changers for both transformers due to the voltage for the next frame is below the configured lower limit, 114.5kV.
2. Subsequently, because of the setup delay for the load tap changers, the tap position values maintains at 1 for three frames. This situation is shown on the main window with the demonstration of "Not enough Counts yet = [1/3]", in which 1 indicates the current count of delay, and 3 indicates the maximum count needed to exit the delay.
3. Finally, after the Local Voltage Controller made a decision to raise the load tap changers' position on both transformers from 5 to 6, there is a mechanism (highlighted with red arrows) prevents the raise of tap position since the tap positions are too far apart from the initial position, see Fig.1(b).

```
============== MainWindowTempLog_2017-02-06_08:39:29.390 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 00       VoltsV: 115
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 00       VoltsV: 115
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 115  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 1     VoltsV: 119.199996948242  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Sub Taps No Conrtols Needed this Cycle
 = 0 PAMP SC242_CTL 1 114.190605163574 < 113.5 or gtvr = 9.55053615570068
 = 1 CREW SC242_CTL 1 119.802375793457 < 113.5 or gtvr = 9.55053615570068
 = Cap Bank Control: CAPBANK 1 is TRIP
 =  CapControl CREW SC242_CTL TRIP 119.802375793457
 =
============== MainWindowTempLog_2017-02-06_08:39:30.214 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 00       VoltsV: 114.630958557129
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 00       VoltsV: 114.630958557129
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.190605163574  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.802375793457  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Sub Taps No Conrtols Needed this Cycle
 = 0 PAMP SC242_CTL 1 114.108390808105 < 113.5 or gtvr = 10.2065572738647
 = 1 CREW SC242_CTL 0 119.722190856934 < 113.5 or gtvr = 10.2065572738647
 = Not enough CAPBANK Counts 1 < 5 1 < 5  3 < 3
 =
============== MainWindowTempLog_2017-02-06_08:39:31.021 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 00       VoltsV: 114.554695129395
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 00       VoltsV: 114.554695129395
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.108390808105  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.722190856934  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Control Decision FARMLTC TX4LTC_CTL RAISE
 = Control Decision FARMLTC TX5LTC_CTL RAISE
 = 0 PAMP SC242_CTL 1 114.024047851563 < 113.5 or gtvr = 10.8812694549561
 = 1 CREW SC242_CTL 0 119.639976501465 < 113.5 or gtvr = 10.8812694549561
 = Not enough CAPBANK Counts 2 < 5 2 < 5  0 < 3
============== MainWindowTempLog_2017-02-06_08:39:31.838 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 01       VoltsV: 114.476501464844
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 01       VoltsV: 114.476501464844
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.024047851563  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.639976501465  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [1<3]
 = 0 PAMP SC242_CTL 1 113.937591552734 < 113.5 or gtvr = 11.5752239227295
 = 1 CREW SC242_CTL 0 119.555725097656 < 113.5 or gtvr = 11.5752239227295
 = Not enough CAPBANK Counts 3 < 5 3 < 5  1 < 3
 =
```

Fig. 1(a) Main Window Snapshot in Scenario A

```
============== MainWindowTempLog_2017-02-06_08:39:48.937 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 05       VoltsV: 114.050849914551
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 05       VoltsV: 114.050849914551
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.564727783203  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.226654052734  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Control Decision FARMLTC TX4LTC_CTL RAISE
 = Control Decision FARMLTC TX5LTC_CTL RAISE
 = 0 PAMP SC242_CTL 1 113.457328796387 < 113.5 or gtvr = 30.6977653503418
 = Determine Cap Bank Control...
 = 1 CREW SC242_CTL 0 119.192741394043 < 113.5 or gtvr = 30.6977653503418
 = Not enough CAPBANK Counts 18 < 5 5 < 5  0 < 3
 =
============== MainWindowTempLog_2017-02-06_08:39:49.777 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 06       VoltsV: 113.951263427734
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 06       VoltsV: 113.951263427734
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.457328796387  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.192741394043  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = LTCs are too far apart must EXIT [6/5.2]
 = 0 PAMP SC242_CTL 1 114.209167480469 < 113.5 or gtvr = 31.911075592041
 = 1 CREW SC242_CTL 0 119.820571899414 < 113.5 or gtvr = 31.911075592041
 = Not enough CAPBANK Counts 19 < 5 5 < 5  1 < 3
 =
```

Fig. 1(b) Main Window Snapshot in Scenario A

Scenario B: Operation of Load Tap Changers on Only One Transformer

1. Similarly to Scenario A, a low-voltage incident is going to occur on the buses of the Farmville substation in this situation. However, in this case, only one transformer is able to raise the load tap changer since the other one has already been raised to the highest tap position. On the main window of OpenECA Client Application in Fig.2(a), we can see there is a key frame (highlighted) that the Local Voltage Controller decides to raise the load tap changers from 12 to 13 due to the voltage for the next frame is below the configured lower limit, 114.5kV.
2. Subsequently, because of the setup delay for the load tap changers, the tap position values maintains at 1 for three frames. This situation is shown on the main window with the demonstration of “Not enough Counts yet = [1/3]”, in which 1 indicates the current count of delay, and 3 indicates the maximum count needed to exit the delay.
3. After 3 frames of delays (highlighted with blue arrows), both load tap changers have reached the highest tap position, 16, in this case, none of further controls could be made to ameliorate such a low-voltage situation, see Fig.2(b).

```
============== MainWindowTempLog_2017-02-06_08:40:15.318 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.554695129395
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 12       VoltsV: 114.554695129395
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.108390808105  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.722190856934  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = CON TAP Control Deselected No control
 = Control Decision FARMLTC TX5LTC_CTL RAISE
 = 0 PAMP SC242_CTL 1 114.024047851563 < 113.5 or gtvr = 10.8812694549561
 = 1 CREW SC242_CTL 0 119.639976501465 < 113.5 or gtvr = 10.8812694549561
 = Not enough CAPBANK Counts 2 < 5 2 < 5  0 < 3
 =
============== MainWindowTempLog_2017-02-06_08:40:16.123 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.476501464844
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 13       VoltsV: 114.476501464844
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.024047851563  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.639976501465  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [1<3]
 = 0 PAMP SC242_CTL 1 113.937591552734 < 113.5 or gtvr = 11.5752239227295
 = 1 CREW SC242_CTL 0 119.555725097656 < 113.5 or gtvr = 11.5752239227295
 = Not enough CAPBANK Counts 3 < 5 3 < 5  1 < 3
 =
============== MainWindowTempLog_2017-02-06_08:40:16.930 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.396362304688
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 13       VoltsV: 114.396362304688
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.937591552734  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.555725097656  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [2<3]
 = 0 PAMP SC242_CTL 1 113.848976135254 < 113.5 or gtvr = 12.2891302108765
 = 1 CREW SC242_CTL 0 119.469345092773 < 113.5 or gtvr = 12.2891302108765
 = Not enough CAPBANK Counts 4 < 5 4 < 5  2 < 3
 =
============== MainWindowTempLog_2017-02-06_08:40:17.757 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.31421661377
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 13       VoltsV: 114.31421661377
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.848976135254  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.469345092773  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [3<3]
 = 0 PAMP SC242_CTL 1 113.75813293457 < 113.5 or gtvr = 13.0235967636108
 = 1 CREW SC242_CTL 0 119.380798339844 < 113.5 or gtvr = 13.0235967636108
 = Not enough CAPBANK Counts 5 < 5 5 < 5  3 < 3
 =
============== MainWindowTempLog_2017-02-06_08:40:18.567 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.230026245117
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 13       VoltsV: 114.230026245117
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.75813293457  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.380798339844  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = CON TAP Control Deselected No control
 = Control Decision FARMLTC TX5LTC_CTL RAISE
 = 0 PAMP SC242_CTL 1 113.664901733398 < 113.5 or gtvr = 13.7878675460815
 = 1 CREW SC242_CTL 0 119.289970397949 < 113.5 or gtvr = 13.7878675460815
 = Not enough CAPBANK Counts 6 < 5 5 < 5  0 < 3
 =
```

Fig. 2(a) Main Window Snapshot in Scenario B

```
=============== MainWindowTempLog_2017-02-06_08:40:26.673 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.129409790039
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 15       VoltsV: 114.129409790039
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.649612426758  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.275001525879  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = CON TAP Control Deselected No control
 = Control Decision FARMLTC TX5LTC_CTL RAISE
 = 0 PAMP SC242_CTL 1 113.553031921387 < 113.5 or gtvr = 22.1003341674805
 = 1 CREW SC242_CTL 0 119.222946166992 < 113.5 or gtvr = 22.1003341674805
 = Not enough CAPBANK Counts 12 < 5 5 < 5  0 < 3
 =
=============== MainWindowTempLog_2017-02-06_08:40:27.481 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.040115356445
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 16       VoltsV: 114.040115356445
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.553031921387  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.222946166992  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [1<3]
 = 0 PAMP SC242_CTL 1 113.463714599609 < 113.5 or gtvr = 22.9370155334473
 = Determine Cap Bank Control...
 = 1 CREW SC242_CTL 0 119.194770812988 < 113.5 or gtvr = 22.9370155334473
 = Not enough CAPBANK Counts 13 < 5 5 < 5  1 < 3
 =
=============== MainWindowTempLog_2017-02-06_08:40:28.289 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 113.957191467285
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 16       VoltsV: 113.957191467285
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 113.463714599609  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.194770812988  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [2<3]
 = 0 PAMP SC242_CTL 1 114.230560302734 < 113.5 or gtvr = 23.9105644226074
 = 1 CREW SC242_CTL 0 119.841407775879 < 113.5 or gtvr = 23.9105644226074
 = Not enough CAPBANK Counts 14 < 5 5 < 5  2 < 3
 =
=============== MainWindowTempLog_2017-02-06_08:40:29.097 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.668121337891
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 16       VoltsV: 114.668121337891
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.230560302734  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.841407775879  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [3<3]
 = 0 PAMP SC242_CTL 1 114.101165771484 < 113.5 or gtvr = 25.014009475708
 = 1 CREW SC242_CTL 0 119.715103149414 < 113.5 or gtvr = 25.014009475708
 =
=============== MainWindowTempLog_2017-02-06_08:40:29.942 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 16       VoltsV: 114.54793548584
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 16       VoltsV: 114.54793548584
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.101165771484  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0     VoltsV: 119.715103149414  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = CON TAP Control Deselected No control
 = CON TAP Control Deselected No control
 = 0 PAMP SC242_CTL 1 113.967750549316 < 113.5 or gtvr = 26.1511497497559
 = 1 CREW SC242_CTL 0 119.585075378418 < 113.5 or gtvr = 26.1511497497559
```

Fig. 2(b) Main Window Snapshot in Scenario B

Test 2: Operation of Capacitor Banks' Breakers

Scenario A: Switch On the Capacitor Bank's Breaker due to Low-Voltage

1. In this situation, there is a low-voltage incident is going to occur on the buses of Farmville substation which is near the Pamplin substation. Both the capacitor banks breakers' values of Pamplin substation and Crewe substation are initially set as "0", indicating the capacitor breakers are open, and the capacitor banks are operating offline.

2. After running the program, on the main window of OpenECA Client Application in Fig.3, we can see there is a key frame (highlighted) that the Local Voltage Controller decides to close the breaker of the capacitor banks of Pamplin substation, because the voltage value has decreased to 113.438kV, which is lower to the lower limit of 113.5kV for the capacitor bus.

```
============== MainWindowTempLog_2017-02-06_08:40:57.251 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 02       VoltsV: 114.000900268555
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 02       VoltsV: 114.000900268555
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 0    VoltsV: 113.510856628418  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0    VoltsV: 119.20964050293  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Not enough Counts yet = [3<3]
 = 0 PAMP SC242_CTL 0 113.43766784668 < 113.5 or gtvr = 15.7822723388672
 = Determine Cap Bank Control...
 = Cap Bank Control: CAPBANK 0 is CLOSE
 =  1 CREW SC242_CTL 0 119.186561584473 < 113.5 or gtvr = 15.7822723388672
 = CapControl PAMP SC242_CTL CLOSE 113.43766784668
 =
============== MainWindowTempLog_2017-02-06_08:40:58.065 ==============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 02       VoltsV: 113.933082580566
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 02       VoltsV: 113.933082580566
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1    VoltsV: 113.43766784668  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0    VoltsV: 119.186561584473  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Sub Taps No Conrtols Needed this Cycle
 = 0 PAMP SC242_CTL 1 114.205383300781 < 113.5 or gtvr = 16.6172981262207
 = 1 CREW SC242_CTL 0 119.816856384277 < 113.5 or gtvr = 16.6172981262207
 = Not enough CAPBANK Counts 1 < 5 1 < 5  3 < 3
 =
```

Fig. 3 Main Window Snapshot in Scenario C

Scenario B: Switch Off the Capacitor Bank's Breaker due to High-Voltage

1. In this situation, there is a high-voltage incident is going to occur on the buses of Farmville substation, which is near the Crewe substation. Both the capacitor banks breakers' values of Pamplin substation and Crewe substation are initially set as "1", indicating the capacitor breakers are closed, and the capacitor banks are operating online.
2. After running the program, on the main window of OpenECA Client Application in Fig.4, we can see there is a key frame (highlighted) that the Local Voltage Controller decides to trip the capacitor banks of Crewe substation, because the voltage value has increased to 119.802kV, which is higher to the higher limit of 119.7kV for the capacitor bus.

```
=============== MainWindowTempLog_2017-02-06_08:41:30.188 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 00       VoltsV: 115
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 00       VoltsV: 115
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 115  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 1 ⬅   VoltsV: 119.199996948242  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Sub Taps No Conrtols Needed this Cycle
 = 0 PAMP SC242_CTL 1 114.190605163574 < 113.5 or gtvr = 9.55053615570068
 = 1 CREW SC242_CTL 1 119.802375793457 < 113.5 or gtvr = 9.55053615570068
 = Cap Bank Control: CAPBANK 1 is TRIP
 =  CapControl CREW SC242_CTL TRIP 119.802375793457
 =
=============== MainWindowTempLog_2017-02-06_08:41:31.005 ===============

        - ControlTransformers_FARMLTC_TX4LTC_CTL        | TapV: 00       VoltsV: 114.630958557129
        - ControlTransformers_FARMLTC_TX5LTC_CTL        | TapV: 00       VoltsV: 114.630958557129
        - ControlCapBanks_SC242PAMPLIN  | CapBkrV: 1     VoltsV: 114.190605163574  Clov:113.5 Chiv:119.7
        - ControlCapBanks_SC242CREWE    | CapBkrV: 0 ⬅   VoltsV: 119.802375793457  Clov:113.5 Chiv:119.7
%%        Program control is ON
 = Sub Taps No Conrtols Needed this Cycle
 = 0 PAMP SC242_CTL 1 114.108390808105 < 113.5 or gtvr = 10.2065572738647
 = 1 CREW SC242_CTL 0 119.722190856934 < 113.5 or gtvr = 10.2065572738647
 = Not enough CAPBANK Counts 1 < 5 1 < 5  3 < 3
 =
```

Fig. 4 Main Window Snapshot in Scenario D

9 PMU SYNCHROSCOPE

The Goal is to develop a generalized tool to provide synchroscope functionality to a remote location overcoming communication (and other) delays by estimating end-to-end delays and predicting closing time through:

- Manual (computer supervised) controlled close
- Automatic (computer actuated) controlled close
- Block control when parameters are out of bounds

In other words, the analytic should be able to connect to a stream of synchrophasor data from the openECA platform and send control signals back to the openECA from a remote location. For the Alpha version of the Analytic, a mockup application showing measurements and controls had been designed. A constant delay (in milliseconds) is introduced. In the future, estimation of this delay along with its distribution will be included along with actual data measurements from a simple predefined model.

Synchronizing Method:

To synchronize two separate islands, we need to retrieve voltage phasor measurements and frequency at Bus A and B. Cumulative Delays are calculated depending upon network configuration and traffic of the path adopted. (To be tested first with constant delays). Following the procedure as stated before, we calculate Advanced Angle of operations considering delays and difference in Voltage angles. Depending upon power flow requirements extra constraint can be added to the algorithm for **f**incoming>**f**Reference. Also δlim window for angle would also be modified based upon the Adv. Angle.

$$Adv.Ang = \left\{\frac{(Slip)cyc}{sec}\right\}\left\{\frac{sec}{60\ cyc}\right\}\left\{\frac{360}{cyc}\right\}\{(Cumulative\ Delay)cyc\}$$

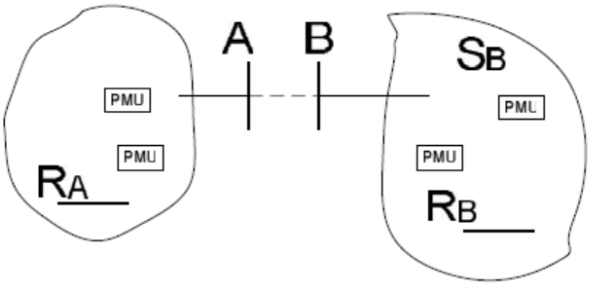
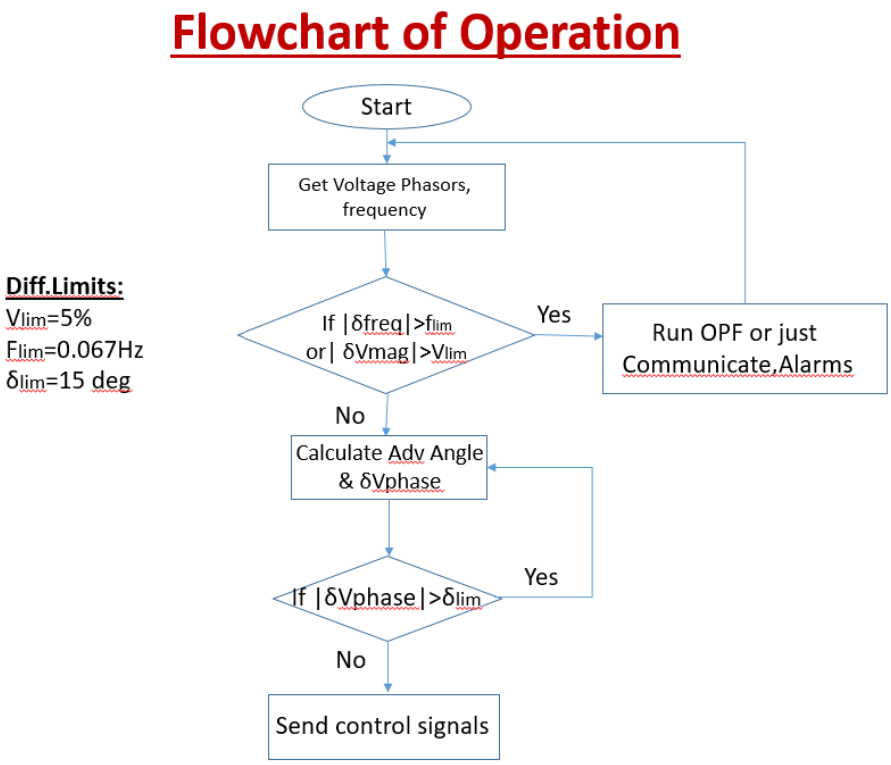


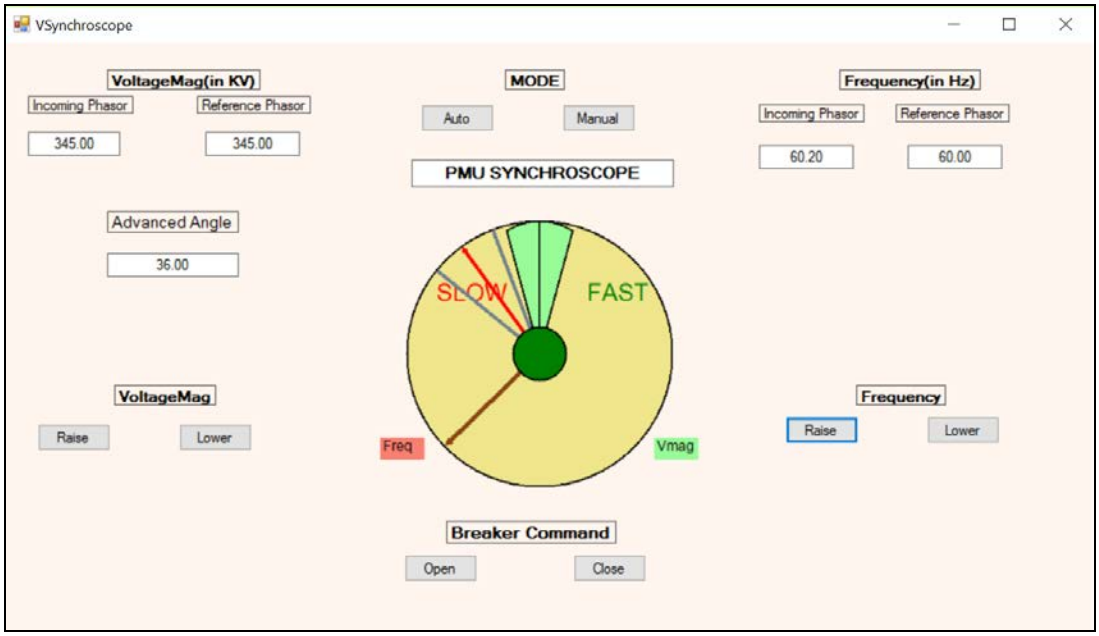
Figure 1: Two islands to be synchronized by closing breaker between Bus A and Bus B



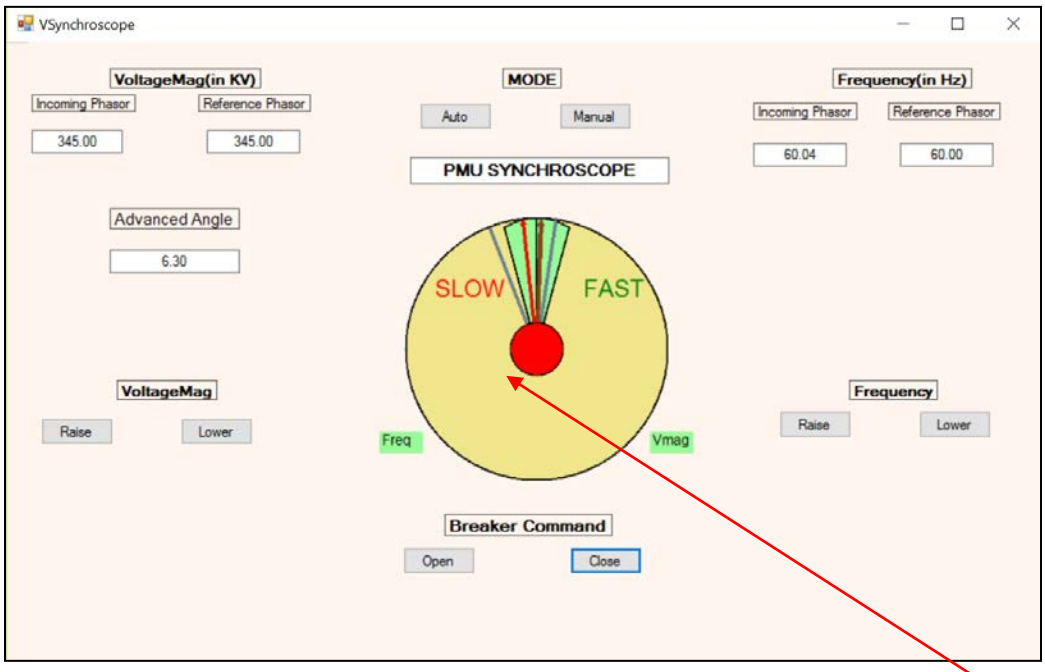
Initial Mockup:

The initial Mockup for the Alpha Test was uploaded on Github. A simple Rotating Synchroscope Mockup Application (without any stream of Data) is built. Run the VSynchroscope.cs file to open the application and its associated code. Once the application is run the following window pops up as shown below. Selection between Auto and Manual Mode is provided left to the discretion of the user. Manual Mode of Operation is only shown as of now. Voltage and Frequency measurements are displayed at the top for both the Reference phasor and the incoming phasor. The breaker command controls are at the bottom of the mockup.

Voltage magnitude and Frequency can be regulated as seen from the output of the program. The size of the Voltage phasor would change depending upon the predetermined increment. In this version, one click is set to change 0.05/3 pu (i.e. 3 clicks changes 0.05 pu of V_mag). Also frequency can be regulated depending upon which the voltage phasor may either rotate slowly or even in the opposite direction depending upon whether it is greater than or lesser than the Reference frequency (One click changes 0.033 Hz of slip). Depending upon whether Voltage magnitude and frequency difference tolerance limits are satisfied, indications are displayed. (Green-Satisfied, Red-Not satisfied).



A constant delay (of 500 milliseconds) has been incorporated in our mockup. Depending upon the slip frequency and this delay and using the equation as shown earlier, the Advanced Angle is calculated and displayed and thus the modified tolerance window is formed. Any breaker close command within this window will result in the closing of the synchronizing breaker within the actual tolerance window subjected to the fact that all other requirements were met during breaker close command initiation i.e. breaker close Command can only be initiated when all the criteria are met.



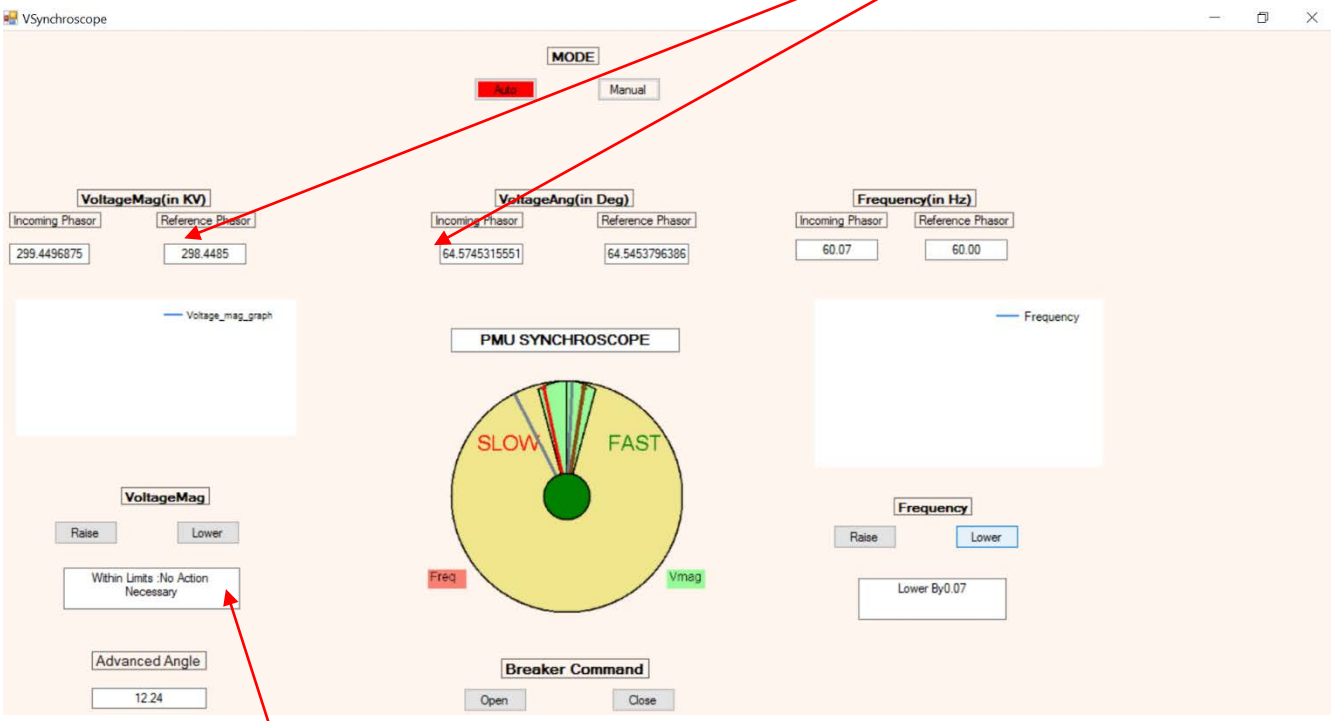
Indication for Successful Synchronization would be reflected in the center (Color Change from Green to Red)

In the Alpha Version of the analytic, the functionality of such a remote synchroscope is depicted without any stream of data. Ideally the incoming phasor is supposed to run at a greater frequency than the Reference phasor so that power would flow from island of incoming phasor to island of Reference phasor. But in our analytic it can be done either ways (forward/reverse synchronization). Depending upon requirements, reverse synchronization (counter-clockwise rotation mode) may be switched off.

From Alpha to Beta

In the Beta version, the analytic will be running on openECA platform while the input measurements will be simulated using real time PMU simulator. This section presents the preliminary results without real time simulator and using sample data from the openECA platform.

Run the solution file of the C# project VoltageInput_Synch as uploaded. The code for running the synchroscope analytic Windows form application with the help of openECA is developed. Sample data representing two buses in an electric system is derived from the openECA platform and integrated to our Windows form Application (Files uploaded in GitHub Folder).

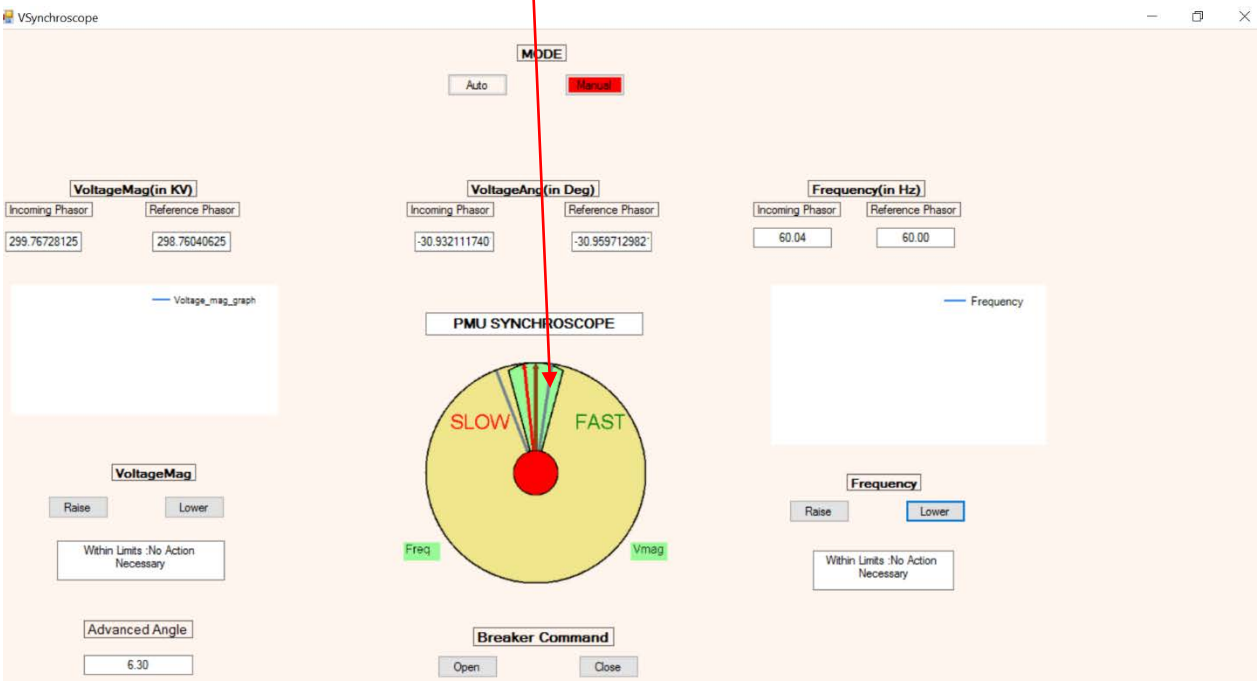


As seen from the figure above, sample data for Voltage Magnitude and Angle values from two buses are streaming into the application and as per the angular difference and frequency slip between two buses, the incoming phasor rotates.

As we are streaming the data as of now, the controls on the voltage magnitude could not be made possible and thus additional alarm spaces have been included which mentions by how much the voltage and frequency must be changed to bring it back within limits or not.

Also for the sake of proper depiction of the functionality of the analytic, frequency input is a user provided quantity which can be changed at user's will (later it would be the real time frequency from the system) according to which the incoming phasor rotates as well as the angular difference between the two buses increases and decreases periodically.

Auto and manual mode are being implemented. Auto mode will constantly check for the incoming phasor to be within the tolerance windows and would initiate the breaker close command once the criteria are fulfilled such that synchronization occurs at the 12 o'clock position automatically and switch to manual mode for further actions if needed by the user.



Thus tests that were proposed to be conducted have been complied with and verified.

Test 1: Proper Depiction: The analytic would be tested whether it successfully represents the real time measurements in the form of phasors along with accurate rotation of incoming phasor proportional to slip frequency with regards to reference phasor.

Test 2: Annunciation Display: It would be verified if Proper alarms are raised for meeting the criteria for successful synchronization along with checklist for the same.

Test 3: Compatibility with openECA.

Work relating to further modifying the analytic representation along with plotting real time values of Voltages and frequencies along with error models is in progress and would be completed in the near future. The analytic would next be tested with the Opal RT phasor simulator to check its accuracy in the synchronizing process.

10 CT/PT CALIBRATION

In order to realize the analytic of instrument transformers(CT/PT) calibration, we firstly use PSS\E to conduct the power flow of the IEEE standard 118-bus system to gather the simulated voltage and current data; then, realize the functionality including system topology analysis, single transmission line CT/PT calibration, and whole system calibration on Matlab to validate the methodology. The alpha version program is designed to be operating without openECA platform.

Program Details (Alpha Version)

Program Process

1. PSS\E power system operation simulation
   Use PSS\E to conduct power flow based on the IEEE 118-bus power system and the morning load pick-up curve to generate the voltages of the 345KV buses and currents flowing though corresponding transmission lines.

2. Raw data processing
   Read in CSV file generated by Python and PSS\E.

3. Building error model (For test plan)
   Add CT/PT measurement errors and PMU errors to the raw data of voltages and currents based on the derived error model; record the positive sequence errors and the true line impedance and susceptance.

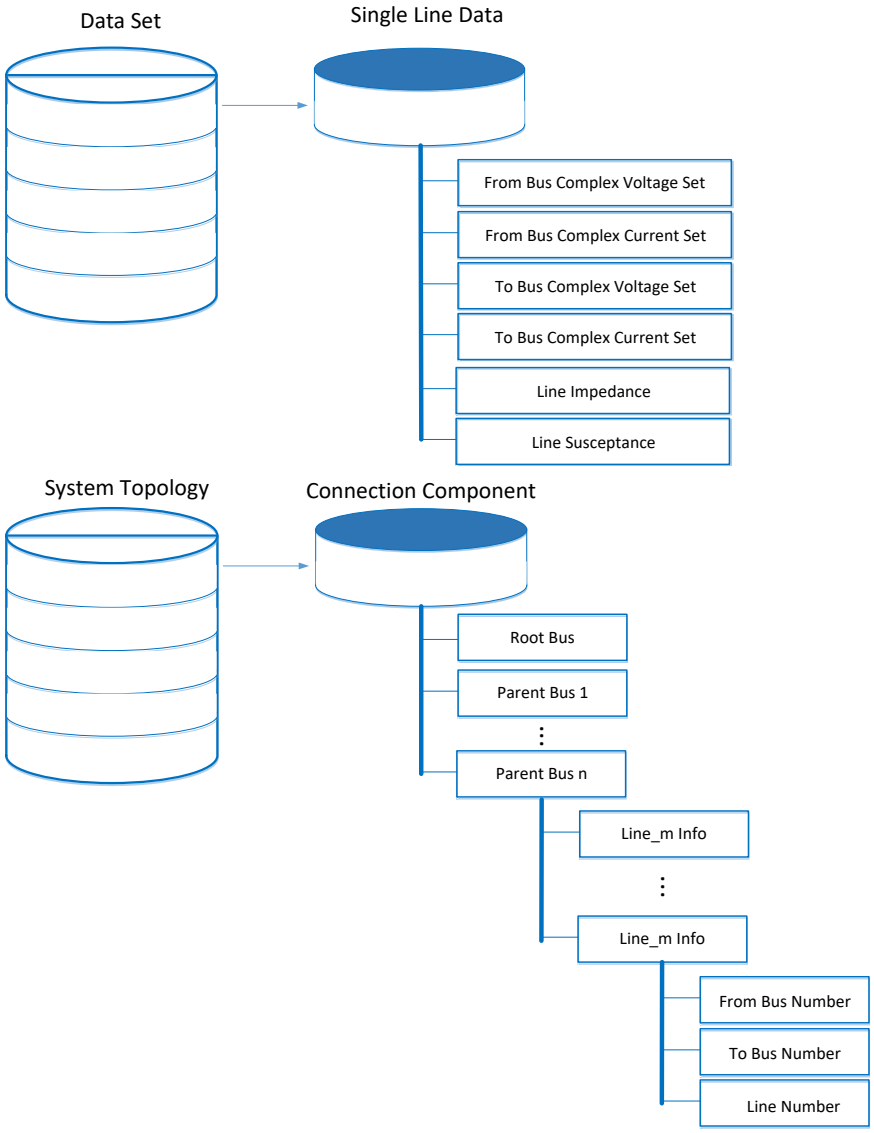
4. System topology analysis
   Analyze the system topology based on the from-bus and to-bus information of the concerned lines; find the order of calibration propagation.

5. CT/PT calibration
   Conduct the CT/PT calibration starting from the 345KV bus and corresponding line that equipped with revenue transducers; use the injection propagation method aforementioned to calibrate the whole 345KV system.

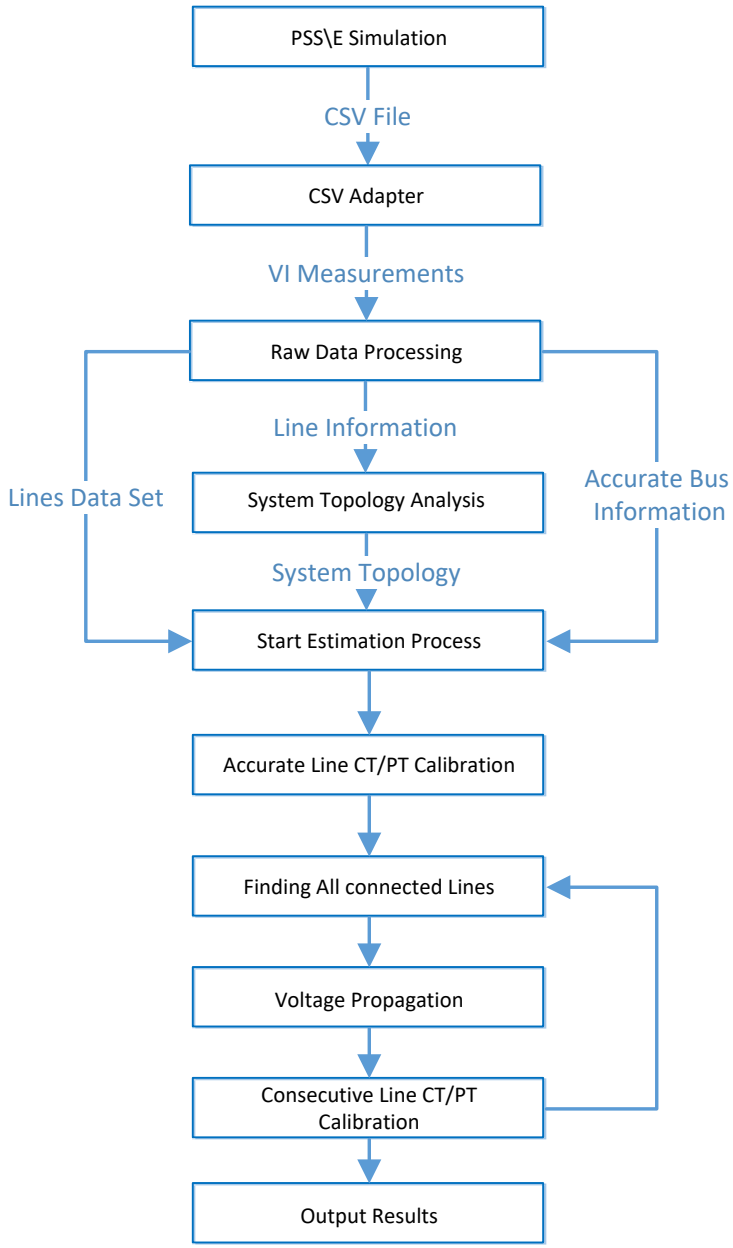
Test Results(Alpha Version)

Data Structure

The data structure of the alpha version controller is shown as follows:



Data Flow



Calibration Test Results

Raw Data Generation

In this section, the PSS\E simulation is conducted to generate voltage and current data of the concerned power system. The PSS\E is accessed through Python.

Step 1: Locate in to the folder maned as *Step_1_VI_Acquisition*; run the file *IEEE_118_data_generation_main.py* to start generating voltage and current measurements data.

Step 2: The generated voltage and current data can be found in the file named as *VI_Measurement_All_345KV_Buses_Peak.csv*; copy this file and paste it into the *Step_2_Error Model* folder.

Error Model Construction

In this section, the CT/PT and PMU errors are added into the simulated data to construct the error model.

Step 1: Run *Matlab_CSV_adapter_IEEE_118.m* through Matlab to acquire the bus information, voltage and current simulated data from the CSV file, *VI_Measurement_All_345KV_Buses_Peak.csv*; the results include
1) the 345KV bus number set, saved in *Bus_number_set_345KV.mat*,
2) the true values of the positive sequence voltages on each 345 KV buses, saved in *V_true_value_positive_sequence.mat*,
3) the true positive sequence currents flowing through all the lines, two-winding transformers, and three-winding transformers connected to the 345KV buses, saved in *I_true_value_positive_sequence.mat*, *I_true_value_positive_sequence_trn.mat*, and *I_true_value_positive_sequence_gen.mat* respectively,
4) the from-bus numbers and to-bus numbers of each transmission line, two-winding transformers, and three-winding transformers connected to the 345KV buses, saved in *line_bus_info_all_lines.mat*, *line_bus_info_trn.mat*, *line_bus_info_gen.mat*.

Step 2: Run *Line_data_generation_IEEE_118.m* through Matlab to acquire the power system network information, save the true value of the voltages and currents of each line or transformer equivalent line, and construct the error model introduced previously; the network information is saved in *AC_line_info.mat* which is formed as 11 column vectors, i.e. [line number, line ID, line type, from bus number, KV1, KI1, to bus number, KV2, KI2, Z, y], as well as the bus number information of all the 345KV transmission lines, saved in *line_bus_info_345KV.mat*; each transmission line or transformer equivalent line is assigned a line number, and the three-phase true value of the voltages and currents of each line is saved in the files named as *line_(line number)_true_3_phase.mat*; the true positive sequence values are saved in the files named as *line_(line number)_true_positive_sequence.mat* in the format of [from-bus voltages, from-bus currents, to-bus voltages, to-bus currents]; the positive sequence values added errors are referred to as measured value and are saved in the files named as *line_(line number)_measured_positive_sequence.mat* with the same format as true value files; the total line number is 24 in the test case.

Step 3: Run *True_impedance_calculation_IEEE_118.m* through Matlab to acquire 345KV transmission lines' impedances and susceptances and assign such data to the 10^{th} and 11^{th} column of *AC_line_info.mat* respectively and save the AC_line_info matrix in the file *AC_line_info_true_value_Zy.mat*.

Step 4: Copy the following files and paste it into the *Step_3_CTPT Calibration* folder:
AC_line_info_true_value_Zy.mat,
Bus_number_set_345KV.mat,
line_(every linenumber)_measured_positive_sequence.mat,
line_(every line number)_true_positive_sequence.mat,
line_(every line number)_true_3_phase.mat (optional),
line_bus_info_345KV.mat.

CTPT Calibration

In this section, the CT/PT calibration is conducted based on the simulated data throughout the 345KV subsystem within the IEEE 118 system.

Step 1: Run *CT_PT_calibration_IEEE_118.m* through Matlab to start the impedance calibration process; notice that only run the following part of the code at the first time of the tests based on the same accurate bus to save the original voltage and current data of that bus and corresponding line.

```
%-------------------------------------------------------------------------------------------------------------------------%
    line_name=['line_',num2str(original_accurate_line_number), '_measured_positive_sequence.mat'];
    VI_origin_struct=load(line_name);
    VI_measurement_set = VI_origin_struct.VI_measurement_set;
    line_name=['line_',num2str(original_accurate_line_number),'_measured_positive_sequence_origin.mat'];
```

```
        save(line_name,'VI_measurement_set');
%-------------------------------------------------------------------------------------------------------------------------%
```

Step 2: The Results are saved in the file named as *line_estimation_results.mat* in the form of *[line number, from bus number, KV1_hat, KI1_hat, to bus number, KV2_hat, KI2_hat, Z, y],* and the errors of the calibration are shown in the command window of Matlab as attached table.

CTPT Calibration Results Analysis

For some of the lines in IEEE 118 bus system, there are generators that connect to the relevant buses directly, which is not feasible in real-world power system. There is a good reason to believe that such scenario that generator -> line -> bus -> line -> bus connection should has negative influence to the estimation accuracy. That is why the estimation results of the Line 7 current correction factors are larger than other lines. Besides, given the propagation process, the estimation error of the previous pi-section will surely affect the consecutive ones, i.e. accumulation effect of the errors. Therefore, the later the lines are visited, the larger the estimation errors will be. For the further versions of the application, the algorithms will be improved to provide more accurate results.

line_number	KV1_error	KI1_error	KV2_error	KI2_error
10	0+0i	0.00040057+0.00013761i	1.6929e-06+8.4113e-07i	-0.00021291+0.0011885i
9	8.584e-07+6.792e-07i	-2.4843e-05+4.6566e-05i	1.3021e-06+5.2792e-07i	-5.503e-05-2.2845e-05i
6	1.0863e-07+5.7767e-07i	-0.00033709+0.0012504i	-0.00013771+0.00012165i	-0.00086444+0.00077404i
8	5.3731e-07+4.4439e-07i	-0.012391-0.0032769i	-4.4827e-05-2.9944e-05i	-0.01115-0.0046912i
5	-0.00014586+0.00011735i	0.00028095-0.0003915i	-0.0001269+0.00012148i	6.8097e-05-0.00099755i
7	-4.4232e-05-3.0314e-05i	-0.14917+0.037664i	-0.00044129+7.6225e-05i	-0.14002+0.019552i
2	-0.00013314+0.00011124i	-0.00037344+0.0011114i	-0.00015294+0.0001398i	0.00016024+0.0015098i
4	-0.00012835+0.00011273i	-0.0019483-0.00053381i	-0.00021409+0.00013982i	-0.0019852+0.00025517i
1	-0.00014735+0.00014621i	0.00025017-6.2754e-05i	-0.00013432+0.00013596i	0.00018329-0.00017233i
3	-0.00013189+0.00014263i	-0.0004051-0.0009068i	-0.00014682+0.00013647i	-0.00071951-0.00069915i

From Alpha to Beta

Application Realization

The alpha version of the application is built on the Matlab platform. The calculation are based on the input data from CSV files.

For the beta version, the openECA platform will be integrated into the application. The input data will be formed as the standard measurement streams on openECA. C# project will be developed based on the data feeds from the openECA. The system configuration will be created and analyzed. The functionality of the CT/PT Calibration will also be realized as C# code.

The similar functionality test on Alpha version will be conducted again on the C# project.

User Interface Design and Realization

The user interface will be developed. Such interface will be designed as a universal media of all the three analytics including CT/PT Calibration, Transmission Line Impedance Calibration, and Real-time Impedance Calculation. The system topology will be demonstrated and the calculation results of different analytics will also be demonstrated.

From Alpha to Beta

Application Realization

The alpha version of the application is built on the Matlab platform. The calculation are based on the input data from CSV files.

For the beta version, the openECA platform will be integrated into the application. The input data will be formed as the standard measurement streams on openECA. C# project will be developed based on the data feeds from the openECA. The system configuration will be created and analyzed. The functionality of the CT/PT Calibration will also be realized as C# code.

The similar functionality test on Alpha version will be conducted again on the C# project.
User Interface Design and Realization

The user interface will be developed. Such interface will be designed as a universal media of all the three analytics including CT/PT Calibration, Transmission Line Impedance Calibration, and Real-time Impedance Calculation. The system topology will be demonstrated and the calculation results of different analytics will also be demonstrated.

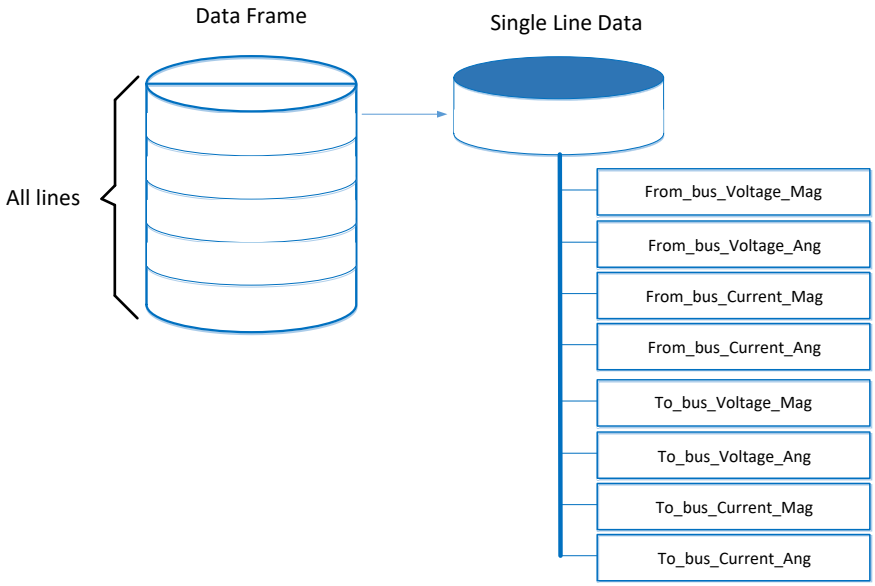
11 Real-time Transmission Line Parameter Calculator

In order to realize the analytic of real-time transmission line parameter calculator we firstly use PSS\E to conduct the power flow of the IEEE standard 118-bus system to gather the simulated voltage and current data; then, realize the functionality on Matlab to validate the methodology; finally, create measurements on the openECA platform and generate C# project to implement the analytic and demonstrate the calculation results on the test harness window.

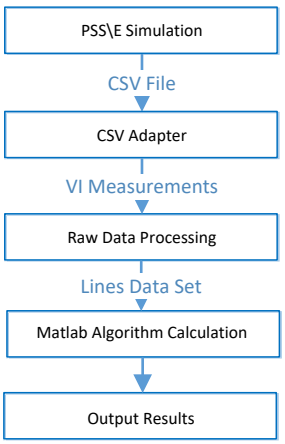
Test Results(Alpha Version)

Algorithm validation - Matlab

Data Structure



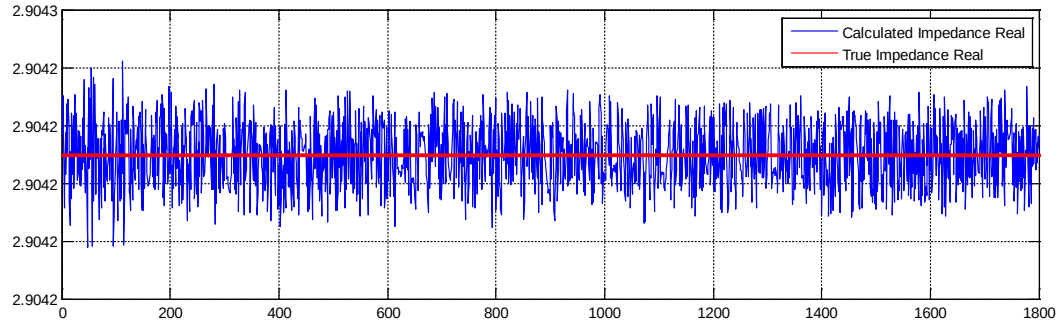
Validation Flow Chart

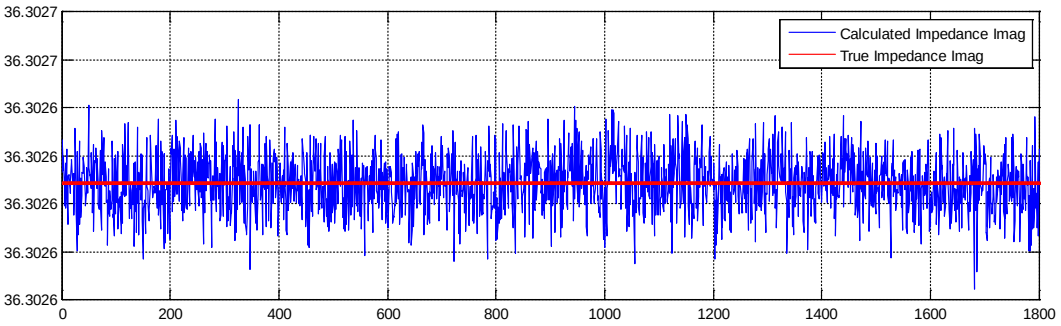


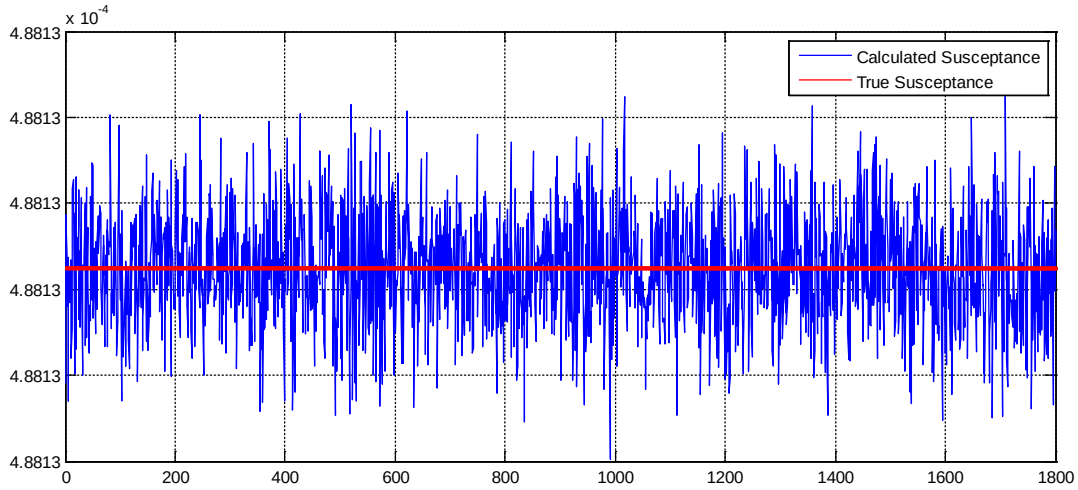
Calculation Results

The calculation results of the real-time impedance calculator are shown in the following figures. The application use the data frames which contain complex voltage phasors and current phasors of both sides of the concerned transmission line as the input. The line parameters are calculated every time one data frame is provided and 1800 times in total.

We can see that the line parameter results are not smooth with respect to time. But the actual error rates are at the 0.1% level. Such minor fluctuations indicate that the calculation is valid.

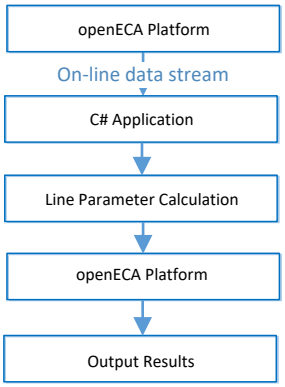




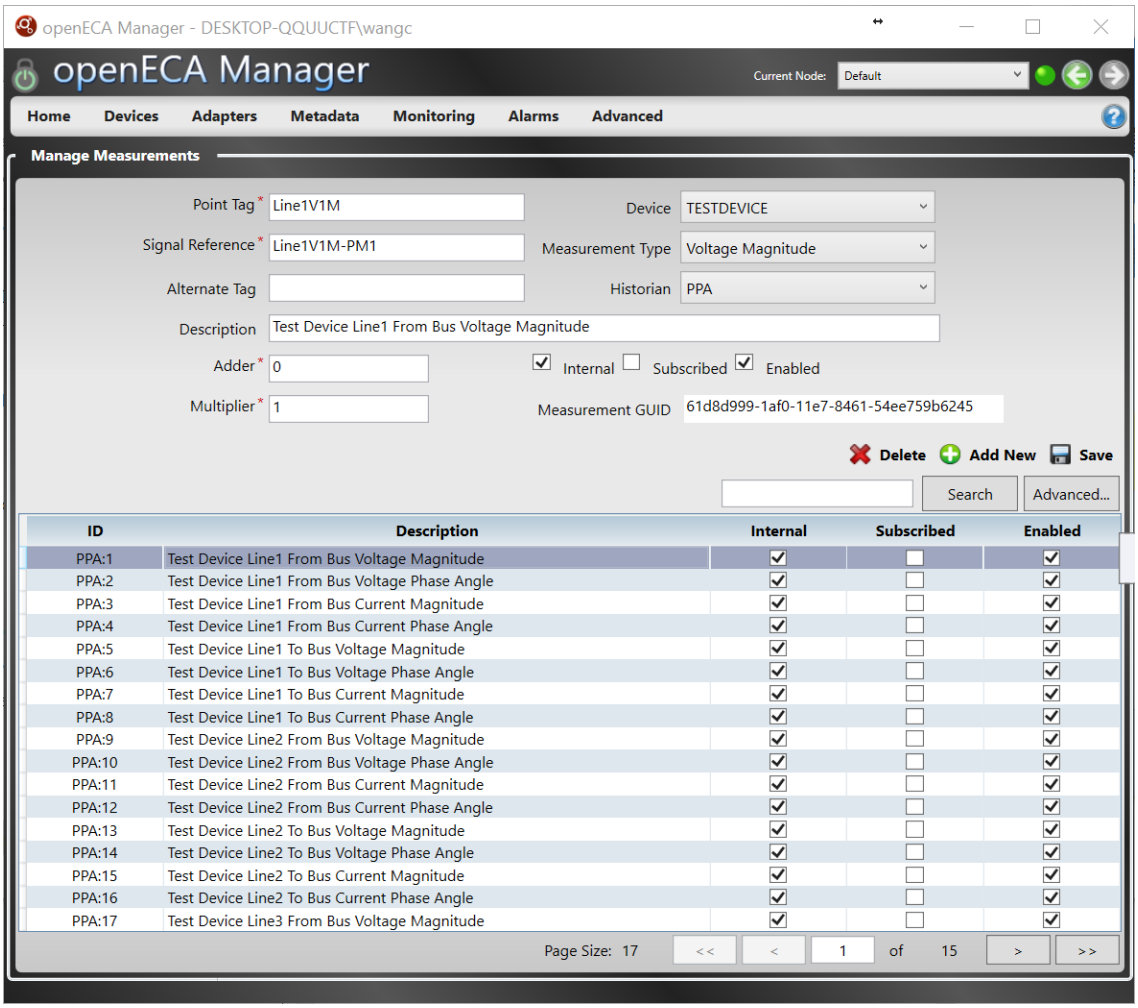


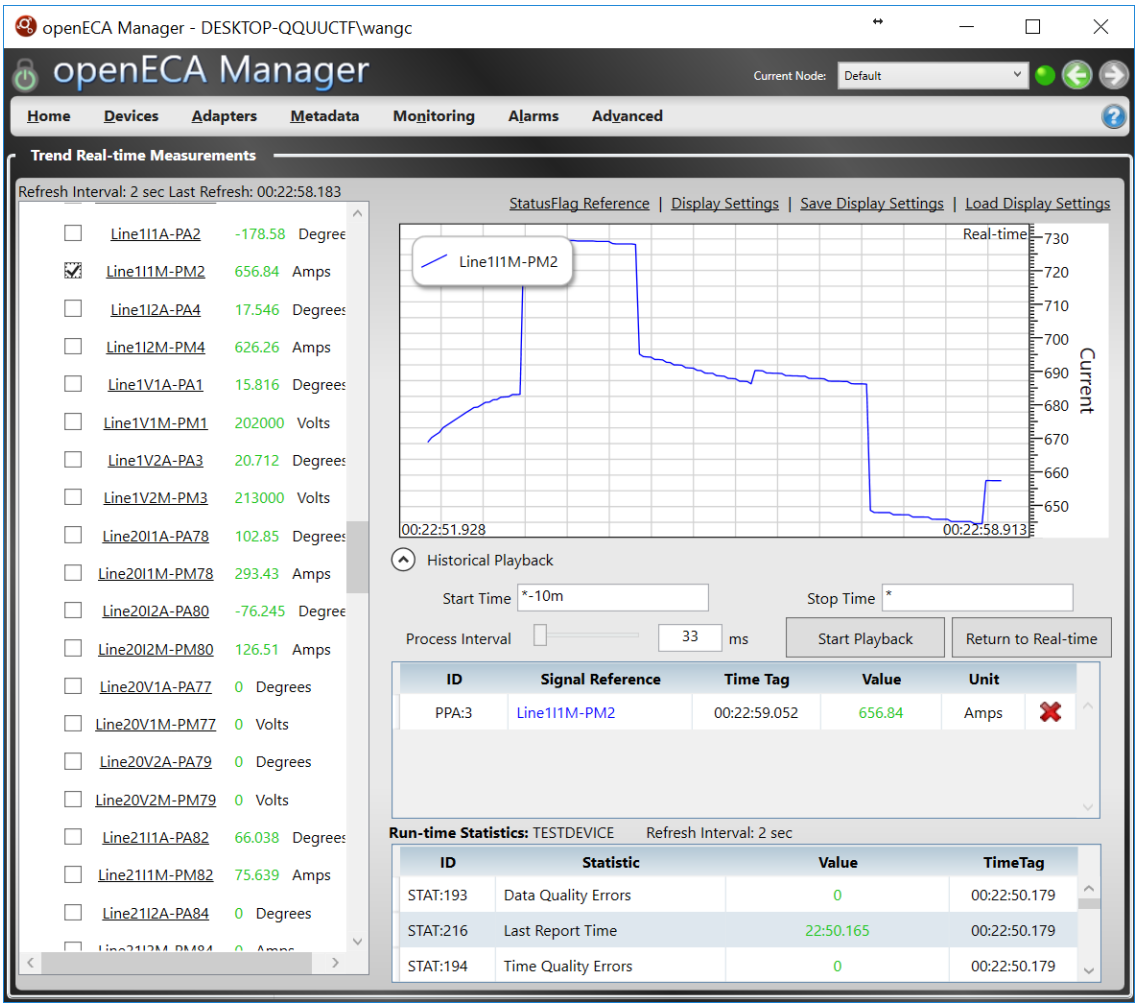
Application realization - openECA

Application Flow Chart



The simulated voltage and current measurements are integrated into the openECA platform as shown in the following figures.



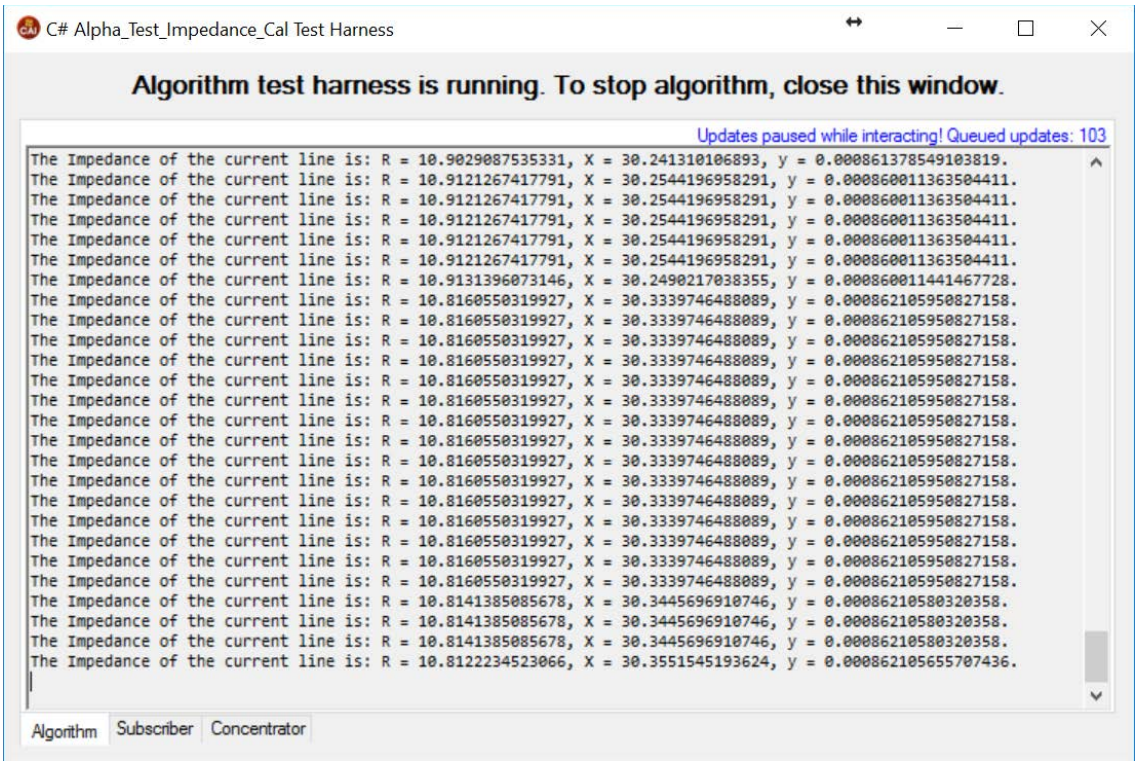


There have been some problems during the integration process. At first, considering the data set scale, we planned to create more than 190 measurements in the metadata. But the method of inputting the measurements meanually is much time-consuming and easy to cause errors. After communicating with GPA, we were provided the suggestion of utilizing MySQL base script to complete the configuration of the openECA platform. Such method does provide an efficient way to create and alter large scale of measurements.

The second problem we met is that for the Alpha version, the CSV adapter provided by the platform has some restrictive script and raw data file requirements. Most of the restrictions has been identified after the communication with GPA. The CSV adapter is still the most reliable and efficient way to upload local database

There is also a problem awaiting fixed. When using the openECA client generating C# projects, we found that not all the data channels created in the manager can be found and maaped to the objects defined. We are still seeking the inner logic and solution to this issue.

The C# project is generated from the openECA client. Corresponding algorithm is realized in the project and the calculation results are shown in the test harness window as following:



The calculated line resistance, reactance, and susceptance are physically reasonable, stable, and close to the true value. And for future versions, the user interface will be improved.

From Alpha to Beta

Application Realization

The alpha version of the application is only dealing with one transmission line parameters calculation.

For the beta version, the system configuration will be created and analyzed. The computation will be conducted on all the transmission lines that have enough voltage and current measurements.

User Interface Design and Realization

The user interface will be developed. Such interface will be designed as a universal media of all the three analytics including CT/PT Calibration, Transmission Line Impedance Calibration, and Real-time Impedance Calculation. The system topology will be demonstrated and the calculation results of different analytics will also be demonstrated.

12 SYNCHRONOUS MACHINE PARAMETER ESTIMATION

Test Approach

The basic test configuration will be modified for testing of this analytic. The analytic is currently prototyped in Matlab and is in the process of being ported to .NET. Testing will be conducted on the Matlab prototype. Testing will be repeated within the openECA framework at a later date. Configuration information will be programmatically inserted into the analytic. Each test case requires parametric metadata for proper functionality. The parameters will be embedded within the analytic's code base for testing purposes. When "settings" facilities become available in openECA beta the test suite will be modified accordingly.

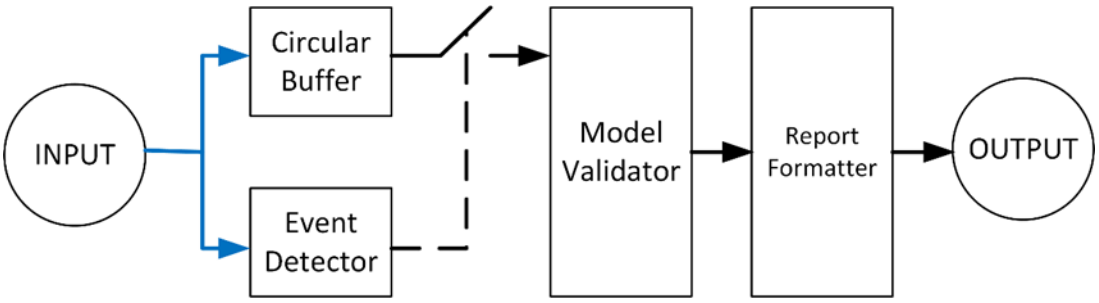
Test Environment

The test platform will be 64-bit Windows 7 configured as a workstation running on an Intel i7-3770 with 16Gb of memory and 8 cores. The software will be compiled under Visual Studio as "Any CPU" targeting the .NET Framework compatible with the openECA client compilation.

Analytics Overview

As with line impedance parameters, improving the quality of synchronous machine model parameters will provide benefits both in planning and operations. It has recently been shown that an effective method to identify and validate synchronous machine model parameters is to compare and match event signatures captured by PMUs against simulated event signatures generated by the machine model under test. This periodic analytic component will automate the process of synchronous machine model parameter estimation and validation – a process which is currently labor-intensive and which requires expertise from highly skilled personnel.

In the openECA use case, a set of synchronous machine parameters supplied in a user-provided configuration are occasionally tested against simulation results. The validation routine runs when the analytic detects a system event that sufficiently excites the synchronous machine parameters. It is conceivable, therefore, that this analytic may only return a result a few times per year. The ultimate goal for this analytic is to facilitate automated validation for the purpose of assisting utilities with NERC MOD027 compliance. MOD027 requires that utilities validate the models of major generating units every five years.
A signal flow diagram of the analytic is shown below.



Pre-Beta Features to be Tested

The synchronous machine parameter estimation analytic delivers the following outputs.
Feature 1: Analytic detects discrete power system events suitable to initiate validation routine.

Feature 2: Analytic buffers appropriate amount of data for analysis, and appropriately locks the buffer when an event is detected.

Feature 3: Analytic performs the validation routine to acceptable accuracy upon event trigger.

Tests Conducted

Test 1: "Analytic properly detects events" This test determines whether the analytic properly detects events of sufficient magnitude such that the validation routines can accurately assess the correctness of the model. This will be a fixed dataset test. With analytic properly configured a dataset containing simulated events with differing energy content will be fed to the event detection routine. Only events with sufficient energy content should trigger the detector.

Implementation: Representative data sets from system events have been selected and formatted. These representative data sets have been fed to the "front end" of the analytic to test the event trigger logic. The representative event must contain sufficient spectral content to excite all states of the model to be examined. The test passes if the event detector logic triggers on a robust data set and does not trigger on a quiescent data set.

Status: In Process. Initial results indicate promise of success.

Test 2: "Circular buffer functions properly" This test determines whether the circular buffer properly stores pre- and post-trigger information for use by the validation routine. This will be a fixed dataset test. With analytic properly configured a dataset will be fed to the buffer routine. When the buffer routine is exposed to a simulated trigger it should store a snapshot of data with appropriate duration.

Implementation: When an event is detected, e.g. as a result of the functionality tested in Test1, the analytic must then perform signal processing on a buffer of data from the previous minute of data. Therefore, the analytic must keep a buffer of data in memory. For this test a simple dataset was created and fed through the analytic. Upon a triggered event, a snapshot of the buffered data was passed to the analytic's signal processing routine.

Status: In Process. Initial results indicate promise of success.

Test 3: "Analytic accurately assesses model parameters" This test assesses the accuracy of the model validation routine. This will be a fixed dataset test. With analytic properly configured, including a synchronous machine model appropriate for the dataset(s), a dataset containing simulated events will be fed to the analytic. The test passes if the analytic validates the model to an acceptable precision.

Implementation: Actual and simulated data sets will be selected and formatted. These data sets will be processed by the analytic's signal processing engine and checked for accuracy.

Status: In Process. We are working with BPA to obtain a library of robust data. The setup for this step is complex because we must have both an accurate model as well as a data set with sufficient bandwidth.

13 ACCELERATION TREND RELAY ENHANCEMENTS

Test Approach

The analytic, as originally proposed, is experimental. It is unclear at this time whether synchrophasor measurements delivered with delays of up to 30ms can be used to augment ATR functionality. The ATR currently uses only local information at much higher sample rates than can be provided by a remote PMU. Initial results are promising. Because this analytic is experimental it is not yet ready for testing within the openECA environment. Tests will be conducted in Matlab on data sets produced by simulated and with parameters and configuration information provided by Northwestern Energy.

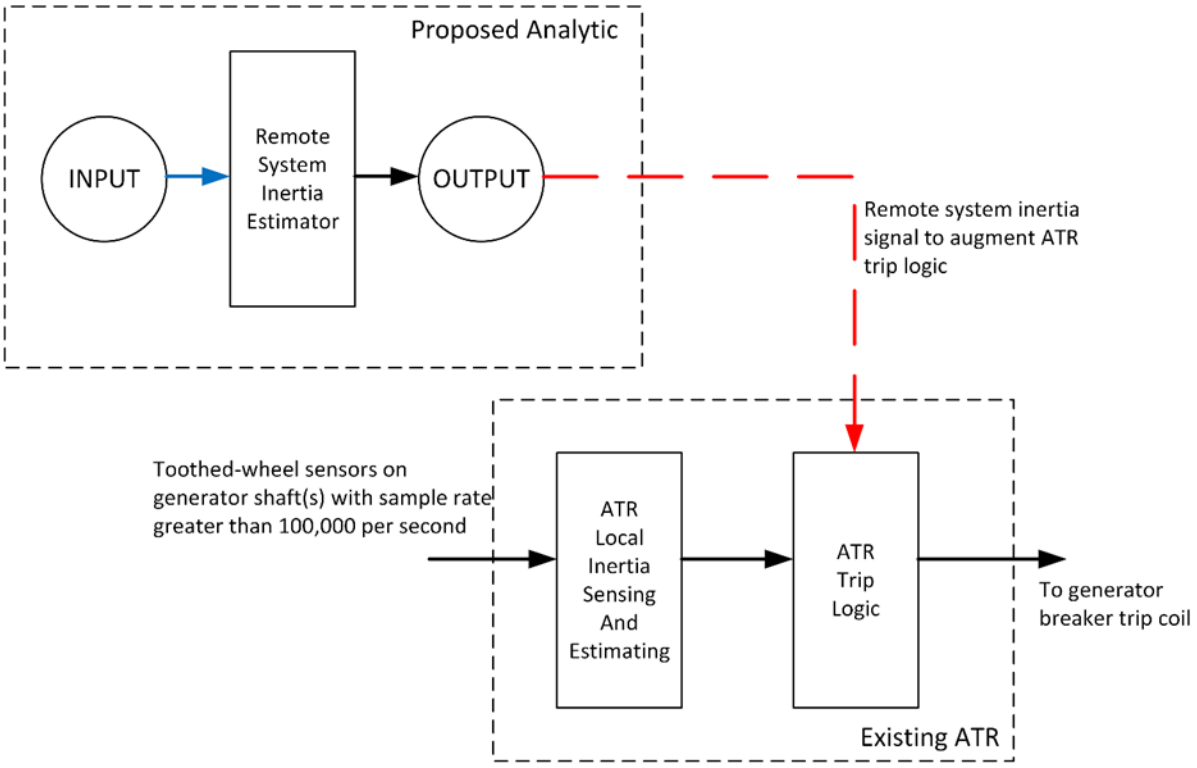
Test Environment

The test platform will be 64-bit Windows 7 configured as a workstation running on an Intel i7-3770 with 16Gb of memory and 8 cores. The software will be compiled under Visual Studio as “Any CPU” targeting the .NET Framework compatible with the openECA client compilation.

Analytics Overview

Most power plants are tightly coupled to a mesh of high-voltage transmission. Some, however, are by necessity located at the end of a loosely-coupled and radial transmission line. In the latter case, and particularly when the power plant is a thermal unit, a serious risk of catastrophic loss of synchronization exists. The Acceleration Trend Relay (ATR) is designed to protect a unit from loss of synchronization if it detects rapid acceleration. ATR functionality can be greatly enhanced by augmenting the shaft speed signals with remote phasor measurements. An ATR will trip a generation unit when the unit is detected to have a high probability of losing synchronization with the grid. This analytic, if validated by offline studies, has the potential to improve the accuracy of an ATR by reducing the number of false positives attributed to the ATR.

This module will be used as an important use case to demonstrate the development of specialized openECA adapters to conduct issue-specific analysis of phasor data and will serve as a template for subsequent development. A signal flow diagram is shown below.



Pre-Beta Features to be Tested

The synchronous machine parameter estimation analytic delivers the following outputs.
Feature 1: Theoretical effectiveness of the proposed analytic.

Tests Conducted

Test 1: "Theoretical effectiveness of incorporating phasor measurements into ATR trip logic" This test determines the theoretical effectiveness of the proposed analytic. Initial results are promising, however there is still a possibility that remote synchrophasor measurements delivered via network technologies are not fast enough to provide substantive benefit. This will be a fixed dataset test. With analytic properly configured a dataset containing simulated events will be fed to the analytic. Success will be through a subjective determination of whether the incorporation of synchrophasor data improved the performance of the ATR.

Status: In Process. A reduced-order power system model capable of simulating transient stability events has been created. A Matlab representation of the ATR as it exists today at the Colstrip power plant hase been created. The research question is whether remote information provided by a PMU can enhance and improve the existing ATR algorithm thereby improving bulk grid reliability. Hundreds of simulations have been conducted on the combined system/ATR model. The project team has found initial results that indicate *false trips* may be avoided by using remote information as a reference input to the ATR. Meetings with Northwestern Energy are planned for next quarter. Publication of the results is also planned for summer 2017.