An Analysis of Control Area Synchronization on the Convergence of Distributed State Estimators

by
Blaise Clarke

A THESIS

submitted to

Oregon State University

University Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Electrical and Computer Engineering
(Honors Scholar)

Presented Feburary 25th, 2016
Commencement June 2016

AN ABSTRACT OF THE THESIS OF

Blaise Clarke for the degree of Honors Baccalaureate of Science in Electrical and Computer Engineering presented on Feburary 25th, 2016. Title: An Analysis of Control Area Synchronization on the Convergence of Distributed State Estimators.

Abstract approved:

____

Jinsub Kim

This honors undergraduate thesis examines the process of simulating power system state estimation, the use of the alternating direction method of multipliers in the problem of distributed power system state estimation, and provides a rudimentary asynchronous implementation of the above for a linearized DC power system. The asynchronous implementation is applied to a model of the IEEE 57 bus test system; convergence to a set of centralized state estimates and to the true state of the system is examined and compared to the results of a control synchronous implementation of the ADMM algorithm whose iterations were restricted by the weighting factors of the asynchronous method. We find that the state estimation from the asynchronous ADMM algorithm satisfactorily converges to estimates done by a centralized state estimator as well as the real states of the system; we conclude with future work that can follow from these conclusions.

Key Words: Power systems, Asynchronous distributed state estimation, Alternating Direction of Multipliers Method

Corresponding e-mail address: Clarkebl@oregonstate.edu

An Analysis of Control Area Synchronization and Organization on the Convergence of Distributed State Estimators

by
Blaise Clarke

A THESIS

submitted to

Oregon State University

University Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Electrical and Computer Engineering
(Honors Scholar)

Presented Feburary 25th, 2016.
Commencement June 2016

Honors Baccalaureate of Science in Electrical and Computer Engineering project of Blaise Clarke presented on Feburary 25th, 2016

APPROVED:

Jinsub Kim, Mentor, representing Electrical and Computer Engineering

Roger Traylor, Committee Member, representing Electrical and Computer Engineering

Thinh Nguyen, Committee Member, representing Electrical and Computer Engineering

Toni Doolen, Dean, University Honors College

I understand that my project will become part of the permanent collection of Oregon State University, University Honors College. My signature below authorizes release of my project to any reader upon request.

Blaise Clarke, Author

Table of Contents

An Analysis of Control Area Synchronization on the Convergence of Distributed State Estimators

Blaise Clarke, *Student, Oregon State University*

Abstract

This honors undergraduate thesis examines the process of simulating power system state estimation, the use of the alternating direction method of multipliers in the problem of distributed power system state estimation, and provides a rudimentary asynchronous implementation of the above for a linearized DC power system. The asynchronous implementation is applied to a model of the IEEE 57 bus test system; convergence to a set of centralized state estimates and to the true state of the system is examined and compared to the results of a control synchronous implementation of the ADMM algorithm whose iterations were restricted by the weighting factors of the asynchronous method. We find that the state estimation from the asynchronous ADMM algorithm satisfactorily converges to estimates done by a centralized state estimator as well as the real states of the system; we conclude with future work that can follow from these conclusions.

Index Terms

Power systems, Asynchronous distributed state estimation, Alternating Direction of Multipliers Method.

I. Power System State Estimation

A. Power Systems

FAST and accurate state estimation is a critical part of the creation and maintaining of electrical power systems. As the population of a region grows, the demand for power increases - though this increase in consumption can be partially alleviated with the introduction and regulation of new power generators to the power grid, there are still limits on the current infrastructure of transmission lines, substations, and other power system elements that allows for less leeway on the mismatch of power being generated and the demands of the areas being serviced. These tighter tolerances require an increased precision in the measurement of the electrical state of the system - with higher amounts of generation and load, small systemic or localized errors that were previously acceptable in smaller systems may compound and cause a larger system to fail. Furthermore, as smaller, distributed sources of electrical generation such as wind or wave energy are integrated into applicable areas, the local grid grows increasingly complex to measure and balance by hand; additionally, operators who manually equalize generation with load are still influenced by measurements that may be missing or corrupted by noise. [1]

B. Why use state estimators?

The most direct way an engineer can determine how to modify power generation for a given area is to directly measure the voltage and current phasors of every element in the system that are generally achieved through very accurate and synchronized phasor measurements of all bus voltages and branch currents in the system; however, this approach is very vulnerable to measurement errors or telemetery failures of the meters themselves. Additionally, it is not guaranteed that every meter will be able to make a measurement, accurate or not, at any possible time an engineer controlling the power grid may require - an operator must make a guess whether or not the measurement being acquired by the meter is still relevant to the system at the time the request was made. [2] These issues mean that a brute-force method of direct measurements is in practice difficult to implement and does not scale well. Instead, the system can be broken up into a set of interconnected subcircuits or network of elements, each of which can be characterized by a single input or output voltage phasor. As there is a shared input and/or output for all of the elements and the network infrastructure is fixed in place barring maintenance or repair, an engineer can simply use Kirchoff's junction and loop rules to determine the desired voltage and current phasors of a single component of the subcircuit; that is, even if the entire system is not locally observable, it can be broken into electrical subsystems that in themselves are individually observable. As the engineer wants to control the power being delivered to a specific component in order to avoid damage through too much power dissipation or to avoid the loss of service due to too little power being supplied to an area. As these direct readings can be estimated using the given input or output voltage phasor, the phasors thus act as state estimators for the entire subcircuit, avoiding many of the scaling issues that direct measurements run into, as well as reducing the points of entry that an unauthorized person could use to access the state of a power system. For this simulation, we will generalize each power system into a set of interconnected buses each of which is describable by a single voltage phasor; the voltage phasors of the buses will act as the true state of the system, and the estimates of the voltages for each bus/state from the various algorithms put forth in this thesis will act as state estimates for the entire electrical power system. [3] Using state estimators rather than directly measuring the voltage across or current through every element within a power system in the region can also reduce the effects of measurement error inherent to the meters being used by implementing models focused on minimizing the mean squared error of state estimates from test readings (weighted least squares methods), minimizing the variance (minimum variance methods), or maximizing the chance that the estimate is within a given tolerance of the actual reading (maximum likelihood methods). All of these algorithms use redundant neighboring measurements in order to improve their specific metrics, as well as provide estimates on the error generated by each meter in the area, filter out small errors due to model mismatches and measurement inaccuracies, and detect major, unexpected errors that may be caused by a bad data attack. Additionally, it is substantially easier to base a decision on a single voltage phasor for a large electrical system rather than using an algorithm that compensates for every single constraint of the subsystem or by implementing several algorithms that only have local control of parts of the subsystem, making those algorithms locally optimize changes to generation, and then forcing the partial results from each algorithm to be compatible with the aggregate results. From this, it is easy to see that not only do state estimators allow us to simplify the description of a power control system, they also simplify decision making for that same power system, as well as reducing the effort needed to view the state of a system at any given moment. [2]

C. *Linearized DC modeling*

For this thesis, we will be examining a linearized DC model of power systems as a proof of concept for the use of an asynchronous alternating direction method of multipliers algorithm for state estimation. By using the linearized DC model, we greatly simplify the system and can look at the state of a given bus as a magnitude rather than a phasor; additionally, we force our measurements to be time-invariant and not subject to time skew present in AC models. [4] This second advantage is key to our simulations, as we are relying on a number of successive iterations to converge to a final estimate of the system; the addition of time-varying elements to our state estimation calculations would add a time factor that is not necessarily synchronized to the iteration delays and would force a more intricate method of modeling that is inappropriate for this level of analysis. Finally, I frankly do not have enough experience with non-linear optimization or the power systems knowledge to accurately improve the model to include non-linear modeling or AC analysis of a system; the main focus of this thesis is to act as a starting point for someone more knowledgeable than myself for further investigation of the subject.

II. Approaches Used for State Estimation

A. Centralized State Estimation

The simplest method of estimating the state of a power system is to designate a controller node that is responsible for gathering all the data from its surrounding buses and meters, collating the information, and giving an overall estimate of the voltage phasor that acts as a state estimator. The main benefit for this method of power system state estimation over other approaches is that it only requires the vector of current state measurements **z** and the Jacobian matrix of the system **H** by using the Newton-Raphson solution method [5] - **H** is composed of the partial derivatives with respect to the magnitude and phase of the voltage phasor of the in-phase and out of phase components of real power injection (generation) at each bus, the in-phase and out of phase power flow on the branches connected to each bus, and the partial derivatives of the magnetic effect of the above flows. The composition of this matrix is outlined in the *Acquisition of Parameters from the Power System* subsection of the below section. However, the linearization of the system greatly simplifies the process of generating the partial derivatives and makes **H** relatively sparse, and the steady state estimation being done means that the Jacobian only needs to be calculated once, with no iterative updates as the system changes.

For small systems, this approach is adequate. However, the typical implementation of centralized voltage estimation for power control systems only scale well to a certain extent - as more elements are added to the power system undergoing state estimation, there is a greater delay between a change in the measurement from an outlying area solely due to the longer distance that communications take to get to the centralized state estimation processor. Another major drawback to this method is the processing power required in the controller node; as the system grows, the processing power needed to give a state estimate with the same approximate accuracy and frequency as a smaller system grows at a $O[n^2]$ pace, where n is the number of buses in the system being modeled. [5] Once the size of the system grows past an example such as the IEEE 300-bus ”East Coast” model, the cost of implementing a full-scale centralized state estimator of incredibly high precision becomes prohibitive for worst case modeling. [6] Though this is unlikely to impact someone who is testing different algorithms on these models, it is definitely a concern for real life applications of a centralized state estimator. Additionally, though consistency of measurements taken by centralized state estimation algorithms reduce the impact of noise and randomized error, it is still vulnerable to a concerted ”bad-data” attack wherein an attacker deliberately injects incorrect measurements into a system over a period of time in order to force the system into emergency or recovery states, either of which would require extended periods of brownouts or blackouts in order for the power system to return to a stable mode of operation. [7]

B. Decentralized (Synchronous) State Estimation

In order to avoid some of the issues inherent to centralized state estimation in very large power systems, investigation into a decentralized method of power system state estimation has become increasingly relevant over the last decade. [8] The simplest forms of decentralized state estimation involve breaking a large system up into several control areas, each with their own localized measurements and processing node. Each of these control areas operate under an algorithm analogous to a centralized state estimator for their one region. Once a state estimate for the entire control area is found, the areas communicate with one another in order to iterate to a state estimate of the entire system. This methodology has several advantages over the simpler centralized state estimation described above.

- Since each control area is relatively small, a decentralized approach does not have as much of a scaling issue as a centralized approach; as a network grows larger, the number of control areas for the network can grow in addition to the size of each area. The cost of communication between control areas does increase in this situation, but only at the rate of $O[n \times log(n)]$, where n is the number of control areas of the system being simulated. [9]
- If a single control area has enough independence to make changes to its own power generation, the local response to a sudden change in the demand of electrical power can be substantially faster than in a centralized approach. Instead of waiting for a decision to be made on the entire network as a whole based on a centralized state estimation, a control area can find that its local state estimates require immediate action and correct the problem quickly. [10]
- There is some protection against bad data attacks with a decentralized state estimator. For the worst case examples of a bad data attack not being detected and negated by the estimator for the area, the organization of the power systems network into distinct control areas means there is a much higher chance that damage to the generators and other infrastructure elements will be limited to only the control areas where the bad data attack was based as well as system elements shared by that control area with its neighbors. [7]

Despite its many benefits over the centralized state estimator, decentralized state estimation still has multiple drawbacks. For one, because the leaders of each control area must share their localized state estimations with one another and iterate together to a system-wide state estimate, any non-localized estimates will be slower than in the centralized estimate in any case but the most trivial; the only time a decentralized state estimate would be created faster than its equivalent centralized estimate is if only a handful of iterations are used, which only occurs in the simplest of test cases. Additionally, since the local estimates created by each control area are communicated over the same branches that measurements are taken from, any issues with noise or disruption between buses shared by multiple areas or on branches that connect two areas are amplified as they impact the convergence to the centralized estimate; however, if a robust estimation system is being used, the impact of noise and disruption will be minimized as part of the algorithm. Finally, the decentralized power system state estimation is still vulnerable to chronic bad data attacks if they are focused in a single control area and not intense enough to outright pull the control area into a state of failure - if a control area is consistently put into a recovery state, it will still communicate bad data and request more assistance from its neighbors, spreading the effects of the attack. [11] Though the method of iterating to a overall state estimation of the power system mitigates the effects of bad data being injected to any number of control areas over a short period of time, it still cannot distinguish between chronic attacks and actual, denoised measurements from its meters.

C. The Alternating Direction Method of Multipliers

There are many algorithms that can accomplish a decentralized state estimate for power systems - however, one family of algorithms that has been studied heavily in recent years is the alternating direction method of multipliers (ADMM). In essence, ADMM algorithms integrate the good convergence rates of the method of multipliers algorithms and the decomposability of dual ascent methods. ADMM algorithms solve the general family of functions that minimize $f(\mathbf{x}) + g(\mathbf{z})$ subject to the constraints $\mathbf{Ax} + \mathbf{Bz} = \mathbf{c}$, where $\mathbf{x} \in \Re^n$, $\mathbf{z} \in \Re^m$, $\mathbf{A} \in \Re^{p \times n}$, $\mathbf{B} \in \Re^{p \times m}$, and $\mathbf{c} \in \Re^p$. The structure of this minimization problem is similar to the general case of minimizing $f(\mathbf{x})$ with the constraint $\mathbf{Ax} = \mathbf{C}$, where vectors $\mathbf{x}$ and $\mathbf{c}$ and matrix $\mathbf{A}$ are defined above; the main difference is the separation of the vector of variables $\mathbf{x}$ into two separable vectors of variables $\mathbf{x}$ and $\mathbf{z}$ respectively. [12] This allows the user to update each vector of variables independently of one another, allowing for the minimization of the first vector using a static value for the second vector, the minimization of the second vector with the first fixed at another static value, and the updating of a dual vector of variables that forces the two minimization results to a compromise value between the two - this acts as the starting point for the next iteration of the ADMM algorithm. In essence, it is an abstracted version of the dual ascent method of minimization. Typically, the minimization steps are performed using an augmentented Lagrangian submethod; however, instead of jointly minimizing $\mathbf{x}$ and $\mathbf{z}$ together, each variable is updated sequentially in an alternating pattern ($\mathbf{x}$ then $\mathbf{z}$, or $\mathbf{z}$ then $\mathbf{x}$), which brings about the name of alternating directions of multipliers.

It can be shown that for the general form of the alternating direction method of multipliers, convergence will eventually result given that the functions $f(\mathbf{x})$ and $g(\mathbf{z})$ are closed, convex functions and that the Lagrangian L of the system has a saddle point (there exists some point $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ such that $L(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}) \leq L(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*) \leq L(\mathbf{x}, \mathbf{y}, \mathbf{z}^*)$). [13] From the first of these assumptions, the implication that subproblems formed by the decomposition of the problem into x-update and z-update steps are both solvable can be seen; that is, there exists some $\mathbf{x}$ and $\mathbf{z}$ that result in the global minimization the augmented Lagrangian of the system as both functions $f(\mathbf{x})$ and $g(\mathbf{z})$ are closed and convex. From this assertion, we can see that the exists a finite saddle point $L(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ on the Lagrangian because such a minimization exists. This means that $\mathbf{x}^*, \mathbf{z}^*$ exists as a solution to the original problem of minimizing $f(\mathbf{x}) + g(\mathbf{z})$, given $\mathbf{Ax} + \mathbf{Bz} = \mathbf{c}$, which leads to the conclusions that $\mathbf{Ax}^* + \mathbf{By}^* = \mathbf{c}$ is valid for $\mathbf{x}^* < \infty, \mathbf{y}^* < \infty$. An explicit derivation of the general proof of convergence can be found at [14], and specific proofs of convergence for ADMM in power systems can be found in [7], along with most papers using this family of algorithms for power systems.

D. Synchronous vs. Asynchronous ADMM

The majority of implementations using a variant of the alternating direction method of multipliers algorithm apply them in synchronous applications for a variety of reasons. For one, it is easy to visualize a system as a large collection of segmented subsystems working in lock-step on partially optimizing the system using variables local to them; it follows a basic iterative process of decompose, optimize, share results, and re-optimize that can be monitored to see each area converges to results consistent with the rest of the system at every iteration. Furthermore, it is substantially easier to find proof of convergence or proof on the rate of convergence as a heuristic of a given application of a synchronous ADMM algorithm by using well-defined iterations, something substantially more difficult if an asynchronous schema is being used. If not all subsystems of the system are updating variables at the same time, it is difficult to say when one updating iteration ends and the next begins. From this, based on the type of asynchrony being used, it can be tougher to simulate asynchronous ADMM implementations. The simplest form of asynchronous implementations to simulate is one based around different subsystems only being active a portion of the time; when inactive, the next iterative state is identical to its previous iteration. This methodology can be split into two different subtypes, probability based methods versus event driven ones.

Event driven ADMM implementations consist of update occurring during iterations only when specific event conditions are met within the subsystem, such as a control area receiving a certain number of updates from its neighbors, a given number of iterations passing since the control area last updated or received an update, or if a controller detects interference under a given threshold in the channel the control area communicates its updates over. [15] Initially, the simulation for this thesis used an event-driven approach that caused an update to be sent out when a control area detects that it has received an update from a majority of its neighbors or if it had been 20 iterations since the last update was received from any neighbor. However, this method ran into issues where state updates were very densely packed together with several iterations between them with no updates from any control area. This made it clear that after a short period of time it devolved into a slowed, semi-synchronized ADMM implementation due to the second condition on updates. When this was removed in order to avoid the gradual synchronization of updates, the algorithm stalled after a handful of iterations with active updates without coming close to converging; when the condition was made more strict, the algorithm became even more strongly synchronous early in the iterative process. Additionally, it is nearly impossible to compare the time to convergence of a properly built event driven asynchronous ADMM algorithm with a synchronous implementation; there is no consistent context for iteration values in a truly event driven system. In order to avoid this gradual synchronization, inability to compare to synchronous systems, and other issues, I switched to a probability based algorithm for the final thesis work.

Probability based methods are centered around the control areas of a system having a given probability per iteration of being active. [16] In this case, an active control area updates its local state estimation vector and sends out the updated estimate to its neighbors; if it is not active, the power systems state estimate is the same of the last estimation made when the control area was last active. In theory, the activation probability of each control area should be based around the slowest updating meters within the control area - the less often the slowest meter can give updates to the organizer of the control area, the less often that control area should be able to update and thus the lower the probability that it should be active per iteration. This is consistent with how more realistic synchronous alternating direction method of multiplier algorithms are built; updates only occur when a new reading from each meter has been taken in by the control area, which is bottlenecked by the slowest updating local meter. [17] Due to this, it is relatively simple to see that a probability based implementation of the asynchronous ADMM algorithm will devolve to a synchronous implementation by setting the probability that each control area will be active per iteration to 100% - with a control synchronous implementation for comparison, we can see that the devolved asynchronous state estimation matches the synchronous estimation at every iteration. Additionally, we can make a mock synchronous system by setting each weighting value to the largest of the set of the asynchronous algorithm that allows us to directly compare how many iterations a synchronous implementation still bottlenecked by the weakest control area would take to converge; the alternative would be to see how few updates it takes for a synchronous implementation of the alternating direction method of multipliers to converge, running a version of the asynchronous implementation that only allows each control area to that limited number of times, and looking at the mean squared error of the output of the limited asynchronous algorithm. [15]

III. Application of the ADMM to Power System State Estimation

A. Acquisition of Parameters from the Power System

Before any sort of implementation of the ADMM algorithm can be used, several parameters related to the power system being simulated must be acquired. For this thesis, we began with data in the IEEE Common Format for Load Flow Data, made available at [18]. From this data, we directly get information on the base MVA rating of the system as well as data structures filled with information on the buses, branches, generators, and generator costs. From those structures, we pull information about the number of branches, buses, and meters for the segmentation of the system into control areas for the distributed algorithms. From the bus matrix, we isolate the transformer and line charge information in order to transform them into their equivalent transmission line equivalents of susceptances and conductances.

- For transformers, if a transformer with turn ratio α and serial impedance Z_t is on the branch connecting buses m and n with nominal voltages V_m and V_n respectively, the current $\mathrm{I_m}$ from m to n at bus m is changed as follows: $\alpha^{-1}\mathrm{I_m} = \alpha^{-1}(\frac{V_\mathrm{m}}{\alpha} - V_\mathrm{n})(Z_\mathrm{t})^{-1}$. Since both susceptance and conductance terms correspond to $(Z_\mathrm{t})^{-1}(V_\mathrm{m} - V_\mathrm{n})$, they are proportional on $\mathrm{I_m}$ and must be changed accordingly. Similar corrections for the current running from bus n to m must be made as well; use the above adjustments with the two buses transposed. [1]

- Line charge information is treated as shunt capacitance in the system matrix. For every branch with line charge susceptance, the buses on each end of the branch have their susceptance increase by half of that of the line charge parameter. So, parsing through the entries of the branch data structure that correspond to line charge information, every time a given bus appears in either the front bus or tail bus columns, take the corresponding susceptance entry, divide it by two, and add it to the susceptance of the bus in the system matrix. [1]

Once the effects of transformers and line charge have been applied to susceptance and conductance elements of the matrix, the DC model of the system can be built. The admittance of the linearized DC system is simply the combination of the susceptances and conductances entered previously; the conductance matrix is the real portions of the admittance, while the susceptance matrix is the imaginary portion of the admittance. The DC system can be built directly using the susceptance matrix; the diagonal elements of the DC system matrix are the sum of the of the non-diagonal entries in the relevant row of the susceptance, while the non-diagonal elements of the DC system matrix are the opposite of the susceptance element values. [1] Because we are using the first node as a reference node, we can strip the first row and column from the system matrix for the creation of other power system parameters for meter and state estimates. Using the branch-to-phase incidence matrix and the matrix of susceptances, we can build the DC Jacobian matrix. Because we are only using a DC linearized model, we only need to consider meters for the real power flows and real power injections on each bus; from the overall DC Jacobian matrix, take the elements that correspond to the above and concatenate them using the set of lines (l) and buses (b) of the system to create the **H** matrix. The rows of **H** correspond to the states of the system being estimated; the first set of l elements in each row of **H** relate to the positive real power flow elements of the system; the next set of l elements relate to the negative real power flow; and the last group of b elements correspond to the real power injections on each bus. This results in a Jacobian matrix of $|\mathcal{S}| \times |2l + b|$, where $\mathcal{S}$ is the set of states of the system.

B. Centralized State Estimation of Power Systems

These system parameters are sufficient to perform linearized DC state estimation for our purposes. For the most basic centralized power system state estimation, we only need the overall Jacobian matrix **H** matrix formed in the above steps, and the vector of measurements from the system **z** as mentioned above in section 2. The latter is created from measurements taken directly from the system. In order to perform a centralized state estimate of a power system, do the following -

- Set a signal-to-noise ratio and a standard deviation for noise for the system. In this implementation, a SNR of 6000 and standard deviation of 0.001 were used.
- Calculate the mean signal energy of the system by using the nominal state of the system derived from the original system data.
- Calculate the noise of the system by taking the square root of average signal energy divided by the SNR.
- The true state of the system is calculated by adding variance based on the standard deviation set previously to the nominal state of the system.
- **z** is the noised vector of measurements from the system - calculate it by taking the true state of the system and adding noise that matches the previously set signal-to-noise ratio.

The centralized state estimation vector $\mathbf{x}_c$ is calculated as $\mathbf{x}_c = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}(\mathbf{H}^{\mathrm{T}})\mathbf{z}$. [17] If we were not doing steady state estimation of the power system, **z** would need to be recalculated and **H** would be updated every iteration; if the power system under voltage phasor estimation was non-linear rather than linearized, $\mathbf{H}$ would be substantially less sparse.

C. Decentralized State Estimation of Power Systems

When converting from centralized to decentralized state estimation, we need to separate the power system into several control areas that collaboratively perform the state estimation. Due to this, decentralized state estimators return a set of vectors $\mathbf{x}_k$ rather than a single state vector $\mathbf{x}$. Thus, in order to perform decentralized power system state estimation, extra information on the decomposition of system must be gathered before the algorithm begins. The most obvious of this is the number of control areas that the power system is being split into; for this thesis, this will be represented by k. From this, we can determine which subset of states from the entire system are associated with control area k, $\mathcal{S}_\text{k}$, as well as the meters associated with the control area, $\mathcal{M}_\text{k}$. Another caveat of the decentralized state estimation algorithms is that neighboring control areas can share line meters (and thus have access to measurements of bus meters and states on the exterior fringe of neighboring control areas), so the set of system measurements local to a given control area are not necessarily only available to that area and elements of the state vectors $\mathbf{x}_k$ are not necessarily unique. This is actually beneficial rather than a drawback - without these shared measurements, the control areas would only be locally optimizing the isolated regions that they have access to, and would not be guaranteed to converge to the centralized state estimation of the system as we desire. In essence, the algorithm would be solving k independent problems of the form

$$\min_{\mathbf{x}_k \in \mathcal{X}_\text{k}} \quad f_k(\mathbf{x}_k; \mathbf{z}_k, \mathbf{H}_k)$$

where f_k is a convex function of $\mathbf{x}_k$, generally chosen to be the maximum-likelihood estimate of $\mathbf{x}_k$ assuming Gaussian noise in the system (which is the assumption in this simulation); $\mathcal{X}_\text{k}$ is a convex set that captures prior information of the system such as operational limits or non-injection buses; $\mathbf{z}_k$ is the set of local measurements accessible to the control area; and $\mathbf{H}_k$ is the submatrix of the Jacobian relevant to the local elements in control area k. [7] Instead, the shared bus voltage and current measurements inherent in interconnected control areas of a power system makes the above set of direct minimizations into a single joint optimization problem of the form

$$\begin{aligned} \min_{\mathbf{x}_k \in \mathcal{X}_\text{k}} \quad & \sum_{k=1}^{K_{max}} f_k(\mathbf{x}_k) \\ \text{such that} \quad & \mathbf{x}_k[l] = \mathbf{x}_l[k], \ \forall l \in \mathcal{N}_\text{k}, \ \forall k \end{aligned}$$

where $\mathbf{x}_k$[l] and $\mathbf{x}_l$[k] are a state estimate shared by control areas k and l respectively and $\mathcal{N}_\text{k}$ is the set of meters in the group associated with control area k that are also visible to a control area l, k $\neq$ l.

The above optimization problem can be solved using an implementation of alternating direction method of multipliers algorithm, regardless of the synchronocity or lack thereof of the implementation. In doing so, the joint optimization devolves into a trio of simpler optimization problems to be performed every iteration. These three problems update the vectors $\mathbf{x}_k$, $\mathbf{s}_k$, and $\mathbf{p}_k$ based on minimizations listed in [21]. As mentioned before, the ADMM algorithm uses an augmented Lagrangian function $L(\mathbf{x}_k,\mathbf{s}_k,\mathbf{p}_k)$ from its basic roots in the direction of multipliers method; the Lagrangian for this simulation is set to $L(\mathbf{x}_k,\mathbf{s}_k,\mathbf{p}_k)$ = $\sum\limits_{k=1}^{K_{max}}(f_k(\mathbf{x}_k) + \sum\limits_{i \in \mathcal{N}_\text{k}} (\mathbf{p}_k^T(\mathbf{x}_k[\text{i}] - \mathbf{s}_k) + \frac{c}{2}(\mathbf{x}_k[\text{i}] - \mathbf{s}_k)^2))$, where $\mathbf{x}_k$[i] is element i in $\mathbf{x}_k$. [12]

- In the first step of each updating iteration, the state vectors $\mathbf{x}_k$ for each control area are updated by choosing those state vectors that minimize the augmented Lagrangian function given a fixed value for

$\mathbf{s}_k$ and $\mathbf{p}_k$; typically, these fixed values are taken from the vectors generated by the previous iteration of the algorithm.

- Updating the shared meter states $\mathbf{s}_k$ - Fix $\mathbf{x}_k$ using the values found in the previous step and fix $\mathbf{p}_k$ using the values found in the previous iteration, then minimize the augmented Lagrangian to find the new values of $\mathbf{s}_k$.
- Updating the dual variable $\mathbf{p}_k$ - find the change to all $\mathbf{x}_k$ this iteration using a gradient ascent method on $L(\mathbf{x}_k,\mathbf{s}_k,\mathbf{p}_k)$ using the difference between the newly calculated $\mathbf{x}_k$ and $\mathbf{s}_k$ vectors, weighted by a Lagrangian step size set by the user.

In a synchronous implementation of the alternating direction method of multipliers algorithm, all three of these steps would execute for each control area every iteration. This is not the case in the asynchronous implementation. Instead, the first step only executes for control areas found to be active during the current iteration; if they are inactive, the "new" $\mathbf{x}_k$ for that iteration is set to the values of the previous iteration. After that, if a control area k is found to be active or receives an updated $\mathbf{x}_l$ (where k $\neq$ l), the second and third steps execute; if there is no new information accessible by the control area, there is no reason to update its estimates of its shared state $\mathbf{s}_k$, and thus there will be no change to the dual variable $\mathbf{p}_k$. [19]

Our original plan for the implementation of the asynchronous ADMM algorithm in this thesis was to take the method that has its convergence proved in Wei's paper [9] and altering it to fit the area of power system state estimation by using elements from Kekatos and Giannakis [7]. However, in the process of matching the convergence results of the asynchronous ADMM to its equivalents in the synchronous implementation of the same system, we found that it would be easier to base the asynchronous method around the Kekatos's implementation and adding elements from Wei in order to make the synchronous algorithm lose its synchronocity. This does mean that the current implementation does not have explicit proof of $O(1/k)$ convergence as we had hoped for initially; however, since the purpose of this paper was to examine the effects of adding asynchronous elements on the convergence of the state estimates, it makes more sense to start with a synchronous algorithm with known convergence rates and seeing the impact of making it asynchronous on its rate of convergence. Looking at the methodology used in both papers, it can be seen that they both share many elements that make both implementations relevant to the problem of power system state estimation - both solve the global optimization problems outlined in the above subsection *Decentralized State Estimation of Power Systems*, both rely on converging to the saddle point of an augmented Lagrangian function to get to a state estimate for the system, and the process to reach that above saddle point is generally the same (updating the vector of states, updating the message vector, updating the dual variable vector). Unfortunately, Wei's algorithm relies on an explicit formation of the $\mathbf{H}$ and $\mathbf{D}$ matrices that is not present in Kekatos's algorithm. However, due to the nature of the power system being modeled by this implementation, the diagonality of $\mathbf{H}$ required by Wei is fulfilled by the creation of the set of $\mathbf{D}_k$ matrices in Kekatos; the effects of $\mathbf{D}$ used by Wei - that each row of the constraints have only unique elements $\mathbf{x}_i$ and $\mathbf{z}_i$ - are duplicated by the uniqueness of each element of $\mathbf{s}_k$ in Kekatos as well as the uniqueness of each bus (for $\mathbf{x}_i$) and branch (for $\mathbf{z}_i$) in a power system; and the method of activating control areas in our implementation of asynchronous variants of the algorithm proposed by Kekatos in effect implements the random variables outlined by Wei. Still, there are enough differences between Wei's definitions and what was implemented in this algorithm that it is not correct to say that it is an implementation of Wei's asynchronous ADMM algorithm in the realm of power systems - at best it can be said to be a hybridized method of Kekatos and Wei. The pseudocode for this process as well as the necessary steps in converting the data extracted from the system model into structures suitable for an asynchronous implementation of the alternating direction method of multipliers follows.

D. ADMM Pseudocode

Required information:

- K_{max} - the number of control areas k in the system being simulated.
- b - the number of buses in the system being simulated.
- $\mathcal{S}$ - the set of states of the entire power system
- $\mathcal{M}_{\text{k}}$ - the set of meters (bus and line flow) associated with control area k.
- $\mathcal{S}_{\text{k}}$ - the set of states associated with $\mathcal{M}_{\text{k}}$.
- Matrix $\mathbf{H}$ - the known Jacobian of the linearized power system being simulated.
- $\mathbf{z}$ - the vector of measurements for the system.

Parameters for the algorithm:

- c - step size constant for the augmented Lagrangian function
- itr_{max} - the maximum number of iterations for the algorithm to perform
- w_{k} - Natural numbers assigned to each control area to determine activation probability; the lower the value of a given w_{k}, the higher the chance that control area k is active every iteration

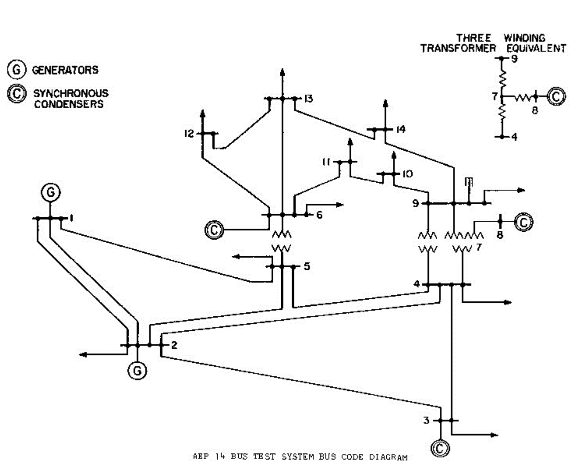
Algorithm

```
1) $\mathcal{N}_{\text{k}}^{\text{i}}$ - compose the set of control areas that share access to state i with control area k.
   • Initialize all elements of $\mathcal{N}_{\text{k}}^{\text{i}}$ to $\emptyset$
   • For control areas k=1,2,...$K_{max}$,
      ◦ For system states i=1,2,...$|\mathcal{S}|$, if i$\in\mathcal{S}_{\text{k}}$:
         ∗ For control areas j=1,2,...$K_{max}$, j$\neq$k; if i$\in\mathcal{S}_{\text{j}}$, append control area j to $\mathcal{N}_{\text{k}}^{\text{i}}$
2) $\mathbf{H}_k$ - using $\mathcal{M}_{\text{k}}$, $\mathcal{S}_{\text{k}}$ and $\mathbf{H}$, decompose the Jacobian into submatrices for each control area.
   • For control areas k=1,2,...$K_{max}$, $\mathbf{H}_k$ = $\mathbf{H}(\mathcal{M}_{\text{k}}, \mathcal{S}_{\text{k}})$
3) $\mathbf{D}_k$ - using $\mathcal{S}_{\text{k}}$ and $\mathcal{N}_{\text{k}}^{\text{i}}$, form the diagonal matrix showing how many control areas have access to
   each state in $\mathcal{S}_{\text{k}}$
   • Initialize $\mathbf{D}_k$ to a $|\mathcal{S}_{\text{k}}| \times |\mathcal{S}_{\text{k}}|$ matrix of zeros
   • For control areas k and all states i$\in\mathcal{S}_{\text{k}}$, $\mathbf{D}_k$(i,i) = $\left|\mathcal{N}_{\text{k}}^{\text{i}}\right|$
4) $\mathbf{z}_k$ - using $\mathcal{M}_{\text{k}}$, decompose the vector of measurements into vectors of local measurements accessible
   by control area k.
   • For control areas k=1,2,...$K_{max}$, $\mathbf{z}_k$ = $\mathbf{z}(\mathcal{M}_{\text{k}})$
5) Let $\mathbf{x}_k^{itr}$, $\mathbf{s}_k^{itr}$, and $\mathbf{p}_k^{itr}$ be vectors representing the values of vectors $\mathbf{x}_k$, $\mathbf{s}_k$, and $\mathbf{p}_k$ during iteration
   itr. Initialize $\mathbf{x}_k^0$, $\mathbf{s}_k^0$, and $\mathbf{p}_k^0$ to $|\mathcal{S}|$ length vectors of zeros.
For $\mathbf{itr = 1, 2... \; itr_{max}}$,
6) Initialize $\mathcal{K}$, the set of control areas receiving updated states for this iteration, to $\emptyset$
7) Randomly pick an integer $\alpha$ in the uniform discrete distribution U[1,n] where n is the least common
   multiple of all $w_{\text{k}}$
8) For each control area k=1,2,..$K_{max}$, if ($\alpha \bmod w_{\text{k}}$)=0,
   a) Update the values of $\mathbf{x}_k$ from the previous iteration
      • Let $\mathbf{p}_k^{temp}$ be a $|\mathcal{S}_{\text{k}}|$ length temporary vector of the elements in $\mathbf{p}_k^{itr-1}$ corresponding to the
        states in $\mathcal{S}_{\text{k}}$
      • $\mathbf{p}_k^{temp}$ = $\mathbf{p}_k^{itr-1}(\mathcal{S}_{\text{k}})$
      • Let temporary state vector $\mathbf{x}_k^{temp}$ = $(\mathbf{H}_k^T\mathbf{H}_k$+c$\mathbf{D}_k)^{-1}(\mathbf{H}_k^T\mathbf{z}_k$+c$\mathbf{D}_k\mathbf{p}_k^{temp})$
      • Initialize $\mathbf{x}_k^{itr}$ to a $|\mathcal{S}|$ length vector of zeros
```

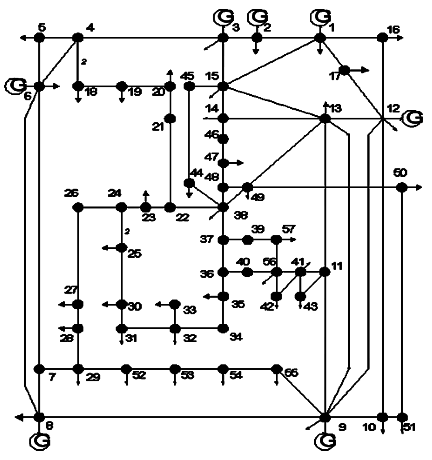
```
            • $\mathbf{x}_k^{itr}(\mathcal{S}_{\rm k}) = \mathbf{x}_k^{temp}$
        b) Updated control areas share $\mathbf{x}_k^{itr}$ with control areas that share states with the updated control area
        c) Determine the set of control areas that received state vectors this iteration
            • For all states i $\in \mathcal{S}_{\rm k}$ and all control areas j $\in \mathcal{N}_{\rm k}^{\rm i}$,
                ◦ If j $\notin \mathcal{K}$, append control area j to $\mathcal{K}$
     9) For all control areas where $(\alpha \bmod w_{\rm k}) \neq 0$, $\mathbf{x}_k^{itr} = \mathbf{x}_k^{itr-1}$
    10) For all control areas k $\in \mathcal{K}$, update $\mathbf{s}_k$ and $\mathbf{p}_k$
        a) Update the value of $\mathbf{s}_k$ from the previous iteration
            • Initialize $\mathbf{s}_k^{itr}$ to a $|\mathcal{S}|$ length vector of zeros
            • For all i $\in \mathcal{S}_{\rm k}$ such that $\mathcal{N}_{\rm k}^{\rm i} \neq \emptyset$, $\mathbf{s}_k^{itr}({\rm i}) = \frac{1}{\left|\mathcal{N}_{\rm k}^{\rm i}\right|} \sum_{j \in \mathcal{N}_{\rm k}^{\rm i}} \mathbf{x}_j^{itr}({\rm i})$
        b) Update the value of $\mathbf{p}_k$ from the previous iteration
            • Initialize $\mathbf{p}_k^{itr}$ to a $|\mathcal{S}|$ length vector of zeros
            • For all i $\in \mathcal{S}_{\rm k}$ such that $\mathcal{N}_{\rm k}^{\rm i} \neq \emptyset$, $\mathbf{p}_k^{itr}({\rm i}) = \mathbf{p}_k^{itr-1}({\rm i}) + \mathbf{s}_k^{itr}({\rm i}) - \frac{\mathbf{x}_k^{itr-1}(i) + \mathbf{s}_k^{itr-1}(i)}{2}$
    11) For all control areas k $\notin \mathcal{K}$, $\mathbf{s}_k^{itr} = \mathbf{s}_k^{itr-1}$ and $\mathbf{p}_k^{itr} = \mathbf{p}_k^{itr-1}$
End for loop
    12) Return the set of vectors of $\mathbf{x}_k^{itr\max}$, k=1,2,..$K_{max}$
```

IV. Simulation Setup

A. *Initial work*


GENERATORS
SYNCHRONOUS CONDENSERS
THREE WINDING TRANSFORMER EQUIVALENT
AEP 14 BUS TEST SYSTEM BUS CODE DIAGRAM

(a) Diagram of the 14-bus IEEE test system.



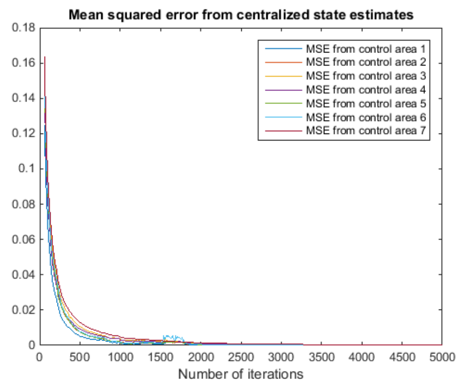
(b) The one-line diagram of the IEEE 57 bus "Midwest" test system

To begin with the investigation of asynchronous alternating direction method of multipliers for power systems, I began with the 14 bus, 4 control area IEEE system available from University of Washington's Power Systems Test Case Archive. [20] To begin with, I started with a previously built synchronous implementation of the alternating direction method of multipliers as outlined in Kekatos and attempted to modify it so it updated asynchronously rather than synchronously. As mentioned in the *Synchronous vs. Asynchronous ADMM* section above, the first attempt at the implementation of the algorithm was event-driven as described in [6] rather than the probability based model outlined in the pseudocode above. When I discovered the event-driven implementation's issues of devolving into a synchronous ADMM implementation, I adjusted the secondary updating parameter of number of updateless iterations, but had no luck in creating a reliable algorithm for the 14-bus system, much less the 57-bus system used in the final simulations. Instead, I turned to a prototype of the probability method outlined above that used the weighting constants w_k = [2 2 3 4]. Once I found a prototype that seemed to converge in the 14-bus case, I decided to shift the algorithm to a 57-bus system. Though the 57 bus system is still relatively small compared to any real life power system, it was still reasonably large enough to show that the ADMM algorithm would work on infrastructure not identical to the 14-bus case, and would give us insight on how its performance scaled with larger power systems.

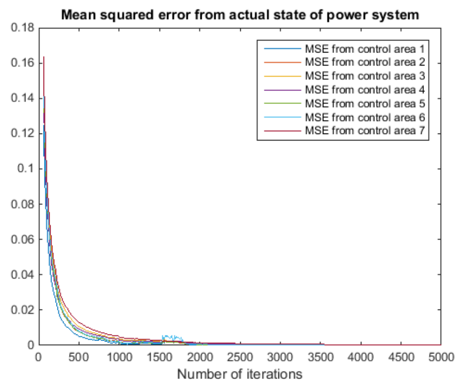
B. *IEEE 57 bus system*

For this thesis, we used the IEEE 57 bus "Midwest" power system available at [20]. Using the process outlined in section IV, substantial hints from the associated file for the 14-bus system, and the power systems state estimation chapter [1] that Professor Kim gave me, I set up a file from the IEEE data file format for the 57 bus test system that created its Jacobian matrix, formed real state measurements that we will be treating as a perfect reference, made noised state measurements that the algorithm will be acting upon, and decomposed the set of buses that are controlled by each control area as recommended in [22]. w_k was directly set to [8 4 5 1 3 2 2], giving us the range of [1,120] for α. Iterations for the asynchronous ADMM algorithm were capped at 5000; the algorithm was going to automatically terminate after convergence criterion for the system was met for all control areas within an iteration, but that was removed during the change from the event-driven implementation to the probability-based implementation. The signal to noise ratio for the power system was set to 6000, and the standard deviation of the noise was set to 0.001; using this, the set of true states of the buses of the system was created using the meter state measurements inherent in the data file with added variance, and the noised set of measurements **z** were created from those true states using the noise calculated from the SNR and the standard deviation set previously.

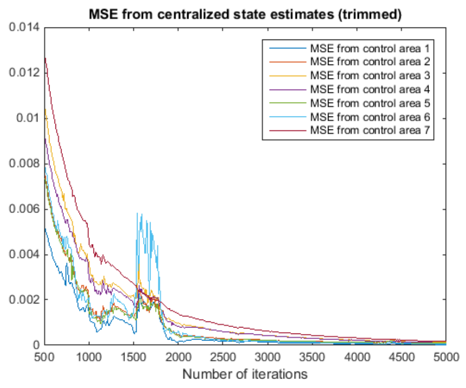
V. Results



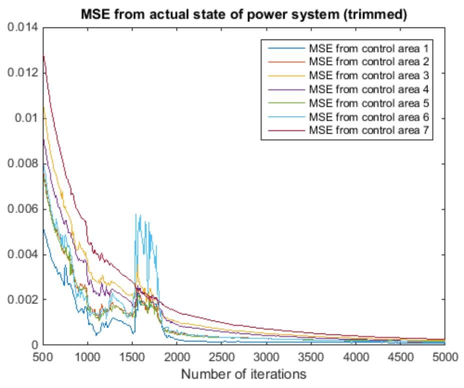
(a) The mean squared error of the asynchronous ADMM algorithm with the centralized state estimation as reference.



(b) The mean squared error of the asynchronous ADMM algorithm with the real states of the power system as reference.



(a) Mean squared error of the above after iteration 500.



(b) Mean squared error of the above after iteration 500.

A. *Results from asynchronous ADMM simulations from centralized state estimator*

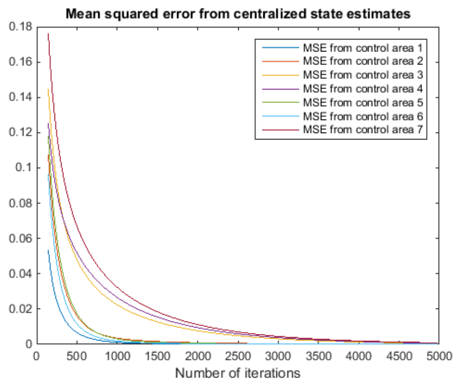
The above figures show the mean squared error of each bus in control area k from the centralized state estimate, MSE_{k}. This was calculated as MSE_{k}(itr) = $\sqrt{\sum_{j \in \mathcal{S}_{\text{k}}} (\mathbf{x}_c(j) - \mathbf{x}_k^{itr}(j))^2}$, where $\mathbf{x}_c(j)$ is the centralized estimate of state j and $\mathbf{x}_k^{itr}(j)$ is the estimate of state j from control area k in the asynchronous ADMM algorithm at iteration itr. After the values of MSE_{k} have been calculated for the simulation, plot MSE_{k} for k=1,2,...K$_{max}$. For these simulations, the resolution of the graph was set to 10 iterations in order to reduce the computational time needed for a single run of the simulations to complete; with a computationally stronger computer system, a simulation with higher resolution results could be made. Additionally, due to the high amount of error in the first iterations from setting the initial conditions $\mathbf{x}_k^0$ to a $|\mathcal{S}|$ length vector of zeros, a trimmed version of the plot for MSE values after iteration 500 are also provided.

B. *Results from asynchronous ADMM simulations from denoised state measurements*

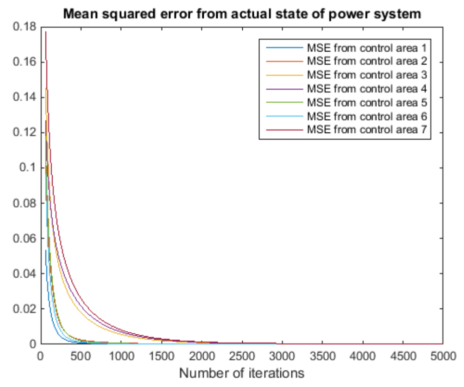
The above figures show the mean squared error of each bus in control area k from the centralized state estimate, MSE_{k}. This was calculated as MSE_{k}(itr) = $\sqrt{\sum_{j \in \mathcal{S}_{\text{k}}} (\mathbf{x}_r(j) - \mathbf{x}_k^{itr}(j))^2}$, where $\mathbf{x}_r(j)$ is the real state of the system, generated from system information, j and $\mathbf{x}_k^{itr}(j)$ is the estimate of state j from control area k in the asynchronous ADMM algorithm at iteration itr. After the values of MSE_{k} have been calculated for the simulation, plot MSE_{k} for k=1,2,...K$_{max}$. Like the above, the resolution of the graph was set so that the results of only every tenth iteration were saved and used in error calculations. Similar to the centralized state estimator, from the large amount of error in the first iterations from setting the initial conditions $\mathbf{x}_k^0$ to a $|\mathcal{S}|$ length vector of zeros we have provided a trimmed version of the plot for MSE values after iteration 500.

C. *Results from Equivalent Synchronous ADMM simulations*

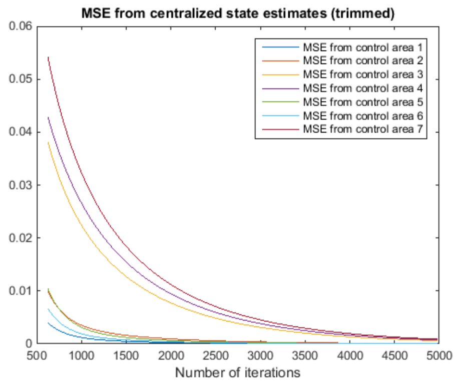
For comparison, we recreated the synchronous alternating direction of multipliers implementation for power systems that we initially tested the 57 bus test system with; in this case, we set w_{k} to [1 1 1 1 1 1 1]. However, this synchronous ADMM was not a equivalent match to the asynchronous algorithm; rather than being bottlenecked by the probability that the slowest updating state within the control area, it was assumed that each control area would update at every iteration. In order to fix this mismatch between models, we added a mock penalty factor that dilated the axis of iterations by the average activation chance over all control areas k in the asynchronous model. That is, with asynchronous w_{k} set to [8 4 5 1 3 2 2], we found a weighting factor of $(\frac{\frac{1}{8}+\frac{1}{4}+\frac{1}{5}+1+\frac{1}{3}+\frac{1}{2}+\frac{1}{2}}{7})^{-1} = \frac{840}{349}$ that was used to scale the iterations of the synchronous model. The scaled iterative axis was then stripped to 5000 iterations, same as the asynchronous model, with another graph trimmed to exclude the first 500 iterations in order to emphasize the final values of the simulations.



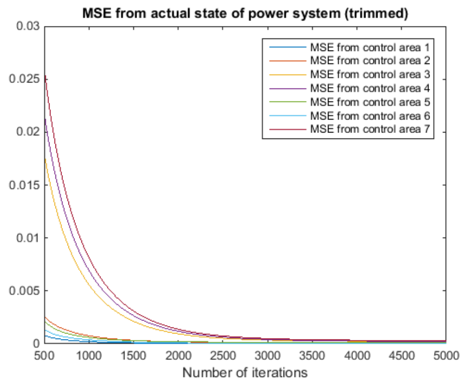
(a) The mean squared error of the synchronous ADMM algorithm with the centralized state estimation as reference.



(b) The mean squared error of the synchronous ADMM algorithm with the real states of the power system as reference.



(a) Mean squared error of the above after iteration 500.



(b) Mean squared error of the above after iteration 500.

VI. Conclusions and Remarks

With the above graphs, we can see that the asynchronous alternating direction method of multipliers for power systems state estimation eventually converges to a centralized state estimate of the same system, similar to the synchronous implementation. One thing to note in the above graphs is the large spike in mean squared error in the asynchronous algorithm starting around iteration 1500. This was likely caused control area 6 (the teal line) being inactive for many iterations while its neighbors have sent it new state estimates; in this situation, the previously comatose control area updated its estimates in one large batch and overshot the actual value of the system estimator there considerably. However, it is comforting to note that this jump in error disappears within a few hundred iterations and that its corrections occurs faster than it took for the mean squared error of the control area to initially drop from the peak of the spike to the values at its base. Additionally, barring these spikes in error, the trend for MSE values is downward for the large majority of the algorithm, including the last thousands of iterations. This implies that the current implementation of the asynchronous alternating method of multipliers algorithm will eventually converge to a desired level of accuracy of the centralized state estimate and actual state of the linearized, DC system.

Another point of interest in the above graphs is the fact that the asynchronous implementation of the ADMM algorithm converges slightly faster than the equivalent scaled synchronous model. This makes some sense, as when we descale the synchronous model to see how many iterations actually occurred during the simulation, we see that only around 2000 true iterations of the synchronous model were equivalent to the 5000 of the asynchronous simulation. Even removing the spike of error that is present during iteration 2000 of the asynchronous model, we can see that the error at that iteration is much greater than that of the error of the same model at iteration 5000. However, we can also see that the error at iteration 2000 for the asynchronous implementation is also greater than the error at iteration 2000 for the synchronous implementation. This implies that, per iteration, the asynchronous model on average converges slower than the synchronous model; the overall asynchronous model only converges faster because many more asynchronous iterations occur in the same amount of real life time than synchronous updates, as expected.

Additionally, an interesting thing to note is that the synchronous algorithm does not have the large spikes in error that the asynchronous result has. This is likely a side-effect of forcing every control area to update in lock-step; as postulated in the first paragraph of this section, it could be that the spike in error only occurs if neighboring control areas update lopsidedly where one side is inactive for several of its neighbor update cycles. In this case, we could introduce an element of the event-driven asynchronous algorithm and force a control area to update if it has received a certain number of updates from its neighbors without updating itself; however, this hybrid method is outside the scope of this thesis at this point. Another possible explanation is that the synchronous algorithm is still vulnerable to these spikes in error but few enough simulations of the algorithm were made and the vulnerability was not exposed. This was the case for the asynchronous ADMM implementation for some time; only occasionally would the simulation return a result where the mean squared error of a control area increased over multiple iterations. However, if this is the case that heavily implies that there is an undiscovered error in my model of the IEEE 57 bus system. In any case, future study should be invested to see if this quirk is an issue inherent to the pure probability based implementation of the asynchronous alternating direction method of multipliers for power systems state estimation, but that is also outside of the scope of this thesis.

VII. Areas for Future Work

As mentioned previously, the code used in this implementation of the asynchronous ADMM is a method that implements aspects of Wei's asynchronous methodology and Kekatos's implementation of robust power system state estimation, but falls short of applying his algorithm to the problem. Thus, an immediate direction for future work would be to directly implement Wei's pseudocode for asynchronous ADMM into a power systems state estimation setting and comparing any differences between the direct implementation and this hybridized method, with great detail spent on how to implement the matrices $\mathbf{H}$ and $\mathbf{D}$ [9] that were not explicitly present in framework of Kekatos. If it turns out that this new implementation integrates the $\mathbf{H}$ and $\mathbf{D}$ matrices into the power systems framework as we hypothesized above, this current implementation of asynchronous ADMM on power system state estimation can be mathematically proven to converge to the centralized state estimate; if not, then an asynchronous implementation proven to converge at the $O(1/k)$ will still have been created and is thus usable as a base for other future work.

Additionally, this thesis only scratches the surface of power system state estimation using an asynchronous implementation of the alternating direction method of multipliers. The next step of inquiry would to be examining how to mitigate the spikes of error in each control area's state estimate as seen in figures 3 through 6. A good start would be to see if the hybrid asynchronous implementation would remove this vulnerability, as well as seeing if the related synchronous implementation is indeed vulnerable to it as well. Additionally, this thesis only looked at the implementation of the algorithm on a linearized, DC power system; for this algorithm to be used effectively in the real world, the effects of nonlinearities of the power system would have to be integrated into the algorithm, substantially changing the composition of the Jacobian matrix $\mathbf{H}$, amongst other elements of the decomposition of the system into control areas as well as the initial creation of the system as a whole. Additionally, requiring the power system state estimation to be done in an AC setting means that the asynchronous implementation of the alternating direction method of multipliers has to converge to a given of accuracy in a specific amount of time or iterations instead of the unbounded eventual convergence we saw above; this is even an issue with the synchronous implementation of ADMM, and is the reason why we wished to see if the asynchronous algorithm would converge faster. Some of these problems are mentioned in [23], but no pseudocode was written nor convergence results were established. In any case, there is still plenty of investigation to be done before asynchronous implementations of the alternating direction methods of multipliers as a field can be accepted or discarded as a possible improvement on power systems state estimation.

References

[1] A. Abur and A. Gomez-Exposito. *Power System State Estimation: Theory and Implementation*. New York: Marcel Dekker, 2004. Chapter 2.

[2] Kun Zhu. Class Lecture, Topic: "Power System State Estimation". Presented at KTH Royal Institute of Technology, May 2nd, 2013. Accessed Jan 22nd, 2016. Available online: https://www.kth.se/social/upload/518a08d3f27654786295ca51/Lecture_15_StateEstimation.pdf.

[3] Vadim Utkin, Jrgen Guldner, and Jingxin Shi. "Sliding Mode Control in Electromechanical Systems". 1999. Philadelphia, PA - Taylor and Francis, Inc. ISBN 0-7484-0116-4

[4] Abur, Chapter 5.

[5] Raffi Sevlian and Umnoy Ponsukcharoen. *Distributed Power System State Estimation*. Internet: http://web.stanford.edu/~rsevlian/raffi_sevlian_papers/CEE272R.pdf. June 17, 2012 [January 14, 2016].

[6] Jinling Liang and Zidong Wang, *Distributed State Estimation for Discrete-Time Sensor Networks with Randomly Varying Nonlinearities and Missing Measurements*. Internet: http://core.ac.uk/download/files/14/337359.pdf. March 3, 2011 [January 12, 2016].

[7] Vassilis Kekatos and Georgios B. Giannakis. *Distributed Robust Power System State Estimation*. Internet: http://arxiv.org/abs/1204.0991. June 30, 2012 [January 6, 2016].

[8] Alexey Matveev and Andrey Savkin, *The Problem of State Estimation via Asynchronous Communication Channels With Irregular Transmission Times*. Internet: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1193753. April 4, 2003 [January 10, 2016].

[9] Ermin Wei and Asuman Ozdaglar. *On the $O(1/k)$ Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers*. Internet: http://arxiv.org/abs/1307.8254. July 31, 2013 [January 6, 2016].

[10] Mahendra Mallick, Stefano Coraluppi, Craig Carthel. *Advances in Asynchronous and Decentralized Estimation*. Internet: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=931505. 2001 [January 20, 2016].

[11] Oliver Kosut, Liyan Jia, Robert J. Thomas. *On Malicious Data Attacks on Power System State Estimation*. Internet: http://acsp.ece.cornell.edu/papers/KosutJiaThomasTong10UPEC.pdf. September 3, 2010 [Feburary 6th, 2016].

[12] Stephen Boyd, Neal Parikh, et al. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Internet: https://web.stanford.edu/~boyd/papers/pdf/admm_distr_stats.pdf. November 2010 [January 4, 2016].

[13] Stephen Boyd, Neal Parikh, et al. Pages 15-18.

[14] Stephen Boyd, Neal Parikh, et al. Pages 106-110.

[15] Minyi Zhong and Christos Cassandras. *Asynchronous Distributed Optimization With Event-Driven Communication*. IEEE Transactions on Automatic Control, Volume 55, No 12. Published December 2010.

[16] Ali T. Alouani, John E. Gray, Dennis H. McCabe. *Theory of Distributed Estimation using Multiple Asynchronous Sensors*. IEEE Transactions on Power Systems, Volume 41, No. 2. Published April 2005.

[17] Mikael M. Nordman and Matti Lehtonen. *Distributed Agent-Based State Estimation for Electrical Distribution Networks*. IEEE Transactions on Power Systems, Volume 20, No. 2. Published May 2005.

[18] *Power Systems Test Case Archive*. Internet: https://www.ee.washington.edu/research/pstca/formats/cdf.txt. August 1988 [November 11, 2015].

[19] D. M. Falcao, Felix Wu, and Liam Murphy. *Parallel and Distributed State Estimation*. IEEE Transactions on Power Systems, Volume 10, No. 2. Internet: http://hub.hku.hk/handle/10722/42728. May 2, 1995 [January 22, 2016].

[20] *Power Systems Test Case Archive*. Internet: https://www.ee.washington.edu/research/pstca/formats/cdf.txt. August 1988 [November 10, 2015].

[21] Vassilis Kekatos and Georgios B. Giannakis. Pages 3-5.

[22] Pooja Sharma a Navdeep Batish. *Evaluation of IEEE 57 Bus System for Flow Analysis*. International Journal of Engineering Research and Applications. Volume 5, Issue 8. Published August 2015.

[23] Tsung-Hui Chang, Mingyi Hong, Wei-Cheng Liao, Xiangfeng Wang. *Asynchronous Distributed ADMM for Large-Scale Optimization*. Published September 9th, 2015.