EVENT DETECTION AND VISUALIZATION BASED ON PHASOR MEASUREMENT UNITS FOR IMPROVED SITUATIONAL AWARENESS

BY

JOSEPH EUZEBE TATE

B.S., Louisiana Tech University, 2003
M.S., University of Illinois at Urbana-Champaign, 2005

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Professor Thomas J. Overbye, Chair
Professor Peter W. Sauer
Professor David M. Nicol
Assistant Professor Alejandro D. Domínguez-García
James D. Weber, PowerWorld Corporation

ABSTRACT

Knowledge of device statuses on the power grid is a crucial component of situational awareness, and the lack of this knowledge has caused catastrophic failures in the past. Existing techniques for status identification, such as topology processing and state estimation, rely almost exclusively on local area measurements without time stamps or synchronization, and can be significantly improved by leveraging PMU data which is available over wider areas at much faster data rates. The processing needed to extract steady-state angle information from PMUs is the first topic of discussion, and the advantages and disadvantages of two types of digital filters—finite impulse response and median—are investigated. The usefulness of each filtering method is demonstrated with a broad spectrum of signals, both real and simulated. The proposed method of system event identification is then presented in a general form, with specific algorithms defined for detecting single-line, double-line, and generator outages. Test results for each of these event classes are provided to demonstrate the efficacy of the proposed event detection algorithms.

In addition to the new methods presented for the processing of PMU data, techniques for visualization of PMU data in its processed and unprocessed forms are described. A new technique based on graphical processing units (GPUs), developed to allow rendering speeds to match the relatively fast data rate of PMUs, is presented, along with results demonstrating the marked increase in visualization speed. Some of the benefits of speeding up contouring are discussed, including how PMU derivative information can be visualized with GPU-based rendering.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Motivation

With the increasing load on the power system, along with the massive interarea transfers enabled by the deregulation of the 1980s and 1990s, there is a clear need to have reliable information about both the local system and external systems. Tellingly, four of the six major North American blackouts were due in part to a lack of situational awareness [1]. Although there is a clear need for sharing of information, there is limited real-time sharing of SCADA or state estimator information in the United States [2]. However, as phasor measurement units (PMUs) [3] have been deployed throughout the North American power grid, there have been significant efforts to ensure that PMU data is shared between all interested parties [4]. Because PMU data is more widely available in near real-time than other power system measurement data, it can provide unique insights into the global operation of the grid. However, new techniques must be developed to apply the data that PMUs provide in a useful manner.

Extensive research in applying PMU information to improve situational awareness has been conducted since their introduction, including applications in state estimation [5-7], dynamic security assessment [8-10], and visualization [11-13]. Yet another key aspect of situational awareness in the power grid is the knowledge of transmission line, transformer, and generator statuses. Beyond incorporation of PMUs into traditional state estimation, which can include topology estimation [14], there has been little research into how PMUs can be used to enhance topology

information, particularly outside of the local area. Current network topology processors focus on obtaining local system topology information through the use of predominantly local area measurements. However, connectivity information over the wide area is very important for system operations and is a major reason for the existence of the NERC System Data Exchange (SDX) [1]. Because there is significant value in improving system operators' knowledge of external system line outages, a method which utilizes PMU data to effect this improvement has been developed.

The method discussed in this work makes use of SDX information, as it is currently the best source of systemwide line status information. One of the most telling indications of its usefulness is its application in processing transmission loading reliefs (TLRs) on the North American power grid. However, despite increased awareness of the importance of interarea information exchange since the August 2003 blackout, updates to the SDX are still only required on an hourly basis [15]. Because much can happen on the power grid within an hour, there is a need for tools which are capable of providing more current information on external system outages. This work looks at ways to improve upon the status information from local topology processors and the NERC SDX by incorporating PMU data which, like SDX information, is also available over the entire interconnect through phasor data concentrators (PDCs) [4]. Although the methods presented here would work with any source of synchronized angle data, PMUs are the only devices which are deployed

over the entire power grid and are capable of providing geographically dispersed, synchronized, and accurate phasor angle measurements.

The processing needed to extract steady-state angle information from PMUs is the first topic of discussion, and the advantages and disadvantages of two types of digital filters—finite impulse response and median—are investigated. The usefulness of each filtering method is demonstrated with a broad spectrum of signals, both real and simulated, which are meant to be representative of typical power system signals. This process is necessary to convert the raw angle measurements into a form which can be used with traditional steady-state analysis tools such as the power flow. The intended application of this filtering, system event identification is then presented in a general form, with specific algorithms defined for detecting single-line, double-line, and generator outages. Test results for each of these event classes are provided to demonstrate the efficacy of the proposed event detection algorithms using both real and simulated data. Several optimization problems are also defined to guide placement of PMUs for event detection, and results are presented based on usage of exhaustive search and simulated annealing to perform the placement optimization.

In addition to the new methods presented for the processing of PMU data, techniques for visualization of PMU data in its processed and unprocessed forms are described. A new technique based on graphical processing units (GPUs), developed to allow rendering speeds to match the relatively fast data rate of PMUs, is presented, along with results demonstrating the marked increase in visualization speed. Some of

the benefits of speeding up contouring are discussed, including how PMU derivative information can be visualized with GPU-based rendering.

1.2 Literature Review

The basic operations needed for PMUs to work, including relevant filtering and signal processing techniques, were first described in 1983 [16] and the first prototype PMU was developed in 1988 [17]. Shortly after these new measurement devices were described, PMU applications related to situational awareness began to be researched. Initial applications using PMUs were primarily focused on improvements to state estimation based on the ability to directly measure system states [7]. A key aspect of this initial research into PMU usage was the need for traditional SCADA measurements (such as breaker status, voltage magnitude, and current magnitude) to accompany PMU measurements in order to use the new data. The problem with this dependence on SCADA measurements is that, outside of the local control area, SCADA measurements are usually unavailable. One way in which researchers have attempted to get around this dependence on SCADA measurements has been to determine how complete coverage of the power system could be obtained using only PMU measurements. The key paper on this topic demonstrates that complete observability could be obtained with as few as one fourth to one third of the system buses having PMUs [18]. Unfortunately, current coverage of PMUs is still well below this number (e.g., for the Eastern Interconnect, only 60 PMUs are currently online [19]). Additional work in limited placement of PMUs for incomplete observability fails to provide applications in which this partial observability would be

useful beyond incorporation into traditional state estimators [20]. Besides applications in state estimation, additional research has been performed in the usage of PMUs for system monitoring. Monitoring of power system harmonics based on PMU data is discussed in [21] and [22], although there is limited discussion of how PMU location and filtering can affect the applicability of this monitoring task for improving operator awareness. Fault detection and classification based on highly localized PMU measurements has also been researched. Reference [23] provides an overview of the different methods proposed in this area and notes that all of the currently developed methods for fault detection require PMUs to be located at one or more of the terminal buses of the faulted transmission line. Finally, direct visualizations of phasor angles for system monitoring have been developed [10-12], although there is little guidance in how the visualized angles should be interpreted by system operators.

The notion that PMU measurements can be applied in more problem domains, particularly event detection and classification, is supported by the literature. The close correspondence between PMU angle measurements and correct steady-state phasor angles on the system has been demonstrated in numerous studies [24-26]. In addition, the IEEE standard governing PMU accuracy requires that the total vectorized error (TVE), defined as the normalized Euclidean distance between the true and measured steady-state phasors, be within 1% [27]. Although it cannot be assumed that SCADA and state estimator data is available over the entire interconnect, some additional wide area information, such as that available through

the NERC System Data Exchange (SDX) [28], can also be used for interpretation of PMU data due to the system-wide availability of this information. Usage of PMU measurements for comprehensive dynamic information is less promising, mainly due to the lack of a unifying standard on PMU behavior during transient conditions.

Event detection and classification can be divided into three key components: detecting when an event has occurred, extracting salient information about the event from the raw data, and classifying the event based on this information. For the first two tasks, detection and information extraction, the largest body of research is in the area of image processing due to the need for accurate edge detectors in many applications. Edge detectors are essentially step change detectors and can be used to return not only step change locations but also the amplitude of step changes. An excellent survey of classic edge detection techniques is provided in [29]. A widely used "hill climbing" method used to accurately detect edges which involve ramp rather than step changes is detailed in [30] and serves as the basis for the edge detection techniques described later in the dissertation. Moving beyond linear filters, nonlinear filtering for edge detection, in particular median filtering, has been shown to outperform linear filtering for certain image processing [31] and power electronics applications [32]. However, there are two key issues which make nonlinear filters less appealing—the potentially higher computational burden relative to linear filters, and the inability to determine the filter's performance a priori. One commonality among all edge detectors surveyed is the domain-specific length of the filter used. In all cases, analysis of representative data must be conducted to determine the most

appropriate filter length for a given application [30]. In addition to the image processing literature, there has also been extensive research into event detection, feature extraction, and event classification in power systems, primarily centered on fault detection and classification. Modern methods of solving this problem rely on techniques from the field of artificial intelligence (AI), such as neural networks [33] and expert systems [34]. There has also been significant work in detecting power quality disturbances using various data processing techniques, including wavelet analysis, expert systems, neural networks, and genetic algorithms [35]. Although these methods could be used for detecting changes and extracting information about the changes in system states, adapting these techniques to wide area event classification is not feasible due to the unavailability of quality training data. In the area of large-scale, steady-state event detection, the primary tools currently used are topology processing and state estimation, relying almost exclusively on traditional power system data [36]. One notable exception is [37], which provides a method for direct determination of external system outages based on expected correlations between changes in internal system states and changes in boundary flows.

Power system visualization was first discussed in the context of visualizing load flow solutions using the IBM System/360 mainframe computer [38]. Additional work in visualization techniques continued throughout the 1970s, 80s, and 90s; reference [39] provides a concise overview of state-of-the-art visualization techniques developed up to the mid-1990s. There has since been substantial research and deployment of system visualizations throughout control centers with a focus on

increasing situational awareness of grid operators [40, 41]. In addition, formal evaluation has shown the potential benefits of several visualization techniques such as voltage contouring and network flow animation [42, 43]. One key similarity among all currently used visualization methods is the reliance on the CPU to perform any necessary calculations. This has been sufficient in the past because rendering times have only had to keep up with the relatively slow rates of SCADA and state estimator data, typically on the order of seconds or minutes. Because PMU measurements provide data at rates which are an order of magnitude faster, CPU-based techniques will be difficult to use if real-time visualization is desired. Taking as one example voltage contouring, one of the fastest techniques currently used for CPU-based voltage contouring requires 0.63 s to render a contour of a medium-sized system [44]; however, real-time visualization of PMU data, which has a data rate of 30 frames per second (fps), would require rendering times to be shorter than 0.033 (1/30) s. It has been shown in numerous applications that one way of achieving significant visualization speedups is through the usage of programmable GPUs. The promise of improved graphical performance through the utilization of GPUs is based on the massive computation power of modern GPUs—for instance, some of the latest GPUs such as the GeForce GTX 280 have theoretical processing speeds in excess of 900 billion floating point operations per second (Gflops) which is much higher than modern CPUs which are only capable of around 30 Gflops [45]. References [46] and [47] provide a broad survey of applications in which GPUs have provided significant acceleration of visualization techniques. The use of GPUs to speed up scattered data

interpolation, the more formal name for the contouring techniques used in power systems, of three- and four-dimensional data is presented in [48]; one of the key results is that interpolating data on a 128^3 grid, with 350 000 data points, requires only 0.328 s. In addition, it was found that using GPU-based data interpolation can facilitate real-time updating of data-driven displays at frames rates of approximately 10 fps. Using power system visualizations to illustrate data trends, as suggested in [12], would also require significant reductions in rendering times to obtain frame rates which are still perceived as responsive [49].

1.3 Dissertation Overview

PMUs provide detailed angle information that is sampled simultaneously at each measured bus. This capability allows for extremely fast detection of changes in system angles, which in turn can provide information about events occurring on the system. Power system events can be modeled as a change in power injection at the system buses, which can in turn be related to changes in angles through the power flow equations. Figure 1.1 illustrates how an event on the system is mapped into a change in angles through a combination of event modeling and application of the power flow equations.

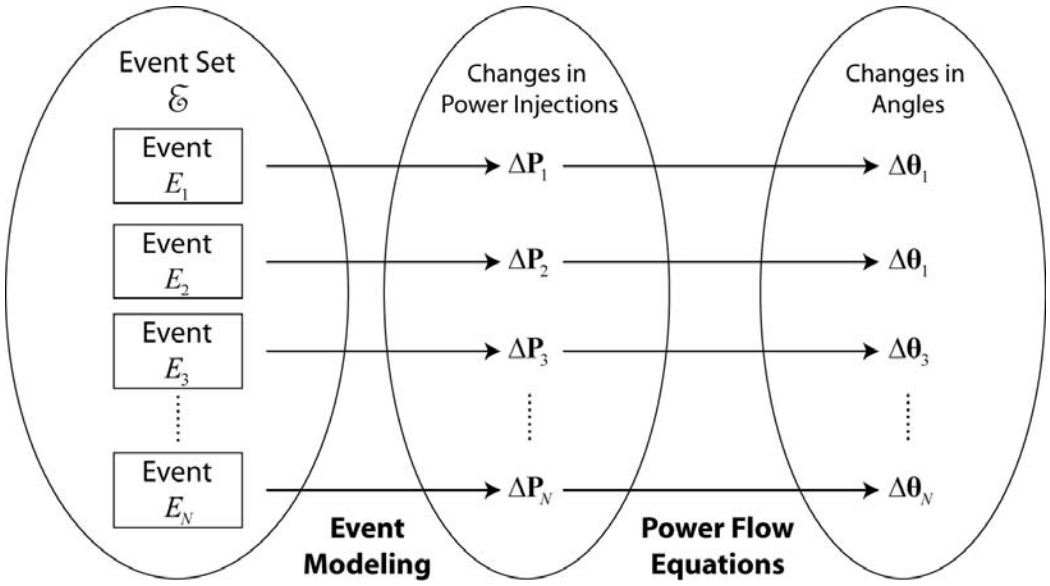

Figure 1.1: Determining the angle changes on the system resulting from an event.

For each event E from the event set $\mathfrak{E}$, a vector $\mathbf{\Delta P}$ is constructed which models the event as a set of changes in power injections on the system. The construction of this $\mathbf{\Delta P}$ vector varies based on the type of event being considered and is discussed in more detail below within the sections dealing with each event type. Once the $\mathbf{\Delta P}$ vector is determined, the power flow equations are then used to determine the associated changes in angles on the system, $\mathbf{\Delta\theta}$. Either the ac or dc power flow equations can be used to perform the mapping from $\mathbf{\Delta P}$ to $\mathbf{\Delta\theta}$, and the choice between the two is generally a tradeoff between accuracy and computation time [50]. From a performance standpoint, the ac power flow solution is slower than the dc power flow solution but provides higher solution accuracy. In addition, the ac power flow equations require the full state of the system to be known. On the other hand, utilization of the dc power flow equations only requires knowledge of the system topology and line parameters. Because complete state information is not available

over the entire interconnect, the dc power flow equations are used in this work. One form of the dc power flow equations is [50]

$$\begin{aligned} \mathbf{\Delta\theta} &= \mathbf{B}^{-1}\mathbf{\Delta P} \\ B_{ik} &= -\frac{1}{X_{ik}}, k \neq i \\ B_{ii} &= \sum_{i\omega k} \frac{1}{X_{ik}} \end{aligned} \tag{1.1}$$

where i ω k indicates the existence of a branch between the two buses i and k, and X_{ik} is the reactance of the branch connecting buses i and k.

Event detection can be considered the inverse of the procedure shown in Figure 1.1, i.e., determination of the event which resulted in an observed set of angle changes on the system. A formal definition of the problem is

$$\begin{aligned} E^* &= \arg\min_{E\in\mathfrak{E}} \left\| \Delta\mathbf{\theta}_{observed} - \mathbf{f}(E) \right\| \\ \Delta\mathbf{\theta}_{observed} &\in \mathbb{R}^K \\ \mathbf{f} &: \mathfrak{E} \rightarrow \mathbb{R}^K \end{aligned} \tag{1.2}$$

where K is the number of PMU-observable buses on the system. The mapping $\mathbf{f}$ is represented in Figure 1.1 as the transformation from an event E to a change in angles $\Delta\theta$.

Because $\left\| \Delta\mathbf{\theta}_{observed} - \mathbf{f}(E) \right\|$ in (1.2) is calculated independently for each event, it is possible to separate the event set into C different classes of events and minimize over each individual class of events:

$$\begin{aligned} E_1^* &= \arg\min_{E\in\mathfrak{E}_1} \left\| \Delta\mathbf{\theta}_{observed} - \mathbf{f}(E) \right\| \\ E_2^* &= \arg\min_{E\in\mathfrak{E}_2} \left\| \Delta\mathbf{\theta}_{observed} - \mathbf{f}(E) \right\| \\ &\vdots \\ E_C^* &= \arg\min_{E\in\mathfrak{E}_C} \left\| \Delta\mathbf{\theta}_{observed} - \mathbf{f}(E) \right\| \end{aligned} \tag{1.3}$$

The event detection algorithm of (1.2) can then be broken down into the following steps:

1. Determine $\Delta\boldsymbol{\theta}_{observed}$
2. For each class of events c
    a. Find the minimizing event E_c^* according to (1.3)

Chapter 2 and Section 3.1 provide details on how step 1 is performed. The majority of Chapter 3 is devoted to describing in detail how step 2.a is performed for single-line outage, generator outage, and double-line outage event classes. Chapter 4 provides results and analysis for outages on a 37-bus study system (using simulated data) and the Tennessee Valley Authority system (using real PMU measurements). Chapter 5 looks at several different objective functions and optimization methods which can be used to place PMUs for event detection. Chapter 6 describes a new GPU-based contouring method along with the advantages in terms of usability and applications. Finally, Chapter 7 presents conclusions based on the remainder of the dissertation, along with several potential avenues of future research.

2 FILTERING OF PMU MEASUREMENTS TO EXTRACT STEADY STATE VALUES

In order to evaluate the possibility of an event having occurred on the system, it is first necessary to determine the quasi-steady-state changes in measured phasor angles, $\Delta\boldsymbol{\theta}_{observed}$. The phasor angle measurements at bus *i* are referred to as $\theta_i[n]$, where *n* is the *n*th sample of the phasor angle. Because only the quasi-steady-state angle values are of interest, rather than the total dynamic response, filtering must be applied to the signal. The key objectives for the filtering process are:

2.C1. Eliminate noise and oscillations.

2.C2. Maintain step changes due to switching of devices.

2.C3. Minimize the amount of delay between an event occurring and the extraction of the angle changes on the system.

Two filtering methods are evaluated for the purposes of event detection—finite impulse response (FIR) linear filtering and median filtering. Each of these has its own advantages and disadvantages which are discussed in detail below. In addition, a method of quantitatively evaluating filter performance is presented, and the different filtering methods are evaluating for a range of real and simulated PMU signals.

2.1 FIR Filtering of Measurements

FIR filtering is one of the most common types of digital signal processing used, due primarily to its stability and linear phase characteristics. In addition, this type of filter has shown promise in other applications related to PMU signal processing [51-54]. A general *N*-order FIR filter takes the following form:

$$x_{filt}^{FIR}[n] = \sum_{i=0}^{N} \alpha_i x[n-i] \tag{2.1}$$

FIR filtering was chosen rather than infinite impulse response (IIR) filtering, the other common linear filtering method, for several reasons. First, FIR filters are guaranteed to be BIBO stable because there is no feedback in the system. This can be an important feature for this application, particularly if measurement noise cannot be bounded ahead of time. For example, impulse noise has been observed in real PMU data, and BIBO stability is necessary to ensure that impulse noise does not have a disastrous effect on the output signal. A second desirable quality of FIR filters is their linear phase characteristic. Maintaining edge information is crucial in detecting where angle changes have occurred, along with the magnitude of the angle changes. Considering the Fourier transform of a unit step function [55],

$$\frac{1}{1-e^{-j\omega}} + \sum_{k=-\infty}^{\infty} \pi\delta(\omega + 2\pi k) \tag{2.2}$$

nonuniform displacement of different frequency components could lead to difficulties in detecting angle changes and properly quantifying the magnitude of the changes.

FIR filters do suffer from some shortcomings relative to IIR filters. FIR filters typically require higher orders in order to obtain similar frequency response to an IIR filter. This can lead to increased delay between the occurrence of an event and the detection of the event. In addition, the lower order of IIR filters results in fewer computations, although the typical PMU sampling rate of 30 samples/s is orders of magnitude slower than modern processors which operate in the MHz to GHz range.

Design of FIR filters is also more complex than IIR filter design, primarily due to the lack of closed-form design equations for FIR filters [55].

Because closed form design equations do not exist for FIR filters, several different design techniques are typically employed depending on the application. The two methods most commonly used in FIR filter design are window-based methods and optimization-based methods. Window-based FIR filter design is conceptually and computationally simple, with the following three basic design stages [55]:

1. Specify the ideal response within the frequency domain.
2. Determine the corresponding impulse response via the inverse Fourier transform.
3. Apply a window to determine the causal FIR coefficients which approximate the ideal impulse response.

This application requires low-pass filtering in order to attenuate undesired oscillations, leading to three design parameters for a window-based FIR filter design: the window type, the low-pass cutoff frequency, and the filter order.

In the literature, both Blackman [52] and Hamming [54] windows have been used to process PMU data. The Blackman window is characterized by a wide transition band with decaying sidelobes, while the Hamming window has a sharper transition band with constant sidelobe amplitudes [55]. These two windows, along with the simpler rectangular window (which has the narrowest transition band but the largest sidelobe amplitudes), were tested with a simulated PMU angle signal sampled at 30 samples/s for a FIR low-pass filter design having a cutoff frequency of 0.1 Hz. Figure

2.1 shows the results of applying each of these windows to sample PMU data; clearly, the Hamming window provides the best combination of oscillation attenuation and delay minimization. Figure 2.2 provides the magnitude response of each of the different filters used in Figure 2.1; the filter based on the Hamming window provides a good compromise between the responses of the rectangular window-based filter, which has poorly attenuated sidebands, and the Blackman window-based filter, which has a very wide transition band.

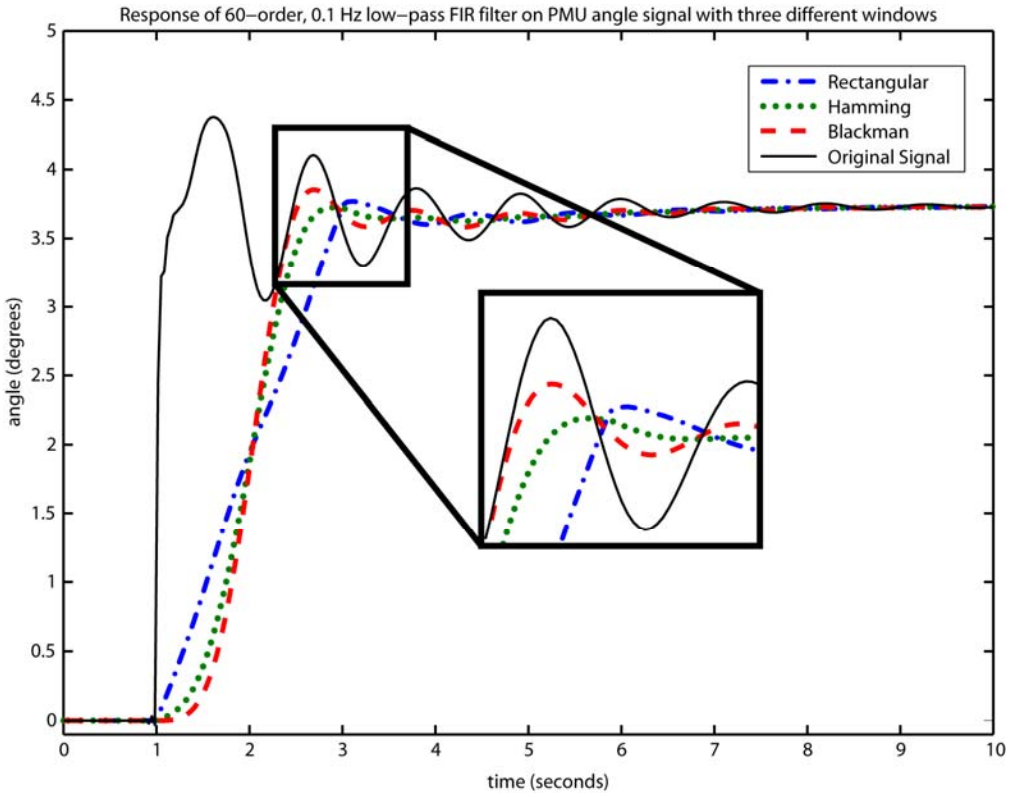

Figure 2.1: Responses of 60-order, low-pass, 0.1-Hz FIR filters on a PMU angle signal using rectangular, Hamming, and Blackman windows.

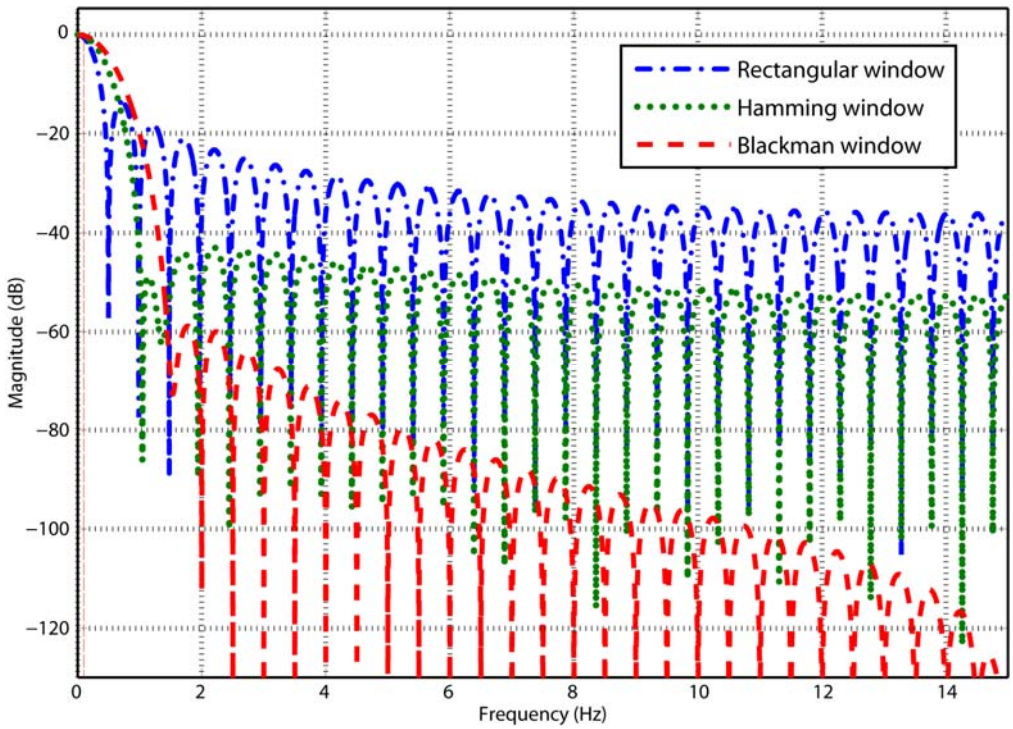

Figure 2.2: Magnitude response of 0.1-Hz, 60-order, low-pass FIR filters with three different window types.

The choice of cutoff frequency is based on the lowest frequency anticipated in the angle oscillations after a system event occurs. Typical power system oscillations at the 15-Hz frequency (the Nyquist frequency for the a PMU data rate of 30 samples/s) and below can be divided into two categories: local modes in the frequency range of 0.7 to 2.0 Hz and interarea modes in the frequency range of 0.1 to 0.8 Hz [56]. The frequencies of the interarea modes on a power system are a function of many factors, including generator excitation systems, load characteristics, and typical system operating conditions [57]. To eliminate as much of these low-frequency oscillations as possible, the low-pass cutoff frequency was set to 0.1 Hz in the FIR filter design. The cutoff frequency could also be tailored to reflect the expected minimum

frequency of electromechanical oscillations on a given system, e.g., 0.5 Hz for the Eastern Interconnect [57] and 0.1 Hz for the Western Interconnect [58].

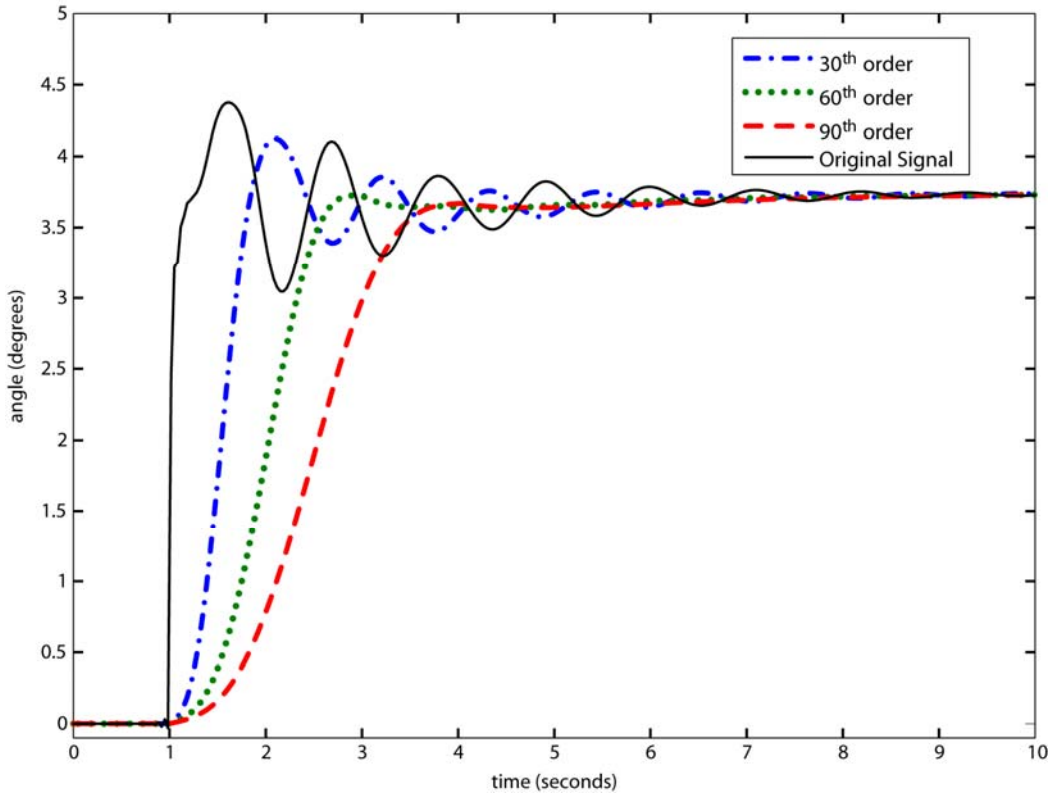

Figure 2.3: Effect of a 0.5-Hz, Hamming-window, low-pass FIR filter on a PMU angle signal with 30, 60, and 90 filter orders.

The last design parameter, filter order, serves as a tradeoff between delay in detection of the event (which, for a symmetric FIR filter, is equal to the filter order minus one divided by two) and attenuation of undesired frequency components. In Figure 2.3, the same signal from Figure 2.1 was filtered with three FIR, Hamming-window filters using filter orders of 30, 60, and 90. The 60-order filter provides better attenuation of the initial overshoot compared to the 30-order filter and has roughly the same attenuation capabilities as the 90-order filter but with reduced delay. Analysis of real and simulated data has shown that using a filter of order 61 is a good tradeoff

between delay (1 s, for a 30 samples/s signal) and attenuation. In addition, Chapter 4 provides a detailed look at how a broad range of FIR filter orders affect the ability to detect events.

Because the goal is to provide a real-time application for use in control centers, the computational complexity of any filtering operation must be taken into account. This provides another bound on the filter order, although, as shown above, the delay and attenuation characteristics are the primary factors used in choosing the desired filter order. For an N-order FIR filter, the calculation of each element of the output signal x_{filt}^{FIR} requires N multiplications and N adds (also known as a multiply and accumulate (MAC) sequence [59]), according to (2.1). Therefore, the order of the FIR filter implementation, using "big-O" notation, is O(N). In terms of storage requirements, only the last N data points must be stored in order to compute the corresponding output point; therefore, storage is also O(N).

2.2 Median Filtering of Measurements

Because much is known about the typical frequency characteristics of power system signals, and FIR linear filters are designed using explicit frequency response characteristics, FIR filters provide a good way to attenuate unwanted oscillations (criterion 2.C1). However, linear filters in general do a poor job of maintaining step changes in signals (criterion 2.C2). Instead of maintaining the sharp transition, linear filters tend to convert the step change into a ramp [32], and this effect can cause a delay in the determination of the angle change vector (criterion 2.C3). To maintain step changes and reduce delay, median filtering provides a possible alternative to FIR

filtering. In addition, the usage of median filtering to smooth out signals in power electronics control applications [32], FNET monitoring [60], and PMU calculations [61] indicates that this filter type can be a viable alternative to linear filtering.

The basic idea of a median filter is to run a sliding window over the data samples, then use the median value from each window as the output value. The typical form used to describe the median filter is [32]

$$x_{filt}^{Med}[n'] = \text{median of}\left\{x[n'-k],\ldots,x[n'],\ldots,x[n'+k]\right\} \qquad (2.3)$$

for a median filter of length $N = 2k + 1$. However, this form of the filter is noncausal because the output samples are dependent on later samples of the input data. To convert this into a causal filter, the substitution $n = n' + k$ is made, resulting in the following causal median filter:

$$x_{filt}^{Med}[n] = \text{median of}\left\{x[n-N+1], x[n-N+2],\ldots,x[n]\right\} \qquad (2.4)$$

By converting from the noncausal form of (2.3) to the causal form (2.4), a delay of $\frac{N-1}{2}$ samples is introduced into the output signal. This delay effect is illustrated in Figure 2.4 for a step change input signal, with the step change occurring at sample 50. The dependence of the delay on the filter length indicates that lowering the filter length improves the ability of the median filter to satisfy criterion 2.C3.

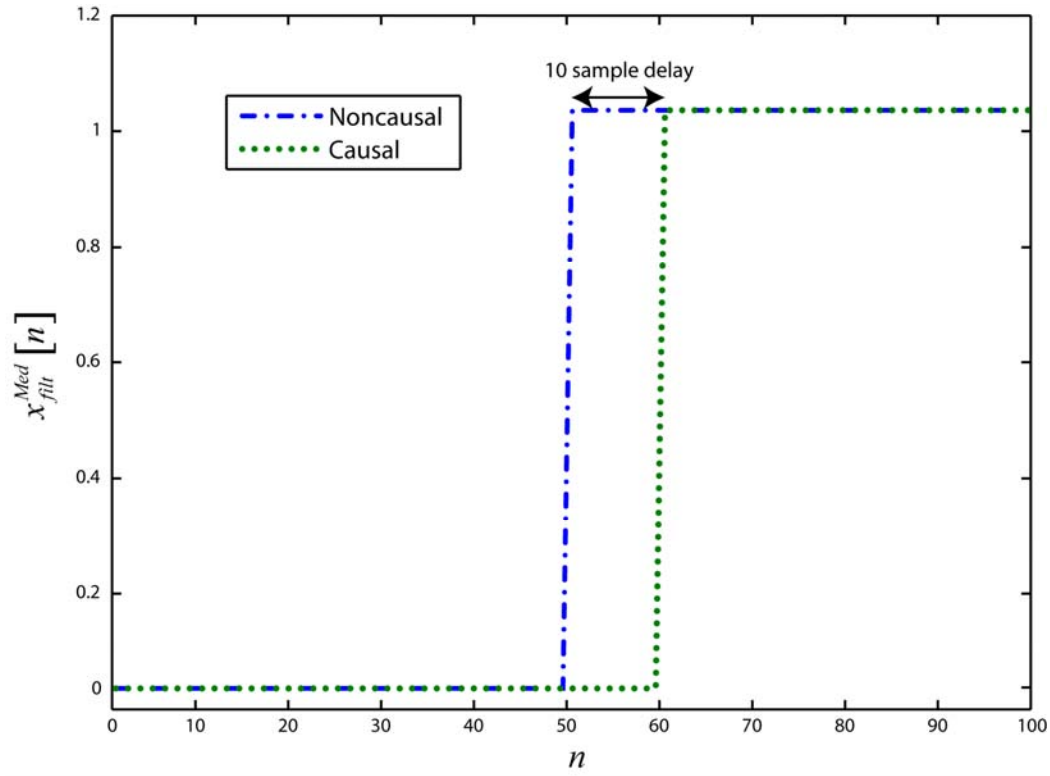

Figure 2.4: Illustration of the delay introduced by using a causal median filter with window length 21 for a step change input signal.

As shown in Figure 2.4, the median filter does not corrupt an unmodified step change, except for the known delay introduced by the causal median filter. Unfortunately, power system signals do not typically follow this ideal step change pattern and instead contain step changes combined with oscillations (as mentioned in the previous section). Therefore, the ability of the median filter to handle low-frequency oscillations must be demonstrated in order to ensure it is a viable alternative to FIR filtering. As presented in (2.4), the median filter only has one design parameter—*N*, the size of the window. To ascertain the performance of the median filter for different values of *N*, the same PMU signal used in Figures 2.1 and 2.3 was tested with several different window sizes. The outputs of these filters are shown in Figure 2.5; as with FIR filtering, it is clear that the window length is closely

tied to the ability to attenuate unwanted oscillations; in addition, an increased window size results in increased delay as expected.

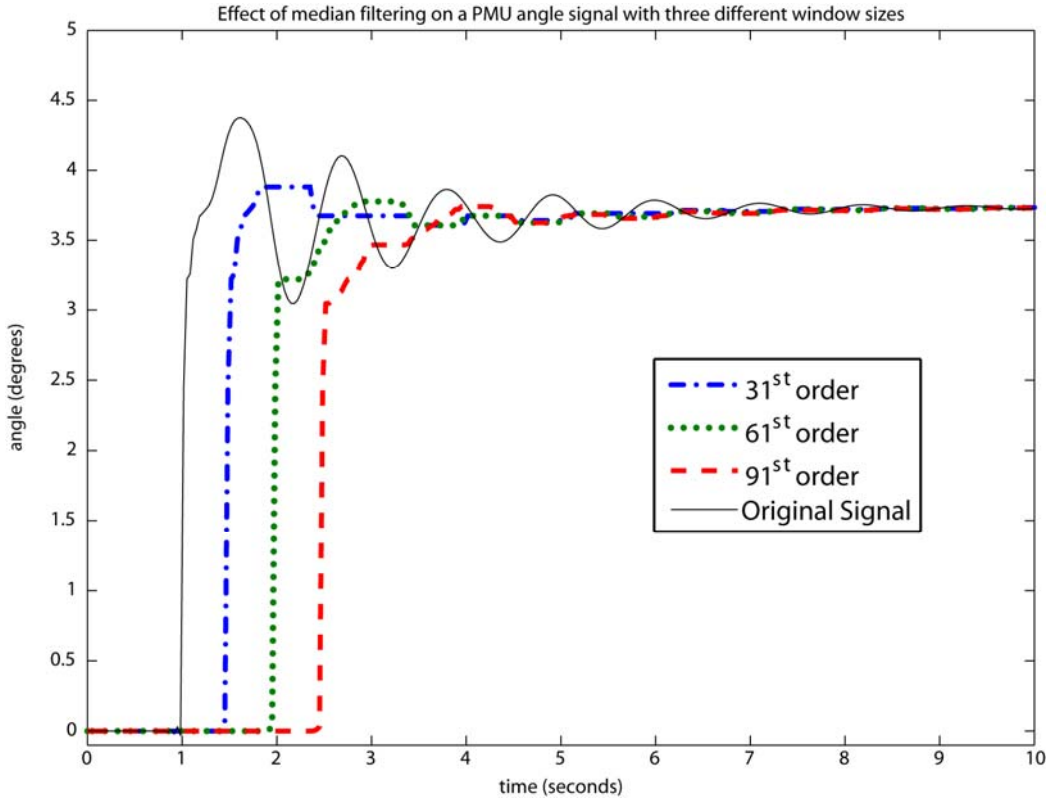

Figure 2.5: Effect of a median filter on a PMU angle signal with window sizes of 31, 61, and 91.

Computational and storage requirements of median filtering are based on the need to sort incoming data points to determine the median value over each window. The fact that the window is moved one sample at a time, so that each new median is taken with just one sample changed, allows for several optimized implementations of the median filtering algorithm [62], including one method which is capable of calculating each new output value in O(log N) time with O(N) storage requirements. Therefore, the computational and storage burden of median filtering is very close to that of FIR filtering. Practically speaking, median filtering data at 4 kHz with a window length of

5 was achievable as early as 1992 using a 20-MHz digital signal processor [32], and the computational requirements of median filtering are not expected to be a determining factor in the filter design.

2.3 Comparing FIR and Median Filtering

In order to compare the performance of FIR and median filtering, several metrics are adapted from [63] and address the filter criteria given as 2.C1-2.C3:

- Percent overshoot (*PO*):

$$PO = \left| \frac{x_{filt}[n_{max}] - x_{ss}}{x_{ss}} \right| \times 100\% \qquad \arg\max_{n \in [n_{begin}, \infty)} \left| x_{filt}[n] \right| = n_{max} \tag{2.5}$$

- Rise samples (RS):

$$RS = n_{95\%} - n_{5\%} \qquad \forall n \geq n_{p\%}, \left| x_{filt}[n] \right| \geq \frac{p}{100\%} \times \left| x_{ss} \right| \tag{2.6}$$

- Delay samples (DS):

$$DS = n_{50\%} - n_{begin} \tag{2.7}$$

The metrics as defined in (2.5)-(2.7) assume the signal has the steady-state value before the onset of the event subtracted from all signal values so that the steady-state value before the onset of the event is adjusted to 0. The percent difference metric is used to quantify how well the post-event oscillations are damped. The sample value n_{begin} is the last sample before the event occurs (i.e., the last sample in which x maintains its pre-event steady-state value) in the original signal $x[n]$. The quantity x_{ss} is the steady-state value associated with the post-event system conditions. In Figure 2.6, the quantities used to calculate percent overshoot are labeled for a given filter

output. As $x_{filt}[n_{max}]$ becomes closer to x_{ss}, the percent overshoot approaches zero and the criterion 2.C1, removal of unwanted noise and oscillations, is better satisfied. Therefore, the best filter would be one which has a percent overshoot of zero, and the closeness of a given filter to zero is a quantitative measure of fitness with respect to criterion 2.C1.

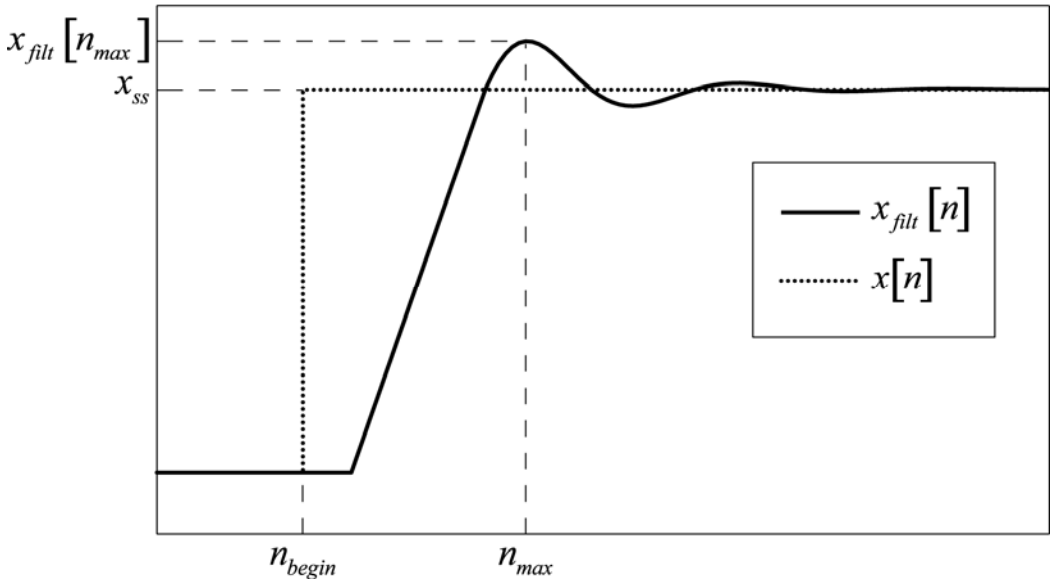

Figure 2.6: Values used to calculate percent overshoot (*PO*).

The second metric given above, rise samples (*RS*), provides a measure of how many samples it takes for the output of the filter to reach the new steady state. Delay is eliminated from this calculation by referencing the rise interval from the sample at which 5% the new steady state is attained, rather than n_{begin}. This separates the delay introduced by the filter from the spreading of the edge information, i.e., it separates the filter performance with respect to criteria 2.C2 and 2.C3. Figure 2.7 illustrates the values needed to determine the *RS* value for a given filter's output. For perfect edge preservation, $n_{95\%}$ and $n_{5\%}$ would be the same sample, leading to an *RS* value of 0 samples. At the other extreme, for a filter which results in converting the step change

into a very slowly rising signal, *RS* will be a high number. Therefore, *RS* provides a quantitative measure of the fitness of the filter with respect to criterion 2.C2.

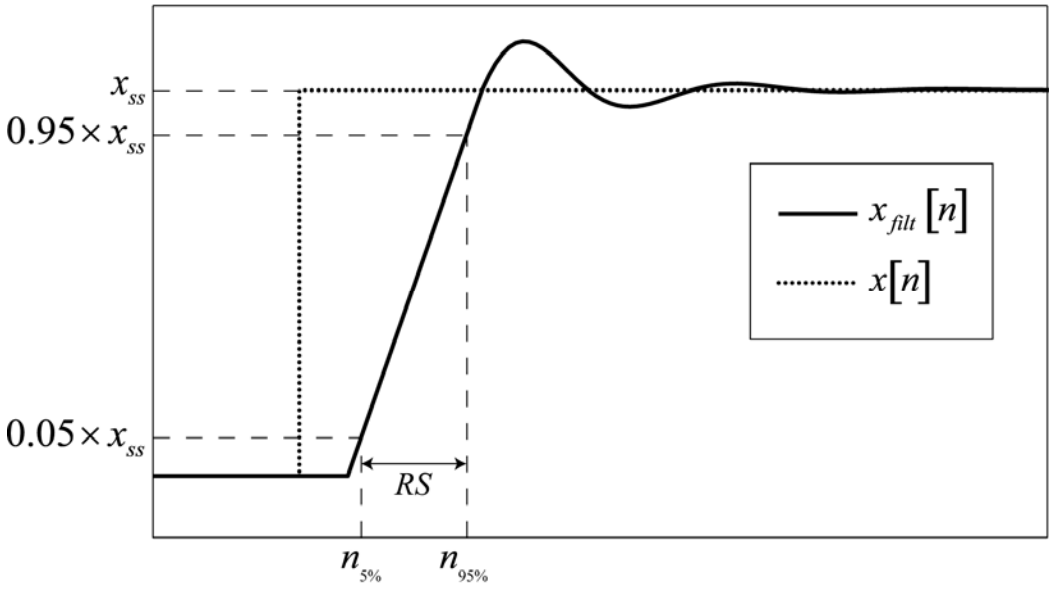

Figure 2.7: Values used to calculate rise samples (*RS*).

The last of the metrics defined above, delay samples (*DS*), is a measure of how long after an event occurs before the change in state can be observed. The delay amount is based on the number of samples it takes for the filtered signal to reach 50% of the final steady-state value x_{ss}. This is an important metric in terms of situational awareness, because any delay introduced by the filtering operation can reduce the ability of system operators to react to changing system conditions. Figure 2.8 illustrates how the *DS* value is calculated for a given filter output. By reducing the amount of delay introduced by the filter, the *DS* value can be brought to zero; however, as discussed in the previous sections on FIR and median filtering, this is generally not possible due to the usage of causal filtering. However, minimizing *DS*

is a key concern in order to improve the capability of operators to react to changing network conditions and serves as a quantitative measurement of criterion 2.C3.

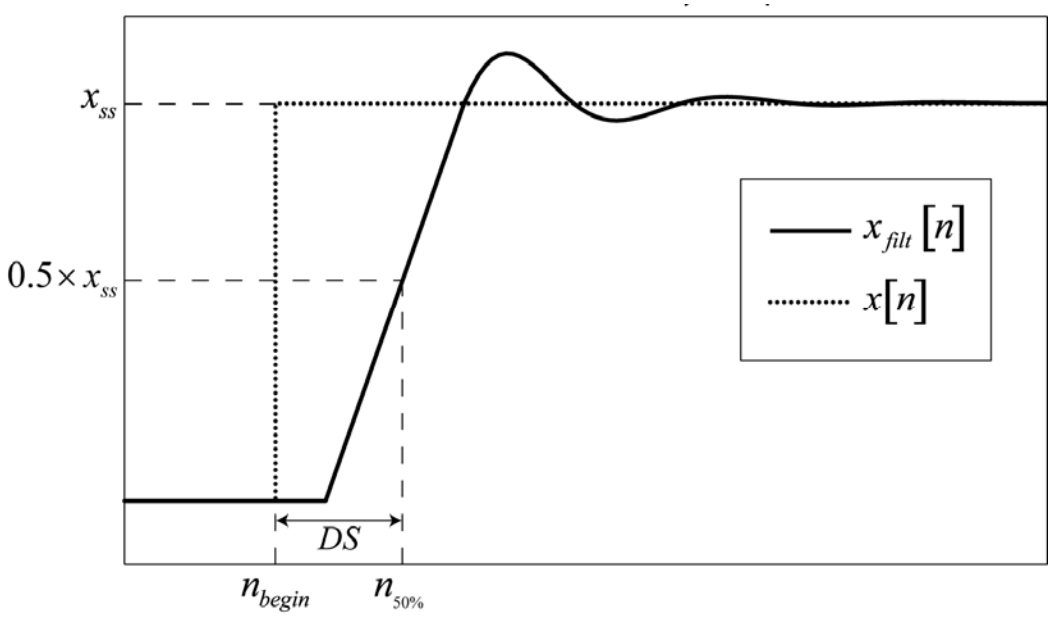

Figure 2.8: Values used to calculate delay samples (*DS*).

2.4 Filter Evaluation

Using the *PO*, *RS*, and *DS* definitions, it is possible to provide a quantitative description of the different filtering options available for usage in PMU-based event detection. A series of tests was run in order to evaluate the capability of various FIR and median filter designs for this application. The results of these tests are presented below.

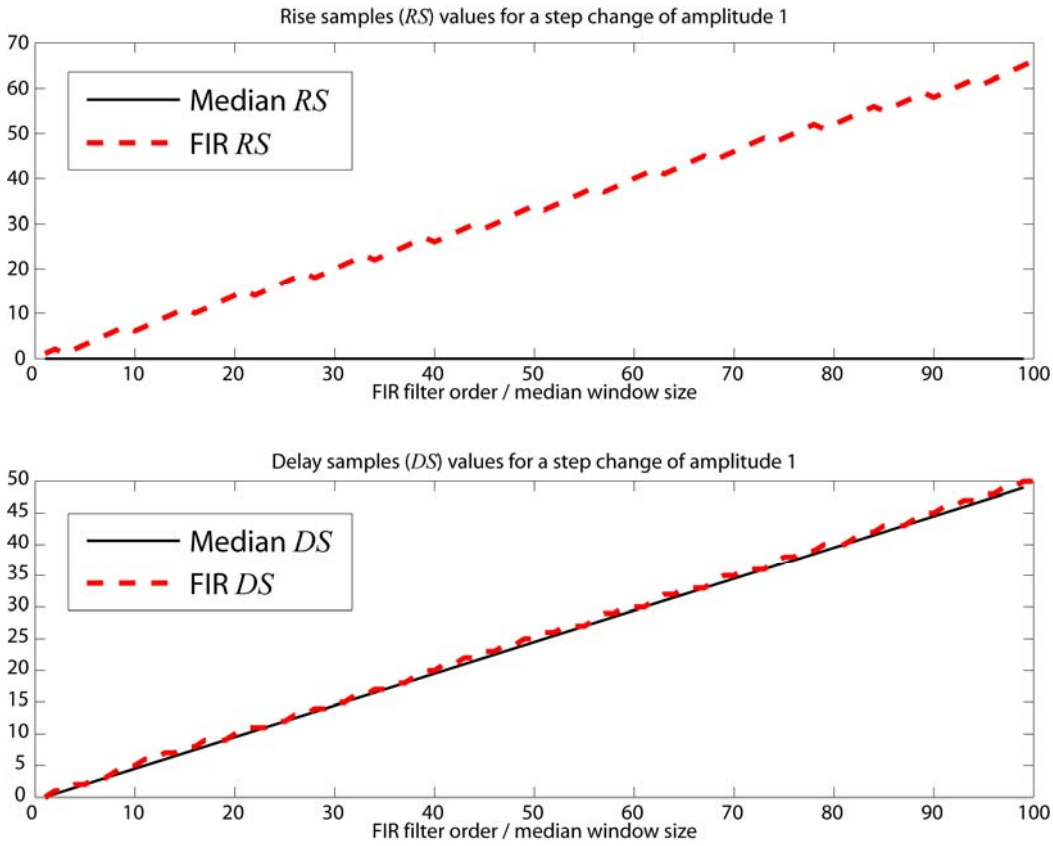

Figure 2.9: *RS* and *DS* values for FIR and median filters with filter orders / window sizes in the range of 1-100 and 1-99 applied to a unit step change.

2.4.1 Step change with no oscillations

The first test performed looks at the response of each filter to a step change in the input signal with no oscillations after the step change. This would correspond to a system which has complete damping of all modes after an event occurs. The signal was constructed to have a sampling time of 1/30 s, corresponding to typical PMU signals. The onset of the step change was set to occur at sample 60 (i.e., $n_{begin} = 60$), and the amplitude of the step change was set to 1.0. Figure 2.9 illustrates the performance of the FIR and median filters for filter orders and window lengths ranging from 1 to 99. The *PO* values are not displayed, as the values were zero to within machine precision for all of the filters tested with the basic step change. The

uniformly zero value of *RS* for the median filters indicates that the median filter does not corrupt the edge in any way; in contrast, the FIR filters convert the step change into a ramp and result in significantly higher *RS* values. In terms of delay, the linear relationship shown in the second half of Figure 2.9 shows that the FIR and median filters have equivalent delays for a given filter order or window size (i.e., using an FIR filter of order *N* results in approximately the same delay as using a median filter with window size *N*).

Although these results show extremely linear behavior for *RS* and *DS* in terms of the filter order, the likelihood of observing a true step change on the power system is very small; therefore, more realistic signals must also be tested.

2.4.2 Step change with 1-Hz oscillations

The second type of signal which the filters were tested with is a unit step change with oscillations added after the step change. The oscillation was set to a frequency of 1 Hz, with a 0.2 damping ratio and magnitude of 0.2, giving rise to the following signal:

$$x[n] = \begin{cases} 0 & n < 60 \\ 1 + 0.2e^{-2\frac{(n-60)}{30}} \sin\left(2\pi \frac{(n-60)}{30}\right) & n \geq 60 \end{cases} \tag{2.8}$$

The unfiltered signal is shown in Figure 2.10.

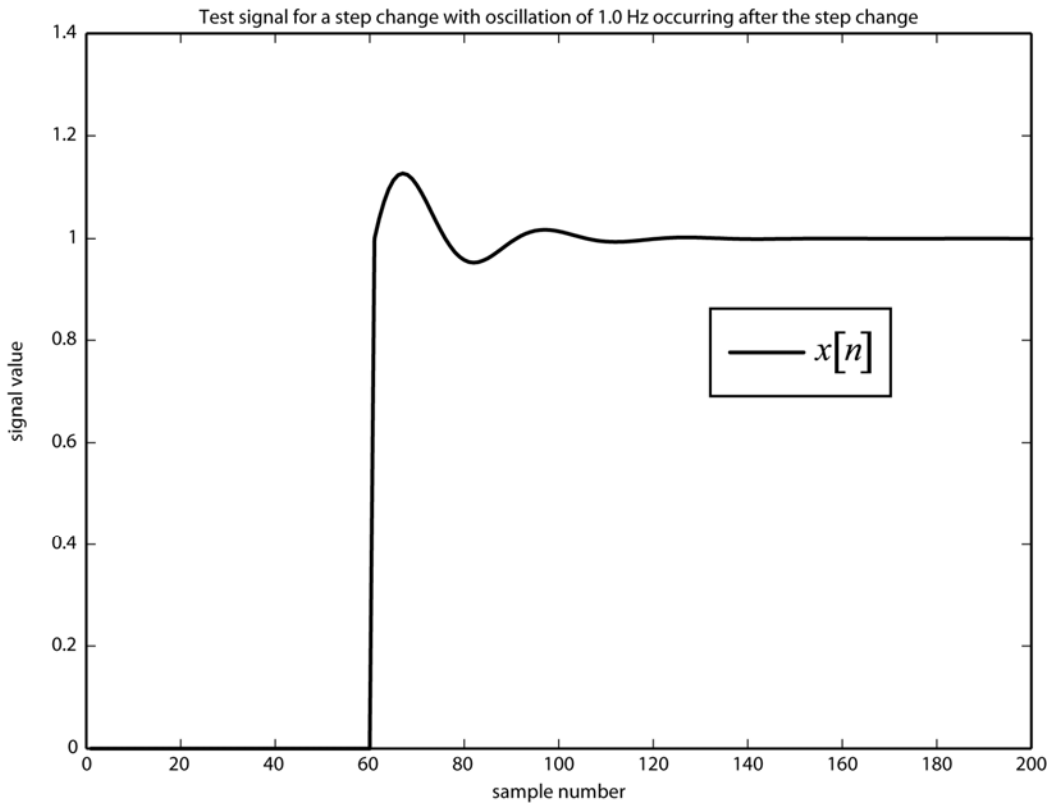

Figure 2.10: Unfiltered test signal used to test filter responses for a step change with 1-Hz oscillations after the step change.

As with the step change test outlined in the previous section, the *RS* and *DS* values were calculated for FIR filter orders of 1-100 and median window lengths of 1-99. The results obtained are shown in Figure 2.11. By comparing this figure to Figure 2.9, it is clear that the *RS* and *DS* values associated with a given FIR filter order or median window length are due primarily to the length of the filter rather than the characteristics of the original signal.

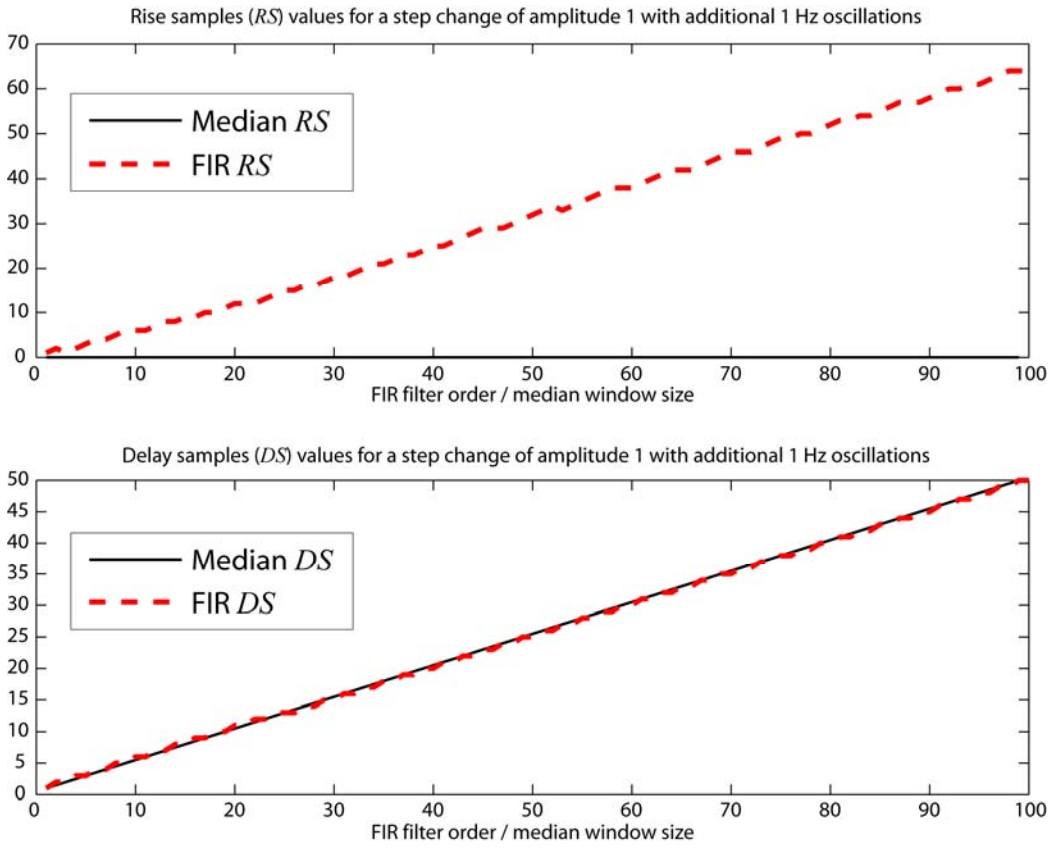

Figure 2.11: *RS* and *DS* values for FIR and median filters with filter orders / window sizes in the range of 1-100 and 1-99 applied to a unit step change with post-step oscillations of 1 Hz.

The percent overshoot values associated with the different FIR and median filter lengths are shown in Figure 2.12. For low filter lengths (less than 30), the median filters perform slightly better than the FIR filters; for filter lengths between 35 and 55, the FIR filters perform slightly better than the median filters; and for filter lengths beyond 55, the two filters are essentially equivalent in performance. The key result shown in this figure is that, although the median filters have consistently lower *RS* values compared to the FIR filters, there are very few differences between the two filter types in terms of *DS* and *PO* for a given filter length. Therefore, usage of

median filtering results in significant gains with respect to edge corruption with little to no compromise in delay or percent overshoot.

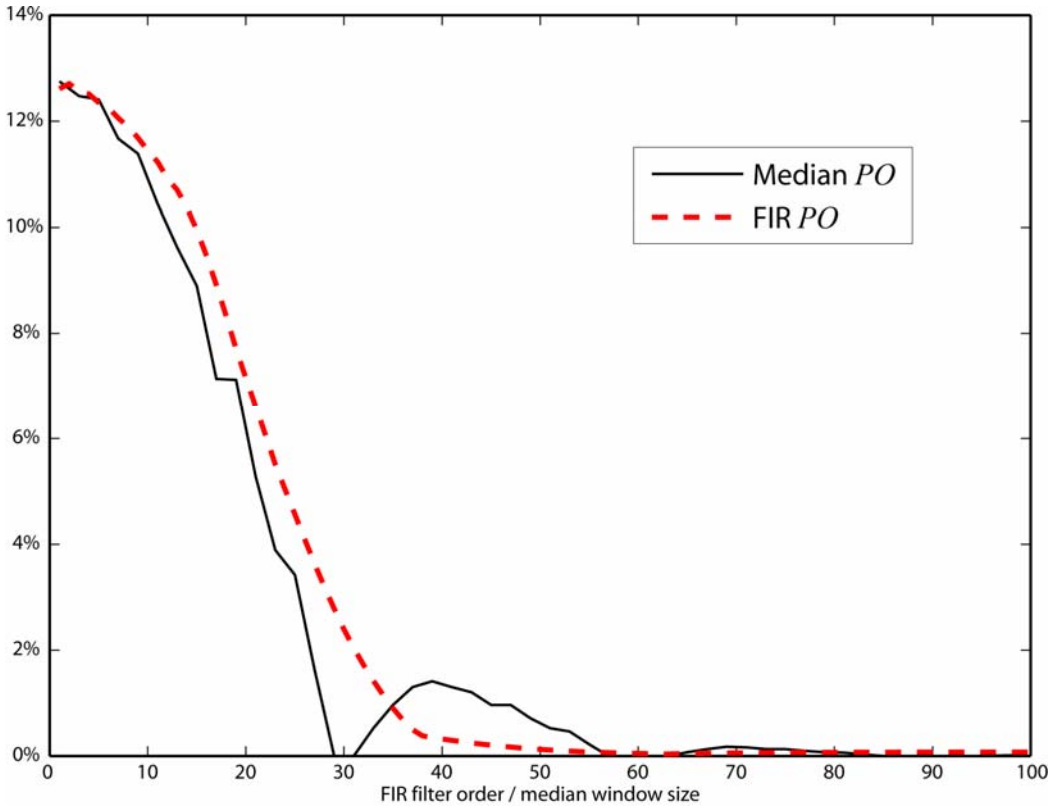

Figure 2.12: *PO* values for FIR and median filters with filter orders / window sizes in the range of 1-100 and 1-99 applied to a unit step change with post-step oscillations of 1 Hz.

2.4.3 Step change with variable oscillations

Because the data captured by PMUs connected to a real power system can exhibit oscillatory modes over the range of 0.1 to 15 Hz, it is important to understand how the filters behave in the presence of these frequencies. To achieve this goal, *RS*, *DS*, and *PO* values were calculated for signals identical to Figure 2.10 (i.e., with amplitude of 0.2 and damping of 2), with the poststep oscillation frequency varied in 0.1-Hz increments from 0.1 to 15 Hz. The FIR filters tests were conducted with filter orders

ranging from 1 to 99 in increments of 2, and the median filter tests were conducted with window lengths ranging from 1 to 99 in increments of 2.

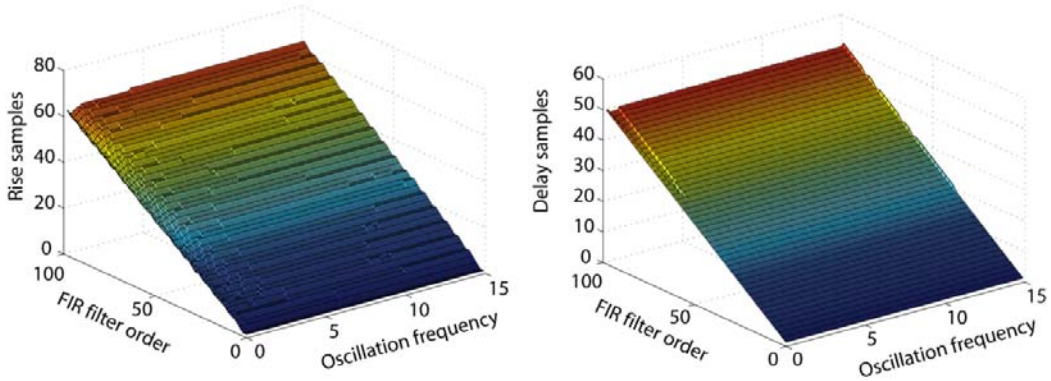

Figure 2.13: *RS* and *DS* values for FIR filters with filter orders in the range of 1-99 applied to a unit step change with poststep oscillations of 0.1-15 Hz.

Figure 2.13 shows that the *RS* and *DS* values associated with a particular FIR filter order are independent of the oscillation frequency. The *DS* value is independent of frequency because the delay of an FIR filter is a function of its filter order, and *RS* is independent because this metric quantifies the effect of the filter on the edge, not the oscillations. The *RS* and *DS* results for the median filters are shown in Figure 2.14; as with the signal tests in Sections 2.4.1 and 2.4.2, median filters have similar *DS* values to the FIR filters of the same length. The nonzero *RS* values are due to the interference of the poststep oscillations with the median determination around the step boundary (known as "edge jitter" [64]); however, the highest *RS* value obtained is nine samples, much lower than the corresponding FIR filters with similar delays.

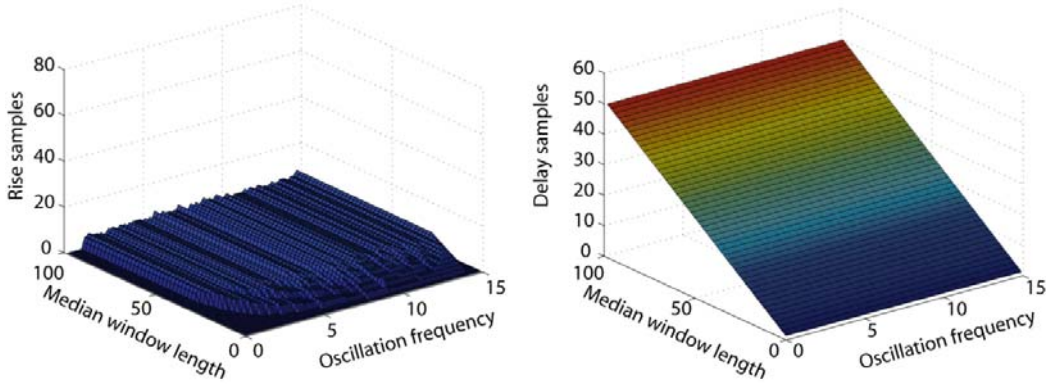

Figure 2.14: *RS* and *DS* values for median filters with window lengths in the range of 1-99 applied to a unit step change with poststep oscillations of 0.1-15 Hz.

The most significant differences with respect to the two filter types are exhibited in the *PO* values. The *PO* values for the FIR and median filters are provided in Figures 2.15 and 2.16, respectively. The *PO* value for each oscillation frequency consistently decreases as the FIR filter order is increased, as expected from the frequency response shown in Figure 2.2. On the other hand, the median filters perform about the same at both low and high frequency extremes.

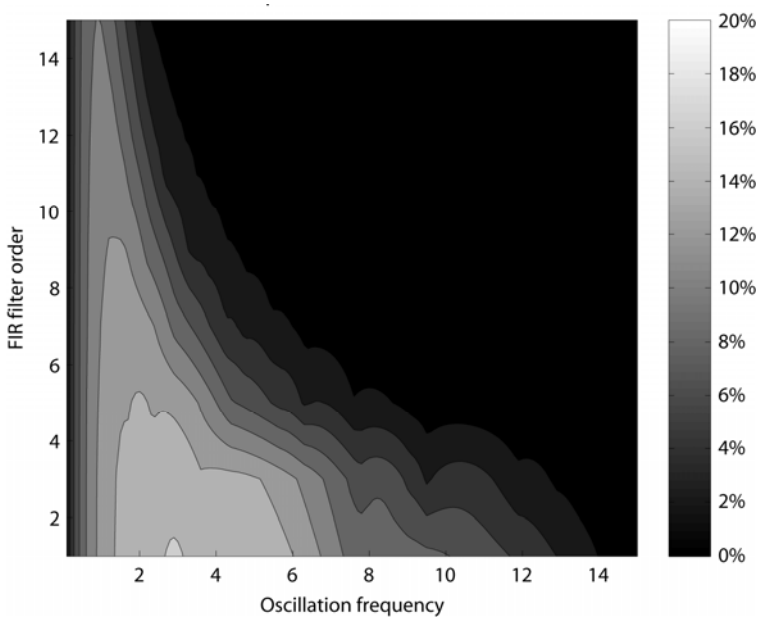

Figure 2.15: *PO* values for FIR filters with filter orders in the range of 1-15 applied to a unit step change with poststep oscillations of 0.1-15 Hz.

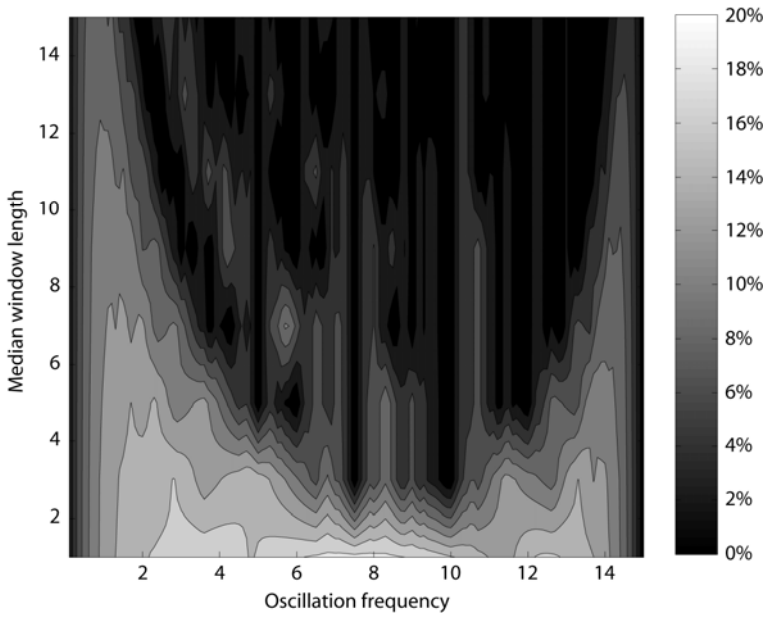

Figure 2.16: *PO* values for median filters with window lengths in the range of 1-15 applied to a unit step change with poststep oscillations of 0.1-15 Hz.

Although the *RS* values are significantly higher with FIR filters regardless of the filter length or oscillation frequency, there is a tradeoff between *DS* and *PO* values for both filter types. To investigate this relationship, a new quantity is defined, $PODS_{x\%}^{\text{filter type}}$, which is the minimum *DS* value such that for all oscillation frequencies in the range of 0.1-15 Hz, the *PO* value is less than or equal to *x*% for the filter type. The difference in this metric for the two filter types, $PODS_{x\%}^{\text{FIR}} - PODS_{x\%}^{\text{median}}$, can indicate which filter to chose for a desired minimum *PO*. Figure 2.17 shows that for small percent overshoot requirements ($x < 14$), median filtering can achieve the desired percent overshoot with less delay than FIR filtering.

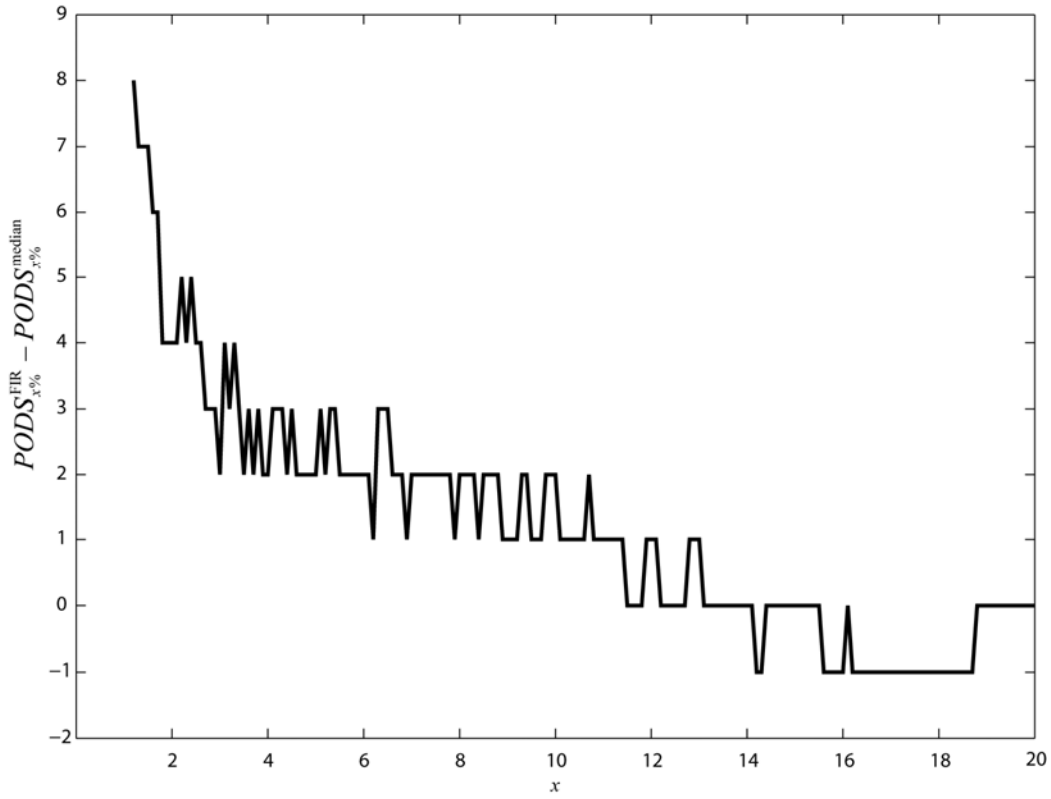

Figure 2.17: $PODS_{x\%}^{\mathrm{FIR}} - PODS_{x\%}^{\mathrm{median}}$ based on unit step and oscillation signals with frequencies of 0.1-15 Hz.

2.4.4 Real PMU signal

To test the filters' performance in extracting steady-state angles from PMU signals, a test was also conducted using a PMU angle signal obtained from a North American power company. The particular signal used for this test, along with the steady state estimate, is shown in Figure 2.18. The value for n_{begin} was determined by inspection to be 401. The steady state value after the event occurs was estimated to be 4.28, obtained by taking the mean of the signal values after n_{begin}.

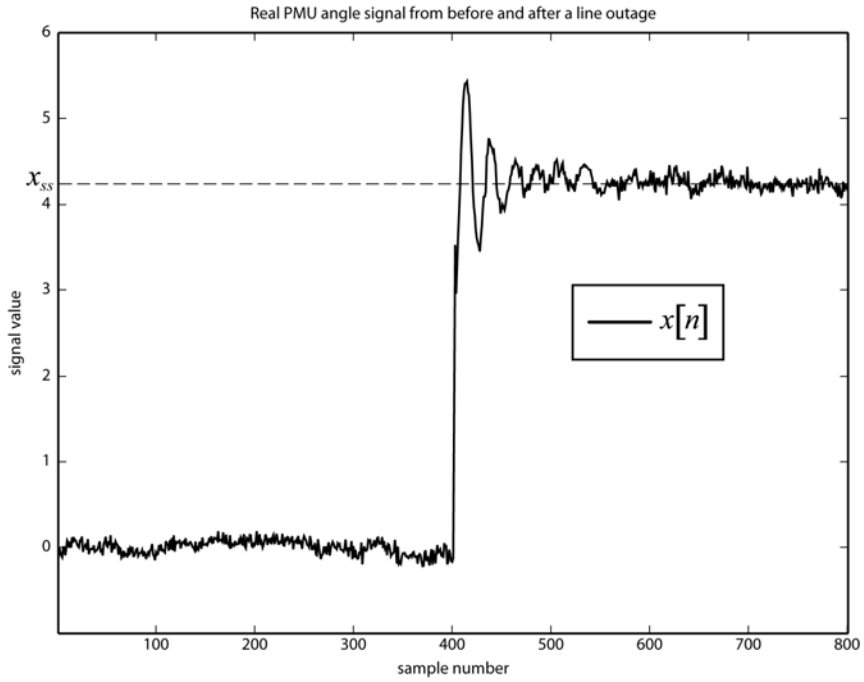

Figure 2.18: Real PMU angle signal used to test filters.

FIR and median filters were tested with filter lengths in the range of 1-99. The *RS* and *DS* values obtained are shown in Figure 2.19. As in previous tests, the *DS*

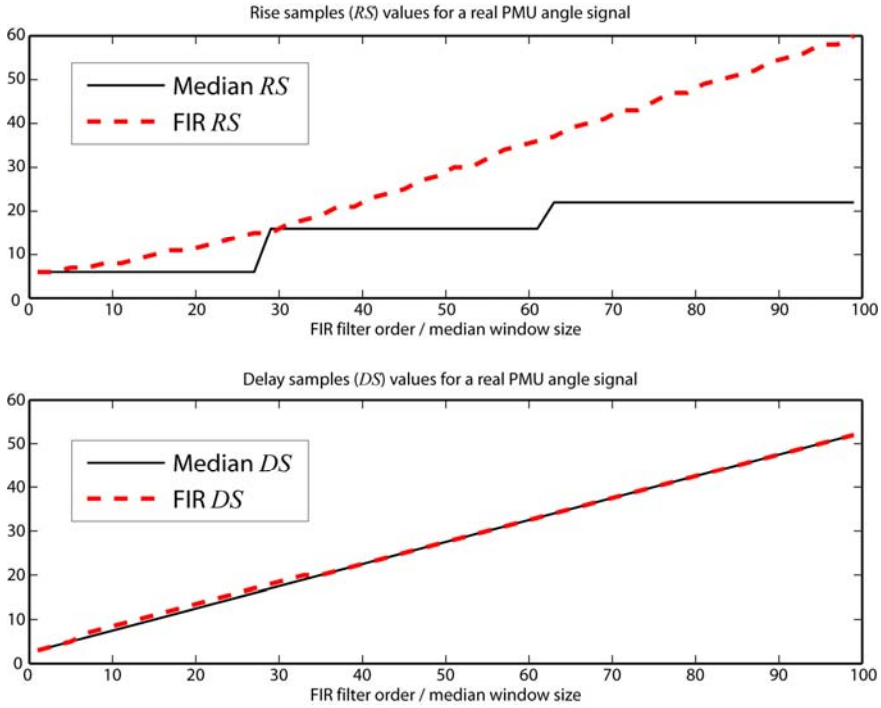

Figure 2.19: *RS* and *DS* values for median and FIR filtering of a real PMU angle signal.

values are the same for a given filter order or median length, and the median filter has a lower *RS* value than the FIR filter for all filter lengths except one (the filter length of 29). As in the previous tests, the median filters are able to preserve the edge shape (lower *RS* value) for the same amount of delay (*DS* value).

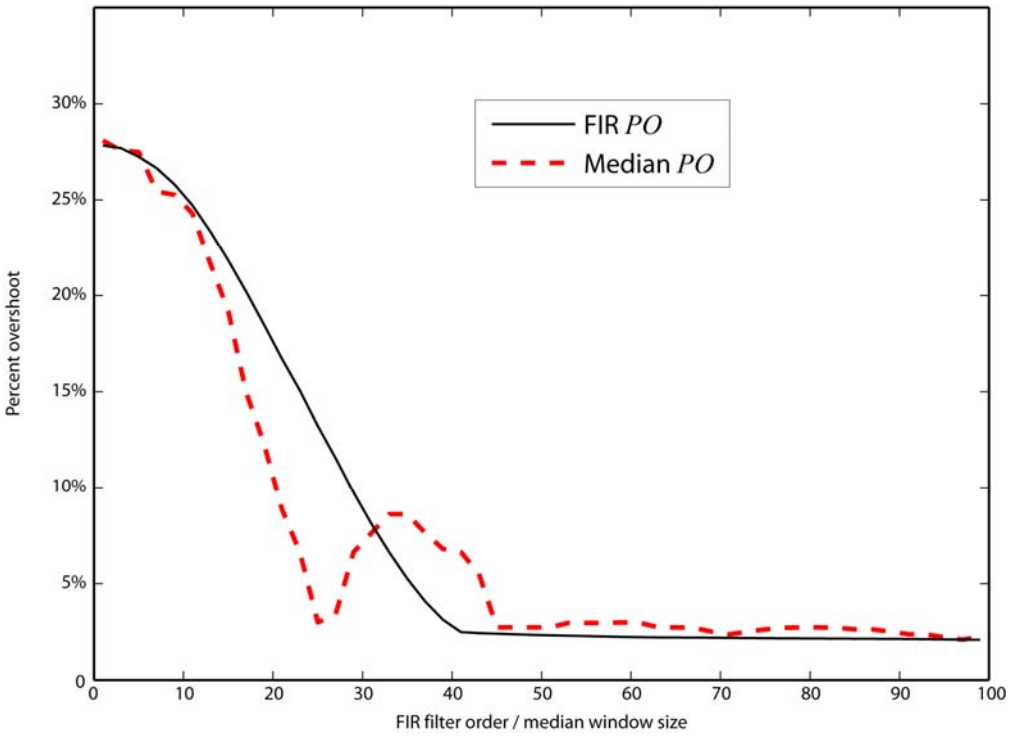

Figure 2.20: *PO* values for median and FIR filtering of a real PMU angle signal.

The final metric, percent overshoot, is displayed in Figure 2.20. Based on the results shown in Figure 2.15, the expected monotonic decrease in *PO* values for increasing FIR filter orders is clearly seen. The characteristics of median filtering are more difficult to characterize; for median window lengths which closely match the oscillation period of the original signal, the median filter does an excellent job of eliminating oscillations and results in very low overshoot values. On the other hand, as the median window length increases, the *PO* value increases. This is not desirable,

as it indicates an increased amount of delay and processing time is actually reducing the effectiveness of the filter. This type of behavior is also seen in Figure 2.16 and shows that the median filter is not as robust as FIR filtering, particularly if the oscillation frequencies are either unknown or widely varying.

2.5 Conclusions

The sensitivity of the filter response to oscillation frequencies is much more prominent in median filtering than in FIR filtering. For systems in which the postevent oscillation frequencies are easily predicted and at specific frequencies, median filtering can provide much better edge preservation while maintaining low percent overshoot. On the other hand, for systems with broad ranges of oscillation frequencies, higher-order FIR filtering is more robust. Because the oscillations seen on a system are due to an array of factors, including the type of event and the event location, it is recommended that FIR filtering be used to maintain robustness. The main downside of FIR filtering, edge distortion, is accounted for in the determination of $\Delta\theta_{observed}$, as discussed in the following chapter.

3 DETECTION OF EVENTS USING PMU DATA

3.1 Event Detection and Determination of the Angle Change Vector

3.1.1 Basic procedure

The raw angle measurements are first filtered using one of the methods presented in the previous chapter, with the output of the filter named $\theta_{i,filt}[n]$ for the filtered phasor angle measurements from bus *i*. Once the angles have been filtered, a candidate angle change signal $\Delta\theta_{i,candidate}[n]$ is constructed for each bus *i*:

$$\Delta\theta_{i,candidate}[n] = \theta_{i,filt}[n] - \theta_{i,filt}[n - N_{trans}] \tag{3.1}$$

where N_{trans} is the number of samples over which the difference in angles is calculated (see Figure 3.1).

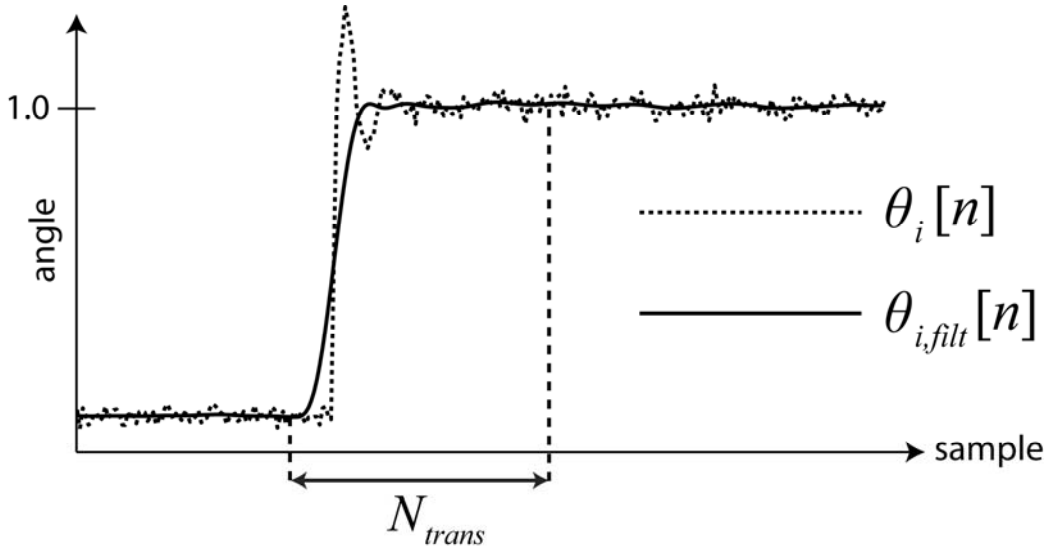

Figure 3.1: Definition of N_{trans}, which is used to generate candidate signals $\Delta\theta_{i,candidate}[n]$.

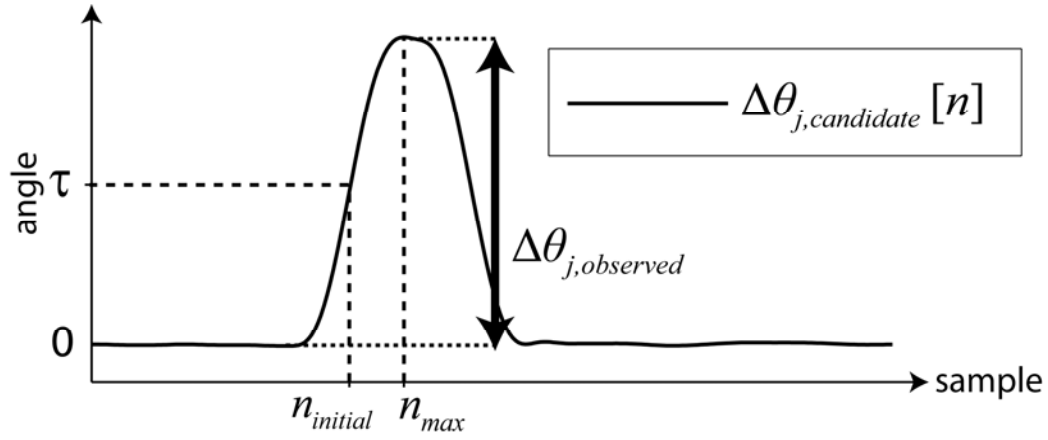

Figure 3.2: Peak detection and determination of the angle change vector.

To detect whether an event has occurred, a method commonly used in edge detection [30] was adapted for our purposes. The first step in the process is to continuously compare each of the candidate signals $\left|\Delta\theta_{i,candidate}[n]\right|$ against a threshold value τ. If the candidate signal at bus *j* exceeds the threshold value at sample $n_{initial}$, candidate signal $\left|\Delta\theta_{i,candidate}[n]\right|$ is then tracked for $n > n_{initial}$ until it begins to decrease. A decrease implies that the maximum of $\left|\Delta\theta_{i,candidate}[n]\right|$ has been reached, and the $\Delta\boldsymbol{\theta}_{observed}$ vector is then constructed using the angle information from all of the buses. The pseudo-code for this "hill climbing" procedure, visualized in Figure 3.2, is:

$$
\begin{aligned}
&\text{if } \left|\Delta\theta_{j,candidate}[n_{initial}]\right| > \tau \text{ for a bus } j,\\
&n_{max} \leftarrow n_{initial}\\
&n_{test} \leftarrow n_{initial}\\
&\Delta^2\theta_{j,candidate} \leftarrow +\infty\\
&\Lambda \leftarrow \text{sign}\left(\Delta\theta_{j,candidate}[n_{initial}]\right)\\
&\text{while } \left(\Delta^2\theta_{j,candidate} \geq 0\right)\\
&\quad \Delta^2\theta_{j,candidate} \leftarrow \Lambda \times \left(\Delta\theta_{j,candidate}[n_{test}+1] - \Delta\theta_{j,candidate}[n_{test}]\right)\\
&\quad n_{test} \leftarrow n_{test}+1\\
&\quad \text{if } \left(\Delta^2\theta_{j,candidate} > 0\right) \text{ then } n_{max} \leftarrow n_{test}
\end{aligned}
\tag{3.2}
$$

After n_{max} has been determined according to (3.2) using the angle information from bus *j*, the observed angle change vector is then constructed using the n_{max} samples from all of the candidate signals:

$$\Delta\boldsymbol{\theta}_{observed} = \begin{bmatrix} \Delta\theta_{1,candidate}[n_{max}] \\ \Delta\theta_{2,candidate}[n_{max}] \\ \vdots \\ \Delta\theta_{K,candidate}[n_{max}] \end{bmatrix} \tag{3.3}$$

A key assumption is needed for (3.3) to truly represent the change in steady state angles at all buses—namely, that the n_{max} value obtained from the measurements at bus *j* corresponds to the steady state changes at the other buses. This assumption will be correct if there is a constant delay in the transition from the old to new steady-state angles at each bus in the system (i.e., the *DS* and *RS* values are the same for the angle signal at each bus). For coherent systems, such as regional networks, this is likely to be the case. For large interconnected systems, different buses may reach their steady-state value at a later or earlier time than the first signal to cross the threshold τ. If this behavior is expected, (3.2) can be run independently for each measurement as it passes the threshold, in which case the n_{max} value used in (3.3) would potentially be different for each bus.

3.1.2 Parameter selection

3.1.2.1 Transition samples (N_{trans})

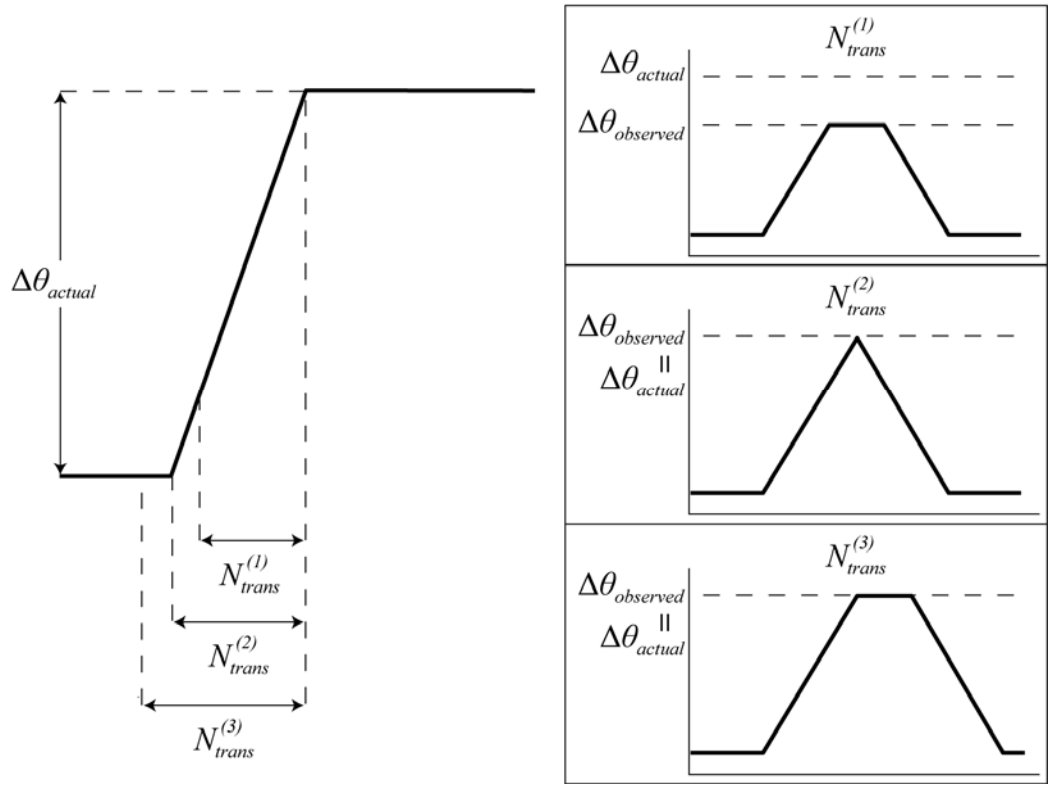

Figure 3.3: Illustration of the relationship between N_{trans} and $\Delta\theta_{observed}$.

There are two key parameters involved in both the event detection and angle change vector determination as formulated in the previous section: N_{trans} and τ. The first parameter, N_{trans}, is the number of samples over which the difference in angle measurements is taken. Figure 3.3 illustrates how different values of N_{trans} impact the accuracy and delay in determination of $\Delta\theta_{observed}$. The value $N_{trans}^{(1)}$ represents the case where N_{trans} is lower than the number of samples over which the transition occurs. The resulting $\Delta\theta_{observed}$ underestimates the true change in angles, $\Delta\theta_{actual}$. The second value shown, $N_{trans}^{(2)}$, represents the case where N_{trans} is optimal and

$\Delta\theta_{observed}$ is equal to $\Delta\theta_{actual}$. The third value, $N_{trans}^{(3)}$, represents the case where N_{trans} is larger than the number of samples needed to transition between states; in this case, it takes longer for the $\Delta\theta_{candidate}$ signal to decrease below $\Delta\theta_{actual}$. The result of this delay is that the while loop condition in (3.2) will take longer to become false, thereby delaying the detection of the event.

Based on the results shown in Figure 3.3, upper and lower bounds on N_{trans} can be described. The lower bound on N_{trans} is based on the need for the entire transition region to fit inside N_{trans} samples (otherwise the resulting $\Delta\theta_{observed}$ signal will underestimate the true difference in steady state angle values as with $N_{trans}^{(1)}$). A new quantity is defined to measure how well a given value of N_{trans} contains the transition band :

$$\Delta\theta_{miss} = \left|\Delta\theta_{observed} - \Delta\theta_{actual}\right| \tag{3.4}$$

Because *RS* provides a measure of the number of samples in the transition from the pre- to postevent angles, *RS* can be considered a lower bound for N_{trans} when using a particular filter. An upper bound on N_{trans} is more difficult to quantify; however, choosing too large of a value for N_{trans} could result in misidentification of the pre-event steady state angle and will delay the number of samples before the while loop in (3.2) is exited. To measure this last effect, a new quantity is defined, n_{while}, representing the number of samples spent inside of the while loop of (3.2). The longer the algorithm spends in the while loop, the longer it takes for the event to be detected; as a result, minimizing n_{while} is essential for obtaining timely event information. The optimal N_{trans} can then be described as the value which minimizes

$\Delta\theta_{miss}$ and n_{while}. Referring back to Figure 3.3, $N_{trans}^{(1)}$ is a poor choice because it results in large $\Delta\theta_{miss}$ and n_{while} values, $N_{trans}^{(3)}$ is a poor choice because it results in a large n_{while} value, and $N_{trans}^{(2)}$ is the best choice because it minimizes both $\Delta\theta_{miss}$ and n_{while}.

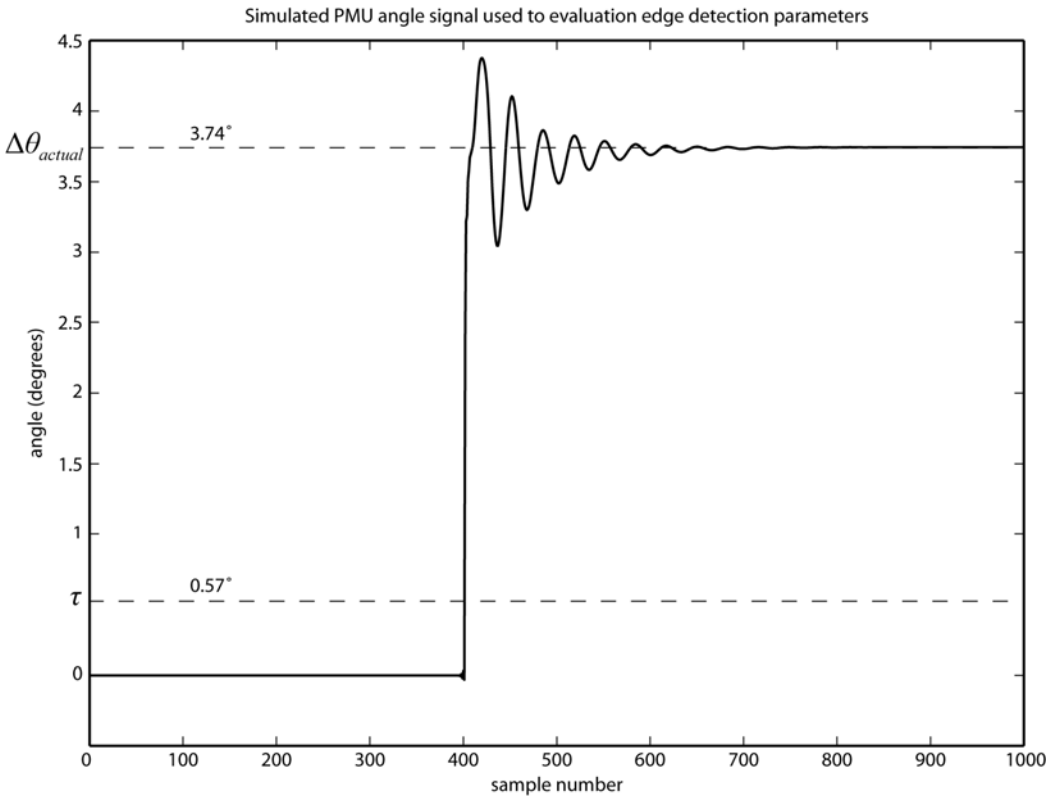

Figure 3.4: Simulated PMU angle signal used to evaluate edge detection parameters.

To examine the effects of N_{trans} on a PMU angle signal, the algorithm defined in (3.2) was run on the simulated PMU angle signal shown in Figure 3.4, with the threshold τ set to 0.57 degrees (the maximum error bound from the IEEE PMU standard [27]). Because of the different behavior of FIR and median filtering, particularly with respect to *RS*, both filter types were evaluated, with filter lengths ranging from 1 to 99 samples.

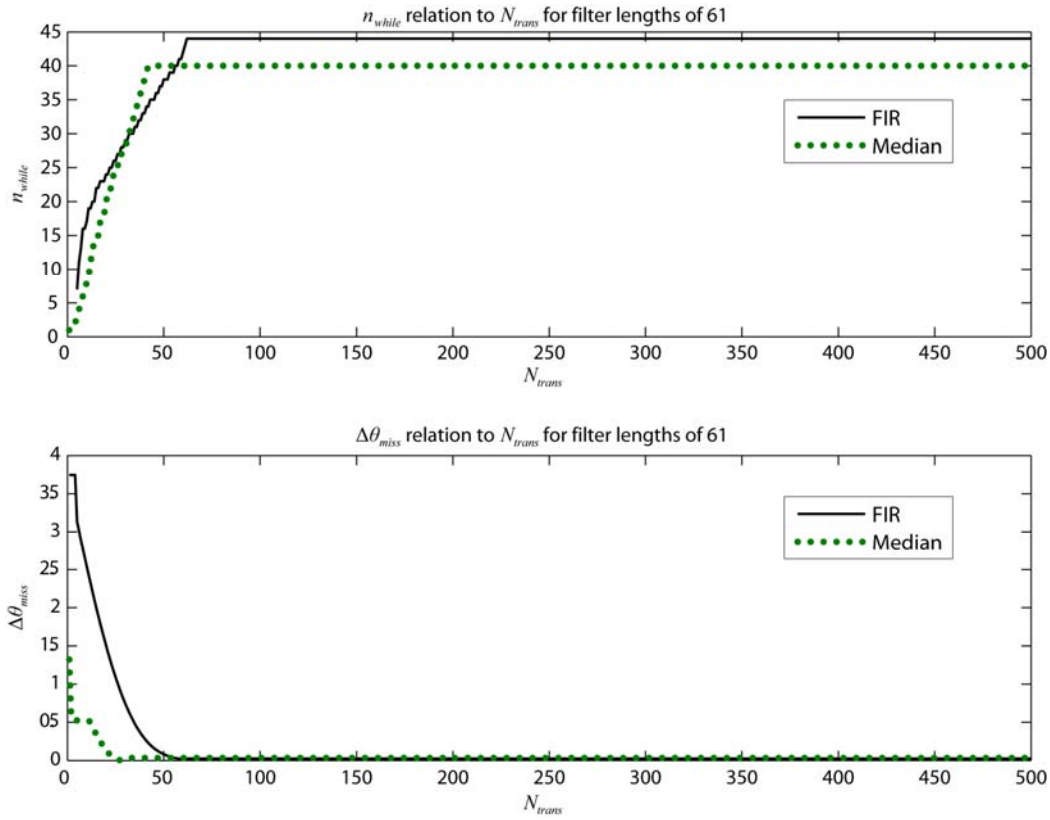

Figure 3.5: $\Delta\theta_{miss}$ and n_{while} values using 61-order FIR and median filtering as N_{trans} values range from 1 to 500 samples.

Figure 3.5 shows the $\Delta\theta_{miss}$ and n_{while} values using FIR and median filters of length 61 as N_{trans} values range from 1 to 500 samples. The median filter is able to minimize $\Delta\theta_{miss}$ with shorter values of n_{while}; this is due primarily to conservation of the step change after the median filter is applied and is directly related to the lower *RS* values associated with median filtering as described in the previous chapter. On the other hand, the FIR filter results in a better $\Delta\theta_{miss}$ value as N_{trans} gets very large (0.0237 degrees vs. 0.0336 degrees); this indicates that FIR filtering, for a given filter length, is slightly better at filtering out the oscillations present in the original signal. This result corresponds to the reduced *PO* values associated with FIR filtering, as discussed in the previous chapter. Therefore, the choice of using an FIR or median

filter is a tradeoff between minimizing $\Delta\theta_{miss}$ (for which FIR filtering is best) and n_{while} (for which median filtering is best). Similarly, the choice of N_{trans} is also a tradeoff between accuracy and delay—a larger value of N_{trans} tends to increase the value of n_{while} while reducing the value of $\Delta\theta_{miss}$.

In Figure 3.5, the n_{while} values level off after N_{trans} exceeds a certain value for filter lengths of 61. This is because the filters are not able to completely eliminate the overshoot in the original signal; as a result, the problem illustrated with $N_{trans}^{(3)}$ in Figure 3.3 does not occur. In contrast, Figure 3.6 shows that using a much higher order filter (in this case, with a length of 147) results in enough attenuation that the case illustrated by $N_{trans}^{(3)}$ does occur for FIR filtering. This phenomenon does not occur using median filtering because of the discontinuities present in the final signal as shown in Figure 3.7.

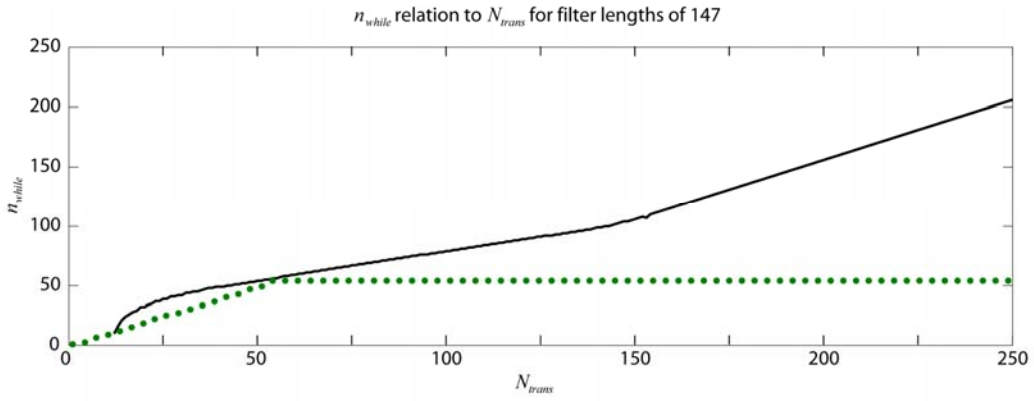

Figure 3.6: n_{while} values using a filter length of 147 FIR and median filtering as N_{trans} values range from 1 to 250 samples.

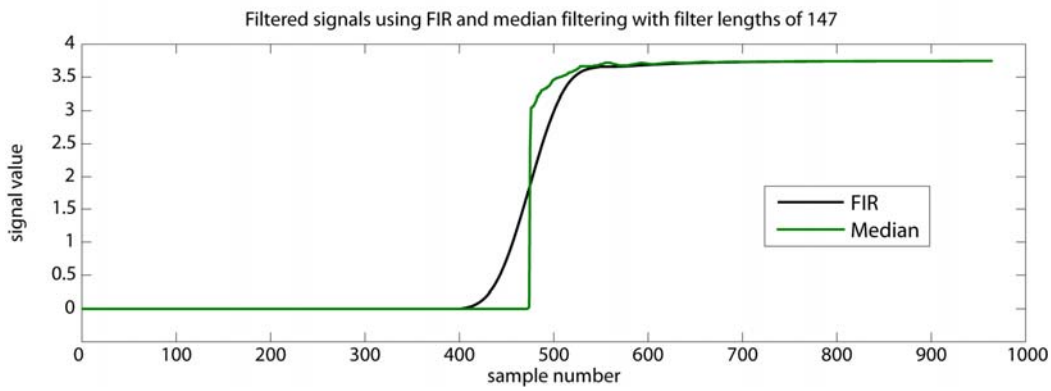

Figure 3.7: Filtering signals obtained using FIR and median filtering of length 147.

3.1.2.2 Difference threshold (τ)

The threshold value τ must also be chosen with care, because setting the threshold value too high might result in missing events that only result in small angle changes (e.g., outages of lines with low pre-outage flow), whereas choosing a threshold value which is too low could result in misclassification of noise as an event. Because the IEEE standard governing PMU behavior [27] requires angle measurements to be accurate within 0.57 degrees of the true angle value, this can serve as a useful upper bound on the threshold value. A useful lower bound on τ is more difficult to specify, primarily due to the lack of documentation detailing the performance of PMUs from different manufacturers. One lower bound is the accuracy of the GPS timing signals, approximately 0.5 μs, which translates to a 0.01-degree phase error [27]. In addition, a study of PMUs manufactured by Schweitzer Engineering Laboratories indicates that they are accurate to within 0.01 degrees [63] for a nominal system signal. The ultimate choice of τ must balance the occurrence of false positives (setting τ too low and incorrectly triggering on nonevents) with false negatives (setting τ too high and neglecting to trigger on events). In addition, usage of filtering to attenuate noise allows for a lower τ without increases in false

positives; therefore, there is a tradeoff between the lower bound on τ and the filter length. This can also be considered as a tradeoff between delay and event misclassification, where increasing delay results in less misclassification.

3.2 Single-Line Outage Detection

3.2.1 Analytical basis for single-line outage detection

When $\mathcal{E}$ is restricted to a set of single line outages on the system, then the problem defined in (1.3) becomes

$$\text{line outaged } l^* = \arg\min_{l\in\{1,2,\ldots,L\}}\left(\min_{P_l}\left\|\Delta\boldsymbol{\theta}_{observed} - deltaAngles_l\left(P_l\right)\right\|\right) \tag{3.5}$$

where L is the number of lines in service before the event is detected and $deltaAngles_l\left(P_l\right)$ is a function which returns the estimated change in angles for the outage of line l with a pre-outage flow of P_l. Because P_l is unknown a priori, it is allowed to vary in order to achieve the best match in observed and calculated angles. Therefore, a unique solution of (3.5) requires that each line outage be distinguishable from the outage of other lines regardless of the pre-outage flow on each line. Solution of (3.5) requires the ability to relate the pre-outage flow on a line l to the observed angle changes on that line if it were to be outaged (represented by $deltaAngles_l\left(P_l\right)$). A simple expression for $deltaAngles_l\left(P_l\right)$ is obtained if the dc power flow equations (1.1) are used. When the dc power flow equations are used, the effect of the outage of a line l can be approximated by a power transfer between the line's "from" bus l_{from} and its "to" bus l_{to} [65]. The transfer amount $\tilde{P}_l$ can be determined from the following equation:

$$\tilde{P}_l = \frac{P_l}{1 - PTDF_{l,l_{from}-l_{to}}} \tag{3.6}$$

where P_l is the pre-outage flow on line l defined as positive if flowing from l_{from} to l_{to}. The value $PTDF_{l,l_{from}-l_{to}}$ is the power transfer distribution factor (PTDF) relating the change in flow on line l due to a transfer from bus l_{from} to bus l_{to} and can be calculated using only topology and impedance information if the dc power flow assumptions are used [65]. If the denominator is zero (i.e., $PTDF_{l,l_{from}-l_{to}} = 1$), this indicates that the line constitutes a radial connection between two otherwise disconnected systems and the outage cannot be represented as a transfer across the line. In this case, the changes in generator dispatch in the disconnected systems must be modeled to capture the line outage. Section 3.3 below proposes methods for modeling generation redispatch.

If the power transfer $\tilde{P}_l$ is imposed on the system, then a change in angles occurs at all buses. To distinguish the observable angles from the complete set of angles at all buses, a $K \times N$ matrix $\mathbf{K}$ is introduced:

$$\mathbf{K} = \left[\mathbf{I}_{K \times K} \quad \mathbf{0}_{K \times (N-K)} \right] \tag{3.7}$$

where K is the number of phasor angles observable from the PMUs, N is the total number of system buses, $\mathbf{I}_{K \times K}$ is the $K \times K$ identity matrix, and $\mathbf{0}_{K \times (N-K)}$ is a $K \times (N-K)$ matrix of zeros. The set of angle changes at the observable buses, which is denoted as $\Delta\boldsymbol{\theta}_{calc,l}^{\tilde{P}_l}$, is then found by applying Equation (1.1):

$$\begin{aligned}\boldsymbol{\Delta\theta}_{calc,l}^{\tilde{P}_l} &= \mathbf{KB}^{-1}\begin{bmatrix}0\\ \tilde{P}_l\\ -\tilde{P}_l\\ 0\end{bmatrix}\begin{matrix}\\ \leftarrow l_{from}\\ \leftarrow l_{to}\\ \\ \end{matrix}\\ &= \tilde{P}_l\mathbf{KB}^{-1}\begin{bmatrix}0\\ 1\\ -1\\ 0\end{bmatrix}\begin{matrix}\\ \leftarrow l_{from}\\ \leftarrow l_{to}\\ \\ \end{matrix}\\ &= \tilde{P}_l\boldsymbol{\Delta\tilde{\theta}}_{calc,l}\end{aligned} \tag{3.8}$$

As shown in (3.8), the changes in angles are linear with respect to $\tilde{P}_l$; therefore, the calculated changes in angles for a particular pre-outage flow on line l can be written as a scalar $\tilde{P}_l$ multiplying a constant vector $\boldsymbol{\Delta\tilde{\theta}}_{calc,l}$. In turn, (3.5) can be rewritten with $deltaAngles_l\left(P_l\right)$ replaced by the appropriate scalar-vector product:

$$\begin{gathered}\text{line outaged } l^* = \\ \underset{l\in\{1,2,\ldots,L\}}{\arg\min}\left(\min_{\tilde{P}_l}\left\|\Delta\boldsymbol{\theta}_{observed} - \tilde{P}_l\boldsymbol{\Delta\tilde{\theta}}_{calc,l}\right\|\right)\end{gathered} \tag{3.9}$$

The optimization given in (3.9) can be performed very quickly using dot products. To see why this is the case, first consider two arbitrary vectors **a** and **b**. From linear algebra, it is known that the projection of **b** onto **a**, $proj_{\mathbf{a}}\mathbf{b}$, is the vector that minimizes **b** - k**a**, where k is allowed to take on any value [66]. The formula for calculating $proj_{\mathbf{a}}\mathbf{b}$ is

$$\begin{gathered}k^* = \frac{\mathbf{a}\cdot\mathbf{b}}{\mathbf{a}\cdot\mathbf{a}} = \arg\min_k\left\|\mathbf{b} - k\mathbf{a}\right\| \\ proj_{\mathbf{a}}\mathbf{b} = k^*\mathbf{a}\end{gathered} \tag{3.10}$$

Comparing Equations (3.9) and (3.10), the inner minimization of (3.9) can be rewritten as

$$\tilde{P}_l^* = \frac{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}}$$
$$\min_{\tilde{P}_l} \left\| \Delta\boldsymbol{\theta}_{observed} - \tilde{P}_l \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right\| = \left\| \Delta\boldsymbol{\theta}_{observed} - \tilde{P}_l^* \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right\| \tag{3.11}$$

The inner minimization can then be eliminated and the complete minimization rewritten using dot products:

$$\text{line outaged } l^* = \arg\min_{l \in \{1,2,\ldots,L\}} \left(\left\| \Delta\boldsymbol{\theta}_{observed} - \left(\frac{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \right) \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right\| \right) \tag{3.12}$$

Some manipulation of (3.12) can provide additional intuition into the nature of the minimization process. Consider first the expansion of the norm into dot products:

$$\begin{aligned}
&\arg\min_{l \in \{1,2,\ldots,L\}} \left(\left\| \Delta\boldsymbol{\theta}_{observed} - \left(\frac{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \right) \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right\| \right) = \\
&\arg\min_{l \in \{1,2,\ldots,L\}} \begin{pmatrix} \left(\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\boldsymbol{\theta}_{observed} \right) \\ -2 \left(\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right) \left(\frac{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \right) \\ + \left(\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right) \left(\frac{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \right)^2 \end{pmatrix} = \\
&\arg\min_{l \in \{1,2,\ldots,L\}} \begin{pmatrix} \frac{\left(\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right)^2}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \\ -2 \frac{\left(\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right)^2}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \end{pmatrix} = \\
&\arg\min_{l \in \{1,2,\ldots,L\}} \left(\frac{-\left(\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l} \right)^2}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \right) = \\
&\arg\max_{l \in \{1,2,\ldots,L\}} \left(\left| \left(\frac{\Delta\boldsymbol{\theta}_{observed}}{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\boldsymbol{\theta}_{observed}} \right) \cdot \left(\frac{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l}} \right) \right| \right)
\end{aligned} \tag{3.13}$$

The last expression indicates that the minimization of (3.12) is equivalent to the maximization of the dot product between the normalized $\Delta\boldsymbol{\theta}_{observed}$ and $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$

vectors. Furthermore, maximization of the dot product is equivalent to minimization of the inverse cosine:

$$\begin{aligned}&\arg\max_{l\in\{1,2,\ldots,L\}}\left(\left|\left(\frac{\Delta\boldsymbol{\theta}_{observed}}{\Delta\boldsymbol{\theta}_{observed}\cdot\Delta\boldsymbol{\theta}_{observed}}\right)\cdot\left(\frac{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}\right)\right|\right)=\\&\arg\min_{l\in\{1,2,\ldots,L\}}\left(\cos^{-1}\left|\left(\frac{\Delta\boldsymbol{\theta}_{observed}}{\Delta\boldsymbol{\theta}_{observed}\cdot\Delta\boldsymbol{\theta}_{observed}}\right)\cdot\left(\frac{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}\right)\right|\right)\end{aligned} \tag{3.14}$$

This last expression indicates that the line outage which best matches the observed angle changes is the one which has a $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ vector that matches the direction of the $\Delta\boldsymbol{\theta}_{observed}$ vector. A normalized angle distance (*NAD*) metric is defined to quantify the difference in the direction between the observed and expected angle changes for each line:

$$\begin{aligned}\theta_l &= \frac{\cos^{-1}\left|\left(\frac{\Delta\boldsymbol{\theta}_{observed}}{\Delta\boldsymbol{\theta}_{observed}\cdot\Delta\boldsymbol{\theta}_{observed}}\right)\cdot\left(\frac{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l}}\right)\right|}{2}\\ NAD_l &= 2\sin\theta_l\end{aligned} \tag{3.15}$$

As shown in Figure 3.8, an *NAD* value of 0 would correspond to a perfect match between the expected and observed angle changes and the maximum *NAD* value of $\sqrt{2}$ would correspond to the worst possible match (where $\Delta\boldsymbol{\theta}_{observed}$ and $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ are perpendicular).

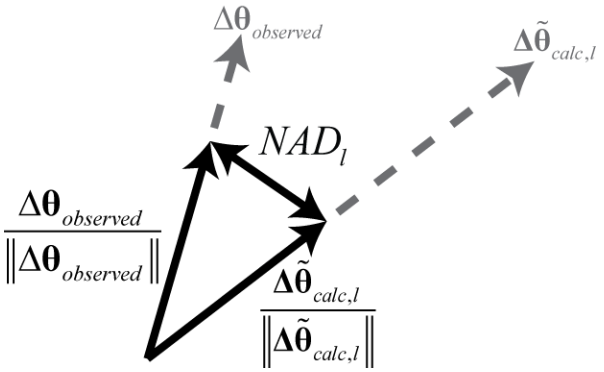

Figure 3.8: Normalized angle difference (*NAD*) metric.

3.2.2 Basic single-line outage event detection algorithm

In order to detect a line outage, identify the outaged line, and determine the pre-outage flow on that line, the following basic algorithm is used:

1. For each line l:
    a. Calculate $\mathbf{\Delta\tilde{\theta}}_{calc,l}$ using (3.8).
    b. Calculate $\tilde{P}_l^*$ using (3.11).
    c. Calculate the *NAD* value for line l using (3.15), then store the calculated error value in the indexed array *NADVals*:

$$NADVals_l = NAD_l \tag{3.16}$$

2. Determine the line l^* that was outaged by sorting *NADVals*:

$$l^* = \arg\min_l NADVals_l \tag{3.17}$$

3. Determine the pre-outage flow on the line which best fits the observed angle, $P_{l^*}^*$, using (3.6):

$$P_{l^*}^* = \tilde{P}_{l^*}^* \left(1 - PTDF_{l^*,l_{from}^*-l_{to}^*}\right) \tag{3.18}$$

3.2.3 Computational complexity

To calculate the angle change due to the outage of a line l using (3.8), the **B** matrix must be factored using LU decomposition [67]. This is the most expensive operation in the algorithm, but it only needs to be performed once per change in topology. Once **B** is factored, $\mathbf{\Delta\tilde{\theta}}_{calc,l}$ and the necessary PTDF values can be computed using forward and backward substitution. In addition, because only certain elements of the angle vector are needed, fast-forward and fast-backward solution can be used to reduce computation time. Outside of step 1.a, the algorithm requires only

addition and dot products with vectors of dimension K and a sort operation. These last operations are highly parallelizable and would see performance gains on the order of the number of computing cores available.

3.2.4 Heuristic modifications to the basic algorithm

Additional information is available about the power system, and this information can be exploited to reduce the search space and improve the performance of the line outage detection algorithm. One useful modification is to reject potential outages requiring excessive pre-outage flow on the line in question. For the studies presented below, this was set to either 150% of the line rating in MVA or 5 GW in cases where the rating was unavailable. In addition, system experts may be capable of evaluating potential outages for their feasibility; to facilitate evaluation by operators, the algorithm can easily output the top outage candidates, along with estimated pre-outage flows.

3.3 Generator Outage Detection

3.3.1 Analytical basis for generator outage detection

As with single-line outages, the basic event detection formulation from (1.3) is first restricted to the set of generator outages:

$$\text{generator outaged } g^* = \arg\min_{g\in\{1,2,\ldots,G\}}\left(\min_{P_l}\left\|\Delta\boldsymbol{\theta}_{observed} - deltaAngles_g\left(P_g\right)\right\|\right) \tag{3.19}$$

In (3.19), $deltaAngles_g\left(P_g\right)$ represents the expected changes in observed angles due to the outage of generator g which, before the outage, is generating P_g. As with the

single-line outage detection, the dc power flow equations are used to derive an expression for $deltaAngles_g\left(P_g\right)$.

Unlike in the line outage case, there are nonzero changes to power injections on the system upon the outage of a generator. In typical power flow studies, the pickup of the remaining generators is modeled in a variety of ways. One common method is through the designation of a slack bus which, upon the outage of a generator on the system, will pick up the "slack" left by the outaged generator. In this case, the power injection vector would have only one nonzero entry at the row corresponding to the outaged generator. This method is the easiest way to account for changes in generation, but it does not factor in the characteristics of the system's generators.

In real system operations, a loss in generation is usually picked up by a combination of the remaining generators rather than just one. Usage of participation factors attempts to capture this behavior. In this method, the entry for each generator bus in the power injection vector is determined based on the participation factor (pf) of the corresponding generating unit, i.e. :

$$\Delta P = \begin{bmatrix} pf_1 \times P_g \\ pf_2 \times P_g \\ \vdots \\ -P_g \\ \vdots \\ pf_G \times P_g \end{bmatrix} \begin{array}{l} \\ \\ \\ \leftarrow \text{row corresponding to} \\ \quad \text{outaged generator} \\ \\ \\ \sum_{i \in \{1,2,\ldots,G\} \setminus g} pf_i = 1 \end{array} \tag{3.20}$$

The summation in (3.20) ensures that all generation lost by the outaged unit g is picked up by the remaining units. The assumption implicit in having the participation factors sum to one is that changes in losses on the system are negligible; if this is not

the case, then the right-hand side of the equality can be modified to reflect the expected changes in losses.

While the formulation in (3.20) is much more flexible than the slack bus approach, the validity of (3.20) is dependent on the accuracy of the participation factors used. These factors are commonly used in contingency analyses and are, as a result, available on an area-wide basis for many control areas; however, a systemwide database of participation factors is unavailable. Nonetheless, participation factors can be estimated based on the characteristics of each generator [68] which are available from systemwide dynamic models. For short term redispatch of generation, one of the key determining factors is each generator's droop and machine base. Droop, otherwise known as speed regulation, is defined as follows [68]:

$$R = \frac{\left(\dfrac{\text{decrease in frequency}}{\text{nominal system frequency}}\right)}{\left(\dfrac{\text{increase in power output}}{\text{maximum power output}}\right)} \times 100\% = \frac{\left(\dfrac{f_{\text{no load}} - f_{\text{full load}}}{f_{\text{nominal}}}\right)}{\left(\dfrac{P^{\max} - 0}{P^{\max}}\right)} \times 100\% = \left(\frac{f_{\text{no load}} - f_{\text{full load}}}{f_{\text{nominal}}}\right) \times 100\% \tag{3.21}$$

The subscripted f quantities refer to electrical frequency at various operating conditions (no load, full load, and nominal system frequency), and $P^{\max}$ refers to the maximum power output of the generator, which is assumed to be the same as the machine base. For example, if the droop of a generator is 1%, then a 1% deviation in frequency (normalized to system frequency) results in a 100% change in power output of the unit. Figure 3.9 provides an illustration of droop for a single generator.

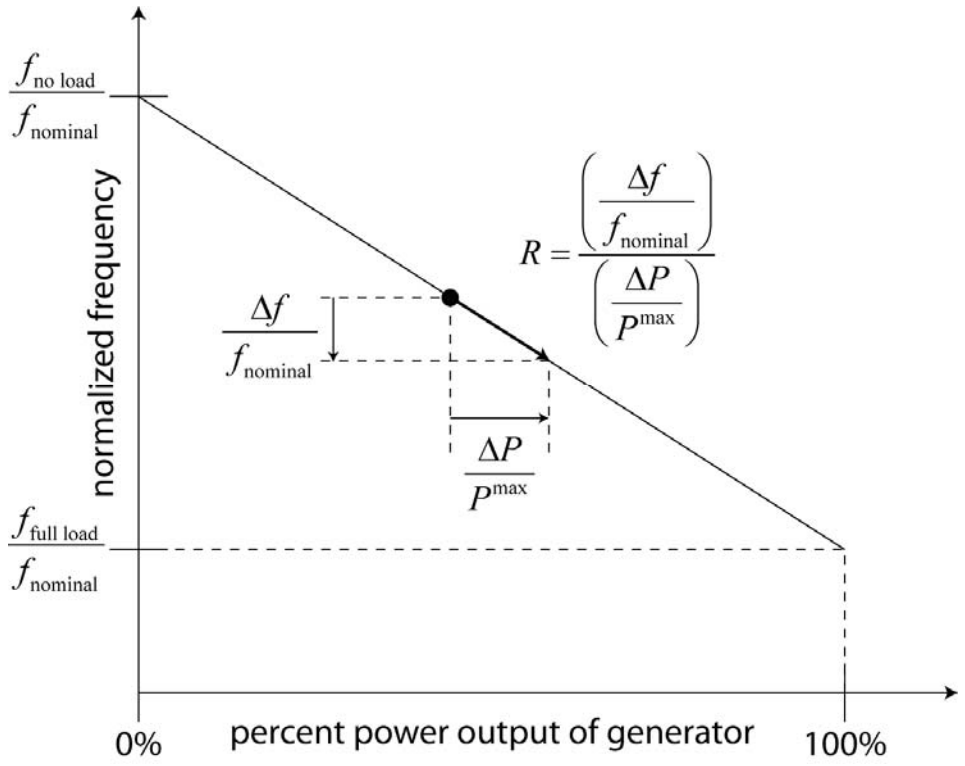

Figure 3.9: Illustration of droop characteristics.

Whenever there is a frequency change on the system, this frequency change will propagate throughout the electrical network and affect the output of each generator which is still connected to the system based on droop settings. If each generator *i* that is connected to the system has a droop value of R_i and maximum power output of $P_i^{\max}$, the following relations must hold:

$$R_1 \frac{\Delta P_1}{P_1^{\max}} = R_2 \frac{\Delta P_2}{P_2^{\max}} = \cdots = R_G \frac{\Delta P_G}{P_G^{\max}} = \frac{\Delta f}{f_{\text{nominal}}} \tag{3.22}$$

after the new system frequency has been reached. The change in frequencies, Δf, is measured at each PMU and is provided in the PMU data set; however, a priori knowledge of which generators are connected to the system is required to determine ΔP_i via (3.22). To calculate ΔP_i if generator *g* has lost connectivity, (3.20) and (3.22) can be combined:

$$
\begin{aligned}
&R_i^{'} = \frac{R_i}{P_i^{\max}} \\
&R_1^{'} \times pf_1 \times P_g = \cdots = R_G^{'} \times pf_G \times P_g \\
&R_1^{'} \times pf_1 = \cdots = R_{g-1}^{'} \times pf_{g-1} = R_{g+1}^{'} \times pf_{g+1} = \cdots = R_G^{'} \times pf_G \\
&\sum_{i \in \{1,2,\ldots,G\} \setminus g} pf_i = 1
\end{aligned}
\tag{3.23}
$$

This set of equations is solvable for the participation factors using the following set of linear equations:

$$
\begin{bmatrix}
R_1^{'} & -R_2^{'} & 0 & \cdots & \cdots & 0 \\
0 & R_2^{'} & -R_3^{'} & 0 & \cdots & 0 \\
0 & 0 & \ddots & \ddots & 0 & \vdots \\
\vdots & \cdots & 0 & \ddots & \ddots & 0 \\
0 & \cdots & \cdots & 0 & R_{G-1}^{'} & -R_G^{'} \\
1 & \cdots & \cdots & \cdots & 1 & 1
\end{bmatrix}
\begin{bmatrix} pf_1 \\ pf_2 \\ \vdots \\ \vdots \\ pf_{G-1} \\ pf_G \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}
\tag{3.24}
$$

where $\Delta P_i = pf_i \times P_g$

The participation factors obtained via (3.24) can then be used with (1.1), (3.7), and (3.20) to estimate the changes in angles for the outage of generator *g* providing pre-outage power P_g:

$$
\begin{aligned}
\Delta \boldsymbol{\theta}_{calc,g}^{P_g} &= \mathbf{K}\mathbf{B}^{-1}
\begin{bmatrix}
pf_1 \times P_g \\ \vdots \\ pf_{g-1} \times P_g \\ -P_g \\ pf_{g+1} \times P_g \\ \vdots \\ pf_G \times P_g
\end{bmatrix} \\
&= P_g \mathbf{K}\mathbf{B}^{-1}
\begin{bmatrix}
pf_1 \\ \vdots \\ pf_{g-1} \\ -1 \\ pf_{g+1} \\ \vdots \\ pf_G
\end{bmatrix} \\
&= P_g \Delta \boldsymbol{\theta}_{calc,g}
\end{aligned}
\tag{3.25}
$$

As with the single-line outage events, the expected changes in angles due to a generator outage can be represented as a scalar-vector multiplication. As a result, similar analysis provides the optimizing P_g value for a given observed change in angles:

$$P_g^* = \frac{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\boldsymbol{\theta}_{calc,g}}{\Delta\boldsymbol{\theta}_{calc,g} \cdot \Delta\boldsymbol{\theta}_{calc,g}}$$
$$\min_{P_g} \left\| \Delta\boldsymbol{\theta}_{observed} - P_g \Delta\boldsymbol{\theta}_{calc,g} \right\| = \left\| \Delta\boldsymbol{\theta}_{observed} - P_g^* \Delta\boldsymbol{\theta}_{calc,g} \right\| \tag{3.26}$$

In addition, an *NAD* value can also be defined for generator outages:

$$\theta_g = \frac{\cos^{-1}\left| \left(\frac{\Delta\boldsymbol{\theta}_{observed}}{\Delta\boldsymbol{\theta}_{observed} \cdot \Delta\boldsymbol{\theta}_{observed}} \right) \cdot \left(\frac{\Delta\boldsymbol{\theta}_{calc,g}}{\Delta\boldsymbol{\theta}_{calc,g} \cdot \Delta\boldsymbol{\theta}_{calc,g}} \right) \right|}{2}$$
$$NAD_g = 2\sin\theta_g \tag{3.27}$$

3.3.2 Verification of droop-based participation factors

To verify the usefulness of droop-based participation factors, generator outages were run for each of the nine generators within the system described in 0. Each generator uses an IEEEG1 model [69] with K_G set to 20 (i.e, droop set to 5%). Using these values, the differences between the participation factors estimated from Equation (3.24) and calculated from dynamic simulation were calculated for each outage at each sample time *t*, where *t* = 0 corresponds to the time of the outage event:

$$\Omega_i = \begin{cases} 1 & ,i \neq \text{outaged gen} \\ 0 & ,i = \text{outaged gen} \end{cases}$$

$$\Delta P_i^{\text{simulated}}(t) = \Omega_i \times \Delta P_i(t) \text{ simulated}$$

$$\Delta P_{\text{total}}^{\text{simulated}}(t) = \sum_{i=\{1,\ldots,G\}} \Delta P_i^{\text{simulated}}(t)$$

$$\mathbf{pf}_{\text{simulated}}(t) = \left[\frac{\Delta P_1^{\text{simulated}}(t)}{\Delta P_{\text{total}}^{\text{simulated}}(t)} \quad \cdots \quad \frac{\Delta P_G^{\text{simulated}}(t)}{\Delta P_{\text{total}}^{\text{simulated}}(t)} \right]^T \tag{3.28}$$

$$pf_i^{\text{estimate}} = \Omega_i \times pf_i \text{ , estimated participation factor using equation 4.21}$$

$$\mathbf{pf}_{\text{estimate}} = \left[pf_1^{\text{estimate}} \quad \cdots \quad pf_G^{\text{estimate}} \right]^T$$

$$PFD(t) = \frac{\left\| \mathbf{pf}_{\text{simulated}}(t) - \mathbf{pf}_{\text{estimate}} \right\|}{\left\| \mathbf{pf}_{\text{simulated}}(t) \right\|} \times 100\%$$

The droop-based participation factors require the change in frequency seen by each generator to be the same. To quantify how well the frequencies on the system match this condition, a frequency deviation quantity is defined:

$$f_{avg}(t) = \frac{\sum_{g \in \{1,\ldots,G\}} \Omega_g \Delta f_g(t)}{\sum_{g \in \{1,\ldots,G\}} \Omega_g}$$

$$f_{dev,g}(t) = \Omega_g \left(\Delta f_g(t) - f_{avg}(t) \right) \tag{3.29}$$

$$f_{diff}(t) = \left\| \begin{bmatrix} f_{dev,1}(t) \\ \vdots \\ f_{dev,G}(t) \end{bmatrix} \right\|$$

where Δf_g is the change in frequency from the pre-outage frequency at the bus generator g is connected to. Figure 3.10 shows that the frequency difference between the generators takes between 10 and 20 s to reach zero, which implies that the condition given in (3.22) will only be reached 10 to 20 s after the outage occurs.

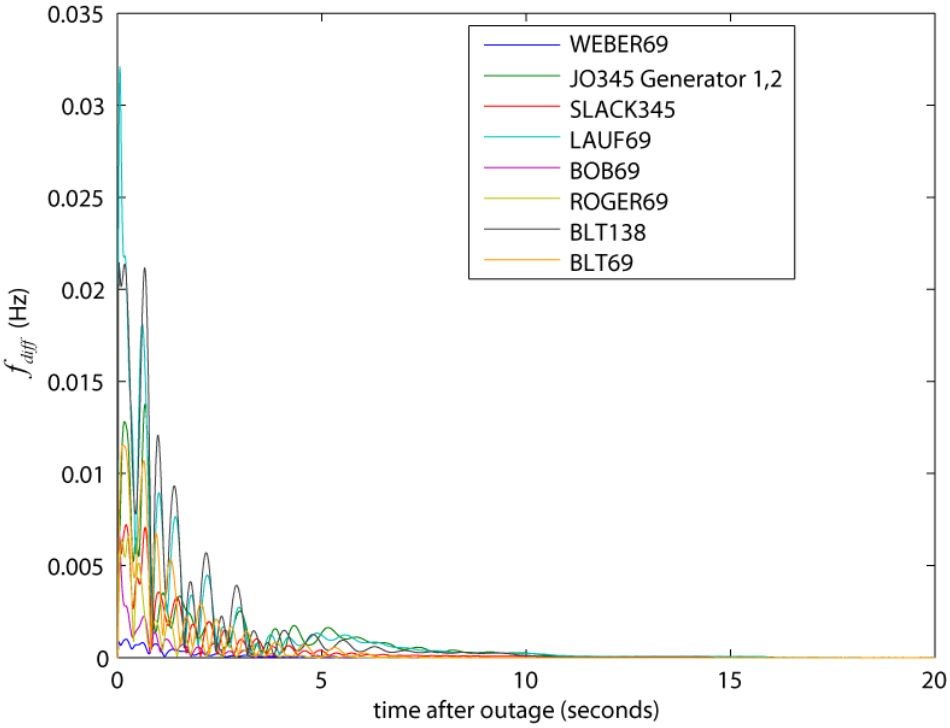

Figure 3.10: Evolution of frequency difference f_{diff} over time for each generator outage.

The *PFD* values, which reflect the deviation between the participation factors based on droop and the true participation factors, reach zero several seconds after the frequency deviations reach zero. If the droop response were instantaneous, then *PFD* would decay along with f_{diff}; however, because there are delays in regulation due to machine inertia and governor delay, there is a lag of several seconds between the frequency deviations going to zero and the *PFD* value going to zero. Comparing Figures 3.10 and 3.11, it is clear that regulation delay results in the *PFD* values for each outage reaching a minimum several seconds after the f_{diff} value reaches zero.

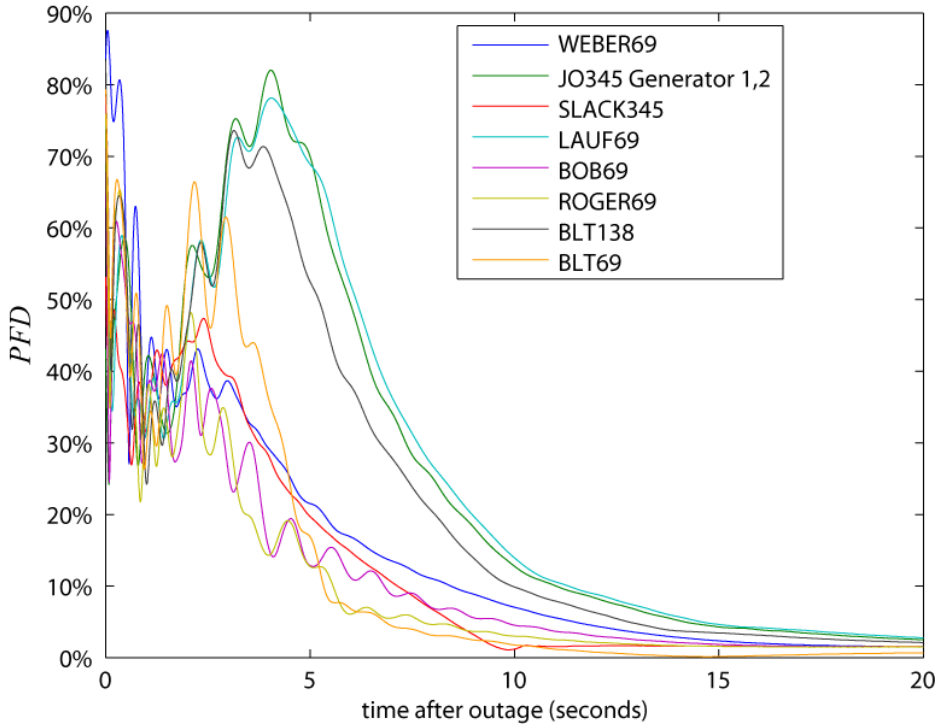

Figure 3.11: Evolution of *PFD* values over time for each generator outage.

The *PFD* values for each generator outage are provided in Table 3.1 using frequency and power output values 20 s after the outage event (i.e., at the rightmost point in Figure 3.11). The results shown in Table 3.1 are promising in that they indicate droop-based participation factors can provide a very close approximation (<2% error) to the true participation factors.

Table 3.1: Accuracy of droop-based participation factors using model parameters

Generator Outaged	WEBER69	JO345	SLACK345	LAUF69	BOB69	ROGER69	BLT138	BLT69
PFD	1.60%	1.67%	1.48%	1.72%	1.60%	1.59%	1.56%	1.00%

For each generator outage, the actual droop can be determined using (3.21) based on the change in frequency and power output at each generator. The mean and standard deviation (taken over the set of nine generator outages) in the simulated droop values for each of the generators are provided in Table 3.2. Although the values are different than the 5% value specified in the generator governor models, the

low standard deviation indicates that determining droop values through offline simulation is a viable means of correctly assessing the droop. Using the droop values in Table 3.2 instead of the 5% value specified in the governor models, the *PFD* values after 20 s were evaluated again.

Table 3.2: Droop means and standard deviations determined from simulation

Generator:	WEBER69	JO345	SLACK345	LAUF69	BOB69	ROGER69	BLT138	BLT69
Droop mean	4.93%	4.81%	4.91%	4.84%	4.94%	4.91%	4.81%	4.64%
Droop st. dev.	0.02%	0.02%	0.01%	0.01%	0.01%	0.02%	0.01%	0.01%

The *PFD* determined using the droop values of Table 3.2 are provided in Table 3.3. The usage of the droop values from Table 3.2 provides a more accurate representation of the generator response, as evidenced by the decrease in the *PFD* values for each of the generator outages.

Table 3.3: Accuracy of droop-based participation factors using Table 3.2 droop values

Generator Outaged	WEBER69	JO345	SLACK345	LAUF69	BOB69	ROGER69	BLT138	BLT69
PFD	0.36%	0.06%	0.08%	0.07%	0.13%	0.06%	0.13%	0.10%

Additional tests were run with the droop value for the generator at bus WEBER69 set to 10% rather than 5% to test the sensitivity of this method with respect to individual generator droop settings. Table 3.4 provides the droop means and standard deviations determined from simulation. The droop values for all buses except for WEBER69 stay the same as in Table 3.2, while the WEBER69 droop is approximately 10% as specified in the model of the governor at this generator.

Table 3.4: Droop means and standard deviations determined from simulation, with WEBER69 droop set to 10% instead of 5%

Generator:	WEBER69	JO345	SLACK345	LAUF69	BOB69	ROGER69	BLT138	BLT69
Droop mean	9.74%	4.81%	4.91%	4.84%	4.94%	4.91%	4.81%	4.64%
Droop st. dev.	0.06%	0.02%	0.01%	0.01%	0.01%	0.02%	0.02%	0.01%

Using the droop values in Table 3.4, participation factors were calculated using (3.24) and the PFD values given in Table 3.5 were determined 20 s after the outage. These low error values demonstrate that droop-based participation factors are effective in modeling generator response independently of the droop value, as long as sufficient time is allowed for the generators to reach the operating points specified by their respective droop settings.

Table 3.5: : Accuracy of droop-based participation factors using Table 3.4 droop values, with WEBER69 droop set to 10% instead of 5%

Generator Outaged	WEBER69	JO345	SLACK345	LAUF69	BOB69	ROGER69	BLT138	BLT69
PFD	0.39%	0.07%	0.13%	0.05%	0.14%	0.05%	0.14%	0.11%

3.3.3 Basic generator outage detection algorithm

Because the expected angle changes for a single-line outage and a generator outage can both be expressed as scalar-vector multiplications, the algorithms are very similar:

1. For each generator *g*:
    a. Calculate $\Delta\boldsymbol{\theta}_{calc,g}$ using (3.25).
    b. Calculate P_g^* using (3.26).
    c. Calculate the *NAD* value for generator *g* using (3.27), then store the calculated error value in the indexed array *NADVals*:

$$NADVals_g = NAD_g \tag{3.30}$$

2. Determine the generator g^* that was outaged by sorting *NADVals*:

$$g^* = \arg\min_g NADVals_g \tag{3.31}$$

3. Report back the estimated pre-outage generation $P_{g^*}^*$ calculated in step 1.b.

3.3.4 Computational complexity

Although the methods used to determine the most likely single line outage and generator outage events are very similar, there are two key differences. The first difference is in the density of the ΔP vector used to calculate $\Delta\boldsymbol{\theta}_{calc,g}$. In the single-line outage case, the ΔP vector has only two nonzero entries corresponding to the from and to buses of the outaged line. With generator outages, the ΔP vector will have nonzero entries at each bus with an online generator attached. This is typically a small subset of the total number of system buses, but may result in increased computation time.

The second key difference is that there are fewer generators than lines within a power system, so the number of events is drastically reduced. This reduction in events causes fewer calculations of $\Delta\boldsymbol{\theta}_{calc}$ and NAD values and a faster sort operation in determining the most likely generator outage.

3.3.5 Heuristic modifications to the basic algorithm

By taking advantage of known generator ratings, it is possible to further restrict the event set. The way this is handled in the algorithm is to remove generators from consideration if the estimated pre-outage generation is greater than 1.5 times the rating of the generation. The extra 50% leeway allows for possible errors introduced by the dc power flow assumptions, participation factor selection, and errors in determining $\Delta\theta_{observed}$.

3.4 Double-Line Outage Detection

3.4.1 Analytical basis for double-line outage detection

When $\mathcal{E}$ is restricted to the set of double-line outages on the system, then the problem defined in (1.3) becomes

$$\text{lines outaged } \left\{l_1^*, l_2^*\right\} = \underset{\{l_1,l_2\}\in\{1,2,\ldots,L\}\times\{1,2,\ldots,L\}}{\arg\min}\left(\min_{P_{l,1},P_{l,2}}\left\|\Delta\boldsymbol{\theta}_{observed} - deltaAngles_{l_1,l_2}\left(P_{l,1},P_{l,2}\right)\right\|\right) \tag{3.32}$$

where L is the number of lines in service before the event is detected and $deltaAngles_{l_1,l_2}\left(P_{l,1},P_{l,2}\right)$ is a function which returns the estimated change in angles for the outages of lines l_1 and l_2 with pre-outage flows of $P_{l,1}$ and $P_{l,2}$, respectively. Because the pre-outage flows are unknown a priori, each is allowed to vary in order to achieve the best match in observed and calculated angles.

To characterize the function $deltaAngles_{l_1,l_2}\left(P_{l,1},P_{l,2}\right)$ under the dc power flow assumptions, the outages are modeled by power injections at the terminal buses of the outaged lines. Figure 3.12 illustrates the relevant quantities used to model a double-line outage. Based on the flows and injections shown in Figure 3.12, the following two equalities must hold for the flow from the rest of the system to the outaged lines, F_1 and F_2, to be zero:

$$\begin{bmatrix}\tilde{\tilde{P}}_{l,1}\\ \tilde{\tilde{P}}_{l,2}\end{bmatrix} = \begin{bmatrix}\left(\begin{array}{c}P_{l,1}+PTDF_{l_1,l_{1,from}-l_{1,to}}\times\tilde{\tilde{P}}_{l,1}+\\ PTDF_{l_1,l_{2,from}-l_{2,to}}\times\tilde{\tilde{P}}_{l,2}\end{array}\right)\\ \left(\begin{array}{c}P_{l,2}+PTDF_{l_2,l_{1,from}-l_{1,to}}\times\tilde{\tilde{P}}_{l,1}+\\ PTDF_{l_2,l_{2,from}-l_{2,to}}\times\tilde{\tilde{P}}_{l,2}\end{array}\right)\end{bmatrix} \tag{3.33}$$

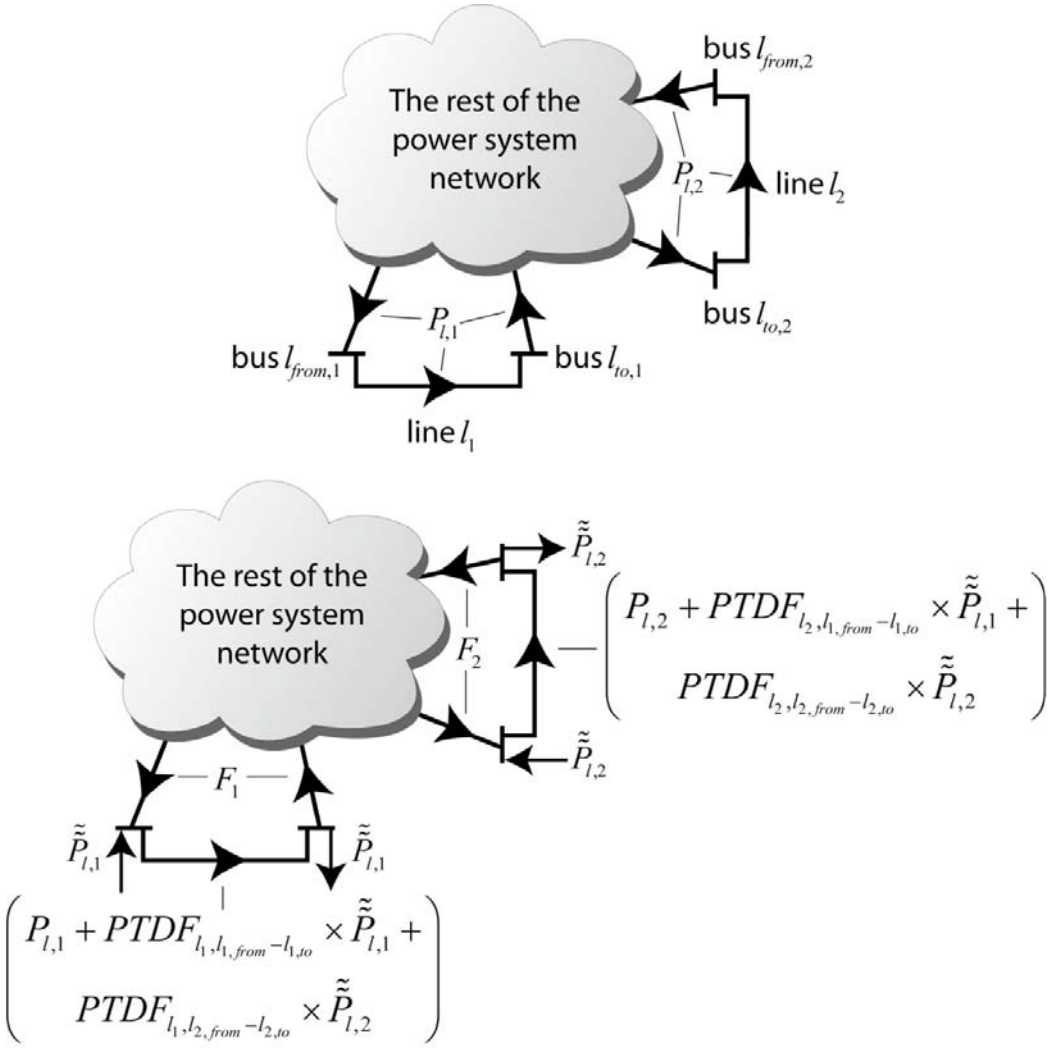

Figure 3.12: Double-line outage model with power injections.

Rewriting this in matrix form, an expression for $\tilde{\tilde{P}}_{l,1}$ and $\tilde{\tilde{P}}_{l,2}$ in terms of PTDFs and pre-outage line flows can be obtained:

$$\begin{bmatrix} \tilde{\tilde{P}}_{l,1} \\ \tilde{\tilde{P}}_{l,2} \end{bmatrix} = \begin{bmatrix} 1 - PTDF_{l_1,l_{1,from}-l_{1,to}} & -PTDF_{l_1,l_{2,from}-l_{2,to}} \\ -PTDF_{l_2,l_{1,from}-l_{1,to}} & 1 - PTDF_{l_2,l_{2,from}-l_{2,to}} \end{bmatrix}^{-1} \begin{bmatrix} P_{l,1} \\ P_{l,2} \end{bmatrix} \qquad (3.34)$$

The inverse matrix in (3.34) exists only if the outage of the two lines does not result in islanding [70]. If the lines do result in islanding of the system, then the outages cannot be modeled as transfers across the lines and must instead be modeled as

changes in power injection at the boundary buses of the original system, taking into account any necessary redispatch of generation as discussed in Section 3.3.1.

Using the formula for the inverse of a 2 x 2 matrix, the solution in (3.34) can be rewritten:

$$\begin{aligned}
\alpha_{1,2} &= \left(1 - PTDF_{l_1,l_{1,from}-l_{1,to}}\right)\left(1 - PTDF_{l_2,l_{2,from}-l_{2,to}}\right) - \\
&\quad \left(PTDF_{l_2,l_{1,from}-l_{1,to}}\right)\left(PTDF_{l_1,l_{2,from}-l_{2,to}}\right) \\
\begin{bmatrix} \tilde{\tilde{P}}_{l,1} \\ \tilde{\tilde{P}}_{l,2} \end{bmatrix} &= \frac{1}{\alpha_{1,2}} \begin{bmatrix} 1 - PTDF_{l_2,l_{2,from}-l_{2,to}} & PTDF_{l_1,l_{2,from}-l_{2,to}} \\ PTDF_{l_2,l_{1,from}-l_{1,to}} & 1 - PTDF_{l_1,l_{1,from}-l_{1,to}} \end{bmatrix} \begin{bmatrix} P_{l,1} \\ P_{l,2} \end{bmatrix} \\
\begin{bmatrix} \tilde{\tilde{P}}_{l,1} \\ \tilde{\tilde{P}}_{l,2} \end{bmatrix} &= \frac{P_{l,1}}{\alpha_{1,2}} \begin{bmatrix} 1 - PTDF_{l_2,l_{2,from}-l_{2,to}} \\ PTDF_{l_2,l_{1,from}-l_{1,to}} \end{bmatrix} + \frac{P_{l,2}}{\alpha_{1,2}} \begin{bmatrix} PTDF_{l_1,l_{2,from}-l_{2,to}} \\ 1 - PTDF_{l_1,l_{1,from}-l_{1,to}} \end{bmatrix}
\end{aligned} \tag{3.35}$$

Once the double outage model injection vectors, $\tilde{\tilde{P}}_{l,1}$ and $\tilde{\tilde{P}}_{l,2}$, are obtained, the resulting change in angles can then be determined:

$$\begin{aligned}
\Delta\theta_{calc,l_1,l_2}^{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} &= \mathbf{KB}^{-1}\left(\begin{bmatrix} 0 \\ \tilde{\tilde{P}}_{l,1} \\ -\tilde{\tilde{P}}_{l,1} \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow l_{from,1} \\ \leftarrow l_{to,1} \\ \end{matrix} + \begin{bmatrix} 0 \\ \tilde{\tilde{P}}_{l,2} \\ -\tilde{\tilde{P}}_{l,2} \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow l_{from,2} \\ \leftarrow l_{to,2} \\ \end{matrix}\right) \\
&= \tilde{\tilde{P}}_{l,1}\left(\mathbf{KB}^{-1}\begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow l_{from,1} \\ \leftarrow l_{to,1} \\ \end{matrix}\right) + \\
&\quad \tilde{\tilde{P}}_{l,2}\left(\mathbf{KB}^{-1}\begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow l_{from,2} \\ \leftarrow l_{to,2} \\ \end{matrix}\right) \\
&= \tilde{\tilde{P}}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} + \tilde{\tilde{P}}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}
\end{aligned} \tag{3.36}$$

Note that the $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ vectors in (3.36) are the same vectors defined in (3.8) for single-line outage modeling. The optimization of (3.32) can then be rewritten in terms of the vectors defined in (3.36):

$$\text{lines outaged } \left\{l_1^*, l_2^*\right\} = \underset{\{l_1,l_2\}\in\{1,2,\ldots,L\}\times\{1,2,\ldots,L\}}{\arg\min}\left(\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}}\left\|\Delta\boldsymbol{\theta}_{observed}-\left(\tilde{\tilde{P}}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}+\tilde{\tilde{P}}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right)\right\|\right) \tag{3.37}$$

If $\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}$, which occurs if line l_1 and l_2 are in parallel, then it is possible to easily determine the values of $\tilde{\tilde{P}}_{l,1}$ and $\tilde{\tilde{P}}_{l,2}$ which result in the best match with $\Delta\boldsymbol{\theta}_{observed}$. In this case, the following one-dimensional minimization is performed:

$$\begin{gathered}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_{1,2}} = \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} \\ \min_{\tilde{\tilde{P}}_{l,1,2}}\left\|\Delta\boldsymbol{\theta}_{observed}-\tilde{\tilde{P}}_{l,1,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_{1,2}}\right\| = \left\|\Delta\boldsymbol{\theta}_{observed}-\tilde{\tilde{P}}^*_{l,1,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_{1,2}}\right\| \\ \tilde{\tilde{P}}^*_{l,1,2} = \frac{\Delta\tilde{\boldsymbol{\theta}}_{calc,l_{1,2}}\cdot\Delta\boldsymbol{\theta}_{observed}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l_{1,2}}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_{1,2}}}\end{gathered} \tag{3.38}$$

Using the $\tilde{\tilde{P}}^*_{l,1,2}$ scaling factor, it is possible to determine the pre-outage flows on each of the parallel lines:

$$\begin{gathered}\tilde{\tilde{P}}^*_{l,1,2} = P_{l_1} + PTDF_{l_1,l_{1,from}-l_{1,to}}\times\tilde{\tilde{P}}^*_{l,1,2} + \\ P_{l_2} + PTDF_{l_2,l_{2,from}-l_{2,to}}\times\tilde{\tilde{P}}^*_{l,1,2} \\ P_{l_1}+P_{l_2} = \tilde{\tilde{P}}^*_{l,1,2}\left(1-PTDF_{l_1,l_{1,from}-l_{1,to}}-PTDF_{l_2,l_{2,from}-l_{2,to}}\right) \\ \frac{P_{l_1}}{P_{l_2}} = \frac{B_{l_1}}{B_{l_2}} = \frac{PTDF_{l_1,l_{1,from}-l_{1,to}}}{PTDF_{l_2,l_{2,from}-l_{2,to}}} \\ \begin{bmatrix}P_{l_1}\\P_{l_2}\end{bmatrix} = \tilde{\tilde{P}}^*_{l,1,2}\frac{\left(1-PTDF_{l_1,l_{1,from}-l_{1,to}}-PTDF_{l_2,l_{2,from}-l_{2,to}}\right)}{PTDF_{l_1,l_{1,from}-l_{1,to}}+PTDF_{l_1,l_{1,from}-l_{1,to}}}\begin{bmatrix}PTDF_{l_1,l_{1,from}-l_{1,to}}\\PTDF_{l_2,l_{2,from}-l_{2,to}}\end{bmatrix}\end{gathered} \tag{3.39}$$

The last expression is based on the fact that the same angle difference exists across both lines, so the flows must be in proportion to their reactances. Figure 3.13 provides a graphical depiction of the relevant quantities used to derive the equations in (3.39) which model a parallel line outage with power injections.

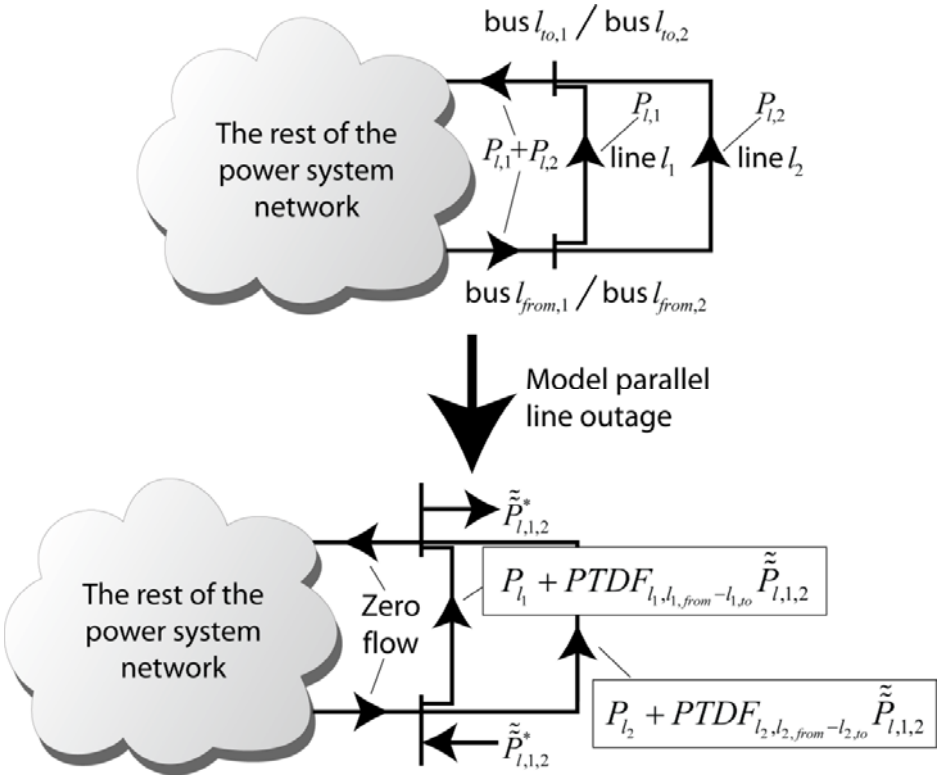

Figure 3.13: Modeling the outage of two parallel lines with a power transfer.

If the outage of the parallel lines results in island formation, then $\left(1 - PTDF_{l_1,l_{1,from}-l_{1,to}} - PTDF_{l_2,l_{2,from}-l_{2,to}}\right)$ will be equal to zero and modeling of the double line outage must account for changes in generation due to islanding.

For cases where the lines are not in parallel, the minimization in (3.37) can be solved as a least squares minimization:

$$\begin{bmatrix}\tilde{\tilde{P}}^{*}_{l,1}\\ \tilde{\tilde{P}}^{*}_{l,2}\end{bmatrix}=\underset{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}}{\arg\min}\left\|\Delta\boldsymbol{\theta}_{observed}-\begin{bmatrix}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} & \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\end{bmatrix}\begin{bmatrix}\tilde{\tilde{P}}_{l,1}\\ \tilde{\tilde{P}}_{l,2}\end{bmatrix}\right\|$$

$$\begin{bmatrix}\tilde{\tilde{P}}^{*}_{l,1}\\ \tilde{\tilde{P}}^{*}_{l,2}\end{bmatrix}=\left(\begin{bmatrix}\left(\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\right)^{T}\\ \left(\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right)^{T}\end{bmatrix}\begin{bmatrix}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} & \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\end{bmatrix}\right)^{-1}\begin{bmatrix}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\boldsymbol{\theta}_{observed}\\ \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\cdot\Delta\boldsymbol{\theta}_{observed}\end{bmatrix}$$

$$=\begin{bmatrix}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} & \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\\ \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} & \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\end{bmatrix}^{-1}\begin{bmatrix}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\boldsymbol{\theta}_{observed}\\ \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\cdot\Delta\boldsymbol{\theta}_{observed}\end{bmatrix} \quad (3.40)$$

$$=\frac{1}{\left\|\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\right\|^{2}\left\|\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right\|^{2}-\left(\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right)^{2}}\times$$

$$\begin{bmatrix}\left\|\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right\|^{2} & -\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\\ -\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} & \left\|\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\right\|^{2}\end{bmatrix}\begin{bmatrix}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}\cdot\Delta\boldsymbol{\theta}_{observed}\\ \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\cdot\Delta\boldsymbol{\theta}_{observed}\end{bmatrix}$$

Using the optimal injections defined in (3.40), the optimization of (3.37) can be simplified further:

$$\text{lines outaged } \left\{l_1^{*},l_2^{*}\right\}=\underset{\{l_1,l_2\}\in\{1,2,\ldots,L\}\times\{1,2,\ldots,L\}}{\arg\min}\left(\left\|\Delta\boldsymbol{\theta}_{observed}-\left(\tilde{\tilde{P}}^{*}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}+\tilde{\tilde{P}}^{*}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right)\right\|\right) \quad (3.41)$$

The normalized angle distance (*NAD)* metric can also be extended to double-line outage events as follows:

$$\theta_{l_1,l_2}=\cos^{-1}\left(\frac{\Delta\boldsymbol{\theta}_{observed}}{\left\|\Delta\boldsymbol{\theta}_{observed}\right\|}\cdot\frac{\left(\tilde{\tilde{P}}^{*}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}+\tilde{\tilde{P}}^{*}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right)}{\left\|\left(\tilde{\tilde{P}}^{*}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}+\tilde{\tilde{P}}^{*}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}\right)\right\|}\right) \quad (3.42)$$

$$NAD_{l_1,l_2}=2\sin\theta_{l_1,l_2}$$

With the error metric properly defined, the algorithm can be defined for double-line outage event detection, which closely aligns with those for single-line and generator outage events.

3.4.2 Basic double-line outage event definition algorithm

In order to detect double-line outage events, identify the outaged lines, and determine the pre-outage flows on the lines, the following basic algorithm is used:

1. For each line $l_1 \in [1, L]$
    a. Calculate $\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}$ using (3.36).
    b. For each line $l_2 \in [l_1 + 1, L]$
        i. Calculate $\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}$ using (3.36).
        ii. Calculate $\tilde{\tilde{P}}^*_{l,1}$ and $\tilde{\tilde{P}}^*_{l,2}$ using (3.40) for nonparallel lines or $\tilde{\tilde{P}}^*_{l,1,2}$ using (3.38) for parallel lines.
        iii. Calculate the *NAD* value for the double outage $\{l_1, l_2\}$ using (3.42), then store the calculated error value in the indexed array *NADVals*:

$$NADVals_{\{l_1,l_2\}} = NAD_{l_1,l_2} \tag{3.43}$$

2. Determine the lines $\{l_1^*, l_2^*\}$ that were outaged by sorting *NADVals*:

$$\{l_1^*, l_2^*\} = \arg\min_{l_1,l_2} NADVals_{\{l_1,l_2\}} \tag{3.44}$$

3. Determine the pre-outage flows on the lines which correspond to the optimal double outage model injections using (or (3.39) for parallel lines):

$$\begin{bmatrix} P^*_{l,1} \\ P^*_{l,2} \end{bmatrix} = \begin{bmatrix} 1 - PTDF_{l_1,l_{1,from}-l_{1,to}} & -PTDF_{l_1,l_{2,from}-l_{2,to}} \\ -PTDF_{l_2,l_{1,from}-l_{1,to}} & 1 - PTDF_{l_2,l_{2,from}-l_{2,to}} \end{bmatrix} \begin{bmatrix} \tilde{\tilde{P}}^*_{l,1} \\ \tilde{\tilde{P}}^*_{l,2} \end{bmatrix} \tag{3.45}$$

3.4.3 Computational complexity

The extension of events from single- to double-line outages can greatly increase both storage and computational demands. The key difference between evaluating

single- and double-line outages is the number of potential events. For single-line outage events, only L events need to be considered, if L is the number of lines which are currently in service. On the other hand, for double-line outages, the number of considered events is $\binom{L}{2}$, or $\left[L\left(L-1\right)\right]/2$. For a typical system, with lines numbering in the tens of thousands, this can represent a tremendous increase in the amount of work needed to find the minimizing event.

3.4.4 Heuristic modifications to the basic algorithm

There are several ways to reduce the dimensionality of the search space, with a tradeoff between consideration of improbable events and reduction in storage and computational burden. One method is to filter the event set to only include outages with probability above a certain threshold $\text{Prob}_{threshold}$:

$$\text{lines outaged } \left\{l_1^*, l_2^*\right\} = \underset{\{l_1,l_2\}\in\{1,2,\ldots,L\}\times\{1,2,\ldots,L\},\text{Prob}(\{l_1,l_2\})>\text{Prob}_{threshold}}{\arg\min} \left(\cdots\right) \quad (3.46)$$

The probability data needed to execute the filtering in (3.46) should be based on detailed modeling or operational data. Unfortunately, this data is currently unavailable over the wide area; however, the need for this data to improve event identification could serve as a motivating factor to improve the collection and distribution of outage statistics.

A second way of filtering the event set is to account for the geographic proximity of two lines on the system. Simultaneous line outages have a tendency to occur in shared rights of way [71], and the wide availability of line locations makes this option more feasible than (3.46). The basic formulation of this filtering method is

$$\text{lines outaged } \left\{l_1^*, l_2^*\right\} = \underset{\{l_1,l_2\}\in\{1,2,\ldots,L\}\times\{1,2,\ldots,L\}, d(l_1,l_2)<d_{threshold}}{\arg\min} \left(\cdots\right) \tag{3.47}$$

where a distance function is defined between any two lines, $d\left(l_1, l_2\right)$, and this distance must be below a threshold value of $d_{threshold}$ for the two lines to be considered together. Because line locations rarely change at the transmission level, the determination of $d\left(l_1, l_2\right)$ for each pairwise combination of lines can be stored and used for long periods of time.

One simple but conservative distance metric that can be used for $d\left(l_1, l_2\right)$ is

$$\begin{aligned} \left(x_{l_i}, y_{l_i}\right) &= \underset{(\bar{x}_i,\bar{y}_i)}{\arg\min}\left\{R_i\left(\bar{x}_i, \bar{y}_i\right) = \underset{\text{all points } (x,y)\in\text{line } i}{\max}\left\|\left(\bar{x}_i, \bar{y}_i\right) - \left(x, y\right)\right\|\right\} \\ d_{circ}\left(l_1, l_2\right) &= \max\left\{\begin{matrix}\left\|\left(x_{l_1}, y_{l_1}\right) - \left(x_{l_2}, y_{l_2}\right)\right\| - R_1\left(x_{l_1}, y_{l_1}\right) - R_2\left(x_{l_2}, y_{l_2}\right), \\ 0\end{matrix}\right\} \end{aligned} \tag{3.48}$$

A visualization of this distance function is provided in Figure 3.14. This formulation provides a lower bound on the pointwise distance between any two points in either line. The first equation in (3.48) represents the solution of the minimum enclosing circle (MEC) problem, with the line vertices as the points to be enclosed. This is a well-studied problem, and algorithms exist which can calculate $\left(x_{l_i}, y_{l_i}\right)$ and the associated radius in $\mathrm{O}\left(n_i\right)$ time, where n_i is the number of vertex points for line i [72]. Pairwise evaluation of these distances can be done very quickly once the $\left(x_{l_i}, y_{l_i}\right)$ and R_i values are known, with each distance calculation requiring only three add and two multiply operations.

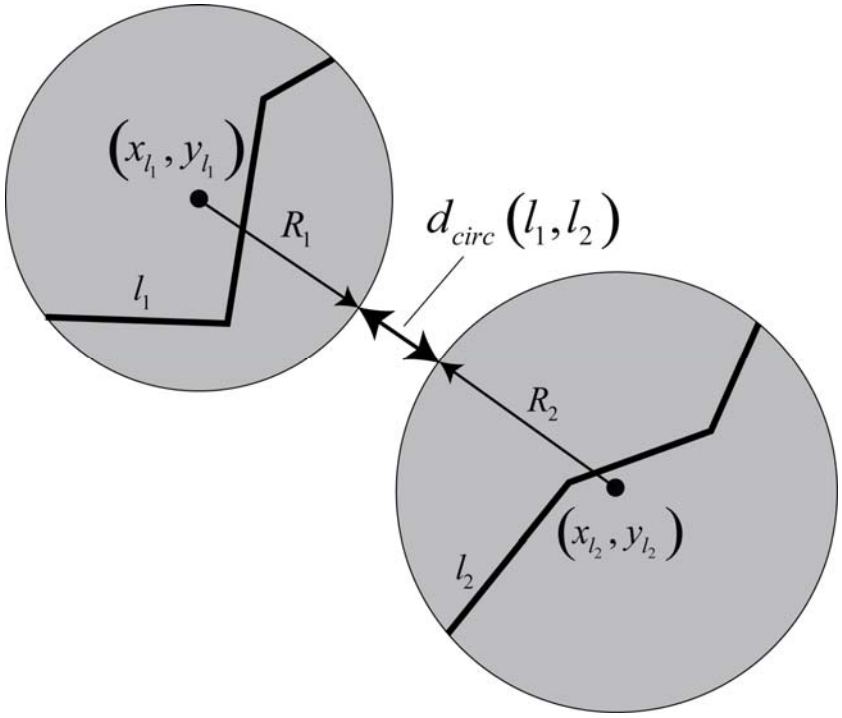

Figure 3.14: Distance between two lines, as defined in (3.48).

An alternative method of defining the distance between two lines is to use the minimum distance between any two points on the lines:

$$d_{min}(l_1, l_2) = \min_{\substack{\text{all points } (x_1, y_1) \in \text{line } 1, \\ \text{all points } (x_2, y_2) \in \text{line } 2}} \left\| (x_1, y_1) - (x_2, y_2) \right\| \tag{3.49}$$

This distance function is illustrated in Figure 3.15. The relationship between d_{min} and d_{circ} is that d_{circ} is a lower bound on $d_{min}(l_1, l_2)$, since any point on a line must by definition be within the MEC of the line. The relative magnitude of d_{min} and d_{circ} depends on the line configuration, but d_{min} should be substantially higher since the MEC tends to greatly enlarge the influence region of a line, particularly for long lines.

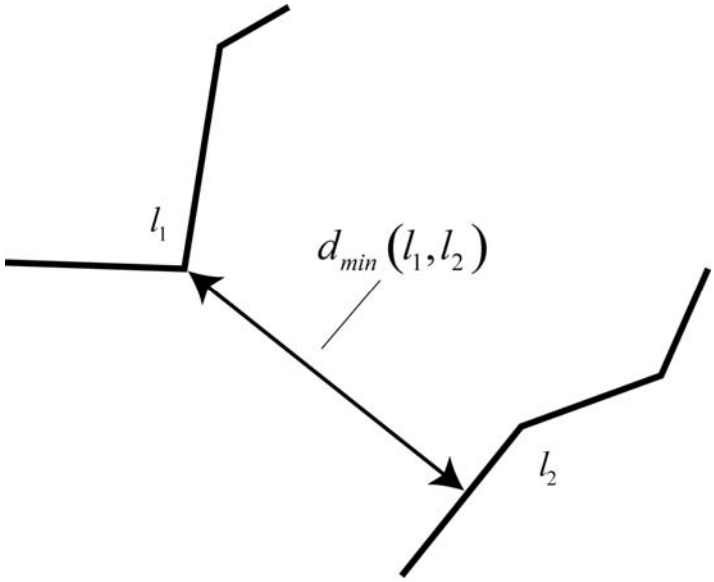

Figure 3.15: Minimum distance between two lines, as defined in (3.49).

These distances can be calculated offline and stored in a table for later retrieval. Determining what the set of considered double line outages is for a given threshold is then a matter of looking for all entries in the table below a threshold, which can be greatly accelerated by utilizing sorted lists for storage. As a result, the calculations involved in the filtering operation should be much lower than the search of the complete double outage event space and result in a net reduction in computation time and storage.

To evaluate the usefulness of the filtering method defined in (3.47) and (3.48), a large system with 4609 lines, illustrated in Figure 3.16, was used. The minimum distance between lines is zero, resulting from the case where two lines have the same terminal bus. The maximum distance between any two lines is 1422 miles using d_{circ} and 1429 miles using d_{min}, and the mean distance is 342 miles using d_{circ} and 347 miles using d_{min}.

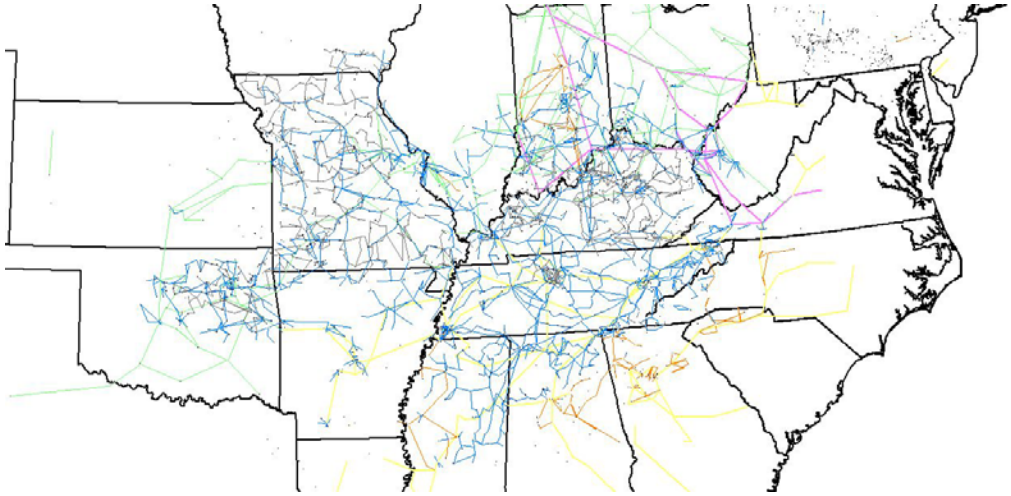

Figure 3.16: System used to test the impact of distance-based double-outage filtering.

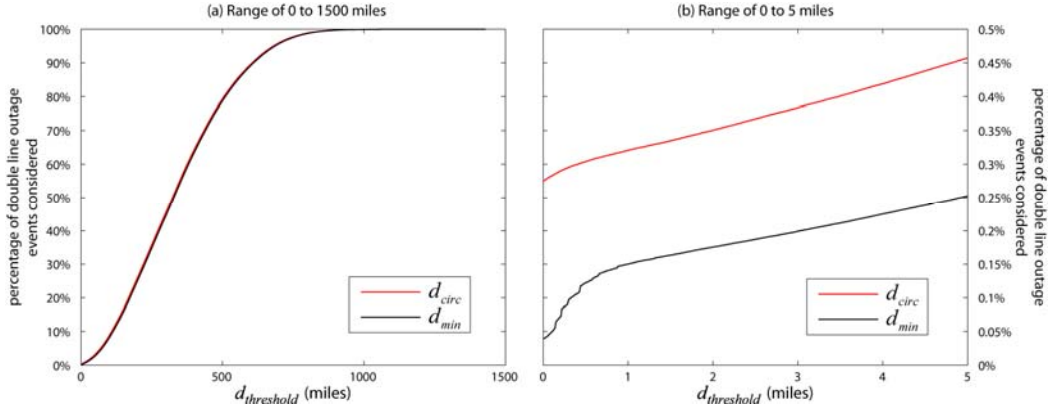

Figure 3.17: Percentage of double-outage events considered as a function of the distance threshold $d_{threshold}$ using d_{circ} and d_{min} for threshold ranges of 0-1500 miles (a) and 0-5 miles (b).

Figure 3.17 illustrates how the percentage of double-line outage events filtered by the distance function varies with the distance cutoff values ranging from 0 to 1500 miles on the left side and 0 to 5 miles on the right side. These results indicate that even a conservative threshold of 5 miles removes over 99.75% of the double-outage events from consideration if the larger distance d_{min} is used. For real-time implementation, Figure 3.17 could be used to find the distance threshold corresponding to a defined number of events (e.g., if a maximum search time is

specified) or the number of events corresponding to a defined distance threshold (e.g., if a maximum interline distance is specified).

The other heuristic applied to double-line outage detection is the rejection of potential outages when the estimated pre-outage flow is above 150% of the line rating or 5 GW. This is the same heuristic used in single line outage detection and described above in Section 3.2.4.

4 EVENT DETECTION EVALUATION

4.1 Single-Line Outage Examples

To test the single-line outage detection algorithm, two tests were performed. The first set of tests, using the system defined in **Error! Reference source not found.**, examines the algorithm's performance for each of the 56 single line outages on the system. The second set examines the algorithm's performance using real data from the Tennessee Valley Authority system recorded during the outage of a major 500-kV transmission line.

To evaluate single-line outage detection, several performance metrics are defined. The first is an indicator function to signal whether or not a particular outage is detected by the algorithm for a given set of detection parameters:

$$AboveThreshold\left(l_{outaged}\right)=\begin{cases}1 & \text{, outage of line } l_{outaged} \text{ detected}\\ 0 & \text{, otherwise}\end{cases} \tag{4.1}$$

where $l_{outaged}$ is the outaged line. Next, a metric is defined to represent the *NAD* value of the outaged line:

$$NADOutaged\left(l_{outaged}\right)=NADVals_{l_{outaged}} \tag{4.2}$$

This metric indicates how well the expected and observed angles match up for the line that was outaged. The rank of the outaged line in the sorted *NADVals* list is also important:

$$RankOutaged\left(l_{outaged}\right)=\left|\left\{l: NADVals_l < NADVals_{l_{outaged}}\right\}\right|+1 \tag{4.3}$$

Ideally, *RankOutaged* would be one, indicating that the algorithm has correctly matched the outaged line with the observed angle changes.

Taken together, these metrics define whether a line outage is detected (*AboveThreshold*), how well the observed and expected angle changes match (*NADOutaged*), and how well the expected angles from the outaged line compare with those from other lines (*RankOutaged*).

4.1.1 All 37-bus single-line outages

The first set of tests was conducted with the system described in 0. The 56 in-service lines were each outaged and a transient simulation was run for 5 s after the outage. For the transient simulation, the default simulation parameters were used from PSS/E version 31, including the default simulation time step of 1/120 s (i.e., a sampling rate of 120 Hz). To simulate the data that would be obtained from a phasor measurement unit, the raw simulated data was downsampled to a sampling rate of 30 Hz. A low-pass antialiasing filter with a cutoff of 15 Hz and delay compensation was used to ensure that no out-of-bandwidth components were present in the simulated measurements. An example of the simulated angle measurements at each bus on the system for the outage of the line connecting buses 28 and 31 is shown in Figure 4.1.

The results of the transient simulation were then filtered using one of the methods described in Chapter 2, and the observed angle changes were determined using the method described in Chapter 3.1. These results were then processed using the algorithm defined in Sections 3.2.2 and 3.2.4 to detect the outaged line.

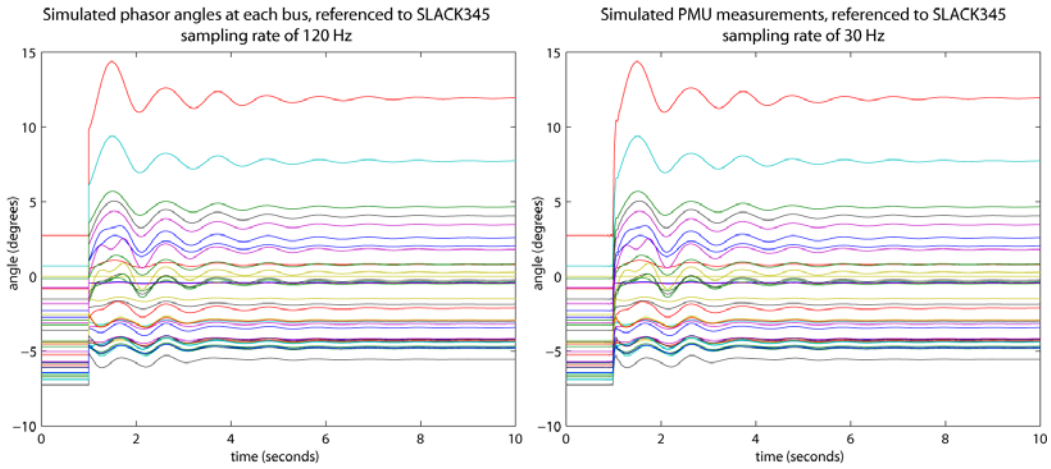

Figure 4.1: Original simulated angles (left side) and simulated PMU angle measurements (right side) for the outage of the line connecting buses 28 and 31, with all angles referenced to the SLACK345 bus angle.

For each study, one or more of the following parameters was varied:

- *FilterLength*: the length of the digital signal filter, equal to N from (2.1) (for FIR filtering) or (2.4) (for median filtering)
- τ: the threshold used to determine whether an event has occurred, defined in Section 3.1.1
- *PMUSet*: the set of bus numbers where measured angles are available

In addition, N_{trans} was set to *FilterLength* for FIR filtering and $\left\lfloor \frac{FilterLength}{2} \right\rfloor$ for median filtering based on the results shown in Figure 3.5.

4.1.1.1 All buses monitored

The first set of tests examines the effect of changing the threshold and filtering methods with full PMU coverage (i.e., $PMUSet = \{\text{all system buses}\}$). Because all buses are monitored, this test is designed to provide best case results for specific filter parameters and angle detection thresholds.

4.1.1.1.1 Median filtered angles

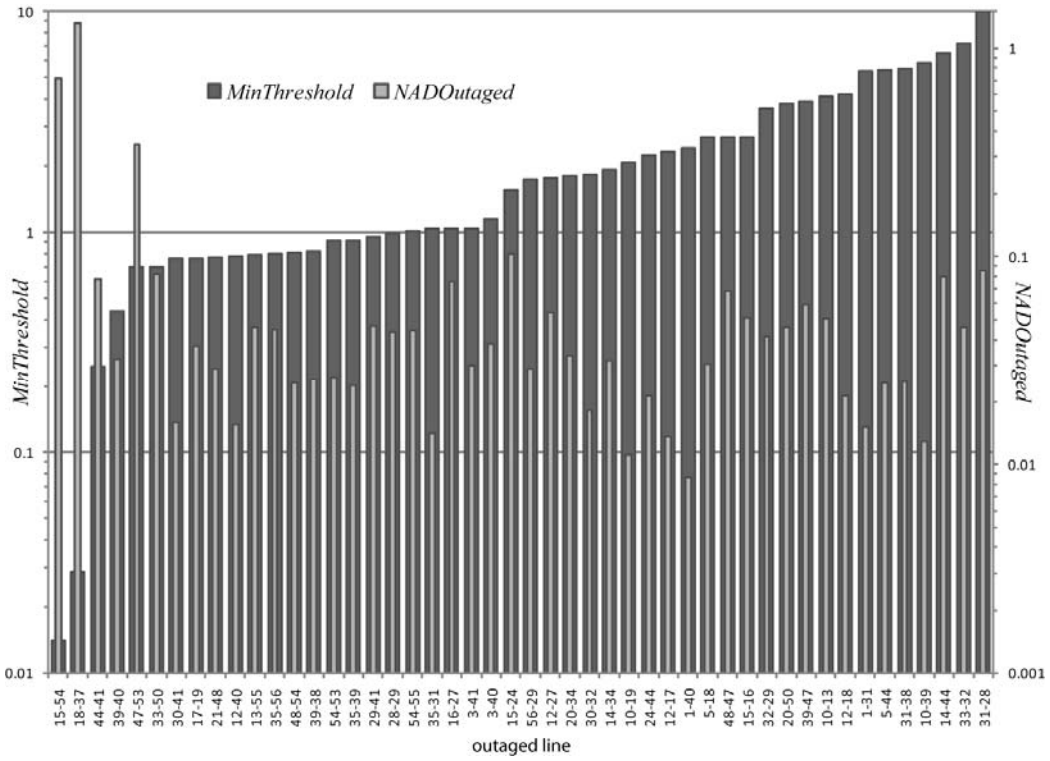

Figure 4.2: Minimum threshold values and error values for each line, ordered in ascending order by minimum threshold, using a median filter of length 31.

One of the two filtering methods discussed in Chapter 2 is median filtering. The first parameter considered is τ, which determines how much of an angle change needs to occur before it is recognized as an event. Plotted in Figure 4.2 are the *NADOutaged* and *MinThreshold* (the minimum value of τ needed to detect the event) for each line outage on the system, sorted in ascending order by the *MinThreshold* and using a window length of 31. For parallel lines, only one of the lines is provided in this graph. A log scale is used for both y-axes due to the widely varying values of *MinThreshold* and *NADOutaged* among the set of all outaged lines. Several useful insights can be obtained by analyzing this figure. First of all, the minimum threshold value needed to detect every line outage is 0.01 degrees; however,

those events which require extremely low threshold values also have the highest *NADOutaged* values, indicating that choosing the minimal threshold value will result in misclassification of these lines. To illustrate this behavior, Figure 4.3 shows the average *NADOutaged* value, taken over all line outages, for varying values of the angle threshold. The reduction in *NADOutaged* as the angle threshold is brought past 0.03 degrees, in essence ignoring the troublesome outages between buses 15-54 and 18-37, can clearly be seen in this plot and illustrates how the threshold value can impact the *NADOutaged* results.

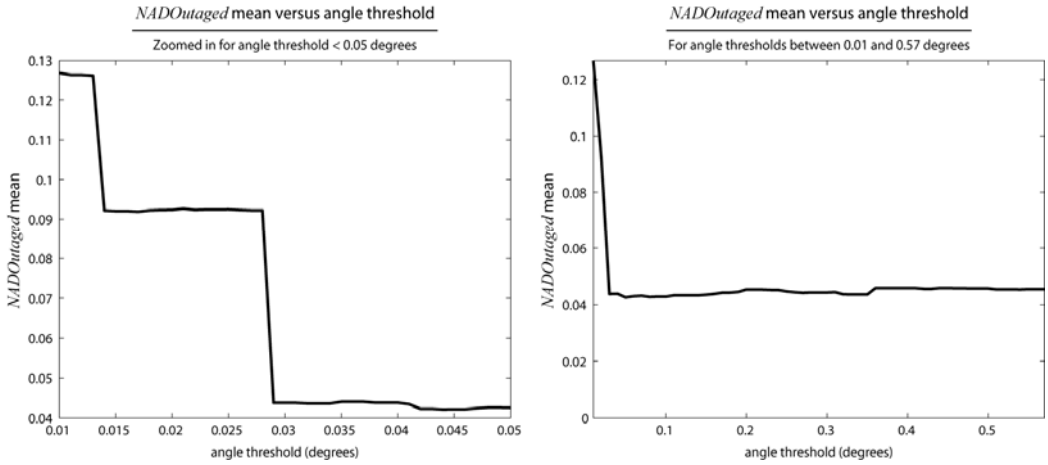

Figure 4.3: Mean *NADOutaged* values for median filtering using a window length of 31 with the angle threshold varied from 0.01 to 0.57 degrees; the left plot shows the right plot zoomed in to the region $\tau \in [0.01, 0.05]$.

To understand why the *NADOutaged* values are so high for the outages of lines 15-54 and 18-37, Figure 4.4 provides a detailed look at the angle changes for all system buses due to the outage of the line with the highest *NADOutaged* value, 18-37, using both ac and dc power flow solutions. Because the dc power flow assumptions are used in constructing the expected angle changes, a failure of these assumptions to hold for the outage of this line is responsible for the large *NADOutaged* value. The

norm of the differences in angle changes between the ac and dc power flow solutions is 0.045 degrees for this outage, which is comparable in magnitude to the norm of the estimated angle changes under the dc power flow assumptions (0.04 degrees). One possible reason for the poor performance of the dc power flow for the outage of the 18-37 or 15-34 lines is that these lines have the highest R/X ratios on the system, with R/X = 1.82, and lines with high R/X ratios are known to cause errors in the accuracy of the dc power flow.

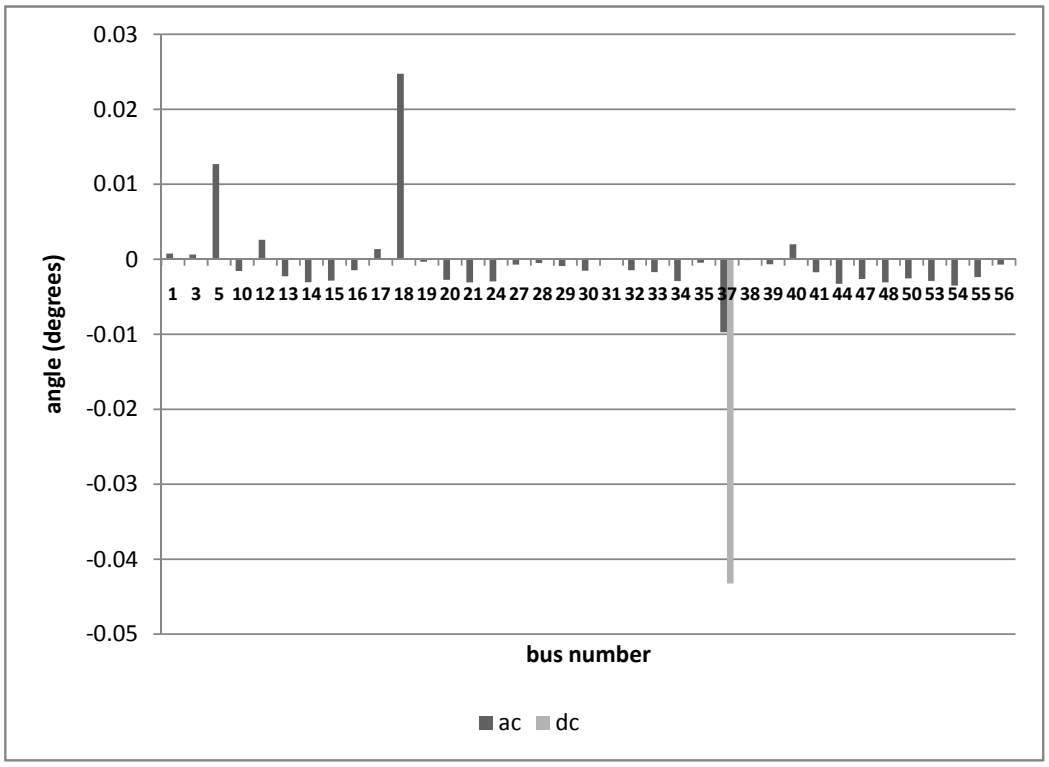

Figure 4.4: Calculated changes in angles at all system buses due to the outage of a line connecting buses 18 and 37 using the ac and dc power flow.

A high *NADOutaged* in and of itself does not guarantee that the algorithm is failing to properly identify the outaged line; what really matters is where a particular line outage ranks in the set of ordered *NAD* relative to all the other possible outages. Figure 4.5 shows that the rank of the line in the *NADVals* list, *RankOutaged*, is indeed

closely tied to the *NADOutaged* value of the outaged line. This figure is a scatter plot of *NADOutaged* versus *RankOutaged* for a wide range of filter lengths (from 3 to 61 in increments of 2) and angle thresholds (from 0.01 to 0.57 degrees in increments of 0.01 degrees). As shown in the plot, for any detected event with an *NADOutaged* value less than 0.8, the algorithm correctly ranks the outaged line. As the true system response becomes less similar to the response predicted by the dc power flow, *NADOutaged* gets larger and the ranking of the outaged line becomes poorer, in some cases resulting in rankings as high as 43 out of 56 (for the outage of the two lines between buses 18 and 37 using window lengths of 29 or 31 with angle thresholds of 0.01 or 0.02 degrees).

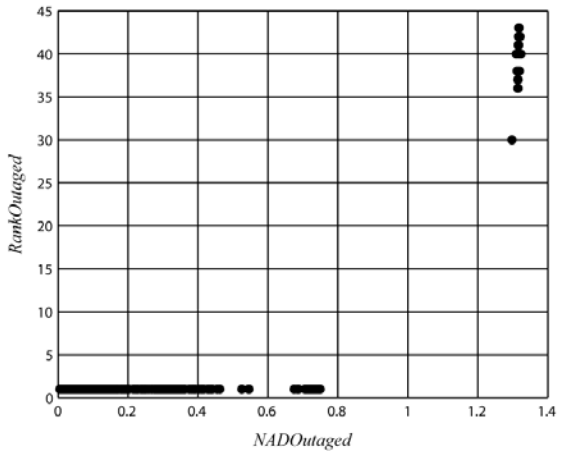

Figure 4.5: Scatter plot of *NADOutaged* versus *RankOutaged* for 84 867 detected line outages using window lengths from 3 to 61 in increments of 2 and angle thresholds from 0.01 to 0.57 degrees in increments of 0.01 degrees.

In addition to the effect of the angle threshold on successful line outage detection, the window length can also have an effect on the value of *NADOutaged* by changing the attenuation of oscillations and the estimation of the steady state angle changes. Figure 4.6 shows the relationship between the median window length and the average

NADOutaged value using a constant angle threshold value of 0.05 degrees. There is a sharp decrease in the *NADOutaged* mean as the filter length is increased from 3 to 19, then a more gradual decrease as the filter length is increased further. This corresponds to the results shown in Figure 3.5, where $\Delta\theta_{miss}$ shows a rapid decay for small window lengths and a more gradual decay for larger window lengths.

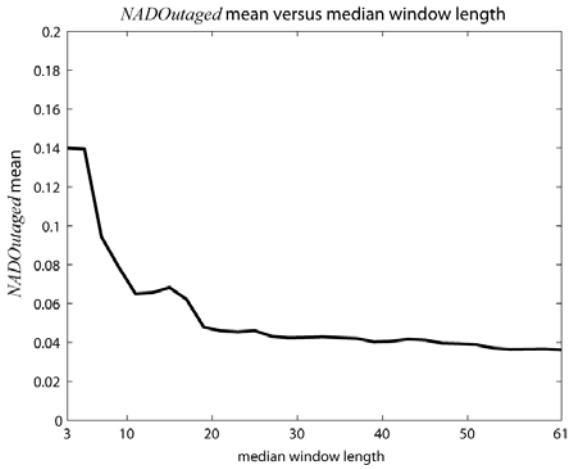

Figure 4.6: Effect of median window length on the mean *NADOutaged* value using an angle threshold of 0.05 degrees.

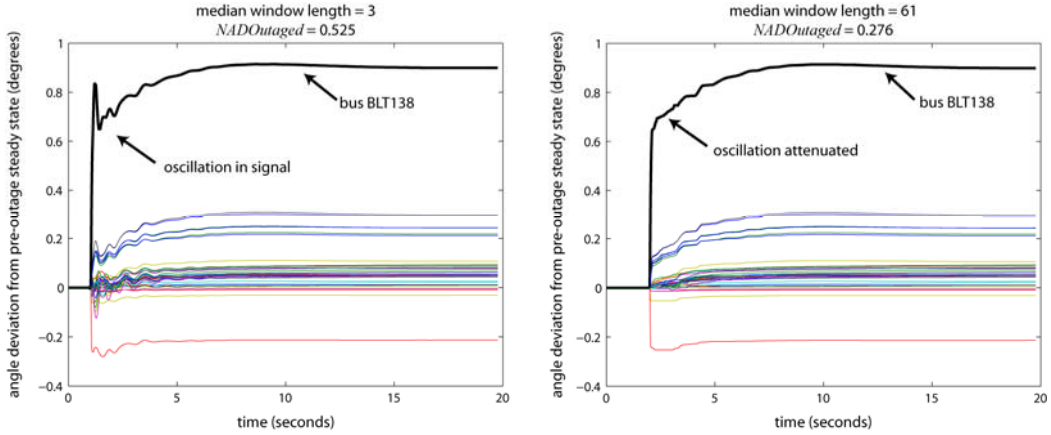

Figure 4.7: Elimination of an oscillation in the BLT138 bus angle signal for the outage of line 47-53 as the median window length is changed from 9 to 41.

The decrease in *NADOutaged* as the window length is increased is due to improved attenuation. As an illustrative example, Figure 4.7 shows an oscillation in

the angle signal of bus BLT138 that is eliminated as the median window length changes from 9 to 41.

4.1.1.1.2 FIR filtered angles

The same outage cases were also analyzed using FIR filtering to determine the angle change vectors. A graph analogous to that of Figure 4.2 showing the *MinThreshold* and *NADOutaged* values for each of the lines in the system is provided in Figure 4.8. The same problem at the low end of the graph can be seen—lines which require very low threshold values have high *NADOutaged* values. This shows that the disparity between the ac and dc power flow solutions (e.g., as shown in Figure 4.4) cannot be compensated for by changing the filtering method and is a fundamental limitation due to the usage of the dc power flow equations to detect the outage event.

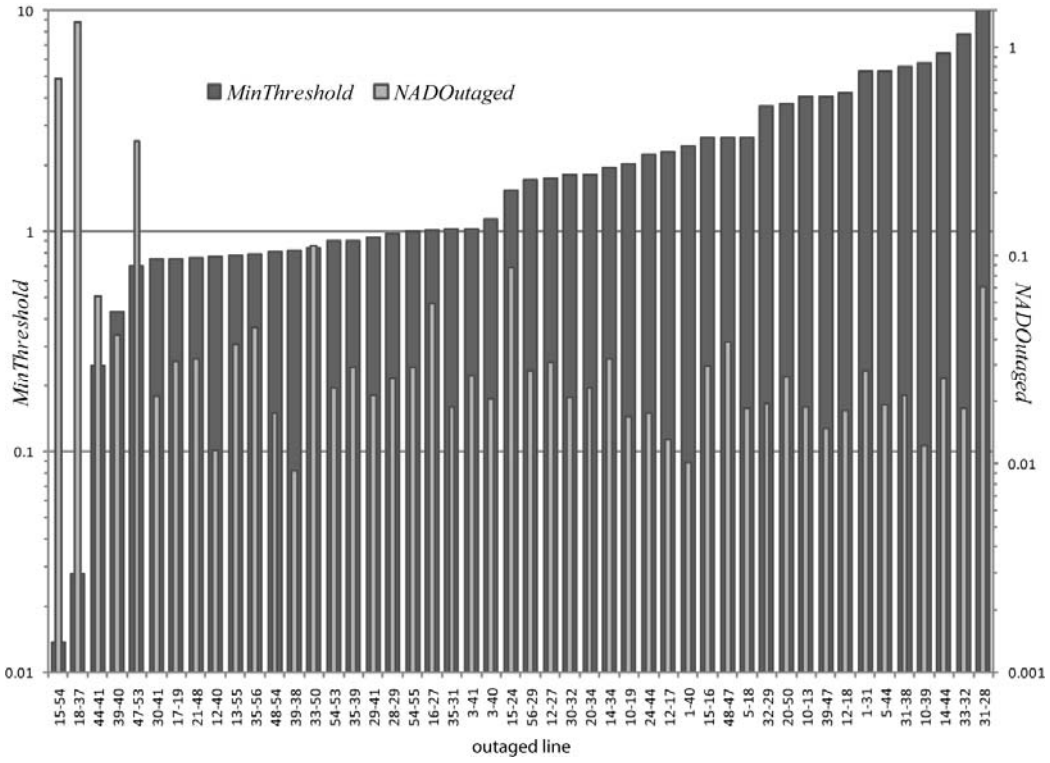

Figure 4.8: Minimum threshold values and error values for each line, ordered in ascending order by minimum threshold, using an FIR filter of order 31.

The usage of higher order FIR filters also has a minimizing effect on the *NADOutaged* mean, as illustrated in Figure 4.9. Of particular note is that there are no portions of the curve in which the *NADOutaged* mean increases as the FIR filter order increases. Comparing this to Figure 4.6, the inverse relationship between filter length and *NADOutaged* mean is more consistent for FIR filtering than median filtering. One reason for this behavior is that FIR filtering attenuates better at all oscillation frequencies above the cutoff frequency as the filter order is increased (see Figure 2.15), whereas this is not necessarily the case for median filtering (see Figure 2.16).

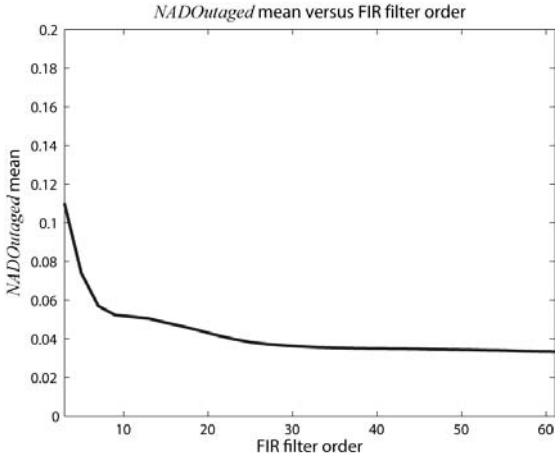

Figure 4.9: Effect of FIR filter order on the mean *NADOutaged* value using an angle threshold of 0.05 degrees.

The scatter plot of *NADOutaged* versus *RankOutaged* in Figure 4.10 shows that the relationship between these two values obtained from median filtering also holds if FIR filtering is used, indicating that the algorithm is more sensitive to the accuracy of the dc power flow assumptions than the filtering method for the case where all bus angles are monitored.

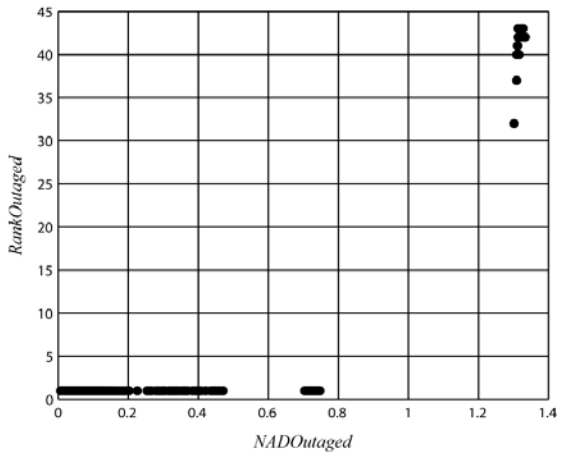

Figure 4.10: Scatter plot of *NADOutaged* versus *RankOutaged* for 86 438 detected line outages using filter orders from 3 to 61 and angle thresholds from 0.01 to 0.57 degrees.

4.1.1.2 Eighteen buses monitored

To examine how the algorithm performs with lower PMU deployment, additional tests were run with only half the buses monitored, using $PMUSet = \{3,10,13,15,17,19,21,27,29,31,33,35,38,40,44,48,53,55\}$.

Figure 4.11 shows the *NADOutaged* versus *RankOutaged* scatter plot for FIR and median filtering using the same parameter range used to construct Figures 4.5 and 4.10. The outliers from the scatter plot using full PMU placement are no longer on the plot because the reduced PMU set does not result in detection of the problematic line outages. Of all the detected events, only 166 have rank not equal to one, and all 166 of these misrankings are with the outage of line 33-50.

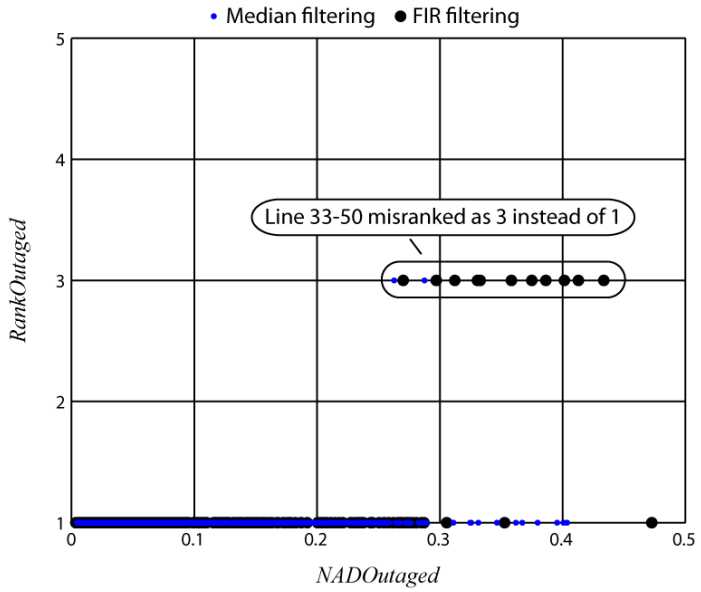

Figure 4.11: Scatter plot of *NADOutaged* versus *RankOutaged* for 82 289 detected line outages (FIR filtering) and 81 846 detected line outages (median filtering) using filter lengths from 3 to 61 and angle thresholds from 0.01 to 0.57 degrees.

Figure 4.12 shows the *RankOutaged* values for all instances in which the outage of line 33-50 is detected with FIR and median filtering. Each blue dot represents a combination of filter order and threshold such that the ranking of the 33-50 line in the *NADVals* array is 3, and each green dot represents the combination such that the ranking is 1. The areas without dots represent the cases where the event was not detected because no candidate signals exceeded the threshold angle τ. This figure demonstrates that setting the angle threshold to a low value can result in misclassification of lines by the algorithm. In this case, the very small amount of flow on line 33-50 before the outage, 2.3 MW, results in small observed angle changes which are difficult to classify.

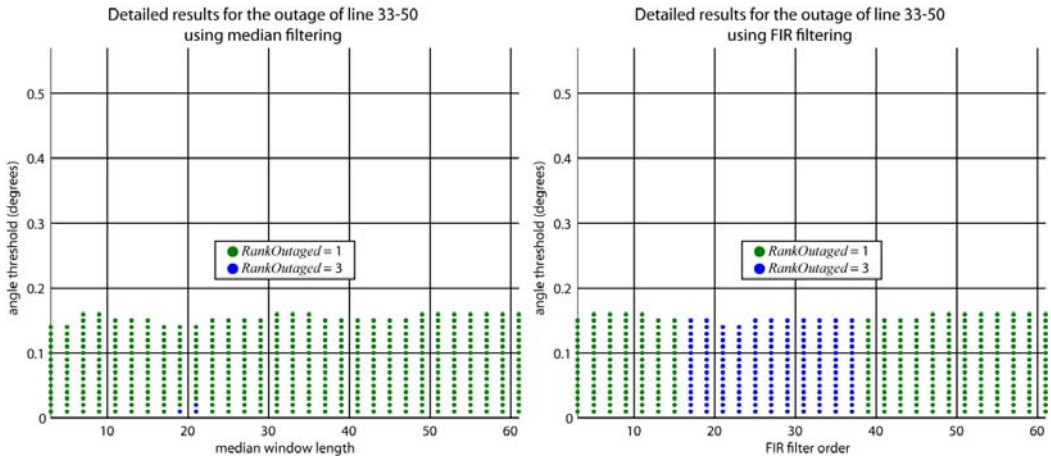

Figure 4.12: Detailed *RankOutaged* distribution for the outage of line 33-50 using FIR filter orders of 3-61 and angle thresholds of 0.01 to 0.57.

One additional result of changing from having a PMU at every bus to having a PMU at 18 buses is how low of an *NADOutaged* is needed to guarantee correct ranking of the line, referred to as *NADOutagedMin*. A higher value of *NADOutagedMin* means that more confidence can be placed in the detection algorithm results if the *NADOutaged* value for the detected line is much lower than *NADOutagedMin*. For median filtering, the *NADOutagedMin* is 0.867 for complete PMU coverage but drops to 0.252 in the case where only 18 PMUs are placed on the system. This is expected, since having more information about the system should allow for more confidence to be placed in the results based on that information. Using FIR filtering gives similar results—*NADOutagedMin* is 0.254 in the case where only 18 PMUs are placed on the system and 0.8403 for a full PMU placement.

The biggest difference between the performance of the algorithm using full PMU coverage and half PMU coverage is in the number of lines which are distinguishable with respect to the algorithm. Two line outages are indistinguishable with respect to

the algorithm if the $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ values calculated using (3.8) are scalar multiples of each other, i.e.,

$$\exists \alpha \in \mathbb{R}, \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \alpha \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} \tag{4.4}$$

One way this can occur is if two lines are in parallel. For the 37-bus case, an example where this occurs is with the two parallel lines connecting buses 21 (WOLEN69) and 48 (BOB69). The outage of either of these lines is modeled as an injection and withdrawal at bus 21 and 48, respectively, as discussed in Section 3.2.1 above. As a result, the $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ values calculated for the outage of either line will be the same, regardless of the PMU deployment on the system. For full PMU deployment, as considered in Section 4.1.1.1, this is the only situation in which two lines will satisfy (4.4). This is due to the fact that the **K** matrix used in (3.8) is the $N \times N$ identity matrix when all buses are monitored.

As the number of PMUs on the system is reduced, there are situations where nonparallel lines still satisfy (4.4). In particular, this can occur when there is an unmonitored portion of the system connected to only one or two PMU-monitored buses.

The first case considered is the where there are two boundary buses. In Figure 4.13, the "Buses not monitored with PMUs" region is eliminated and replaced by loads at the boundary buses equal to the flows out of the boundary bus. The systems on the left and right of Figure 4.13 result in the same angles at the monitored buses.

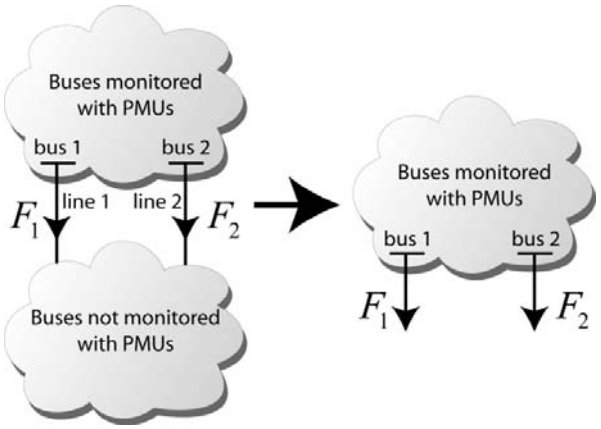

Figure 4.13: Reducing the full system to only buses that are monitored with PMUs.

To see why this is true, first the power flow equations for the left system are given:

$$\begin{aligned} &Buses_m = \{\text{monitored buses}\}, Buses_u = \{\text{unmonitored buses}\} \\ &\forall \text{ buses } i \in Buses_m, \sum_{\substack{j\omega i, \\ j\in Buses_m \\ j\neq i}} F_{i,j}(\boldsymbol{\theta}_m) + \sum_{\substack{j\omega i, \\ j\in Buses_u \\ j\neq i}} F_{i,j}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_u) = P_{inj,i} \\ &\forall \text{ buses } k \in Buses_u, \sum_{\substack{j\omega k, \\ j\in Buses_u \\ j\neq k}} F_{k,j}(\boldsymbol{\theta}_u) + \sum_{\substack{j\omega k, \\ j\in Buses_m \\ j\neq k}} F_{k,j}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_u) = P_{inj,k} \end{aligned} \tag{4.5}$$

where x ω y means there is a line connecting buses x and y, $F_{i,j}$ is the power flow from bus i to bus j, $\boldsymbol{\theta}_m$ is the set of angles at the monitored buses, $\boldsymbol{\theta}_u$ is the set of angles at the unmonitored buses, and $P_{inj,i}$ is the power injected into bus i (positive for generators and negative for loads). Referring to Figure 4.13, the power flow equations at the monitored buses can be rearranged so that $\boldsymbol{\theta}_u$ is eliminated and replaced by the boundary flow:

$$\begin{aligned} &\forall \text{ buses } i \in \left\{Buses_m \setminus \{1,2\}\right\}, \sum_{\substack{j\omega i, \\ j\in Buses_m \\ j\neq i}} F_{i,j}(\boldsymbol{\theta}_m) = P_{inj,i} \\ &\sum_{\substack{j\omega 1, \\ j\in Buses_m \\ j\neq 1}} F_{1,j}(\boldsymbol{\theta}_m) + \sum_{\substack{j\omega 1, \\ j\in Buses_u \\ j\neq 1}} F_{1,j}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_u) = \sum_{\substack{j\omega 1, \\ j\in Buses_m \\ j\neq 1}} F_{1,j}(\boldsymbol{\theta}_m) + F_1 = P_{inj,1} \\ &\sum_{\substack{j\omega 2, \\ j\in Buses_m \\ j\neq 2}} F_{2,j}(\boldsymbol{\theta}_m) + \sum_{\substack{j\omega 2, \\ j\in Buses_u \\ j\neq 2}} F_{2,j}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_u) = \sum_{\substack{j\omega 2, \\ j\in Buses_m \\ j\neq 2}} F_{2,j}(\boldsymbol{\theta}_m) + F_2 = P_{inj,2} \end{aligned} \tag{4.6}$$

Because Equation (4.6) constitutes the power flow equations for the right-side system in Figure 4.13, the power flow solution $\boldsymbol{\theta}_m$ is the same for both the left and right systems.

If an event occurs on the unmonitored system which results in changes to the boundary flows, the external system can still be removed and replaced with injections using the new line flows between the two systems, as shown in Figure 4.14. Barring any net changes in power injections within the unmonitored system (e.g., if the event is a line outage or a transfer between two buses within the unmonitored system), then the sum of the power entering the unmonitored system must stay constant [70]. As a result, the sum of the changes in flows and, consequently, the sum of the changes in equivalent injections must equal zero (i.e., $\Delta F_1 = -\Delta F_2$).

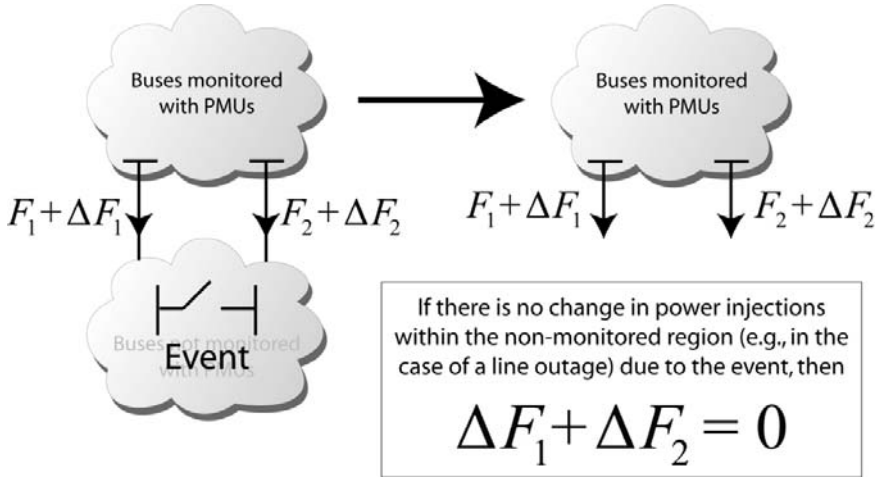

Figure 4.14: The changes in the equivalent injections due to the event will sum to zero if there is no change in power injections within the unmonitored system.

One consequence of this fact is that any two line outages within the unmonitored system will result in angle changes that are scalar multiples of one another (i.e., $\exists \alpha \in \mathbb{R}, \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \alpha \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}$, $l_1, l_2 \in$ unmonitored system). To see why this is the case,

first let $\Delta F_{1,l_1}$ be the change in flow on one of the boundary lines due to the outage of line l_1, and let $\Delta F_{1,l_2}$ be the change in flow on one of the boundary lines due to the outage of line l_2. Referring to Figure 4.14, the outages can be represented by injections of $\left\{\Delta F_{1,l_1}, -\Delta F_{1,l_1}\right\}$ and $\left\{\Delta F_{1,l_2}, -\Delta F_{1,l_2}\right\}$ at the two boundary buses. The estimated change in angles within the monitored system for the outage of line l_1 can then be expressed as

$$
\begin{aligned}
\mathbf{\Delta\theta}_{calc,l_1} &= \mathbf{KB}^{-1}\begin{bmatrix} 0 \\ \tilde{\tilde{P}}_{l,1} \\ -\tilde{\tilde{P}}_{l,1} \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow l_{1,from} \\ \leftarrow l_{1,to} \\ \\ \end{matrix} \\
&= \tilde{\tilde{P}}_{l,1}\mathbf{\Delta\tilde{\theta}}_{calc,l_1} \\
&= \mathbf{KB}^{-1}\begin{bmatrix} 0 \\ \Delta F_{1,l_1} \\ -\Delta F_{1,l_1} \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 1} \\ \leftarrow \text{boundary bus 2} \\ \\ \end{matrix} \\
&= \Delta F_{1,l_1}\mathbf{KB}^{-1}\begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 1} \\ \leftarrow \text{boundary bus 2} \\ \\ \end{matrix} \\
&= \Delta F_{1,l_1}\mathbf{\Delta\theta}_{boundary}
\end{aligned}
\tag{4.7}
$$

$$
\mathbf{\Delta\tilde{\theta}}_{calc,l_1} = \frac{\Delta F_{1,l_1}}{\tilde{\tilde{P}}_{l,1}}\mathbf{\Delta\theta}_{boundary}
$$

Similarly, the change in angles due to the outage of line l_2 can be expressed as:

$$
\begin{aligned}
\Delta\boldsymbol{\theta}_{calc,l_2} &= \mathbf{KB}^{-1}\begin{bmatrix} 0 \\ \tilde{\tilde{P}}_{l,2} \\ -\tilde{\tilde{P}}_{l,2} \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow l_{2,from} \\ \leftarrow l_{2,to} \\ \\ \end{matrix} \\
&= \tilde{\tilde{P}}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} \\
&= \mathbf{KB}^{-1}\begin{bmatrix} 0 \\ \Delta F_{1,l_2} \\ -\Delta F_{1,l_2} \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 1} \\ \leftarrow \text{boundary bus 2} \\ \\ \end{matrix} \\
&= \Delta F_{1,l_2}\mathbf{KB}^{-1}\begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 1} \\ \leftarrow \text{boundary bus 2} \\ \\ \end{matrix} \\
&= \Delta F_{1,l_2}\Delta\boldsymbol{\theta}_{boundary}
\end{aligned}
\tag{4.8}
$$

$$
\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} = \frac{\Delta F_{1,l_2}}{\tilde{\tilde{P}}_{l,2}}\Delta\boldsymbol{\theta}_{boundary}
$$

Combining the results from (4.7) and (4.8), the condition in (4.4) holds:

$$
\begin{aligned}
\frac{\tilde{\tilde{P}}_{l,1}}{\Delta F_{1,l_1}}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \Delta\boldsymbol{\theta}_{boundary} = \frac{\tilde{\tilde{P}}_{l,2}}{\Delta F_{1,l_2}}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} \\
\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \alpha\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}, \alpha = \frac{\Delta F_{1,l_1}}{\tilde{\tilde{P}}_{l,1}}\frac{\tilde{\tilde{P}}_{l,2}}{\Delta F_{1,l_2}}
\end{aligned}
\tag{4.9}
$$

An example of where this occurs for the study case with 18 PMUs is shown in Figure 4.15. Relating this diagram to the discussion in the previous paragraph, buses 14, 34, 20, and 50 constitute the unmonitored system, buses 33 and 44 are the boundary buses, and lines 14-44 and 33-50 are the boundary lines.

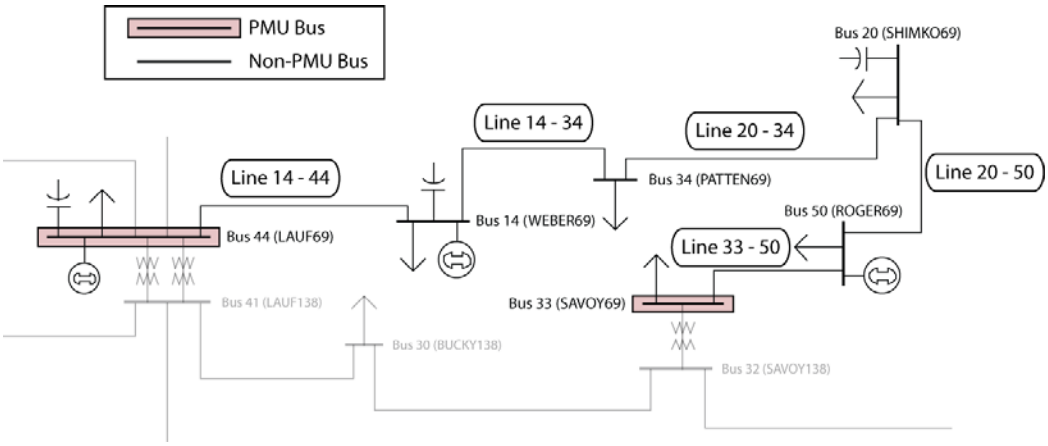

Figure 4.15: Indistinguishable line outages using the 18 PMU configuration of the 37-bus case.

Table 4.1: Algorithm results for lines shown in Figure 4.15 using order 61 FIR filtering and angle threshold of 0.01 degrees

From bus	To bus	*NADOutaged*	*RankOutaged*	*SharedRank*	*FlowError* (MW)
14	34	0.0138	1	4	0.6641
14	44	0.0636	1	4	0.8682
20	34	0.0394	1	4	0.0002
20	50	0.0349	1	4	0.6832
33	50	0.1966	1	4	0.6573

Using FIR filtering of order 61 and an angle threshold of 0.01 degrees, the algorithm is able to correctly identify each line when it goes out (i.e., *RankOutaged* = 1 for all outages) and predict the pre-outage flow accurately. Table 4.1 provides details on the algorithm's performance for the outage of each of the lines in the unmonitored system with two additional performance metrics:

$$SharedOutaged\left(l_{outaged}\right) = \left|\left\{l : NADVals_l = NADVals_{l_{outaged}}\right\}\right| - 1$$
$$FlowError\left(l_{outaged}\right) = P_{l_{outaged}}^{(\text{estimated by algorithm})} - P_{l_{outaged}}^{(\text{from case})} \quad (4.10)$$

Ideally, *SharedOutaged* would be zero (indicating that the outaged line is distinguished from all other lines) and *FlowError* would also be zero (indicating that the algorithm predicted the pre-outage flow perfectly). The *FlowError* values in Table 4.1 are very low, and the *SharedOutaged* confirms that the topology results in each line being indistinguishable from the other with respect to the algorithm. One way to mitigate the impact of having several lines that are indistinguishable is to present the results by highlighting all lines which have *RankOutaged* equal to one on the one-line diagram. This would still provide very useful information; namely, the subset of the lines on the system that contains the outage line.

For the case where there is only one boundary bus between the monitored and unmonitored system (i.e., the unmonitored system is radially connected to the monitored system), the change in flow on the boundary lines will be zero if the net power injections within the unmonitored system do not change (e.g., for a line outage within the unmonitored system). Therefore, the change in equivalent injection representing the unmonitored system is zero and the angles at the PMU-monitored buses do not change. One instance of this topology for the 37-bus case is shown in Figure 4.16. Because bus 37 is only connected to the rest of the system through one bus, unless there is a PMU measurement from bus 37 there will be no estimated change in the estimated system angles ($\mathbf{\Delta\tilde{\theta}}_{calc,l} = \mathbf{0}$). This is also shown in Figure 4.4, where the angle changes obtained by running the dc power flow solution after the outage of one of the lines connecting buses 18 and 37 are shown to be zero for all buses besides bus 37.

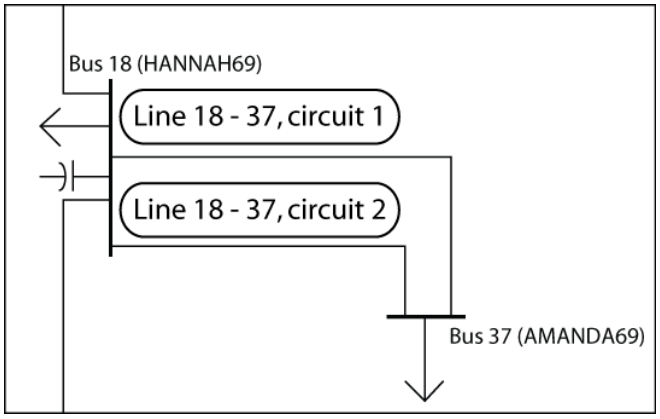

Figure 4.16: Instance of a radial system (single boundary bus) from the 37-bus case.

The inability of the algorithm to detect radial lines without monitoring of the radial buses can be significant. For instance, if one of the lines between buses 18 and 37 goes out, it would be very helpful for the operator to know that there is only one line left to serve the load at bus 37. However, to allow the algorithm to detect one of the lines going out, a PMU must be placed at bus 37. This fact can help guide system planners in determining where to place new PMUs on the system, particularly if there is a known issue with reliability of one of the lines.

4.1.2 TVA system line outage

The second event tested is based on real measurements from the Tennessee Valley Authority (TVA) system after the outage of a 500-kV transmission line carrying approximately 1000 MW of power. The buses with PMU angle signals are shown in Figure 4.17. The unfiltered PMU signals obtained from these PMUs are shown in Figure 4.18, with the 8CORDOVA bus angle taken as the reference.

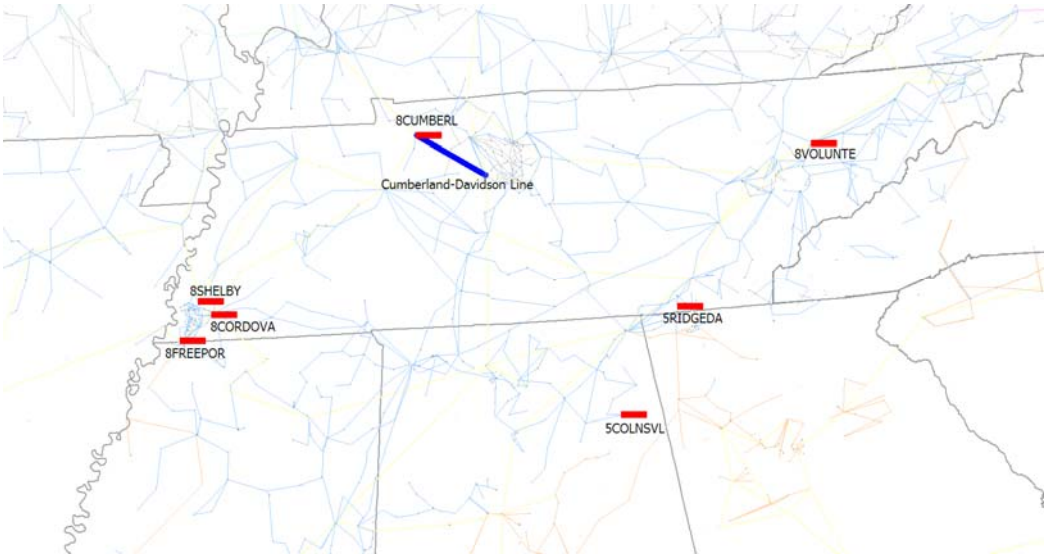

Figure 4.17: Location of PMU angle measurements on TVA system and location of the line outage.

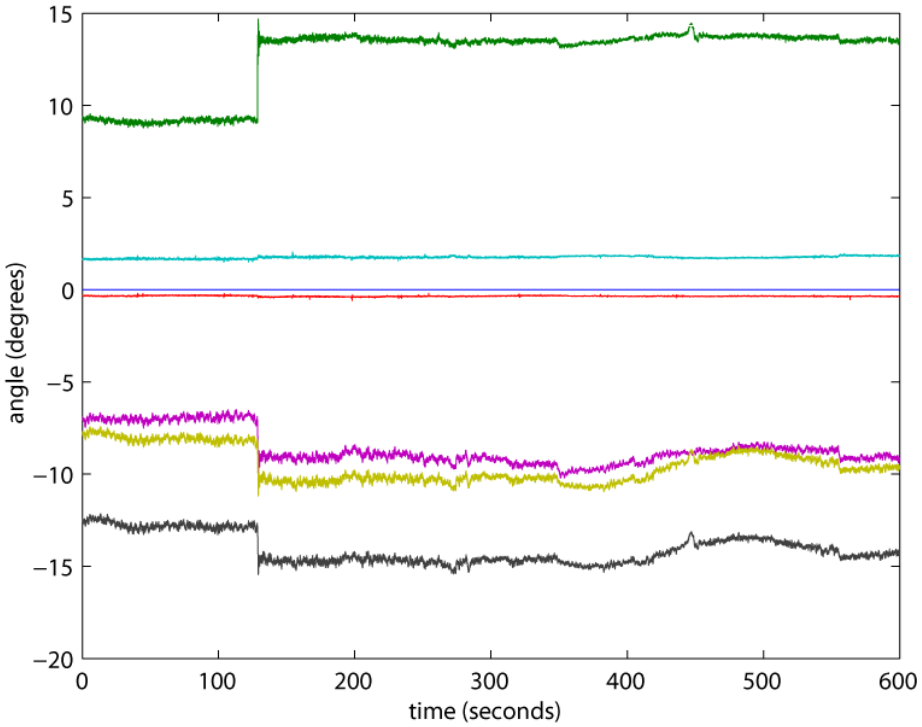

Figure 4.18: Unfiltered angle measurements from before and after the line outage.

As with the 37-bus outages, an extensive sweep of filter type, filter length, and angle threshold was conducted to test the algorithm with this event. Figure 4.19 shows the *RankOutaged* values for FIR and median filtering using filter lengths in the

range of 3 to 401 and τ values between 0.1 and 0.5 degrees. Using a larger value of τ results in an improved ranking for a given filter length due to the reduced interference of noise in determining the sample n_{max}, which is in turn used to determine $\Delta\boldsymbol{\theta}_{observed}$. The median filter also performs much poorer than FIR filtering for these signals, primarily due to the lack of noise attenuation (as shown in Figure 4.20).

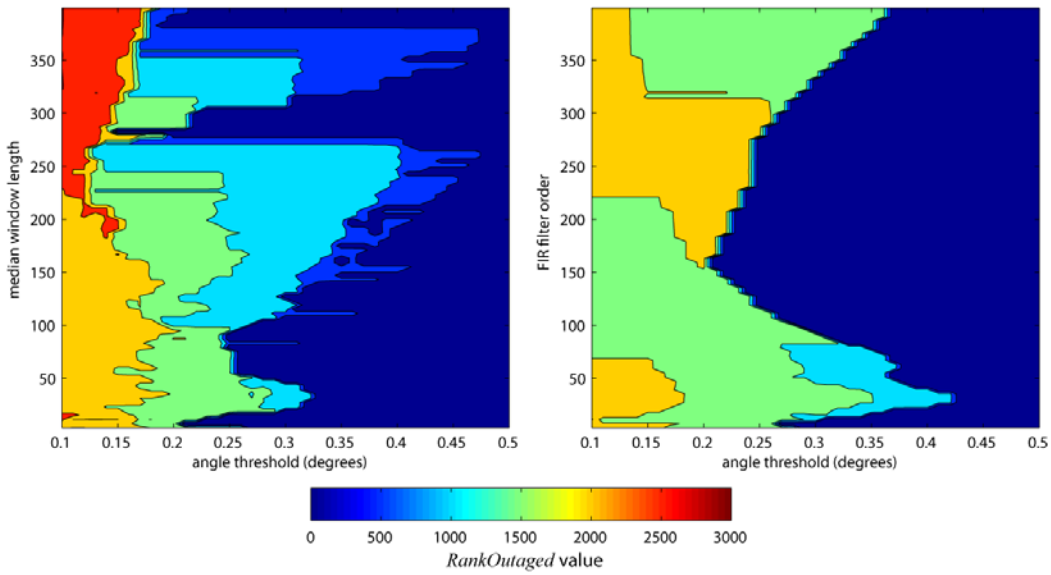

Figure 4.19: *RankOutaged* values for detection of the TVA outage event using filter lengths in the range of 3 to 401 and angle thresholds in the range of 0.1 to 0.5 degrees.

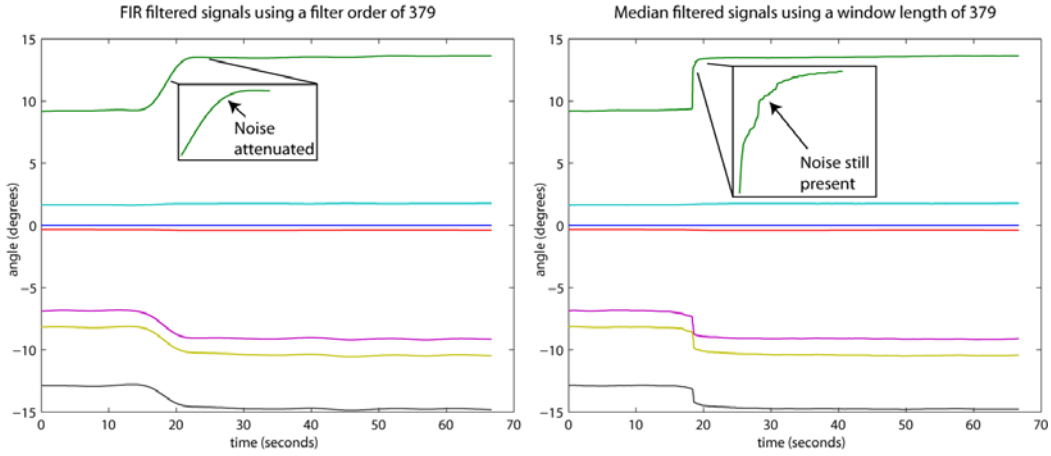

Figure 4.20: Differences in noise attenuation for FIR and median filtering for a filter length of 379.

The presence of noise in the TVA PMU signals is the key difference between these values and those simulated for the 37-bus case. Because noise is present, the lower bound on τ must be raised to avoid any misclassification of noise as an event. To show how a poor choice of τ and filter length can cause false positives, the following metric is defined:

$$EarlyEdge\left(FilterLength,\tau\right)=\begin{cases}1 & , n_{max} < n_{line\ switch} \\ 0 & , \text{otherwise}\end{cases} \tag{4.11}$$

where $n_{line\ switch}$ is the sample number corresponding to the outage of the Cumberland-Davidson line. Figure 4.21 shows the contour plots of *EarlyEdge* for FIR and median filtering, calculated with filter lengths between 3 and 401, and angle thresholds between 0.1 and 0.5 degrees. The red areas indicate which combinations of filter length and angle threshold result in misclassification of noise as an event. For any filter length, increasing the threshold results in lower misclassification; increasing the filter length can occasionally result in additional false positives due to the random noise present in the signals. These results show that choosing a higher value of τ will tend to give better results, although the downside of using a large value of τ is the potential increase in false negatives.

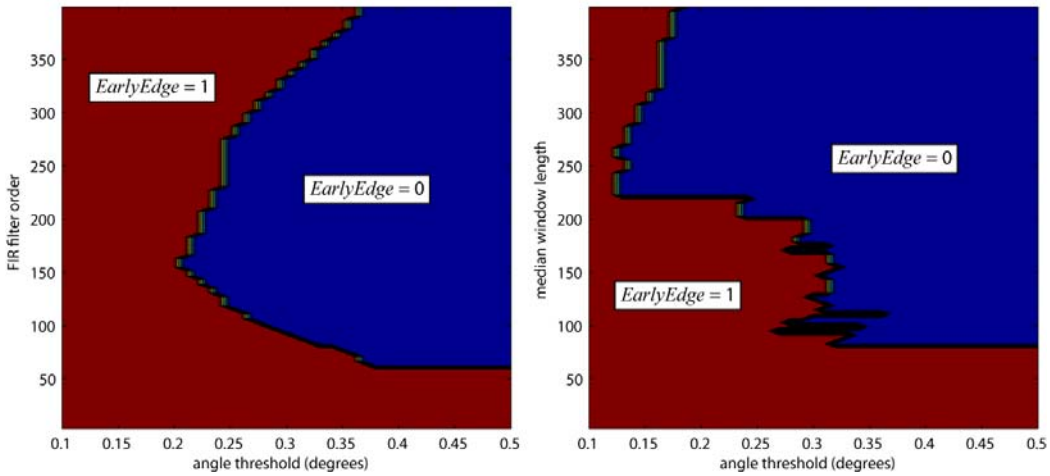

Figure 4.21: *EarlyEdge* contour plot for FIR (left) and median (right) filtering using filter lengths in the range of 3 to 401 and angle thresholds in the range of 0.1 to 0.5 degrees.

Because the outaged line is connected to the PMU at 8CUMBERL, it is interesting to see how the algorithm performs when the angle measurement at 8CUMBERL is removed. Figure 4.22 shows that the algorithm performs just as well for certain combinations of filter length and angle threshold, but for other settings there is significant performance degredation. One cause of the poorer performance without the 8CUMBERL angle is that the oscillations are easier to attenuate for the other buses due to the lower amplitudes, and as a result it takes longer for the algorithm to settle on the n_{max} value. As n_{max} gets larger, more of the postoutage drift in the steady state angle is incorporated into $\Delta\boldsymbol{\theta}_{observed}$ and results in poorer algorithm performance. The correct detection of the line outage for lower filter orders is promising, and indicates that the line outage can be detected with relatively low delay. These results also provide strong evidence that a basic median filter is likely to perform worse than a similarly sized FIR filter with noisy measurements due to poorer attenuation of noise.

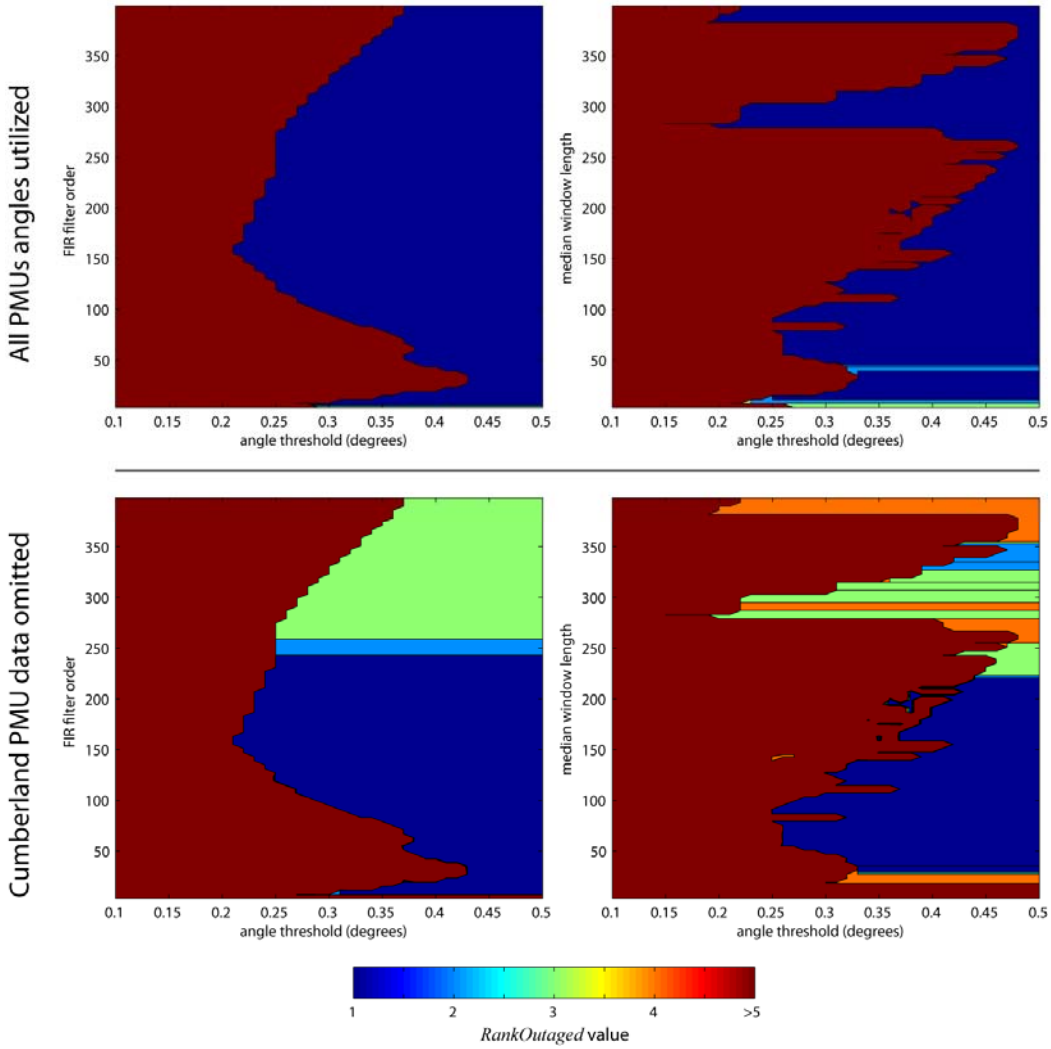

Figure 4.22: *RankOutaged* values with and without the usage of 8CUMBERL PMU angle measurements.

4.2 Generator Outage Examples

The next set of examples looks at how well the algorithm can detect generator outages on the system. The 37-bus case used to test single-line outage detection was also used to simulate PMU signals due to generator outages. Each of the nine generator outages was simulated using PSS/E, and the data was then processed in the manner discussed at the beginning of Section 4.1.1.

4.2.1 All buses monitored

As with the single-line outages, the first case examined is the case where there are PMUs located at all buses on the system. The two sets of contours in Figures 4.23 and 4.24 show how well the algorithm ranks the outaged generator based on the angle measurements using FIR filtering with filter orders between 3 and 301 and angle thresholds between 0.1 and 3.0 degrees. The blank spaces in the contours show where the event was not detected (i.e., no angle exceeded the threshold).

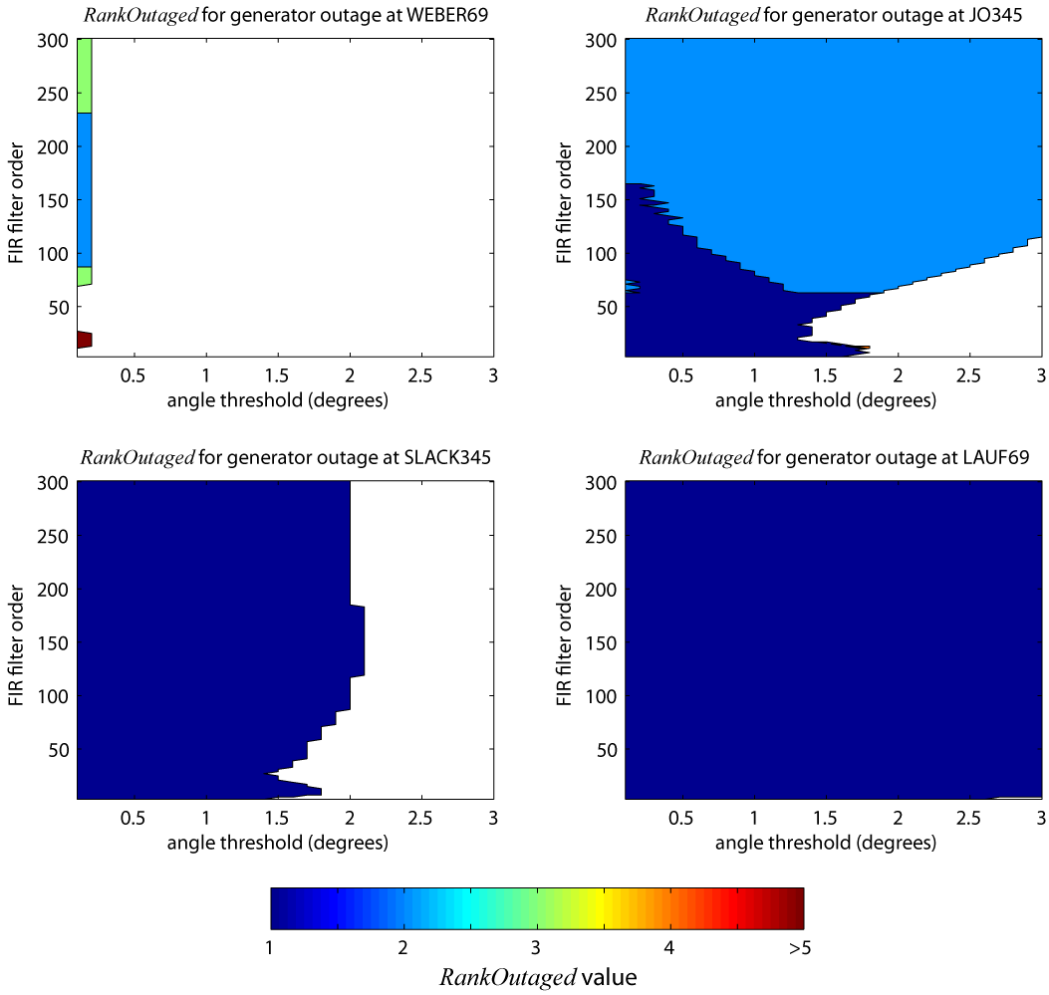

Figure 4.23: Contour plot of *RankOutaged* for generator outages at WEBER69, JO345, SLACK345, and LAUF69 using FIR filtering orders of 3 to 301 and angle thresholds of 0.1 to 3.0 degrees.

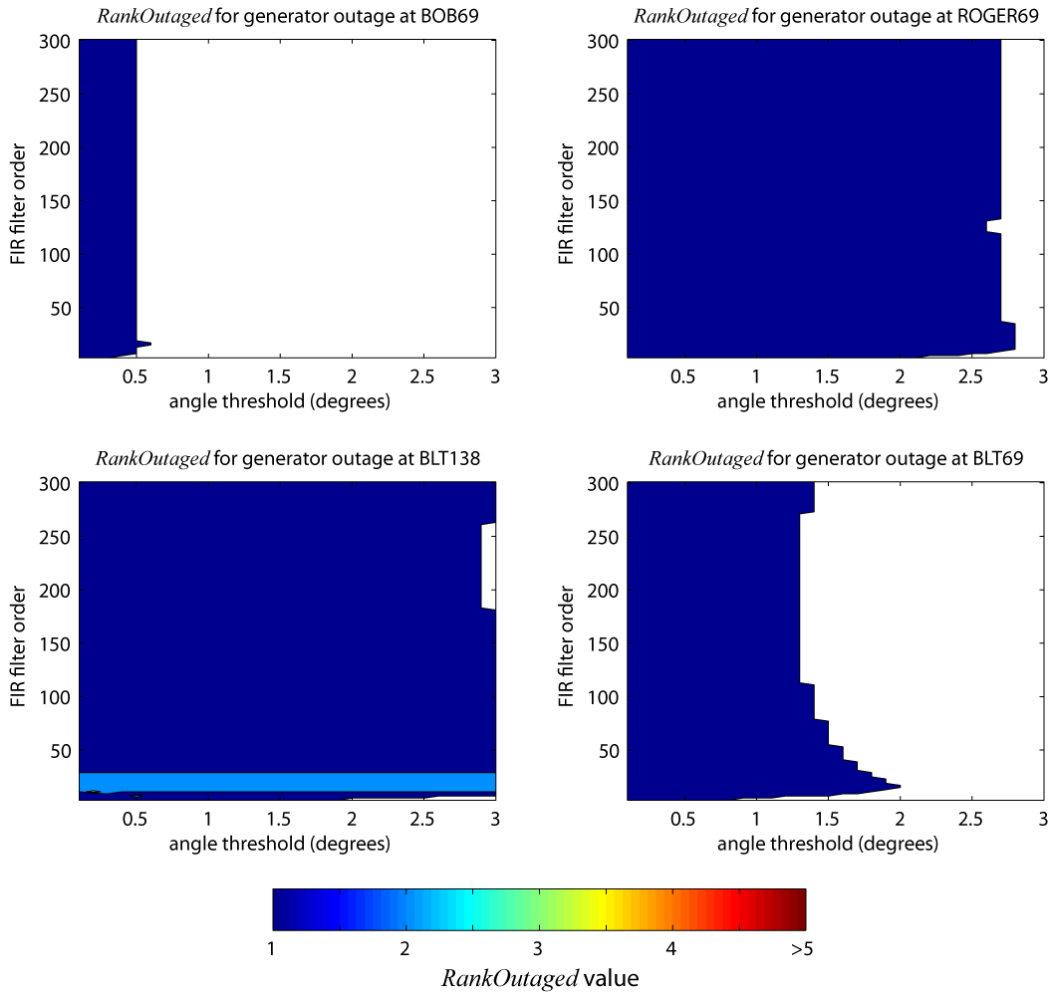

Figure 4.24: Contour plot of *RankOutaged* for generator outages at BOB69, ROGER69, BLT138, and BLT69 using FIR filtering orders of 3 to 301 and angle thresholds of 0.1 to 3.0 degrees.

The only generator that is grossly misranked is WEBER69. As with the single-line outages that resulted in small angle changes, the small deviation in angles due to the outage of this generator make it very difficult to match the correct outage to the observed angle changes. This is further evidence that selecting the correct value of τ is necessary to ensure the algorithm detects events properly.

The reason that the JO345 outages have *RankOutaged* equal to two for large portions of the contour is the inability of the algorithm to attenuate the oscillations with a filter cutoff frequency of 0.1 Hz. Figure 4.25 shows that an FIR filter of order 151 with a cutoff frequency of 0.1 Hz is incapable of properly damping the oscillations that are caused by this outage. Lowering the cutoff frequency to 0.01 Hz can improve results, as shown in Figure 4.26, but only for filter orders above 400.

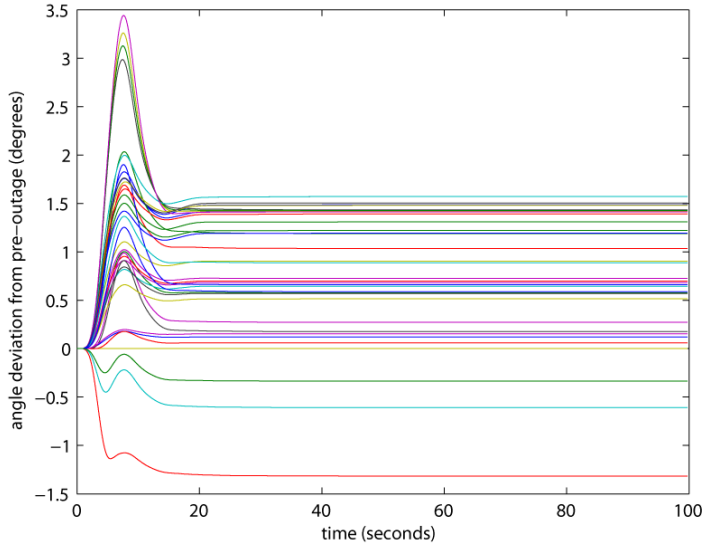

Figure 4.25: Prevailing oscillations after the outage of a JO345 generator using FIR filtering with filter order 151 and cutoff frequency of 0.1 Hz.

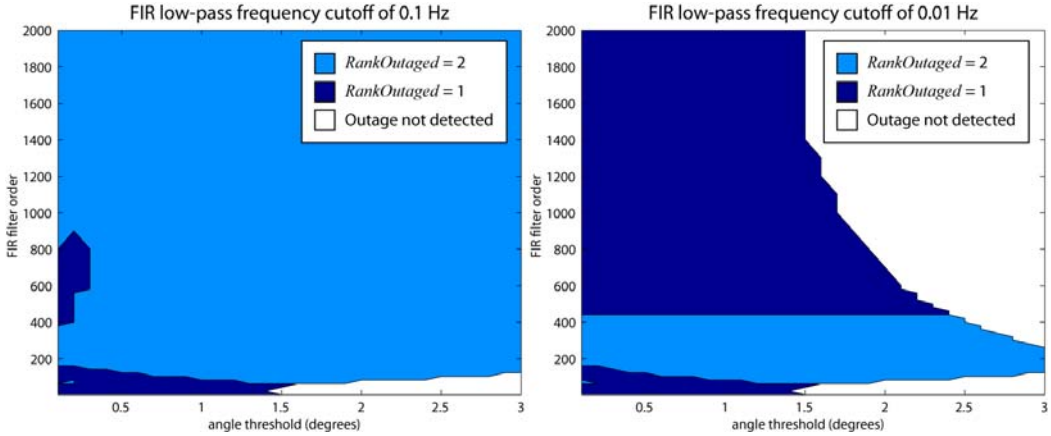

Figure 4.26: Effect of FIR low-pass cutoff frequency on the detection and rank of the JO345 generator outage.

The same tests were run using median filtering instead of FIR filtering, and the resulting contour plots are shown in Figures 4.27 and 4.28. The performance of the algorithm is similar for both filtering methods, although median filtering is able to correctly rank the JO345 outage slightly more often than FIR filtering. The other key difference is that the band in Figure 4.24 where *RankOutaged* is equal to two for the BLT138 outage is absent in the same contour of Figure 4.28.

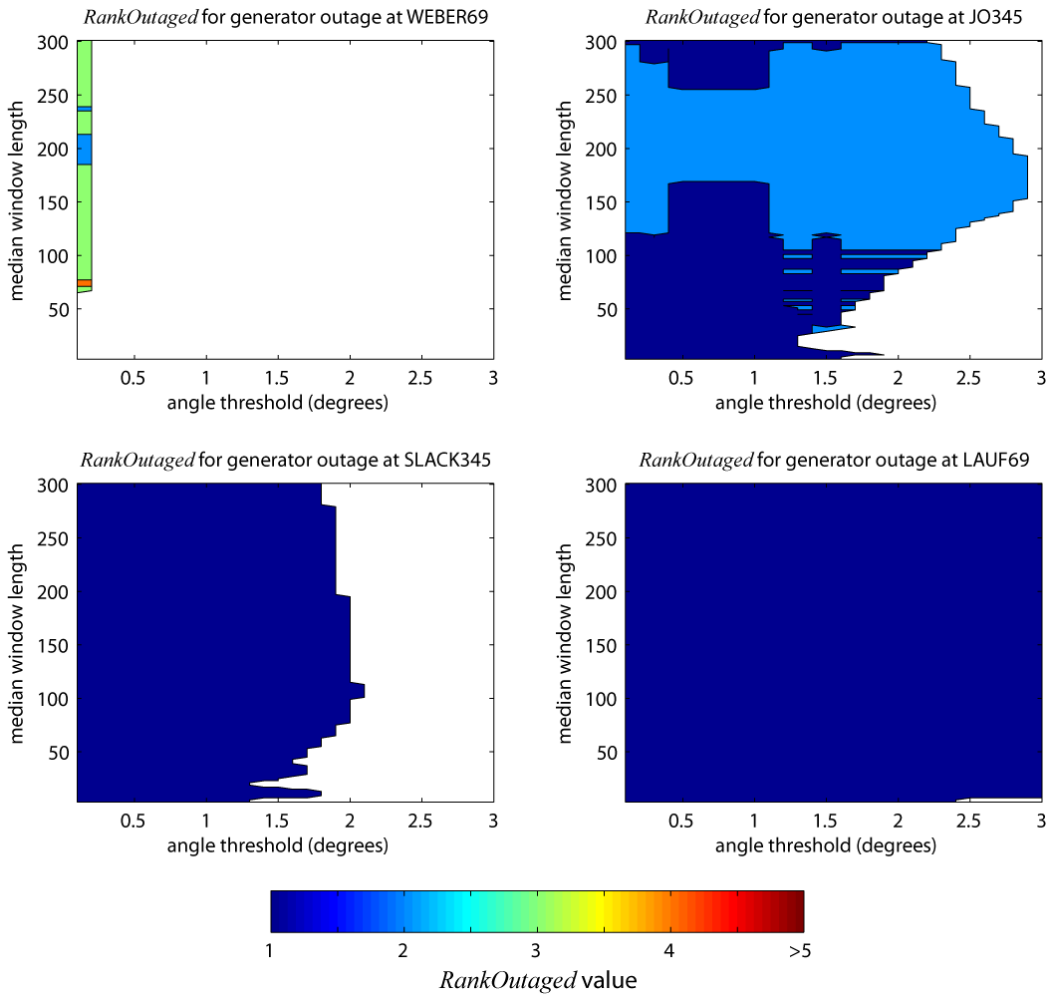

Figure 4.27: Contour plot of *RankOutaged* for generator outages at WEBER69, JO345, SLACK345, and LAUF69 using median filtering with window lengths of 3 to 301 and angle thresholds of 0.1 to 3.0 degrees.

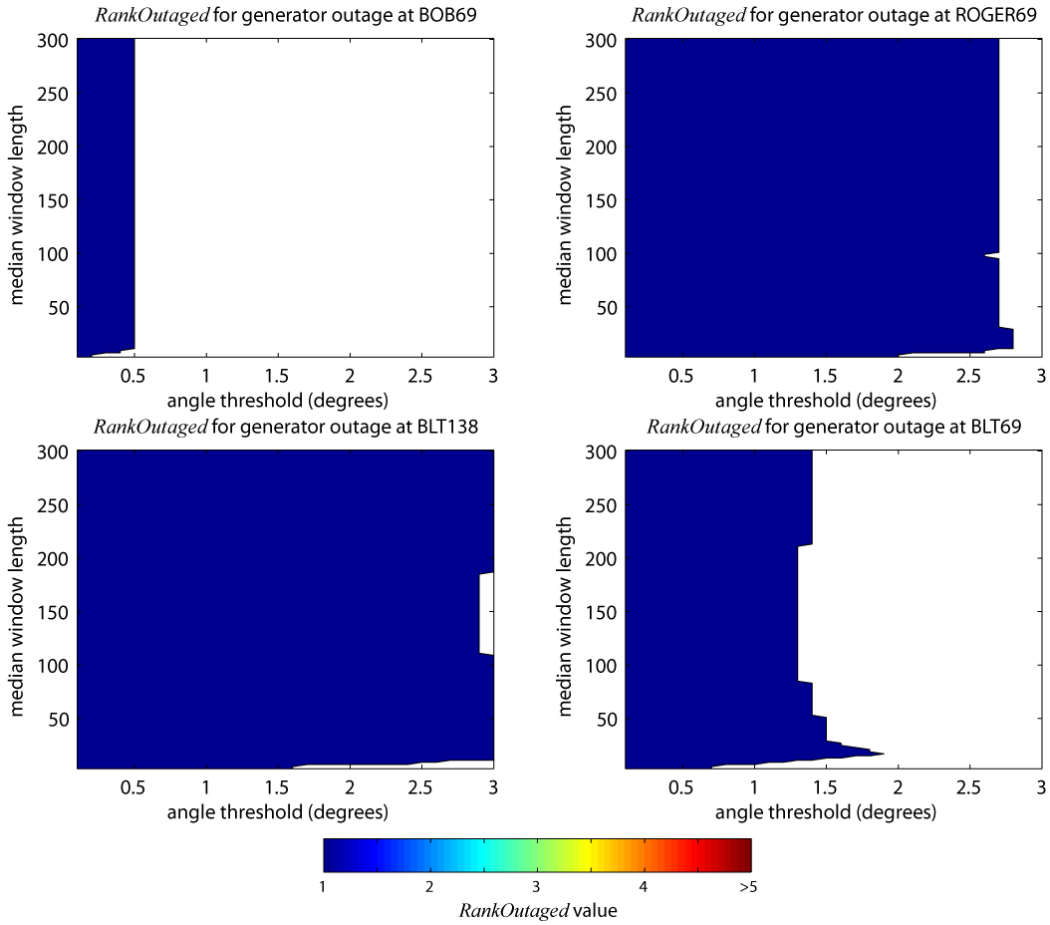

Figure 4.28: Contour plot of *RankOutaged* for generator outages at BOB69, ROGER69, BLT138, and BLT69 using median filtering with window lengths of 3 to 301 and angle thresholds of 0.1 to 3.0 degrees.

Because the median filter has only one parameter, the window length, there is no need to tune any additional parameters such as the cutoff frequency. Figure 4.29 is presented to contrast the performance when median and FIR filtering are used. Comparing Figures 4.26 and 4.29, median filtering results in a *RankOutaged* value of one significantly more often than FIR filtering with 0.1 or 0.01 Hz cutoff frequencies.

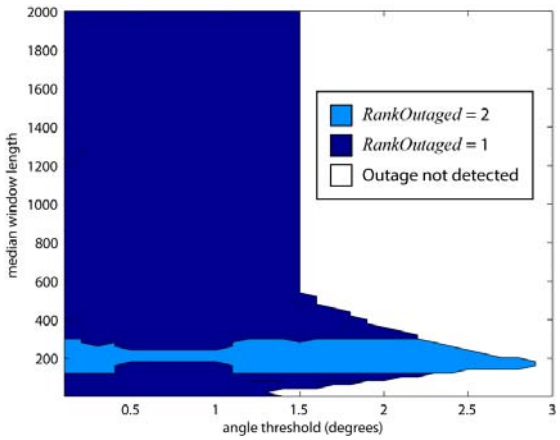

Figure 4.29: A detailed look at ranking and detection of the JO345 generator outage using median filtering.

4.2.2 Eighteen buses monitored

The same placement of eighteen PMUs used in the single outage study was used to see how well the generator outage detection algorithm performs with sparser PMU deployment. Figures 4.30 and 4.31 show the *RankOutaged* plots for the generator outages using this reduced PMU set with FIR filtering.

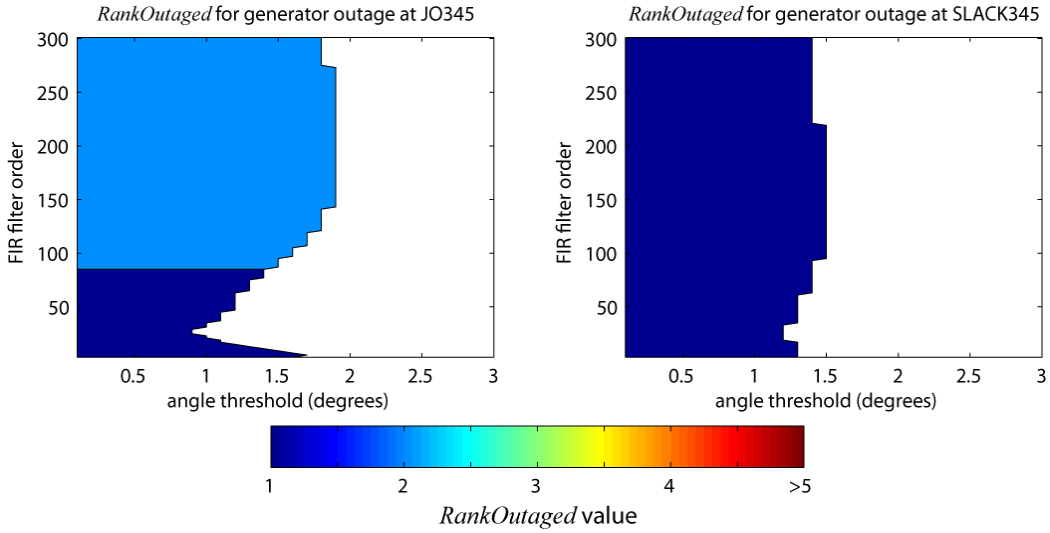

Figure 4.30: Contour plots analogous to those in Figure 4.23 for FIR filtering with 18 buses monitored instead of full bus monitoring. The WEBER69 outage is undetected and the LAUF69 outage has *RankOutaged* equal to one over the entire region, so these contours are not shown.

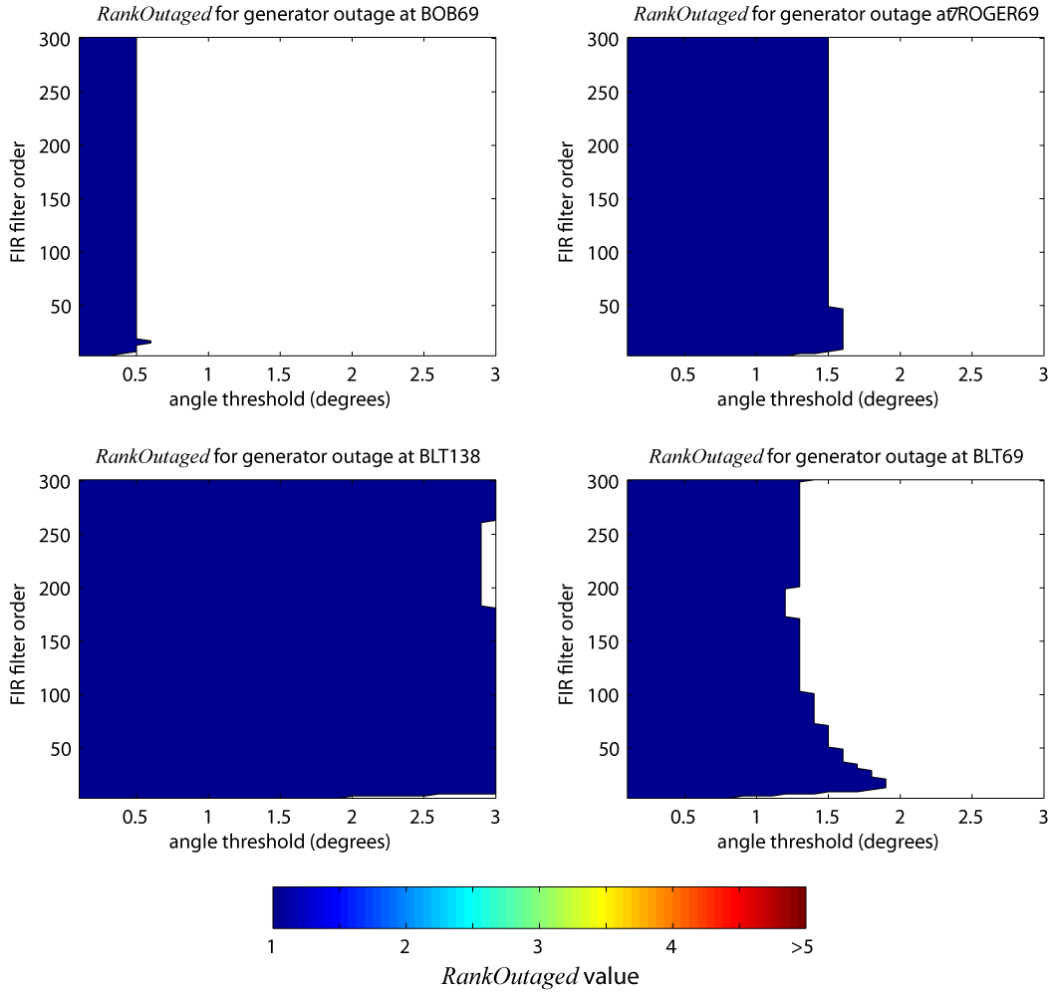

Figure 4.31: Contour plots analogous to those in Figure 4.24 for FIR filtering with 18 buses monitored instead of full bus monitoring.

As the number of monitored buses decreases, the set of angles which can signal the occurrence of an event is reduced. This is why there are larger empty regions in the contours with only 18 buses monitored—some angle measurements needed to detect the line with higher values of τ are no longer available. On real systems, this knowledge of how a change in the set of monitored buses impacts the number of false negatives can help in deciding where to place PMUs on the system. Finally, Figures 4.32 and 4.33 show that median filtering results in a similar reduction in the ability to detect the outaged generators.

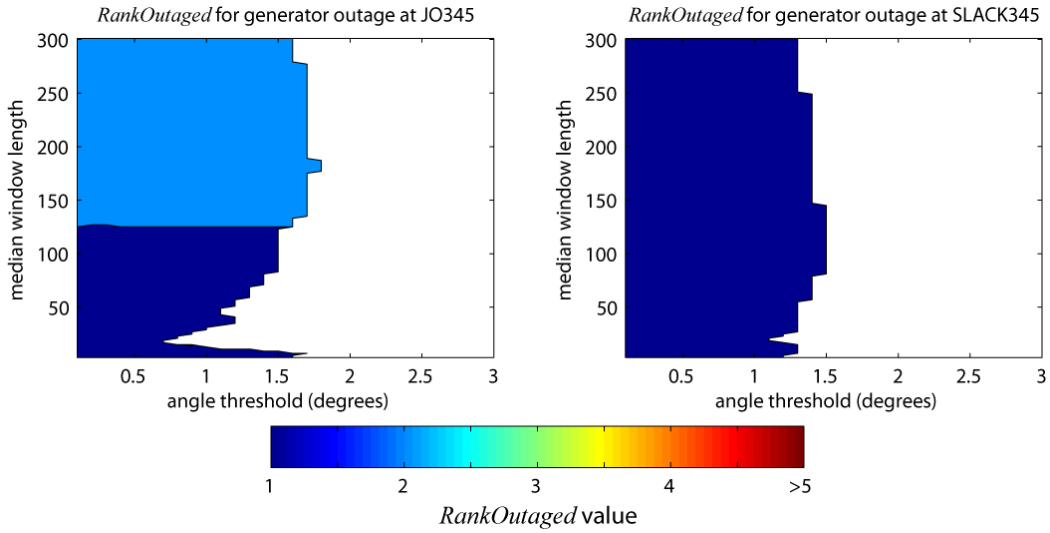

Figure 4.32: Contour plots analogous to those in Figure 4.27 for median filtering with 18 buses monitored instead of full bus monitoring. The WEBER69 outage is undetected and the LAUF69 outage has *RankOutaged* equal to one over the entire region, so these contours are not shown.

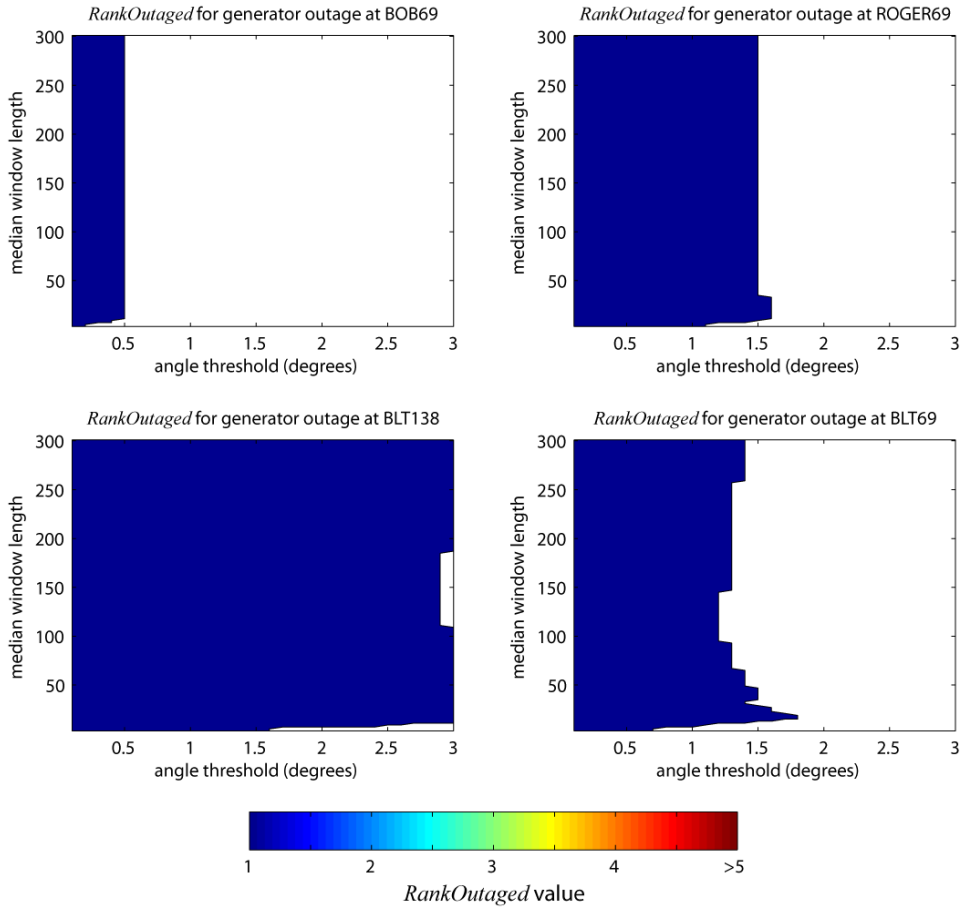

Figure 4.33: Contour plots analogous to those in Figure 4.28 for median filtering with 18 buses monitored instead of full bus monitoring.

4.3 Double-Line Outage Examples

4.3.1 All nonislanding double-line outages

The same 37-bus system was used to test the performance of double-line outage detection. A time domain simulation was run for each of the 1504 nonislanding double-line outages to see how well the algorithm detects double-line outages using the methods described in Section 3.4.

4.3.1.1 All buses monitored

The first PMU configuration considered is with PMUs monitoring every system bus. The algorithm was run using filter lengths of 3, 9, 31, and 61 with FIR and median filtering using τ values of 0.02, 1.83, 2.5, and 5.16 degrees. These τ values were chosen so that, for signals filtered with an FIR filter of order 31, 100%, 75%, 50%, and 25% (respectively) of the outages were detected by at least one angle exceeding the threshold.

Figure 4.34 shows in matrix form the *RankOutaged* value for each double-line outage on the system, with each row/column representing one of the 56 in-service lines. The color of each square is used to indicate how the algorithm performed; red indicates a misranking (i.e., *RankOutaged* > 1) and green indicates correct ranking (i.e., *RankOutaged* = 1). The blank squares represent single-line outages and double-line outages that were not simulated due to islanding. Of the 1504 total double-line outages considered, 1231 are ranked correctly and 273 are misranked; of these 273, 255 involve the two lines between buses 18 and 37 and the three lines between buses 15 and 54. These lines correspond to the rows with high red density at the top of Figure 4.34. Given the high R/X ratios of these lines, it is unsurprising that the

algorithm, which relies on the dc power flow equations, is unable to properly rank these outages. Also, these were the same lines that caused problems for the single-line outage detection (see Figure 4.2). The only way to solve this problem would be to use the full ac power flow equations rather than the dc power flow equations when attempting to match outages with observed measurements; however, this would lead to significantly higher computational costs and require much more state information than is currently available in real power systems over a suitably wide area.

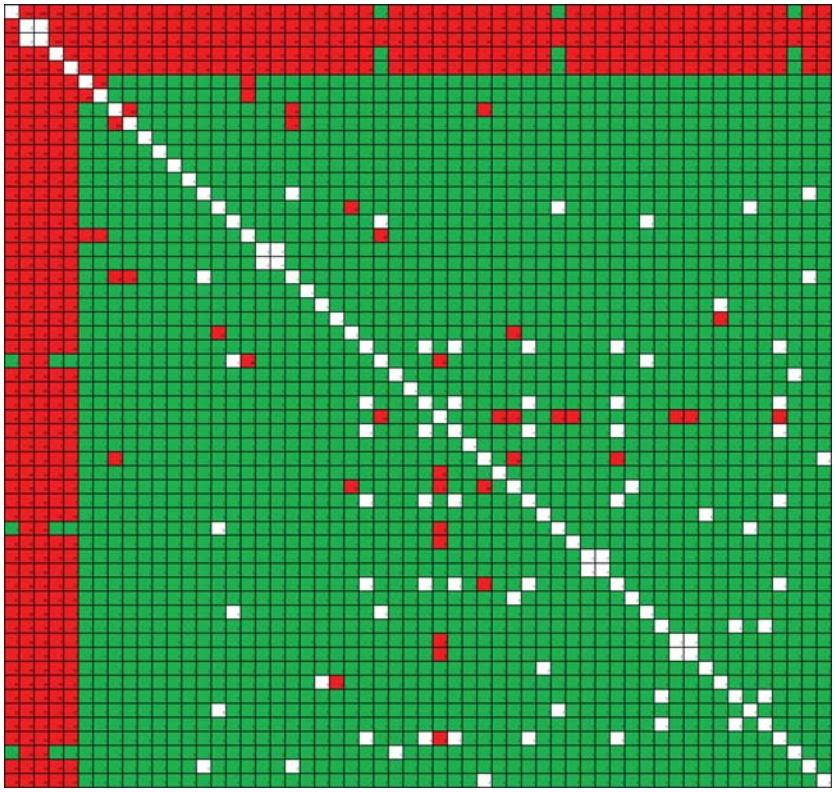

Figure 4.34: Ranking of each line-line combination using FIR filter order 31, angle threshold of 0.02 degrees. Green squares indicate double-line outages where *RankOutaged* = 1, red squares indicate *RankOutaged* > 1 (i.e., misclassification), and white squares indicate outages that were not tested.

Fortunately, due to the small angle changes associated with the outages of the high R/X lines, many of the double outages involving lines 18-37 and 15-54 are undetected if τ is set higher. Figure 4.35 shows the *RankOutaged* matrix for the four

values of τ used to test the algorithm, where black squares are used to indicate events that were not detected. With the threshold set to 1.18 degrees, the number of misranked events involving the 18-37 and 15-54 lines drops from 255 to 100. The undesirable side effect is that several outages which were ranked properly in the $\tau = 0.02$ degree runs are no longer detected.

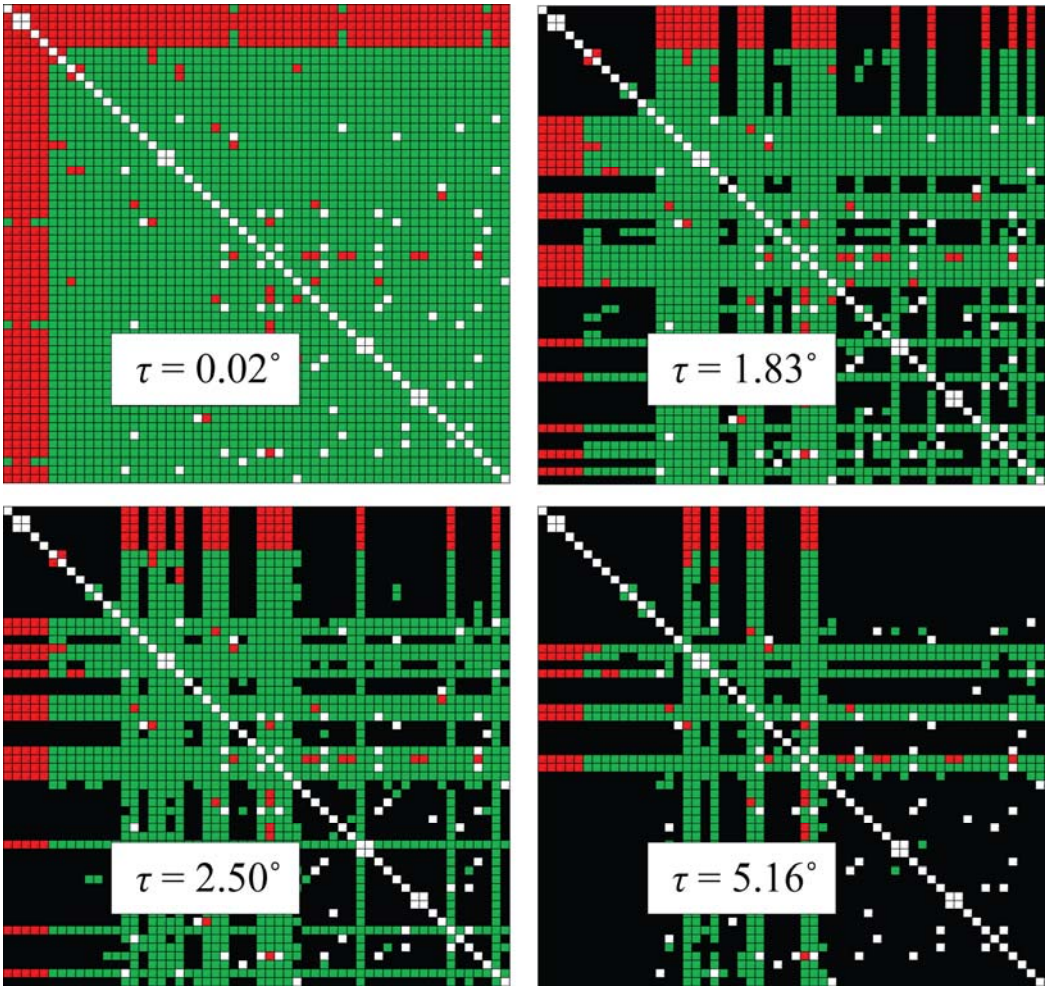

Figure 4.35: Effect of angle threshold on event detection with 31-order FIR filtering; black squares represent outages that were undetected due to insufficient angle changes.

Outside of the problems due to the 18-37 and 15-54 lines, there are still several instances where a double line outage is incorrectly identified. Using the $\tau = 0.02$

degree results, there are 21 additional misranked outages. Detailed information about these misranked outages is provided in Table 4.2, with the entries sorted by *RankOutaged*.

Table 4.2: Misrankings that do not involve 18-37 or 15-54 using FIR filtering with a filter order of 31 and angle threshold of 0.02 degrees.

From bus (line 1)	To bus (line 1)	Circuit ID (line 1)	From bus (line 2)	To Bus (line 2)	Circuit ID (line 2)	*RankOutaged*	*NADOutaged*
39	38	1	39	38	2	105	0.0165
44	41	1	44	41	2	105	0.0137
24	44	1	47	53	1	44	0.0151
17	19	1	31	28	1	33	0.0549
31	38	1	17	19	1	30	0.0270
31	28	1	39	40	1	26	0.0205
10	13	1	10	39	1	23	0.0193
10	39	1	47	53	1	23	0.0212
31	28	1	47	53	1	21	0.0173
44	41	2	5	44	1	21	0.0129
44	41	1	5	44	1	19	0.0208
31	28	1	21	48	1	9	0.0062
31	28	1	21	48	2	9	0.0353
31	28	1	14	34	1	8	0.0162
31	28	1	13	55	1	6	0.0197
32	29	1	3	40	1	5	0.0571
31	28	1	35	31	1	4	0.0201
24	44	1	33	50	1	2	0.0209
39	38	1	31	38	1	2	0.0336
39	38	2	31	38	1	2	0.0200
44	41	1	24	44	1	2	0.0227

The two entries with the highest *RankOutaged* values in Table 4.2 correspond to the outage of parallel lines between buses 39 and 38 and between buses 44 and 41. To better understand what happens in the case of the outage of the two lines between buses 39 and 38, Figure 4.36 shows the observed angle changes due to the outage along with the best match in angle changes due to this event (i.e., $\tilde{\tilde{P}}^{*}_{39-38,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,39-38,1} + \tilde{\tilde{P}}^{*}_{39-38,2}\Delta\tilde{\boldsymbol{\theta}}_{39-38,2}$ as defined in (3.41)) and the best match in angle

changes due to the simultaneous outage of line 39-38, circuit 1, and line 39-47, circuit 1 ($\tilde{\tilde{P}}^{*}_{39-38,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,39-38,1} + \tilde{\tilde{P}}^{*}_{39-47,1}\Delta\tilde{\boldsymbol{\theta}}_{39-47,1}$).

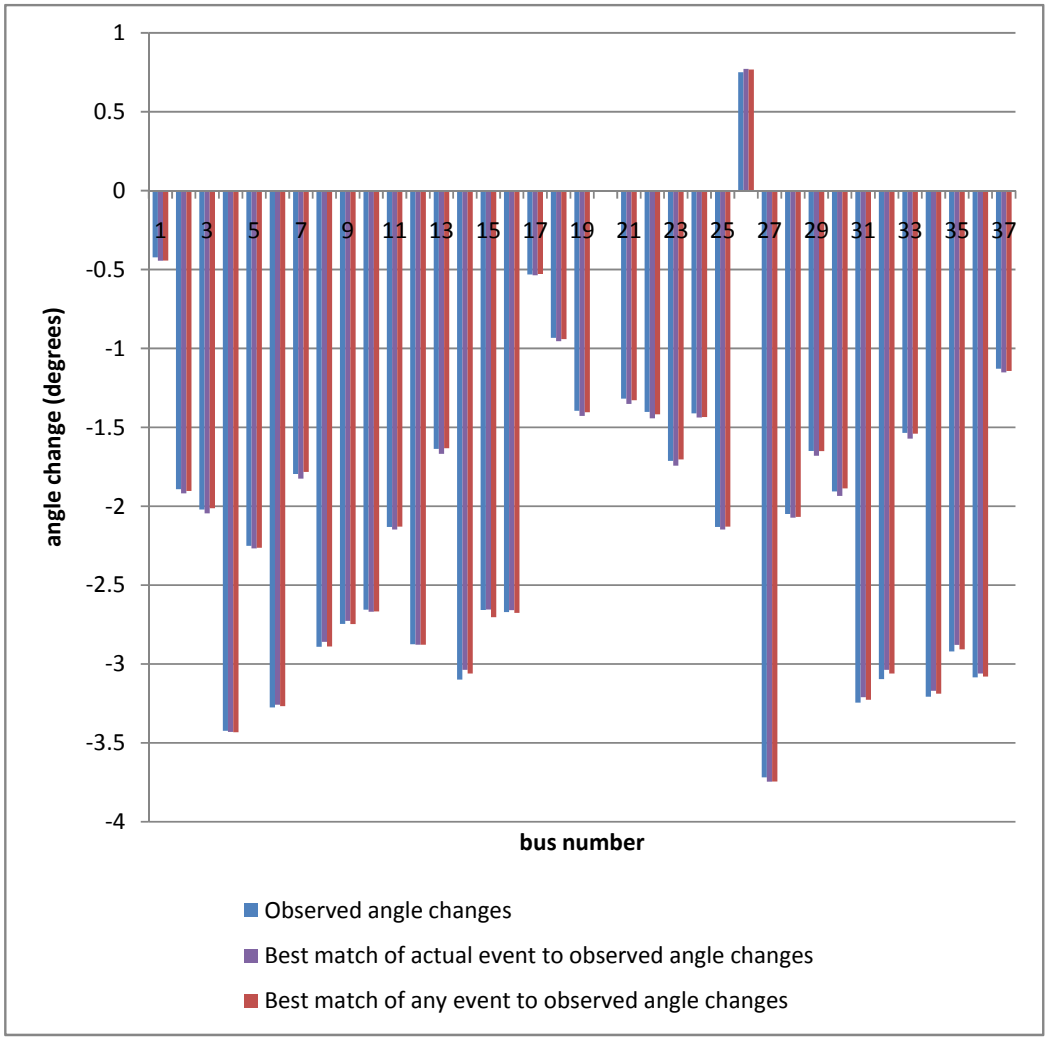

Figure 4.36: A detailed look at the observed and event matching angle changes due to the outage of the parallel lines between buses 39 and 38.

The predicted angle changes due to both the true outaged event and the top-ranked event are very close, as shown in the figure. In addition, the top-ranked event includes one of the parallel lines; in fact, in each of the 104 double-line outages which have lower *NAD* values than the true line outage, either line 39-38, circuit 1 or line 39-38, circuit 2 is one of the two outages. Because only one parameter is used to fit

parallel lines to observed angle changes (as discussed in Section 3.4.1), the extra degree of freedom when other lines are considered as possibilities allows the algorithm to more closely match the observed angles. Restriction of the double-outage event set to highly probable outages should help to mitigate this effect. In addition, if the top 10 results were presented to an operator, the fact that one of the 39-38 lines shows up in each entry would suggest that something has happened to one of those lines.

The highest *RankOutaged* value for a nonparallel set of lines is the outage of 24-44, circuit 1 and 47-53, circuit 1. In this case, all of the double-line outages with *NAD* values less than the true event *NAD* value include line 24-44, circuit 1. *RankOutaged* does not decrease as the filter order is increased to 61, indicating that the inability of the algorithm to recognize line 47-53, circuit 1 as the second outaged line is due to the mismatch between the dc power flow solution and the true system response to the outages.

The double-line outages do not exhibit any of the problematic high amplitude, low-frequency oscillations which make filtering such a crucial component of generator outage detection. Accordingly, the differences in results based on FIR and median filtering are minimal. Table 4.3 contains a summary of the detection results for each combination of filter length and angle threshold mentioned at the beginning of this section. The statistics based on FIR filtering are outside of parentheses, and the statistics based on median filtering are inside parentheses. The similarity in the results for the two filtering methods, particularly as the filter length increases, is

further indication that failure in the dc power flow assumptions is the primary cause of incorrect ranking of events.

Table 4.3: Summary of algorithm results for FIR (median) filtered angle measurements with complete bus monitoring.

Events where *RankOutaged* = 1

Filter length	Angle threshold (degrees) 0.02	1.83	2.5	5.16
3	1180 (1179)	660 (604)	493 (413)	84 (61)
9	1216 (1215)	805 (777)	645 (625)	304 (280)
31	1228 (1231)	815 (814)	660 (651)	327 (318)
61	1238 (1242)	826 (829)	663 (665)	330 (328)

Events where *RankOutaged* > 1

Filter length	Angle threshold (degrees) 0.02	1.83	2.5	5.16
3	318 (319)	136 (167)	103 (114)	19 (24)
9	288 (289)	125 (118)	102 (93)	46 (39)
31	276 (273)	120 (119)	92 (98)	50 (58)
61	266 (262)	110 (113)	89 (89)	47 (48)

Undetected Events

Filter length	Angle threshold (degrees) 0.02	1.83	2.5	5.16
3	6 (6)	708 (733)	908 (977)	1401 (1419)
9	0 (0)	574 (609)	757 (786)	1154 (1185)
31	0 (0)	569 (571)	752 (755)	1127 (1128)
61	0 (0)	568 (562)	752 (750)	1127 (1128)

In summary, double-line outage detection is substantially more likely to generate misrankings due to the extra degree of freedom in choosing two, rather than one, pre-outage flows to match expected angles to observed angles. As a result, careful attention must be paid to how the double-outage event set is defined, not only to reduce the computational effort involved in searching through the events, but also to reduce the chances of misranking the true outage event. Also, the choice of filtering method and filter length has only a slight impact on performance, as indicated in Table 4.3.

4.3.1.2 Eighteen buses monitored

In addition to testing with PMUs at all buses, tests of the double-line detection algorithm were run with the 18 bus set defined in Section 4.1.1.2. As with single-line outage detection, there are some special cases of double-line outages which deserve special consideration if some buses are unmonitored.

The first case considered is where both outaged lines are in the unmonitored system and are connected to the remaining system through two boundary buses. Let l_1 and l_2 denote the lines which are outaged. Then, because the vectors used to minimize the difference in observed and expected angles are the same in the single- and double-line outage detection (see (3.8) and (3.36)), the change in angles on the monitored system must be colinear (i.e., $\exists \alpha, \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} = \alpha \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2}$). As a result, the minimization of the difference between the observed and expected angles, (3.37), collapses to a one-dimensional minimization,

$$\begin{aligned} &\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} \left\| \Delta\boldsymbol{\theta}_{observed} - \left(\tilde{\tilde{P}}_{l,1} \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} + \tilde{\tilde{P}}_{l,2} \Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} \right) \right\| = \\ &\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} \left\| \Delta\boldsymbol{\theta}_{observed} - \left(\tilde{\tilde{P}}_{l,1} \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} + \tilde{\tilde{P}}_{l,2} \alpha \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \right) \right\| = \\ &\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} \left\| \Delta\boldsymbol{\theta}_{observed} - \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \left(\tilde{\tilde{P}}_{l,1} + \tilde{\tilde{P}}_{l,2} \alpha \right) \right\| \end{aligned} \tag{4.12}$$

which has the following solution:

$$\begin{aligned} &\left(\tilde{\tilde{P}}_{l,1} + \tilde{\tilde{P}}_{l,2} \alpha \right)^* = \frac{\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \cdot \Delta\boldsymbol{\theta}_{observed}}{\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \cdot \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1}} \\ &\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} \left\| \Delta\boldsymbol{\theta}_{observed} - \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \left(\tilde{\tilde{P}}_{l,1} + \tilde{\tilde{P}}_{l,2} \alpha \right) \right\| = \\ &\left\| \Delta\boldsymbol{\theta}_{observed} - \Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \left(\tilde{\tilde{P}}_{l,1} + \tilde{\tilde{P}}_{l,2} \alpha \right)^* \right\| \end{aligned} \tag{4.13}$$

An additional equation would allow for determination of $\tilde{\tilde{P}}_{l,1}$ and $\tilde{\tilde{P}}_{l,2}$, but the monitored system only "sees" the composite effect of the two outages, and it is

therefore not possible to determine $\tilde{\tilde{P}}_{l,1}$ and $\tilde{\tilde{P}}_{l,2}$ uniquely. On the other hand, the outage event can still be ranked by comparing the expected and observed angle changes.

Another interesting case is when one of the outaged lines connects an unmonitored portion of the system to a single bus within the monitored system. As discussed in Section 4.1.1.2, the outage of a radial line connected to one boundary bus results in no angle changes on the monitored system (i.e., $\Delta\tilde{\boldsymbol{\theta}}_{calc,\text{radial line}} = \mathbf{0}$). Referring to the boundary line as l_2, the following formulation of the angle matching minimization is obtained:

$$\begin{aligned}
&\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} \left\| \Delta\boldsymbol{\theta}_{observed} - \left(\tilde{\tilde{P}}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} + \tilde{\tilde{P}}_{l,2}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_2} \right) \right\| = \\
&\min_{\tilde{\tilde{P}}_{l,1},\tilde{\tilde{P}}_{l,2}} \left\| \Delta\boldsymbol{\theta}_{observed} - \left(\tilde{\tilde{P}}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} + \tilde{\tilde{P}}_{l,2}\mathbf{0} \right) \right\| = \\
&\min_{\tilde{\tilde{P}}_{l,1}} \left\| \Delta\boldsymbol{\theta}_{observed} - \tilde{\tilde{P}}_{l,1}\Delta\tilde{\boldsymbol{\theta}}_{calc,l_1} \right\|
\end{aligned} \tag{4.14}$$

Because there are no angle changes associated with the boundary line l_2, the minimization reduces to a single-line outage detection problem for the other line. If both lines are connected through only one bus to the monitored system, then both $\Delta\tilde{\boldsymbol{\theta}}_{calc}$ vectors are zero and there is no way to identify the outage with the algorithm.

Finally, there are instances in which two or more double-line outages are indistinguishable from one another even though the individual $\Delta\tilde{\boldsymbol{\theta}}_{calc}$ of each line are not collinear. Just as in the single-line outage case indistinguishable lines resulted from there being only two buses connecting the monitored and unmonitored system, indistinguishable double-line outages occur when there are only three buses connecting the monitored and unmonitored system (see Figure 4.37).

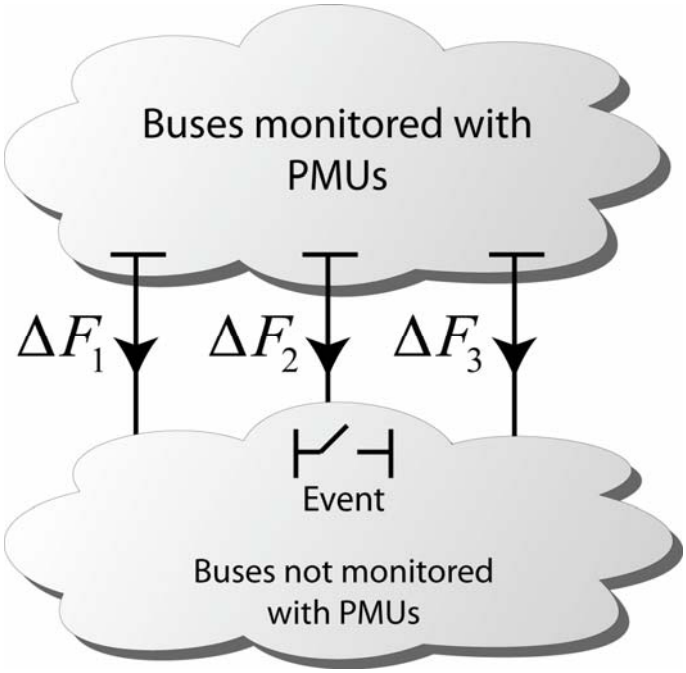

Figure 4.37: Unmonitored system connected to monitored system through three boundary buses.

Two double-line outages are indistinguishable if the following condition is true:

$$\begin{gathered} \forall a, b \in \mathbb{R}; \exists c, d \in \mathbb{R}; \\ a\Delta\boldsymbol{\theta}_{calc,l_{1,EV1}} + b\Delta\boldsymbol{\theta}_{calc,l_{2,EV1}} = c\Delta\boldsymbol{\theta}_{calc,l_{1,EV2}} + d\Delta\boldsymbol{\theta}_{calc,l_{2,EV2}} \end{gathered} \tag{4.15}$$

In (4.15), one double-line outage in the unmonitored system is referred to as *EV*1, and the other double-line outage is referred to as *EV*2. The lines constituting the outage *EV*1 are $l_{1,EV1}$ and $l_{2,EV1}$; similarly, $l_{1,EV2}$ and $l_{2,EV2}$ refer to the lines in the double-outage event *EV*2. If the relation in (4.15) is true, then when matching either event against the observed angle changes the same minimum difference will be obtained.

To determine the cases under which (4.15) holds, first let ΔF_1^{EV1} be the change in flow on boundary line 1 due to *EV*1, with similar notation to designate the other boundary flows ($\Delta F_2^{EV1}, \Delta F_3^{EV1}, \Delta F_1^{EV2}, \Delta F_2^{EV2}$ and ΔF_3^{EV2}) depicted in Figure 4.37. If no islanding results from the outages in the unmonitored system, then the bus power

injections in the unmonitored system will stay the same and the boundary flows must sum to zero for each outage:

$$\Delta F_1 + \Delta F_2 + \Delta F_3 = 0 \tag{4.16}$$

As was done in 4.1.1.2, it can be shown that the $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ vector for the outage of any single line *l* in the unmonitored system can be determined by representing the changes in boundary flows as changes in injections at the boundary buses:

$$\Delta\tilde{\boldsymbol{\theta}}_{calc,l} = \mathbf{KB}^{-1}\begin{pmatrix} \Delta F_1^l \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \leftarrow \text{boundary bus 1 row} + \\ \Delta F_2^l \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \leftarrow \text{boundary bus 2 row} + \\ \Delta F_3^l \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \leftarrow \text{boundary bus 3 row} \end{pmatrix} \tag{4.17}$$

Combining (4.16) and (4.17), $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ can be expressed in terms of ΔF_1^l and ΔF_2^l by eliminating ΔF_3^l:

$$\Delta F_3^l = -\left(\Delta F_1^l + \Delta F_2^l\right)$$

$$\Delta\tilde{\boldsymbol{\theta}}_{calc,l} = \mathbf{KB}^{-1}\begin{pmatrix} \Delta F_1^l \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 1 row} \\ \leftarrow \text{boundary bus 3 row} \\ \\ \end{matrix} + \\ \Delta F_2^l \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 2 row} \\ \leftarrow \text{boundary bus 3 row} \\ \\ \end{matrix} \end{pmatrix} \tag{4.18}$$

Equation (4.18) can then be simplified further to give $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ as the sum of two scalar-vector products:

$$\Delta\tilde{\boldsymbol{\theta}}_{calc,l} = \Delta F_1^l \mathbf{KB}^{-1} \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 1 row} \\ \leftarrow \text{boundary bus 3 row} \\ \\ \end{matrix} +$$

$$\Delta F_2^l \mathbf{KB}^{-1} \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{boundary bus 2 row} \\ \leftarrow \text{boundary bus 3 row} \\ \\ \end{matrix}$$

$$\Delta\tilde{\boldsymbol{\theta}}_{calc,l} = \Delta F_1^l \Delta\boldsymbol{\theta}_1 + \Delta F_2^l \Delta\boldsymbol{\theta}_2 \tag{4.19}$$

Combining (4.19) with (4.15), the condition under which two double-line outages are indistinguishable can be restated in terms of the $\Delta\boldsymbol{\theta}_1$ and $\Delta\boldsymbol{\theta}_2$ vectors from (4.19):

$$\begin{gathered} \forall a,b \in \mathbb{R}, \exists c,d \in \mathbb{R}, \\ a\left(\Delta F_1^{l_1,EV1}\Delta\boldsymbol{\theta}_1 + \Delta F_2^{l_1,EV1}\Delta\boldsymbol{\theta}_2\right) + b\left(\Delta F_1^{l_2,EV1}\Delta\boldsymbol{\theta}_1 + \Delta F_2^{l_2,EV1}\Delta\boldsymbol{\theta}_2\right) = \\ c\left(\Delta F_1^{l_1,EV2}\Delta\boldsymbol{\theta}_1 + \Delta F_2^{l_1,EV2}\Delta\boldsymbol{\theta}_2\right) + d\left(\Delta F_1^{l_2,EV2}\Delta\boldsymbol{\theta}_1 + \Delta F_2^{l_2,EV2}\Delta\boldsymbol{\theta}_2\right) \\ \left(a\Delta F_1^{l_1,EV1} + b\Delta F_1^{l_2,EV1}\right)\Delta\boldsymbol{\theta}_1 + \left(a\Delta F_2^{l_1,EV1} + b\Delta F_2^{l_2,EV1}\right)\Delta\boldsymbol{\theta}_2 = \\ \left(c\Delta F_1^{l_1,EV2} + d\Delta F_1^{l_2,EV2}\right)\Delta\boldsymbol{\theta}_1 + \left(c\Delta F_2^{l_1,EV2} + d\Delta F_2^{l_2,EV2}\right)\Delta\boldsymbol{\theta}_2 \end{gathered} \tag{4.20}$$

One condition under which (4.20) is true is if the coefficients on the $\Delta\boldsymbol{\theta}_1$ and $\Delta\boldsymbol{\theta}_2$ vectors match on both sides of the equation:

$$\begin{gathered} c\Delta F_1^{l_1,EV2} + d\Delta F_1^{l_2,EV2} = a\Delta F_1^{l_1,EV1} + b\Delta F_1^{l_2,EV1} \\ c\Delta F_2^{l_1,EV2} + d\Delta F_2^{l_2,EV2} = a\Delta F_2^{l_1,EV1} + b\Delta F_2^{l_2,EV1} \\ \begin{bmatrix} \Delta F_1^{l_1,EV2} & \Delta F_1^{l_2,EV2} \\ \Delta F_2^{l_1,EV2} & \Delta F_2^{l_2,EV2} \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a\Delta F_1^{l_1,EV1} + b\Delta F_1^{l_2,EV1} \\ a\Delta F_2^{l_1,EV1} + b\Delta F_2^{l_2,EV1} \end{bmatrix} \\ \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \Delta F_1^{l_1,EV2} & \Delta F_1^{l_2,EV2} \\ \Delta F_2^{l_1,EV2} & \Delta F_2^{l_2,EV2} \end{bmatrix}^{-1} \begin{bmatrix} a\Delta F_1^{l_1,EV1} + b\Delta F_1^{l_2,EV1} \\ a\Delta F_2^{l_1,EV1} + b\Delta F_2^{l_2,EV1} \end{bmatrix} \end{gathered} \tag{4.21}$$

The inverse matrix in (4.21) exists if and only if the determinant is nonzero, i.e.,

$$\begin{gathered} \Delta F_1^{l_1,EV2}\Delta F_2^{l_2,EV2} - \Delta F_1^{l_2,EV2}\Delta F_2^{l_1,EV2} \neq 0 \\ \frac{\Delta F_1^{l_1,EV2}}{\Delta F_1^{l_2,EV2}} \neq \frac{\Delta F_2^{l_1,EV2}}{\Delta F_2^{l_2,EV2}} \end{gathered} \tag{4.22}$$

The condition in (4.22) will be true if the line outages are distinguishable with respect

to the single-line outage detection algorithm (i.e., the angle change vectors associated with each line are not scalar multiples of one another). If (4.22) is false, then the double outages $EV1$ and $EV2$ may still be indistinguishable, but for other reasons (e.g., if there are three lines in parallel). If (4.22) is true, then (4.21) gives the unique values of c and d for each value of a and b that are needed to satisfy the criteria for indistinguishable double outages, (4.15). Therefore, except for those double-outage events where condition (4.22) is violated, the double-line outages in the unmonitored system connected by three buses to the monitored system will not be distinguishable.

Figure 4.38 shows a portion of the 37-bus system that, when monitored at the 18 buses defined above, has 11 indistinguishable double-line outages. The set of double-line outages that are indistinguishable is provided in Table 4.4. There are six lines within the unmonitored system, which are, using the notation (from bus number, to bus number, circuit ID), defined as: (35, 31, 1), (35, 56, 1), (56, 29, 1), (28, 29, 1), (28, 29, 2), and (31 ,28 ,1). The indistinguishable double-outage set does not include four of the line combinations: {(29, 28, 1), (29, 28, 2)}, {(29, 28, 1), (31, 28, 1)}, {(29, 28, 2), (31, 28, 1)}, and {(35, 56, 1), (56,29,1)}. What distinguishes these double-outage sets from the rest of the double-outage sets is that these sets involve two lines which have collinear $\Delta\tilde{\boldsymbol{\theta}}_{calc,l}$ vectors. As a result, the condition in (4.22) is not true and the condition for two double-line outages to be indistinguishable is not satisfied.

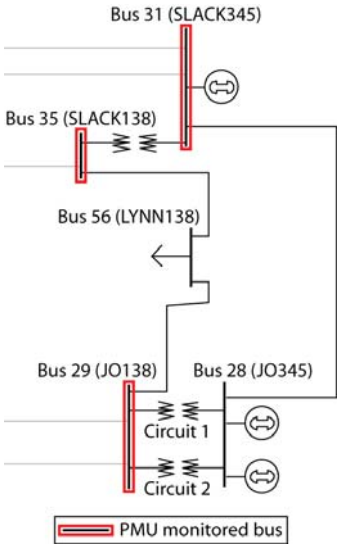

Figure 4.38: Portion of the 37-bus system that, when monitored with the 18 PMU set, has 11 indistinguishable double-line outages.

Table 4.4: Indistinguishable double-line outages for the portion of the 37 bus system illustrated in Figure 4.38

Line 1 From bus	Line 1 To bus	Line 1 Circuit ID	Line 2 From bus	Line 2 To bus	Line 2 Circuit ID
35	31	1	35	56	1
28	29	1	35	31	1
28	29	1	35	36	1
28	29	1	56	29	1
28	29	2	35	31	1
28	29	2	35	36	1
28	29	2	56	29	1
31	28	1	35	31	1
31	28	1	35	36	1
31	28	1	56	29	1
35	31	1	56	29	1

Due to the complicating factors associated with having unmonitored buses, the algorithm uses several steps to ensure each possible double outage is properly handled when attempting to match the event with the observed angle changes. Figure 4.39 shows the process that is undertaken for each event that is detected. The set of diamonds on the right-hand side of the figure constitute a cascaded classification system to determine how the double-line outage event should be modeled and fit to the observed angle changes.

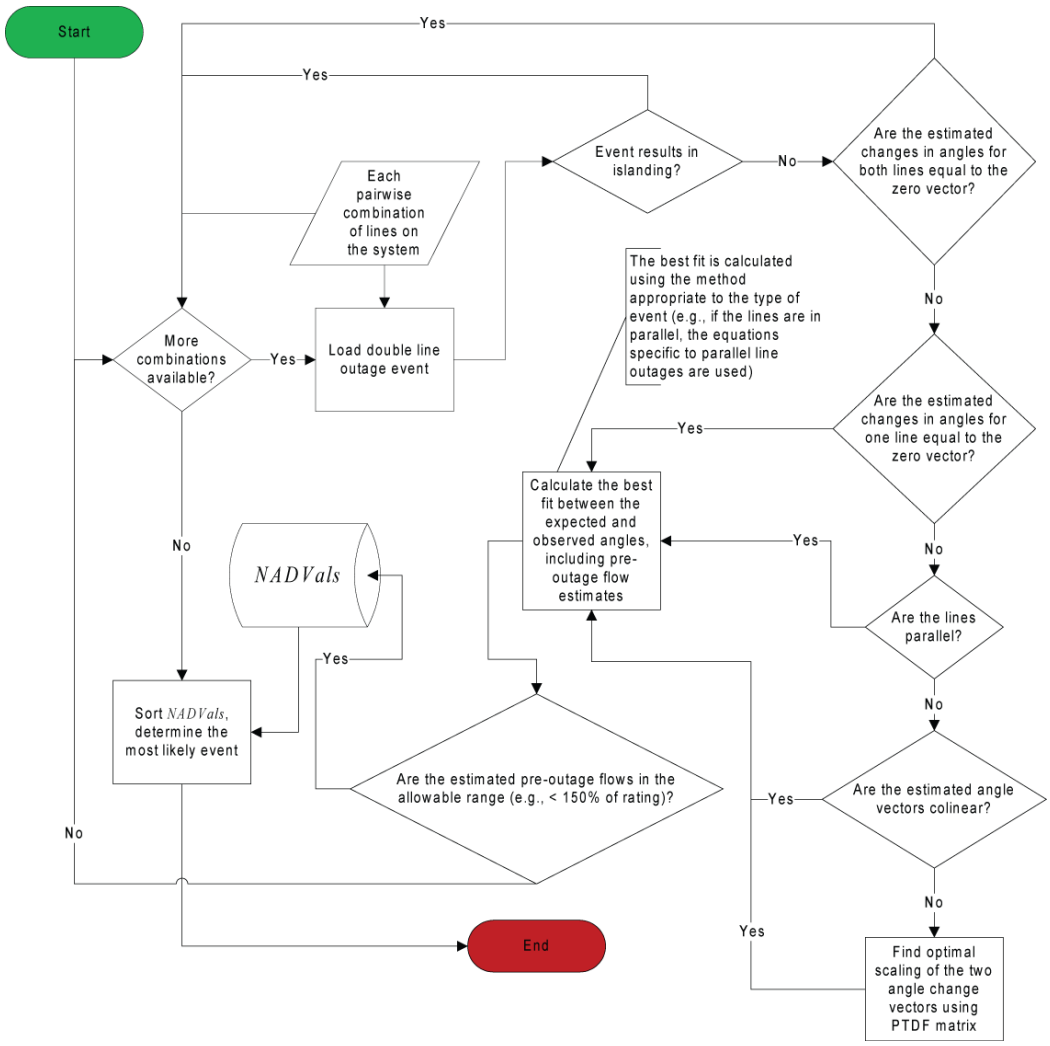

Figure 4.39: Program flow for the double-line outage detection implementation, including cascading event classification

To test the performance of the algorithm with the lower number of PMUs, filter lengths of 3, 9, 31, and 61 were used along with angle thresholds of 0.006, 0.71, 1.04, and 2.31 degrees. These angle thresholds were chosen based on detecting 100%, 75%, 50%, and 25% of the events using FIR filtering of length 31.

The results based on FIR filtering are very similar to those results obtained with full PMU placement. Figure 4.40 provides the *RankOutaged* matrix for FIR filtering of order 31 using the reduced PMU set. Comparing this with Figure 4.35, the

performance of the algorithm with the reduced PMU set is very similar to using the full PMU set. Also, the negative impact of the lines between buses 18 and 37 and the lines between buses 15 and 54 is clearly not mitigated by reducing the number of PMUs. Figure 4.41 shows that the *RankOutaged* values also change very little when median filtering is used on the reduced PMU measurements.

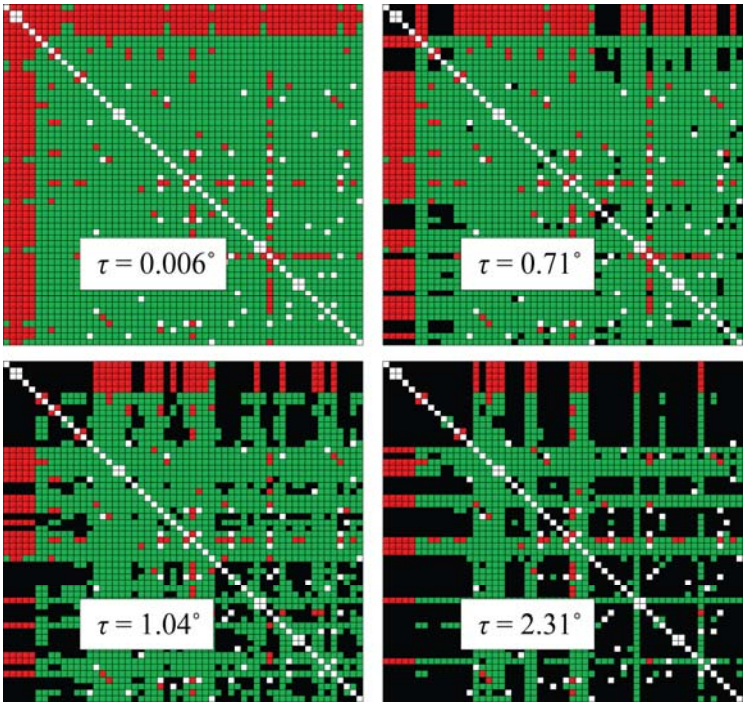

Figure 4.40: FIR filtered results, using a filter order of 31 and the set of 18 PMUs defined in Section 4.1.1.2.

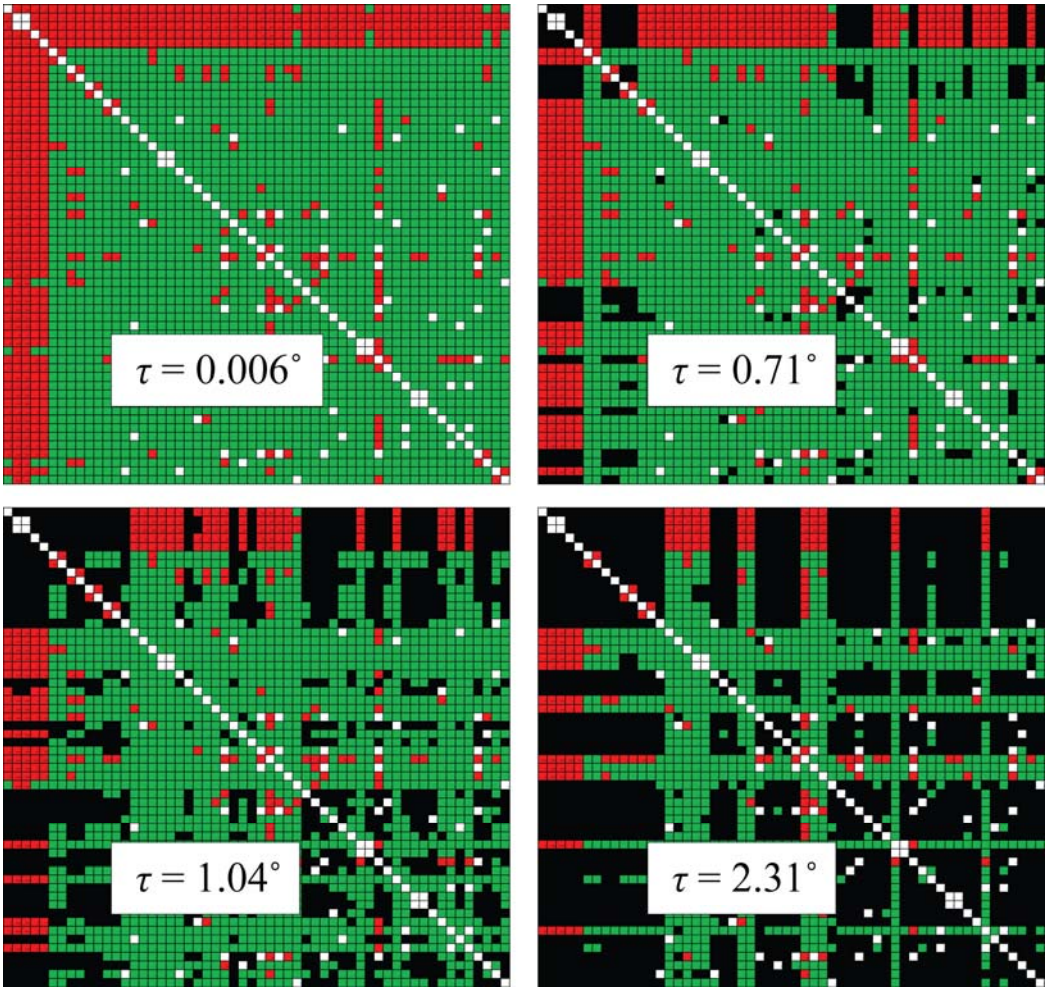

Figure 4.41: Median filtered results, using a window length of 31 and the set of 18 PMUs defined in Section 4.1.1.2.

As with the single-line outage tests, the most significant difference between having a full PMU deployment and 18 PMUs is the number of events which are indistinguishable. To quantify this effect, the *SharedOutaged* quantity defined in (4.10) is examined for the two different PMU deployments with 31-order FIR filtering and τ set to the minimum needed to detect all events using the full PMU deployment. Figure 4.42 shows that the ability of the algorithm to differentiate between the true outage event and other events on the system is significantly impaired by a decrease in PMU deployment. From an operational standpoint, higher *SharedOutaged* values

require listing $SharedOutaged + 1$ events rather than just listing one event, but this can still be useful information for operators if the results are visualized.

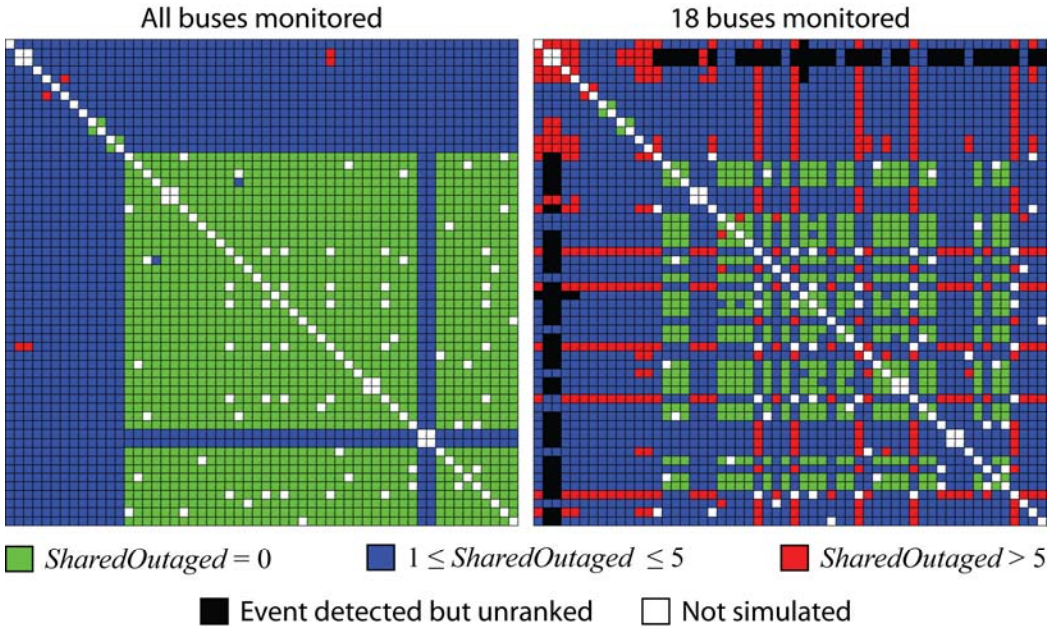

Figure 4.42: Effect of reducing the PMU set on *SharedOutaged* with 31-order FIR filtering and τ set to the minimum value needed to detect the maximum number of events.

The black squares in Figure 4.42 designate double-line outages which were detected but not ranked. This can only occur when $\Delta\tilde{\boldsymbol{\theta}}_{calc}$ is equal to $\mathbf{0}$ for one or both of the lines outaged. If only one of the lines has $\Delta\tilde{\boldsymbol{\theta}}_{calc} = \mathbf{0}$, then (4.14) is applied, and if the necessary pre-outage flow on the other line is greater than 1.5 times the line rating, then the event is removed from consideration (as discussed at the end of Section 3.4.4) and not ranked. If both $\Delta\tilde{\boldsymbol{\theta}}_{calc}$ vectors are $\mathbf{0}$, then it is not possible to perform the minimization in (3.37) and the event is not ranked.

4.3.2 All nonislanding double-line outages that share at least one terminal

The results presented in Section 4.3.1 are based on the simultaneous, unforced outage of each pairwise combination of lines on the system. In real power systems,

two lines are unlikely to go out simultaneously unless they are close to one another, which is the basis for the filtering mechanism described in Section 3.2.4. To ascertain the performance of the algorithm for more realistic double-line outage events, additional tests were run using only those outages where at least one terminal bus is shared between the two lines. The set of events $\mathfrak{E}$ tested to match with the observed angle changes was similarly restricted. This restriction reduces the number of possible double line outage events from 1504 to 115.

The algorithm was first tested with full PMU deployment using the same filter lengths and angle thresholds from Section 4.3.1.1. A summary of the algorithm performance is provided in Table 4.5. Comparing these results with those from Table 4.3, there are significantly fewer misranked events. Figure 4.43 shows the *RankOutaged* values for the full double-outage event set (left side) and the set of double outages where a terminal bus is shared (right side) using the same row/column ordering and color scheme as the figures in Section 4.3.1. The misranked double-line outages which share a terminal bus are still concentrated in the first few rows—11 of the 16 misranked outages include lines between buses 18-37 and 15-54. As in the full event case, there are still problems with parallel line outages, although the *RankOutaged* value is reduced due to the smaller size of the event detection set $\mathfrak{E}$.

Table 4.5: Summary of algorithm results for FIR (median) filtered angle measurements with complete bus monitoring considering only double line outages where a terminal bus is shared

Events where *RankOutaged* = 1

Filter length	Angle threshold (degrees) 0.02	1.83	2.5	5.16
3	93 (93)	54 (53)	41 (34)	13 (8)
9	98 (97)	59 (61)	51 (55)	24 (27)
31	99 (101)	62 (65)	52 (57)	28 (27)
61	99 (99)	62 (62)	52 (52)	28 (26)

Events where *RankOutaged* > 1

Filter length	Angle threshold (degrees) 0.02	1.83	2.5	5.16
3	22 (22)	10 (11)	7 (6)	0 (1)
9	17 (18)	10 (8)	10 (6)	7 (1)
31	16 (14)	9 (6)	9 (4)	3 (4)
61	16 (16)	9 (9)	9 (9)	3 (5)

Undetected Events

Filter length	Angle threshold (degrees) 0.02	1.83	2.5	5.16
3	0 (0)	51 (51)	67 (75)	102 (106)
9	0 (0)	46 (46)	54 (54)	84 (87)
31	0 (0)	44 (44)	54 (54)	84 (84)
61	0 (0)	44 (44)	54 (54)	84 (84)

Figure 4.43: *RankOutaged* results for FIR filtering with angle threshold of 0.02 degrees, for the full double-outage set (left side) and the double outages which share a terminal bus (right side). Red squares indicate where *RankOutaged* is greater than one and green squares indicate where *RankOutaged* equals 1. White squares indicate events which were not considered.

If the PMU coverage is reduced to 18 PMUs, the ranking of the events is not significantly impacted. In addition, reducing the event set to only those double-line

outages that share a terminal bus reduces the increase in *SharedOutaged*. Figure 4.44 shows the change in *SharedOutaged* for each of the double-line outages as the PMU set is reduced to 18 PMUs. Although there is a small increase in *SharedOutaged* values as the number of PMUs is reduced, the maximum *SharedOutaged* value is 4. This is much lower than the maximum *SharedOutaged* value of 14 which is obtained if all events are considered and 18 PMUs are used (shown in Figure 4.42).

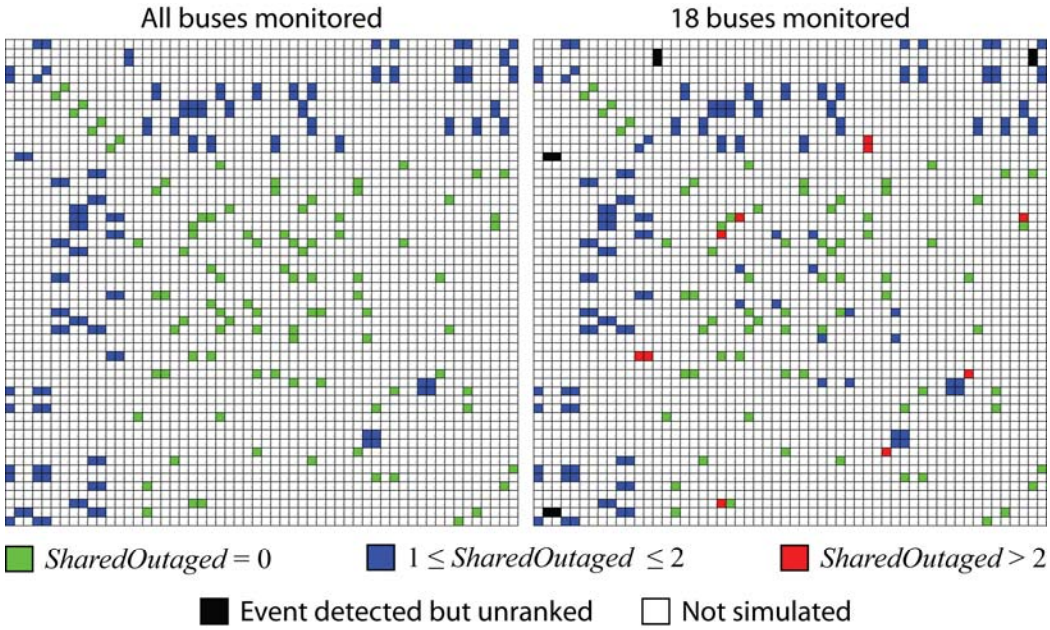

Figure 4.44: Effect of reducing the PMU set on *SharedOutaged* with 31-order FIR and τ set to the minimum value needed to detect all events, with the event set restricted to those lines that share a terminal bus.

5 PHASOR MEASUREMENT UNIT PLACEMENT FOR EVENT DETECTION

5.1 Overview

If the goal of deploying additional PMUs is to improve event detection, then there are some key characteristics of a given set which must be considered:

- Detection of events of interest
- Proper ranking of detected events
- Differentiation between events

Optimized placement of PMUs requires two components: an objective function to maximize (or minimize) and a method of searching the space of PMU sets so that the objective function is maximized (or minimized). The first section below deals with defining an objective function to meet certain criteria, while the second section provides details on simulated annealing (SA), the method used to search the set space. Finally, the third section presents some results showing how exhaustive search and SA perform in placing PMUs based on the objective functions.

5.2 Objective Function Definition

To evaluate each PMU placement set, an objective function must be defined which provides a numerical fitness value for each PMU set. For the purposes of PMU placement, it is assumed that filtering is applied to the raw measurements such that the observed angle change vector is equal to the difference in steady state angles (i.e., all oscillations around the new steady state angles are completely attenuated). To determine the observed angle change vector corresponding to each event, the

postevent angle after 20 s was subtracted from the original angle, with the measurements filtered using a 61-order FIR filter. On the other hand, the angle threshold, which determines which events are detected by the algorithm, is allowed to vary so that the optimization function is maximized. The objective functions below include τ in their definition, and the impact of τ on the objective function is discussed in each individual section.

5.2.1 Objective 1: Maximize the number of correctly ranked events

The simplest objective function is to evaluate the PMU set by maximizing the number of events that are detected and ranked correctly. An optimization problem which would meet this objective is

$$\begin{aligned} X_{Rank1}\left(e,\tau,PMUSet\right) &= \begin{cases} 1 & , \ RankOutaged\left(e,\tau,PMUSet\right)=1 \\ 0 & , \ \text{event undetected} \end{cases} \\ ObjFunc_{Rank1}\left(PMUSet\right) &= \max_{\tau} \sum_{e\in\mathcal{E}} X_{Rank1}\left(e,\tau,PMUSet\right) \\ PMUSet^{*} &= \arg\max_{PMUSet} ObjFunc_{Rank1}\left(PMUSet\right) \end{aligned} \tag{5.1}$$

where $\mathcal{E}$ is the set of events used to evaluate the PMU placement. Note that the set $PMUSet^{*}$ can have several members, corresponding to several PMU placements which result in the same maximum value of the objective function. In (5.1), the dependence of *RankOutaged* on the angle threshold is made explicit in determining *RankOutaged* and X_{Rank1}. Notice that if an event is undetected, it does not impact the objective function; as a result, this particular objective function does not penalize a particular PMU placement for resulting in misranked events. Therefore, the inner maximization over τ should always choose τ equal to the minimum angle change over all angles for each event, since choosing a different τ would reduce the number

of possible events where *RankOutaged* = 1. The primary shortcomings of this objective function are the lack of a penalty for misranking of events and the lack of a penalty for having events which share the same rank.

5.2.2 Objective 2: Maximize the number of correctly ranked events with *no* misranked events

Building upon the first objective function, it is possible to define an objective function which penalizes a PMU placement if it results in misranked events. One such optimization definition is

$$\begin{aligned} &X_{NoMisrank}\left(e,\tau,PMUSet\right)= \\ &\quad\begin{cases} 1 & , \ RankOutaged\left(e,\tau,PMUSet\right)=1 \\ -\infty & , \ RankOutaged\left(e,\tau,PMUSet\right)>1 \\ 0 & , \ e \text{ undetected given } \tau \text{ and } PMUSet \end{cases} \\ &ObjFunc_{NoMisrank}\left(PMUSet\right)=\max_{\tau}\sum_{e\in\mathfrak{E}}X_{NoMisrank}\left(e,\tau,PMUSet\right) \\ &PMUSet^{*}_{NoMisrank}=\max_{PMUSet}ObjFunc_{NoMisrank}\left(PMUSet\right) \end{aligned} \tag{5.2}$$

Once again, the dependence of *RankOutaged* on the angle threshold is made explicit in determining *RankOutaged* and $X_{NoMisrank}$. Unlike with the previous objective function, the optimal value of τ for the inner maximization is not necessarily the minimum angle change. Consider the case where there is an event (such as the outage of one of the 18-27 lines) that is misranked no matter how many angle measurements are taken due to failure of the dc power flow assumptions. In this case, the only way to improve the objective function would be to stop detecting this event, which will be the case if τ is set sufficiently high. Therefore, the inner maximization must be explicitly calculated for this optimization problem. Fortunately, this does not pose much of a computation burden due to the finite number of τ values which impact the

results; in the worst case, the number of evaluations needed to exhaustively solve the inner maximization is equal to the number of events.

This objective function is a significant improvement over the first, in that it explicitly accounts for the problem of misranking by removing from consideration any combination of PMU placement and τ that would result in a misranked event. The number of events that share the same rank is still not accounted for, but this optimization provides a much better match with the objectives mentioned at the beginning of the chapter.

5.2.3 Objective 3: Maximize the number of correctly ranked events with *no* misranked events and *no* events with the same rank

Taking the optimization problem from the previous section one step further, it is also possible to remove any PMU set and threshold combination if it results in events that share the same rank. The optimization to be solved in this case is:

$$
\begin{aligned}
&X_{NoShared}\left(e,\tau,PMUSet\right)= \\
&\begin{cases}
1 & ,RankOutaged\left(e,\tau,PMUSet\right)=1 \text{ and} \\
 & SharedOutaged\left(e,\tau,PMUSet\right)=0 \\
-\infty & ,RankOutaged\left(e,\tau,PMUSet\right)>1 \text{ or} \\
 & SharedOutaged\left(e,\tau,PMUSet\right)>0 \\
0 & ,e \text{ undetected given } \tau \text{ and} \\
 & PMUSet
\end{cases} \\
&ObjFunc_{NoShared}\left(PMUSet\right)=\max_{\tau}\sum_{e\in\mathbb{E}}X_{NoShared}\left(e,\tau,PMUSet\right) \\
&PMUSet^{*}_{NoShared}=\arg\max_{PMUSet}ObjFunc_{NoShared}\left(PMUSet\right)
\end{aligned}
\tag{5.3}
$$

This is a much more restrictive objective function, in that only events which are ranked one and have no other events sharing that rank count towards the objective function value for a particular PMU placement. While this last objective function

meets each of the objectives stated in Section 5.1, not all instances of *SharedOutaged* greater than zero are problematic if visualization of the event detection is used to present the results (e.g., in the case of parallel lines). As a result, this objective function may result in PMU placements which are poorer than other placements from a practical sense.

5.3 Searching Methods

5.3.1 Exhaustive search

One method of determining the optimal placement of PMUs with respect to these objective functions is to perform an exhaustive search. This can become computationally intractable, depending on how *PMUSets* is constructed. If, for instance, *PMUSets* consists of all possible PMU combinations on an *N*-bus system, then the number of possible PMU configurations to evaluate is:

$$\left|PMUSets\right| = \sum_{n=1}^{N} \binom{N}{n} = \sum_{n=1}^{N} \frac{N!}{n!(N-n)!} \tag{5.4}$$

Even if the number of PMUs to be placed is fixed, the size of *PMUSets* is likely to make exhaustive search computationally intractable. For instance, placing 4 PMUs on a 37-bus system requires the search of a space that is of size 66 045. If each evaluation of the objective function is carried out in 0.5 s, then it would still take 9 h to process the whole set of possible PMU combinations. In the results presented below, exhaustive search was only used for the placement of up to 6 PMUs due to the amount of time needed to perform exhaustive search. The good thing about exhaustive search is that it is guaranteed to find the global optimal value of the

objective function and all PMU placements that attain this optimal value, unlike the simulated annealing method described in the next section.

5.3.2 Simulated annealing

An alternative method which has been used to perform large-scale combinatorial optimization in other PMU placement applications [18, 20] is simulated annealing (SA) [73]. This method is based on the properties of material annealing (heating and then gradual cooling), where atoms within a material will tend to reach a globally minimum energy state if the material is cooled slowly enough. Leveraging this idea to perform optimization, the following algorithm is defined:

1. Initialize the algorithm temperature and configuration:

$$\begin{aligned} T &\leftarrow T_{initial} \\ PMUSet &\leftarrow PMUSet_{Initial} \\ f_{max} &\leftarrow ObjFunc\left(PMUSet_{Initial}\right) \\ PMUSet_{max} &\leftarrow PMUSet_{Initial} \\ StallIterations &\leftarrow 0 \\ IterationsAtTemperature &\leftarrow 0 \end{aligned} \tag{5.5}$$

2. Until exit condition is reached:
    a. While *IterationsAtTemperature* < *NumIterationsAtTemperature* and exit condition has not been met
        i. Evaluate a new PMU placement:

$$
\begin{aligned}
&StallIterations \leftarrow StallIterations + 1 \\
&PMUSet' \leftarrow Neighbor\left(PMUSet\right) \\
&\Delta \leftarrow ObjFunc\left(PMUSet'\right) - ObjFunc\left(PMUSet\right) \\
&\text{if } \Delta > 0 \\
&\quad PMUSet \leftarrow PMUSet' \\
&\quad \text{if } ObjFunc\left(PMUSet'\right) > f_{max} \\
&\qquad f_{max} \leftarrow ObjFunc\left(PMUSet'\right) \\
&\qquad PMUSet_{max} \leftarrow PMUSet' \\
&\qquad StallIterations \leftarrow 0 \\
&\text{elseif } e^{\left(\frac{\Delta}{T}\right)} > Uniform\left(\left[0,1\right]\right) \\
&\quad PMUSet \leftarrow PMUSet'
\end{aligned}
\tag{5.6}
$$

ii. Check for exit condition

$$
\begin{aligned}
&\text{if } StallIterations = StallIterationsMax \\
&\quad \text{exit algorithm}
\end{aligned}
\tag{5.7}
$$

iii. If exit condition has not been met, increment *IterationsAtTemperature*:

$$IterationsAtTemperature \leftarrow IterationsAtTemperature + 1 \tag{5.8}$$

b. Update the temperature and reset *IterationsAtTemperature*

$$
\begin{aligned}
&T \leftarrow 0.99 \times T \\
&IterationsAtTemperature \leftarrow 0
\end{aligned}
\tag{5.9}
$$

The algorithm parameters which have not been previously defined are as follows:

- T, $T_{initial}$: T is the current "temperature" in the optimization. High temperatures correspond to hot materials in which the atoms or molecules can move around freely, whereas low temperatures correspond to cold materials where movement is restricted to those movements which will directly lower the energy of the system. The key feature of simulated annealing is that it allows for suboptimal changes in the current *PMUSet* so that the algorithm

does not get stuck in local maxima, as would be the case using a pure hill-climbing algorithm such as steepest ascent. The mathematical model of this behavior is represented by the conditional acceptance of poorer PMU placements in (5.6) based on the Boltzmann distribution. $T_{initial}$, the initial value of the temperature, is chosen so that essentially any change in the PMU placement is accepted at the start of the algorithm. For the results presented below, this was set to 10 times the minimum of P, the number of PMUs in the placement set, and $N - P$, the number of buses without PMUs.

- *IterationsAtTemperature*, *NumIterationsAtTemperature*: The *IterationsAtTemperature* counter keeps track of how many *PMUSet*'s are tested at the current temperature. After *NumIterationsAtTemperature* PMUSet's have been tested at a given temperature, the temperature is adjusted. Several iterations need to be run at each temperature to allow the current state of the system to settle into the probability distribution associated with a given temperature [73]. For the results presented below, *NumIterationsAtTemperature* was set to the minimum of P, the number of PMUs in the placement set, and $N - P$, the number of buses without PMUs.
- $PMUSet_{Initial}$: An initial guess at what the optimal PMU placement should be; for a finite number of PMUs, the only restriction is that the reference bus is within this set and that the cardinality of the set is the number of PMUs to be placed. For the results presented below, this was set to include PMUs at the

referenced bus and P-1 other buses chosen at random, where P is the number of PMUs to be placed.

- *ObjFunc*: One of the three objective functions defined in Section 5.2.
- *StallIterations*: Keeps track of the number of iterations since the last increase in the objective function due to a new optimal PMU placement set.
- *StallIterationsMax*: The maximum number of iterations allowed after the last increase in the objective function before the algorithm exits. In generating the results presented below, this was set to 1000 times the minimum of P, the number of PMUs in the placement set, and $N - P$, the number of buses without PMUs. The main purpose of setting this to a high number is to ensure the local maximum is attained as the temperature reaches low values.
- $Neighbor(PMUSet)$: A function which takes a PMU placement set and returns a new PMU placement set which is a variation on the previous set. For the results presented below, this function randomly selects a PMU from a nonreference bus and moves it to an adjacent bus without a PMU, where two buses are adjacent if there is a line connecting the buses.
- $PMUSet_{max}$, f_{max}: The best PMU placement set found at any point during the run of the algorithm, and the optimal objective function value associated with the best PMU placement set.
- $Uniform([0,1])$: A sample from the uniform distribution with a lower bound of zero and upper bound of one.

Using these parameters, SA produces a PMU placement set $PMUSet_{max}$ which would ideally be equal to the true optimal $PMUSet^*$. There is no guarantee that the SA algorithm will reach the true optimal, but as shown in the results below the algorithm does a good job of finding an objective function maximum that is close to the true optimal while drastically reducing the amount of computation needed relative to exhaustive search.

5.4 Results for Optimal PMU Placement to Detect Single-Line Outages

As mentioned in the beginning of Section 5.2, the impact of filtering can be neglected in the placement problem by first determining $\Delta\theta_{observed}$ for each outage using a high order filter of the simulated angles and calculating the changes in angles several seconds after the outage occurs. Using these $\Delta\theta_{observed}$ values, *RankOutaged, AboveThreshold*, and *SharedOutages* are determined for a given PMU placement set *PMUSet* and angle threshold τ. These quantities were evaluated using the methods described in Section 3.2 and then used to evaluate the objective functions defined in Section 5.2. Exhaustive search was used for the placement of three and six PMUs, and SA was used for placement of 3, 6, 9, 18, and 27 PMUs. SA was run three times for each PMU placement problem, and the best result is reported below. The computation time given in the results tables for SA includes all three runs of SA.

5.4.1 Three PMUs placed by exhaustive search and simulated annealing

The first set of results looks at the optimal placement of three PMUs on the system with each objective function using both exhaustive search and simulated annealing. The total number of possible PMU placements for three PMUs is 630.

Table 5.1: Placement of three PMUs using exhaustive search and SA

Objective function	Search method	Highest objective function value	Corresponding *PMUSet*(s)	Time taken to perform optimization (seconds)
$ObjFunc_{Rank1}$	Exhaustive	40	{18, 31, 37}	1.31
$ObjFunc_{Rank1}$	SA	40	{18, 31, 37}	42.71
$ObjFunc_{NoMisrank}$	Exhaustive	15	{1, 9, 31}	1.29
$ObjFunc_{NoMisrank}$	SA	14	{12, 31, 32}	42.03
$ObjFunc_{NoShared}$	Exhaustive	3	{1, 13, 31} {1, 28, 31} {1, 38, 31} {5, 13, 31} {13, 18, 31} {13, 21, 31} {13, 28, 31} {13, 37, 31} {13, 38, 31} {13, 48, 31} {17, 38, 31} {19, 38, 31} {21, 38, 31} {28, 30, 31} {28, 33, 31} {38, 48, 31}	1.58
$ObjFunc_{NoShared}$	SA	3	{13, 31, 37} {13, 31, 48} {17, 31, 38}	43.31

As seen in Table 5.1, for the $ObjFunc_{Rank1}$ objective function both SA and exhaustive search determine the global optimal of {18, 31, 37} which results in 40 outages being ranked correctly. The time taken to perform the optimization is much lower for exhaustive search in this case due to the small number of possible configurations for a 37-bus case. The results for the other two objective functions are

also shown in Table 5.1; the SA results are slightly suboptimal for one of the objective functions, $ObjFunc_{NoMisrank}$, and the time to perform the optimization is significantly higher, indicating that for very small PMU deployments on this system simulated annealing is not as efficient as exhaustive search. Also noteworthy is the large number of PMU placements that optimize the $ObjFunc_{NoShared}$ objective function. Because so many of the PMU sets optimize this objective, SA can easily find optimal PMU placements, and this is reflected in the fact that all three runs of SA result in optimal PMU placements. The $ObjFunc_{NoShared}$ results also illustrate a key shortcoming of SA—namely, that it does not provide an exhaustive list of optimizing placement sets, whereas exhaustive search does.

5.4.2 Six PMUs placed by exhaustive search and simulated annealing

Next, the placement of six PMUs on the system is examined using both exhaustive search and simulated annealing. Placing six PMUs pushes the bounds of how many PMUs can be placed in a reasonable amount of time with exhaustive search; doing the same search for seven PMUs would take approximately four times longer, resulting in search times on the order of hours rather than minutes. Going beyond seven PMUs would require days of computation if exhaustive search were used.

The results shown in Table 5.2 illustrate the difference in computation time for SA relative to exhaustive search—there is a roughly fivefold decrease in computation time. This shows that SA scales much better with the number of PMUs to be placed than exhaustive search does. In addition, SA has many more parameters which can be

tweaked, particularly the exit condition *StallIterationsMax*, which can be modified to reduce this computational time.

Table 5.2: Placement of six PMUs using exhaustive search and SA

Objective function	Search method	Highest objective function value	Corresponding *PMUSet*(s)	Time taken to perform optimization (seconds)
$ObjFunc_{Rank1}$	Exhaustive	49	{1,3,5,21,55,31}	589.95
$ObjFunc_{Rank1}$	SA	46	{3,5,10,21,31,54} {13,21,24,31,34,37}	98.04
$ObjFunc_{NoMisrank}$	Exhaustive	31	{10,15,21,48,56,31}	620.55
$ObjFunc_{NoMisrank}$	SA	21	{12,24,28,30,31,40}	90.03
$ObjFunc_{NoShared}$	Exhaustive	9	{1,21,24,28,38,31} {1,24,28,38,48,31} {1,24,28,38,53,31}	710.72
$ObjFunc_{NoShared}$	SA	8	{19,28,30,31,38,48}	118.15

Table 5.2 also showcases the other main issue in using an approximate optimizer such as simulated annealing—the SA result is suboptimal for all three objective functions, and is significantly suboptimal for the $ObjFunc_{NoMisrank}$ objective function. One possible reason for this behavior is that the objective function has infinitely steep derivatives in the presence of misranked events. This can prevent the SA algorithm from transitioning between configurations, particularly as the temperature gets low and the algorithm stops allowing suboptimal transitions. One solution would be to reformulate the objective function so that the penalty for having misranked events is less than infinity, although the choice of penalty would have to be tailored to each system and event set in order to continue avoiding misranking. Another option would be to try additional combinatorial optimization methods (e.g., genetic algorithms [74], particle swarm [75], or a hybrid of several methods as in [76]). Ultimately, there is no

way to avoid the fact that with an extremely large search space tradeoffs must be made between computational effort and optimality.

5.4.3 Higher numbers of PMUs placed by simulated annealing

The final set of results illustrate the performance of SA for the placement of PMUs in which exhaustive search is computationally infeasible. SA was run once for each combination of objective function and number of PMUs *P*. The results of these runs are given in Table 5.3. The steady increase in optimal values of the three objective functions indicates that SA is finding better PMU placements as the number of PMUs increases. This fits well with the results from Section 4.1.1.2, where the reduction in PMUs from 37 to 18 led to slightly poorer rankings and more events with nonzero *SharedOutaged* values. Moreover, these results show that if 27 PMUs are placed on this system, there exists a PMU placement and angle threshold such that 51 of the 56 events are correctly detected without any misrankings. This confirms the usefulness of the single-outage detection algorithm for reduced PMU placements, particularly if misranking is to be minimized. Also, although these results are based on the dynamic simulations with the initial conditions given in 0, running SA with different system configurations (e.g., summer peak, winter peak, midday, midnight, etc.) would be feasible due to the small amount of time needed to perform the search. On the other hand, testing a wide variety of system configurations would be prohibitively expensive from a computation standpoint if a more exhaustive search approach is used.

Table 5.3: Placement of 9, 18, and 27 PMUs by simulated annealing

Objective function	Number of PMUs (P)	Highest objective function value	Time taken to perform optimization (seconds)
$ObjFunc_{Rank1}$	9	47	65.33
$ObjFunc_{NoMisrank}$	9	26	56.41
$ObjFunc_{NoShared}$	9	11	55.74
$ObjFunc_{Rank1}$	18	52	148.76
$ObjFunc_{NoMisrank}$	18	38	121.96
$ObjFunc_{NoShared}$	18	24	172.83
$ObjFunc_{Rank1}$	27	54	76.52
$ObjFunc_{NoMisrank}$	27	51	62.87
$ObjFunc_{NoShared}$	27	37	84.99

6 GPU-BASED VISUALIZATION OF PMU DATA

6.1 Overview

The high data rate of phasor measurement units requires sufficiently fast rendering if real-time visualization is to be achieved. Graphical processing units (GPUs) have shown remarkable progress in the past decade, providing unprecedented computational power and programmability for advanced graphics applications. The use of GPUs to render PMU data is a natural fit between the high data rate of PMUs and the high rendering rate of GPU-based visualization algorithms. Accordingly, two applications have been developed and are presented in the remainder of this chapter. The first section considers how one of the more common visualization techniques, bus data contouring, can be significantly accelerated by using GPU rendering. The second section discusses some additional processing of the data that can make the data visualization more content-rich. Analysis of these techniques shows how combining cutting-edge visualization techniques with cutting-edge measurement techniques can provide significant improvements in situational awareness.

6.2 Implementation of Contouring on the GPU

6.2.1 Introduction to contouring on the GPU

Based on the recommendations of the August 2003 blackout report [1], a heavy emphasis has been placed on improving situational awareness in control centers around the world. One visualization technique which has been used to increase situational awareness has been contouring [40]. This technique, also known as scattered data interpolation (see [77] for a survey of the state-of-the-art), aims to

provide a weather-map-like visualization of the power grid, showing various bus- or substation-based data throughout an area of interest. This type of visualization has been used in many different fields, e.g., meteorology [78] and medical imagery [79], and its usefulness has been demonstrated for power system applications through human factors testing [42].

One key similarity between the methods used to generate power system contours in the past has been a reliance on the CPU to do the necessary contour calculations. CPUs, which are optimized for general purpose computing, are not well-suited for the parallel calculations that are inherent in some contouring algorithms. Several incremental improvements have been made, but CPU-based contouring is fundamentally constrained by the serial nature of CPU program execution. On the other hand, graphics processing units (GPUs), which are present in most modern computer systems, are ideally suited to the computations needed to generate contours because of the parallel nature of GPU program execution. Furthermore, some work has already been done in demonstrating the effectiveness of using GPUs to perform real-time interpolation of three- and four-dimensional data [48]. This work suggests that there is a potentially large benefit to using GPUs in two-dimensional power system contours.

In order to investigate the possible performance gains of using GPU- rather than CPU-based contouring for power system visualizations, a prototype GPU-based contouring algorithm has been developed. The remainder of this chapter discusses this work, and is divided into several sections. Section 6.2.2 provides a brief

overview of the contouring problem. Section 6.2.3 discusses some of the fundamental aspects of GPU programming, while Section 6.2.4 provides the algorithm and implementation details. Section 6.2.5 provides some performance results for contouring a real power system area as parameters such as influence radius and contour resolution are varied. Section 6.2.6 discusses some of the key advantages and disadvantages of moving contouring to the GPU, and Section 6.3 details several benefits and potential applications of GPU-based contouring.

6.2.2 Contouring background

Formally, the purpose of contouring (or scattered data interpolation) in the power system context is to find a function $F(x, y)$ such that for each bus k, located at position (x_k, y_k),

$$F(x_k, y_k) = f_k \tag{6.1}$$

where f_k is the value to be contoured (e.g., voltage), known to be a specific value at each bus k. This value is typically obtained from a state estimator or directly from a measurement device such as a PMU. There are an infinite number of possible functions that can satisfy this constraint, so additional restrictions are useful in reducing the set of possible functions. One additional constraint imposed on F in this application is that F must be "smooth," i.e., continuous and once differentiable [80].

While there are many different methods which can be used to determine the contouring function F (see [77]), this work focuses on inverse distance weighted methods due to their intuitive nature and ease of parallelization. The intuition behind

inverse distance weighted methods is that the function value at any point should be the weighted average of all points with known values. In this method of contouring, the function F is defined as follows:

$$F(x,y) = \frac{\sum_{k=1}^{N} w_k(x,y) f_k}{\sum_{k=1}^{N} w_k(x,y)} \tag{6.2}$$

where $w_k(x,y)$ is known as the weighting function. The weighting function first proposed for this scheme, which gives rise to the contouring algorithm known as Shepherd's method [81], is

$$\begin{aligned} d_k &= \sqrt{(x-x_k)^2 + (y-y_k)^2} \\ w_{k,SM}(x,y) &= \frac{1}{d_k^2} \end{aligned} \tag{6.3}$$

This weighting function gives a nonzero value of $w_k(x,y)$ for all values of x and y. Because each bus k has a nonzero contribution to every point on the screen, using the weighting function of Equation (6.3) is characterized as a global contouring algorithm.

Global methods such as Shepherd's method are computationally burdensome due to the need to consider each bus's contribution at each pixel on the screen. As a result, several local methods have been proposed in the literature [80]. For the implementation of contouring on the GPU, the local weighting function developed by Franke and Little is used:

$$w_{k,FL}(x,y) = \begin{cases} \left[\frac{(R-d_k)}{Rd_k}\right]^2 & , d_k < R \\ 0 & , d_k \geq R \end{cases} \tag{6.4}$$

where d_k is defined in (6.3). For this local weighting function, only pixels within a distance R of bus k will be affected by the value f_k. This weighting function also maintains continuity and first-order differentiability of the interpolating function F [82] and reduces to the weighting function of (6.3) as $R \to \infty$.

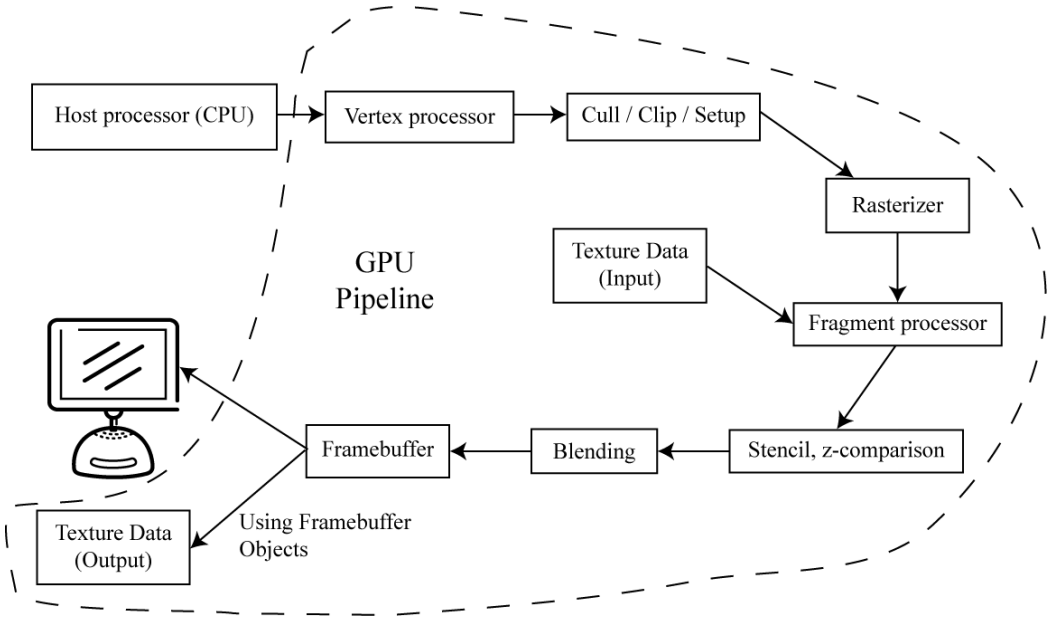

Figure 6.1: The GPU rendering pipeline.

6.2.3 GPU Programming [83]

6.2.3.1 The rendering pipeline

GPU programming is significantly different from CPU programming, and this must be taken into account when adapting any algorithm for GPU implementation. The basic GPU pipeline, illustrated in Figure 6.1, was originally constructed to efficiently render polygons, lines, and points to a graphical display. The basic steps in the GPU rendering pipeline are [83] as follows:

1. Polygon vertex coordinates, viewable area definitions, color values, texture coordinates, and textures are sent to the GPU via OpenGL calls from the CPU.

2. The vertex processor transforms coordinates defined via function calls from the CPU in step 1 into coordinates relative to the display. Also, vertices are grouped into primitives (points, lines, and triangles) for the later stages of the pipeline.
3. The Cull / Clip / Setup stage takes the primitives that are output from the vertex processor and removes unviewable objects. Also, any primitives that are only partially inside the viewable area are clipped.
4. The rasterization stage determines which fragments are covered by each primitive. Fragments are the rasterized pieces of the polygon and correspond to a single pixel of the final output. Also, per-vertex values such as colors and texture coordinates are linearly interpolated across each primitive and assigned to each enclosed fragment.
5. The fragment processor runs on each fragment, performing texture lookups and determining the final color that the fragment will be. For this contour implementation, two nondefault fragment processors are used to change the behavior of this step.
6. Once fragment colors have been assigned by the fragment processor, several tests (e.g., stencil and z-compare) are run on the fragments to see if they should be discarded. If the pixel passes all tests, then it is blended with the current framebuffer entry at its location and the final pixel value is written to the framebuffer. This write to the framebuffer is the final stage of the OpenGL pipeline. For traditional graphics applications, the framebuffer

represents the pixels on the current video display. Framebuffer objects (FBOs) make it possible to draw into a texture in video memory rather than directly to the video display [84]. This texture can then be read back on a subsequent rendering pass. This technique is used in the GPU-based contouring implementation, as discussed below.

6.2.3.2 Important fragment processor characteristics

Two of the programmable units on the GPU are the vertex processor and the fragment processor, which allow for custom processing of vertices and fragments in stages 2 and 5 of the graphics pipeline. The default vertex processor is used in the contouring implementation, so it is not discussed any further. On the other hand, two custom fragment processors are used for contour generation.

One important aspect of fragment processor programming is that there can be fragments at different stages of the pipeline at any given time. As a result, the fragment processor is not allowed to write to any locations in video memory except for the framebuffer at the end of stage 5. Otherwise, each fragment might have to wait for another fragment to complete its processing before it can be sent out to the framebuffer. In other words, if the framebuffer is conceptualized as an array of bytes, then the fragment processor can only output data to one particular array index in the framebuffer, and this array index cannot be changed from within the fragment processor. Although it cannot write directly to texture memory, the fragment processor is capable of reading directly from texture memory; in fact, this is how traditional polygonal texturing is performed for graphics applications. Because of

these features, it is helpful to think of a fragment processor as having a large set of memory addresses it can read from, and only one memory address it can write to, with no overlap between these address spaces. This is fundamentally different from CPU programs, where programs routinely read and write to the same memory address.

Although fragment processors must be programmed with a restrictive set of operations, particularly when it comes to memory reads and writes, their sheer processing power makes them ideal for speeding up computations that can be performed in parallel. For instance, modern GPUs such as the NVIDIA GeForce 7-series used in these experiments are capable of performing over 165 billion floating point operations per second (gigaflops or Gflops), whereas a CPU of the same generation, such as a dual-core Pentium 4, is only capable of around 24.6 Gflops [45]. More advanced chips, such as the recently released GeForce 8800 GTX, have theoretical processing speeds in excess of 500 Gflops. For cases where the GPU can be run near its processing limits, this difference in processing power allows GPUs to easily outperform CPUs.

There are two key issues involved in fully taking advantage of the GPU's computational power. First, memory reads should be performed as sequentially as possible in order to take advantage of hardware caches built into the GPU. The GPU contouring implementation is designed to ensure coherent texture reads by making all reads from rectangular textures, where each texel is accessed in a sequential fashion from the top-left to the bottom-right. The other key issue in programming for the GPU is to use algorithms which are computationally limited (as opposed to I/O

limited). Contouring algorithms clearly fall into this category, as the bulk of the time spent in creating a contour is based on the need to evaluate (6.4) for each pixel within the influence region of each bus. Also, because the processing elements of the GPU work in parallel, defining the algorithm using for/next loops allows for simple implementation as a fragment processor program.

6.2.4 Algorithm definition and implementation details

6.2.4.1 Definition of the contouring algorithm

Based on Equations (6.2) and (6.4), one definition of the contouring algorithm which is well-suited for GPU implementation is as follows:

1. For each point (x, y)
   - 1.1. $N(x, y) \leftarrow 0$
   - 1.2. $D(x, y) \leftarrow 0$
   - 1.3. $F(x, y) \leftarrow 0$
2. For each bus k
   - 2.1. For each point (x, y) such that $w_{k,FL}(x, y) \geq 0$
     - 2.1.1. $N(x, y) \leftarrow N(x, y) + w_{k,FL}(x, y) f_k$
     - 2.1.2. $D(x, y) \leftarrow D(x, y) + w_{k,FL}(x, y)$
3. For each point (x, y) such that $D(x, y) > 0$
   - 3.1. $F(x, y) = \dfrac{N(x, y)}{D(x, y)}$
4. For display to the screen, each value of F is mapped to a color as discussed in [40] and stored as the color value for point (x, y).

Table 6.1: Algorithm implementation on the CPU and the GPU

Algorithm Steps	CPU Implementation	GPU Implementation
1	Allocate space for *N* and *D* in main system memory.	Allocate space for *AccumTexture* and *ContourTexture* on the GPU.
2	For each bus, calculate and store the numerator and denominator values within each bus's influence region.	Draw a circle of radius *R* around each bus with the video output redirected to *AccumTexture*. Use a custom fragment shader to calculate and store the *N* values in the red channel and the *D* values in the green channel. Use additive blending to perform accumulation as each bus is drawn.
3	Iterate through each element in the *N* and *D* arrays and divide the numerator by the denominator. Look up the corresponding color in the color map and write this color to a bitmap.	Draw a rectangle of size *Res* with its texture set to *AccumTexture* and the video output redirected to *ContourTexture*. Using a custom fragment shader, divide the red channel values of *AccumTexture* by the green channel values. For each pixel, look up the corresponding color in the color map and write this color to *ContourTexture*.

Each of the algorithm steps given above can be translated into a set of instructions on the CPU or GPU, as shown in Table 6.1. For the GPU implementation, the C programming language was used with the GLUT library [85] for OpenGL programming and the Cg language was used for fragment processor programming [86]. The implementation of the algorithm was split into three stages, explained in detail below.

6.2.4.2 Implementation step 1—allocation of textures and framebuffer objects

To begin, texture memory is allocated for two textures: *AccumTexture* and *ContourTexture*. *AccumTexture* is created as a GL_RGBA16F_ARB formatted texture, which allocates four components (red, green, blue, and alpha) for each texel. Each component stores an IEEE-formatted 16-bit floating point value. *ContourTexture* is allocated as a standard 32-bit texture containing 8-bit red, green,

blue, and alpha components for each texel. *AccumTexture* stores $N(x,y)$ in its red channel and $D(x,y)$ in its green channel. *ContourTexture* stores the finished, color-mapped contour as a texture which can then be rendered to the display. Both textures are created with the same dimensions, denoted as *Res*. This size is the resolution of the contour to be created (e.g., if a contour is to be created which exactly fits into a 640×480 window, then the two textures will have dimensions 640×480). Because the algorithm requires rendering to these textures rather than simply reading from them, a framebuffer object is created that allows the GPU to write to the textures [84]. The framebuffer objects used to write to each texture are named *AccumFBO* and *ContourFBO*. The allocation of texture memory occurs only when the desired contour resolution changes (typically when a window is resized), which reduces the amount of time spent allocating memory on the GPU.

6.2.4.3 Implementation step 2—accumulation of the numerator and denominator values for each pixel

Once the textures are allocated, *AccumFBO* is set as the output framebuffer for the GPU. By setting *AccumFBO* as the output location, all pixels drawn by the GPU are sent to *AccumTexture* rather than the video display. After *AccumFBO* is attached, *glClear()* is called, which performs steps 1.1 and 1.2 of the algorithm by clearing the red and green channels of *AccumTexture*. Next, a custom fragment processor is bound to the GPU that takes as arguments the position of the bus that is currently being drawn, (x_k, y_k), and writes out the values $w_{k,FL}(x,y)f_k$ to the red channel and $w_{k,FL}(x,y)$ to the green channel. The final setup step is to enable additive blending on the GPU, which performs the assignment and sum operations

needed in steps 2.1.1 and 2.1.2 of the algorithm. Once all the setup operations are completed, for each bus k, a circle of radius R is drawn around the point (x_k, y_k). After this has been done for each bus in the system, the red channel of *AccumTexture* contains $N(x, y)$ and the green channel contains $D(x, y)$ for all points in the contour.

6.2.4.4 Implementation step 3—evaluation of the interpolating function and color mapping

For the next step in the contouring process, *ContourFBO* is attached as the render target for the GPU. A call to glClear() is then made to execute step 1.3 of the algorithm. Next, *AccumTexture* is set as the current texture to be read within the fragment processor of the GPU. A custom fragment program is then bound to the GPU which performs steps 3.1 and 3.2. Finally, a rectangle of size *Res* is drawn with texture coordinates assigned so that each point in *AccumTexture* corresponds to one output pixel. Once this draw operation has completed, *ContourTexture* contains the color-mapped contour. This texture can then be drawn to the screen by binding *ContourTexture* as the current texture and rendering an appropriately sized rectangle using the default fragment processor.

6.2.5 GPU-based contouring algorithm performance results

For the results given below, the test system used consisted of an AMD Athlon 64 X2 Dual Core 3800+ CPU with an NVIDIA GeForce 7600 GT GPU. Timings for the GPU algorithm were determined using the high-resolution Windows performance counter [87], with an average taken over 100 contour renderings.

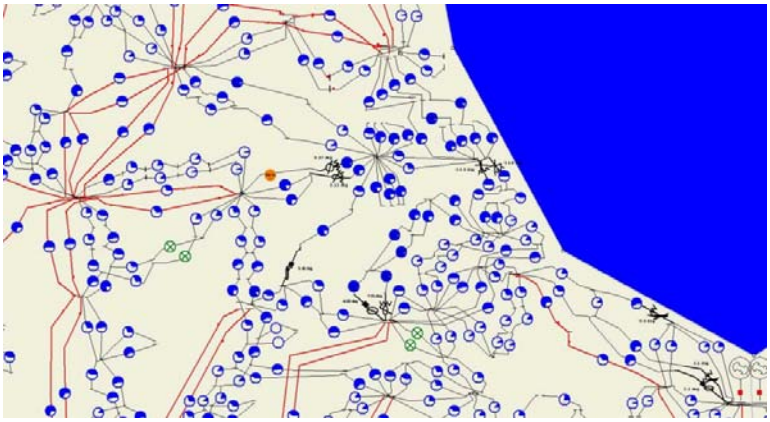

Figure 6.2: One-line diagram of contour area.

The power system one-line diagram used as the basis for the results and figures shown below is provided in Figure 6.2. The number of buses that influence the contour area ranges from 199 (for an influence radius of 1) to 818 (for an influence radius of 400). The data that is contoured is the per unit voltage level at each bus. The color map used to convert voltage levels to colors is shown in Figure 6.3.

Figure 6.3: Contour color map relating per unit voltage to displayed color.

6.2.5.1 The effect of changes in *Res*

There are two key parameters which affect the performance of the contouring algorithm. The first of these parameters, *Res*, controls how large a contour image to create. For the highest-quality contour, *Res* should be set to the same size as the screen area the contour covers upon final rendering to the video display. However,

reducing *Res* to a smaller value results in a decrease in rendering time; therefore, a tradeoff must sometimes be made between contour resolution and rendering speed. To explore the effects of varying *Res*, the system was contoured with several different resolution levels. In these tests, *R*, the radius of influence, was held at a constant value of 100. Figure 6.4 shows an example contour rendering with *Res* set to 1024×768.

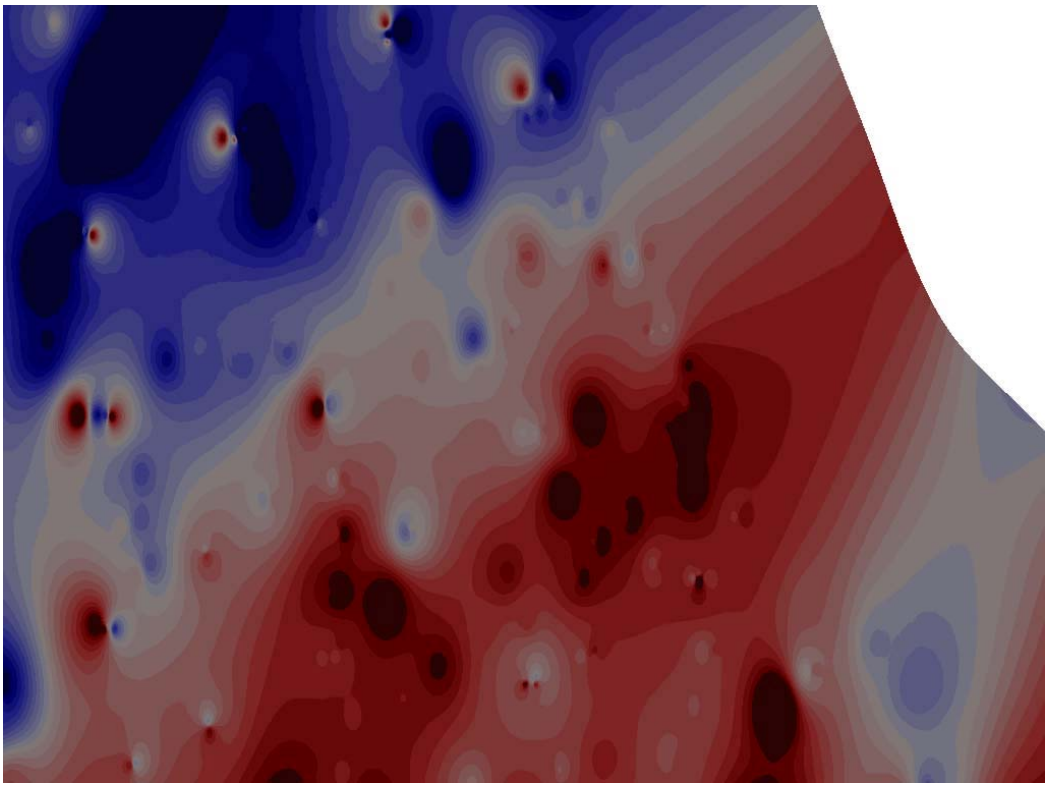

Figure 6.4: Contour rendering with resolution set to 1024x768, radius of influence set to 100.

Table 6.2 shows the timing results obtained for various contour resolutions. In order to gauge the performance of the algorithm in a typical control room setting, resolutions were chosen which are commonly used in both projection and conventional video displays.

Table 6.2: Timings for seven contour resolutions with the influence radius set to 100

Res			
Width	Height	Common Name	Time to render a contour (seconds)
3840	2400	WQUXGA	0.938
1920	1200	WUXGA	0.249
1600	1200	UXGA	0.208
1280	1024	XGA+	0.150
1024	768	XGA	0.100
800	600	SVGA	0.066
640	480	VGA	0.050

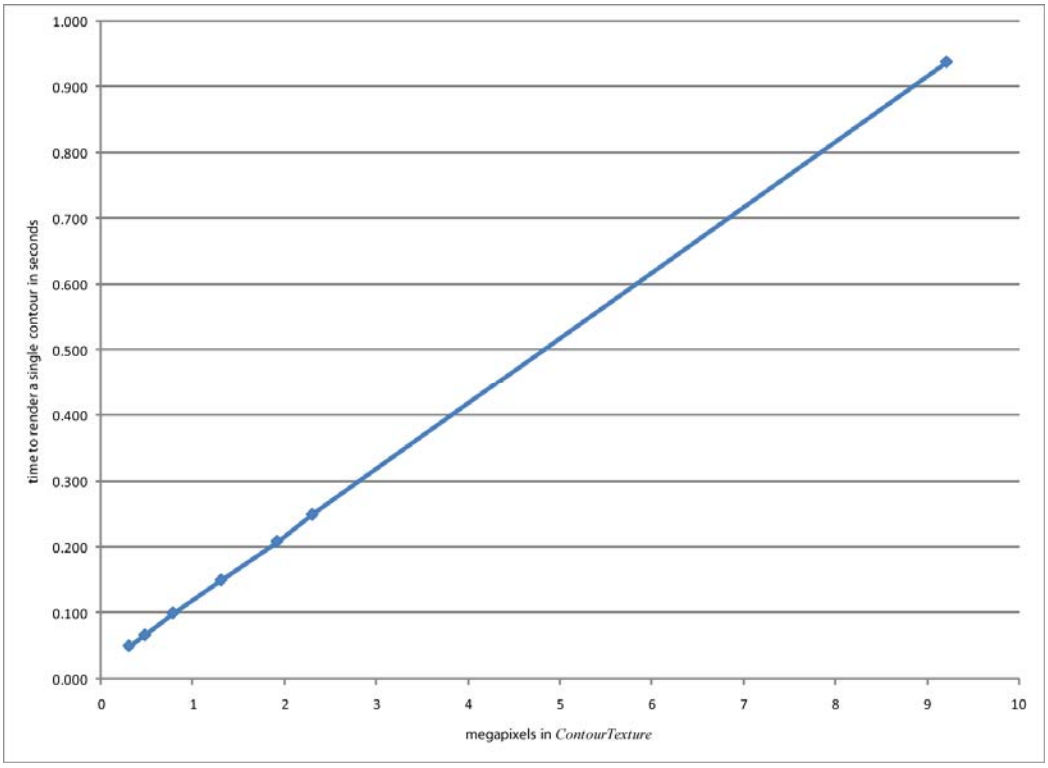

Figure 6.5: Effect of contour resolution on contour rendering times.

Figure 6.5 illustrates the effect on rendering time as the contour resolution is varied. The x-axis is the number of megapixels associated with each tested resolution, and the y-axis is the amount of time it takes to render one contour. The linear relationship between resolution and rendering time indicates that the time required to render a particular contour is on the order of the number of pixels in the contour. The fact that this linear relationship holds for resolutions ranging from

640×480 all the way up to 3840×2400 indicates that the algorithm does not suffer from scalability issues.

It is also worthwhile to consider the visual impact of changing the contour resolution. For a given screen size Res_{Screen}, the highest quality contour would be obtained by setting $Res = Res_{Screen}$, assuming the contour takes up the entire screen display. However, by setting $Res < Res_{Screen}$, the speed at which the contour is rendered can be greatly accelerated. In addition, rendering the contour at a lower resolution is not necessarily noticeable. Finally, the usage of advanced texture filtering methods can also help to improve the rendered quality of a lower resolution contour.

6.2.5.2 The effect of changes in R

The other key parameter which affects contour accuracy and rendering speed is the radius of influence, R. Because a circle of radius R is drawn around each bus (i.e., each bus affects a locus of points of radius R around its location), the contour rendering time should scale in proportion to the area of the circle drawn around each bus, i.e., quadratically in R. However, for a contour with a finite area, this relationship does not always hold.

Figure 6.6 illustrates a simple three-bus contouring example where the relationship between R and the number of processed fragments is not quadratic. The left side of the figure shows the case where the circles of radius R_1 drawn around buses 1 and 2 fit entirely within the contour area, whereas the circle around bus 3 is entirely outside the contour area. Because the circle drawn around bus 3 does not

intersect the contour area, there is no need to draw the circle around bus 3. On the right side of the figure, as the radius is increased to R_2, the circles surrounding all three buses are partially inside of the contour area. In this case, a circle is drawn around each bus, but the GPU clips away the portions of each circle that lie outside the contour area before the fragment processor stage [83]. As a result, steps 2.1.1 and 2.1.2 of the algorithm are skipped for the darkened areas of each circle. Based on Figure 6.6, it is clear that linear increase in the influence radius will cause a quadratic increase in computation time only if the set of buses influencing the contour area does not change and all bus influence regions are entirely enclosed in the contour area. Otherwise, the increase in computation time is dependent on the characteristics of the area being contoured and the distribution of the influencing buses.

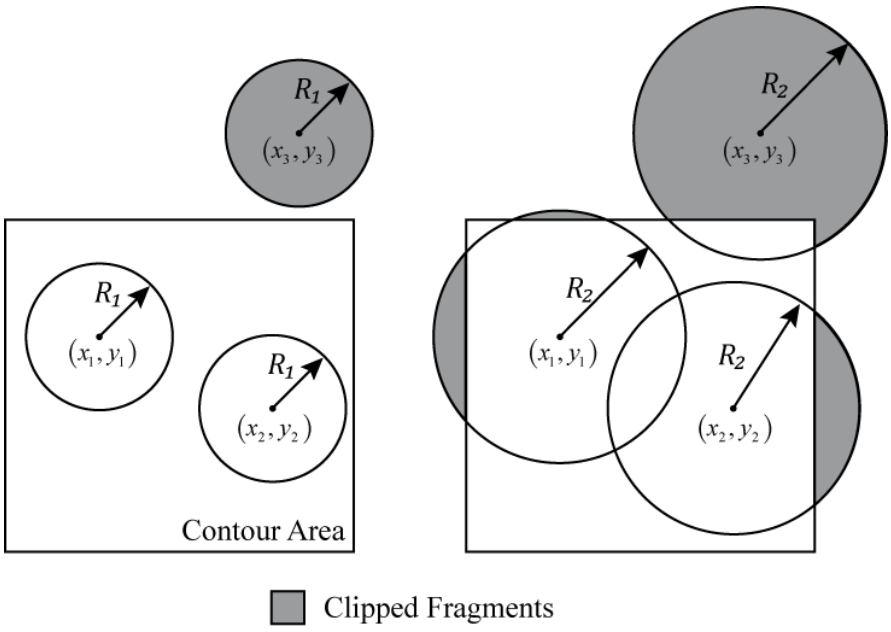

Figure 6.6: Effect of fragment culling on radius of influence effects.

To examine the effect of changing the influence radius, several timing measurements were taken with a fixed contour resolution of 1024×768 while R was

varied from 1 to 400. The results of these tests are given in Table 6.3. In addition, Figure 6.7 illustrates the complex relationship between the radius of influence and the time to construct a single contour.

Table 6.3: Timing results for several values of influence radius with a constant contour resolution of 1024×768

R	Number of buses influencing the contour area	Time to render a contour (seconds)
1	199	0.0225
5	206	0.0227
10	221	0.0229
25	274	0.0249
50	367	0.0361
75	438	0.0665
100	485	0.1001
125	514	0.1331
150	580	0.1838
200	653	0.2669
300	753	0.4251
400	818	0.4994

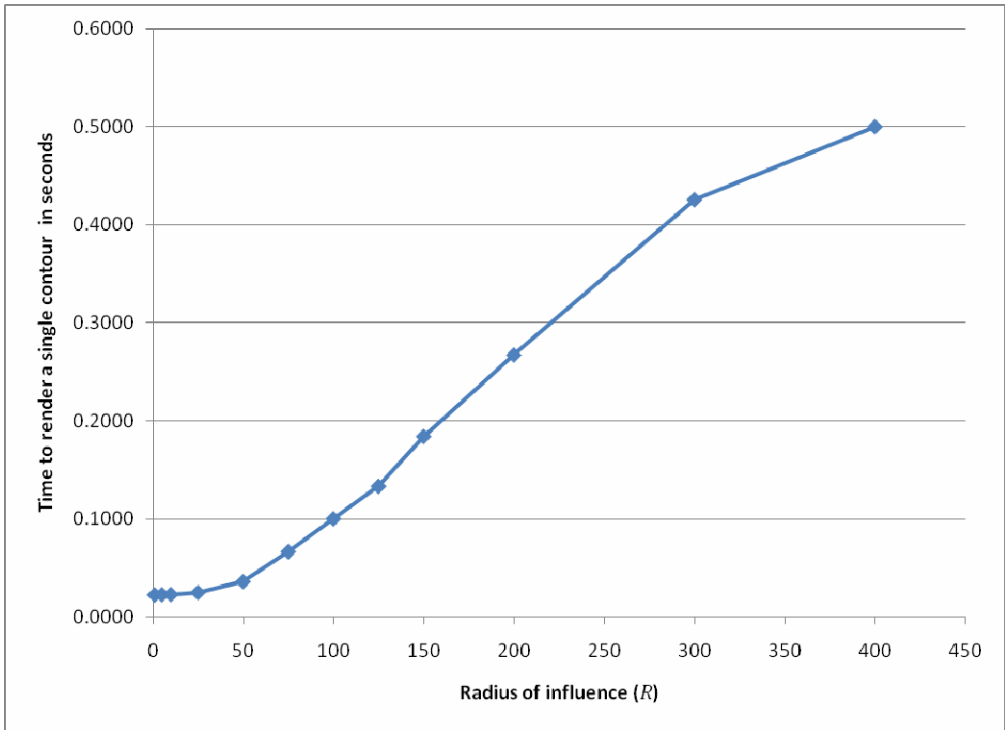

Figure 6.7: Effect of influence radius on contour rendering time with contour resolution set to 1024×768.

For the first few values of influence radius—1, 5, and 10—the time to render a contour is basically constant. This indicates that for very low influence radii, the time spent rendering a contour is based more on other factors (such as the texture lookups, OpenGL calls, etc.) than on the processing time needed to construct the contour. As the influence radius increases to values between 10 and 150, a quadratic relationship is evident. For radii greater than 150, a much more complicated relationship is demonstrated that confirms the behavior discussed above and illustrated in Figure 6.6.

As with the other tunable parameter *Res*, there is a tradeoff between how large of an influence radius is used and how fast the contours are rendered. In general, the proper choice of R will be dependent on many factors, including the density of buses within the power system under study. In fact, this is one justification for the usage of dynamic influence regions when contouring power system data, as discussed in [44].

To illustrate the visual impact of changing the influence radius, Figure 6.8 shows how reducing the influence radius from 100 to 25 affects the contour shown in Figure 6.4. In Figure 6.8, it is very easy to distinguish the circles that surround each bus, and the silhouette of the contour is a series of arc segments, typical for contours with relatively low influence radii. On the other hand, in Figure 6.4, a much smoother contour is shown with no obvious circles present. Ultimately, the individual using the contour must decide which value of R provides a good representation of the underlying data without sacrificing too much in terms of rendering speed.

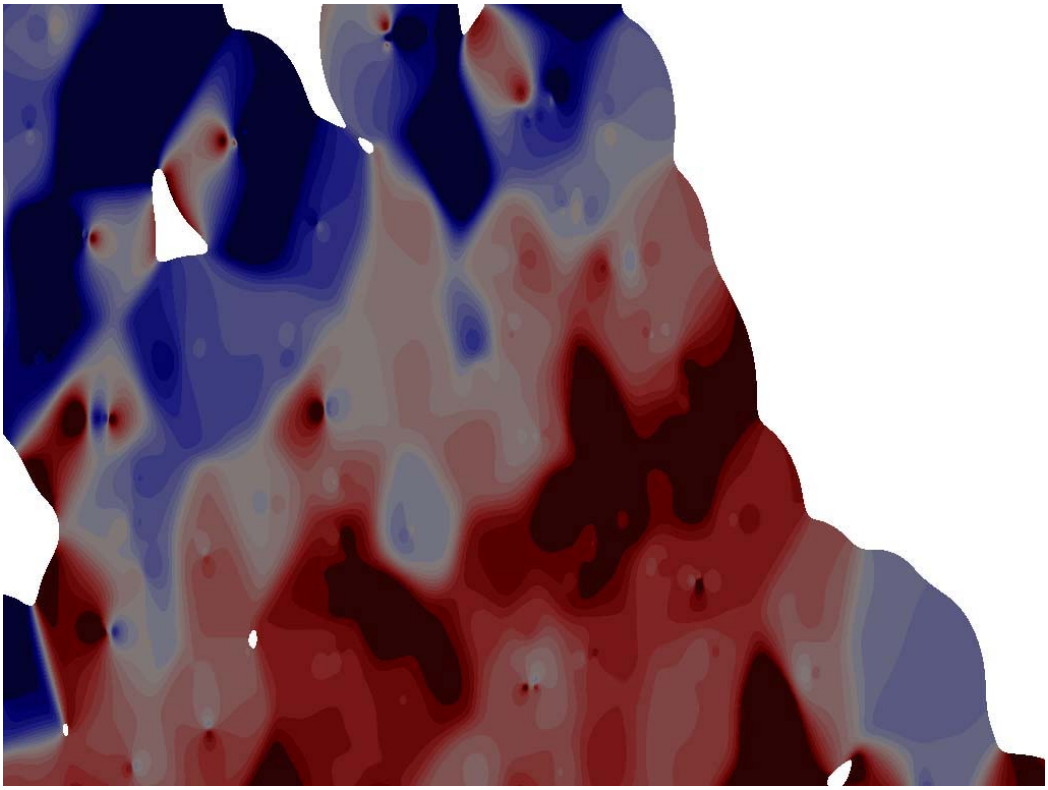

Figure 6.8: Contour rendering with resolution set to 1024×768, radius of influence set to 25.

6.2.6 Advantages and disadvantages of GPU-based contouring versus CPU-based contouring

Because implementation of a GPU-based contouring algorithm can require significant expenditure of time and resources, it must be justified as an alternative to current CPU-based contouring methods. The greatest benefit of moving contouring to the GPU is the decrease in rendering time for contours. For example, contouring the power system area in Figure 6.2 using a resolution of 640×640, with a radius of influence of 150, the GPU-based contour takes 0.102 s to render. Rendering the same contour using the CPU (with the method outlined in [44]) takes 5.10 s.

The rendering speeds of CPU-based contouring methods are significantly slower than GPU-based methods for several reasons. First, CPUs are designed to process instructions in a serial fashion, rather than for parallel operations. GPUs, on the other

hand, have always been built to process multiple vertices and fragments in parallel to quickly render polygons to the screen. For instance, the NVIDIA GeForce 7600 GT GPU used in the above tests has 12 fragment processors, each capable of performing steps 2.1.1 and 2.1.2 of the algorithm independently. On a CPU, each of these operations must be performed serially. Secondly, modern CPUs typically have a 6.4-GB/s memory bandwidth, whereas modern GPUs have memory bandwidths of 32 GB/s and higher [83]. As a result, reading and writing to memory locations can occur at a much faster rate on the GPU. Finally, the clipping operations built into GPUs are able to efficiently exclude calculations that do not influence the contour plot (e.g., calculations on the shaded pixels in Figure 6.6). On the other hand, when using a CPU to perform contouring, the effects of clipping at the contour boundaries must be explicitly calculated. Another key benefit of using GPUs for contouring is that the CPU is then freed up to perform tasks better suited for CPU computation. For instance, while the CPU is used to calculate load flows or perform filtering operations, the GPU can independently construct and render a contour with limited CPU interaction.

GPU-based contouring also has several disadvantages when compared to CPU-based contouring. The greatest potential disadvantage to using GPU-based contouring is the potential loss in accuracy due to floating point errors. Although many of the latest GPUs support 32-bit operations within the fragment and vertex processors, using 32-bit floating point incurs a performance penalty relative to 16-bit operations. Another potential difficulty with using 32-bit floating point textures is

that blending of 32-bit floating point values, needed to implement steps 2.1.1 and 2.1.2 of the algorithm, is only possible on the very latest (and most expensive) video cards such as the NVIDIA GeForce 8- and 9-series. Usage of older GPUs with 16-bit precision can lead to numerical instability, a problem which does not occur with modern CPUs that are capable of handling arbitrarily high precision floating point values.

Another potential problem with using GPU-based contouring is the wide variety of capabilities of GPUs. Although there are some standards for GPU programmability as defined by the OpenGL Architecture Review Board, it is nontrivial to write a GPU contouring program which takes advantage of all GPUs equally. For instance, writing textures with an NVIDIA GPU is optimized for a different set of OpenGL instructions than an ATI GPU. CPUs do not typically suffer as much from this difficulty due to the existence of well-established instruction sets.

6.3 Benefits and Applications of Accelerated Contouring

6.3.1 Improvements in usability

The ability to render faster contours is not inherently useful; the usefulness comes from the improvements in usability and new applications which are enabled by the faster rendering rates. Two key criteria used to evaluate a system's usability are the efficiency of the interface, typically measured by the amount of time needed to complete a specific task, and user satisfaction in using the interface, which can be measured by both subjective and objective means [88]. The improvements in contour rendering times facilitated by using the GPU impact both of these criteria.

6.3.1.1 Improvements to user efficiency

There are two common tasks which are typically used interact with a power system contour—navigation and changing of contour parameters. The first of these tasks, navigation, consists primarily of zooming and panning the contour area. For example, if an operator wants to see a more detailed voltage contour within a particular area, the operator must first select the area of detail and then wait for the contour to be rendered again. As mentioned above, CPU-based contouring can take up to five times as long as GPU-based contouring of the same area. In terms of user efficiency, this means that after the user has signaled to the software that the contour area needs to change, it takes a significantly longer time for CPU contouring to catch up with the user's request. As a result, using GPU-based contouring can significantly improve the efficiency of users' interactions. Furthermore, any attempt by the user to fine-tune either zooming or panning is better facilitated by the rapid contouring speeds enabled by the method presented in Section 6.2.

Changing contouring parameters such as influence radius, resolution, and color mapping are also ways in which users interact with contours to better highlight certain aspects of the underlying data. For example, if a color mapping is initially defined for a range of 0.95 to 1.05 per unit voltage, it may be necessary to change the range to highlight areas where the voltage is lower than 0.95 or higher than 1.05. The speed at which the contour can be regenerated directly impacts the amount of time it takes for any such changes to take effect. Therefore, this is another type of task in which improvements to contour rendering can significantly increase the efficiency of using contours to understand power system behavior.

6.3.1.2 Improvements to user satisfaction

It has been shown in many previous studies that frame rate is directly related to the both performance and satisfaction of users when interacting with computer systems (see [49] for a survey of more than 50 such studies). Figure 3 in [49], which summarizes the satisfaction findings from 16 different studies, shows that the lowest acceptable refresh rate over a wide variety of tasks is 5 Hz. Meeting this target would require contour rendering that takes 0.2 s or less, a goal that is not easily achievable with CPU-based contouring. One way to get these speeds using CPU contouring is to reduce the contour resolution, but evidence suggests that lowering resolution can have as much of an impact on user satisfaction as having a low frame rate [89]. Because GPUs allow contours to be rendered an order of magnitude faster than CPU contours, there should be a marked increase in user satisfaction if contouring is performed on the GPU instead. In addition, one study suggests that increasing the frame rate can actually reduce the stress of operators using the visualization [90], which can be particularly important if the system operators are already in a state of heightened stress due to poor system conditions.

6.3.2 Visualizing faster data

Because the rendering times for contours using GPUs are very short, it should be possible to display data that arrives at a much faster rate than if CPU-based contouring is used. Using the aforementioned rendering times of 0.102 s and 5.10 s for GPU- and CPU-based creation of a single contour, the CPU-based method could only keep up with a data rate below 0.2 Hz, whereas the GPU-based contouring method could keep up with a data rate of 10 Hz. CPU-based methods have been

adequate in the past for state estimator visualization because the data coming out of the state estimator changes around once per minute, or 0.02 Hz. On the other hand, CPU-based contouring of SCADA data, which is sampled every 2 to 3 s, can require a significant reduction in both influence radius and resolution to keep up with the faster data rate. Moving forward, the real-time visualization of PMU data, which arrives at rates as high as 30 Hz, is impractical for CPU-based contouring and is a clear case where GPU-based contouring should be used instead.

By upgrading the video card, improvements on the 0.102-s rendering time can be attained. For instance, using a GeForce 8800 GT card, the average rendering time drops to 0.033 s, which corresponds to a refresh rate of 30 Hz. Therefore, if this video card is used, contours can be updated in real time based on PMU data coming in at 30 samples/s. One of the other advanced metering devices deployed on the grid, the FNET sensor, returns data to the central server at 10 samples/s. In addition, contouring has been used by the research team developing FNET in order to interpret the raw frequency data [91]. Although the videos hosted at [91] were created using offline processing tools in MATLAB, real-time contouring of the frequency data could easily be handled by either the 7600 or 8800 GT hardware.

One final application investigated is the visualization of angle difference information at the PMU-monitored buses. The data visualized are the $\Delta\theta_{candidate}$ signals defined in (3.1), determined for each bus using an N_{trans} value corresponding to 60 s. This is a much bigger value of N_{trans} than is used for event detection, and the reason a longer N_{trans} is used is that it keeps any changes in the angles visible to the

operator for an extended period of time as illustrated by the $N_{trans}^{(3)}$ case in Figure 3.3. Showing these angle contours to an operator would be a useful complement to the event detection work, in that an operator, upon noticing a substantial shift in phasor angles, could then query the event detection algorithms to help determine what causes these angle changes. Figure 6.9 shows how this visualization would look during the TVA line outage analyzed in Section 4.1.2. The color mapping used is shown in Figure 6.10. The change in the contour before and after the event highlights the change in angles on the system and would draw the attention of system operators for further investigation. Also, the use of an N_{trans} value of 60 s lets the change stay visible for 60 s after the event occurs; this is why the same contour is shown at 49 s and 103 s. The deadband between -0.75 and 0.75 degrees in the colormap corresponds to having an angle threshold of 0.75 degrees in event detection, since the angle change must exceed 0.75 degrees before being highlighted.

Alternative methods of processing the incoming data could be used, or alternative data such as voltage or frequency could be visualized without the extensive processing used to obtain useful angle signals. Regardless, using GPUs provides a great deal more flexibility in terms of PMU data visualization, and this technique should significantly improve the potential of having real-time data visualization within control centers.

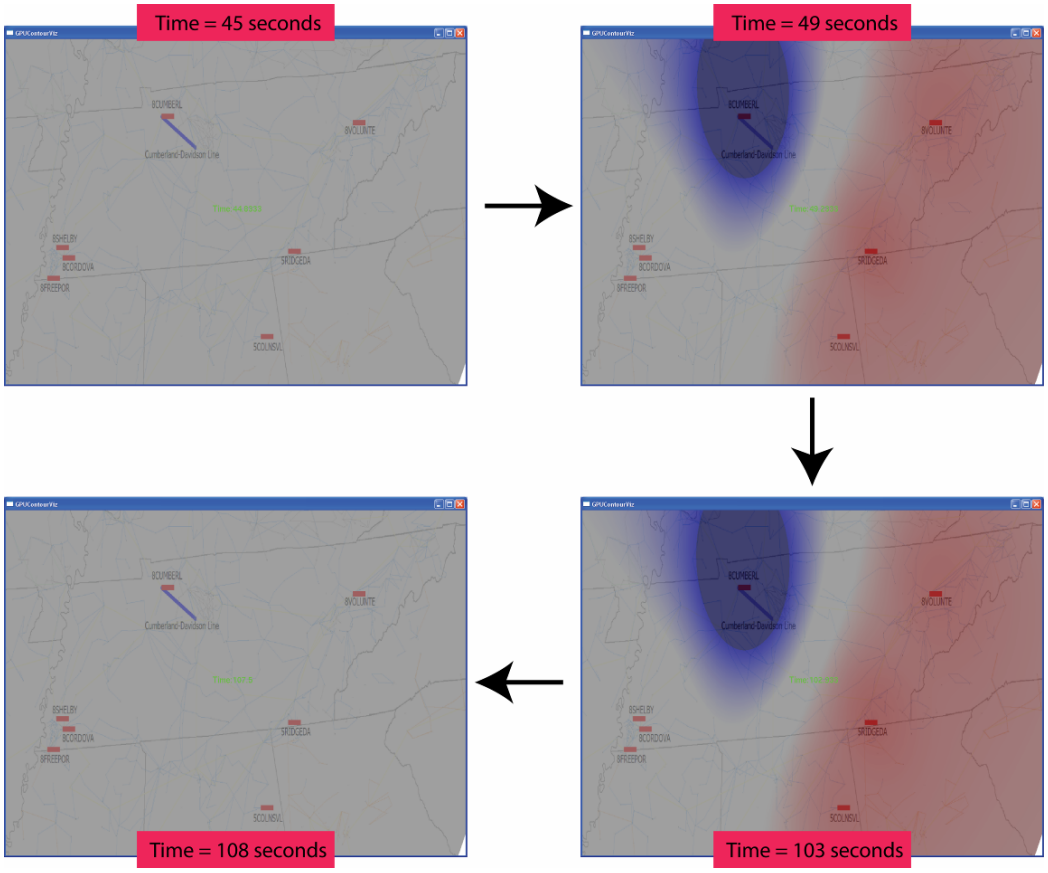

Figure 6.9: Contour of angle changes on the TVA system before and after the line outage.

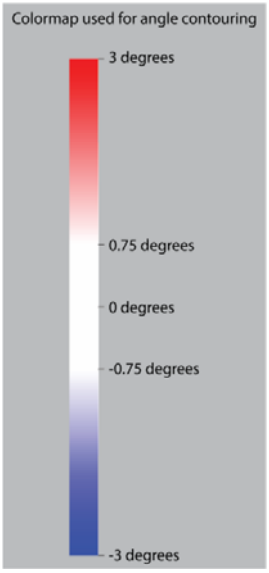

Figure 6.10: Colormap used for angle contours in Figure 6.9.

7 CONCLUSIONS AND FUTURE WORK

Chapter 2 of the dissertation deals with the processing of phasor measurement data in order to detect system events. The results presented for FIR and median filtering highlight the key differences in these two filtering methods—FIR filtering provides more consistent attenuation of noise and unwanted frequency components, whereas median filtering preserves step changes in the system angles. When FIR filtering is used in event detection, the main downside is the increased delay due to the corruption of the step change and the need to capture a longer transition region. Also, as seen with the filtering of the generator outage data, the choice of the cutoff frequency can have a significant impact on the performance of FIR filtering. On the other hand, with median filtering, performance is difficult to predict due to the widely varying response depending on filter length. Also, the median filtering results based on the real data obtained from TVA show that median filtering is less effective at removing noise from the angle measurement signals. Ultimately, the choice of using FIR or median filtering boils down to a tradeoff between minimizing delay in the detection of the event (for which median filtering is superior) and minimizing errors in the determination of $\Delta\theta_{observed}$ (for which FIR filtering is superior). There are several useful avenues of research for future work in the area of PMU data filtering. One area of research, mentioned briefly in the main text, is to determine whether or not hybrid FIR-median filters, which have been shown to work in image processing applications, have similar benefits when applied to PMU measurements. In addition,

there are entire classes of filtering methods which have not been considered, including infinite impulse response (IIR) filtering and other types of nonlinear filters.

Chapter 3 discusses three algorithms which can be used to detect three different types of events—single-line outages, double-line outages, and generator outages. The way single- and double-line outages are modeled is typical when using the dc power flow approximations, and it is unlikely that any improvements can be made to the line outage modeling without adding additional state information. The usage of geographically based filtering of the double-outage set is shown to significantly reduce the search space for double-line outage detection. Future work examining whether or not the filtered event set captures real power system events would be useful. The generator outage modeling makes some significant assumptions in order to predict the postevent angles on the system, including frequency uniformity across the grid. This assumption, which results in accurate droop-based participation factors, leads to a significant delay between event occurrence and detection. One possible area of future research would be to look at different ways to model the changes in the system due to generator outages in order to detect and classify these outages more rapidly.

The third chapter examines the performance of the three algorithms from Chapter 3 in classifying events. All of the results show that in the majority of the events the algorithms can properly classify events based on the observed angle changes due to the events. Reducing the number of PMUs on the system is shown with both single- and double-line outages to result in a significant increase in the number of events that

cannot be differentiated. This is one justification for increasing the PMU deployment on the power system, which currently has a very sparse deployment of these advanced sensors. In addition, the failure of the dc power flow assumptions for several of the single- and double-line outages due to high R/X ratios is clearly seen, showing that the algorithms are ultimately dependent on how well the real system meets the dc power flow assumptions. For the generator outage studies, the algorithm performs well in ranking each generator outage, although lag between event occurrence and classification should be reduced in order to improve situational awareness. One way to extend this research is to obtain and test the algorithms with more real-world data and larger test systems.

The PMU placement problem is discussed in Chapter 5, beginning with definition of several possible objective functions and moving on to description of two means of optimizing based on these objectives—exhaustive search and simulated annealing. The overwhelming computational burden of exhaustive search and the shortcomings of using an approximate optimizer like simulated annealing are both discussed. Results from the simulated annealing and exhaustive search optimizations for the placement of three and six PMUs show that simulated annealing does provide useful results, although it is suboptimal. Testing other methods of large-scale combinatorial optimization such as genetic algorithms would be useful, particularly if the results are compared to simulated annealing and exhaustive search. The results obtained by using simulated annealing for the placement of 9, 18, and 27 PMUs show that deploying more PMUs will definitely increase the performance of the event detection

algorithms and also that simulated annealing scales much better than exhaustive search.

Finally, Chapter 6 presents a new method of contouring power system data which leverages the abundant computational power of modern graphics cards. The results indicate that moving contouring from the CPU to the GPU can result in significant improvements to usability and provide opportunities for development of applications that render newer measurement data in real time. In continuing this work, GPU-based visualization techniques from other domains such as medical imagery and scientific visualization can also be evaluated to see whether or not they provide a better understanding of the system conditions to operators. Formal evaluations of how GPU-based techniques impact user performance and satisfaction would also be useful to obtain.

As new measurement devices are developed and deployed on the grid, applications to improve situational awareness should be one of the top priorities due to its importance in power system operations. This dissertation, which focuses on both the processing and presentation of the new data available from PMUs, should help in improving the awareness of grid conditions and, by doing so, improve the reliability of the power grid.

APPENDIX A 37-BUS SYSTEM DESCRIPTION

The basis for this system is the 37-bus system from Example 13.9 in [92]. The machine and exciter models were changed to the GENROU [93] and IEEET1 [94] models to provide a more detailed model than the original case, and the parameters for these models were taken from similarly sized units in the Eastern Interconnect. In addition, an IEEEG1 governor [69] was added to each machine using parameters from similarly sized units in the Eastern Interconnect. A full description of the system is provided in Tables A.1-A.10. Figures A.1 and A.2 show the full system one-line diagram and the 18 PMUs monitored for outage testing, respectively.

Table A.1: Bus information

Number	Name	Nom kV	PU Volt	Angle (Deg)
1	TIM345	345	1.02621	-0.75766
3	MORO138	138	1.00773	-3.2384245
5	HOMER69	69	1.00817	-5.252666
10	RAY69	69	1.02293	-4.2961769
12	TIM69	69	1.01852	-5.0273795
13	FERNA69	69	1.00666	-5.786334
14	WEBER69	69	1.02	-4.3555994
15	UIUC69	69	1.00707	-6.1080384
16	PETE69	69	1.00424	-6.6900287
17	PAI69	69	1.01186	-5.9505262
18	HANNAH69	69	1.00546	-6.904346
19	GROSS69	69	1.01225	-5.7058263
20	SHIMKO69	69	1.01331	-4.3103828
21	WOLEN69	69	0.98743	-7.2428098
24	HALE69	69	1.00509	-5.7531943
27	HISKY69	69	1.00605	-6.5725245
28	JO345	345	1.03	2.7282064
29	JO138	138	1.02513	0.7128861

Number	Name	Nom kV	PU Volt	Angle (Deg)
30	BUCKY138	138	1.01008	-1.820895
31	SLACK345	345	1.03	0
32	SAVOY138	138	1.01242	-1.5171459
33	SAVOY69	69	1.01303	-2.7431788
34	PATTEN69	69	1.01277	-4.7015119
35	SLACK138	138	1.02684	-0.8142443
37	AMANDA69	69	1.00408	-6.945878
38	RAY345	345	1.02706	-0.7578956
39	RAY138	138	1.02132	-2.5541735
40	TIM138	138	1.01074	-3.6131575
41	LAUF138	138	1.00651	-2.2658136
44	LAUF69	69	1.02	-2.9113064
47	BOB138	138	1.01703	-4.51683
48	BOB69	69	1	-6.4654179
50	ROGER69	69	1.02	-3.0960746
53	BLT138	138	1.02	-4.4752307
54	BLT69	69	1.01	-6.0853167
55	DEMAR69	69	1.00124	-6.4195685
56	LYNN138	138	1.02457	-0.151776

Table A.2: Generator information

Name of Bus	ID	Status	MVA Base	Gen MW	Gen Mvar	Set Volt	Min MW	Max MW	Min Mvar	Max Mvar
WEBER69	1	Closed	41	10	24.12	1.02	0	41	0	5
JO345	1	Closed	200	150	-2.85	1.03	0	200	-60	60
JO345	2	Closed	200	150	-2.85	1.03	0	200	-60	60
SLACK345	1	Closed	295	88.96	23.3	1.03	0	295	-90	120
LAUF69	1	Closed	221.4	150	-30.66	1.02	0	221.4	-20	40
BOB69	1	Closed	97.58	16	-21.21	1	0	97.58	-14	26
ROGER69	1	Closed	101.8	38	5.63	1.02	0	101.8	-18	33
BLT138	1	Closed	176.5	140	76.54	1.02	0	176.5	0	45
BLT69	1	Closed	147.06	75.23	34.75	1.01	0	147.06	-20	60

Table A.3: Line information

From Number	To Number	Circuit	Status	R	X	B	Limit (MVA)
1	31	1	Closed	0.00117	0.017	0.182	597
1	40	1	Closed	0.001	0.0623	0	250
3	40	1	Closed	0.0048	0.0368	0.0182	233
3	41	1	Closed	0.00902	0.0571	0.0135	233
5	18	1	Closed	0.03133	0.07675	0.0015	102
5	44	1	Closed	0.03463	0.08253	0.0016	102
10	13	1	Closed	0.03375	0.0789	0.0012	82
10	19	1	Closed	0.0405	0.0953	0.0021	72
10	39	1	Closed	0.00106	0.03949	-0.0041	186.7
12	17	1	Closed	0.02172	0.06935	0.0019	82
12	18	1	Closed	0.02747	0.08909	0.0019	106
12	27	1	Closed	0.0142	0.07557	0.0695	112
12	40	1	Closed	0.00107	0.04039	-0.0042	187
12	40	2	Closed	0.00107	0.04039	-0.0042	187
13	55	1	Closed	0.02216	0.0904	0.0017	90
14	34	1	Closed	0.02625	0.06429	0.0012	72
14	44	1	Closed	0.04211	0.08545	0.0474	81
15	16	1	Closed	0.00753	0.02582	0.0206	93
15	24	1	Closed	0.01835	0.02845	0.0221	74
15	54	1	Closed	0.00846	0.00465	0.021	77
15	54	2	Closed	0.00855	0.0047	0.0213	77
15	54	3	Closed	0.00859	0.00472	0.0214	77
16	27	1	Closed	0.00366	0.01312	0.046	112
17	19	1	Closed	0.02703	0.03613	0.002	50
18	37	1	Closed	0.01017	0.00559	0.0253	68
18	37	2	Closed	0.01017	0.00559	0.0253	68
20	34	1	Closed	0.02423	0.05862	0.0011	72
20	48	1	Open	0.02133	0.0521	0.0009	82
20	50	1	Closed	0.0424	0.07509	0.0008	82
21	48	1	Closed	0.01824	0.04236	0.0075	72
21	48	2	Closed	0.01829	0.04246	0.0078	72
24	44	1	Closed	0.03993	0.09965	0.002	82
28	29	1	Closed	0.00087	0.051	0	220
28	29	2	Closed	0.00087	0.051	0	220
31	28	1	Closed	0.00224	0.03268	0.35	600
32	29	1	Closed	0.0103	0.05681	0.0164	191
29	41	1	Closed	0.01868	0.1259	0.0356	191
56	29	1	Closed	0.00771	0.0544	0.0138	133
30	32	1	Closed	0.00225	0.0134	0.004	191
30	41	1	Closed	0.00735	0.04395	0.0123	191
35	31	1	Closed	0.00087	0.051	0	220
31	38	1	Closed	0.00075	0.01092	0.117	597
33	32	1	Closed	0.0025	0.0723	0	101
33	50	1	Closed	0.0502	0.101	0.0025	82
35	39	1	Closed	0.01028	0.07253	0.0176	288
35	56	1	Closed	0.01243	0.07847	0.0192	100
39	38	1	Closed	0.00094	0.05107	-0.0127	224
39	38	2	Closed	0.00095	0.05116	-0.0124	224
39	40	1	Closed	0.01161	0.08085	0.0225	233
39	47	1	Closed	0.00775	0.05244	0.0154	233
44	41	1	Closed	0.0025	0.07144	-0.0045	101
44	41	2	Closed	0.00244	0.06829	-0.006	101
48	47	1	Closed	0.00101	0.03925	-0.0035	187
47	53	1	Closed	0.00204	0.00692	0.1601	185
48	54	1	Closed	0.01473	0.036	0.0744	105
54	53	1	Closed	0.00134	0.04988	-0.0004	187
54	55	1	Closed	0.06246	0.08246	0.0003	50

Table A.4: Load information

Number of Bus	ID	Status	MW	Mvar
3	1	Closed	12.3	5
5	1	Closed	14	3.2
10	1	Closed	16.8	2.5
12	1	Closed	22.9	6.5
13	1	Closed	23	7
14	1	Closed	22.2	15.2
15	1	Closed	58.2	36.3
16	1	Closed	57.8	40.4
17	1	Closed	32.8	12.9
18	1	Closed	45	12
19	1	Closed	18.3	5
20	1	Closed	15.3	5
21	1	Closed	74.4	26.8
24	1	Closed	36.3	10.4
27	1	Closed	20	6
30	1	Closed	23.4	6.2
33	1	Closed	28	6
34	1	Closed	22.81	3
37	1	Closed	27	3
44	1	Closed	59.8	12.3
48	1	Closed	55.8	12.5
50	1	Closed	14.1	3
53	1	Closed	59.5	27.8
54	1	Closed	12.43	5.73
55	1	Closed	22.65	6.15
56	1	Closed	14	3.7

Table A.5: Switched shunt information

Number of Bus	ID	Reg Bus Num	Status	Control Mode	Regulates	Actual Mvar	Volt High	Volt Low	Reg Volt	Nominal Mvar	Max Mvar	Min Mvar
13	1	13	Closed	Discrete	Volt	4.86	1.025	0.998	1.0067	4.8	4.8	0
14	1	14	Closed	Discrete	Volt	7.49	1.025	0.998	1.02	7.2	7.2	0
15	1	15	Closed	Discrete	Volt	12.78	1.025	0.998	1.0071	12.6	12.6	0
16	1	16	Closed	Discrete	Volt	29.04	1.025	0.998	1.0042	28.8	28.8	0
17	1	17	Closed	Discrete	Volt	15.97	1.025	0.998	1.0119	15.6	15.6	0
18	1	18	Closed	Fixed	Volt	18.2	1.025	0.998	1.0055	18	14.4	0
20	1	20	Closed	Discrete	Volt	7.39	1.025	0.998	1.0133	7.2	7.2	0
44	1	44	Closed	Discrete	Volt	0	1.025	0.998	1.02	0	20	0

Table A.6: Transformer control information

From Number	To Number	Circuit	Type	Status	Tap Ratio	Phase (Deg)	XF Auto	Reg Bus	Reg Value	Reg Error	Reg Min	Reg Max	Tap Min	Tap Max	Step Size
1	40	1	LTC	Closed	1	0	No	1	1.02621	0.02621	0.99	1	0.9	1.1	0.00625
10	39	1	LTC	Closed	1.00625	0	Yes	10	1.02293	0	1.015	1.025	0.9	1.1	0.00625
12	40	1	LTC	Closed	1.0125	0	Yes	12	1.01852	0	1.015	1.025	0.9	1.1	0.00625
12	40	2	LTC	Closed	1.0125	0	Yes	12	1.01852	0	1.015	1.025	0.9	1.1	0.00625
28	29	1	LTC	Closed	1	0	No	28	1.03	0.03	0.99	1	0.9	1.1	0.00625
28	29	2	LTC	Closed	1	0	No	28	1.03	0.03	0.99	1	0.9	1.1	0.00625
35	31	1	LTC	Closed	1	0	No	35	1.02684	0.02684	0.99	1	0.9	1.1	0.00625
33	32	1	LTC	Closed	1	0	No	33	1.01303	0.01303	0.99	1	0.9	1.1	0.00625
39	38	1	Fixed	Closed	1	0	No	0	0	-0.51	0.51	1.5	0.51	1.5	0.00625
39	38	2	Fixed	Closed	1	0	No	0	0	-0.51	0.51	1.5	0.51	1.5	0.00625
44	41	1	LTC	Closed	1.02125	0	No	44	1.02	0	1	1.02	0.99	1.1	0.00625
44	41	2	LTC	Closed	1.02125	0	No	44	1.02	0	1	1.02	0.99	1.1	0.00625
48	47	1	LTC	Closed	1	0	No	48	1	-0.015	1.015	1.025	0.9	1.1	0.00625
54	53	1	LTC	Closed	1	0	No	0	0	-1	1	1.02	0.99	1.1	0.00625

Table A.7: Line shunt information

From Number	To Number	Circuit	Bus Number Located At	ID	G	B	Status
12	40	1	40	1	0.0006	0	Closed
12	40	2	40	1	0.0006	0	Closed
39	38	1	38	1	0.0014	0	Closed
39	38	2	38	1	0.0014	0	Closed
44	41	1	41	1	0.0007	0	Closed
44	41	2	41	1	0.0007	0	Closed
48	47	1	47	1	0.0006	0	Closed
54	53	1	53	1	0.0004	0	Closed

Table A.8: Machine model information (all nonspecified parameters are zero)

Bus Name	Gen ID	Type	MVA Base	Device Status	H	Xd	Xq	Xdp	Xqp	Xdpp	Xl	Tdop	Tqop	Tdopp	Tqopp	S(1.0)	S(1.2)
WEBER69	1	GENROU	41	Active	2.35	1.77	1.7	0.255	0.454	0.2	0.14	4	0.514	0.032	0.062	0.083	0.39
JO345	1	GENROU	200	Active	3.021	1.695	1.616	0.239	0.389	0.1795	0.1	5.3	0.49	0.034	0.081	0.108	0.455
JO345	2	GENROU	200	Active	3.021	1.695	1.616	0.239	0.389	0.1795	0.1	5.3	0.49	0.034	0.081	0.108	0.455
SLACK345	1	GENROU	295	Active	5.93	1.65	1.6	0.216	0.895	0.12	0.1	7.6	1.5	0.055	0.265	0.13	0.456
LAUF69	1	GENROU	221.4	Active	2.08	2.1	2.02	0.295	0.505	0.225	0.16	4	0.55	0.032	0.06	0.094	0.421
BOB69	1	GENROU	97.58	Active	4.321	1.37	1.32	0.175	0.29	0.135	0.075	5.6	0.54	0.036	0.083	0.1214	0.5897
ROGER69	1	GENROU	101.8	Active	4.51	1.22	1.18	0.126	0.22	0.085	0.08	5.6	0.48	0.032	0.078	0.1	0.48
BLT138	1	GENROU	176.5	Active	2.27	1.69	1.61	0.23	0.4225	0.17	0.0845	9.6	0.3	0.05	0.05	0.0963	0.555
BLT69	1	GENROU	147.059	Active	2.7779	1.7604	1.7503	0.383	0.7689	0.3076	0.2322	4.54	0.5	0.05	0.06	0.11	0.48

Table A.9: Exciter model information

Bus Name	Gen ID	Type	MVA Base	Device Status	Tr	Ka	Ta	Vrmax	Vrmin	Ke	Te	Kf	Tf	Switch	E1	SE(E1)	E2	SE(E2)
WEBER69	1	IEEET1	41	Active	0	24.86	0.02	1	-1	0	0.588	0.024	1	0	2.1	0.08	2.81	0.314
JO345	1	IEEET1	200	Active	0.06	25	0.2	1	-1	0	0.5	0.0805	0.35	0	3.375	0.0684	4.5	0.2667
JO345	2	IEEET1	200	Active	0.06	25	0.2	1	-1	0	0.5	0.0805	0.35	0	3.375	0.0684	4.5	0.2667
SLACK345	1	IEEET1	295	Active	0	40	0.15	1	-1.5	0	0.5	0.08	1	0	2.5	0.1	3	0.4
LAUF69	1	IEEET1	221.4	Active	0	50	0.02	1	-1	-0.057	0.541	0.086	1.02	0	3.08	0.038	4.11	0.3
BOB69	1	IEEET1	97.58	Active	0	400	0.02	7.3	-6.6	1	0.8	0.03	1	0	3	0.5	4	0.86
ROGER69	1	IEEET1	101.8	Active	0.06	25	0.2	1	-1	-0.045	0.05	0.07	0.35	0	2.25	0.066	3	0.267
BLT138	1	IEEET1	176.5	Active	0.06	25	0.2	1	-1	0	0.5	0.1	1.351	0	2.98	0.077	3.968	0.302
BLT69	1	IEEET1	147.059	Active	0.06	40	0.1	1	-1	0	0.67	0.11	1	0	3	0.09	4	0.368

Table A.10: Governor model information (all nonspecified parameters are zero)

Bus Name	Gen ID	Type	MVA Base	Device Status	K	T1	T2	T3	Uo	Uc	Pmax	Pmin	T4	K1	K2	T5	K3	K4	T6	K5
WEBER69	1	IEEEG1	41.0000	Active	20	0	0	0.05	999	-999	1	0	0.065	0	0	0.036	1	0	0	0
JO345	1	IEEEG1	200.0000	Active	20	0.083	0.05	0.2	0.27	-0.25	1	0	0.05	0.3	0	7.85	0.25	0	0.4	0.45
JO345	2	IEEEG1	200.0000	Active	20	0.083	0.05	0.2	0.27	-0.25	1	0	0.05	0.3	0	7.85	0.25	0	0.4	0.45
SLACK345	1	IEEEG1	295.0000	Active	20	0	0	0.15	0.012	-0.012	1	0	0.275	0.224	0	0.1	0.395	0	0.3	0.381
LAUF69	1	IEEEG1	221.4000	Active	20	0	0	0.1	0.1	-1	1	0	0.25	0.3	0	7.5	0.4	0	0.4	0.3
BOB69	1	IEEEG1	97.5800	Active	20	0	0	0.1	0.1	-1	0.9	0	0.4	1	0	0	0	0	0	0
ROGER69	1	IEEEG1	101.8000	Active	20	0	0	0.1	0.1	-1	0.9	0	0.4	1	0	0	0	0	0	0
BLT138	1	IEEEG1	176.5000	Active	20	0.083	0	0.2	0.1	-0.1	1	0	0.218	0.218	0	7.77	0.782	0	0	0
BLT69	1	IEEEG1	147.0590	Active	20	0	0	0.1	0.007	-0.007	1	0	0.01	0	0	0	0	0	0	1

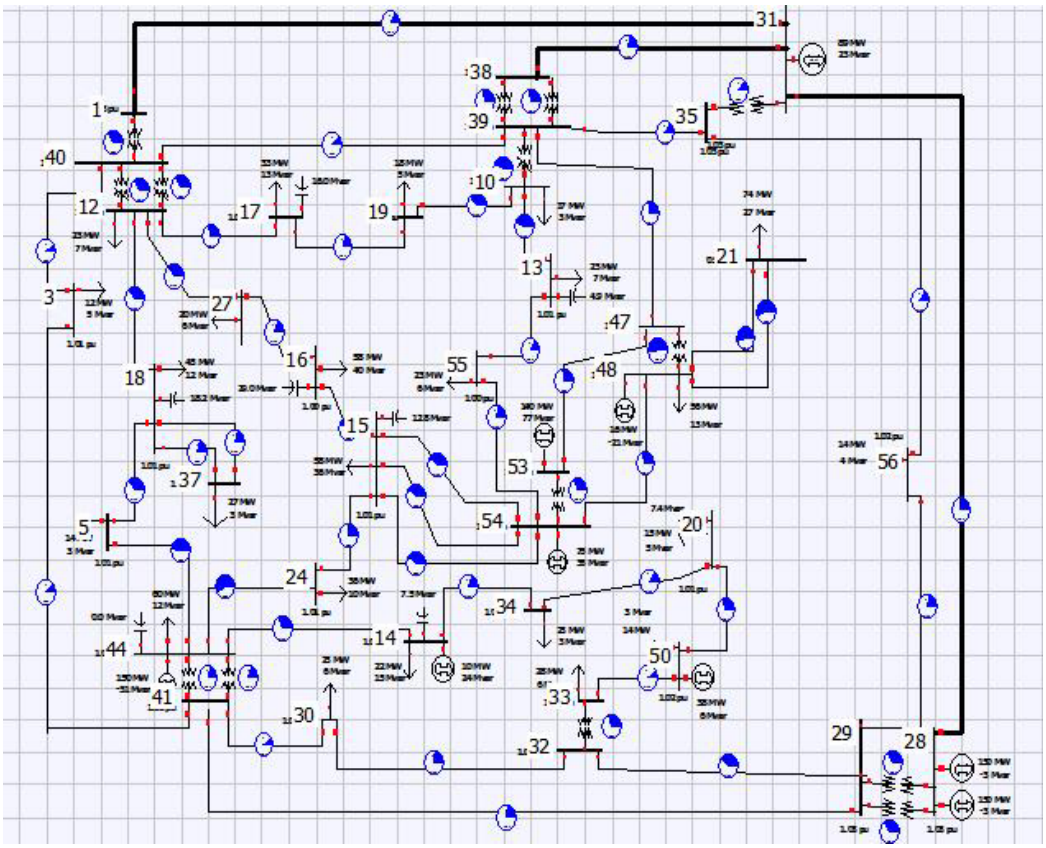

Figure A.1: One-line diagram of the 37-bus system.

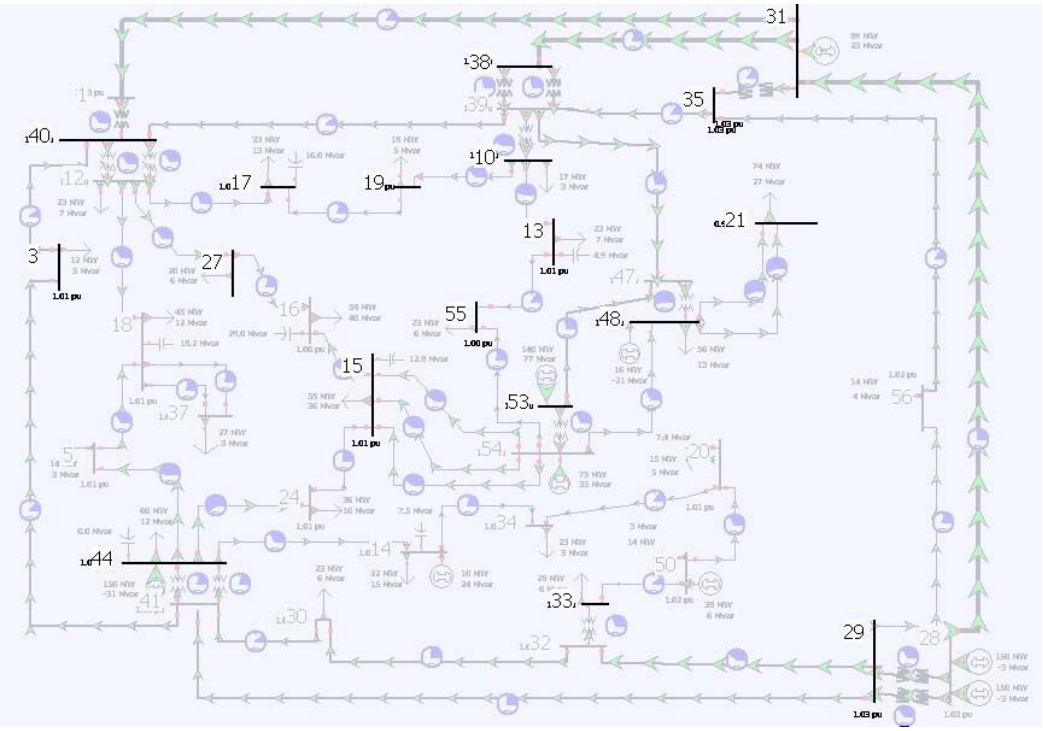

Figure A.2: One-line diagram with 18 PMU buses as defined in Section 4.1.1.2 highlighted.

REFERENCES

[1] US-Canada Power System Outage Task Force, "Final report on the August 14, 2003 blackout in the United States and Canada," April 2004.

[2] U.S. Department of Energy and Federal Energy Regulatory Commission, "Steps to establish a real-time transmission monitoring system for transmission owners and operators within the Eastern and Western Interconnections," February 2006.

[3] A. G. Phadke, "Synchronized phasor measurements in power systems," *IEEE Computer Applications in Power*, vol. 6, no. 2, pp. 10-15, 1993.

[4] M. Donnelly, M. Ingram, and J. R. Carroll, "Eastern interconnection phasor project," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, 2006, p. 245a.

[5] J. Weiqing, V. Vittal, and G. T. Heydt, "A distributed state estimator utilizing synchronized phasor measurements," *IEEE Transactions on Power Systems*, vol. 22, no. 2, pp. 563-571, 2007.

[6] M. Zhou, V. A. Centeno, J. S. Thorp, and A. G. Phadke, "An alternative for including phasor measurements in state estimators," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1930-1937, 2006.

[7] J. S. Thorp, A. G. Phadke, and K. J. Karimi, "Real time voltage-phasor measurement for static state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-104, no. 11, pp. 3098-3106, 1985.

[8] A. P. S. Meliopoulos et al., "PMU data characterization and application to stability monitoring," in *IEEE Power Engineering Society General Meeting*, 2006, pp. 1-8.

[9] K. Sun, S. Likhate, V. Vittal, V. S. Kolluri, and S. Mandal, "An online dynamic security assessment scheme using phasor measurements and decision trees," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1935-1943, 2007.

[10] A. R. Khatib, R. F. Nuqui, M. R. Ingram, and A. G. Phadke, "Real-time estimation of security from voltage collapse using synchronized phasor measurements," in *IEEE Power Engineering Society General Meeting*, 2004, vol. 1, pp. 582-588.

[11] G. Zhang, P. Hirsch, and S. Lee, "Wide area frequency visualization using smart client technology," in *IEEE Power Engineering Society General Meeting*, 2007, pp. 1-8.

[12] R. Klump, R. E. Wilson, and K. E. Martin, "Visualizing real-time security threats using hybrid SCADA / PMU measurement displays," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005, p. 55c.

[13] S. Chun-Lien and J. Bo-Yuan, "Visualization of large-scale power system operations using phasor measurements," in *International Conference on Power System Technology*, 2006, pp. 1-6.

[14] A. Monticelli, "Modeling circuit breakers in weighted least squares state estimation," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1143-1149, 1993.

[15] US-Canada Power System Outage Task Force, "Final report on the implementation of the task force recommendations," September 2006.

[16] A. G. Phadke, J. S. Thorp, and M. G. Adamiak, "A new measurement technique for tracking voltage phasors, local system frequency, and rate of change of frequency," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-102, no. 5, pp. 1025-1038, 1983.

[17] A. G. Phadke, "Synchronized phasor measurements-a historical overview," in *IEEE/PES Asia Pacific Transmission and Distribution Conference and Exhibition*, 2002, vol. 1, pp. 476-479.

[18] T. L. Baldwin, L. Mili, M. B. Boisen, Jr., and R. Adapa, "Power system observability with minimal phasor measurement placement," *IEEE Transactions on Power Systems*, vol. 8, no. 2, pp. 707-715, 1993.

[19] S. E. Widergren, Z. Huang, and J. E. Dagle, "Electric system-wide measurements: North American directions," in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, 2007, p. 120.

[20] R. F. Nuqui and A. G. Phadke, "Phasor measurement unit placement techniques for complete and incomplete observability," *IEEE Transactions on Power Delivery*, vol. 20, no. 4, pp. 2381-2388, 2005.

[21] B. Fardanesh, S. Zelingher, A. P. Sakis Meliopoulos, G. Cokkinides, and J. Ingleson, "Multifunctional synchronized measurement network," *IEEE Computer Applications in Power*, vol. 11, no. 1, pp. 26-30, 1998.

[22] A. R. Messina, V. Vittal, D. Ruiz-Vega, and G. Enriquez-Harper, "Interpretation and visualization of wide-area PMU measurements using Hilbert analysis," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1763-1771, 2006.

[23] S. M. Brahma and A. A. Girgis, "Fault location on a transmission line using synchronized voltage measurements," *IEEE Transactions on Power Delivery*, vol. 19, no. 4, pp. 1619-1622, 2004.

[24] R. O. Burnett, Jr. et al., "Synchronized phasor measurements of a power system event," *IEEE Transactions on Power Systems*, vol. 9, no. 3, pp. 1643-1650, 1994.

[25] J. F. Hauer, "Validation of phasor calculations in the Macrodyne PMU for California-Oregon transmission project tests of March 1993," *IEEE Transactions on Power Delivery*, vol. 11, no. 3, pp. 1224-1231, 1996.

[26] J. F. Hauer, N. B. Bhatt, K. Shah, and S. Kolluri, "Performance of WAMS East in providing dynamic information for the northeast blackout of August 14, 2003," in *IEEE Power Engineering Society General Meeting*, 2004, vol. 2, pp. 1685-1690.

[27] IEEE Power Engineering Society, "IEEE standard for synchrophasors for power systems," IEEE Std C37.118-2005, 2006.

[28] North American Electric Reliability Corporation, "Reliability coordinator reference document," 2004.

[29] L. S. Davis, "A survey of edge detection techniques," *Computer Graphics and Image Processing*, vol. 4, no. 3, pp. 248-270, 1975.

[30] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2003.

[31] A. C. Bovik and J. D. C. Munson, "Edge detection using median comparisons," *Computer Vision, Graphics, and Image Processing*, vol. 33, no. 3, pp. 377-389, 1986.

[32] W. C. Karl, S. B. Leeb, L. A. Jones, J. L. Kirtley, and G. C. Verghese, "Applications of rank-based median filters in power electronics," *IEEE Transactions on Power Electronics*, vol. 7, no. 3, pp. 437-443, 1992.

[33] M. Kezunovic, I. Rikalo, and D. J. Sobajic, "Real-time and off-line transmission line fault classification using neural networks," *International Journal of Engineering Intelligent Systems*, vol. 4, no. 1, pp. 57-63, March 1996.

[34] C. J. Kim and B. D. Russell, "Classification of faults and switching events by inductive reasoning and expert system methodology," *IEEE Transactions on Power Delivery*, vol. 4, no. 3, pp. 1631-1637, 1989.

[35] W. R. Anis Ibrahim and M. M. Morcos, "Artificial intelligence and advanced mathematical tools for power quality applications: A survey," *IEEE Transactions on Power Delivery*, vol. 17, no. 2, pp. 668-673, 2002.

[36] H. Singh and F. L. Alvarado, "Network topology determination using least absolute value state estimation," *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1159-1165, 1995.

[37] F. L. Alvarado, "Determination of external system topology errors," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 11, pp. 4553-4561, 1981.

[38] F. Schlaepfer, T. C. Kelly, and A. G. Dewey, "An interactive load flow program," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-91, no. 1, pp. 78-84, 1972.

[39] G. Pires de Azevedo, C. Sieckenius de Souza, and B. Feijo, "Enhancing the human-computer interface of power system applications," *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 646-653, 1996.

[40] J. D. Weber and T. J. Overbye, "Voltage contours for power system visualization," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 404-409, 2000.

[41] Y. Sun and T. J. Overbye, "Visualizations for power system contingency analysis data," *IEEE Transactions on Power Systems*, vol. 19, no. 4, pp. 1859-1866, 2004.

[42] T. J. Overbye, D. A. Wiegmann, A. M. Rich, and Y. Sun, "Human factors aspects of power system voltage contour visualizations," *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 76-82, 2003.

[43] D. A. Wiegmann, G. R. Essenberg, T. J. Overbye, and Y. Sun, "Human factor aspects of power system flow animation," *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1233-1240, 2005.

[44] D. Savageau and T. J. Overbye, "Adaptive influence distance algorithm for contouring bus-based power system data," in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, 2007, p. 117.

[45] D. Geer, "Taking the graphics processor beyond graphics," *Computer*, vol. 38, no. 9, pp. 14-16, 2005.

[46] M. Pharr, Ed., *GPU Gems 2*. Reading, MA: Addison-Wesley, 2005.

[47] H. Nguyen, Ed., *GPU Gems 3*. Reading, MA: Addison-Wesley, 2007.

[48] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. Hamann, "A framework for real-time volume visualization of streaming scattered data," in *10th International Fall Workshop on Vision, Modeling, and Visualization*, 2005, pp. 225-232.

[49] J. Y. C. Chen and J. E. Thropp, "Review of low frame rate effects on human performance," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 37, no. 6, pp. 1063-1076, 2007.

[50] B. Stott, "Review of load-flow calculation methods," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 916-929, 1974.

[51] Y. Ota, T. Hashiguchi, H. Ukai, M. Sonoda, Y. Miwa, and A. Takeuchi, "Monitoring of interconnected power system parameters using PMU based WAMS," in *Power Tech 2007 Conference*, Lausanne, Switzerland, 2007, p. 5.

[52] Y. Jun-Zhe, L. Chih-Wen, and W. Wen-Giang, "A hybrid method for the estimation of power system low-frequency oscillation parameters," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2115-2123, 2007.

[53] K. Mei, S. M. Rovnyak, and O. Chee-Mun, "Dynamic event detection using wavelet analysis," in *IEEE Power Engineering Society General Meeting*, 2006, p. 7.

[54] I. Kamwa, J. Beland, and D. McNabb, "PMU-based vulnerability assessment using wide-area severity indices and tracking modal analysis," in *IEEE Power Systems Conference and Exposition*, 2006, pp. 139-149.

[55] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[56] M. Klein, G. J. Rogers, and P. Kundur, "A fundamental study of inter-area oscillations in power systems," *IEEE Transactions on Power Systems*, vol. 6, no. 3, pp. 914-921, 1991.

[57] P. Kundur and W. Lei, "Small signal stability analysis: Experiences, achievements, and challenges," in *International Conference on Power System Technology*, 2002, vol. 1, pp. 6-12.

[58] R. L. Cresap and J. F. Hauer, "Emergence of a new swing mode in the western power system," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 4, pp. 2037-2045, 1981.

[59] N. Sankarayya, K. Roy, and D. Bhattacharya, "Optimizing computations in a transposed direct form realization of floating-point LTI-FIR systems," in *IEEE/ACM International Conference on Computer-Aided Design*, 1997, pp. 120-125.

[60] J. K. Wang, R. M. Gardner, and L. Yilu, "Analysis of system oscillations using wide-area measurements," in *IEEE Power Engineering Society General Meeting*, 2006, pp. 1-6.

[61] E. Chen, H. S. Timorabadi, and F. P. Dawson, "Real-time phasor measurement method including a GPS common time-stamp for distributed power system monitoring and control," in *Canadian Conference on Electrical and Computer Engineering*, 2005, pp. 441-444.

[62] M. Juhola, J. Katajainen, and T. Raita, "Comparison of algorithms for standard median filtering," *IEEE Transactions on Signal Processing*, vol. 39, no. 1, pp. 204-208, 1991.

[63] Z. Huang, T. Faris, K. Martin, J. Hauer, C. Bonebrake, and J. Shaw, "Laboratory performance evaluation report of SEL 421 phasor measurement unit," Pacific Northwest National Laboratory, November 2007.

[64] J. Astola and P. Kuosmann, *Fundamentals of Nonlinear Digital Filtering*. Boca Raton, FL: CRC Press, 1997.

[65] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control*. New York, NY: Wiley, 1984.

[66] H. Anton, *Elementary Linear Algebra*, 8th ed. New York, NY: Wiley, 2000.

[67] M. Crow, *Computational Methods for Electric Power Systems*. Boca Raton, FL: CRC Press, 2003.

[68] P. Kundur, *Power System Stability and Control*. New York, NY: McGraw-Hill, 1994.

[69] IEEE Task Force on Overall Plant Response, "Dynamic models for steam and hydro turbines in power system studies," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-92, no. 6, pp. 1904-1915, 1973.

[70] T. Guler and G. Gross, "Detection of island formation and identification of causal factors under multiple line outages," *IEEE Transactions on Power Systems*, vol. 22, no. 2, pp. 505-513, 2007.

[71] B. L. Silverstein and D. M. Porter, "Contingency ranking for bulk system reliability criteria," *IEEE Transactions on Power Systems*, vol. 7, no. 3, pp. 956-964, 1992.

[72] N. Megiddo, "Linear-time algorithms for linear programming in R^3 and related problems," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 759-776, 1983.

[73] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, May 1983.

[74] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," August 1975. [Online]. Available: http://hdl.handle.net/2027.42/4507.

[75] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995, vol. 4, pp. 1942-1948.

[76] K. P. Wong and Y. W. Wong, "Combined genetic algorithm/simulated annealing/fuzzy set approach to short-term generation scheduling with take-or-pay fuel contract," *IEEE Transactions on Power Systems*, vol. 11, no. 1, pp. 128-136, 1996.

[77] I. Amidror, "Scattered data interpolation methods for electronic imaging systems: A survey," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 157-176, 2002.

[78] W. A. Nuss and D. W. Titley, "Use of multiquadric interpolation for meteorological objective analysis," *Monthly Weather Review*, vol. 122, no. 7, pp. 1611-1631, July 1994.

[79] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1049-1075, 1999.

[80] R. J. Renka, "Multivariate interpolation of large sets of scattered data," *ACM Transactions on Mathematical Software*, vol. 14, no. 2, pp. 139-148, 1988.

[81] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 23rd ACM National Conference*, 1968, pp. 517-524.

[82] R. E. Barnhill, "Representation and approximation of surfaces," in *Mathematical Software III*, J. R. Rice, Ed. New York, NY: Academic Press, 1977, pp. 517-523.

[83] E. Kilgariff and R. Fernando, "The GeForce 6 series GPU architecture," in *GPU Gems 2*, M. Pharr, Ed. Reading, MA: Addison-Wesley, 2005, pp. 471-491.

[84] S. Green, "The OpenGL framebuffer object extension," presented at Game Developers Conference, San Francisco, CA, 2005.

[85] M. Kilgard, “The OpenGL Utility Toolkit (GLUT) programming interface API version 3,” 2008. [Online]. Available: http://www.opengl.org/documentation/specs/glut/spec3/spec3.html.

[86] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard, "Cg: A system for programming graphics hardware in a C-like language," in *ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques*, 2003, pp. 896-907.

[87] J. D. Meier, S. Vasireddy, A. Babbar, and A. Mackman, "How to: time managed code using QueryPerformanceCounter and QueryPerformanceFrequency," Microsoft Developer Network, May 2004. [Online]. Available: http://msdn2.microsoft.com/en-us/library/ms979201.aspx.

[88] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human-Computer Interaction*, 3rd ed. New York, NY: Pearson, 2004.

[89] J. D. McCarthy, M. A. Sasse, and D. Miras, "Sharp or smooth: Comparing the effects of quantization vs. frame rate for streamed video," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vienna, Austria, 2004.

[90] G. M. Wilson, "Psychophysiological indicators of the impact of media quality on users," in *CHI Extended Abstracts on Human Factors in Computing Systems*, 2001, pp. 95-96.

[91] Virginia Tech Power IT Lab, "Frequency wave propagation," 2008. [Online]. Available: http://www.powerit.vt.edu/FNET/05-2-Application-frequency_wave_propagation.htm.

[92] J. D. Glover, M. S. Sarma, and T. Overbye, *Power System Analysis and Design*, 4th ed. Boston, MA: Thomson Engineering, 2008.

[93] F. P. de Mello and L. H. Hannett, "Validation of synchronous machine models and derivation of model parameters from tests," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 2, pp. 662-672, 1981.

[94] "Computer representation of excitation systems," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-87, no. 6, pp. 1460-1464, 1968.

AUTHOR'S BIOGRAPHY

Joseph Euzebe Tate was born in Opelousas, Louisiana, on December 16, 1980. He graduated from Louisiana Tech University in May 2003 with a Bachelor of Science in Electrical Engineering with a minor in Mathematics. He completed a Master of Science in Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign (UIUC) in May 2005. Following the completion of his Ph.D., also at UIUC, Tate will begin work as an assistant professor at the University of Toronto in the Department of Electrical and Computer Engineering.